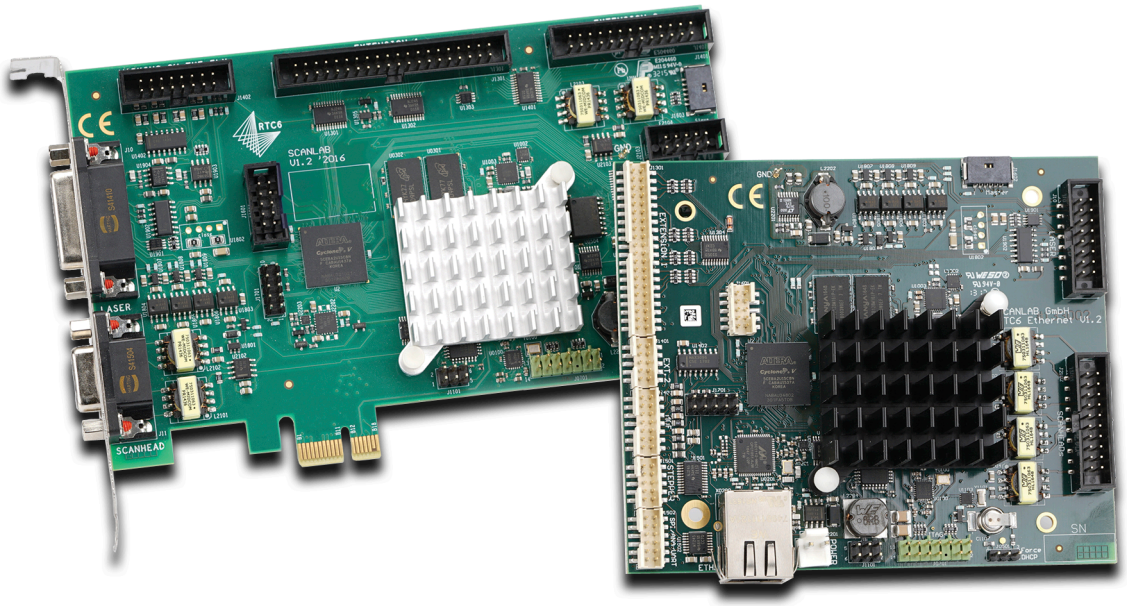
SCANLAB
innovators for industry

Installation und Inbetriebnahme

RTC6 PCIe-Karte
RTC6 Ethernet-Karte

Echtzeitsteuerung von Scan-Systemen und Lasern
RTC6-Software-Paket 1.19.1



RTC6-Handbuch
Doc. Rev. 1.1.2 de-DE

2024-10-28

SCANLAB GmbH
Siemensstr. 2a
82178 Puchheim
Deutschland

Tel. +49 (89) 800 746-0
Fax +49 (89) 800 746-199

info@scanlab.de
www.scanlab.de

© SCANLAB GmbH

SCANLAB GmbH behält sich vor, dieses Dokument jederzeit und ohne Ankündigung inhaltlich zu aktualisieren.
Kein Teil dieses Dokuments darf in irgendeiner Form (Fotokopie, Druck, Mikrofilm oder in einem anderen Verfahren) ohne ausdrückliche schriftliche Genehmigung der SCANLAB GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.
Alle erwähnte Marken unterliegen dem Markenschutz der jeweiligen Markeninhaber.

Inhalt

1 Über dieses Handbuch 25
1.1 Hersteller 25
1.2 Weiterführende Dokumente 26
1.3 Glossar und Abkürzungen 27
2 Produktübersicht 32
2.1 Kennzeichnung 32
2.2 Auspackanleitung und typischer Lieferumfang 32
2.3 Mitgeliefertes RTC6-Software-Paket 32
Verzeichnis Correction Files 33
Verzeichnis CorrectionFileConverter 33
Verzeichnis Demo Files 33
Verzeichnis HPGL 33
Verzeichnis iSCANCfg 33
Verzeichnis RTC6 Driver 33
Verzeichnis RTC6 Files 34
Verzeichnis RTC6 Tools 35
2.4 Bestimmungsgemäße Verwendung 36
2.5 Systemanforderungen 38
2.5.1 Hardware 38
2.5.2 Software 38
2.6 Optionen 39
2.7 Jumperkonfigurationen und Typenbezeichnungen 41
2.7.1 Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste 42
2.7.2 Lötjumperfeld B – Signal an EXTENSION 2-Stiftleiste Pin (17) 43
2.7.3 Lötjumperfeld C – Signal an EXTENSION 2-Stiftleiste Pin (15) 44
2.8 Zubehör für die RTC6 PCIe-Karte 45
2.8.1 XY2-100-Konverter 45
2.8.2 Laser-Adapter 45
2.8.3 Datenkabel 45
2.8.4 Slotblech mit 9-poliger Sub-D-Buchse für die 2. SCANHEAD-Stiftleiste 46
2.8.5 Slotblech mit 15-poliger Sub-D-Buchse für die MARKING ON THE FLY-Stiftleiste 46
2.8.6 Slotblech mit 15-poliger Sub-D-Buchse und 9-poliger Sub-D-Buchse 47
2.8.7 Erweiterungskarte “RTC5/6 varioSCAN FLEX Extension” 47
2.9 Ergänzende Software 48
2.10 Hinweise an RTC4-Benutzer 49
2.10.1 Hardware-Änderungen 49
Scan-Systeme steuern 49
Laser steuern 49
EXTENSION 1-Stiftleiste 49
EXTENSION 2-Stiftleiste 50
MARKING ON THE FLY-Stiftleiste 50
Sonstige Hardware-Schnittstellen 50
2.10.2 RTC4-Quellcodes für die RTC6 PCIe-Karte überarbeiten 50
Veränderte Initialisierung 50
Befehlsänderungen 51
Erhöhte Parameter-Auflösung 52
Verändertes Zeitverhalten 53
2.10.3 Neue und geänderte Funktionalitäten 54
Schnittstelle zum PC 54

Scan-Systeme steuern ... 54
Laser steuern ... 55
Schnittstellen zur Peripherie ... 56
Allgemeine Programmierung ... 57
Laser-Markieren ... 57
Spezielle Funktionen ... 58
2.11 Hinweise an RTC5-Benutzer ... 60
2.11.1 RTC6 PCIe-Karte und RTC5-Karten im Vergleich ... 60
Funktionalitätsänderungen ... 60
Parameteränderungen ... 61
2.11.2 RTC6 PCIe-Karten und *SCAN*ahead-Systeme ... 61
2.11.3 Einschränkungen bei RTC6 PCIe-Karten ... 62
Verfügbarkeit technischer Varianten ... 62
Schnittstellen zur Peripherie ... 62
Funktionalität ... 62
2.12 Quellcode für RTC6-Anwenderprogramme ... 62
2.12.1 RTC6-Quellcode neu erstellen ... 62
2.12.2 RTC5-Quellcode für die RTC6 PCIe-Karte anpassen ... 62
Zwingend erforderliche Schritte ... 62
Optionale Schritte und Hinweise zu weiteren Anpassungen und Optimierungen ... 63
2.13 Änderungen am RTC6-Befehlssatz ... 64
2.13.1 Geändert im RTC6-Befehlssatz ... 64
2.13.2 Nicht mehr vorhanden im RTC6-Befehlssatz ... 64

3 Sicherheit bei Installation und Inbetriebnahme ... 65
3.1 Betriebssicherheit ... 65
3.2 Lasersicherheit ... 65

4 RTC6 PCIe-Karte – Layout und Schnittstellen ... 66
4.1 Layout – Oberseite ... 66
4.2 Layout – Unterseite ... 67
4.3 Schnittstelle zum PC ... 68
4.4 Master-Stiftleiste, Slave-Stiftleiste ... 68
4.5 Schnittstellen zum Scan-System ... 69
4.5.1 Scan-Kopf-Anschlüsse und Übertragungsprotokoll ... 69
SCANHEAD-Buchse ... 70
2. SCANHEAD-Stiftleiste ... 70
4.5.2 XY2-100-Konverter (Zubehör) ... 71
4.5.3 Datenkabel (Zubehör) ... 73
4.6 Schnittstellen zum Laser und zur Peripherie ... 75
4.6.1 LASER-Buchse ... 75
Lasersteuersignale ... 76
Externe Steuersignale ... 76
BUSY-Listenausführungsstatus ... 76
2-Bit-Digital-Eingang ... 77
2-Bit-Digital-Ausgang ... 77
12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2 ... 77
Ein-/Ausgangsbeschaltung ... 77
Laser-Adapter (Zubehör) ... 79
4.6.2 EXTENSION 1-Stiftleiste ... 81
Ausgangssignal-Pegel einstellen ... 81
16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang ... 81
Synchronisation der Datenübergabe ... 82

BUSY-Listenausführungsstatus ... 82
4.6.3 EXTENSION 2-Stiftleiste ... 83
Einstellungen mittels Lötjumper ... 83
Lasersteuersignale ... 84
8-Bit-Digital-Ausgang ... 84
4.6.4 MARKING ON THE FLY-Stiftleiste ... 85
Encodereingänge ... 85
Externe Steuersignale ... 85
Analog-Ausgang ... 85
BUSY-Listenausführungsstatus ... 85
MARKING ON THE FLY-Slotblech (Zubehör) ... 86
4.6.5 RS232-Stiftleiste ... 86
4.6.6 McBSP/ANALOG-Stiftleiste ... 87
McBSP-Schnittstelle ... 87
Analog-Eingänge ... 89
4.6.7 STEPPER MOTOR-Stiftleiste ... 90

5 Installation und Inbetriebnahme ... 91
5.1 Jumpereinstellungen prüfen ... 91
5.2 RTC6 PCIe-Karte einbauen ... 91
5.3 RTC6-Kartentreiber installieren ... 92
5.4 RTC6-Software installieren ... 93
5.5 RTC6 PCIe-Karten-BIOS upgraden und downgraden ... 94
5.6 Sichere Einschalt- und Ausschaltreihenfolge ... 94
5.7 Funktionstest ... 95
5.8 Anwenderprogramme und Demoprogramme ... 95

6 RTC6-Anwenderprogramme entwickeln ... 96
6.1 Grundlegendes zum RTC6 Software-Konzept ... 96
6.1.1 Scan-Systeme und Laser ansteuern – Beispiel zur Einführung ... 96
6.1.2 Kontrollbefehle und Listenbefehle ... 97
6.2 Initialisierung und Programmstart ... 98
6.2.1 DLL-Aufrufkonvention ... 98
6.2.2 Befehle importieren ... 98
Pascal ... 99
C ... 99
C++ ... 99
C# ... 99
6.2.3 Initialisierung der RTC6-DLL und RTC6-Kartenverwaltung ... 100
6.2.4 Inbetriebnahme der RTC6 PCIe-Karte ... 101
Karte initialisieren ... 101
Karte konfigurieren ... 101
Scan-System-Ansteuerung initialisieren ... 102
Laser-Steuerung initialisieren ... 102
Listen laden und abarbeiten ... 102
6.2.5 Beispiel-Code (C) ... 103
6.3 RTC6-Listenspeicher ... 105
6.3.1 Listen und geschützter RTC6-Listenspeicher-Bereich ... 105
“Liste 1” und “Liste 2” ... 105
“Liste 3” – Geschützter RTC6-Listenspeicher-Bereich ... 105
6.3.2 RTC6-Listenspeicher konfigurieren ... 106
6.4 Listen-Handling ... 108
6.4.1 Listen laden ... 108

"Unbedingtes" Laden ... 108
Mit Schutzfunktion laden ... 109
Listen abschließen ... 109
6.4.2 Listenstatus ... 110
6.4.3 Listenausführungsstatus ... 111
6.4.4 Listen ausführen und beenden ... 112
6.4.5 Listen zur Synchronisation der Ausgabe unterbrechen ... 113
6.4.6 Listen automatisch wechseln ... 113
Einmaliger Listenwechsel ... 113
Alternierender Listenwechsel ... 114
6.5 Strukturierte Programmierung ... 115
6.5.1 Unterprogramme ... 115
Nichtindizierte Unterprogramme ... 115
Indizierte Unterprogramme ... 116
Allgemeines zum Aufruf von Unterprogrammen ... 117
Indexverwaltung und Defragmentierung ... 118
Nachträglich nichtindizierte Unterprogramme schützen und konvertieren ... 119
Unterprogramme entschützen ... 120
6.5.2 Zeichensätze und Zeichenfolgen ... 121
Indizierte Zeichensätze definieren ... 121
Indizierte Zeichen aufrufen ... 122
Indizierte Zeichenfolgen für Uhrzeit, Datum und Seriennummer definieren ... 122
Indizierte Zeichenfolgen aufrufen ... 123
Indizierte Zeichen und Zeichenfolgen verwalten ... 123
6.5.3 Sprünge ... 123
6.5.4 RTC4-Ringpuffer-Modus ... 124
6.5.5 Schleifen ... 125
6.6 Mehrere RTC6 PCIe-Karten in einem PC verwenden ... 126
6.6.1 Multi-Board-Programmierung ... 126
Beispiele (Pascal) ... 126
6.6.2 Single-Board-Programmierung ... 127
6.6.3 Master/Slave-Betrieb ... 127
Initialisierung mit RTC6-Software-Paket ≥ V1.5.0 (≥ RBF 619) ... 128
Initialisierung mit RTC6-Software-Paket < V1.5.0 (≤ RBF 618) ... 128
Synchrone Starts und synchrone Stopps ... 129
Beispiel-Code (Delphi) ... 130
6.7 RTC6 PCIe-Karten durch mehrere Anwenderprogramme verwenden ... 131
6.7.1 Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm ... 131
6.8 Fehlerbehandlung (Error-Handling) ... 133
6.8.1 Download-Überprüfung ... 134
6.8.2 Überprüfung auf Überläufe ... 135
6.8.3 Beispiel-Code (C) ... 136
6.9 Verschiedenes ... 138
6.9.1 Freie Variablen ... 138
7 Grundlegende Funktionen zur Scan-Kopf-Steuerung und Laser-Steuerung ... 139
7.1 Punkte, Linien und Bögen markieren ... 139
7.1.1 Mit Vektor-Befehlen und Bogen-Befehlen markieren ... 139
Sprung-Befehle ... 140
Mark-Befehle ... 140
Arc-Befehle ... 140
Polygonzüge ... 140
Ellipse-Befehle ... 141

[*]Para[*]-Befehle ... 142
7.1.2 Zerlegung in Mikroschritte ... 143
7.1.3 Einzelne Punkte markieren ... 144
7.1.4 Beispiel-Code (C) ... 145
7.2 Delayeinstellungen – Scan-Kopf-Steuerung und Laser-Steuerung abstimmen ... 148
7.2.1 Laser-Delays ... 148
LaserOn-Delay ... 149
LaserOff-Delay ... 149
7.2.2 Scanner-Delays ... 150
Jump-Delay ... 150
Variable Jump-Delays ... 151
Mark-Delay ... 152
Polygon-Delay ... 153
Variable Polygon-Delays ... 154
Benutzer-definierte “Variable Polygon-Delays” ... 156
7.2.3 Hinweise zur Optimierung der Delays ... 158
Empfohlene Optimierungsreihenfolge ... 158
Automatische Delay-Anpassung ... 158
Mögliche Fehler beim Optimieren der Delays ... 160
7.2.4 Sky-Writing ... 163
Sky-Writing-Modus 1 ... 163
Sky-Writing-Modus 2 ... 164
Sky-Writing-Modus 3 ... 165
Sky-Writing-Modus 4 ... 166
Synchronisation ... 167
Sky-Writing mit Mindest-Markiergeschwindigkeit ... 171
Sky-Writing mit Skalierung der Ausholbewegung ... 173
7.3 Scan-Kopf-Steuerung ... 175
7.3.1 Referenzsystem ... 175
7.3.2 Bildfeld-Größe und Bildfeld-Kalibrierung ... 175
Typisches Bildfeld ... 176
Kompatibilitätsmodi ... 176
7.3.3 Virtuelles Bildfeld ... 177
Koordinatentransformationen im virtuellen Bildfeld ... 177
7.3.4 3D-Bildfeld ... 178
Justierung vornehmen ... 178
z-Achsen-Kalibrierung überprüfen ... 178
3-Achsen-Scan-Systeme mit F-Theta-Objektiv testen ... 180
7.3.5 Bildfeld-Korrektur und Korrekturtabellen ... 181
Bildfeld-Verzeichnung ... 181
Bildfeld-Korrektur-Algorithmus ... 182
Bildfeld-Korrektur aktivieren ... 182
2D- und 3D-Korrekturdateien ... 182
Korrekturtabellen laden ... 183
Geladene Korrekturtabellen zuweisen ... 183
1to1-Korrekturtabellen ... 185
Inverse Tabellen ... 185
ct5-Korrekturdateikopf ... 186
Konvertierung von Korrekturdateien ... 188
7.3.6 Ausgabewerte an das Scan-System ... 189
Berechnung ... 189
Wertebereiche ... 190

Vorausberechnung und Diagnose ... 190
Takt-Überläufe ... 190
7.3.7 Statusüberwachung und Diagnose ... 191
Statusinformationen vom Scan-System ... 191
7.4 Laser-Steuerung ... 192
7.4.1 Lasersteuersignale freigeben, aktivieren und schalten ... 192
Signale für den “Laser active”-Betrieb ... 194
Signale für den “Laser standby”-Betrieb ... 194
Automatische Unterdrückung der Lasersteuersignale ... 195
7.4.2 LASER-Buchse konfigurieren ... 195
7.4.3 CO_2-Modus ... 196
7.4.4 YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5 ... 198
Q-Switch-Signal ... 200
FirstPulseKiller-Signal ... 200
YAG-Modus-Unterschiede ... 200
Lampenstrom (Laserleistung) ... 200
7.4.5 Laser-Mode 4 ... 201
FirstPulseKiller-Signal ... 201
7.4.6 Laser-Mode 6 ... 203
7.4.7 Softstart-Modus (noch nicht implementiert) ... 205
7.4.8 Pulse-Picking-Laser-Modus ... 206
7.4.9 “Automatische Laser-Steuerung“ ... 208
Positions-abhängige Laser-Steuerung ... 210
Geschwindigkeits-abhängige Laser-Steuerung ... 212
Encodergeschwindigkeits-abhängige Laser-Steuerung ... 215
Nichtlinearitätskurve laden und ermitteln ... 215
Vektorkontrollierte Laser-Steuerung ... 217
2D-Positions-abhängige Laser-Steuerung ... 218
7.4.10 Synchronisierung von RTC6-Takt und externem Taktsignal ... 219
7.4.11 Puls-Synchronisations-Modus ... 220
7.5 Uhrzeit, Datum und Seriennummern markieren ... 222
7.5.1 Uhrzeit und Datum markieren ... 222
7.5.2 Seriennummern markieren ... 222

8 Erweiterte Funktionen zur Scan-Kopf-Steuerung und Laser-Steuerung ... 224

8.1 i*DRIVE*-Funktionen ... 224
8.1.1 Übertragungsprotokoll ... 224
8.1.2 Datensignal-Übertragungsverhalten des Scan-Systems anpassen ... 225
Datentypen einstellen ... 225
Daten auslesen ... 225
8.1.3 Positionierung überwachen ... 226
8.1.4 Tuning (Dynamikeinstellung) auswählen ... 228
8.1.5 Sprungmodus ... 229
Funktionsweise ... 229
Voraussetzung und Aktivierung ... 230
Sprunglängen-abhängige Jump-Delays ... 230
Jump-Delay-Werte experimentell ermitteln ... 231
Hinweise zum Laden der ermittelten Jump-Delay-Werte ... 231
Jump-Delay-Tabelle automatisch ermitteln ... 233
8.1.6 PosAck-Grenzwert anpassen ... 234
8.1.7 Effektive Kalibrierung anpassen ... 234
8.1.8 Startverhalten anpassen ... 235
8.1.9 Fehlerdiagnose und Funktionstest ... 235

8.2 Koordinatentransformationen ... 236
8.3 Online-Positionierung ... 240
8.3.1 "Lokale Online-Positionierung" ... 240
"Lokale Online-Positionierung" konfigurieren ... 241
8.3.2 "Globale Online-Positionierung" ... 243
"Globale Online-Positionierung" konfigurieren ... 243
8.4 Wobbelmode ... 244
Beispiel-Code (C++) ... 245
8.4.1 Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte ... 246
"Klassische" Wobbelfiguren ... 246
"Frei definierbare Wobbelfiguren" ... 247
8.5 2D-Scan-Systeme und 3D Scan-Systeme steuern ... 248
8.5.1 2D-Scan-Systeme ... 248
8.5.2 3D-Scan-Systeme ... 249
Vorgesehene Verwendung ... 249
Anschluss und Initialisierung ... 249
3D-Befehle ... 250
Schleppverzüge anpassen ... 251
Erweiterte 3D-Korrektur ... 252
8.5.3 Mehrere Korrekturtabellen verwenden ... 253
8.6 "Processing on the fly" ... 254
8.6.1 Verwendung und Initialisierung ... 254
Übersicht ... 254
8.6.2 Lineare Bewegungen ausgleichen ... 255
Korrektur über Encoderzähler ... 255
Korrektur über McBSP-Schnittstelle ... 257
Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe ... 258
Korrektur über McBSP-Schnittstelle mit erweiterter McBSP-Eingabe ... 258
8.6.3 Rotationsbewegungen ausgleichen ... 259
Korrektur über Encoderzähler ... 259
Korrektur über McBSP-Schnittstelle ... 260
Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe ... 260
8.6.4 2D-Bewegungen ausgleichen ... 261
2D-Encoderkompensation für xy-Verfahrtische ... 261
8.6.5 "Processing on the fly"-Korrektur deaktivieren ... 263
8.6.6 Virtuelles Bildfeld mit "Processing on the fly" ... 264
8.6.7 "Processing on the fly"-Anwendungen synchronisieren ... 264
8.6.8 Encoderresets ... 266
8.6.9 "Processing on the fly"-Korrektur überwachen ... 267
Kundenspezifischer Überwachungsbereich ... 268
8.6.10 Schleppverzugskompensation von Encoderwerten für "Processing on the fly"-Anwendungen ... 269
8.6.11 "Processing on the fly"-Korrektur für die z-Achse ... 269
8.6.12 "Fly-Erweiterung"-Befehle ... 271
Hinweise zur Verwendung der "Fly-Erweiterung"-Befehle ... 272
8.7 Pixelausgabemodus ... 275
8.7.1 Funktionsprinzip ... 275
8.7.2 RTC6-Befehle ... 275
8.7.3 Laser-Steuerung ... 277
8.7.4 Synchronisation ... 280
8.8 **micro_vector[*]**-Befehle ... 285
8.9 Zeitbemessene Befehle ... 286

8.10 Automatische Selbstkalibrierung ... 287
8.10.1 Verwendung zur Drift-Kompensation ... 287
8.10.2 Funktionsweise ... 287
8.10.3 Referenzwerte bestimmen ... 289
8.10.4 Kalibrierung während der Anwendung ... 289
Automatische Selbstkalibrierung ... 289
Kundeneigene Kalibrierung ... 290
Weitere Hinweise zur Kalibrierung ... 290
8.11 Camming ... 291
8.12 Zeitmessungen ... 293
8.12.1 RTC6-Timer ... 293
8.12.2 Zeitstempel ... 293
8.13 *SCAN*ahead-Funktionalität ... 294
8.13.1 RTC6–*SCAN*ahead-Funktionen ... 294
8.13.2 Schnelleinstieg für erfahrene Benutzer ... 298
Beispiel-Code (C++) ... 298
Bestehende Anwenderprogramme für *SCAN*ahead-Systeme anpassen – Grundlegende Schritte ... 300
Bestehende Anwenderprogramme für *SCAN*ahead-Systeme anpassen – Einschränkungen berücksichtigen ... 301
Anwenderprogramme anpassen – Weitere Schritte, Hintergrundinformationen zu *SCAN*ahead-System-bezogenen Funktionalitäten ... 301

9 Peripherie-Schnittstellen programmieren ... 306
9.1 Signal-Ausgabe ... 306
9.1.1 16-Bit-Digital-Ausgang ... 306
9.1.2 8-Bit-Digital-Ausgang ... 307
9.1.3 2-Bit-Digital-Ausgang ... 307
9.1.4 12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2 ... 308
9.1.5 Schrittmotoren ansteuern ... 309
Ausgangssignale ... 309
Referenzfahrten und Positions-Initialisierung ... 310
Zielpositionsfahrten ... 310
Signale und Statuswerte abfragen ... 311
Endlosfahrten anhalten ... 311
Parameter WaitTime ... 311
9.1.6 RS-232-Schnittstelle ... 312
9.1.7 McBSP-Schnittstelle ... 312
9.2 Signal-Eingabe ... 313
9.2.1 16-Bit-Digital-Eingang ... 313
9.2.2 2-Bit-Digital-Eingang ... 313
9.2.3 RS-232-Schnittstelle ... 313
9.2.4 McBSP-Schnittstelle ... 313
9.3 Steuerung mit externen Signalen ... 314
9.3.1 Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden ... 314
Externer Stopp ... 314
Externer Start ... 315
Externer Start mit Streckenverzögerung ... 317
Regelmäßige (periodische) externe Starts ... 318
9.3.2 Ausführung Bedingter Befehle ... 320
Beispiel-Code (Pascal) ... 321
9.3.3 Synchronisation durch Encodersignale ... 323

Vorgesehene Verwendung ... 323
Eingänge für externe Encodersignale ... 324
Encodersimulation ... 324
9.3.4 Synchronisation und Online-Positionierung mit McBSP-Signalen ... 325
9.4 Periodische I/O-Signale ... 326
10 RTC6-Befehle ... 327
10.1 Übersicht ... 327
10.1.1 Kontrollbefehle ... 328
10.1.2 Listenbefehle ... 334
Normaler Listenbefehl, Kurzer Listenbefehl, Variabler Listenbefehl und Multipler Listenbefehl ... 339
10.2 RTC6-Befehlssatz ... 342
10.2.1 Allgemeiner Aufbau der Befehlsbeschreibungen ... 343
10.2.2 Datentypen ... 345
10.2.3 Zeiger auf Arbeitsspeicherbereiche ... 345
10.2.4 Befehlsbeschreibungen ... 346
acquire_rtc ... 346
activate_fly_1_axis ... 348
activate_fly_2_axes ... 349
activate_fly_2d ... 351
activate_fly_2d_encoder ... 352
activate_fly_xy ... 354
activate_fly_xy_encoder ... 354
activate_scanahead_autodelays ... 355
activate_scanahead_autodelays_list ... 356
apply_mcbsp ... 357
apply_mcbsp_list ... 358
arc_abs ... 359
arc_abs_3d ... 360
arc_rel ... 361
arc_rel_3d ... 362
auto_cal ... 363
auto_change ... 366
auto_change_pos ... 367
bounce_supp ... 368
camming ... 369
clear_fly_overflow ... 371
clear_fly_overflow_ctrl ... 371
clear_io_cond_list ... 372
config_laser_signals ... 373
config_laser_signals_list ... 374
config_list ... 375
control_command ... 377
copy_dst_src ... 379
create_dat_file ... 381
demux_rcm_ctrl ... 383
disable_laser ... 385
enable_laser ... 386
eth_assign_card ... 387
eth_assign_card_ip ... 388
eth_boot_dcmd ... 389
eth_boot_timeout ... 389

eth_check_connection ... 391
eth_config_waveform_streaming_ctrl ... 392
eth_configure_link_loss ... 393
eth_convert_ip_to_string ... 394
eth_convert_string_to_ip ... 395
eth_count_cards ... 396
eth_found_cards ... 397
eth_full_reset ... 398
eth_get_card_info ... 399
eth_get_card_info_search ... 400
eth_get_com_timeouts ... 401
eth_get_com_timeouts_auto ... 402
eth_get_error ... 403
eth_get_ip ... 405
eth_get_ip_search ... 405
eth_get_last_error ... 406
eth_get_port_numbers ... 408
eth_get_serial_search ... 409
eth_get_standalone_status ... 410
eth_get_static_ip ... 411
eth_max_card ... 412
eth_remove_card ... 413
eth_search_cards ... 414
eth_search_cards_range ... 415
eth_set_com_timeouts ... 416
eth_set_com_timeouts_auto ... 417
eth_set_high_performance_mode ... 418
eth_set_port_numbers ... 419
eth_set_remote_tgm_format ... 420
eth_set_search_cards_timeout ... 421
eth_set_static_ip ... 422
execute_at_pointer ... 423
execute_list ... 424
execute_list_1 ... 424
execute_list_2 ... 424
execute_list_pos ... 425
fly_disable_list ... 426
fly_prediction ... 427
fly_return ... 428
fly_return_1_axis ... 429
fly_return_2_axes ... 430
fly_return_3_axes ... 431
fly_return_z ... 432
free_rtc6_dll ... 433
get_auto_cal ... 434
get_bios_version ... 435
get_card_type ... 435
get_char_pointer ... 436
get_config_list ... 437
get_counts ... 437
get_dll_version ... 438
get_encoder ... 438

get_error 439
get_extended_head_status_block_ctrl 443
get_extended_head_status_value_ctrl 444
get_fly_2d_offset 445
get_free_variable 445
get_galvo_controls 446
get_head_para 448
get_head_status 449
get_hex_version 451
get_hi_data 452
get_hi_pos 453
get_input_pointer 454
get_io_status 454
get_jump_table 455
get_lap_time 455
get_laser_pin_in 456
get_last_error 456
get_list_pointer 457
get_list_serial 458
get_list_space 458
get_marking_info 459
get_master_slave 462
get_mcbsp 463
get_mcbsp_list 463
get_out_pointer 464
get_overrun 464
get_rcm_value_ctrl 465
get_rtc_mode 466
get_rtc_version 467
get_scanahead_params 468
get_serial 470
get_serial_number 471
get_standby 472
get_startstop_info 473
get_status 475
get_stepper_status 477
get_sub_pointer 478
get_sync_status 479
get_table_para 481
get_temperature 482
get_text_table_pointer 483
get_time 484
get_timestamp_long 485
get_transform 486
get_transform_offset 487
get_value 492
get_values 494
get_wait_status 495
get_waveform 496
get_waveform_offset 497
get_z_distance 498
goto_xy 500

goto_xyz 501
home_position 502
home_position_xyz 503
if_cond 504
if_fly_x_overflow 505
if_fly_y_overflow 506
if_fly_z_overflow 507
if_not_activated 508
if_not_cond 509
if_not_fly_x_overflow 510
if_not_fly_y_overflow 511
if_not_fly_z_overflow 512
if_not_pin_cond 513
if_pin_cond 514
init_fly_2d 515
init_fly_2d_list 516
init_rtc6_dll 517
jump_abs 519
jump_abs_3d 520
jump_abs_drill 521
jump_abs_drill_2 521
jump_rel 522
jump_rel_3d 523
jump_rel_drill 524
jump_rel_drill_2 524
laser_on_list 525
laser_on_pulses_list 526
laser_signal_off 528
laser_signal_off_list 528
laser_signal_on 529
laser_signal_on_list 529
list_call 530
list_call_abs 532
list_call_abs_cond 533
list_call_abs_repeat 534
list_call_cond 535
list_call_repeat 536
list_continue 537
list_jump_cond 538
list_jump_pos 539
list_jump_pos_cond 540
list_jump_rel 541
list_jump_rel_cond 542
list_next 543
list_nop 543
list_repeat 544
list_return 545
list_until 546
load_auto_laser_control 547
load_char 549
load_correction_file 550
load_disk 554

load_fly_2d_table 557
load_jump_table 558
load_jump_table_offset 559
load_list 561
load_position_control 563
load_position_control_2d_ctrl 565
load_program_file 566
load_stretch_table 569
load_sub 571
load_text_table 572
load_varpolydelay 573
load_z_table 575
load_z_table_20b 576
load_z_table_no 577
load_z_table_no_20b 578
load_zoom_correction_file 579
long_delay 580
mark_abs 581
mark_abs_3d 582
mark_char 583
mark_char_abs 584
mark_date 585
mark_date_abs 587
mark_ellipse_abs 588
mark_ellipse_rel 589
mark_rel 590
mark_rel_3d 591
mark_serial 592
mark_serial_abs 594
mark_text 595
mark_text_abs 596
mark_time 597
mark_time_abs 599
master_slave_config 600
master_slave_set_offset_ctrl 601
mcbsp_init 602
mcbsp_init_spi 603
measurement_status 604
micro_vector_abs 605
micro_vector_abs_3d 607
micro_vector_rel 608
micro_vector_rel_3d 609
move_to 610
number_of_correction_tables 612
para_jump_abs 613
para_jump_abs_3d 614
para_jump_rel 615
para_jump_rel_3d 616
para_laser_on_pulses_list 617
para_mark_abs 619
para_mark_abs_3d 621
para_mark_rel 622

para_mark_rel_3d 623
park_position 624
park_position_1_axis 626
park_position_2_axes 627
park_return 628
park_return_1_axis 630
park_return_2_axes 631
pause_list 632
periodic_toggle 633
periodic_toggle_list 634
quit_loop 635
range_checking 636
read_abc_from_file 638
read_abc_from_file_20b 639
read_analog_in 640
read_encoder 641
read_image_eth 642
read_io_port 643
read_io_port_buffer 644
read_io_port_list 645
read_mcbsp 646
read_multi_mcbsp 647
read_status 648
read_user_data 650
regulation3 651
release_rtc 652
release_wait 653
reset_error 654
restart_list 655
rs232_config 655
rs232_read_data 656
rs232_write_data 657
rs232_write_text 657
rs232_write_text_list 658
rtc6_count_cards 659
save_and_restart_timer 660
save_disk 661
select_char_set 663
select_cor_table 664
select_cor_table_list 667
select_rtc 668
select_serial_set 670
select_serial_set_list 670
send_user_data 671
set_angle 672
set_angle_list 673
set_auto_laser_control 674
set_auto_laser_params 679
set_auto_laser_params_list 680
set_char_pointer 681
set_char_table 682
set_control_mode 683

set_control_mode_list 686
set_controlpreview_compensation_ctrl 687
set_default_pixel 688
set_default_pixel_list 688
set_defocus 689
set_defocus_list 690
set_defocus_2_list 691
set_defocus_offset 692
set_defocus_offset_list 693
set_delay_mode 694
set_delay_mode_list 696
set_dsp_mode 697
set_duty_cycle_table 698
set_ellipse 699
set_encoder_filter_ctrl 700
set_encoder_speed 702
set_encoder_speed_ctrl 703
set_end_of_list 704
set_eth_boot_control 705
set_extstartpos 706
set_extstartpos_list 706
set_ext_start_delay 707
set_ext_start_delay_list 708
set_firstpulse_killer 709
set_firstpulse_killer_list 709
set_fly_1_axis 710
set_fly_2_axes 711
set_fly_2d 713
set_fly_3_axes 715
set_fly_limits 717
set_fly_limits_z 718
set_fly_rot 719
set_fly_rot_pos 720
set_fly_tracking_error 721
set_fly_x 723
set_fly_x_pos 724
set_fly_y 726
set_fly_y_pos 727
set_fly_z 728
set_free_variable 729
set_free_variable_list 729
set_hi 730
set_input_pointer 731
set_io_cond_list 732
set_jump_mode 733
set_jump_mode_list 736
set_jump_speed 737
set_jump_speed_ctrl 738
set_jump_table 739
set_laser_control 740
set_laser_delays 743
set_laser_delays_ctrl 744

set_laser_mode ... 745
set_laser_off_default ... 746
set_laser_pin_out ... 747
set_laser_pin_out_list ... 747
set_laser_power ... 748
set_laser_pulse_sync ... 749
set_laser_pulses ... 750
set_laser_pulses_ctrl ... 751
set_laser_timing ... 752
set_laser_timing_table ... 753
set_list_jump ... 754
set_mark_speed ... 755
set_mark_speed_ctrl ... 756
set_matrix ... 757
set_matrix_list ... 759
set_max_counts ... 760
set_mcbsp_freq ... 761
set_mcbsp_global_matrix ... 762
set_mcbsp_global_matrix_list ... 763
set_mcbsp_global_rot ... 764
set_mcbsp_global_rot_list ... 764
set_mcbsp_global_x ... 765
set_mcbsp_global_x_list ... 765
set_mcbsp_global_y ... 766
set_mcbsp_global_y_list ... 766
set_mcbsp_in ... 767
set_mcbsp_in_list ... 769
set_mcbsp_matrix ... 770
set_mcbsp_matrix_list ... 771
set_mcbsp_out ... 772
set_mcbsp_out_oie_ctrl ... 773
set_mcbsp_out_oie_list ... 774
set_mcbsp_out_ptr ... 775
set_mcbsp_out_ptr_list ... 777
set_mcbsp_rot ... 778
set_mcbsp_rot_list ... 778
set_mcbsp_x ... 779
set_mcbsp_x_list ... 779
set_mcbsp_y ... 780
set_mcbsp_y_list ... 780
set_multi_mcbsp_in ... 781
set_multi_mcbsp_in_list ... 783
set_n_pixel ... 784
set_offset ... 785
set_offset_list ... 785
set_offset_xyz ... 786
set_offset_xyz_list ... 788
set_pause_list_cond ... 789
set_pause_list_not_cond ... 790
set_pixel ... 791
set_pixel_line ... 792
set_pixel_line_3d ... 795

set_port_default ... 796
set_port_default_list ... 797
set_pulse_picking ... 798
set_pulse_picking_length ... 799
set_pulse_picking_list ... 799
set_qswitch_delay ... 800
set_qswitch_delay_list ... 800
set_rot_center ... 801
set_rot_center_list ... 801
set_rtc4_mode ... 802
set_rtc5_mode ... 803
set_rtc6_mode ... 804
set_scale ... 805
set_scale_list ... 805
set_scanahead_laser_shifts ... 806
set_scanahead_laser_shifts_list ... 807
set_scanahead_line_params ... 808
set_scanahead_line_params_list ... 809
set_scanahead_params ... 810
set_scanahead_speed_control ... 813
set_scanner_delays ... 814
set_scanner_delays_ctrl ... 815
set_serial ... 816
set_serial_step ... 817
set_serial_step_list ... 817
set_short_cmd_mode_ctrl ... 818
set_short_cmd_mode_list ... 818
set_sky_writing ... 819
set_sky_writing_limit ... 820
set_sky_writing_limit_list ... 820
set_sky_writing_list ... 821
set_sky_writing_min_speed_ctrl ... 822
set_sky_writing_min_speed_list ... 823
set_sky_writing_mode ... 824
set_sky_writing_mode_list ... 825
set_sky_writing_para ... 826
set_sky_writing_para_list ... 828
set_sky_writing_scale_ctrl ... 829
set_sky_writing_scale_list ... 830
set_softstart_level ... 831
set_softstart_level_list ... 831
set_softstart_mode ... 832
set_softstart_mode_list ... 832
set_standby ... 833
set_standby_list ... 834
set_start_list ... 835
set_start_list_1 ... 835
set_start_list_2 ... 835
set_start_list_pos ... 836
set_sub_pointer ... 838
set_text_table_pointer ... 839
set_timelag_compensation ... 841

set_trigger 842
set_trigger4 852
set_trigger8 853
set_vector_control 854
set_verify 856
set_wait 857
set_wobbel 858
set_wobbel_control 860
set_wobbel_direction 862
set_wobbel_mode 863
set_wobbel_mode_phase 866
set_wobbel_offset 867
set_wobbel_vector 868
set_wobbel_vector_2 872
set_zoom 872
set_zoom_list 872
simulate_encoder 873
simulate_ext_start 874
simulate_ext_start_ctrl 875
simulate_ext_stop 876
spot_distance 877
spot_distance_ctrl 877
start_loop 878
stepper_abs 879
stepper_abs_list 880
stepper_abs_no 880
stepper_abs_no_list 881
stepper_control 882
stepper_control_list 882
stepper_disable_switch 883
stepper_enable 884
stepper_enable_list 884
stepper_init 885
stepper_rel 887
stepper_rel_list 887
stepper_rel_no 888
stepper_rel_no_list 888
stepper_wait 889
stop_execution 890
stop_list 891
stop_trigger 891
store_encoder 892
store_program 893
store_timestamp_counter 894
store_timestamp_counter_list 894
sub_call 895
sub_call_abs 896
sub_call_abs_cond 897
sub_call_abs_repeat 898
sub_call_cond 899
sub_call_repeat 900
switch_ioport 901

sync_slaves ... 902
time_control_eth ... 904
time_fix ... 905
time_fix_f ... 905
time_fix_f_off ... 906
time_update ... 907
timed_arc_abs ... 908
timed_arc_rel ... 909
timed_jump_abs ... 910
timed_jump_abs_3d ... 911
timed_jump_rel ... 912
timed_jump_rel_3d ... 913
timed_mark_abs ... 914
timed_mark_abs_3d ... 915
timed_mark_rel ... 916
timed_mark_rel_3d ... 917
timed_para_jump_abs ... 918
timed_para_jump_abs_3d ... 919
timed_para_jump_rel ... 920
timed_para_jump_rel_3d ... 921
timed_para_mark_abs ... 922
timed_para_mark_abs_3d ... 923
timed_para_mark_rel ... 924
timed_para_mark_rel_3d ... 925
transform ... 926
uart_config ... 929
upgrade_bios ... 930
upload_transform ... 931
verify_checksum ... 933
wait_for_1_axis ... 934
wait_for_2_axes ... 936
wait_for_encoder ... 938
wait_for_encoder_in_range ... 939
wait_for_encoder_in_range_mode ... 940
wait_for_encoder_mode ... 941
wait_for_mcbsp ... 943
wait_for_timestamp_counter ... 944
wait_for_timestamp_counter_long ... 945
wait_for_timestamp_counter_mode ... 946
write_8bit_port ... 948
write_8bit_port_list ... 948
write_abc_to_file ... 949
write_abc_to_file_20b ... 950
write_da_1 ... 951
write_da_1_list ... 951
write_da_2 ... 952
write_da_2_list ... 952
write_da_x ... 953
write_da_x_list ... 954
write_hi_pos ... 955
write_image_eth ... 956
write_io_port ... 957

write_io_port_list ... 957
write_io_port_mask ... 958
write_io_port_mask_list ... 958
write_port_list ... 959
10.3 RTC6-Befehle in anderen Publikationen ... 961
10.4 Nicht mehr unterstützte RTC4-Befehle ... 963
10.5 Nicht mehr unterstützte RTC5-Befehle ... 966
11 Demo-Programme ... 967
12 Fehlersuche ... 968
13 Kundendienst ... 970
13.1 Wartung und Reparaturen ... 970
13.2 Gewährleistung ... 970
13.3 Kontaktmöglichkeiten zu SCANLAB ... 970
13.4 Entsorgung ... 970
14 Rechtliches ... 971
14.1 EU-Konformitätserklärung – RTC6 PCIe-Karte ... 971
14.2 Konformität mit FCC-Bestimmungen ... 972
15 Technische Spezifikationen – RTC6 PCIe-Karte ... 973
16 Anhang A: RTC6 Ethernet-Karte ... 978
16.1 Produktübersicht ... 978
16.1.1 RTC6 Ethernet-Karte und RTC6 PCIe-Karte – Verwendung und Vergleich ... 978
16.1.2 Systemanforderungen ... 979
Hardware ... 979
Software ... 979
16.1.3 Optionen ... 980
16.1.4 Kennzeichnung ... 980
16.1.5 Typenbezeichnung ... 980
16.1.6 Auspackanleitung und typischer Lieferumfang ... 980
16.1.7 Mitgelieferte Software ... 980
16.1.8 Zubehör für die RTC6 Ethernet-Karte ... 980
16.1.9 Ergänzende Software ... 980
16.2 Layout, Schnittstellen, Jumpereinstellungen ... 981
16.2.1 Layout – Oberseite ... 981
16.2.2 Layout – Unterseite ... 982
16.2.3 Abmessungen und Steckverbinderpositionen ... 983
16.2.4 LASER-Stiftleiste ... 984
Lasersteuersignale ... 984
Externe Steuersignale ... 984
BUSY-Listenausführungsstatus ... 984
2-Bit-Digital-Eingang, 2-Bit-Digital-Ausgang ... 984
12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2 ... 985
Ein-/Ausgangsbeschaltung ... 985
16.2.5 SCANHEADs-Stiftleiste ... 987
16.2.6 POWER-Stiftleiste ... 989
16.2.7 ETH-Buchse ... 989
16.2.8 SPI/ANA/UART-Stiftleiste ... 990
Analog-Eingänge ... 990
McBSP-Schnittstelle ... 990
RS-232 Schnittstelle ... 990
16.2.9 STEPPER-Stiftleiste ... 991

16.2.10MOF-Stiftleiste 992
Encodereingänge 992
Externe Steuersignale 992
BUSY OUT-Status 992
16.2.11EXTENSION 1-Stiftleiste 993
Ausgangssignal-Pegel einstellen 993
16-Bit-Digital-Ausgang und 16-Bit-Digital-Eingang 993
Synchronisation der Datenübergabe 993
BUSY-Listenausführungsstatus 993
16.2.12EXT. 2-Stiftleiste 994
Einstellungen mittels Lötjumper 995
8-Bit-Digital-Ausgang 995
16.2.13Master-Stiftleiste, Slave-Stiftleiste 996
16.2.14Jumpereinstellungen 997
Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste 997
Lötjumperfeld B – Signal an EXT. 2-Stiftleiste Pin (09) 998
Lötjumperfeld C – Signal an EXT. 2-Stiftleiste Pin (08) 999
Steckjumperfeld 'Force DHCP' 1000
16.2.15Echtzeit-Uhr 1000
16.3 Installation und Inbetriebnahme 1001
16.3.1 Hardware-Installation 1001
16.3.2 Software-Installation 1002
16.3.3 Mit einem Netzwerk verbinden 1002
16.4 Hinweise zur Migration und Neu-Programmierung von RTC6-Anwenderprogrammen 1003
16.4.1 RTC6 Ethernet-Karten im Netzwerk finden und Eigenschaften abfragen 1003
16.4.2 Beispiel-Code (C++): Initialisierung mit RTC6 Ethernet-Karten 1004
16.5 RTC6 Ethernet-Karten-Befehle und Funktionen 1007
16.5.1 Hinweise zum Arbeiten mit IP-Adressen 1007
16.5.2 Über die Suche nach RTC6 Ethernet-Karten 1007
16.5.3 Über die RTC6-Kartenverwaltung 1008
16.5.4 Verbindung zur RTC6 Ethernet-Karte prüfen 1009
16.5.5 Befehlssatz für die RTC6 Ethernet-Karte 1009
16.6 Sichere Einschalt- und Ausschaltreihenfolge 1010
16.7 BIOS-ETH-Version upgraden und downgraden 1011
16.8 Standalone-Funktionalität 1012
16.8.1 "Standalone-Basis-Zustand" herstellen 1013
16.8.2 "Standalone-Voll-Zustand" herstellen 1014
16.8.3 Boot-Image 1015
Boot-Image auf dem PC erstellen 1015
Boot-Image auf die Karte(n) kopieren 1015
Vorgehen nach einem vorzeitigen Abbruch der Übertragung 1015
16.8.4 Für das automatische Booten zulässige Kontrollbefehle 1016
16.8.5 Automatisches Booten – Ablauf im Detail 1018
16.9 "High-Performance-Modus" 1020
16.10Remote-Interface-Modus 1021
16.10.1Voraussetzungen für den Remote-Interface-Modus 1022
16.10.2Einschränkungen im Remote-Interface-Modus 1022
16.10.3Remote-Interface-Modus aktivieren 1023
16.10.4Remote-Interface-Modus deaktivieren – vorübergehend 1023
16.10.5Remote-Interface-Modus deaktivieren – dauerhaft 1024
16.10.6Telegramm-Format RAW 1025
Befehls-Telegramm – Struktur 1025

Antwort-Telegramm – Struktur ... 1025
TGM_HEADER ... 1026
TGM_PL_CMD_RAW ... 1027
TGM_PL_ANSW_RAW ... 1027
16.10.7 Fehlercodes für `LastError`, die nur im Remote-Interface-Modus vorkommen ... 1028
16.10.8 Befehls-Telegramme senden und Antwort-Telegramme empfangen ... 1030
16.10.9 Remote-Befehle ... 1030
Remote-Kontrollbefehle ... 1031
Remote-Listenbefehle ... 1042
16.10.10 Nicht als Remote-Befehl verfügbare RTC6-Befehle ... 1054
16.10.11 Hinweise zur Softwareentwicklung ... 1060
16.10.12 Über den C++-Remote-Interface-Wrapper ... 1061
16.11 Rechtliches ... 1063
16.11.1 EU-Konformitätserklärung – RTC6 Ethernet-Karte ... 1063
16.11.2 Konformität mit FCC-Bestimmungen ... 1064
16.11.3 TI-RTOS ... 1064
16.12 Technische Spezifikationen – RTC6 Ethernet-Karte ... 1065

17 Anhang B: UFPM-Erweiterungskarte ... 1070

18 Anhang C: RTC6 EtherBox ... 1073
18.1 Montage ... 1076
18.2 Verkabelung ... 1076
18.3 Inbetriebnahme und Verwendung ... 1077
18.4 Rechtliches ... 1078
18.4.1 EU-Konformitätserklärung – RTC6 EtherBox ... 1078
18.4.2 Bestätigung elektromagnetische Verträglichkeit (EMV) ... 1079
18.4.3 Konformität mit FCC-Bestimmungen ... 1080
18.5 Technische Spezifikationen – RTC6 EtherBox ... 1080

19 Anhang D: RTC6-Software-Pakete – Historie ... 1081

20 Anhang E: i*DRIVE*-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale ... 1083

21 Anhang F: Über Rückkanal-Multiplexing und Erweiterter Scan-Kopf-Status ... 1126
21.1 Rückkanal-Multiplexing ... 1128
21.1.1 Konfigurieren, welche Werte übertragen werden sollen (Beispiel-Code) ... 1129
21.1.2 Übertragung (durch das i*DRIVE*-Scan-System) auslösen ... 1131
21.1.3 Werte zuordnen (Demultiplex) ... 1132
Beispiel-Code (C) ... 1133
21.2 Erweiterter Scan-Kopf-Status ... 1134
21.2.1 Erweiterter Scan-Kopf-Status-Signal-IDs ... 1135

22 Anhang G: Änderungsindex ... 1142

1 Über dieses Handbuch

Dieses Handbuch beschreibt die SCANLAB RTC6-Karten und ihre Verwendung zur synchronen Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten.

In den Hauptteilen wird dazu beispielhaft auf die RTC6 PCIe-Karte (= "RTC6 PCI-Express-Karte") Bezug genommen. Zur Produktübersicht siehe Kapitel 2 "Produktübersicht", Seite 32.

Andere RTC6-Karten-Varianten werden im Anhang beschrieben:

- RTC6 Ethernet-Karte, siehe Kapitel 16 "Anhang A: RTC6 Ethernet-Karte", Seite 978

Dieses Handbuch ist Bestandteil des Produkts. Lesen Sie diese Anweisungen sorgfältig durch, bevor Sie mit der Installation und Inbetriebnahme beginnen.

Beachten Sie insbesondere alle Sicherheitshinweise dieses Handbuchs. Bei Fragen zum Inhalt kontaktieren Sie SCANLAB, siehe Kapitel 1.1 "Hersteller".

Halten Sie dieses Handbuch für Wartung, Pflege, Reparatur und Entsorgung verfügbar. Bei einer Veräußerung der Karte sollten Sie dieses Handbuch weitergeben.

Dieses Handbuch bezieht sich auf:

- RTC6-Software-Paket 1.19.1

DLL-Datei für 32-Bit-Anwender-programme[a]	`RTC6DLL.dll`	DLL 647[b]
DLL-Datei für 64-Bit-Anwender-programme[a]	`RTC6DLLx64.dll`	
Programmdatei für den PCIe-DSP	`RTC6OUT.out`	OUT 650[b]
Programmdatei für den Eth-DSP	`RTC6ETH.out`	ETH 650[b]
Firmwaredatei für das FPGA	`RTC6RBF.rbf`	RBF 643[b]
Hilfsdatei	`RTC6DAT.dat`	DAT 604[b]

(a) Software für Laser–Scan-Prozesse, die RTC6-Karten auf Basis dieser RTC6-DLL ansteuert, wird in diesem Handbuch einheitlich als "Anwenderprogramm" bezeichnet.

(b) Kurzform in diesem Handbuch für die Versionen.

Die Versionsnummern der ausgelieferten RTC6-DLL- und RTC6-Dateien sind in den Dateinamen der entsprechenden `zip`-Dateien enthalten, siehe Abschnitt "Verzeichnis RTC6 Files", Seite 34.

Um nach der Installation die Versionsnummern Ihrer Dateien festzustellen, verwenden Sie **get_dll_version**, **get_hex_version** und **get_rtc_version**.

1.1 Hersteller

SCANLAB GmbH
Siemensstr. 2a
82178 Puchheim
Deutschland
Tel. +49 (89) 800 746-0
Fax +49 (89) 800 746-199
info@scanlab.de
www.scanlab.de

1.2 Weiterführende Dokumente

- Handbuch "excelli*SCAN*-Scan-Köpfe – Funktionsprinzip der *SCAN*ahead-Regelung und Ansteuerung mit RTC6-Karten"
- Handbuch "Kalibrieren eines 3-Achsen-Laser-Scan-System"
- Handbuch "StreamParser-DLL – Schnittstelle zur Anwendungsprogrammierung"

1.3 Glossar und Abkürzungen

[*]mark[*]-Befehl	Alle Befehle mit “mark” als Namensbestandteil. Siehe Abschnitt “Mark-Befehle”, Seite 140.
[*]para[*]-Befehl	Alle Befehle mit “para” als Namensbestandteil. Siehe Abschnitt “[*]Para[*]-Befehle”, Seite 142.
[*]scanahead[*]-Befehl	Alle Befehle mit “scanahead” als Namensbestandteil. Speziell zum Ansteuern von SCANahead-Systemen. Siehe Kapitel 8.13 “SCANahead-Funktionalität”, Seite 294.
3D-Bildfeld	Synonym: Arbeitsvolumen Siehe Kapitel 7.3.4 “3D-Bildfeld”, Seite 178.
Antwort-Telegramm	Im Zusammenhang mit Remote-Interface-Modus: Telegramm *von* der RTC6 Ethernet-Karte, siehe `telegrams.h`. Besteht aus Kopf-Daten und Nutz-Daten.
API	Abkürzung für Application Programming Interface (“Schnittstelle zur Anwendungsprogrammierung”). Programmteil, der anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird.
Arbeitsfeld	Synonym: Bildfeld.
BCD	Binary Coded Decimal. Binärcodierte Dezimalziffer.
Befehls-Telegramm	Im Zusammenhang mit Remote-Interface-Modus: Telegramm *an* die RTC6 Ethernet-Karte, siehe `telegrams.h`. Besteht aus Kopf-Daten und Nutz-Daten.
Bildfeld	Synonym: Arbeitsfeld.
BIOS	Basic Input/Output System. Ist im Flash-Speicher der RTC6-Karte dauerhaft gespeichert. Siehe auch `RTC6conf.exe`.
Bogen-Befehl	Überbegriff für Arc-Befehle und Ellipse-Befehle.
BUSY-Pin	Pin mit BUSY OUT-Signal (= BUSY-Listenausführungsstatus). • RTC6 PCIe-Karte: – LASER-Buchse, Seite 75 – EXTENSION 1-Stiftleiste, Seite 81 • RTC6 Ethernet-Karte: – LASER-Stiftleiste, Seite 984 – EXTENSION 1-Stiftleiste, Seite 993
CalibrationLibrary	Siehe www.scanlab.de.
Data-Streaming	Siehe Handbuch “StreamParser-DLL – Schnittstelle zur Anwendungsprogrammierung”, Kurzinformation Data-Streaming.
DSP	Digitaler Signal Prozessor auf der RTC6-Karte.

Dynamische Fokussiereinheit	Dazu gehören z.B. folgende SCANLAB-Produkte: vario*SCAN*, vario*SCAN*$_{de}$, vario*SCAN* FC, vario*SCAN FLEX* excelli*SHIFT*.
Ethernet Link Loss	Die Kabelverbindung zur RTC6 Ethernet-Karte wurde getrennt.
Flash-Speicher	Nichtflüchtiger Speicher auf der RTC6-Karte, der das EEPROM der RTC5-Karte ersetzt.
FPGA	Field Programable Gate Array auf der RTC6-Karte.
Hardware-Reset	Neustart der RTC6-Karte nach Wiederbestromung. Synonym: “Power-Up”, “Power-Cycle”.
Harter Sprung	Direkte Ausgabe zu einer angegebenen Position. Zerlegung der Mikroschritte in einen einzigen 10 µs-Takt.
i*DRIVE*-Scan-Systeme	Der Begriff subsumiert in diesem Handbuch z.B. folgende SCANLAB-Produkte: intelli*SCAN*, intelli*SCAN*$_{de}$, intelli*SCAN*$_{se}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario*SCAN*$_{de}$, power*SCAN* II 50i, excelli*SCAN*.
intelli*SCAN*	Der Begriff subsumiert in diesem Handbuch z.B. folgende SCANLAB-Produkte: intelli*SCAN*, intelli*SCAN*$_{de}$, intelli*SCAN*$_{se}$, intelli*DRILL*, intelli*cube*, intelli*WELD* power*SCAN* II 50i.
Lasersteuersignale	LASER1-Signal, LASER2-Signal, LASERON-Signal. Siehe z.B. Kapitel 7.4.1 “Lasersteuersignale freigeben, aktivieren und schalten”, Seite 192.
LSB	Least Significant Bit. Das niedrigstwertige Bit.
Markier-Befehl	Dient dazu, um eine Markierbewegung bei *eingeschaltetem* Laser zu veranlassen. Beispiele: Mark, Arc und Ellipse. Siehe 2D-Markierbefehle, Seite 334 und 3D-Markierbefehle[(1)], Seite 334.
McBSP	Multi channel Buffered Serial Port.
micro_vector[*]-Befehl	Alle Befehle mit “micro_vector” als Namensbestandteil. Siehe Kapitel 8.8 “micro_vector[*]-Befehle”, Seite 285.
Mikroschritt	Siehe Kapitel 7.1.2 “Zerlegung in Mikroschritte”, Seite 143.
Mikrovektor	Der Begriff bezieht sich auf micro_vector[*]-Befehle, siehe Kapitel 8.8 “micro_vector[*]-Befehle”, Seite 285.
MSB	Most Significant Bit. Das höchstwertige Bit.
NAND-Speicher	Nichtflüchtiger Speicher auf der RTC6 Ethernet-Karte. Synonym: NAND-Flash.
NULL	Bedeutet zum Einen die Zahl 0, zum Anderen einen Zeiger mit dem Wert 0. Die Schreibweise dafür ist in den verschiedenen Programmiersprachen unterschiedlich.

nullterminiert	“Mit einem Null-Zeichen als Ende-Kennung abgeschlossen” (ein Zeichen mit dem internen Wert Null, oft auch “NUL” genannt, nicht dasselbe wie die Glyphe Null).
OIE	Open Interface Extension. SCANLAB-Produkt (Hardware). Siehe SCANLAB-Website.
PID	Proportional–Integral–Derivative. Regler-Kategorie.
Pixelmode	Kurz für “Pixelausgabemodus”. Siehe Kapitel 8.7 “Pixelausgabemodus”, Seite 275.
Platine	Synonym: Leiterplatte. Englisch: PCB, printed circuit board.
Polygonzug	Eine direkte Aneinanderreihung von [*]mark[*]-Befehlen oder Bogen-Befehlen. Die Markierung erfolgt kontinuierlich.
PosAck	“Position Acknowledge”, siehe PosAck-Signal.
PosAck-Signal	• Ab Scan-System-Firmware ≥ 2001. • PosAck-Signal-fähige SCANLAB Scan-Systeme, z.B. intelliSCAN • Bezieht sich auf die Bedeutung von: – Bit #12, Bit #4, Bit #11, Bit #3 des mit 16-Bit Protokoll übertragenen XY2-100-Statusworts – Bit #16, Bit #8, Bit #15, Bit #7 des mit 20-Bit Protokoll übertragenen XY2-100-Statusworts • Diese Bits sind gesetzt, wenn die Positionsfehler von x-Achse und y-Achse im erlaubten Bereich sind – Weitere Informationen finden Sie in den entsprechenden Scan-Kopf-Handbüchern. • Vgl. TrAck-Signal.
“Processing on the fly”-Session	Ein Abschnitt einer Liste, die externe Eingaben (Encoderpulse, McBSP-Übertragungen) benutzt, um bewegte Werkstück-Positionen zu korrigieren.
RCM	Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.
Remote-Befehl	Überbegriff für Remote-Kontrollbefehle und Remote-Listenbefehle. Siehe Kapitel 16.10 “Remote-Interface-Modus”, Seite 1021.
Remote-Kontrollbefehl	`telegrams.h` > enum REMOTE_ID > Aufzählungskonstanten R_DC_* ≤ 255. Siehe Kapitel 16.10 “Remote-Interface-Modus”, Seite 1021.
Remote-Listenbefehl	`telegrams.h` > enum REMOTE_ID > Aufzählungskonstanten R_LC_* ≥ 256. Siehe Kapitel 16.10 “Remote-Interface-Modus”, Seite 1021.
Reset der RTC6-Karte	Synonym: Software-Reset.

StreamParser-DLL	Siehe Handbuch "StreamParser-DLL – Schnittstelle zur Anwendungsprogrammierung".
*SCAN*ahead-System	SCANLAB Scan-System mit SCANahead-Regelung, z.B. excelli*SCAN*-Baureihe. Vgl. Schleppverzugs-System, Vorverarbeitendes System. Nur SCANahead-Systeme übertragen den HEAD BUSY-Listenausführungsstatus (Kontrollbefehl **get_status**, Bit #23 = 1), siehe auch Kapitel 6.4.3 "Listenausführungsstatus", Seite 111.
*SCAN*ahead-Regelung	Ausstattungsmerkmal bestimmter SCANLAB Scan-Systeme (= "SCANahead-Systeme"), im Wesentlichen basierend auf einer ISB1 als Servo-Karte für die 2 Galvanometerscanner mit entsprechender Firmware und Parametrierung. Siehe auch Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten".
Scanner-Delay	Siehe Kapitel 7.2.2 "Scanner-Delays", Seite 150.
Schleppverzug	Zeitdifferenz zwischen dem geplanten und dem tatsächlichen Erreichen einer bestimmten Spiegel-Position bei konstanter Geschwindigkeit.
Schleppverzugs-Regelung	Die "herkömmliche" Regelung der Schleppverzugs-Systeme.
Schleppverzugs-System	SCANLAB Scan-System mit herkömmlicher Regelung (1. Generation) = mit Schleppverzug sowie keine Vorverarbeitung. Z.B. intelli*SCAN* III. Vgl. Vorverarbeitendes System, SCANahead-System.
set_trigger[*]	Lies: "**set_trigger**/**set_trigger4**/**set_trigger8**".
Software-Reset	Neustart nach **load_program_file**. Dabei wird nicht alles zurückgesetzt, z.B. bleiben geladene Korrekturtabellen erhalten. Synonym: Reset der RTC6-Karte.
SPI	Serial Peripheral Interface.
Sprung-Befehl	Dient dazu, die Achsen des Scan-Systems bei *ausgeschaltetem* Laser zu einer neuen Position zu bewegen. Siehe auch Abschnitt "Sprung-Befehle", Seite 140. Siehe 2D-Sprungbefehle, Seite 334 und 3D-Sprungbefehle, Seite 334.
Standalone-Betriebsmodus	Siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.

TrAck	“Trajectory Acknowledge”, siehe TrAck-Signal.
TrAck-Signal	• Ab Scan-System-Firmware ≥ 5102 + ≥ 5112. • TrAck-Signal-fähige SCANLAB Scan-Systeme, z.B. SCANahead-Systeme • Bezieht sich auf die Bedeutung von: – Bit #12, Bit #4, Bit #11, Bit #3 des mit 16-Bit Protokoll übertragenen XY2-100-Statusworts – Bit #16, Bit #8, Bit #15, Bit #7 des mit 20-Bit Protokoll übertragenen XY2-100-Statusworts • Diese Bits sind gesetzt, wenn die Trajektorienfehler von x-Achse und y-Achse im erlaubten Bereich sind – Siehe auch Handbuch “excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten” – Weitere Informationen finden Sie in den entsprechenden Scan-Kopf-Handbüchern • Vgl. PosAck-Signal.
Trajektorie	In diesem Handbuch: Kurve mit 10 μs-Parameterisierung.
Trajektorienfehler	Abweichung der Ist-Trajektorie zur Soll-Trajektorie.
Vektor-Befehl	Überbegriff für Sprung-Befehle und Mark-Befehle.
Vorverarbeitende Regelung	Regelung (neuerer Generation) in “Vorverarbeitenden Systemen” (siehe Vorverarbeitendes System) = mit Schleppverzug sowie mit Vorverarbeitung.
Vorverarbeitendes System	SCANLAB Scan-System mit Vorverarbeitende Regelung Vgl. Schleppverzugs-System, SCANahead-System.

2 Produktübersicht

2.1 Kennzeichnung

Die Seriennummer der RTC6 PCIe-Karte finden Sie auf einem Aufkleber auf der Karte.

Die Artikelnummer und die Konfiguration der Karte sind auf dem jeweiligen Lieferschein aufgeführt, siehe Kapitel 2.6 "Optionen", Seite 39 und Kapitel 2.7 "Jumperkonfigurationen und Typenbezeichnungen", Seite 41.

2.2 Auspackanleitung und typischer Lieferumfang

(1) Nehmen Sie die RTC6 PCIe-Karte vorsichtig aus der Verpackung.

(2) Bewahren Sie sowohl die Umverpackung als auch den antistatischen Beutel, in dem die RTC6 PCIe-Karte ausgeliefert wird, auf. Damit kann die Karte im Fall einer Reparatur wieder optimal verpackt an SCANLAB geschickt werden.

(3) Entnehmen Sie auch alle weiteren Artikel aus der Verpackung. Überprüfen Sie an Hand des Lieferscheins die Vollständigkeit der Lieferung. Der Lieferumfang besteht typischerweise aus einer RTC6 PCIe-Karte und einer Daten-CD (mit dem RTC6-Software-Paket, s.u., und diesem Handbuch). Unter Umständen sind auch Zusatzkomponenten enthalten, siehe Kapitel 2.8 "Zubehör für die RTC6 PCIe-Karte", Seite 45.

2.3 Mitgeliefertes RTC6-Software-Paket

- Siehe auch Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081.

Das mitgelieferte RTC6-Software-Paket enthält den RTC6-Kartentreiber(1) für die 32-Bit- und 64-Bit-Versionen der Betriebssysteme Microsoft Windows 10, 8, 7.

- *Windows XP wird nicht unterstützt!*
- *Windows Vista wird nicht unterstützt!*

Beachten Sie den Sicherheitshinweis zu alten RTC-Kartentreibern in Kapitel 2.5.2 "Software", Seite 38.

Der Inhalt des RTC6-Software-Pakets

- `RTC6-<SW-V>.zip`

ist wie folgt:

- Verzeichnis Correction Files
- Verzeichnis CorrectionFileConverter
- Verzeichnis Demo Files
- Verzeichnis HPGL
- Verzeichnis iSCANCfg
- Verzeichnis RTC6 Driver
- Verzeichnis RTC6 Files
- Verzeichnis RTC6 Tools

(1) Der RTC6-Kartentreiber wird für RTC6 Ethernet-Karten nicht benötigt.

Verzeichnis Correction Files

- `Cor_1to1.ct5`
  1to1-Korrekturdatei[1][2]

Verzeichnis CorrectionFileConverter

- `CorrectionFileConverter.exe`
  Dieses Win32-basierte Programm konvertiert RTC4-Korrekturdateien `*.ctb` in RTC5/RTC6-Korrekturdateien `*.ct5` und umgekehrt. Das zugehörige Handbuch wird in Englisch und Deutsch mitgeliefert.

Verzeichnis Demo Files

Unterverzeichnis Remote Interface

Siehe Kapitel 16.10 "Remote-Interface-Modus", Seite 1021.

- `example_1.cpp`

Verzeichnis HPGL

- `Hpgl.exe` ist ein Win32-basierter HPGL-zu-RTC6-Konverter. Siehe auch Kapitel 5.7 "Funktionstest", Seite 95.
- Das Verzeichnis enthält einige `*.plt`-Dateien (Grafikdateien im Hewlett Packard HPGL-Format (Vektorgrafik-Plotter-Dateien) für Testzwecke.
- Das Verzeichnis enthält die RTC6-Dateien und die `RTC6DLL.dll`, weil `Hpgl.exe` diese im gleichen Verzeichnis benötigt.

Verzeichnis iSCANCfg

- `iSCANcfg.exe` ist ein Win32-basiertes Diagnose- und Konfigurationsprogramm für iDRIVE-Scan-Systeme. Unterstützt werden RTC4, RTC5 und RTC6 (sowohl PCI/PCIe als auch Ethernet-Varianten).
- Benötigt die RTC6-Dateien und die `RTC6DLL.dll` im gleichen Verzeichnis und zusätzlich die `RtcHalDLL.dll`.
  Das zugehörige Handbuch wird in Englisch (`Manual_iSCANcfg_v1-7.pdf`) und Deutsch (`Handbuch_iSCANcfg_v1-7.pdf`) mitgeliefert.

Verzeichnis RTC6 Driver

(RTC6-Kartentreiber für Windows)

- `RTC6DRV.sys`, `RTC6DRVx64.sys`, `RTC6DRVx86.sys`
  RTC6-Kartentreiber-Dateien
- `RTC6DRV.inf`
  Installations-Datei (Setup-Information)
- `RTC6DRVx64.cat`, `RTC6DRVx86.cat`
  Sicherheitskatalog-Dateien
- `amd64/WdfCoInstaller01009.dll`, `x86/WdfCoInstaller01009.dll`
  Installations-Assistenten-Hilfsdateien
- `AfterInstallation/ScanlabClassChecker.cmd`, Security-Installations-Script mit Hinweisen in `ReadMe_ScanlabClassChecker.pdf`

(1) Weitere Korrekturdateien (D2_XXX.CT5, D3_YYY.CT5) und `*_ReadMe.txt`-Datei(en) sind nicht Bestandteil des RTC6-Software-Pakets.

(2) Siehe auch Abschnitt "1to1-Korrekturtabellen", Seite 185.

Verzeichnis RTC6 Files

- changelog.md
  Änderungsliste RTC6-Software-Paket. Englisch
- IMPORTANT_RELEASE_NOTES.txt
  "Wichtige Informationen". Englisch
- liesmich.txt
  Beschreibung RTC6-Software-Paket. Deutsch
- readme.txt
  Beschreibung RTC6-Software-Paket. Englisch
- release_notes_de.md
  Bekannte Einschränkungen. Deutsch
- release_notes_en.md
  Bekannte Einschränkungen. Englisch
- RTC6_Aenderungshistorie.pdf
  Kein Inhalt mehr – verschoben nach changelog.md und versions.md. Deutsch
- RTC6_Release_Notes.pdf
  Kein Inhalt mehr – verschoben nach release_notes_en.md und release_notes_de.md. Bilingual Englisch und Deutsch
- RTC6_RevisionHistory.pdf
  Kein Inhalt mehr – verschoben nach changelog.md und versions.md. Englisch
- versions.md
  Versionen von Release-Komponenten. Englisch
- WICHTIGE_RELEASE_HINWEISE.txt
  "Wichtige Informationen". Deutsch
- Unterverzeichnis Linux
- Unterverzeichnis Program Files
- Unterverzeichnis Remote Interface
- Unterverzeichnis Windows

Unterverzeichnis Linux

- Nur für Linux-Entwickler:
  Debian Pakete mit Shared Libraries zur Ansteuerung von RTC6 Ethernet-Karten
  - readme.txt
    Paketinhalt und Installationsanleitung
  - Unterverzeichnis debian-bullseye
    Paket für Debian Bullseye
  - Unterverzeichnis debian-buster
    Paket für Debian Buster
  - Unterverzeichnis debian-bookworm
    Paket für Debian Bookworm

Unterverzeichnis Program Files

- RTC6-Dateien
  - RTC6OUT.out
    Programmdatei für den PCIe-DSP
  - RTC6ETH.out
    Programmdatei für den Eth-DSP
  - RTC6RBF.rbf
    Firmwaredatei für das FPGA
  - RTC6DAT.dat
    Binäre Hilfsdatei
- RTC6-Dateien und RTC6-DLL verschiedener Versionen können nicht beliebig miteinander kombiniert werden. Die folgenden zip-Dateien (jeweils inklusive Text-Datei mit Versions-/Kompatibilitätsinformationen) sind vorhanden, um die Versionen leichter identifizieren zu können:
  - RTC6OUT_<aktuelle OUT-Versionsnummer>.zip (mit RTC6OUT.out und RTC6ETH.out)
  - RTC6RBF_<aktuelle RBF-Versionsnummer>.zip
  - RTC6DAT_<aktuelle DAT-Versionsnummer>.zip

Unterverzeichnis Remote Interface

Siehe Kapitel 16.10 "Remote-Interface-Modus", Seite 1021.

- rtc6_rif_wrapper.cpp
  C++-Remote-Interface-Wrapper-Source-Datei
  Enthält eine abstrakte Klasse, in der das Senden und Empfangen von Befehls-Telegrammen implementiert ist
- rtc6_rif_wrapper.h
  C++-Remote-Interface-Wrapper-Header-Datei
- telegrams.h
  Header-Datei für rtc6_rif_wrapper.h. Definiert die Struktur der Befehls-Telegramme sowie die Remote-Befehl-IDs

Unterverzeichnis Windows

- RTC6-DLL
  - `RTC6DLL.dll`
    Win32-basierte RTC6 dynamische Link Library
  - `RTC6DLLx64.dll`
    Win64-basierte RTC6 dynamische Link Library
- Utility-Dateien für C, C++ und C#
  - `RTC6expl.c`
    C-Funktionen zum DLL-Handling für das explizite Linken
  - `RTC6expl.h`
    C-Funktionsprototypen der RTC6 zum expliziten Linken der RTC6-DLL
  - `RTC6DLL.lib`
    Visual C++ Import Libraries zum impliziten Linken der RTC6-DLL für Win32-basierte Anwenderprogramme
  - `RTC6DLLx64.lib`
    Visual C++ Import Libraries zum impliziten Linken der RTC6-DLL für Win64-basierte Anwenderprogramme
  - `RTC6impl.h`
    C-Funktionsprototypen der RTC6 zum impliziten Linken der RTC6-DLL
  - `RTC6impl.hpp`
    C++-Funktionsprototypen der RTC6 zum impliziten Linken der RTC6-DLL
  - `RTC6Wrap.cs`
    Importdeklarationen der Wrapper-Klasse zum impliziten Linken der RTC6-DLL unter C#
- Utility-Datei für Delphi
  - `RTC6Import.pas`
    Importdeklarationen für Delphi
- RTC6-Dateien und RTC6-DLL verschiedener Versionen können nicht beliebig miteinander kombiniert werden. Die folgende `zip`-Datei (inklusive Text-Datei mit Versions-/ Kompatibilitätsinformationen) ist vorhanden, um die Versionen leichter identifizieren zu können:
  - `RTC6DLL_<aktuelle DLL-Versionsnummer>.zip`
    (mit zugehörigen Utility-Dateien)

Verzeichnis RTC6 Tools

- `RTC6conf.exe`
  "RTC6 Configuration Tool"
  mit folgenden Haupt-Features
  - für RTC6 PCIe-Karten und RTC6 Ethernet-Karten: Karteninformationen anzeigen (Seriennummer, freigeschaltete Optionen, BIOS-Version), Optionen freischalten (erfordert eine Hilfsdatei, die bei SCANLAB erworben werden muss), BIOS upgraden und downgraden
  - für RTC6 Ethernet-Karten zusätzlich: Karten in (konfigurierbaren) Netzwerksegmenten (Subnetzen) suchen, Netzwerkeinstellungen für die statische IP-Konfiguration anzeigen und ändern.
- `RTC6conf.pdf`
  Beschreibung der Verwendung von `RTC6conf.exe`
- `RTC6BIOSOUT_24.out`
  DSP-Programmdatei für RTC6 PCIe-Karten-BIOS (Version `0x24`)
- `RTC6BIOSETH_43.out`
  DSP-Programmdatei für RTC6 Ethernet-Karten-BIOS (Version `0x43`)(1)
- Die RTC6-Dateien und RTC6-DLL dieses RTC6-Software-Pakets
  - `RTC6DAT.dat`
  - `RTC6DLL.dll`
  - `RTC6ETH.out`
  - `RTC6OUT.out`
  - `RTC6RBF.rbf`
- `SleepMode.cmd`
  Script zum Deaktivieren *aller* Windows Sleep- und Hibernatemodi.
- `ReadMe_SleepMode.pdf`
  Beschreibung der Verwendung von `SleepMode.cmd`

(1) Siehe auch Kapitel 16.7 "BIOS-ETH-Version upgraden und downgraden", Seite 1011.

2.4 Bestimmungsgemäße Verwendung

Die SCANLAB RTC6 PCIe-Karte ist zusammen mit dem zugehörigen RTC6-Software-Paket zur synchronen Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten in Echtzeit über einen Windows-PC mit PCIe-Bus-Schnittstelle vorgesehen.

RTC6-Karten gibt es in verschiedenen Hardware-Varianten, siehe Kapitel 1 "Über dieses Handbuch", Seite 25.

Die mitgelieferte RTC6-DLL stellt einen umfassenden Befehlssatz zum Steuern zur Verfügung. Damit ist eine schnelle und flexible Entwicklung von Software für Laser–Scan-Prozesse möglich.

Die RTC6 PCIe-Karte ist mit einem digitalen Signalprozessor (DSP) bestückt. Er führt bei der Umsetzung der Steuerbefehle auch komplexere Signalbearbeitungsprozesse aus, etwa für die gleichzeitige Ansteuerung zweier Scan-Systeme oder für Koordinatentransformationen.

Zusätzlich können Steuerbefehle (= Listenbefehle) auch auf der RTC6 PCIe-Karte gespeichert und ihre Ausführung dann zu einem späteren Zeitpunkt gestartet werden. Die Ausführung der Befehle durch die RTC6 PCIe-Karte kann so unabhängig vom PC erfolgen. Dadurch ist die harte Echtzeit-Anforderung bei der Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten fortwährend erfüllt, auch wenn der PC gleichzeitig noch weitere Prozesse zur Maschinensteuerung, Netzwerkkommunikation, o.ä. zu bedienen hat.

Die Schnittstelle zum Scan-System und entsprechende Softwarebefehle erlauben eine bidirektionale Kommunikation mit dem Scan-System und somit sowohl die Ansteuerung als auch die Überwachung des Scan-Systems.

Mit der RTC6 PCIe-Karte können gebräuchliche Lasertypen angesteuert werden. Zur Ansteuerung von Lasern verfügt die RTC6 PCIe-Karte über geeignete Schnittstellen, an denen Lasersteuersignale ausgegeben werden, die über Steuerbefehle an die jeweiligen Anforderungen angepasst werden können.

Benutzer können zwischen verschiedenen Laser-Modi wählen und dabei die Signalparameter (z.B. den Signalpegel – HIGH-aktiv oder LOW-aktiv) oder die Ausgabefrequenz geeignet einstellen.

Zur Ansteuerung von Peripheriegeräten und zur Einbindung externer Steuersignale verfügt die RTC6 PCIe-Karte über eine Reihe von Schnittstellen (u.a. einen 16-Bit-Digital-Eingang, einen 16-Bit-Digital-Ausgang, 12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2 und eine RS-232-Schnittstelle, siehe Kapitel 4 "RTC6 PCIe-Karte – Layout und Schnittstellen", Seite 66) und entsprechende Softwarebefehle.

Es können so viele RTC6 PCIe-Karten in einem PC gleichzeitig betrieben werden, wie es der PCIe-Bus erlaubt. Die Gesamtanzahl aus RTC6 PCIe-Karten und RTC6 Ethernet-Karten darf 255 nicht übersteigen.

Außerdem erlaubt die RTC6-DLL sowohl Multi-Threading als auch Multi-Processing. Es können also auch mehrere Anwenderprogramme gleichzeitig verwendet werden.

Ein und dieselbe Karte kann nicht gleichzeitig von mehreren Anwenderprogrammen verwendet werden. Mehrere Threads *eines* Anwenderprogramms können zwar ein und dieselbe Karte verwenden, aber nicht gleichzeitig Befehle senden. Die RTC6-DLL serialisiert diese Zugriffe automatisch.

RTC6 PCIe-Karten sind in unterschiedlichen Konfigurationen verfügbar, siehe Kapitel 2.6 "Optionen", Seite 39 und Kapitel 2.7 "Jumperkonfigurationen und Typenbezeichnungen", Seite 41.

Die Schnittstellen der RTC6 PCIe-Karte werden ab Kapitel 4 "RTC6 PCIe-Karte – Layout und Schnittstellen", Seite 66, die Installation und Inbetriebnahme von RTC6 PCIe-Karten ab Kapitel 5 "Installation und Inbetriebnahme", Seite 91 und die Programmierung ab Kapitel 6 "RTC6-Anwenderprogramme entwickeln", Seite 96 beschrieben. Die einzelnen Befehlsbeschreibungen sind ab Kapitel 10 "RTC6-Befehle", Seite 327 aufgelistet. Die technischen Spezifikationen der RTC6 PCIe-Karte finden Sie in Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", Seite 973.

Vorsicht!

- Die RTC6 PCIe-Karte darf auf keinen Fall außerhalb eines PCs betrieben werden.
- Die RTC6 PCIe-Karte ist ausschließlich für den gewerblichen Gebrauch bestimmt. Sie ist zum Einbau in eine Maschine (in der Regel in ein Lasersystem) vorgesehen und entspricht *nicht* allen Anforderungen an ein gebrauchsfertiges Endprodukt. Nehmen Sie die RTC6 PCIe-Karte erst dann in Betrieb, wenn sie in eine Maschine eingebaut ist, die den Bestimmungen aller dafür anzuwendenden Richtlinien und Normen (des betreffenden Landes) entspricht. Sie ist *nicht* für den Einsatz als Kinderspielzeug, im Haushalt oder unter widrigen Umgebungsbedingungen (z.B. im Freien) geeignet. Entsprechende Vorsichtsmaßnahmen zur Vermeidung solcher Fehlanwendungen sind von Benutzern zu treffen.
- Die Installation und Inbetriebnahme darf nur durch Personen erfolgen, die über eine ausreichende Qualifikation verfügen, u.a. über Kenntnisse im sicheren Umgang mit elektrischen Geräten. Führen Sie Installations- und Wartungsarbeiten nur bei ausgeschalteter Versorgungsspannung und ausgeschaltetem Laser aus.
- Die RTC6 PCIe-Karte ist eine Einrichtung der Klasse A. Sie kann im Wohnbereich Funkstörungen verursachen; in diesem Fall kann vom Betreiber verlangt werden, angemessene Maßnahmen durchzuführen.

2.5 Systemanforderungen

2.5.1 Hardware

Die RTC6 PCIe-Karte kann in jeden Windows-PC mit PCIe-Bus und mindestens einem freien PCIe-Steckplatz installiert werden.

RTC6 PCIe-Karten, die Master/Slave-synchronisiert werden sollen, sollten (empfohlen) in benachbarte PCIe-Steckplätze gesteckt werden.

2.5.2 Software

Um die RTC6 PCIe-Karte zu betreiben, müssen RTC6-Kartentreiber und RTC6-DLL-Dateien für Microsoft Windows Betriebssysteme verwendet werden. Diese sind im Lieferumfang enthalten (RTC6-Software-Paket). Zu den unterstützten Windows-Versionen siehe Kapitel 2.3 "Mitgeliefertes RTC6-Software-Paket", Seite 32.

Der RTC6-Kartentreiber unterstützt die Plug-and-Play-Fähigkeit der RTC6 PCIe-Karte sowie den gleichzeitigen Betrieb von bis zu 255 RTC6 PCIe-Karten. Die RTC6-DLL-Dateien enthalten den Befehlssatz zur Ansteuerung von Laser-Scan-Systemen.

Achtung!

- Die RTC6 PCIe-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCIe-Busses abgeschaltet werden. Deaktivieren Sie entsprechende Standby- oder Sleep-Modi des Betriebssystems. Siehe auch Abschnitt "Hinweise", Seite 38.

Achtung!

- Falls auf Ihrem PC bereits ein auf WDM-Technologie basierender RTC3/4/5-Kartentreiber
  - installiert ist oder
  - installiert war und deinstalliert wurde oder
  - Sie sich diesbezüglich nicht sicher sind:
    (1) Installieren Sie den RTC6-Kartentreiber.
    (2)Führen Sie `ScanlabClassChecker.cmd` als Administrator aus (im mitgelieferten RTC6-Software-Paket enthalten; siehe dort auch Hintergrundinformationen in `ReadMe_ScanlabClassChecker.pdf`).
    Schritt 2 kann auf neuen PCs, auf denen noch nie ein RTC-Kartentreiber installiert war, auch ausgelassen werden.

Hinweise

- RTC6-Software-Pakete enthalten nur den WDF(1)-Treiber (Version 6.1.7600.16385).
  Ein WDM(2)-Treiber (heute veraltet) ist nicht enthalten.
- Wenn **init_rtc6_dll** aufgerufen wird, unterbindet der RTC6-Kartentreiber eine automatische Aktivierung eines Standby- oder Sleep-Modus (dauerhaft, bis wieder ein System-Neustart erfolgt).
  Dadurch können die RTC6 PCIe-Karten weiter autonom Listen abarbeiten, auch wenn das initiierende Anwenderprogramm bereits beendet ist.
  Allerdings können manuell oder durch entladene Akkus erzwungene Standby- oder Sleep-Modi nicht verhindert werden. Danach sind geladene Listen und sonstige Einstellungen der RTC6 PCIe-Karte verloren. Nach einem Wake-Up ist die RTC6 PCIe-Karte möglicherweise nicht mehr korrekt ansprechbar. Stellen Sie vor dem erstmaligen Aufruf von **init_rtc6_dll** sicher, dass Standby- oder Sleep-Modi des Betriebssystems deaktiviert sind.
  Das im RTC6-Software-Paket (unter RTC6 Tools) enthaltene Script `SleepMode.cmd` deaktiviert *alle* Sleep- und Hibernation-Modi, siehe `ReadMe_SleepMode.pdf`.

(1) Windows Driver Framework. Aktuelle Technologie.

(2) Windows Driver Model. Veraltete Technologie.

2.6 Optionen

RTC6 PCIe-Karten können mit unterschiedlichen Optionen (Ausstattungsmerkmalen) ausgerüstet werden. Bei neuen Karten werden die bestellten Optionen bei Auslieferung im SCANLAB-Werk freigeschaltet/installiert. Zur Nachrüstung von bereits ausgelieferten Karten siehe Abschnitt "Hinweise", Seite 40.

Welche Optionen tatsächlich auf einer bestimmten Karte freigeschaltet sind, kann mit **get_rtc_version** abgefragt werden.

Für RTC6 PCIe-Karten sind die folgenden Optionen erhältlich:

- **Option "Processing on the fly"**
  Erlaubt, dass eine "Processing on the fly"-Korrektur aktiviert werden kann, siehe Kapitel 8.6 ""Processing on the fly"", Seite 254.
- **Option "3D"**
  – **Ansteuerung eines 3-Achsen-Scan-Systems**
  Erlaubt, dass über die beiden Scan-Kopf-Anschlüsse der RTC6 PCIe-Karte synchron zu x-Achse und y-Achse eines Scan-Systems (in der Regel zwei Galvanometerscanner) auch eine dritte Achse (z-Achse, z.B. ein vario*SCAN* als dynamische Fokussiereinheit) angesteuert werden kann, siehe Kapitel 8.5.2 "3D-Scan-Systeme", Seite 249.
- **Option "Zweite-Scan-Kopf-Steuerung"**
  – **Gleichzeitige Ansteuerung zweier Scan-Systeme**
  Erlaubt, dass über die beiden Scan-Kopf-Anschlüsse der RTC6 PCIe-Karte gleichzeitig zwei xy-Scan-Systeme angesteuert werden können, siehe auch Kapitel 4.5.1 "Scan-Kopf-Anschlüsse und Übertragungsprotokoll", Seite 69 und Kapitel 8.5 "2D-Scan-Systeme und 3D Scan-Systeme steuern", Seite 248.
- **Option "SCANa"**
  – **Ansteuerung von SCANahead-Systemen**
  Erlaubt, dass (auch) SCANahead-Systeme (z.B. excelli*SCAN*-Baureihe), angesteuert werden können. Siehe auch Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten".
- **Option "UFPM"**
  – **Pixelausgabemodi mit Pixelausgabefrequenzen über 800 kHz**(1)
  Erlaubt, dass in bestimmten Pixelausgabemodi auch Pixelausgabefrequenzen von > 800 kHz…3,2 MHz erreicht werden können, siehe auch **set_pixel_line**.
- **Option "syncA"**
  – **Unterstützung der sync*AXIS* control-Software**
  Erlaubt, dass die SCANLAB sync*AXIS* control-Software genutzt werden kann (Karte und Software sind im SCANLAB-Lieferumfang für XL *SCAN*-Systeme enthalten).
- **Option "DC/DC-Wandler"**
  RTC6 PCIe-Karten mit der Option "DC/DC-Wandler" sind ab Werk mit einem zusätzlichen DC/DC-Wandler (Optokoppler) ausgestattet.

Achtung!

- Bei RTC6 PCIe-Karten mit der Option "DC/DC-Wandler" sind GND2 und die Lasersteuersignale LASERON, LASER1 und LASER2 an der LASER-Buchse und an der EXTENSION 2-Stiftleiste galvanisch von GND entkoppelt.(a)
- Bei RTC6 PCIe-Karten ohne Option "DC/DC-Wandler" sind GND2 und GND identisch.

(a) RTC6 PCIe-Karten: GND ist die PC-Masse.
RTC6 Ethernet-Karten: GND ist die Masse am POWER-Stecker.

Siehe auch:

– Abschnitt "Lasersteuersignale", Seite 76
Abschnitt "Lasersteuersignale", Seite 84

(1) Ultra Fast Pixel Mode.

Hinweise

- Aus der Artikelnummer einer RTC6 PCIe-Karte ergibt sich, welche der obigen Optionen freigeschaltet und/oder installiert sind. Diese sind im Lieferschein aufgeführt. Dort ist die Benennung wie folgt:
  - "Fly" für Option "Processing on the fly"
  - "3D" für Option "3D"
  - "SSHC" für Option "Zweite-Scan-Kopf-Steuerung"
  - "SCANa" für Option "SCANa"
  - "UFPM" für Option "UFPM"
  - "syncA" für Option "syncA"
  - "DCDC" für Option "DC/DC-Wandler"
- Welche Optionen tatsächlich auf einer Karte freigeschaltet sind, kann mit **get_rtc_version** abgefragt werden.
- Falls Sie eine oder mehrere Optionen benötigen, die auf Ihrer RTC6 PCIe-Karte ab Werk nicht freigeschaltet sind: Bestellen Sie bei SCANLAB unter Angabe der Seriennummer der Karte und den gewünschten Optionen eine entsprechende Lizenz-Datei. Geben Sie diese dann in `RTC6conf.exe` an, um das Upgrade auf die Karte anzuwenden.
- Zur nachträglichen Installation des zusätzlichen DC/DC-Wandlers (Optokoppler) auf der RTC6 PCIe-Karte müssen Sie diese an SCANLAB einschicken.
- Zur optischen Datenübertragung zwischen RTC6 PCIe-Karte und Scan-System ist lediglich ein passendes Datenkabel erforderlich, siehe Kapitel 4.5.3 "Datenkabel (Zubehör)", Seite 73. Weder die RTC6 PCIe-Kartenseite noch die Scan-Systemseite benötigt dafür eine spezielle optische Datenschnittstelle.
- Mit der Software-Paket-Version ≤ V1.4.4 kann die "Processing on the fly"-Funktionalität nicht für SCANahead-Systeme genutzt werden.

2.7 Jumperkonfigurationen und Typenbezeichnungen

SCANLAB liefert RTC6 PCIe-Karten in unterschiedlichen Jumperkonfigurationen aus. Jumper sind Verbindungen, die entweder offen oder geschlossen sind. Die *werksseitige* Lötjumperkonfiguration einer RTC6 PCIe-Karte ergibt sich aus ihrer Artikelnummer.

Außerdem wird eine dreiziffrige Typenbezeichung verwendet, z.B.

- "RTC6 PCI-Express TYPE 000"
  - Keine Signale (d. h. alle Lötjumper sind offen)
- "RTC6 PCI-Express TYPE 124"
  - 5 V Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste
  - DATA7 an Pin (15) der EXTENSION 2-Stiftleiste
  - LATCH an Pin (17) der EXTENSION 2-Stiftleiste

Ziffer 1	Betrifft Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste(1). =0: Keine Signale. =1: 5 V. =2: 3,3 V.
Ziffer 2	Betrifft Pin (15) der EXTENSION 2-Stiftleiste(3). =0: Keine Signale. =1: +5 V. =2: DATA7. =3: GROUND.
Ziffer 3	Betrifft Pin (17) der EXTENSION 2-Stiftleiste(2). =0: Keine Signale. =1: +5 V. =2: DATA7. =3: GROUND. =4: LATCH.

Lötjumperverbindungen können von geschulten Benutzern mit einem Lötkolben umkonfiguriert werden. Die Zuweisung eines gewünschten Signals erfolgt durch Schließen oder Öffnen (Lot oder 0-Ohm-Widerstand aufbringen oder entfernen) entsprechender Lötjumper wie im Folgenden(1)(2)(3) beschrieben.

Achtung!

- Nur zulässige Jumpereinstellungen vornehmen. Anderenfalls wird die Karte beschädigt!

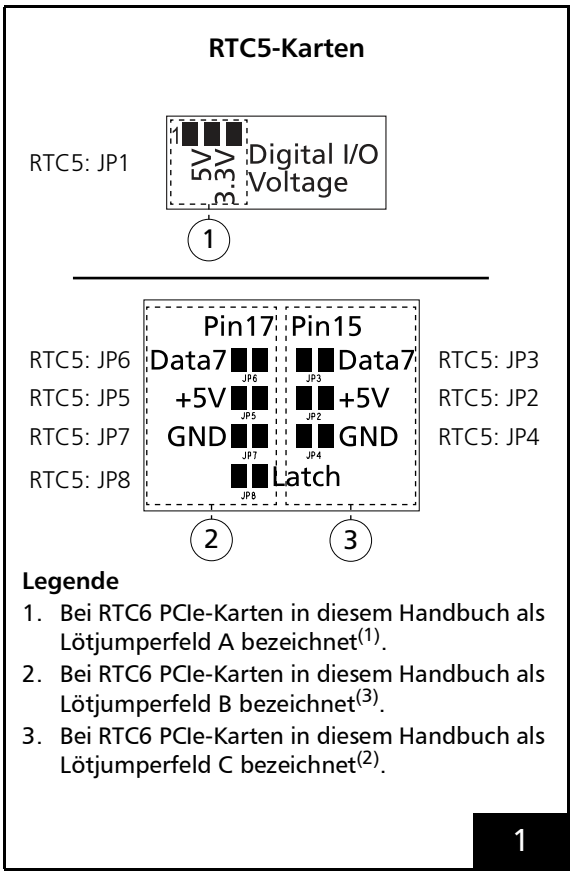
(1) Zur Lötjumperstellung siehe Kapitel 2.7.1 "Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste", Seite 42.

(2) Zur Lötjumperstellung siehe Kapitel 2.7.2 "Lötjumperfeld B – Signal an EXTENSION 2-Stiftleiste Pin (17)", Seite 43.

(3) Zur Lötjumperstellung siehe Kapitel 2.7.3 "Lötjumperfeld C – Signal an EXTENSION 2-Stiftleiste Pin (15)", Seite 44.

Hinweise für RTC5-Benutzer

- RTC6 PCIe-Karten haben im Gegensatz zu RTC5-Karten keine Jumpernummern mehr aufgedruckt. Deswegen können die RTC5-Jumperbezeichnungen in diesem RTC6-Handbuch nicht mehr verwendet werden. Den Bezug der Benennungen stellt Abbildung 1 her.



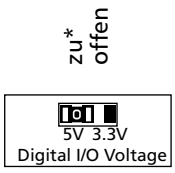
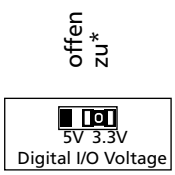
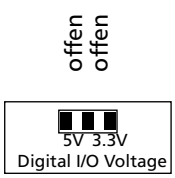
Benennung der Jumper bei RTC5-Karten und RTC6 PCIe-Karten. Abbildung zeigt RTC5-Karte.

2.7.1 Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste

Das Lötjumperfeld A befindet sich auf der Unterseite der RTC6 PCIe-Karte, siehe Abbildung 6.

Es dient zum Einstellen des Pegels (5 V oder 3,3 V) aller Ausgangssignale der EXTENSION 1-Stiftleiste, siehe folgende Tabelle.

Siehe auch Abschnitt "Ausgangssignal-Pegel einstellen", Seite 81.

Zulässige Jumpereinstellung	EXTENSION 1-Stiftleiste
	Ausgangssignal-Pegel 5 V.
	Ausgangssignal-Pegel 3,3 V.
	Keine Ausgangssignale.

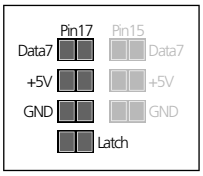
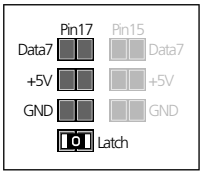
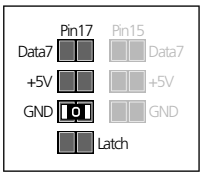
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!

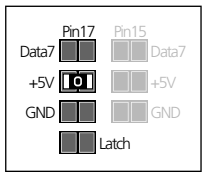
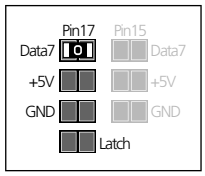
2.7.2 Lötjumperfeld B – Signal an EXTENSION 2-Stiftleiste Pin (17)

Das Lötjumperfeld B befindet sich auf der Unterseite der RTC6 PCIe-Karte, siehe Abbildung 6.

Es dient zum Einstellen des Signals am Pin (17) der EXTENSION 2-Stiftleiste, siehe folgende Tabelle.

Siehe auch "Einstellungen mittels Lötjumper", Seite 83.

Zulässige Jumpereinstellung	Ausgabe an der EXTENSION 2-Stiftleiste Pin (17)
offen offen offen offen 	Kein Signal.
offen offen offen zu* 	LATCH-Signal.
offen offen zu* offen 	GROUND (Low-Pegel).
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!	

Zulässige Jumpereinstellung (Forts.)	Ausgabe an der EXTENSION 2-Stiftleiste Pin (17) (Forts.)
offen zu* offen offen 	+5 V (High-Pegel).
zu* offen offen offen 	DATA7[(a)].
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!	

(a) Synonym: Daten-Bit #7. MSB d. 8-Bit-Ausgangswerts.

Hinweise

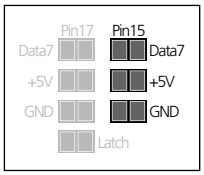
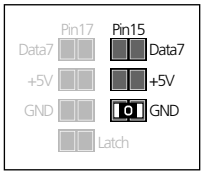
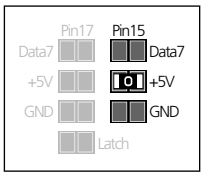
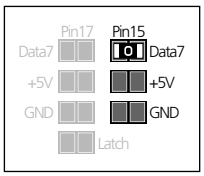
- Die Konfigurationen von Lötjumperfeld B und Lötjumperfeld C sind nicht voneinander abhängig.
- Auf RTC6 Ethernet-Karten zeigt die Bedruckung des Lötjumperfeld B 'Pin17'. Diese wurde bewusst gewählt, um die Konsistenz mit bisherigen RTC-Karten zu bewahren. Tatsächlich wird bei RTC6 Ethernet-Karten jedoch Pin (09) der EXT. 2-Stiftleiste, siehe Kapitel 16.2.12 "EXT. 2-Stiftleiste", Seite 994, konfiguriert.

2.7.3 Lötjumperfeld C – Signal an EXTENSION 2-Stiftleiste Pin (15)

Das Lötjumperfeld C befindet sich auf der Unterseite der RTC6 PCIe-Karte, siehe Abbildung 6.

Es dient zum Einstellen des Signals am Pin (15) der EXTENSION 2-Stiftleiste, siehe folgende Tabelle.

Siehe auch "Einstellungen mittels Lötjumper", Seite 83.

Zulässige Jumpereinstellung	Ausgabe an der EXTENSION 2-Stiftleiste Pin (15)
 offen offen offen	Kein Signal.
 offen offen zu*	GROUND (Low-Pegel).
 offen zu* offen	+5 V (High-Pegel).
 zu* offen offen	DATA7[(a)].
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!	

(a) Synonym: Daten-Bit #7. MSB d. 8-Bit-Ausgangswerts.

Hinweise

- Die Konfigurationen von Lötjumperfeld C und Lötjumperfeld B sind nicht voneinander abhängig.
- Auf RTC6 Ethernet-Karten zeigt die Bedruckung des Lötjumperfeld C 'Pin15'. Diese wurde bewusst gewählt, um die Konsistenz mit bisherigen RTC-Karten zu bewahren. Tatsächlich wird bei RTC6 Ethernet-Karten jedoch Pin (08) der EXT. 2-Stiftleiste, siehe Kapitel 16.2.12 "EXT. 2-Stiftleiste", Seite 994, konfiguriert.

2.8 Zubehör für die RTC6 PCIe-Karte

In Kombination mit der RTC6 PCIe-Karte sollten nur Hardwareerweiterungen von SCANLAB verwendet werden. Für die RTC6 PCIe-Karte und zum zugehörigen Software-Paket sind die folgenden Komponenten zusätzlich erhältlich:

- XY2-100-Konverter, Seite 45
- Laser-Adapter, Seite 45
- Datenkabel, Seite 45
- Slotblech mit 9-poliger Sub-D-Buchse für die 2. SCANHEAD-Stiftleiste, Seite 46
- Slotblech mit 15-poliger Sub-D-Buchse für die MARKING ON THE FLY-Stiftleiste, Seite 46
- Slotblech mit 15-poliger Sub-D-Buchse und 9-poliger Sub-D-Buchse, Seite 47

Hinweise

- Zur UFPM-Erweiterungskarte für die RTC6 PCIe-Karte(1) siehe Kapitel 17 "Anhang B: UFPM-Erweiterungskarte", Seite 1070.
- Die folgende Komponente kann mit RTC6 PCIe-Karten verwendet werden:
  - Erweiterungskarte "RTC5/6 varioSCAN 40 FLEX Extension" (#0128683)(2)
- Die folgenden Komponenten können *nicht* mit RTC6 PCIe-Karten verwendet werden:
  - EXT1 Erweiterungskarte für die RTC5 (#0123804)(3)
  - ADC-Aufsteckplatine (#0121126)
  - Erweiterungskarte "RTC STEP MOTOR EXTENSION" (#0112097)
  - I/O-Erweiterungskarte für RTC3 und RTC4 (#0108285) (#0121721)

(1) Empfohlen für Benutzer mit ANALOG angesteuerten Lasern, die Pixelausgabefrequenzen > 100 kHz erzielen möchten, siehe **set_pixel_line**.

(2) Kann auch mit RTC6 Ethernet-Karten verwendet werden. Erfordert Y-Kabel #0116050 (vgl. Abbildung 83) und Flachbandkabel #0105446. Außerdem müssen Kunden ein 1:1-Kabel bereitstellen (Stecker 1: baugleich zu Würth 61201023021, 10-polig, weiblich; Stecker 2: siehe unter Hinweise, Seite 991).

(3) Für RTC5 entwickelt. Kann aus mechanischen Gründen (Kollision mit Kühlkörper) nicht mit der RTC6 PCIe-Karte verwendet werden (Ersatz: UFPM-Erweiterungskarte, Seite 1070).

2.8.1 XY2-100-Konverter

Mit dem XY2-100-Konverter (Zubehör) kann die RTC6 PCIe-Karte Scan-Systeme ansteuern, die mit einer herkömmlichen XY2-100-Schnittstelle ausgestattet sind.

2.8.2 Laser-Adapter

Der SCANLAB Laser-Adapter wird auf die 15-polige LASER-Buchse der RTC6 PCIe-Karte aufgesteckt. Dann stehen über eine 9-polige Sub-D-Buchse die gleichen Signale mit identischer Pinbelegung wie an der 9-poligen LASER-Buchse der RTC4/RTC5 zur Verfügung. Siehe auch Abschnitt "Laser-Adapter (Zubehör)", Seite 79.

2.8.3 Datenkabel

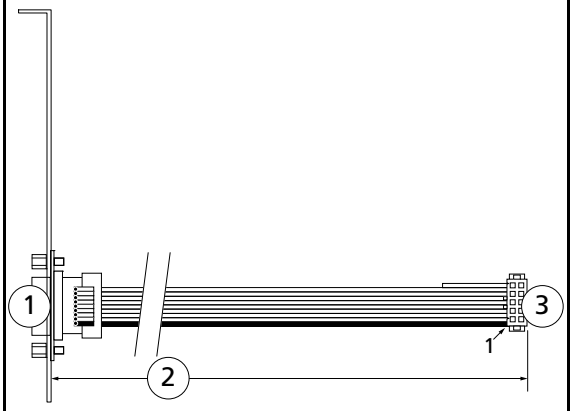
Zur Verbindung der RTC6 PCIe-Karte mit Scan-Systemen sind bei SCANLAB geeignete Datenkabel verschiedener Länge – wahlweise Datenkabel zur elektrischen Datenübertragung oder Lichtwellenleiterkabel zur optischen Datenübertragung – erhältlich. Siehe auch Kapitel 4.5.3 "Datenkabel (Zubehör)", Seite 73.

2.8.4 Slotblech mit 9-poliger Sub-D-Buchse für die 2. SCANHEAD-Stiftleiste

Zum Anschließen eines zweiten Scan-Systems oder einer z-Achse an die 2. SCANHEAD-Stiftleiste ist ein Slotblech erhältlich, siehe Abbildung 2:

- #0115132

Seine 9-polige Sub-D-Buchse (weiblich) hat die gleiche Pinbelegung wie die SCANHEAD-Buchse, siehe Abbildung 10.
Siehe auch Abschnitt "Slotblech für Zweiten Scan-Kopf (Zubehör)", Seite 70.



Legende

1. 9-polige Sub-D-Buchse (weiblich). Pinbelegung wie Abbildung 10.
2. l = 225 mm.
3. Zur 2. SCANHEAD-Stiftleiste.

2

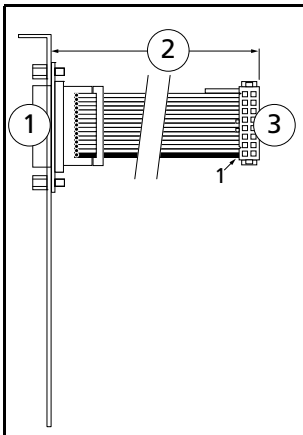
Slotblech für Zweiten Scan-Kopf (Zubehör) #0115132.

2.8.5 Slotblech mit 15-poliger Sub-D-Buchse für die MARKING ON THE FLY-Stiftleiste

Zum Verwenden der Eingänge und Signale an der MARKING ON THE FLY-Stiftleiste ist ein Slotblech erhältlich, siehe Abbildung 3:

- #0109272

Die Pinbelegung seiner 15-poligen Sub-D-Buchse (weiblich) zeigt Abbildung 26.
Siehe auch Abschnitt "MARKING ON THE FLY-Slotblech (Zubehör)", Seite 86.



Legende

1. 15-polige Sub-D-Buchse (weiblich). Pinbelegung wie Abbildung 26.
2. l = 105 mm.
3. Zur MARKING ON THE FLY-Stiftleiste.

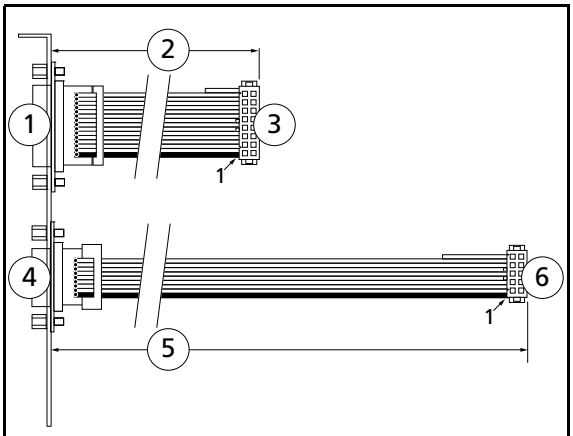
3

MARKING ON THE FLY-Slotblech (Zubehör) #0109272.

2.8.6 Slotblech mit 15-poliger Sub-D-Buchse und 9-poliger Sub-D-Buchse

Die Anschlüsse von #0115132 und #0109272 zusammen auf einem einzigen Slotblech (zum gleichen Zweck wie diese) bietet das Slotblech, siehe Abbildung 4:

- #0130209



Legende

1. 15-polige Sub-D-Buchse (weiblich). Pinbelegung wie Abbildung 26.
2. l = 105 mm.
3. Zur MARKING ON THE FLY-Stiftleiste.
4. 9-polige Sub-D-Buchse (weiblich). Pinbelegung wie Abbildung 10.
5. l = 225 mm.
6. Zur 2. SCANHEAD-Stiftleiste.

4

Slotblech #0130209.

2.8.7 Erweiterungskarte "RTC5/6 varioSCAN FLEX Extension"

Für die RTC5/6-Karten erhältlich ist[1]:

- Erweiterungskarte "RTC5/6 varioSCAN 40 FLEX Extension" (#0128683)

Sie wurde speziell entwickelt, um mit der SCANLAB laserDESK-Software analoge und digitale vario*SCAN* 40 FLEX anzusteuern. Über den Schrittmotor wird eine Positionsänderung der vario*SCAN* 40 FLEX Fokussieroptik bewirkt und somit letztlich der Arbeitsabstand des 3D-Scan-Systems geändert.

Weitere Informationen finden Sie im zugehörigen Handbuch "Installation and Operation RTC5/6 vario*SCAN FLEX* Extension for RTC5 and RTC6 Control Boards".

Hinweise

- Die Erweiterungskarte "RTC5/6 varioSCAN 40 FLEX Extension" (#0128683) unterstützt **move_to** *nicht*[2].

(1) Im Gegensatz zur Erweiterungskarte "RTC STEP MOTOR EXTENSION" (#0112097) – die auch mit der RTC5-Karte verwendet werden kann – bietet sie daneben den Vorteil, dass die EXTENSION 1-Stiftleiste frei bleibt.

(2) move_to wurde für die Erweiterungskarte "RTC STEP MOTOR EXTENSION" (#0112097) eingeführt.

2.9 Ergänzende Software

Zur individuellen Anpassung von RTC-Korrekturdateien auf Basis der Daten Ihrer eigenen Testmessungen ist bei SCANLAB die Software correXion pro mit zugehörigem Handbuch erhältlich, siehe auch Abschnitt "Bildfeld-Korrektur-Algorithmus", Seite 182.

Mit der SCANLAB laserDESK-Software können ohne eigene Softwareentwicklung Laserbeschriftungs- und Lasermaterialbearbeitungsprogramme ("Laserjobs") erstellt und ausgeführt werden. Viele der RTC6-Funktionen werden unterstützt.(1)
Weitere Informationen zu laserDESK finden Sie auf der SCANLAB Homepage.

(1) Siehe auch Achtung!, Seite 75.

2.10 Hinweise an RTC4-Benutzer

Dieses Kapitel gibt eine Übersicht über die wichtigsten Änderungen, die bei RTC6 PCIe-Karten für vergleichbare Funktionen der RTC4-Karten eingeführt wurden.

Unter anderem wird in Kapitel 2.10.2 "RTC4-Quellcodes für die RTC6 PCIe-Karte überarbeiten", Seite 50 auch eine mögliche Vorgehensweise zum Überarbeiten von RTC4-Programmcodes für die RTC6 PCIe-Karte beschrieben.

Bei den einzelnen Befehlsbeschreibungen, Seite 346 wird in der Zeile "RTC4→RTC6" auf Änderungen hingewiesen.

2.10.1 Hardware-Änderungen

Beim Wechsel von der RTC4 zur RTC6 PCIe-Karte müssen die folgenden Hardware-Änderungen für eine korrekte Verkabelung mit den Systemkomponenten berücksichtigt werden.

Scan-Systeme steuern

- Zur Ansteuerung von Scan-Systemen mit XY2-100-Schnittstelle oder XY2-100 Enhanced-Schnittstelle wird zusätzlich der XY2-100-Konverter (Zubehör) benötigt.
- Mit der RTC6 PCIe-Karte können *keine* Scan-Systeme mit XY2-100-O-Schnittstelle (zur optischen Datenübertragung) angesteuert werden.
- Zur Ansteuerung von Scan-Systemen mit SL2-100-Schnittstelle wird ein anderes Datenkabel benötigt (bei SCANLAB erhältlich), siehe Kapitel 4.5.3 "Datenkabel (Zubehör)", Seite 73.
- Zur Verwendung des Anschluss für zweiten Scan-Kopf wird ein anderes Adapterkabel benötigt (bei SCANLAB mit Slotblech erhältlich), siehe Abschnitt "Slotblech für Zweiten Scan-Kopf (Zubehör)", Seite 70.
- Zur Ansteuerung *eines* 3-Achsen-Scan-Systems müssen beide Scan-Kopf-Anschlüsse verwendet werden, siehe Kapitel 8.5.2 "3D-Scan-Systeme", Seite 249.
- Für die gleichzeitige Ansteuerung *zweier* 3-Achsen-Scan-Systeme werden zwei RTC6 PCIe-Karten benötigt, siehe Kapitel 8.5.2 "3D-Scan-Systeme", Seite 249.

Laser steuern

- Die LASER-Buchse am RTC6 PCIe-Karten-Slotblech ist 15-polig (RTC4: 9-polig). Deren Pinbelegung kann nicht über Jumper konfiguriert werden.
  - Die Jumper **X6** und **X7** der RTC4 werden auf der RTC6 PCIe-Karte nicht benötigt, weil alle Signale an der RTC6 PCIe-Karten-LASER-Buchse bereitgestellt werden.
  - Der Ausgangsspannungsbereich der Analog-Ausgänge ist immer 0…10 V (0 V…2,50 V ist nicht mehr möglich; Jumper **X3** der RTC4 wird auf der RTC6 PCIe-Karte nicht mehr bereitgestellt).
- Soll der Laser über das gleiche (9-polige) Kabel mit der RTC6 PCIe-Karte verbunden werden wie zuvor mit der RTC4, so wird ein Adapter mit 9-poliger Sub-D-Buchse benötigt (bei SCANLAB erhältlich).
  - Beim SCANLAB Laser-Adapter werden zwei Jumper zur Konfiguration der Pinbelegung bereitgestellt, siehe Abschnitt "Laser-Adapter (Zubehör)", Seite 79. **JP1** entspricht dem Jumper **X7** der RTC4, **JP2** entspricht dem Jumper **X6** der RTC4.
- Der Signalpegel der Lasersteuersignale wird nicht mehr über eine Jumper-Konfiguration festgelegt, sondern kann/muss über einen RTC6-Befehl festgelegt werden, siehe **set_laser_control**.
  - Den Jumper **X10** der RTC4 gibt es auf der RTC6 PCIe-Karte nicht mehr.

EXTENSION 1-Stiftleiste

- Die EXTENSION 1-Stiftleiste der RTC6 PCIe-Karte ist – bis auf die zusätzlich bereitgestellten Signale an Pin (33)…Pin (35) – identisch zur EXTENSION 1-Stiftleiste der RTC4, siehe Abbildung 22.
- Bei der RTC6 PCIe-Karte kann über einen Jumper für alle Ausgangssignale der EXTENSION 1-Stiftleiste zwischen einem Signalpegel von 5 V oder 3,3 V ausgewählt werden, siehe Abschnitt "Ausgangssignal-Pegel einstellen", Seite 81.

EXTENSION 2-Stiftleiste

- Der EXTENSION2-Anschluß der RTC4 wird auf der RTC6 PCIe-Karte nicht mehr bereitgestellt. Deswegen kann die “I/O-Erweiterungskarte für RTC3 und RTC4” (#0108285, #0121721) auch nicht an die RTC6 PCIe-Karte angeschlossen werden.
- Die EXTENSION 2-Stiftleiste der RTC6 PCIe-Karte ist – bis auf das wahlweise einstellbare LATCH-Signal an Pin (17) – identisch zur LASER EXTENSION-Stiftleiste der RTC4, siehe Abbildung 24.
  - Zur Konfiguration der Pinbelegung werden auf der RTC6 PCIe-Karte Lötjumperfeld B und C bereitgestellt, siehe auch Abschnitt “Einstellungen mittels Lötjumper”, Seite 83. Diese entsprechen den RTC4-Jumpern **X8** und **X9**.

MARKING ON THE FLY-Stiftleiste

- Die MARKING ON THE FLY-Stiftleiste der RTC6 PCIe-Karte und die MARKING ON THE FLY-Stiftleiste der RTC4 sind identisch, siehe Abbildung 25. Beachten Sie den Sicherheitshinweis zur Encoderzähler-Richtung, Seite 59.

Sonstige Hardware-Schnittstellen

- Die folgenden Schnittstellen werden nur auf der RTC6 PCIe-Karte bereitgestellt:
  - Master-Stiftleiste, Slave-Stiftleiste, Seite 68
  - RS232-Stiftleiste, Seite 86
  - McBSP/ANALOG-Stiftleiste, Seite 87
  - STEPPER MOTOR-Stiftleiste, Seite 90

2.10.2 RTC4-Quellcodes für die RTC6 PCIe-Karte überarbeiten

Anwenderprogramme, die für die RTC4 geschrieben wurden, können für RTC6 PCIe-Karten nur dann verwendet werden, wenn der Quellcode geeignet überarbeitet wird. Dies gilt auch dann, wenn der eigentliche Ablauf des Programms unverändert bleibt.

Veränderte Initialisierung

Der Initialisierungsteil des Programmcodes muss mindestens wie folgt überarbeitet werden:

- Zu Beginn des Programms muss ein **init_rtc6_dll** zur Initialisierung der RTC6-DLL und RTC6-Kartenverwaltung eingefügt werden, siehe Kapitel 6.2.3 “Initialisierung der RTC6-DLL und RTC6-Kartenverwaltung”, Seite 100.
- Die Dateien zur Initialisierung der Karte mit **load_program_file** sind andere als bei der RTC4, siehe Befehlsbeschreibung.
- Bei der Initialisierung der Scan-System-Ansteuerung mit **load_correction_file** und **select_cor_table** müssen andere Korrekturdateien (mit der Dateinamenerweiterung `*.ct5`) verwendet werden, siehe Kapitel 7.3.5 “Bildfeld-Korrektur und Korrekturtabellen”, Seite 181.
- Zur Initialisierung der Laser-Steuerung muss unbedingt zusätzlich **set_laser_control** eingefügt werden, siehe Kapitel 7.4 “Laser-Steuerung”, Seite 192.

Befehlsänderungen

Alle nicht mehr unterstützten RTC4-Befehle müssen aus dem Programmcode entfernt oder ersetzt werden. Siehe auch Kapitel 10.4 "Nicht mehr unterstützte RTC4-Befehle", Seite 963.

Geänderte und erweiterte RTC4-Befehle müssen anders in das Programm eingebunden werden (z.B. durch Anpassung von übergebenen Parameterwerten oder durch veränderte Auswertung zurückgegebener Werte). Auf Änderungen bei den weiterhin unterstützten Befehlen wird bei den einzelnen Befehlsbeschreibungen, Seite 346 in der Zeile "RTC4→RTC6" hingewiesen.

RTC4-Befehle, die ersetzt oder überprüft werden müssen, sind:

- **auto_cal** geändert
- **auto_change_pos** geändert
- **control_command** geändert
- **dsp_start** nicht unterstützt
- **get_head_status** geändert
- **get_hi_data** geändert
- **get_list_space** geändert
- **get_marking_info** geändert
- **get_rtc_version** geändert
- **get_startstop_info** geändert
- **get_status** geändert
- **get_value** geändert
- **get_waveform** geändert
- **get_xy_pos** nicht unterstützt
- **get_xyz_pos** nicht unterstützt
- **goto_xy** geändert
- **goto_xyz** geändert
- **list_jump_cond** geändert
- **list_nop** geändert
- **load_correction_file** geändert
- **load_program_file** geändert
- **read_pixel_ad** nicht unterstützt
- **read_status** geändert
- **rtc4_count_cards** nicht unterstützt
- **select_cor_table** geändert
- **select_list** nicht unterstützt
- **select_rtc** geändert
- **set_control_mode** erweitert
- **set_control_mode_list** erweitert
- **set_laser_mode** erweitert
- **set_laser_timing** geändert
- **set_list_mode** nicht unterstützt
- **set_matrix** geändert
- **set_matrix_list** geändert
- **set_offset** geändert
- **set_offset_list** geändert
- **set_piso_control** nicht unterstützt
- **set_pixel** geändert
- **set_pixel_line** geändert
- **set_softstart_level** nicht unterstützt
- **set_softstart_mode** nicht unterstützt
- **set_trigger** erweitert
- **set_wobbel** geändert
- **set_wobbel_xy** nicht unterstützt
- **timed_jump_abs** geändert
- **timed_jump_rel** geändert
- **timed_mark_abs** geändert
- **timed_mark_rel** geändert
- **z_out** nicht unterstützt
- **z_out_list** nicht unterstützt

Erhöhte Parameter-Auflösung

Zusätzlich muss beim Wechsel von der RTC4 zur RTC6 PCIe-Karte – auch bei einigen der oben nicht aufgeführten Befehle – berücksichtigt werden, dass bei vielen Parametern die Auflösung erhöht wurde. Beispiele:

- Bei Befehlen wie **mark_abs** oder **jump_rel** werden die x-Koordinatenwerte und y-Koordinatenwerte für das reale Bildfeld bei der RTC6 PCIe-Karte mit einer Auflösung von 20 Bit (gegenüber 16 Bit bei der RTC4) angegeben, siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175. Für "Processing on the fly"-Anwendungen steht bei der RTC6 PCIe-Karte darüber hinaus ein erweiterter 29-Bit-Wertebereich zur Verfügung, siehe Kapitel 7.3.3 "Virtuelles Bildfeld", Seite 177.
- Bei Befehlen wie **write_da_x** werden analoge Ausgabewerte bei der RTC6 PCIe-Karte mit einer Auflösung von 12 Bit (gegenüber 10 Bit bei der RTC4) angegeben.
- Bei **set_laser_timing** werden Ausgabeperiode und Pulslänge bei der RTC6 PCIe-Karte mit einer Auflösung von 1/64 μs (gegenüber 1/8 μs oder 1 μs bei der RTC4) angegeben.
- Bei **set_laser_delays** werden die Laser-Delays bei der RTC6 PCIe-Karte mit einer Auflösung von 1/64 μs (gegenüber 1 μs bei der RTC4) angegeben.

Alternativ kann die RTC6-DLL mit **set_rtc4_mode** in den RTC4-Kompatibilitätsmodus geschaltet werden. Dann führt die RTC6-DLL eine entsprechende Umrechnung der Parameterwerte automatisch durch, so dass viele Befehlsfolgen aus RTC4-Programmcodes unverändert für die RTC6 PCIe-Karte übernommen werden können.

Der RTC4-Kompatibilitätsmodus wirkt auf die folgenden RTC4-Befehle (bei allen Befehlen wird in der Zeile "RTC4→RTC6" der jeweiligen Befehlsbeschreibung darauf hingewiesen):

- **arc_abs**
- **arc_rel**
- **fly_return**
- **get_z_distance**
- **goto_xy** (geändert)
- **goto_xyz** (geändert)
- **home_position**
- **jump_abs**
- **jump_abs_3d**
- **jump_rel**
- **jump_rel_3d**
- **mark_abs**
- **mark_abs_3d**
- **mark_rel**
- **mark_rel_3d**
- **set_delay_mode**
- **set_ext_start_delay**
- **set_ext_start_delay_list**
- **set_firstpulse_killer**
- **set_firstpulse_killer_list**
- **set_fly_x**
- **set_fly_y**
- **set_jump_speed**
- **set_laser_delays**
- **set_mark_speed**
- **set_pixel** (geändert)
- **set_pixel_line** (geändert)
- **set_rot_center**
- **set_softstart_level**
- **set_standby**
- **set_standby_list**
- **simulate_ext_start**
- **timed_jump_abs** (geändert)
- **timed_jump_rel** (geändert)
- **timed_mark_abs** (geändert)
- **timed_mark_rel** (geändert)
- **write_da_1**
- **write_da_1_list**
- **write_da_2**
- **write_da_2_list**
- **write_da_x**
- **write_da_x_list**

Die oben beschriebene Überarbeitung der Initialisierung und die Überprüfung auf nicht mehr unterstützte oder geänderte RTC4-Befehle muss unabhängig davon durchgeführt werden, ob das Programm im RTC6-Standardmodus oder im RTC4-Kompatibilitätsmodus ausgeführt werden soll.

Verändertes Zeitverhalten

Die folgenden RTC4-Listenbefehle werden von der RTC6 PCIe-Karte als sogenannte “kurze Listenbefehle” abgearbeitet. Dadurch kann sich bei der Anwenderprogrammausführung ein verändertes Zeitverhalten ergeben, siehe Abschnitt “Normaler Listenbefehl, Kurzer Listenbefehl, Variabler Listenbefehl und Multipler Listenbefehl”, Seite 339.

- **clear_io_cond_list**
- **list_call**
- **list_call_cond**
- **list_jump_cond** (geändert)
- **list_return**
- **save_and_restart_timer**
- **set_extstartpos_list**
- **set_firstpulse_killer_list**
- **set_io_cond_list**
- **set_jump_speed**
- **set_laser_delays**
- **set_laser_timing** (geändert)
- **set_list_jump**
- **set_mark_speed**
- **set_scanner_delays**
- **set_standby_list**
- **set_trigger** (erweitert)
- **set_wobbel** (geändert)
- **write_8bit_port_list**
- **write_da_1_list**
- **write_da_2_list**
- **write_da_x_list**
- **write_io_port_list**

Auch durch automatische Delayanpassungen kann sich ein verändertes Zeitverhalten ergeben, siehe Abschnitt “Automatische Delay-Anpassung”, Seite 158.

2.10.3 Neue und geänderte Funktionalitäten

Schnittstelle zum PC

- Der RTC6-Kartentreiber unterstützt den gleichzeitigen Betrieb von beliebig vielen RTC6 PCIe-Karten in einem einzigen PC, siehe Kapitel 6.6 "Mehrere RTC6 PCIe-Karten in einem PC verwenden", Seite 126.
- Stecker und Befehle zur Master/Slave-Synchronisierung mehrerer RTC6 PCIe-Karten, siehe Kapitel 4.4 "Master-Stiftleiste, Slave-Stiftleiste", Seite 68.

Scan-Systeme steuern

- Neue Schnittstelle zum Scan-System, siehe Kapitel 4.5.1 "Scan-Kopf-Anschlüsse und Übertragungsprotokoll", Seite 69:
  - 9-polige Sub-D-Buchse am RTC6 PCIe-Karten-Slotblech und 10-polige Stiftleiste
  - SL2-100-Protokoll
  - 2 Datenkanäle je Scan-Kopf-Anschluss
  - Galvanische Potentialtrennung
  - 20 Bit Positionsauflösung, siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175
  - Erweiterte Statusrückgabe vom Scan-System, siehe Kapitel 7.3.7 "Statusüberwachung und Diagnose", Seite 191
  - Kontrollkanäle und Statuskanäle zum effektiveren Datenaustausch mit iDRIVE-Scan-Systemen
- Zur Datenübertragung gemäß XY2-100-Protokoll steht der XY2-100-Konverter (Zubehör) zur Verfügung.
  - 25-polige Sub-D-Buchse
  - 16 Bit Positionsauflösung
  - Statusrückgabe gemäß XY2-100- oder XY2-100 Enhanced-Protokoll
  - Synchronisationsanpassung für längere Datenkabel über Lötjumper im XY2-100-Konverter (Zubehör)

  Die RTC6 PCIe-Karte stellt die Versorgungsspannung für den XY2-100-Konverter (Zubehör) zur Verfügung.
- Zur optischen Datenübertragung zwischen RTC6 PCIe-Karte und Scan-System wird keine spezielle Variante der RTC6 PCIe-Karte (mit XY2-100-O-Schnittstelle) benötigt. Die optische Datenübertragung kann über ein Datenkabel von SCANLAB mit Lichtwellenleiter-Umsetzung im Sub-D-Stecker-Gehäuse realisiert werden, siehe Kapitel 4.5.3 "Datenkabel (Zubehör)", Seite 73.
- Ansteuerung eines 3-Achsen-Scan-Systems, siehe Kapitel 8.5.2 "3D-Scan-Systeme", Seite 249:
  - beide Scan-Kopf-Anschlüsse müssen benutzt werden
  - Für die gleichzeitige Ansteuerung zweier 3-Achsen-Scan-Systeme werden 2 RTC6 PCIe-Karten benötigt
- Bildfeld-Korrektur
  - Es werden neue Korrekturdateien benötigt, siehe Kapitel 7.3.5 "Bildfeld-Korrektur und Korrekturtabellen", Seite 181:
    - Dateinamenerweiterung "`.ct5`"
    - Korrektur mit höherer Auflösung
    - Auslesbare Informationen im Korrekturdatei-Kopf
    - RTC4-Korrekturdateien (`.ctb`) müssen für die RTC6 PCIe-Karte neu berechnet oder mit der `CorrectionFileConverter.exe` aus dem RTC6-Software-Paket konvertiert werden
  - Es können bis zu 8 Korrekturdateien auf die RTC6 PCIe-Karte geladen werden
  - Erweiterte 3D-Bildfeld-Korrektur mit Stretch-Korrektur-Tabellen, siehe Kapitel "Erweiterte 3D-Korrektur", Seite 252
- Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236:
  - keine Transformationen (Rotation, Verschiebung, Streckung) der Korrekturdatei beim Download mehr
  - Matrixtransformationen werden erst nach der Zerlegung in Mikroschritte angewendet, d.h. die Markiergeschwindigkeit kann sich durch die Transformation ändern
  - "Lokale Online-Positionierung", siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240

- Koordinatentransformationen im virtuellen Bildfeld (inkl. "Globale Online-Positionierung"), siehe Kapitel 7.3.3 "Virtuelles Bildfeld", Seite 177
  - 29-Bit-Positionskoordinaten (virtuelles Bildfeld): Objekte größer als reales Bildfeld möglich
- Überwachung der Positionierung bei iDRIVE-Scan-Systemen durch Rückwärtstransformation von Ist-Positions-Werten, siehe Kapitel 8.1.3 "Positionierung überwachen", Seite 226
- Automatische Selbstkalibrierung, siehe Kapitel 8.10 "Automatische Selbstkalibrierung", Seite 287:
  - Optimierung der bisherigen Funktionen
  - ASC-Hardware-Überprüfung
- Sprungmodus, siehe Kapitel 8.1.5 "Sprungmodus", Seite 229
- Zyklus-Synchronisation, siehe Kapitel 7.4.10 "Synchronisierung von RTC6-Takt und externem Taktsignal", Seite 219

Laser steuern

- Der Signalpegel der Lasersteuersignale wird nicht mehr über eine Jumper-Konfiguration festgelegt, sondern über einen Software-Befehl, siehe **set_laser_control**
- LASER-Buchse mit allen Lasersteuersignalen am RTC6 PCIe-Karten-Slotblech, siehe Kapitel 4.6.1 "LASER-Buchse", Seite 75, 9-polige Sub-D-Buchse nur über den SCANLAB Laser-Adapter, siehe Abschnitt "Laser-Adapter (Zubehör)", Seite 79
- LASER-Buchse mit Software-Befehl konfigurierbar, siehe Kapitel 7.4.2 "LASER-Buchse konfigurieren", Seite 195
- Lasersteuersignale mit 15 ns zeitlicher Auflösung und 20 mA Ausgangsstrom
- Standby-Signale auch in einem YAG-Modus, siehe Kapitel 7.4.4 "YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5", Seite 198
- YAG-Modus 5: Zeit zwischen FirstPulseKiller-Signal und erstem Laserpuls im YAG-Modus frei programmierbar, siehe Kapitel 7.4.4 "YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5", Seite 198
- Laser-Mode 6: LASERON-Signal mit freilaufendem LASER1-Signal synchronisiert, siehe Kapitel 7.4.6 "Laser-Mode 6", Seite 203
- Pulse-Picking-Laser-Modus, siehe Kapitel 7.4.8 "Pulse-Picking-Laser-Modus", Seite 206
- Laser-Pulsperiode, -Pulslänge oder ein Analog-Ausgang können auch innerhalb eines Polygonzugs zwischen zwei Vektoren programmiert werden – wobei der Laser an bleibt(1), siehe Kurzer Listenbefehl, Seite 339
- Befehle zur Positions-abhängige Laser-Steuerung, Geschwindigkeits-abhängige Laser-Steuerung, Vektorkontrollierte Laser-Steuerung und Encodergeschwindigkeits-abhängige Laser-Steuerung, siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208

(1) Bei der RTC4 geht er aus.

Schnittstellen zur Peripherie

- 16-Bit-Digital-Ausgang, siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81, und Kapitel 9.1.1 "16-Bit-Digital-Ausgang", Seite 306:
  - Ausgangssignal-Pegel über einen Jumper einstellbar (3,3 V oder 5 V)
  - LATCH-Signal zur Synchronisation der Datenübergabe
- 8-Bit-Digital-Ausgang, siehe Abschnitt "8-Bit-Digital-Ausgang", Seite 84, und Kapitel 9.1.2 "8-Bit-Digital-Ausgang", Seite 307:
  - wird auf der EXTENSION 2-Stiftleiste bereitgestellt (auf der RTC4 heißt diese Stiftleiste "LASER EXTENSION")
  - LATCH-Signal zur Synchronisation der Datenübergabe
  - "Stopp-Ausgabewert" einstellbar
- Analog-Ausgänge, siehe Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77, und Kapitel 9.1.4 "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 308:
  - 12 Bit Auflösung
  - 0…10 V (0 V…2,50 V nicht mehr möglich)
  - "Stopp-Ausgabewert" einstellbar
- 16-Bit-Digital-Eingang, siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81 und Kapitel 9.2.1 "16-Bit-Digital-Eingang", Seite 313.
  - SYNC-Signal zur Synchronisation der Datenübergabe
- Programmierbare Entprellung externer Startsignale, siehe **bounce_supp** und Abschnitt "Externer Start", Seite 315
- Regelmäßige (periodische) Externer Starts, siehe Abschnitt "Regelmäßige (periodische) externe Starts", Seite 318
- Neue Schnittstellen:
  - Ein 2-Bit-Digital-Eingang und ein 2-Bit-Digital-Ausgang auf der LASER-Buchse, siehe Abschnitt "2-Bit-Digital-Eingang", Seite 77 und Abschnitt "2-Bit-Digital-Ausgang", Seite 77
  - RS-232-Schnittstelle auf 10-poliger Stiftleiste, siehe Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86
  - Schrittmotorsignale für 2 Motoren auf 10-poliger Stiftleiste, siehe Kapitel 4.6.7 "STEPPER MOTOR-Stiftleiste", Seite 90
  - McBSP und ANALOG IN auf 10-poliger Stiftleiste, siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87
- Für die RTC6 PCIe-Karte gibt es keine IO-Erweiterungskarte mehr. Die Stiftleiste zum Anschließen einer Erweiterungskarte ist nicht mehr vorhanden (auf der RTC4 heißt diese Stiftleiste "EXTENSION 2"; die EXTENSION 2-Stiftleiste auf der RTC6 PCIe-Karte entspricht der "LASER EXTENSION" Stiftleiste auf der RTC4).

Allgemeine Programmierung

- Utility-Dateien für C, C++, C# und Delphi, siehe Kapitel 6.2.2 "Befehle importieren", Seite 98, aber keine Utility-Dateien mehr für Basic
- Befehle zur Änderung der Zugriffsberechtigungen für RTC6 PCIe-Karten, siehe Kapitel 6.7 "RTC6 PCIe-Karten durch mehrere Anwenderprogramme verwenden", Seite 131
- Verbessertes und erweitertes Listen-Handling, siehe Kapitel 6.4 "Listen-Handling", Seite 108
  - RTC6-Listenspeicher mit 8.388.608 Speicherplätzen
  - RTC6-Listenspeicher frei konfigurierbar (in 2 RTC6-Listenspeicher-Bereiche und 1 geschützten RTC6-Listenspeicher-Bereich)
  - Definition von geschützten Unterprogrammen
  - Laden von Listen mit Schutzfunktion
  - Schleifen in Listen und Unterprogrammen
  - Kein RTC4-Ringpuffer-Modus mehr (kann aber alternativ realisiert werden)
  - Listenstatus
  - Listenausführungsstatus
- "Kurze" Listenbefehle (z.B. zur Änderung von Geschwindigkeit, Analog-Ausgang, IO-Port etc.) können ohne Zeitverlust ausgeführt werden (es können mehrere "kurze" Listenbefehle innerhalb eines 10 μs-Takts ausgeführt werden, siehe Abschnitt "Normaler Listenbefehl, Kurzer Listenbefehl, Variabler Listenbefehl und Multipler Listenbefehl", Seite 339
- Funktionen zur Fehlerbehandlung und Download-Überprüfung, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133

Laser-Markieren

- Vektoren und Bögen
  - Befehle zur Markierung von Ellipsen, siehe Abschnitt "Ellipse-Befehle", Seite 141
  - Befehle zur Markierung von Helices, siehe Kapitel "3D-Befehle", Seite 250
  - Zeitbemessene Bogen- und 3D-Vektor-Befehle, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286
  - Para-Markier-Befehle und Para-Sprung-Befehle zur Vektorkontrollierte Laser-Steuerung
- Zeichen und Texte
  - Definition von Zeichensätzen und Zeichenfolgen, siehe Abschnitt "Indizierte Zeichensätze definieren", Seite 121
  - Befehle zum Markieren einzelner Zeichen und von Texten (mit auswählbarem Zeichensatz), siehe Abschnitt "Indizierte Zeichen aufrufen", Seite 122
  - Funktionen zur Markierung von Uhrzeit, Datum und Seriennummern (mit auswählbarem Zeichensatz und mit auswählbarem Seriennummernsatz), siehe Kapitel 7.5 "Uhrzeit, Datum und Seriennummern markieren", Seite 222
- micro_vector[*]-Befehle (direkte Positionsausgaben ohne Zerlegung in Mikroschritte), siehe Kapitel 8.8 "micro_vector[*]-Befehle", Seite 285

Spezielle Funktionen

- Synchronisation der Steuerung von Scan-System und Laser
  - Sky-Writing, siehe Kapitel 7.2.4 "Sky-Writing", Seite 163
- Pixelausgabemodus (Markieren von Bitmaps), siehe Kapitel 8.7 "Pixelausgabemodus", Seite 275:
  - Pixelausgabefrequenzen bis 800 kHz, darüber hinaus mit der Option "UFPM" bis 3,2 MHz
  - RTC4-Pixelmode 0 mit 10 μs-Takt-synchroner Ausgabe wird nicht mehr unterstützt
  - 15 ns zeitliche Auflösung
  - 0…100% Laser-Pulslänge
  - kein Pixel-Mode 0 mehr
  - kein Einscannen von Analogspannungen mehr
  - 3D-Pixelzeilen mit **set_pixel_line_3d**
- Befehle zum Bedingen beliebiger Listenbefehle, siehe Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320
- Wobbelbewegungen nicht nur kreisförmig, sondern auch ellipsenförmig, mit stehender oder liegender 8 oder mit "Frei definierbaren Wobbelfiguren" möglich. Die Orientierung kann wahlweise fest im Raum, automatisch an die momentane Bewegungsrichtung oder an eine beliebige andere Bewegungsrichtung angepasst sein. Bei "Frei definierbare Wobbelfiguren" kann auch die Laserleistung variiert werden, siehe Kapitel 8.4 "Wobbelmode", Seite 244
- Camming-Funktionalität, siehe Kapitel 8.11 "Camming", Seite 291
- Erweiterte Möglichkeiten zur Signalaufzeichnung, siehe **set_trigger[*]**
- "Processing on the fly", siehe Kapitel 8.6 ""Processing on the fly"", Seite 254
  - 2 Encodereingänge (RS-422) mit 32 Bit-Zähler für eine "Processing on the fly"-Korrektur mit Encodersignalen auf 2 Achsen, siehe Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323); alternativ: Korrektur mit McBSP-Signalen, siehe Kapitel 9.3.4 "Synchronisation und Online-Positionierung mit McBSP-Signalen", Seite 325
  - 29-Bit-Koordinaten (virtuelles Bildfeld): Objekte größer als reales Bildfeld möglich, siehe Kapitel 7.3.3 "Virtuelles Bildfeld", Seite 177, und Kapitel 8.6.6 "Virtuelles Bildfeld mit "Processing on the fly"", Seite 264
  - Bis zu 8 Objekte innerhalb der "Processing on the fly"-Streckenverzögerung, siehe Abschnitt "Externer Start", Seite 315
  - Exakter "Externer Start": Der Encoderzähler kann – wenn mit **set_control_mode** entsprechend eingestellt – zur Synchronisation eines "Processing on the fly"-Prozesses von externen Startsignalen zurückgesetzt werden. Der Reset erfolgt jeweils exakt (ohne 10 μs-Jitter) mit dem externen Startsignal.
  - Kompensation von 2D-Bewegungen (xy-Verfahrtisch)
  - Encodergesteuerte "Processing on the fly"-Korrektur für die z-Achse ("FlyZ-Korrektur"), siehe Kapitel 8.6.11 ""Processing on the fly"-Korrektur für die z-Achse", Seite 269

Achtung!

- Die Encoderzähler-Richtung von RTC4-Karten (also auch RTC4 Ethernet-Karte und RTC4 *SCAN*alone-Karte) ist entgegengesetzt zu der von RTC5-Karten und RTC6-Karten.
  Beim einem Wechsel der RTC-Karte müssen Sie deshalb entweder die Verkabelung oder Ihr Anwenderprogramm entsprechend anpassen.

2.11 Hinweise an RTC5-Benutzer

RTC6 PCIe-Karten können ansteuern:

- Scan-Systeme mit SL2-100-Schnittstelle
- Scan-Systeme mit XY2-100-Schnittstelle (in Verbindung mit dem XY2-100-Konverter (Zubehör))

Es folgt eine Übersicht über die wichtigsten Änderungen, die bei RTC6 PCIe-Karten im Vergleich zu RTC5-Karten eingeführt werden.

Unter anderem wird im Folgenden eine mögliche Vorgehensweise zum Überarbeiten von RTC5-Programmcode für die Verwendung mit RTC6 PCIe-Karten beschrieben.

Hinweise

- Siehe auch Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten".

2.11.1 RTC6 PCIe-Karte und RTC5-Karten im Vergleich

Die RTC6 PCIe-Karte hat im Vergleich zur RTC5-Kartenfamilie neuere, leistungsstärkere Hardware-Komponenten. Das betrifft vor allem den DSP, das FPGA und den Hauptspeicher.

Für bestehende Anwenderprogramme sind nur wenige Anpassungen erforderlich, siehe Kapitel 2.12.2 "RTC5-Quellcode für die RTC6 PCIe-Karte anpassen", Seite 62.

Funktionalitätsänderungen

- Kurze Vektoren(1) führt die RTC6 PCIe-Karte schneller aus als RTC5-Karten, siehe **set_dsp_mode**.
- Der RTC6-Listenspeicher fasst insgesamt 8.388.608 (= 2^{23}) Listenplätze.
  Liste 1 und Liste 2 werden mit je 4.194.304 (= 2^{22}) Listenplätzen initialisiert.
- Der Pixelausgabemodus ist erheblich erweitert, siehe **set_pixel_line**.
  - RTC6 PCIe-Karten ohne Option "UFPM" können damit "out-of-the-box" Pixelausgabefrequenzen von 800 kHz erzielen.
  - RTC6 PCIe-Karten mit Option "UFPM", sogar bis zu 3,2 MHz.
  - Mit **set_pixel_line** ist es nun möglich, die Laserleistungs-Steuerwerte digital auszugeben. Das erfolgt wahlweise am 8-Bit-Digital-Ausgang (EXTENSION 2-Stiftleiste) oder am 16-Bit-Digital-Ausgang (EXTENSION 1-Stiftleiste).
- Erweiterte Messwert-Aufzeichnung mit der RTC6 PCIe-Karte: 4 (8) Aufzeichnungskanäle für je 8.388.608 (4.194.304) Datenwerte, siehe **set_trigger4** (**set_trigger8**).
  Es können praktisch beliebige Datenmengen aufgezeichnet werden (Ringspeicher).

(1) "Strichlinien"

- 8 3D-Korrekturtabellen können auf der RTC6 PCIe-Karte gleichzeitig mit 8 Aufzeichnungskanälen (siehe **set_trigger[*]**) und dem gesamten RTC6-Listenspeicher verwendet werden. Mit **number_of_correction_tables** kann eine anwenderseitige Begrenzung auf weniger als 8 Korrekturtabellen zur Prüfung auf fehlerhafte Benutzereingaben eingestellt werden.
- Das virtuelle Bildfeld der RTC6 PCIe-Karte ist nun ±28 Bit = 29 Bit.
- Für 3D-Systeme mit unterschiedlichem Schleppverzugsverhalten von xy-Achsen und z-Achse kann eine entsprechende Schleppverzugskompensation eingestellt werden, siehe **set_timelag_compensation**.
- Die Ausgabe-Synchronisation der RTC5 wird bei der RTC6 PCIe-Karte durch eine Takt-Synchronisation ersetzt, siehe Kapitel 7.4.10 "Synchronisierung von RTC6-Takt und externem Taktsignal", Seite 219.

Parameteränderungen

- Die Laser-Delays (Parameter `LaserOnDelay` und `LaserOffDelay` von **set_laser_delays**) sowie `LaserOnShift` bei Sky-Writing-Befehlen haben bei der RTC6 PCIe-Karte eine feinere Auflösung von 1/64 μs.
  Bezüglich der Abwärtskompatibilität zur RTC5-Karte siehe Schritt 3, Seite 63.
- Der Parameter `Timelag` von **set_sky_writing_para** wird zwar im 64-Bit-IEEE Gleitkommaformat ("double") angegeben (1,0 = 1 μs), wird aber jetzt auf der RTC6 PCIe-Karte tatsächlich mit 1/64 μs Auflösung ausgeführt.

2.11.2 RTC6 PCIe-Karten und SCANahead-Systeme

- Siehe Kapitel 8.13 "SCANahead-Funktionalität", Seite 294.
  - Nur die RTC6 PCIe-Karte ermöglicht die Nutzung der SCANahead-Regelung (z.B. excelli*SCAN*-Baureihe).
  - Wird die RTC6 PCIe-Karte mittels **set_scanahead_params** für die Benutzung mit einem SCANahead-System konfiguriert, dann wird bei der Laser-Steuerung automatisch die `PreviewTime` der SCANahead-Regelung berücksichtigt. Dynamik und Genauigkeitsvorteile werden so optimal genutzt.
  - Mit **activate_scanahead_autodelays** kann das SCANahead-System einfach und schnell zum Einsatz gebracht werden. Weder Laser-Delays noch Scanner-Delays müssen selbst ermittelt oder gesetzt werden.
- Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten".

2.11.3 Einschränkungen bei RTC6 PCIe-Karten

Verfügbarkeit technischer Varianten

RTC6-Karten sind als RTC6 PCIe-Karten und RTC6 Ethernet-Karten lieferbar.

Schnittstellen zur Peripherie

Die ADC-Aufsteckplatine (#0121126, optionales Zubehör für die RTC5 PCI-Karte) ist mit der RTC6 PCIe-Karte nicht verwendbar.

Die RTC6 PCIe-Karte stellt die zwei 10 V-Analog-Eingänge ANALOG IN0 und ANALOG IN1 an der McBSP/ANALOG-Stiftleiste, Seite 87 direkt bereit.

Funktionalität

Bei RTC6 PCIe-Karten kann die McBSP/ANALOG-Stiftleiste, Seite 87 nicht im SPI-Modus betrieben werden.

Bekannte Einschränkungen sind in `RTC6_Release_Notes.pdf` genannt.

2.12 Quellcode für RTC6-Anwenderprogramme

2.12.1 RTC6-Quellcode neu erstellen

Quellcode für RTC6-Anwenderprogramme wird neu erstellt wie in Kapitel 6.2 "Initialisierung und Programmstart", Seite 98 beschrieben.

Dabei ist auch Kapitel 2.12.2 "RTC5-Quellcode für die RTC6 PCIe-Karte anpassen", Seite 62 zu beachten.

2.12.2 RTC5-Quellcode für die RTC6 PCIe-Karte anpassen

Anwenderprogramme, die für die RTC5 geschrieben wurden, können für die RTC6 PCIe-Karte nur dann verwendet werden, wenn der Programmcode überarbeitet wird, wie im Folgenden beschrieben.

Dies gilt auch dann, wenn der eigentliche Ablauf des Programms unverändert bleibt.

Zwingend erforderliche Schritte

(1) Stellen Sie sicher, dass folgende Voraussetzungen erfüllt sind:

- RTC6 PCIe-Karte ist eingebaut
- Der RTC6-Kartentreiber für das Betriebssystem ist installiert
- Die folgenden Dateien sind vorhanden:
  `RTC6DLL.dll` oder `RTC6DLLx64.dll`,
  `RTC6OUT.out` (bei RTC6 PCIe-Karten) oder
  `RTC6ETH.out` (bei RTC6 Ethernet-Karten),
  `RTC6RBF.rbf`,
  `RTC6DAT.dat`,
  `*.CT5` (Korrekturdatei/en)
  Hinweis: `RTC6OUT.out`/`RTC6ETH.out`, `RTC6RBF.rbf`, `RTC6DAT.dat` müssen alle im gleichen Verzeichnis sein.

– Die Importdeklarationen wurden in die Software importiert und es wird als Aufrufkonvention `std_call` verwendet, siehe Kapitel 6.2 “Initialisierung und Programmstart”, Seite 98.

(2) Berücksichtigen Sie die Befehlsnamensanpassungen des RTC6-Befehlssatzes.
– Ändern Sie *nicht* pauschal `rtc5_` zu `_rtc6_`!
– Ändern Sie:
`init_rtc5_dll` zu **`init_rtc6_dll`**
`free_rtc5_dll` zu **`free_rtc6_dll`**
`rtc5_count_cards` zu **`rtc6_count_cards`**
– Achtung: Ändern Sie *nicht* pauschal **`set_rtc5_mode`** zu **`set_rtc6_mode`**, siehe dazu Schritt 3)!

(3) **set_rtc6_mode** wird bei Programmstart automatisch eingestellt. Damit steht eine erweiterte Parameterauflösung zur Verfügung für:
– z-Koordinaten von 20 Bit
– Laser-Delays von 1/64 µs
Im Gegensatz dazu gewährleistet der Kontrollbefehl **set_rtc5_mode** die Abwärtskompatibilität der Parameter (“RTC5-Kompatibilitätsmodus”).
– Laser-Delays:
Im RTC5-Kompatibilitätsmodus multipliziert die RTC6-Karte die angegebenen Werte für `LaserOnDelay` und `LaserOffDelay` (von **set_laser_delays**) automatisch mit 32. Das gilt auch für die angegebenen Werte für `LaserOnShift` (von **set_sky_writing**) und `LaserOnShift` (von **set_sky_writing_para**).
– z-Koordinaten:
Im RTC5-Kompatibilitätsmodus multipliziert die RTC6-Karte die z-Koordinaten und Defocus-Werte (z.B. `Shift` von **set_defocus**) automatisch mit 16.

Zu weiteren Anpassungen und Optimierungen an Ihrem Code, siehe folgendes Kapitel.

Optionale Schritte und Hinweise zu weiteren Anpassungen und Optimierungen

- Aus dem RTC6-Befehlssatz entfernte und nicht mehr unterstützte Befehle müssen aus dem Programmcode entfernt oder ersetzt werden, siehe Kapitel 2.13.2 “Nicht mehr vorhanden im RTC6-Befehlssatz”, Seite 64.
- Prüfen Sie, ob die Änderungen an den Befehlen für Ihr Anwenderprogramm relevant sind, siehe Kapitel 2.13.1 “Geändert im RTC6-Befehlssatz”, Seite 64, und passen Ihren Programmcode entsprechend an.
- Beachten Sie das andere Verhalten von **load_program_file**.
Achtung: **load_program_file** hält jetzt ohne Warnung die Liste an. Der Single-Board-Befehl kann auch nach einem Versionskonflikt erneut verwendet werden.

– **Hinweis zur erhöhten Parameterauflösung**
Beachten Sie die erweiterte Parameterauflösung von z-Koordinaten, wenn **set_rtc6_mode** verwendet wird. Siehe auch Schritt 3, Seite 63.
– **Hinweis zu verändertem Zeitverhalten**
Durch den Ersatz der automatischen Delay-Anpassung kann sich für kurze Vektoren ein schnelleres Zeitverhalten ergeben.
Mit `set_dsp_mode( 2 )` kann das ursprüngliche Zeitverhalten der RTC5 wieder hergestellt werden, siehe **set_dsp_mode**.

2.13 Änderungen am RTC6-Befehlssatz

Die meisten RTC5-Befehle sind im RTC6-Befehlssatz enthalten. Auf die wenigen Ausnahmen wird in den folgenden Kapiteln eingegangen:

- *Geänderte* Befehle, siehe Kapitel 2.13.1 "Geändert im RTC6-Befehlssatz", Seite 64
- *Entfernte* Befehle, siehe Kapitel 2.13.2 "Nicht mehr vorhanden im RTC6-Befehlssatz", Seite 64

2.13.1 Geändert im RTC6-Befehlssatz

Bedeutungen:

con	Kontrollbefehl
nor	Normaler Listenbefehl
vk	Verzögerter Kurzer Listenbefehl

(n_) config_list con 375
get_dll_version con 438
(n_) get_hex_version con 451
(n_) get_rtc_version con 467
(n_) get_status con 475
(n_) get_sync_status con 479
(n_) *init_fly_2d con 515
(n_) load_correction_file con 550
(n_) *load_fly_2d_table con 557
(n_) load_program_file con 566
(n_) *load_stretch_table con 569
(n_) mcbsp_init_spi con 603
(n_) number_of_correction_tables con 612
(n_) select_cor_table con 664
(n_) *set_auto_laser_control con 674
(n_) *set_auto_laser_params con 679
(n_) *set_auto_laser_params_list vk 680
(n_) set_dsp_mode con 697
(n_) *set_pixel_line nor 792
(n_) set_trigger4 nor 852

* Hinweis: RTC6-Befehl mit Funktionalitätserweiterung.

Ohne Funktion (derzeit)

(n_) set_softstart_level 831
(n_) set_softstart_level_list 831
(n_) set_softstart_mode 832
(n_) set_softstart_mode_list 832

2.13.2 Nicht mehr vorhanden im RTC6-Befehlssatz

free_rtc5_dll 966
init_rtc5_dll 966
RTC5-Kontrollbefehl 966

3 Sicherheit bei Installation und Inbetriebnahme

Lesen Sie diese Betriebsanweisungen vollständig, bevor Sie mit der Installation und Inbetriebnahme der RTC6 PCIe-Karte beginnen.

Falls Fragen bezüglich des Inhalts dieses Handbuchs offen bleiben sollten, dann wenden Sie sich an SCANLAB.

Für Sicherheitshinweise in diesem Handbuch gelten die folgenden Konventionen:



- Sicherheitshinweise, die auf schwere Verletzung oder sogar Tod aufmerksam machen, sind durch das Gefahrenzeichen und das Signalwort "Warnung!" gekennzeichnet.



- Sicherheitshinweise, die auf eine Gesundheitsgefahr aufmerksam machen, sind durch das Gefahrenzeichen und das Signalwort "Vorsicht!" gekennzeichnet.
- Sicherheitshinweise, die eine sachgemäße Verwendung des Gerätes empfehlen oder vor möglichen Sachbeschädigungen warnen, sind (ohne Gefahrenzeichen) nur mit dem Signalwort "Achtung!" gekennzeichnet.

3.1 Betriebssicherheit

Achtung!

- Prüfen Sie Ihr Anwenderprogramm vor dem Einsatz sorgfältig. Programmierfehler können zum Absturz des Systems führen. Dadurch ginge vorübergehend sowohl die Kontrolle über den Laser als auch über die Bewegungen des Scan-Systems verloren.
- Schützen Sie die Karte vor Feuchtigkeit, Staub, korrosiven Dämpfen und mechanischen Belastungen.

Achtung!

- Vermeiden Sie bei Lagerung und Betrieb der Karte elektromagnetische Felder und statische Elektrizität. Die Elektronik der Karte kann durch solche Felder zerstört werden. Lagern Sie die Karte immer im mitgelieferten antistatischen Beutel.
- Die zulässige Betriebstemperatur beträgt 15 °C bis 60 °C.
- Die Karte muss bei einer Temperatur zwischen –20 °C und +60 °C gelagert werden.

3.2 Lasersicherheit

Die RTC6 PCIe-Karte wird zum Steuern von Scan-Systemen und Lasern verwendet. Deshalb müssen vor der Installation und Inbetriebnahme alle Vorschriften zur Lasersicherheit bekannt sein und beachtet werden. Der Kunde trägt die alleinige Verantwortung dafür, dass die Lasersicherheit im gesamten System gewährleistet ist.

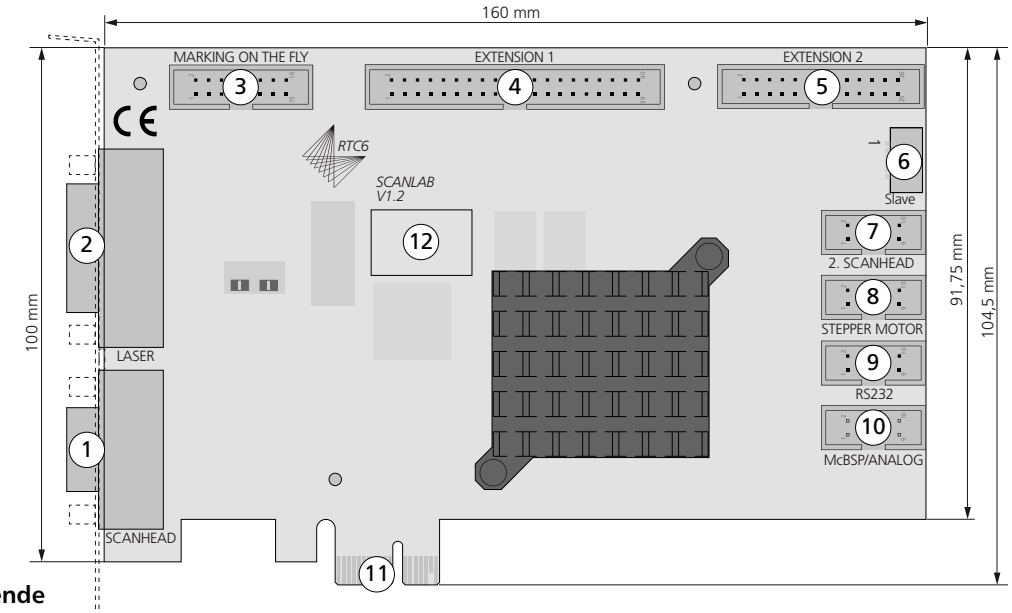


Vorsicht!

- Sicherheitsvorschriften können von Land zu Land variieren. Die entsprechenden Vorschriften müssen bekannt sein und angewandt werden. Es liegt in der Verantwortung des Kunden, die lokalen Vorschriften zu erfüllen.
- Beachten Sie unbedingt die Hinweise bezüglich der Lasersicherheit, die in der Bedienungsanleitung zum Scan-System im Kapitel "Sicherheit bei Installation und Inbetriebnahme" aufgeführt sind.
- *Schalten Sie immer zuerst den PC und erst dann den Laser an. Anderenfalls besteht die Gefahr, dass der Laserstrahl in eine ungewünschte Richtung abgelenkt wird.*
  SCANLAB empfiehlt den Gebrauch eines Shutters, um die unkontrollierte Emission von Laserstrahlung zu vermeiden.

4 RTC6 PCIe-Karte – Layout und Schnittstellen

4.1 Layout – Oberseite



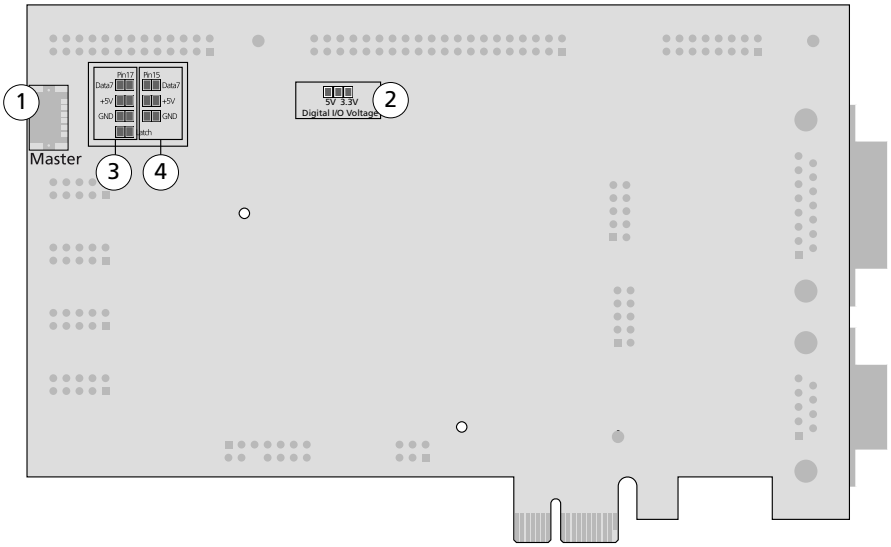
Legende

1. SCANHEAD . . . . . 9-polige Sub-D-Buchse, weiblich. Anschluss für ersten Scan-Kopf. Pinbelegung siehe Abbildung 10.
2. LASER . . . . . 15-polige Sub-D-Buchse, weiblich. Anschluss für den Laser, inkl. digitaler und analoger Ausgänge. Pinbelegung siehe Abbildung 17.
3. MARKING ON THE FLY . . 16-polige Stiftleiste. Mit Encodereingängen für "Processing on the fly"-Anwendungen. Pinbelegung siehe Abbildung 25.
4. EXTENSION 1 . . . . . 40-polige Stiftleiste. Mit einem 16-Bit-Digital-Eingang und einem 16-Bit-Digital-Ausgang. Pinbelegung siehe Abbildung 22.
5. EXTENSION 2 . . . . . 26-polige Stiftleiste. Mit einem 8-Bit-Digital-Ausgang, z.B. für eine alternative Laseransteuerung. Pinbelegung siehe Abbildung 24.
6. Slave . . . . . 6-polige Stiftleiste. Zum Verbinden mit einer anderen RTC6 PCIe-Karte zwecks Taktsynchronisierung. Details siehe "Master-Stiftleiste, Slave-Stiftleiste", Seite 68.
7. 2. SCANHEAD . . . . . 10-polige Stiftleiste. Anschluss für zweiten Scan-Kopf. Pinbelegung siehe Abbildung 11.
8. STEPPER MOTOR . . . . . 10-polige Stiftleiste. Zum Ansteuern von 1…2 Schrittmotoren. Pinbelegung siehe Abbildung 30.
9. RS232 . . . . . 10-polige Stiftleiste. Zur Ansteuerung eines externen Geräts über RS-232-Schnittstelle. Pinbelegung siehe Abbildung 27.
10. McBSP/ANALOG . . . . . 10-polige Stiftleiste. Serielle Erweiterungsschnittstelle. Pinbelegung siehe Abbildung 28.
11. PCIe-x-1-Stecker . . . . . Schnittstelle zum PC. Siehe Kapitel 4.3 "Schnittstelle zum PC", Seite 68.
12. . . . . Aufkleber mit Seriennummer.

5

RTC6 PCIe-Karte: Oberseite.

4.2 Layout – Unterside



Legende

1. Master . . . . . 6-polige Stiftleiste. Zum Verbinden mit einer anderen RTC6 PCIe-Karte zwecks Taktsynchronisierung. Details siehe "Master-Stiftleiste, Slave-Stiftleiste", Seite 68.
2. Lötjumperfeld A. . . . . Zum Konfigurieren des Pegels der Ausgangssignale an der EXTENSION 1-Stiftleiste. Details siehe "Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste", Seite 42.
3. Lötjumperfeld B . . . . . Zum Konfigurieren des Signals an EXTENSION 2-Stiftleiste Pin (17). Details siehe "Lötjumperfeld B – Signal an EXTENSION 2-Stiftleiste Pin (17)", Seite 43.
4. Lötjumperfeld C. . . . . Zum Konfigurieren des Signals an EXTENSION 2-Stiftleiste Pin (15). Details siehe "Lötjumperfeld C – Signal an EXTENSION 2-Stiftleiste Pin (15)", Seite 44.

6

RTC6 PCIe-Karte: Unterseite.

4.3 Schnittstelle zum PC

Schnittstelle zum PC ist der PCIe-x-1-Stecker der RTC6 PCIe-Karte, siehe Abbildung 5.

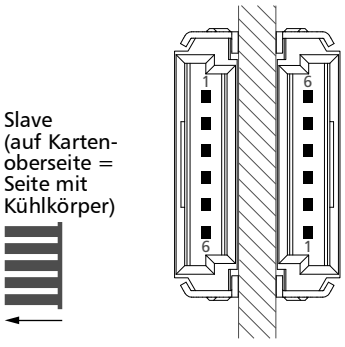
Die RTC6 PCIe-Karte kann in jeden Windows-PC mit PCI-Express-Bus und mindestens einem freien PCI-Express-Steckplatz installiert werden.

Achtung!

- Die RTC6 PCIe-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCIe-Bus abgeschaltet werden. Deaktivieren Sie entsprechende Standby- oder Sleep-Modi des Betriebssystems.
  Siehe auch Abschnitt "Hinweise", Seite 38.

4.4 Master-Stiftleiste, Slave-Stiftleiste

Sowohl die Slave-Stiftleiste als auch die Master-Stiftleiste ist 6-polig, siehe Abbildung 7. Die Slave-Stiftleiste befindet sich auf der Oberseite der RTC6 PCIe-Karte, siehe Abbildung 5, die Master-Stiftleiste auf der Unterseite, siehe Abbildung 6(1).



Hersteller: ERNI.
Artikelnummer: 214013 (2×).

7

Slave-Stiftleiste und Master-Stiftleiste.
Das Rastermaß der Pins ist 1,27 mm.

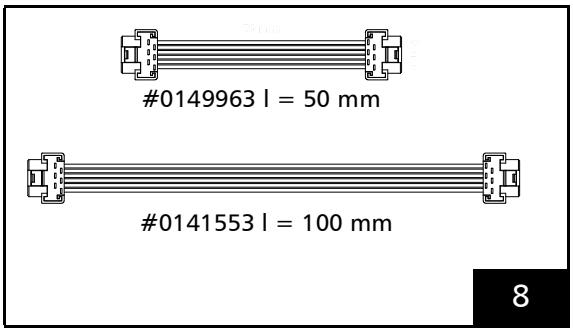
(1) Ist bei RTC6 Ethernet-Karten umgekehrt.

Zweck der beiden Stiftleisten ist, eine Taktsynchronisation mehrerer RTC6 PCIe-Karten zu ermöglichen. Dann müssen die RTC6 PCIe-Karten paarweise über die Master- und Slave-Stiftleisten miteinander verbunden werden. Verbinden Sie jeweils die Master-Stiftleiste einer Karte mit der Slave-Stiftleiste einer anderen Karte mit einem geeigneten Kabel (bei SCANLAB erhältlich, siehe Abbildung 8). Die notwendigen Angaben zum Konfektionieren eigener Kabel zeigt Abbildung 9.

Miteinander verbundene RTC6 PCIe-Karten sollten (empfohlen) in benachbarte PCIe-Steckplätze gesteckt sein.

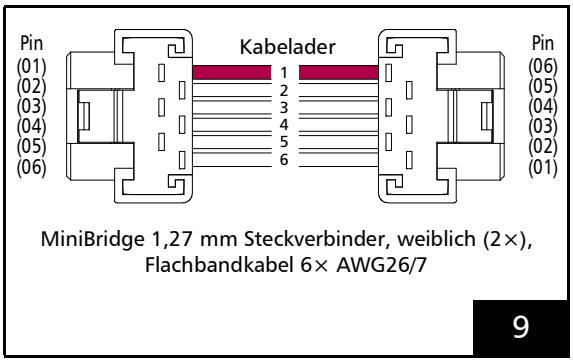
Wichtig: Um die Master/Slave-Funktionalität verwenden zu können, siehe Voraussetzungen in Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.

Siehe auch Kapitel 9.3.1 "Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden", Seite 314.



8

SCANLAB Master/Slave-Verbindungskabel.



MiniBridge 1,27 mm Steckverbinder, weiblich (2×),
Flachbandkabel 6× AWG26/7

9

Kabel zum Verbinden von Master-Stiftleiste und Slave-Stiftleiste: Anforderungen. Länge möglichst kurz halten!

Hinweise

- Mit **master_slave_config** kann für jede RTC6-Karte individuell deren Master/Slave-Schnittstellen-Eigenschaften konfiguriert werden.

4.5 Schnittstellen zum Scan-System

4.5.1 Scan-Kopf-Anschlüsse und Übertragungsprotokoll

Zur digitalen Ansteuerung von Scan-Systemen stehen die SCANHEAD-Buchse und die optional aktivierte 2. SCANHEAD-Stiftleiste zur Verfügung.

Dort werden Ansteuerwerte für das Scan-System ausgegeben und Statussignale empfangen, die vom Scan-System zurückgegeben werden. An beiden Scan-Kopf-Anschlüssen werden jeweils Daten für maximal zwei Achsen übertragen. Welche Statussignale vom Scan-System generiert werden, und wie sie zur Überwachung des Scan-Systems eingesetzt werden können, entnehmen Sie dem Handbuch Ihres Scan-Systems.

Die Datenübertragung zwischen RTC6 PCIe-Karte und Scan-System erfolgt in beide Richtungen gemäß dem SL2-100-Protokoll. Zur Konvertierung der Signale für eine Datenübertragung gemäß dem XY2-100-Protokoll ist der XY2-100-Konverter (Zubehör) erhältlich.

Wenn weder die Option "Zweite-Scan-Kopf-Steuerung" noch die Option "3D" freigeschaltet ist, werden nur am Anschluss für ersten Scan-Kopf Signale für ein xy-Scan-System ausgegeben.

Ist die Option "Zweite-Scan-Kopf-Steuerung" freigeschaltet, können mit einer RTC6 PCIe-Karte zwei xy-Scan-Systeme gleichzeitig angesteuert werden.

Ist die Option "3D" freigeschaltet, dann kann über Anschluss für ersten Scan-Kopf und Anschluss für zweiten Scan-Kopf ein 3-Achsen-Scan-System angesteuert werden, wenn dem Anschluss für ersten Scan-Kopf eine 3D-Korrekturtabelle zugewiesen wird. Am Anschluss für ersten Scan-Kopf werden dann die Signale für den xy-Scan-Kopf und über beide Kanäle des Anschluss für zweiten Scan-Kopf die Signale für die 3. Achse (z-Achse) ausgegeben.

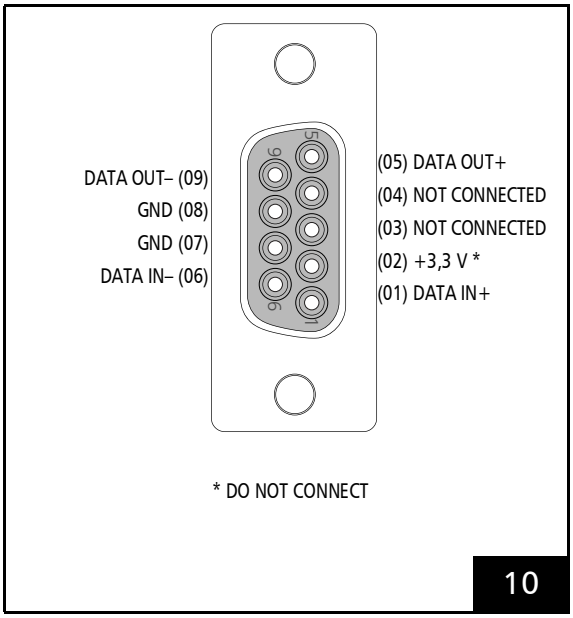
Sind *beide* Optionen (Option "Zweite-Scan-Kopf-Steuerung" *und* Option "3D") freigeschaltet, entscheidet die Zuweisung der Korrekturtabellen darüber, welche Signale (xy oder z) an welchem Anschluss ausgegeben werden, siehe auch Abschnitt "2D- und 3D-Korrekturdateien", Seite 182.

Wenn mehrere RTC6 PCIe-Karten mit freigeschalteter Option "3D" in einen PC eingebaut werden, dann können auch entsprechend viele 3-Achsen-Systeme gleichzeitig angesteuert werden.

SCANHEAD-Buchse

Anschluss für ersten Scan-Kopf

Die Pinbelegung der SCANHEAD-Buchse (9-polige Sub-D-Buchse, weiblich) zeigt Abbildung 10.



SCANHEAD-Buchse (9-polige Sub-D-Buchse, weiblich): Pinbelegung.

Am differentiellen DATA OUT-Ausgang werden die Ansteuerwerte für das Scan-System ausgegeben.

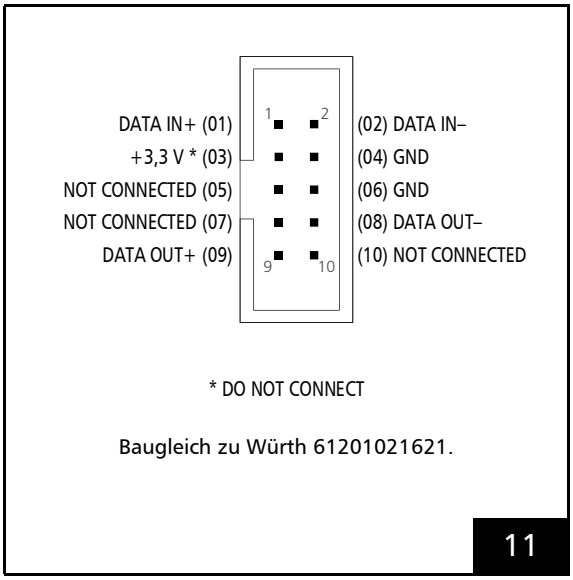
Am differentiellen DATA IN-Eingang werden die Statussignale empfangen, die vom Scan-System übertragen werden.

An Pin (02) wird 3,3 V als Betriebsspannung für den XY2-100-Konverter (Zubehör) bereitgestellt (oder für einen Polymer-Optical-Fiber-Konverter zur optischen Datenübertragung). Für andere Zwecke sollte diese Spannung nicht verwendet werden.

2. SCANHEAD-Stiftleiste

Anschluss für zweiten Scan-Kopf

Die Pinbelegung der 2. SCANHEAD-Stiftleiste (10-polige Stiftleiste) zeigt Abbildung 11.



2. SCANHEAD-Stiftleiste: Pinbelegung.
Das Rastermaß der Pins ist 2,54 mm. Signale für den 2. Scan-Kopf werden nur bei freigeschalteter Option "Zweite-Scan-Kopf-Steuerung" ausgegeben.

Slotblech für Zweiten Scan-Kopf (Zubehör)

Zum Anschließen eines zweiten Scan-Systems oder einer z-Achse an den 2. SCANHEAD-Stiftleiste empfiehlt SCANLAB den Gebrauch eines zusätzlichen Slotblechs, siehe Kapitel 2.8.4 "Slotblech mit 9-poliger Sub-D-Buchse für die 2. SCANHEAD-Stiftleiste", Seite 46.

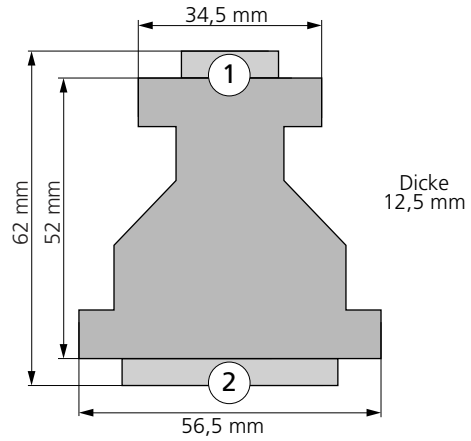
4.5.2 XY2-100-Konverter (Zubehör)

Der SCANLAB XY2-100-Konverter (Zubehör) (#0125377) konvertiert:

- Ansteuersignale gemäß SL2-100-Protokoll (20 Bit) in Ansteuersignale gemäß XY2-100-Protokoll (16 Bit)
- Scan-System-Statussignale gemäß XY2-100-Protokoll in Statussignale gemäß SL2-100-Protokoll, siehe auch Kapitel 7.3.7 "Statusüberwachung und Diagnose", Seite 191

Bei der Scan-System-Ansteuerung verursacht der XY2-100-Konverter (Zubehör) eine Signallaufzeitverzögerung von 10 µs. Um diese auszugleichen, muss das LaserOn-Delay und das LaserOff-Delay um jeweils 10 µs erhöht werden, siehe **set_laser_delays**.

Die Abmessungen zeigt Abbildung 12.



Legende
1. 9-poliger Sub-D-Stecker, männlich
2. 25-polige Sub-D-Buchse, weiblich

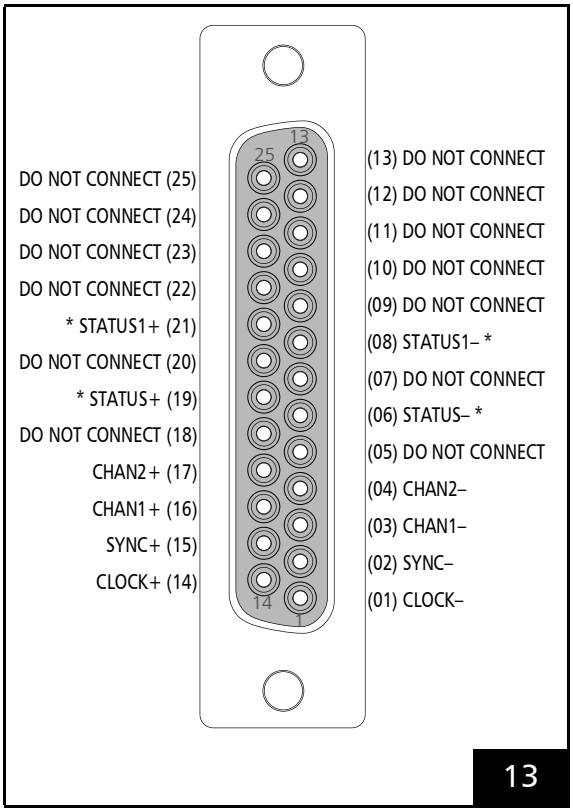
12

XY2-100-Konverter (Zubehör): Abmessungen.

Der 9-polige Sub-D-Stecker (männlich) des XY2-100-Konverter (Zubehör) sollte direkt in den SCANHEAD-Buchse der RTC6 PCIe-Karte eingesteckt oder – über ein möglichst kurzes 1:1-Kabel – mit den entsprechenden Pins der 2. SCANHEAD-Stiftleiste der RTC6 PCIe-Karte verbunden werden. Die Pinbelegung zeigt Abbildung 10 und Abbildung 11.

Die 25-polige Sub-D-Buchse (weiblich) des XY2-100-Konverter (Zubehör) ist mit Scan-Köpfen kompatibel, die mit einer XY2-100-Standard-Schnittstelle ausgestattet sind.

Die Pinbelegung zeigt Abbildung 13.



13

XY2-100-Konverter (Zubehör): Pinbelegung der 25-poligen Sub-D-Buchse (weiblich).

* Bei iDRIVE-Scan-Systeme ist der STATUS± Kanal der Statuskanal für Achse 2 (x-Achse; der Kanal wird dort auch mit STATUS2± bezeichnet) und der STATUS1± Kanal Statuskanal für Achse 1 (y-Achse).
Bei anderen Scan-Systemen ist der STATUS1± Kanal nicht nutzbar: "DO NOT CONNECT".

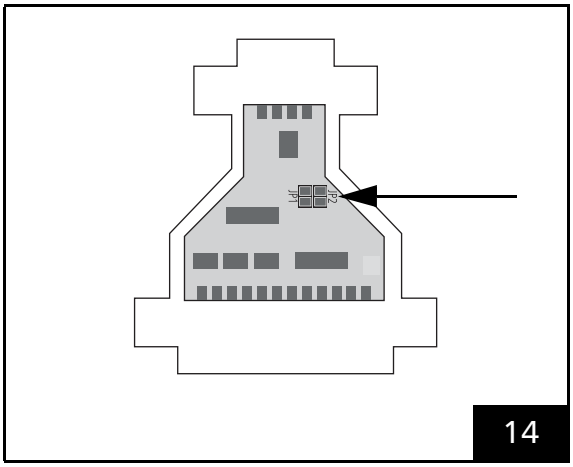
An den Datenkanälen CHAN1 und CHAN2 werden die Ansteuerwerte, an den Kanälen SYNC und CLOCK ein Synchronisations- und ein Taktsignal an das Scan-System ausgegeben.

Am STATUS-Kanal (und gegebenenfalls STATUS1-Kanal) werden die Statussignale empfangen, die vom Scan-System gemäß XY2-100-Standard zurückgegeben werden.

Falls zur Datenübertragung zwischen XY2-100-Konverter (Zubehör) und Scan-System Datenkabel über 20 m (nicht empfohlen) verwendet werden, dann sollte die Synchronisation der bidirektionalen Kommunikation zwischen Scan-Kopf und RTC6 PCIe-Karte angepasst werden.

Das kann über zwei Lötjumper im XY2-100-Konverter (Zubehör) bewerkstelligt werden.

Dazu muss das Gehäuse des XY2-100-Konverter (Zubehör) vorsichtig an seinen 4 Clip-Verriegelungen geöffnet(1) werden. Die Lötjumper **JP1** und **JP2** befinden sich auf der Platine, siehe Pfeil in Abbildung 14.



XY2-100-Konverter (Zubehör): Position der Lötjumper **JP1** und **JP2** auf der Platine.

Die folgende Tabelle zeigt die möglichen Jumpereinstellungen und die zugehörigen Kabellängen. Andere Jumpereinstellungen sind nicht zulässig. Kabellängen über 20 m werden nicht empfohlen.

Jumpereinstellung	**Kabellänge***
JP1 geschlossen **JP2** offen (Standard-Konfiguration) 	0 m bis 20 m
JP1 offen **JP2** geschlossen 	20 m bis 40 m

* Der hier genannte Kabellängenbereich für die jeweilige Jumperkonfiguration hängt u.a. vom verwendeten Kabeltyp ab und kann von den oben genannten Werten abweichen. Kabellängen über 20 m werden generell nicht empfohlen (und setzen umfassende Überprüfungen durch Benutzer vor einem Produktiveinsatz voraus).

Hinweise

- Beachten Sie den Hinweis zu **get_head_status**, Seite 225.

(1) Beispielsweise mittels Schlitzschraubendreher.

4.5.3 Datenkabel (Zubehör)

Zur Übertragung der Datensignale zwischen RTC6 PCIe-Karte (oder dem XY2-100-Konverter (Zubehör)) und Scan-System sind bei SCANLAB geeignete Datenkabel erhältlich. Datenkabel sind in der Regel nicht im Lieferumfang enthalten.

Für SCANLAB Scan-Systeme, die mit einer SL2-100-Schnittstelle (9-polige Sub-D-Buchse) ausgestattet sind, sind Datenkabel zur elektrischen Datenübertragung erhältlich, zur optischen Datenübertragung (nur auf Anfrage) auch Lichtwellenleiterkabel(1).

Für SCANLAB Scan-Systeme, die mit einer XY2-100-Schnittstelle (25-polige Sub-D-Buchse) ausgestattet sind, sind ausschließlich Datenkabel zur elektrischen Datenübertragung erhältlich.

Scan-Systeme, die mit einer optischen Datenschnittstelle (XY2-100-O) ausgestattet sind, können nicht mit RTC6 PCIe-Karten angesteuert werden.

SCANLAB empfiehlt für Eigenbauten die folgende Konfektionierung für die elektrische Datenübertragung:

- Für die Datenübertragung gemäß SL2-100-Protokoll, siehe Abbildung 15, sollte das Kabel an beiden Enden mit 9-poligen Sub-D-Steckern ausgestattet sein. Die zwei Kanäle DATA IN± und DATA OUT± sollten aus verdrillten Kabelpaaren bestehen und an den beiden Sub-D-Steckern *über Kreuz* angeschlossen werden (so dass etwa das DATA OUT-Signal der RTC6 PCIe-Karte am DATA IN-Eingang des Scan-Systems anliegt). Das Kabel sollte nicht länger als 25 m sein. SCANLAB empfiehlt eine Leitungsimpedanz von 110 Ω, unabhängig von der Kabellänge.
- Für die Datenübertragung gemäß XY2-100-Protokoll, siehe Abbildung 16, sollte das Kabel an beiden Enden mit 25-poligen Sub-D-Steckern (männlich) mit identischer Pinbelegung ausgestattet sein. Die fünf (oder sechs) Kanäle SYNC±, CHAN1±, CHAN2±, STATUS± (und STATUS1±) und CLOCK± sollten aus verdrillten Kabelpaaren bestehen. Zusammen mit einem XY2-100-Konverter (Zubehör) mit Standard-Jumper-Konfiguration (s.o.) sollte das Kabel nicht länger als 20 m sein. Falls ein längeres Datenkabel benötigt wird, dann muss die Lötjumpereinstellung des XY2-100-Konverter (Zubehör) angepasst werden.
- Für die Datenübertragung gemäß XY2-100-Protokoll muss am ansteuerseitigen Ende des Datenkabels ein Ferrit baugleich zu Würth WE 74271132 angebracht werden.

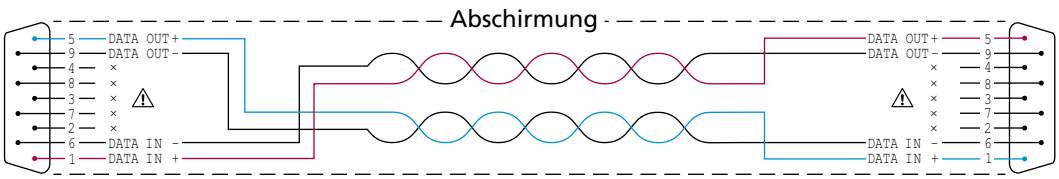
Unabhängig vom Datenübertragungsprotokoll gilt zusätzlich:

- Das Datenkabel muss eine koaxiale Abschirmung haben, die als Kupfergeflecht ausgeführt ist.
- Die Sub-D-Stecker müssen ein voll abgeschirmtes Metallgehäuse haben.
- Die Abschirmung des Kabels darf *nicht* als Drahtverbindung an das Gehäuse ausgeführt sein, sondern muss koaxial über geschirmte Griffschellen übertragen werden.

Einige Scan-Köpfe haben nur einen einzigen Anschluss, der sowohl die Betriebsspannung als auch die Datensignale bereitstellt. SCANLAB empfiehlt für solche Scan-Köpfe, die Verkabelung so auszuführen, dass die Daten in einem separaten Kabel übertragen werden. Dieses sollte so konfektioniert sein, wie oben beschrieben.

(1) Auch die Lichtwellenleiterkabel werden über 9-polige Sub-D-Stecker angeschlossen. Die optische Umsetzung (Polymer-Optical-Fiber-Konvertierung) für die optische Datenübertragung erfolgt in den Sub-D-Steckern. Die Betriebsspannung für die Polymer-Optical-Fiber-Konvertierung wird an den Scan-Kopf-Anschlüssen der RTC6 PCIe-Karte sowie an der digitalen Schnittstelle des Scan-Systems bereitgestellt.

Kabel für Datenübertragung gemäß SL2-100-Protokoll



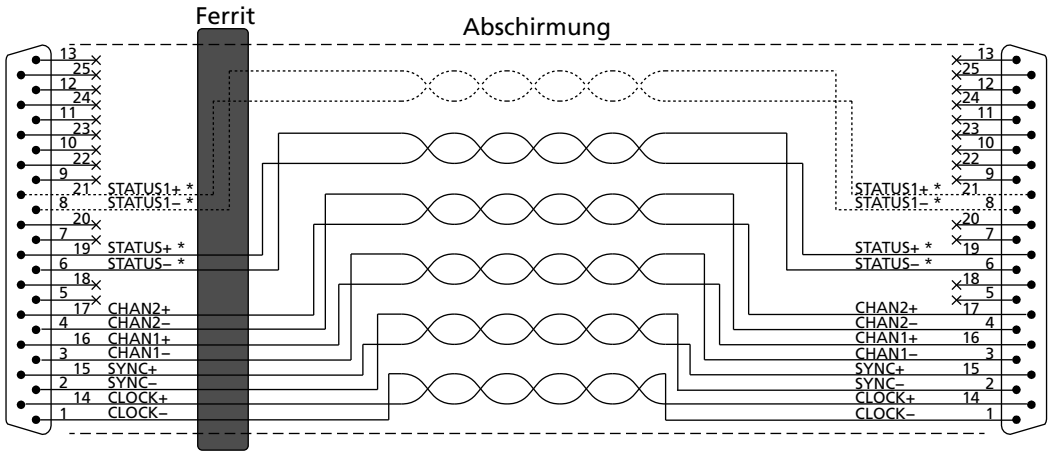
← RTC-Karte
9-poliger Sub-D Stecker, male

Scan-System →
9-poliger Sub-D Stecker, male

15

Kabel für Datenübertragung gemäß SL2-100-Protokoll:
Anforderungen und Pinzuordnungen.

Kabel für Datenübertragung gemäß XY2-100-Protokoll



Zur RTC-Karte
25-poliger Sub-D Stecker, male

Zum Scan-Kopf
25-poliger Sub-D Stecker, male

* Für iDRIVE-Scan-Systeme gilt:
Der STATUS± Kanal ist der Statuskanal für Achse 2 (x-Achse;
dieser Kanal wird dort auch mit STATUS2± bezeichnet).
Der STATUS1± Kanal ist der Statuskanal für Achse 1 (y-Achse).
Bei anderen Scan-Systemen wird der STATUS1± Kanal nicht benötigt.

16

Kabel für Datenübertragung gemäß XY2-100-Protokoll:
Anforderungen und Pinzuordnungen.

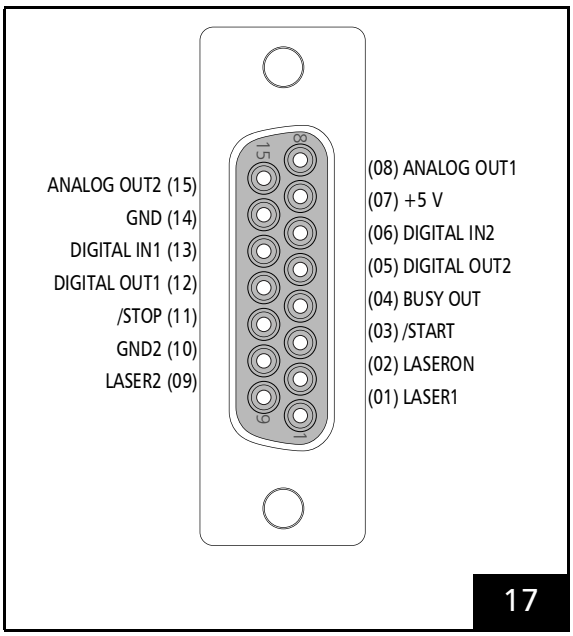
4.6 Schnittstellen zum Laser und zur Peripherie

4.6.1 LASER-Buchse

- Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", LASER-Buchse, Seite 974.

Die LASER-Buchse ist eine 15-polige Sub-D-Buchse (weiblich). Sie befindet sich auf dem Slotblech der RTC6 PCIe-Karte, siehe Abbildung 5.

Die Pinbelegung zeigt Abbildung 17.



LASER-Buchse (15-polige Sub-D-Buchse, weiblich): Pinbelegung. Bei RTC6 PCIe-Karten ohne Option "DC/DC-Wandler" sind GND2 und GND identisch.

Achtung!

- Bei RTC6 PCIe-Karten mit der Option "DC/DC-Wandler" sind GND2 und die Lasersteuersignale LASERON, LASER1 und LASER2 an der LASER-Buchse und an der EXTENSION 2-Stiftleiste galvanisch von GND entkoppelt.(a)
- Bei RTC6 PCIe-Karten ohne Option "DC/DC-Wandler" sind GND2 und GND identisch.

(a) RTC6 PCIe-Karten: GND ist die PC-Masse.
RTC6 Ethernet-Karten: GND ist die Masse am POWER-Stecker.

Achtung!

- Wenn Sie die RTC6 PCIe-Karte in Verbindung mit laserDESK verwenden wollen, beachten Sie folgendes:
  - laserDESK benutzt weitere Pins an der EXTENSION 1-Stiftleiste, um z.B. den Laser zu steuern.
  - Befassen Sie sich deshalb mit der erweiterten Dokumentation zum Laser-Anschluss. Diese finden Sie in der laserDESK-Online-Hilfe (alternativ bei SCANLAB erhältlich oder auch im laserDESK-zip, das Sie von der SCANLAB-Website herunterladen können).

Lasersteuersignale

Die Lasersteuersignale sind LASERON, LASER1 und LASER2. Es sind digitale TTL-Pegel-Signale und beziehen sich auf GND2.

Die Lasersteuersignale LASER1 und LASER2 sind je nach eingestelltem Laser-Modus unterschiedlich, siehe nachfolgende Tabelle und die Zeitdiagramme in Kapitel 7.4 "Laser-Steuerung", Seite 192.

	LASER1 Pin (01)	**LASER2 Pin (09)**
CO_2-Modus	Modulations-Signal 1, Standby-Signal	Modulations-Signal 2, Standby-Signal
YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5	Q-Switch-Signal	FirstPulseKiller-Signal
Laser-Mode 4	Standby-Signal	FirstPulseKiller-Signal
Laser-Mode 6	Standby-Signal	–

Alle Lasersteuersignale können mit **set_laser_control** entweder auf LOW-aktiv oder auf HIGH-aktiv gesetzt werden. "LOW-aktiv" bedeutet, dass eine logische 1 (z.B. "Laser On") durch einen niedrigen Pegel (0 V, TTL) repräsentiert wird. "HIGH-aktiv" bedeutet, dass eine logische 1 durch einen hohen Pegel (+5 V, TTL) repräsentiert wird. Wählen Sie den TTL-Lasersteuersignal-Pegel gemäß der Spezifikation Ihrer Laser-Steuerung. Beachten Sie die Dokumentation Ihres Lasers.

Pin (01), (02) und (09) der LASER-Buchse können mit **config_laser_signals** und **config_laser_signals_list** konfiguriert werden, siehe auch Kapitel 7.4.2 "LASER-Buchse konfigurieren", Seite 195.

Externe Steuersignale

Die externen Steuersignale (Eingangssignale) sind:

- /START
- /STOP

Siehe auch Kapitel 9.3.1 "Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden", Seite 314.

BUSY-Listenausführungsstatus

An Pin (04) wird der BUSY-Listenausführungsstatus als BUSY OUT-Signal (identisch mit BUSY OUT, Seite 976) ausgegeben.

Wenn der BUSY-Listenausführungsstatus gesetzt ist, steht das BUSY OUT-Signal auf HIGH.

Siehe auch Kapitel 6.4.3 "Listenausführungsstatus", Seite 111.

2-Bit-Digital-Eingang

Die RTC6 PCIe-Karte stellt einen 2-Bit-Digital-Eingang bereit:

- DIGITAL IN1
- DIGITAL IN2

Zur Programmierung des Eingangs siehe Kapitel 9.2.2 "2-Bit-Digital-Eingang", Seite 313.

2-Bit-Digital-Ausgang

Die RTC6 PCIe-Karte stellt einen gepufferten 2-Bit-Digital-Ausgang bereit:

- DIGITAL OUT1
- DIGITAL OUT2

Zur Programmierung des Ausgangs siehe Kapitel 9.1.3 "2-Bit-Digital-Ausgang", Seite 307.

12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2

Die RTC6 PCIe-Karte stellt bereit:

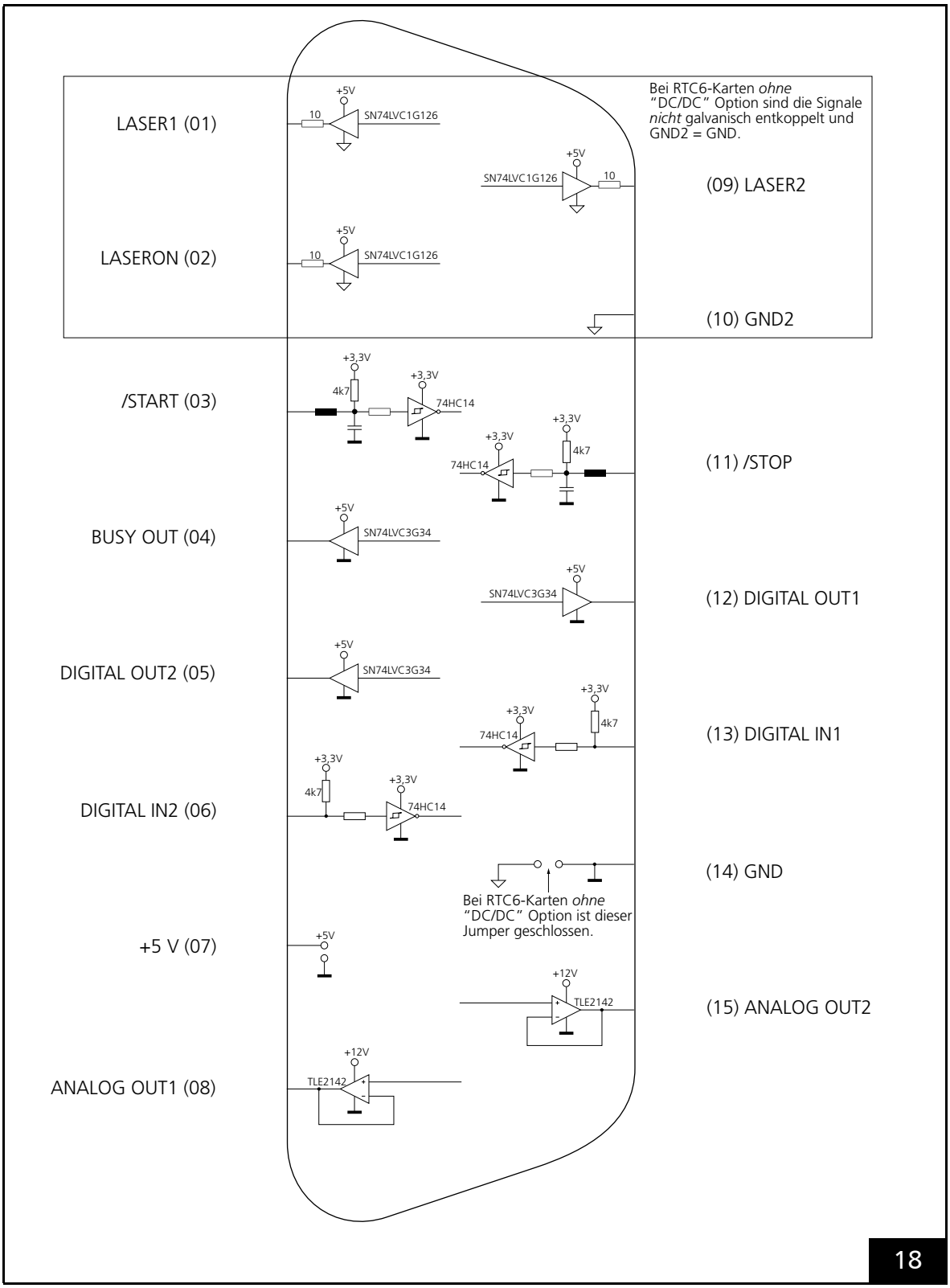
- 12-Bit-Analog-Ausgang 1
  ANALOG OUT1
- 12-Bit-Analog-Ausgang 2
  ANALOG OUT2(1)

Zur Programmierung der Ausgänge siehe Kapitel 9.1.4 "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 308.

Ein-/Ausgangsbeschaltung

Die Ein-/Ausgangsbeschaltung der LASER-Buchse zeigt Abbildung 18.

(1) Das Signal von ANALOG OUT2 ist auch über die MARKING ON THE FLY-Stiftleiste verfügbar, siehe Abschnitt "Analog-Ausgang", Seite 85.



18

LASER-Buchse, siehe auch Abbildung 17: Ein-/Ausgangsbeschaltung.
Siehe auch Abschnitt "Option "DC/DC-Wandler"", Seite 39.

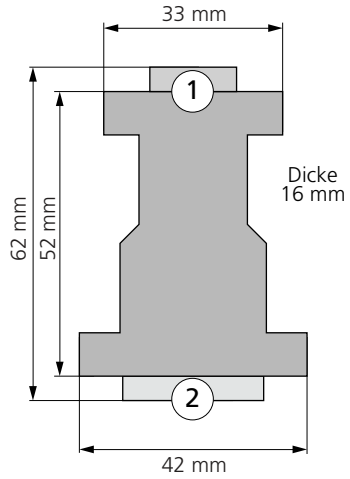
Laser-Adapter (Zubehör)

Der SCANLAB Laser-Adapter (Zubehör; #0114774) wird auf die 15-polige LASER-Buchse der RTC6 PCIe-Karte aufgesteckt. Dann stehen über eine 9-polige Sub-D-Buchse die gleichen Signale und Pinbelegung wie an der 9-poligen LASER-Buchse der RTC4 zur Verfügung.

Hinweise

- Der Laser-Adapter kann *nicht* direkt auf die RTC6 PCIe-Karte aufgesteckt werden, wenn dort bereits ein XY2-100-Konverter (Zubehör) aufgesteckt ist.
- Das BUSY OUT-Signal, der 2-Bit-Digital-Eingang und der 2-Bit-Digital-Ausgang stehen *nicht* über die 9-polige Sub-D-Buchse des Laser-Adapters zur Verfügung.

Die Abmessungen des Laser-Adapters zeigt Abbildung 19.



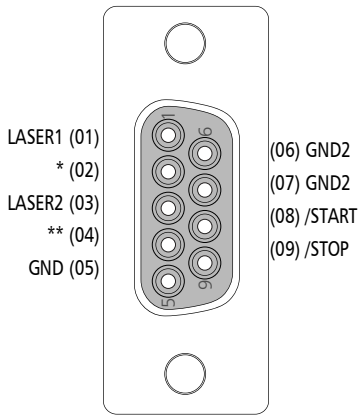
Legende

1. 9-polige Sub-D-Buchse, weiblich
2. 15-poliger Sub-D-Stecker, männlich

19

Laser-Adapter (Zubehör): Abmessungen.

Die Pinbelegung der 9-poligen Sub-D-Buchse zeigt Abbildung 20.



* LASERON oder ANALOG OUT2, abhängig von der Lötjumper-Einstellung (auf der Platine des Laser-Adapters)

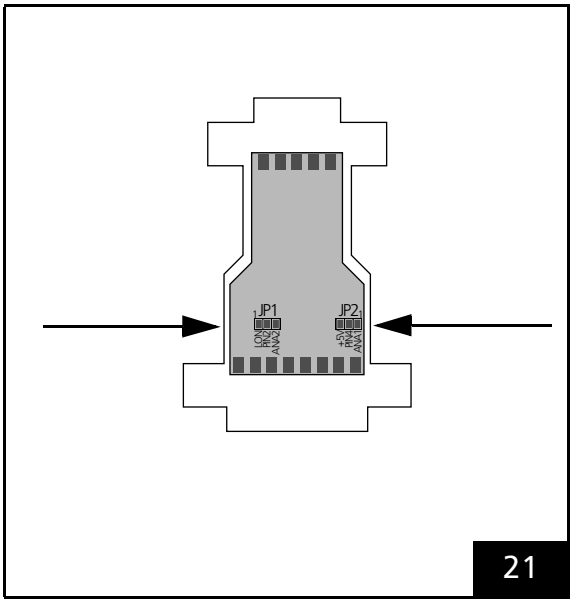
** ANALOG OUT1 oder +5 V, abhängig von der Lötjumper-Einstellung (auf der Platine des Laser-Adapters)

20

Laser-Adapter (Zubehör): Pinbelegung der 9-poligen Sub-D-Buchse, weiblich.

Die Signale an den Pins (2) und (4) der 9-poligen Sub-D-Buchse können über zwei Lötjumper im Laser-Adapter ausgewählt werden. Dazu muss das Gehäuse des Laser-Adapters vorsichtig an seinen 4 Clip-Verriegelungen (z.B. mittels Schraubendreher) geöffnet werden. Die Lötjumper **JP1** und **JP2** befinden sich auf der Platine des Laser-Adapters zwischen den beiden Sub-D-Anschlüssen.

Die Position der Lötjumper auf der Platine zeigt Abbildung 21.



Laser-Adapter (Zubehör): Position von Lötjumper **JP1** und **JP2** auf der Platine.

Die folgende Tabelle zeigt die möglichen Jumpereinstellungen. Andere Jumpereinstellungen sind nicht zulässig.

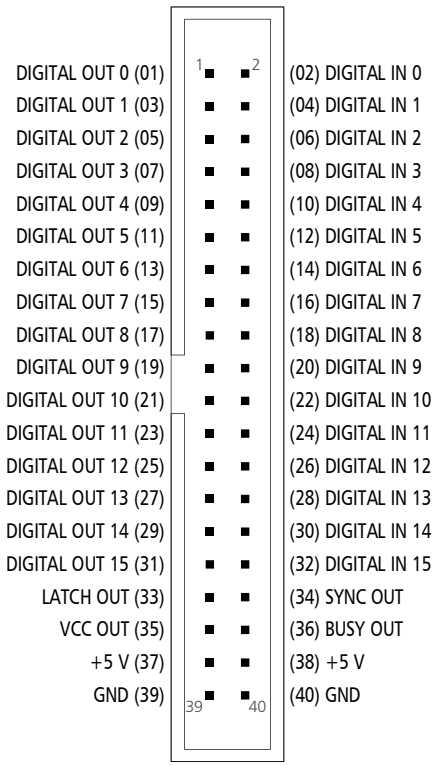
Lötjumper JP1: Signal an Pin (02)	Position 1-2  LASERON (Standard-Konfiguration)	Position 2-3  ANALOG OUT2
Lötjumper JP2: Signal an Pin (04)	Position 1-2  ANALOG OUT1 (Standard-Konfiguration)	Position 2-3  +5 V

4.6.2 EXTENSION 1-Stiftleiste

- Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", EXTENSION 1-Stiftleiste, Seite 975.

Die EXTENSION 1-Stiftleiste ist 40-polig. Sie befindet sich auf der Oberseite der RTC6 PCIe-Karte, siehe Abbildung 5.

Die Pinbelegung zeigt Abbildung 22.



Baugleich zu Würth 61204021621.

22

EXTENSION 1-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm.

Ausgangssignal-Pegel einstellen

Mit dem Lötjumperfeld A auf der Unterseite der RTC6 PCIe-Karte, siehe Abbildung 6, kann für alle Ausgangssignale der EXTENSION 1-Stiftleiste ein Signalpegel von 5 V oder 3,3 V eingestellt werden, siehe Kapitel 2.7.1 "Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste", Seite 42:

- DIGITAL OUT 0…DIGITAL OUT 15
- LATCH OUT
- SYNC OUT
- BUSY OUT
- VCC_OUT

Als Kontrollmöglichkeit wird der eingestellte Signalpegel kontinuierlich an Pin (35) ausgegeben: Signal VCC_OUT.

16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang

Die EXTENSION 1-Stiftleiste, siehe Abbildung 22, stellt bereit:

- 16-Bit-Digital-Eingang
  (DIGITAL IN 0…DIGITAL IN 15)
  geschützt (TTL-Pegel)
- 16-Bit-Digital-Ausgang
  (DIGITAL OUT 0…DIGITAL OUT 15)
  gepuffert (TTL-Pegel)

Dazu muss der Ausgangssignal-Pegel über die Jumpereinstellung konfiguriert sein, siehe Abschnitt "Ausgangssignal-Pegel einstellen", Seite 81.

Zur Programmierung siehe:

- Kapitel 9.1.1 "16-Bit-Digital-Ausgang", Seite 306
- Kapitel 9.2.1 "16-Bit-Digital-Eingang", Seite 313
- Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320

Synchronisation der Datenübergabe

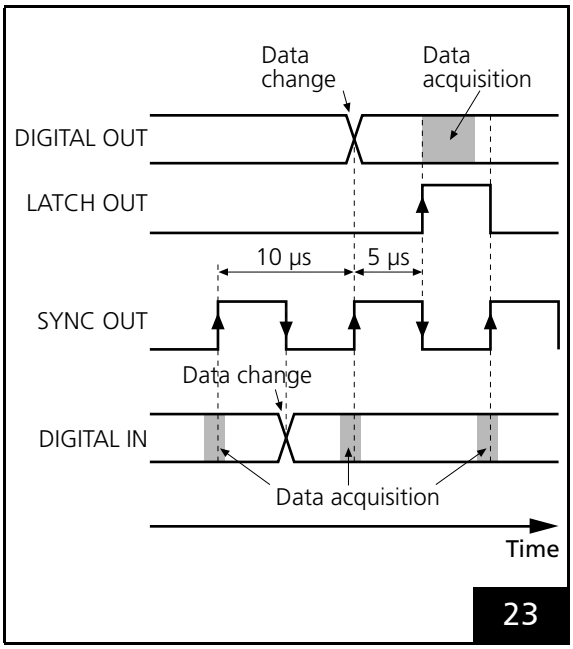
Falls am 16-Bit-Digital-Ausgang oder am 16-Bit-Digital-Eingang mehrere Bits gleichzeitig als Datenworte (und nicht unabhängig voneinander) übertragen werden, dann sollte das LATCH OUT-Signal oder das SYNC OUT-Signal zur Synchronisation der Datenübergabe verwendet werden.

Das LATCH OUT-Signal (ein 5 µs-Puls, HIGH-aktiv) wird als Triggersignal zur Übernahme der Ausgangswerte des 16-Bit-Digital-Ausgangs an Pin (33) ausgegeben. Es wird automatisch von der RTC6 PCIe-Karte erzeugt, wenn der Ausgangswert am 16-Bit-Digital-Ausgang geschrieben wird. Der Ausgangswert sollte mit der steigenden Flanke des LATCH-Signals abgegriffen werden. Die steigende Flanke erfolgt 5 µs nachdem der Wert am 16-Bit-Digital-Ausgang ausgegeben wurde, siehe Abbildung 23.

Zur Synchronisation der Datenübergabe am 16-Bit-Digital-Eingang wird an Pin (34) das SYNC OUT-Signal (ein Rechtecksignal mit Pulslänge 5 µs und Periodendauer 10 µs) kontinuierlich ausgegeben.

Veränderungen des Wertes am 16-Bit-Digital-Eingang sollten jeweils mit einer fallenden Flanke des SYNC OUT-Signals vorgenommen werden. Der DSP der RTC6 PCIe-Karte übernimmt den aktuell angelegten Wert immer mit der steigenden Flanke des SYNC OUT-Signals, siehe Abbildung 23.

Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", EXTENSION 1-Stiftleiste, Seite 975.



Synchronisation der Datenübergabe mit LATCH OUT-Signal oder SYNC OUT-Signal.

BUSY-Listenausführungsstatus

Das BUSY OUT-Signal an Pin (36) und das BUSY OUT-Signal an der LASER-Buchse(1) sind identisch, siehe Abschnitt "BUSY-Listenausführungsstatus", Seite 76.

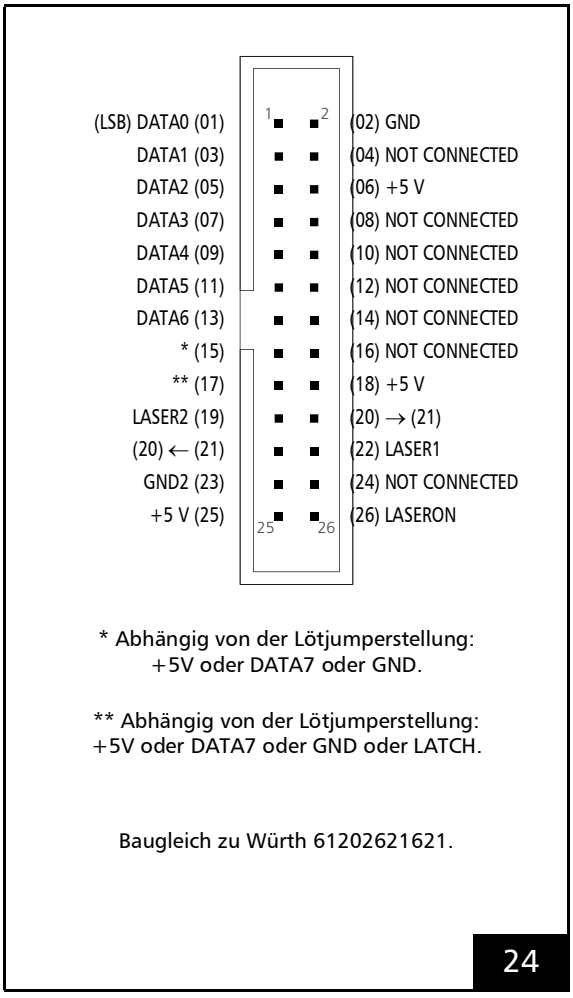
(1) RTC6 Ethernet-Karte: LASER-Stiftleiste.

4.6.3 EXTENSION 2-Stiftleiste

- Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", EXTENSION 2-Stiftleiste, Seite 975.

Die EXTENSION 2-Stiftleiste(1)(2) ist 26-polig. Sie befindet sich auf der Oberseite der RTC6 PCIe-Karte, siehe Abbildung 5.

Die Pinbelegung zeigt Abbildung 24.



EXTENSION 2-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm.

Hinweise

- Pin (15) und Pin (17) können mittels Lötjumper eingestellt werden.

(1) Auf der RTC4 ist die funktional entsprechende Stiftleiste mit LASER EXTENSION bezeichnet.

(2) RTC6 Ethernet-Karten stellen an ihren (10-poligen) EXT. 2-Stiftleisten keine Lasersteuersignale (LASER1 und LASER2) bereit.

Einstellungen mittels Lötjumper

Pin (15) wird über das Lötjumperfeld C konfiguriert, siehe Kapitel 2.7.3 "Lötjumperfeld C – Signal an EXTENSION 2-Stiftleiste Pin (15)", Seite 44.

Pin (17) wird über das Lötjumperfeld B konfiguriert, siehe Kapitel 2.7.2 "Lötjumperfeld B – Signal an EXTENSION 2-Stiftleiste Pin (17)", Seite 43.

Hinweise

- Ist das DATA7-Bit an Pin (15) zugewiesen, dann ist der volle 8-Bit Ausgangswert am Ausgabeport verfügbar (an den ungeradzahligen Pins der EXTENSION 2-Stiftleiste, Pin (01)…Pin (15)).
- Ist Pin (15) auf +5 V (HIGH-Pegel) eingestellt, dann haben die Ausgabewerte einen Offset von 128. Die Ausgabewerte liegen also zwischen 128…255.
- Ist Pin (15) auf GND (LOW-Pegel) eingestellt, dann sind nur Ausgabewerte zwischen 0…127 möglich.
- Das DATA7-Bit kann für andere Zwecke verwendet werden, indem es Pin (17) zugewiesen wird.

Lasersteuersignale

Die Lasersteuersignale LASER1 und LASER2 sind identisch zur LASER-Buchse und hängen vom eingestellten Laser-Modus ab, siehe Abschnitt "Lasersteuersignale", Seite 76.
Sie beziehen sich (wie bei der LASER-Buchse) auf GND2.

Achtung!

- Bei RTC6 PCIe-Karten mit der Option "DC/DC-Wandler" sind GND2 und die Lasersteuersignale LASERON, LASER1 und LASER2 an der LASER-Buchse und an der EXTENSION 2-Stiftleiste galvanisch von GND entkoppelt.(a)
- Bei RTC6 PCIe-Karten ohne Option "DC/DC-Wandler" sind GND2 und GND identisch.

(a) RTC6 PCIe-Karten: GND ist die PC-Masse.
RTC6 Ethernet-Karten: GND ist die Masse am POWER-Stecker.

8-Bit-Digital-Ausgang

Der gepufferte 8-Bit-Digital-Ausgang (TTL-Pegel) ist für Laser mit digitaler Leistungssteuerung vorgesehen (z.B. zum Steuern des Lampenstroms eines YAG-Lasers):

- DATA0…DATA7

Er kann aber natürlich auch für andere Zwecke eingesetzt werden. Vor der ersten Wertezuweisung befindet er sich im hochohmigen Zustand.

Zur Programmierung des Ausgangs siehe Kapitel 9.1.2 "8-Bit-Digital-Ausgang", Seite 307.

Das MSB (DATA7) des Ausgabewerts kann für Sonderzwecke eingesetzt werden. Dazu kann es einem anderem Pin auf der EXTENSION 2-Stiftleiste zugewiesen werden (Pin (15), Pin (17), s.o.).

Ist der Lötjumper entsprechend eingestellt (s.o.), wird an Pin (17) ein LATCH-Signal ausgegeben. Das LATCH-Signal (ein 5 μs-Puls, HIGH-aktiv) wird automatisch von der RTC6 PCIe-Karte erzeugt, wenn der Wert am 8-Bit-Digital-Ausgang ausgegeben wird.

Falls am 8-Bit-Digital-Ausgang mehrere Bits gleichzeitig als Datenwort (und nicht unabhängig voneinander) übertragen werden, dann sollte das LATCH-Signal zur Synchronisation der Datenübergabe verwendet werden.

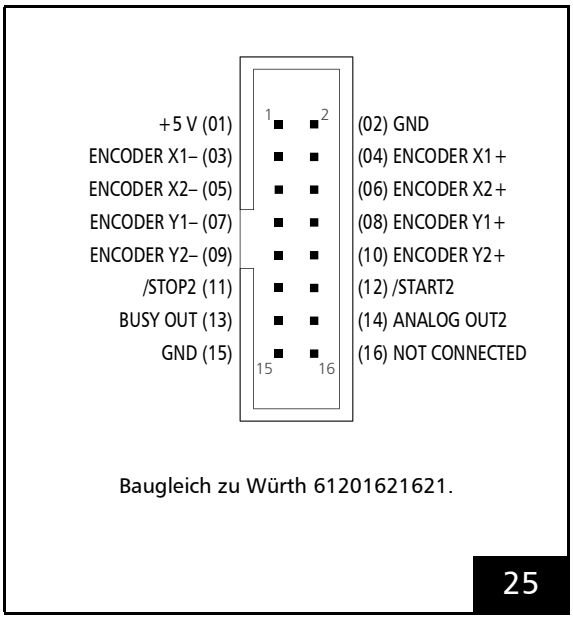
Der Wert am 8-Bit-Digital-Ausgang sollte mit der steigenden Flanke des LATCH-Signals abgegriffen werden. Die steigende Flanke erfolgt 5 μs, nachdem der Wert am 8-Bit-Digital-Ausgang geschrieben wurde, siehe auch Abbildung 23.

4.6.4 MARKING ON THE FLY-Stiftleiste

- Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", MARKING ON THE FLY-Stiftleiste, Seite 976.

Die MARKING ON THE FLY-Stiftleiste ist 16-polig und befindet sich auf der Oberseite der RTC6 PCIe-Karte, siehe Abbildung 5. Sie stellt u.a. Pins für Encodereingaben zur Verfügung.

Die Pinbelegung zeigt Abbildung 25.



MARKING ON THE FLY-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm.

Wenn die Option "Processing on the fly" freigeschaltet ist, kann eine Lasermaterialbearbeitung bewegter Werkstücke (z.B. auf einem Förderband oder auf einer rotierenden Scheibe) realisiert werden, siehe Kapitel 8.6 ""Processing on the fly"", Seite 254.

Encodereingänge

Die RTC6 PCIe-Karte ist mit 2 Encodereingängen ausgestattet:

- ENCODER X
- ENCODER Y

ENCODER X und ENCODER Y sind jeweils für ein Paar (1, 2) von standardisierten differenziellen Eingangssignalen (RS-422) vorgesehen. Siehe auch Abschnitt "Eingänge für externe Encodersignale", Seite 324.

Externe Steuersignale

Die externen Steuersignale (Eingangssignale) sind:

- /START2
- /STOP2

Siehe auch Kapitel 9.3.1 "Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden", Seite 314.

Analog-Ausgang

Das am 12-Bit-Analog-Ausgang 2 ANALOG OUT2 (LASER-Buchse) ausgegebene Signal wird auch an Pin (14) ausgegeben, siehe auch Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77.

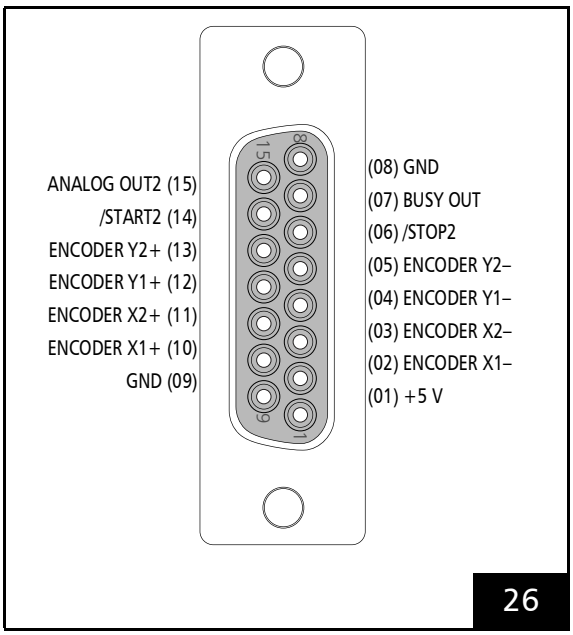
BUSY-Listenausführungsstatus

Das BUSY OUT-Signal an Pin (13) und BUSY OUT-Signal an der LASER-Buchse sind identisch, siehe auch Abschnitt "BUSY-Listenausführungsstatus", Seite 76.

MARKING ON THE FLY-Slotblech (Zubehör)

SCANLAB empfiehlt den Gebrauch eines zusätzlichen Slotblechs bei Verwendung der Eingänge und Signale der MARKING ON THE FLY-Stiftleiste, siehe Kapitel 2.8.5 "Slotblech mit 15-poliger Sub-D-Buchse für die MARKING ON THE FLY-Stiftleiste", Seite 46.

Die Pinbelegung zeigt Abbildung 26.



MARKING ON THE FLY-Slotblech (Zubehör), #0109272: Pinbelegung der 15-poligen Sub-D-Buchse (weiblich).

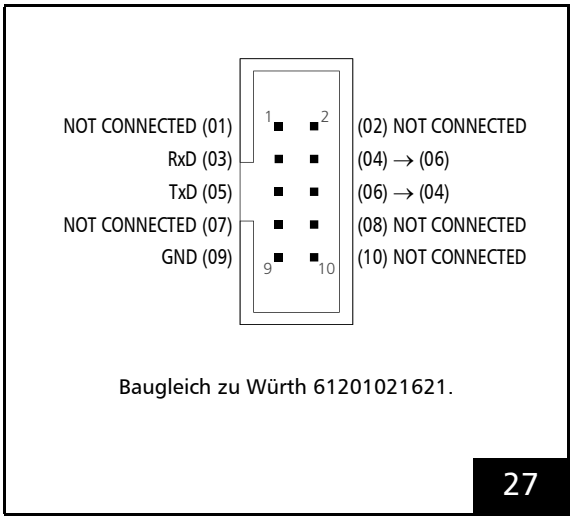
Alternativ steht auch das Slotblech mit 15-poliger Sub-D-Buchse und 9-poliger Sub-D-Buchse, Seite 47 zur Verfügung.

4.6.5 RS232-Stiftleiste

- Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", RS232-Stiftleiste, Seite 976.

Die RS232-Stiftleiste ist 10-polig und befindet sich auf der Oberseite der RTC6 PCIe-Karte, siehe Abbildung 5.

Die Pinbelegung zeigt Abbildung 27.



RS232-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm.

Die RS232-Stiftleiste ist eine Hardware-Schnittstelle, die zum bidirektionalen Datenaustausch vorgesehen ist.

Mit **uart_config** kann (ab DLL 612) die RS-232-Schnittstelle konfiguriert(1) werden. **uart_config** gibt die tatsächlich eingestellte Baud-Rate zurück.

Die Baud-Rate (Voreinstellung: 9600 Baud) kann mit **uart_config** eingestellt werden (160…12.800.000 Baud). Andere Parameter können nicht verändert werden (Data-Bits: 8, Start-Bits: 1, Stopp-Bits: 1, Parity: None).

Zum Ausgeben von Daten über die RS-232-Schnittstelle siehe Kapitel 9.1.6 "RS-232-Schnittstelle", Seite 312.

Zum Einlesen von Daten über die RS-232-Schnittstelle siehe Kapitel 9.2.3 "RS-232-Schnittstelle", Seite 313.

(1) Alternativ kann auch das ältere **rs232_config** verwendet werden, der jedoch keine Baud-Rate zurück gibt. Mit < DLL 612 ist die maximale Baud-Rate auf den Bereich [300…115200] begrenzt.

4.6.6 McBSP/ANALOG-Stiftleiste

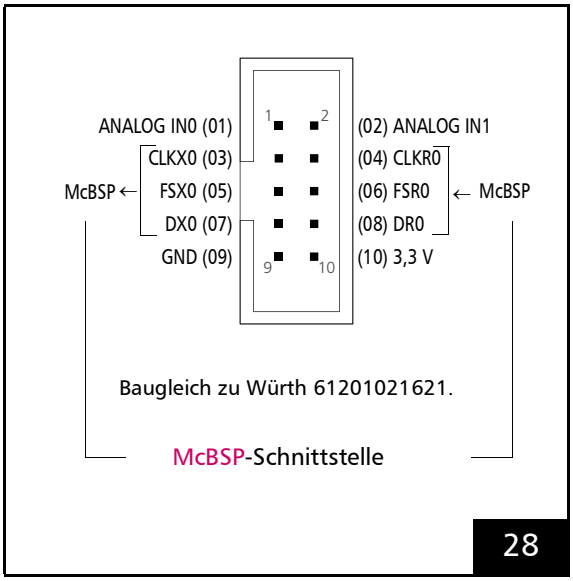
- Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", McBSP/ANALOG-Stiftleiste, Seite 977.

Die McBSP/ANALOG-Stiftleiste(1) ist 10-polig. Sie befindet sich auf der Oberseite der RTC6 PCIe-Karte, siehe Abbildung 5.

Die McBSP/ANALOG-Stiftleiste ist eine Hardware-Schnittstelle

- die zum bidirektionalen Datenaustausch vorgesehen ist (McBSP), siehe Abschnitt "McBSP-Schnittstelle", Seite 87 und
- außerdem zwei 10 V-Analog-Eingänge bereitstellt: ANALOG IN0 an Pin (01) und ANALOG IN1 an Pin (02), siehe Abschnitt "Analog-Eingänge", Seite 89.

Die Pinbelegung zeigt Abbildung 28.



McBSP/ANALOG-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm.

McBSP-Schnittstelle

Die McBSP-Schnittstelle, siehe Abbildung 28, wird durch **load_program_file** initialisiert mit:

- `XDelay` = `RDelay` = 1
- 8 MHz Taktfrequenz

Für McBSP sind Pin (03), (05), (07) für die ausgehenden Signale, und Pin (04), (06), (08) für die eingehenden Signale bestimmt.

Nach **load_program_file** werden ständig folgende Signale ausgegeben:

- das Taktsignal CLKX an Pin (03)
- alle 10 μs ein Frame-Synchronization-Signal FSX an Pin (05)
- ein Datensignal DX an Pin (07) (Voreinstellung: SampleY|SampleX), siehe **set_mcbsp_out**

Die McBSP-Schnittstelle kann in Anwendungen zu verschiedenen Zwecken integriert werden:

- Als Alternative zu Encodersignalen kann die Position eines bewegten Werkstücks direkt eingebunden werden, siehe auch Kapitel "Korrektur über McBSP-Schnittstelle", Seite 257. Voraussetzung: Option "Processing on the fly".
- Als Alternative zu Matrix- und Offset-Befehlen können Koordinatentransformationen zur Ausrichtung eines Werkstücks mit kontrollierbarem Timing übertragen und eingebunden werden (Online-Positionierung), siehe auch Kapitel 9.3.4 "Synchronisation und Online-Positionierung mit McBSP-Signalen", Seite 325.
- Mit **set_multi_mcbsp_in** können neben einer 3D-Fly-Korrektur mit Positionsangaben auch Laserleistung und weitere Parameter übertragen werden, siehe auch Kapitel "Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe", Seite 258.
  Voraussetzung: Option "Processing on the fly".
- Ständige Datenausgabe im 10 μs-Takt. Mit **set_mcbsp_out** und **set_mcbsp_out_ptr** kann ausgewählt werden, welche Datensignale ausgegeben werden.
- Der OIE-Ausgabemodus, Seite 773 kann mit **set_mcbsp_out_oie_ctrl** und **set_mcbsp_out_oie_list** an der McBSP-Schnittstelle eingestellt werden.

(1) Bei RTC5-Karten als SPI / I2C-Stiftleiste bezeichnet.

Zum Ausgeben von Daten über die McBSP-Schnittstelle siehe auch Kapitel 9.1.7 "McBSP-Schnittstelle", Seite 312.

Zum Eingeben von Daten über die McBSP-Schnittstelle, und Kapitel 9.2.4 "McBSP-Schnittstelle", Seite 313.

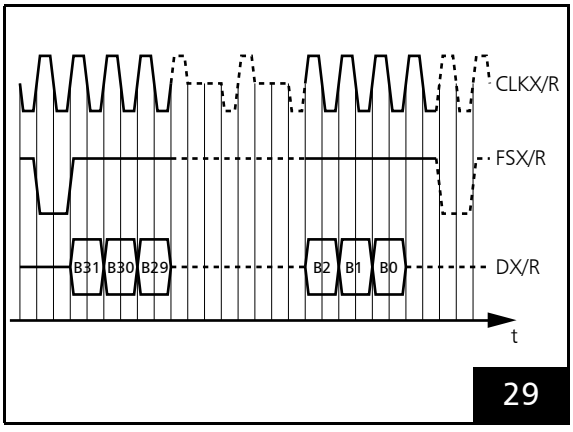
Für die McBSP-Datenübertragung kann das Daten-Delay für Sender und Empfänger getrennt mittels **mcbsp_init** zwischen 0 und 2 eingestellt werden (Voreinstellung: 1).

Alle Signale beziehen sich auf GND(1).

RTC6 PCIe-Karte als Sender

Für CLKX0, FSX0 und DX0 gelten die folgenden Spezifikationen:

- Pegelsignale 3,3 V TTL
- McBSP-Mode:
  - Single Phase Frame
  - Single Element per Frame
  - 32 Bits per Element
  - Daten-Delay `XDelay` Bits
- Das Timing-Diagramm der McBSP-Signale ist in Abbildung 29 dargestellt. Bei steigender Taktflanke wird ein Frame-Synchronization-Signal (LOW-aktiv) erzeugt, das nur einen Takt (1 Daten-Bit) lang ist. Die 32 Daten-Bits werden nach `XDelay`-Takten mit jeweils steigender Taktflanke in der Reihenfolge Bit #31…Bit #0 gesendet. Die Sendefrequenz ist in der Standardeinstellung 8 MHz (4 µs pro Datenwort). Sie kann auch mit **set_mcbsp_freq** auf einen Wert zwischen 4 MHz und 16 MHz eingestellt werden.



Timing-Diagramm der McBSP-Signale bei 1 Bit Daten-Delay. Voreinstellung: `XDelay` = `RDelay` = 1.

RTC6 PCIe-Karte als Empfänger

Für die CLKR0, FSR0, DR0 gelten die folgenden Spezifikationen:

- Pegelsignale 3,3 V oder 5 V TTL.
- McBSP-Mode:
  - Single Phase Frame
  - Single Element per Frame
  - 32 Bits per Element
  - Daten-Delay `RDelay` Bits
- Das Timing-Diagramm der McBSP-Signale ist in Abbildung 29 dargestellt. Ein mit steigender (externer) Taktflanke erzeugtes Frame-Synchronization-Signal (LOW-aktiv) wird mit der nächsten fallenden Flanke des Taktsignals detektiert. Die Dauer des Frame-Synchronization-Signals spielt keine Rolle. Nach `RDelay` Takten werden mit jeder weiteren fallenden Flanke der Taktsignale die 32 Daten-Bits in der Reihenfolge Bit #31…Bit #0 detektiert, wenn sie mit steigender Flanke gesendet werden. Beachten Sie dazu auch die folgenden Hinweise:
  - Das jeweils erste Frame-Synchronization-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP-Schnittstelle ignoriert. Die bereitgestellten Daten werden also nicht übertragen. Gegebenenfalls sollte erst ein Dummy-Wert zur Schnittstelle geschickt werden. Mit **read_mcbsp** kann dann auf eine erfolgreiche Übertragung geprüft werden.
  - Die Bitfrequenz (Empfangsfrequenz) wird ausschließlich aus den eingehenden Taktsignalen bestimmt und darf maximal 16 MHz betragen.
  - Nach dem letzten Daten-Bit #0 muss noch mindestens ein weiteres externes Taktsignal gesendet werden, damit das Datenwort auch von der DSP-seitigen Schnittstelle abgeholt und zwischengespeichert werden kann. Zeitgleich mit diesem Taktsignal kann ein Frame-Synchronization-Signal auch schon wieder eine neue Übertragung einleiten.

(1) Siehe Fußnote auf Seite 39.

- Nach einem **load_program_file** werden die McBSP-Daten intern ständig abgeholt und zwischengespeichert, sobald (neue) Daten übertragen werden. Neuere Datenworte überschreiben ältere.
  - Nach **load_program_file** und/oder nach Aktivierung einer "Processing on the fly"-Korrektur mit **set_fly_x_pos**, **set_fly_y_pos** oder **set_fly_rot_pos** werden die Eingabewerte an die interne Speicherstelle 0 übertragen.
  - Nach Aktivierung einer Online-Positionierung werden die Eingabewerte an die internen Speicherstellen 1 oder 2 übertragen:
    - Für eine "Lokale Online-Positionierung" mit **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** oder **set_mcbsp_matrix**, siehe Abschnitt ""Lokale Online-Positionierung" konfigurieren", Seite 241
    - Für eine "Globale Online-Positionierung" mit **set_mcbsp_global_x**, **set_mcbsp_global_y** und/oder **set_mcbsp_global_rot** oder **set_mcbsp_global_matrix**, siehe Abschnitt ""Globale Online-Positionierung" konfigurieren", Seite 243
  - Der jeweils zuletzt vollständig an eine Speicherstelle übertragene Wert kann mit **read_mcbsp** von der jeweiligen Speicherstelle ausgelesen werden.
  - Nach Aktivierung einer "Processing on the fly"-Korrektur mit **set_mcbsp_in** oder **set_mcbsp_in_list** werden die Eingabewerte an die internen Speicherstellen 0 und 3 übertragen.
  - Nach Aktivierung einer "Processing on the fly"-Korrektur mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** werden die Eingabewerte an die internen Speicherstellen 0 bis 3 übertragen.

Analog-Eingänge

An der McBSP/ANALOG-Stiftleiste (RTC6 Ethernet-Karten: SPI/ANA/UART-Stiftleiste) werden zwei Analog-Eingänge(1) bereitgestellt:

- ANALOG IN0
- ANALOG IN1

Nach dem ersten **read_analog_in**-Aufruf werden dort angelegte Spannungen:

- endlos automatisch konvertiert (Dauer ca. 0,3 ms)(2)
- als 12-Bit-Digitalwerte zum DSP übertragen

Mit dem Kontrollbefehl **read_analog_in** können die anliegenden Werte jederzeit ausgelesen werden.

Technische Spezifikationen siehe Analoge Eingänge, Seite 977.

(1) Wie RTC5 PCI-Express-Karten (aber nicht die RTC5 PCI-Karten).

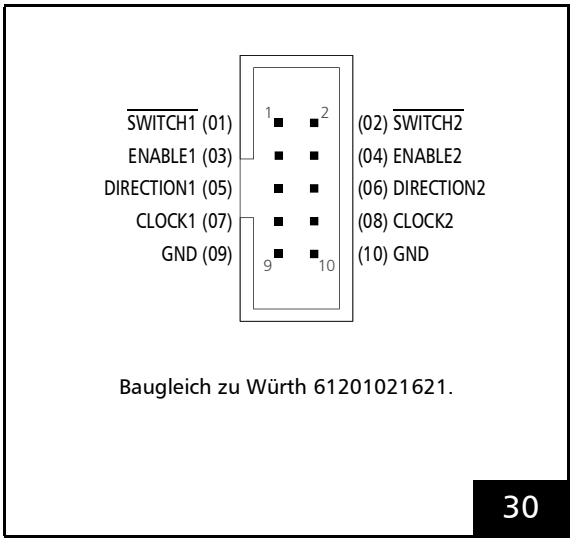
(2) Beachten Sie den Hinweis bei **get_temperature**, Seite 482.

4.6.7 STEPPER MOTOR-Stiftleiste

- Zu den technischen Daten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", STEPPER MOTOR-Stiftleiste, Seite 977.

Die STEPPER MOTOR-Stiftleiste ist 10-polig und befindet sich auf der Oberseite der RTC6 PCIe-Karte, siehe Abbildung 5.

Die Pinbelegung zeigt Abbildung 30.



STEPPER MOTOR-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm.

An der STEPPER MOTOR-Stiftleiste können Signale zum Ansteuern von 1…2 Schrittmotoren ausgegeben werden, siehe Kapitel 9.1.5 "Schrittmotoren ansteuern", Seite 309.

Für jeden Schritt, der gefahren werden soll, generiert die RTC6 PCIe-Karte einen HIGH-aktiven 5 μs-Puls als CLOCK-Signal. Das CLOCK-Signal ist zwischen den Pulsen dauerhaft LOW.

Mit den ENABLE-Signalen kann ein Anwenderprogramm z.B. den Motorstrom ein- und ausschalten.

Mit den DIRECTION-Signalen kann die Richtung festgelegt und mit den CLOCK-Pulsen jeweils ein einzelner Schritt ausgeführt werden.

SWITCH1-Eingang und SWITCH2-Eingang dienen zum Anschließen von End-Schaltern. Die RTC6 PCIe-Karte erkennt eine End-Schalterposition an einem LOW-Pegel (LOW-aktiv).

5 Installation und Inbetriebnahme

Die Installation der RTC6 PCIe-Karte erfolgt in den folgenden Schritten:

(1) Prüfen der RTC6 PCIe-Karten-Jumpereinstellungen, siehe Kapitel 5.1 "Jumpereinstellungen prüfen", Seite 91

(2) Hardware-Installation: Einbau der RTC6 PCIe-Karte, siehe Kapitel 5.2 "RTC6 PCIe-Karte einbauen", Seite 91

(3) Installation des RTC6-Kartentreibers, siehe Kapitel 5.3 "RTC6-Kartentreiber installieren", Seite 92(1)

(4) Installation der RTC6-Software, siehe Kapitel 5.4 "RTC6-Software installieren", Seite 93

(5) Beachten Sie beim Starten des Laser-Systems die sichere Einschaltreihenfolge, siehe Kapitel 5.6 "Sichere Einschalt- und Ausschaltreihenfolge", Seite 94

(6) Für einen Funktionstest nach der Installation kann z.B. das mitgelieferte HPGL-Converter-Programm verwendet werden, siehe Kapitel 5.7 "Funktionstest", Seite 95

5.1 Jumpereinstellungen prüfen

SCANLAB liefert RTC6 PCIe-Karten unterschiedlicher Jumperkonfigurationen aus.

(1) Stellen Sie sicher, dass die einzubauende RTC6 PCIe-Karte die benötigte Jumperkonfiguration aufweist. Wenn Sie die Werkseinstellung verändern müssen, gehen Sie zunächst vor wie in Kapitel 2.7 "Jumperkonfigurationen und Typenbezeichnungen", Seite 41 beschrieben.

(2) Fahren Sie dann mit Kapitel 5.2 "RTC6 PCIe-Karte einbauen", Seite 91 fort.

5.2 RTC6 PCIe-Karte einbauen

Achtung!

- Die Karte sollte immer in einer elektrostatisch neutralen Umgebung im mitgelieferten antistatischen Beutel gelagert werden.
- Die Installation muss an einem Arbeitsplatz durchgeführt werden, der den ESD-Schutzvorschriften entspricht. Beachten Sie beim Einbau der Karte die Anweisungen in der Bedienungsanleitung Ihres PCs.
- Berühren Sie die Karte nicht an den elektrischen Kontakten.
- Schützen Sie die Karte vor Feuchtigkeit, Staub, korrosiven Dämpfen und mechanischen Belastungen.

Der Einbau der RTC6 PCIe-Karte in den PC erfolgt in den folgenden Arbeitsschritten:

(1) Entfernen Sie alle Datenträger aus den Laufwerken Ihres PCs.

(2) Fahren Sie das Betriebssystem herunter, schalten Sie den PC ab und trennen Sie ihn von der Stromversorgung.

(3) Trennen Sie alle Peripheriegeräte wie Monitor, Maus, Tastatur, Drucker etc. von ihrem PC.

(4) Bringen Sie den PC für den Einbau der Karte an einen Arbeitsplatz, der den ESD-Schutzvorschriften gerecht wird.

(5) Entfernen Sie die Abdeckung Ihres PCs und das Slotblech eines freien PCIe-Steckplatzes. Von der Oberseite des Steckplatzes aus muss eine Höhe von mindestens 105 mm verfügbar sein.

(6) Nehmen Sie die RTC6 PCIe-Karte aus dem antistatischen Beutel. Berühren Sie die Karte nicht an den Kontaktstellen.

(1) Schritt 3 entfällt bei RTC6 Ethernet-Karten.

(7) Wenn Sie die Signale der RTC6 PCIe-Karten-Stiftleisten verwenden wollen, dann schließen Sie entsprechende Verbindungskabel an. SCANLAB empfiehlt den Gebrauch zusätzlicher Slotbleche mit geeigneten Steckern. Dann sind die Signale auch bei geschlossenem PC-Gehäuse zugänglich. Alle Anschlüsse und Signale der RTC6 PCIe-Karte sind in Kapitel 4 "RTC6 PCIe-Karte – Layout und Schnittstellen", Seite 66 beschrieben.

(8) Stecken Sie die RTC6 PCIe-Karte in den vorgesehenen PCIe-Steckplatz. Beachten Sie dabei die Hinweise im Handbuch Ihres PCs(1).

(9) Schließen Sie den PC wieder.

(10) Bringen Sie den PC an seinen Einsatzort und verbinden Sie alle Peripheriegeräte.

(11) Verbinden Sie die 9-polige Sub-D-Buchse an der RTC6 PCIe-Karte – gegebenenfalls unter Einbindung des XY2-100-Konverter (Zubehör) – über ein Datenkabel mit dem Scan-System, siehe Kapitel 4.5 "Schnittstellen zum Scan-System", Seite 69.

(12) Verbinden Sie die 15-polige Sub-D-Buchse an der RTC6 PCIe-Karte über eine geeignete Schnittstelle mit dem Laser, siehe Kapitel 4.6 "Schnittstellen zum Laser und zur Peripherie", Seite 75.

(13) Verbinden Sie gegebenenfalls eine oder mehrere RTC6 PCIe-Karten-Stiftleisten (über zusätzliche Slotbleche, s.o.) mit dem Laser, mit einem Handling-System oder anderen Zusatzgeräten der Gesamtmaschine.

(14) Überprüfen Sie die Verbindungen und schalten Sie Ihren PC wieder an.

(1) Sollen mehrere RTC6 PCIe-Karten mit Master/Slave-Synchronisation in einem PC betrieben werden, dann müssen die Karten paarweise über die Master- und Slave-Anschlüsse miteinander verbunden und möglichst (empfohlen) in benachbarte PCIe-Steckplätze gesteckt werden, siehe auch Kapitel 4.4 "Master-Stiftleiste, Slave-Stiftleiste", Seite 68.

5.3 RTC6-Kartentreiber installieren

Achtung!

- Falls auf Ihrem PC bereits ein auf WDM-Technologie basierender RTC3/4/5-Kartentreiber
  - installiert ist oder
  - installiert war und deinstalliert wurde oder
  - Sie sich diesbezüglich nicht sicher sind:
    (1) Installieren Sie den RTC6-Kartentreiber.
    (2) Führen Sie `ScanlabClassChecker.cmd` als Administrator aus (im mitgelieferten Softwarepaket enthalten; siehe dort auch Hintergrundinformationen in `ReadMe_ScanlabClassChecker.pdf`).
    Schritt 2 kann auf neuen PCs, auf denen noch nie ein RTC-Kartentreiber installiert war, auch ausgelassen werden.

Zur Installation des RTC6-Kartentreibers für Windows 10, 8, 7 gehen Sie wie folgt vor:

- Nachdem Sie die RTC6 PCIe-Karte in den PC eingebaut haben, starten Sie den Computer.

Unter Windows 7:

(1) Wenn der "Add Hardware Wizard" nicht automatisch startet, dann starten Sie ihn vom Control-Panel (Systemsteuerung) aus.

(2) Geben Sie im "Add Hardware Wizard" das Verzeichnis an, in dem sich der RTC6-Kartentreiber befindet. Während der Installation wählt dann das Betriebssystem automatisch die geeignete Treiber-Datei (32-Bit oder 64-Bit) aus.

(3) Führen Sie `ScanlabClassChecker.cmd` als Administrator aus (im mitgelieferten Softwarepaket enthalten; siehe dort auch Hintergrundinformationen in `ReadMe_ScanlabClassChecker.pdf`).

Unter Windows 10, 8:

(1) Rufen Sie den Gerätemanager auf. Finden Sie im vom Gerätemanager dargestellten Geräte-Verzeichnis den Eintrag 'PCI-Gerät' und aktualisieren Sie dessen Treiber mit Angabe des Verzeichnisses, in dem sich der RTC6-Kartentreiber befindet.

(2) Führen Sie `ScanlabClassChecker.cmd` als Administrator aus (im mitgelieferten RTC6-Software-Paket enthalten; siehe dort auch Hintergrundinformationen in `ReadMe_ScanlabClassChecker.pdf`).

Achtung!

- Die RTC6 PCIe-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCIe-Busses abgeschaltet werden. Deaktivieren Sie entsprechende Standby- oder Sleep-Modi des Betriebssystems. Deaktivieren Sie insbesondere den Windows-Sleep-Modus (dieser ist bei einigen Windows-Betriebssystemen standardmäßig voreingestellt). Dafür können Sie das im RTC6-Software-Paket (unter `RTC6 Tools`) enthaltene Skript `SleepMode.cmd` verwenden.
  Siehe hierzu auch Abschnitt "Hinweise", Seite 38.

Achtung!

- Bei manchen PCs kann es vorkommen, dass eine RTC6 PCIe-Karte in einem bestimmten PCIe-Slot vom Gerätemanager nicht erkannt wird. Verwenden Sie einen anderen Slot.

5.4 RTC6-Software installieren

Zusätzlich zum RTC6-Kartentreiber muss die RTC6-Software "installiert" werden, d.h. auf die Festplatte des PCs kopiert oder entpackt werden.

RTC6-Software für RTC6 PCIe-Karten besteht aus den folgenden Dateien:

- RTC6-DLL
  - `RTC6DLL.dll`
    Win32-basierte RTC6 dynamische Link Library
  - `RTC6DLLx64.dll`
    Win64-basierte RTC6 dynamische Link Library
- RTC6-Dateien
  - `RTC6OUT.out`
    Programmdatei für den PCIe-DSP
  - `RTC6ETH.out`
    Programmdatei für den Eth-DSP
  - `RTC6RBF.rbf`
    Firmwaredatei für das FPGA
  - `RTC6DAT.dat`
    Binäre Hilfsdatei

Die Dateien sind im RTC6-Software-Paket enthalten. Zur Kennzeichnung und Verwaltung unterschiedlicher Softwareversionen werden einige der Dateien zusätzlich gezippt ausgeliefert (die `zip`-Dateinamen `RTC6<…>_<Version>.zip` enthalten die jeweiligen Versionsnummern).

Führen Sie die folgenden Schritte aus, um die RTC6-Software zu installieren:

- Kopieren oder entpacken Sie die Dateien `RTC6DLL.dll` (oder `RTC6DLLx64.dll`), `RTC6OUT.out`, (bei RTC6 Ethernet-Karten `RTC6ETH.out`) `RTC6RBF.rbf` und `RTC6DAT.dat` (der gewünschten Version) auf Ihren PC.
  Beachten Sie, dass unterschiedliche Dateiversionen nicht beliebig miteinander kombiniert werden können (jede `zip`-Datei enthält jeweils eine Text-Datei mit entsprechenden Versions-Informationen).

Hinweise

- Stellen Sie außerdem die notwendigen Korrekturdatei(en) bereit (bestehende `*.ct5`-Korrekturdateien können weiterhin genutzt werden; speziell angepasste Korrekturdateien nicht überschreiben!).

5.5 RTC6 PCIe-Karten-BIOS upgraden und downgraden

	Anwendungsfall "Upgrade" (Beispiel)	Anwendungsfall "Downgrade" (Beispiel)
Vor Schritt 3	BIOS 23 (a)	BIOS 24(a)
	↓	↓
Datei in Schritt 3	RTC6BIOSOUT_24.out	RTC6BIOSOUT_23.out
	↓	↓
Nach Schritt 3	BIOS 24(a)	BIOS 23(a)

(a) Feststellbar mit RTC6conf.exe.

Voraussetzungen

- RTC6conf.exe, Seite 35
- Dateien gemäß obiger Tabelle

Vorgehen

(1) Starten Sie RTC6conf.exe(1).

(2) Wählen Sie die gewünschte RTC6 PCIe-Karte aus. Beachten Sie, dass die aktuelle BIOS-Version angezeigt wird.

(3) Installieren Sie die Datei (sinngemäß zur Tabelle oben) mittels der Schaltfläche **FLASH BIOS**.

(4) Fahren Sie den PC herunter und starten ihn erneut.

5.6 Sichere Einschalt- und Ausschaltreihenfolge

Beachten Sie beim Einschalten der Komponenten eines Lasersystems die folgende Reihenfolge:

(1) Schalten Sie den PC, in den die RTC6 PCIe-Karte eingebaut ist, ein und starten Sie die Steuerungs-Software (Initialisierung der Karte).

(2) Schalten Sie die benötigten Peripheriegeräte ein.

(3) Schalten Sie die Spannungsversorgung des Scan-Systems ein.

(4) Schalten Sie den Laser ein.

Gehen Sie beim Abschalten des Lasersystems in genau umgekehrter Reihenfolge vor:

(1) Schalten Sie den Laser aus.

(2) Schalten Sie die Spannungsversorgung des Scan-Systems aus.

(3) Schalten Sie die Peripheriegeräte aus.

(4) Beenden Sie die Steuerungs-Software und schalten Sie den PC mit der RTC6 PCIe-Karte aus.

Vorsicht!

- Beim Ein- und Ausschalten des PCs können an den Ausgängen der RTC6 PCIe-Karte kurzzeitige Pegeländerungen, also auch unvorhergesehene Änderungen der Lasersteuersignale stattfinden. Die obige Ein- und Ausschaltreihenfolge muss deswegen unbedingt eingehalten werden, sonst besteht die Gefahr dass der Laser kurzzeitig unerwartet angeht.
- Das Scan-System muss immer nach dem PC und der Steuerungs-Software eingeschaltet oder vorher wieder ausgeschaltet werden, sonst kann es zu unvorhergesehenen Scan-Bewegungen des Scan-Systems kommen. Der Laser muss immer zuletzt eingeschaltet oder zuerst ausgeschaltet werden, sonst besteht die Gefahr, dass der Laserstrahl in eine unkontrollierte Richtung abgelenkt wird.

(1) Siehe auch RTC6conf.pdf.

5.7 Funktionstest

Vorsicht!

- *Schalten Sie immer zuerst den PC und die Scan-Kopf-Stromversorgung an, dann erst den Laser. Anderenfalls besteht die Gefahr, dass der Laserstrahl in eine ungewünschte Richtung abgelenkt wird.*
  SCANLAB empfiehlt den Gebrauch eines Shutters, um die unkontrollierte Emission von Laserstrahlung zu vermeiden.

Zum Testen der Scan-Kopf-Steuerung können Sie den mitgelieferten HPGL-Converter `Hpgl.exe` verwenden, siehe auch Abschnitt "Verzeichnis HPGL", Seite 33. Dieses Programm erlaubt es, Grafikdateien im Hewlett Packard HPGL-Format (Vektorgrafik-Plotter-Dateien `*.plt`) zu laden und an die RTC6 PCIe-Karte auszugeben.

(1) Kopieren Sie `Hpgl.exe` und die mitgelieferte `*.plt`-Dateien in das gleiche Verzeichnis wie die `RTC6DLL.dll`, die RTC6-Dateien und die Korrekturdatei(en).

(2) Starten Sie `Hpgl.exe`.

(3) Wählen Sie **Options** > **Correction**, und dann eine Korrekturdatei.

(4) Wählen Sie **File** > **Load HPGL-File**, und dann ein `*.plt`.

(5) Um die Ausgabe zu starten, wählen Sie **Mark** > **Start Marking**.

5.8 Anwenderprogramme und Demoprogramme

Achtung!

- Überprüfen Sie vor einem Start sorgfältig Ihr Anwenderprogramm. Programmierfehler können einen Systemabsturz verursachen. In diesem Fall lassen sich weder Laser noch Scan-System steuern.

Die DLLs für RTC6-Anwenderprogramme (`RTC6DLL.dll`, `RTC6DLLx64.dll`) unterstützen die RTC6 PCIe-Karte unter den 32-Bit- und 64-Bit-Microsoft-Betriebssystemen Windows 10, 8, 7. Die DLLs stellen alle benötigten Funktionen für den Betrieb der RTC6 PCIe-Karte zur Verfügung.

Die Programmierung von Anwenderprogrammen wird in Kapitel 6 "RTC6-Anwenderprogramme entwickeln", Seite 96, ausführlich beschrieben. Kapitel 6.2 "Initialisierung und Programmstart", Seite 98 zeigt, wie Funktionen der RTC6-DLL in Anwenderprogramme importiert werden, wenn sie in Pascal, C, C++ oder C# geschrieben werden.

Die 64-Bit-Variante des RTC6-Kartentreibers unterstützt auf einem 64-Bit-Betriebssystem sowohl Funktions-Aufrufe aus Win64- als auch aus Win32-Anwenderprogrammen.

Bereits existierende Win32-Anwenderprogramme zur RTC6 PCIe-Karte sind damit auch auf 64-Bit-Systemen ausführbar, wenn die im Paket enthaltene Win32-basierte `RTC6DLL.dll` verwendet wird.

Für Win64-Anwenderprogramme ist im Paket die Win64-basierte `RTC6DLLx64.dll` enthalten. Falls ein Anwenderprogramm zu dieser RTC6-DLL Verbindung durch implizites Linken aufnehmen soll, ist sie mit der Win64-basierten Import-Library `RTC6DLLx64.lib` zu linken.

Um Softwareentwicklern den Einstieg zu erleichtern, sind einige Beispiel-Codes gezeigt in:

- Kapitel 6.2.5 "Beispiel-Code (C)", Seite 103
- Kapitel 6.8.3 "Beispiel-Code (C)", Seite 136
- Kapitel 7.1.4 "Beispiel-Code (C)", Seite 145

6 RTC6-Anwenderprogramme entwickeln

6.1 Grundlegendes zum RTC6 Software-Konzept

6.1.1 Scan-Systeme und Laser ansteuern – Beispiel zur Einführung

Die SCANLAB RTC6 PCIe-Karte und die dazugehörige Software sind für die Ansteuerung von Scan-Systemen und Lasern konzipiert.

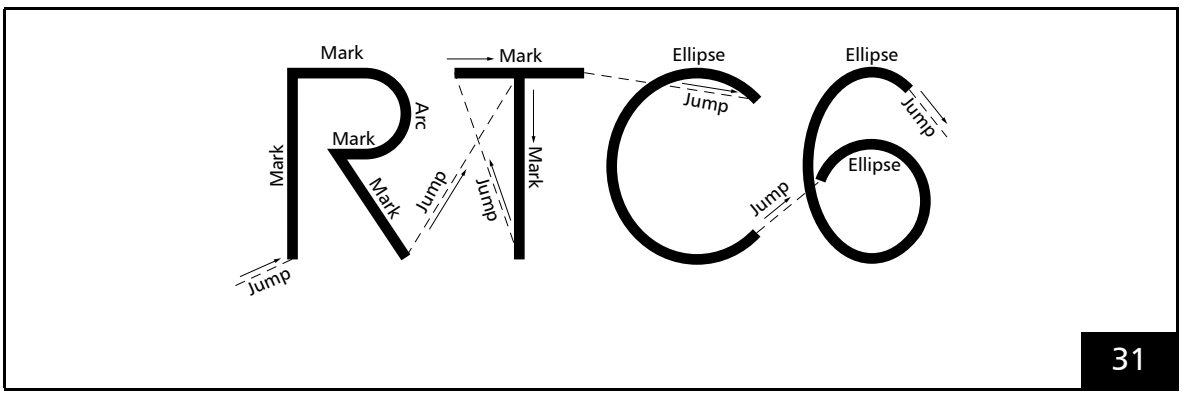
Zur Veranschaulichung des Funktionsprinzips ist in Abbildung 31 ein einfaches Beispiel eines Lasermarkier-Prozesses(1) dargestellt. Der Schriftzug ist aus geraden Liniensegmenten (Vektoren) und Bogensegmenten zusammengesetzt. Zum Abfahren dieser Segmente stellt die RTC6-DLL einen entsprechenden Satz an Jump-, Mark-, Arc- und Ellipse-Befehlen zur Verfügung. Jeder dieser Befehle beschreibt einen einzelnen Vektor oder Bogen. Weitere Befehle ermöglichen die Steuerung des Lasers während des Markierungsprozesses.

Die RTC6 PCIe-Karte bereitet die eingegebenen Befehle in die für die Markierung notwendigen Signale auf und sendet sie nach vorgegebenem Zeitplan präzise an den Laser und (in 10 μs-Intervallen) an den Scan-Kopf. Im Scan-Kopf richten die Galvanometerscanner ihre Ablenkspiegel entsprechend der empfangenen Steuersignale exakt aus.

(1) Lasermarkieren steht in diesem Handbuch stellvertretend für weitere Arten von Lasermaterialbearbeitungs-Prozessen.

Auch kann mit entsprechenden Befehlen über die RTC6 PCIe-Karte der aktuelle Zustand eines Scan-Kopfs abgefragt werden.

Für die Ansteuerung des Lasers stellt die RTC6 PCIe-Karte verschiedene analoge und digitale Signal-Ausgänge zur freien Verfügung.



Beispiel für eine Lasermarkierung.

6.1.2 Kontrollbefehle und Listenbefehle

Der RTC6-Befehlssatz besteht aus:

- Kontrollbefehlen
- Listenbefehlen

Kontrollbefehle werden unmittelbar ausgeführt. Sie werden z.B. zur Initialisierung, zum Steuern der Abarbeitung von Listen, zur Einstellung von allgemeinen Parametern oder zur direkten Laser- und Scan-Kopf-Steuerung eingesetzt.

Bevor *Listenbefehle* an die RTC6 PCIe-Karte übergeben werden können, muss mit einem Kontrollbefehl der Eingabe-Zeiger definiert werden, an den die nachfolgenden Listenbefehle übergeben werden. Das entspricht dem Eröffnen einer Liste, siehe Kapitel 6.4.1 "Listen laden", Seite 108.

Listenbefehle, die danach an die RTC6 PCIe-Karte übergeben werden, werden nicht sofort ausgeführt. Sie werden zunächst im RTC6-Listenspeicher zwischengespeichert. Erst nach Starten der Liste werden sie in Echtzeit abgearbeitet.

Zu den Listenbefehlen zählen sowohl Sprung-Befehle, [*]mark[*]-Befehle und Bogen-Befehle als auch Befehle zur Einstellung verschiedener Scan-Parameter wie der Laserleistung, der Sprunggeschwindigkeit und Markiergeschwindigkeit.

Einige der Listenbefehle sind sogenannte *kurze Listenbefehle*. Sie benötigen zur Befehlsausführung keinen vollen 10 µs-Takt. Sie werden mit dem jeweils nachfolgenden Listenbefehl unmittelbar nacheinander innerhalb eines 10 µs-Takts ausgeführt. Ein Kontrollbefehl kann dazwischen nicht ausgeführt werden. Die gesamte Listenausführungszeit verringert sich dadurch.

Kurze Listenbefehle sind z.B. **list_jump_pos**, **list_call** etc. Bei einem Polygonzug kann z.B. mittels des kurzen Listenbefehls **write_da_x_list** für jeden Vektor separat die Laserleistung eingestellt werden, ohne ihn zu unterbrechen (der Laser bleibt an).

Manche Befehle existieren in zwei Versionen: als Listenbefehl und als Kontrollbefehl. Dazu gehören etwa die I/O-Befehle.

Alle RTC6-Befehle werden ausführlich in Kapitel 10 "RTC6-Befehle" beschrieben.

6.2 Initialisierung und Programmstart

Zur Verwendung der Befehle und Funktionen der RTC6 PCIe-Karte in einem Anwenderprogramm

- muss eine vollständige Installation – der RTC6-Hardware, des RTC6-Kartentreibers und der RTC6-Software – durchgeführt sein, siehe Kapitel 5 "Installation und Inbetriebnahme", Seite 91,
- muss die gewünschte RTC6-DLL (Win32- oder Win64-basiert) vom Anwenderprogramm eingebunden werden, siehe Kapitel 6.2.1 "DLL-Aufrufkonvention", Seite 98 und
- müssen die Funktionen der RTC6-DLL in das Anwenderprogramm importiert werden, siehe Kapitel 6.2.2 "Befehle importieren", Seite 98.

Zu Beginn jedes Anwenderprogramms müssen Befehle aufgerufen werden, mit denen

- die RTC6-DLL für das aufrufende Anwenderprogramm initialisiert wird und die Zugriffsberechtigungen für die installierten RTC6 PCIe-Karten vergeben werden, siehe Kapitel 6.2.3 "Initialisierung der RTC6-DLL und RTC6-Kartenverwaltung", Seite 100, und
- die installierten RTC6 PCIe-Karten selbst initialisiert werden, siehe Kapitel 6.2.4 "Inbetriebnahme der RTC6 PCIe-Karte", Seite 101.

Erst danach kann das Anwenderprogramm die gewünschten Befehlslisten auf den RTC6-Listenspeicher laden und zur Abarbeitung starten.

Die aufgelisteten Schritte werden in den folgenden Kapiteln im Einzelnen erläutert. Sie sind in Kapitel 6.2.5 "Beispiel-Code (C)", Seite 103 in einem Beispielprogramm dargestellt.

6.2.1 DLL-Aufrufkonvention

Binden Sie die `RTC6DLL.dll`/ `RTC6DLLx64.dll`(1) in das Anwenderprogramm ein.

Die DLL-Aufrufkonvention ist `stdcall` für Win32 und `fastcall` für Win64.

Das Struktur-Alignment ist 4 Byte für Win32 und 8 Byte für Win64.

6.2.2 Befehle importieren

Um den Import von Befehlen aus der RTC6-DLL in eine C, C++, C# oder Pascal-Anwendung zu ermöglichen, enthält das RTC6-Software-Paket entsprechende Hilfsdateien.

Achtung!

- Einige RTC6-Befehle können nur mit entsprechenden Optionen *vollständig* ausgeführt werden, siehe Kapitel 2.6 "Optionen", Seite 39.
- Mit **get_rtc_version** können Sie Informationen zu den auf Ihrer RTC6 PCIe-Karte installierten Optionen abrufen.

(1) Siehe Kapitel 5.4 "RTC6-Software installieren", Seite 93.

Pascal

Verwenden Sie die Datei `RTC6Import.pas` als eine Unit und rufen Sie die benötigten RTC6-Befehle auf, wie

```
goto_xy(1000, 2500);
```

für einen Sprung zur Position 1000, 2500.

C

In C können Sie zwischen implizitem Linken (oder statischem Laden oder load-time dynamic linking) und explizitem Linken (oder dynamischem Laden oder run-time dynamic linking) wählen.

Implizites Linken

Zum impliziten Linken fügen Sie die Kopfdatei `RTC6impl.h` ein und linken Sie (für Win32-Anwenderprogramme) mit der (C) Import-Bibliothek `RTC6DLL.lib` (oder für Win64-Anwenderprogramme mit `RTC6DLLx64.lib`), um eine ausführbare Datei zu erzeugen.

Rufen Sie die benötigten RTC6-Befehle auf wie

```
goto_xy(1000, 2500);
```

für einen Sprung zur Position 1000, 2500.

Explizites Linken

Zum expliziten Linken fügen Sie die Kopfdatei `RTC6expl.h` ein. Bevor Sie irgend einen RTC6-Befehl aufrufen, initialisieren Sie die RTC6-DLL durch Aufruf der Funktion `RTC6open` (die in der Datei `RTC6expl.c` definiert wird). Wenn Sie die RTC6-Sitzung beendet haben, dann schließen Sie die RTC6-DLL durch Aufruf der Funktion `RTC6close` (wird auch in der Datei `RTC6expl.c` definiert). Eine ausführbare Datei erzeugen Sie durch Linken mit der Datei `RTC6expl.obj`, die Sie aus dem Quellcode der Datei `RTC6expl.c` generieren können.

Rufen Sie die benötigten RTC6-Befehle auf wie

```
goto_xy(1000, 2500);
```

für einen Sprung zur Position 1000, 2500.

Unterschiede von Implizitem/Explizitem Linken

	Implizites Linken	Explizites Linken
Benötigte Dateien	`RTC6impl.h` oder `RTC6impl.hpp`, `RTC6DLL.lib` oder `RTC6DLLx64.lib`	`RTC6expl.h`, `RTC6expl.c`
Vorteile	Einfachste Linkmethode.	Das Linken der Anwendung mit einer Import-Bibliothek ist unnötig.
Nachteile	Der Benutzer muss die Anwendung mit einer Compiler-spezifischen Import-Bibliothek linken.	Der Benutzer muss die RTC6-DLL initialisieren (`RTC6open`) und schließen (`RTC6close`).

C++

Wenn Sie unter C++ für die Werterückgabe an Funktionsparameter Referenzen statt Zeiger verwenden wollen, z.B. `UINT&Pos` statt `UINT*Pos`, dann verwenden Sie die Datei `RTC6impl.hpp` statt `RTC6impl.h`. Der Import von Befehlen erfolgt ansonsten wie in C (s.o.).

C#

Zum impliziten Linken der RTC6-DLL unter C# wird die Wrapper-Klasse zur Verfügung gestellt. Sie unterstützt auch die Option 'Any CPU' zur gleichzeitigen Verwendung für 32-Bit- und 64-Bit-Programme.

6.2.3 Initialisierung der RTC6-DLL und RTC6-Kartenverwaltung

Es können so viele RTC6 PCIe-Karten in einem PC gleichzeitig betrieben werden, wie es der PCIe-Bus erlaubt. Die RTC6-Kartenverwaltung kann jedoch maximal 255 RTC6 PCIe-Karten und RTC6 Ethernet-Karten gleichzeitig verwalten, siehe Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.

Die RTC6-DLL erlaubt sowohl Multi-Processing(1) als auch Multi-Threading.

Ein und dieselbe RTC6 PCIe-Karte kann nicht gleichzeitig von mehreren Anwenderprogrammen verwendet werden. Zugriffsberechtigungen (auch zeitweise) auf die vorhandenen Karten werden von der RTC6-DLL exklusiv vergeben.

Mehrere Threads *eines* Anwenderprogramms können zwar ein und dieselbe Karte verwenden, aber nicht gleichzeitig Befehle senden (die RTC6-DLL serialisiert die Befehlsaufrufe automatisch).

Die RTC6-Kartenverwaltung, siehe Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008, muss zu Beginn jedes Anwenderprogramms durch **init_rtc6_dll** initiiert werden, damit überhaupt eine RTC6 PCIe-Karte ansprechbar ist. Das gilt auch, wenn nur *eine* RTC6 PCIe-Karte von nur *einem* Anwenderprogramm verwendet werden soll.

init_rtc6_dll führt folgendes aus (zu weiteren Details siehe Befehlsbeschreibung):

- Sucht nach sämtlichen vorhandenen RTC6 PCIe-Karten
- Baut die RTC6-Kartenverwaltung auf
- Vergibt dem Anwenderprogramm automatisch die Zugriffsberechtigung für die gefundenen Karten (wenn die Zugriffsberechtigung nicht bereits an ein anderes Anwenderprogramm vergeben wurde)
- Vergibt RTC6-DLL-interne Nummern für alle gefundenen RTC6 PCIe-Karten (wichtig für Multi-Board-Befehle)
- Macht eine der Karten zur *aktiven* Karte und damit zum Ziel für Nicht-Multi-Board-Befehle
- Sucht *nicht* nach RTC6 Ethernet-Karten. Diese müssen separat hinzugefügt werden, siehe Kapitel 16 "Anhang A: RTC6 Ethernet-Karte", Seite 978.

acquire_rtc, **release_rtc**, **select_rtc** ermöglichen eine Änderung der Zugriffsberechtigungen und einen Wechsel der aktiven Karte zur Laufzeit.

Die Verwendung mehrerer Karten wird in Kapitel 6.6 "Mehrere RTC6 PCIe-Karten in einem PC verwenden", Seite 126, die Verwendung durch mehrere Anwenderprogramme in Kapitel 6.7 "RTC6 PCIe-Karten durch mehrere Anwenderprogramme verwenden", Seite 131 beschrieben.

Nach der RTC6-DLL-Initialisierung mit **init_rtc6_dll** ist der Default-RTC6-DLL-Betriebsmodus der RTC6-Standardmodus. Bei Bedarf kann ein anderer RTC6-DLL-Betriebsmodus eingestellt werden (siehe **set_rtc4_mode**, **set_rtc5_mode** und **set_rtc6_mode**).

Der RTC4-Kompatibilitätsmodus wird zur Verfügung gestellt, damit Anwenderprogramme, die für die RTC4 geschrieben sind, auch von der RTC6 PCIe-Karte (für ansonsten unveränderte Befehle(2)) ohne Programmcode-Änderung abgearbeitet werden können. In den Befehlsbeschreibungen, Seite 346 wird auf diesbezügliche Änderungen hingewiesen, siehe Zeilen "RTC4→RTC6".

Siehe auch Kapitel 2.10.2 "RTC4-Quellcodes für die RTC6 PCIe-Karte überarbeiten", Seite 50.

Analoges gilt für den RTC5-Kompatibilitätsmodus, siehe Zeilen "RTC5→RTC6" der Befehlsbeschreibungen, Seite 346. Siehe auch Kapitel 2.12.2 "RTC5-Quellcode für die RTC6 PCIe-Karte anpassen", Seite 62.

(1) Es können mehrere Anwenderprogramme gleichzeitig laufen.

(2) Befehle mit identischen Namen mit sinngemäßer Parametrierung und Bedeutung. Beispiel: `mark_abs( X, Y )`.

6.2.4 Inbetriebnahme der RTC6 PCIe-Karte

Am Anfang eines jeden RTC6-Anwenderprogramms sollten – nach Initialisierung der RTC6-DLL, siehe Kapitel 6.2.3 "Initialisierung der RTC6-DLL und RTC6-Kartenverwaltung", Seite 100 – die folgende Schrittabfolge ausgeführt werden, um die RTC6 PCIe-Karte in Betrieb zu nehmen:

(1) Karte initialisieren, Seite 101

(2) Karte konfigurieren, Seite 101

(3) Scan-System-Ansteuerung initialisieren, Seite 102

(4) Laser-Steuerung initialisieren, Seite 102

(5) Listen laden und abarbeiten, Seite 102

Karte initialisieren

- Rufen Sie **load_program_file** auf. Zu den einzelnen Vorgängen, siehe Funktion, Seite 566. Nach der Ausführung von **load_program_file** ist die Positionsausgabe auf den Nullpunkt (0|0)(1) eingestellt und die Laser-Steuerung ist *deaktiviert*(2)(3).

  Um die RTC6-Dateien (`RTC6DLL.dll`/`RTC6DLLx64.dll`, `RTC6OUT.out`/`RTC6ETH.out`, `RTC6RBF.rbf`, `RTC6DAT.dat`) miteinander kombinieren zu können, müssen sie bestimmte Dateiversionen haben, siehe auch Kapitel 5.4 "RTC6-Software installieren", Seite 93. **load_program_file** führt eine entsprechende Versionsprüfung durch. Liegt ein Versionsfehler vor (Fehlercode 7 und **get_last_error**-Returncode `RTC6_VERSION_MISMATCH`), wird die Karte wie mit **release_rtc** freigegeben. Sie ist damit für weitere RTC6-Befehle gesperrt, außer für solche, die keine Zugriffsberechtigung erfordern.

  RTC6 PCIe-Karten können nur in Betrieb genommen werden, wenn alle RTC6-Dateien in einer zulässigen Versionskombination verfügbar sind, siehe Kapitel 5.4 "RTC6-Software installieren", Seite 93. Wenn die Versionsprüfung fehlschlägt und die Karte nicht von einem anderen Anwenderprogramm belegt ist, kann jederzeit auch mit **load_program_file** eine korrekte Programmversion geladen werden. Anschließend kann die Karte mit **select_rtc** oder **acquire_rtc** in Besitz genommen werden.

  Wenn mehrere RTC6 PCIe-Karten Master/Slave verbunden sind, siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.

Karte konfigurieren

- Konfigurieren Sie mit **config_list** bei Bedarf den RTC6-Listenspeicher.
  Nach **load_program_file** können dann aufnehmen, siehe Abbildung 32:
  - Der RTC6-Listenspeicher-Bereich "Liste 1" 4.194.304 Listenbefehle
  - Der RTC6-Listenspeicher-Bereich "Liste 2" 4.194.304 Listenbefehle
  - Der geschützte RTC6-Listenspeicher-Bereich "Liste 3" 0 Listenbefehle

(1) Zentrum des Bildfelds.

(2) An den entsprechenden Pins liegen keine Lasersteuersignale (LASERON, LASER1, LASER2) an. Sie sind im hochohmigen Zustand.

(3) Zum Zustand der verschiedenen Ausgänge siehe Seite 567.

Scan-System-Ansteuerung initialisieren

(1) Laden Sie mit **load_correction_file** die benötigten Korrekturdatei(en) auf die RTC6 PCIe-Karte (das Laden einer Korrekturtabelle kann vor oder nach **load_program_file**, sollte aber unbedingt vor **select_cor_table** erfolgen; es sollte zumindest eine 1:1-Korrekturdatei geladen werden). Informationen über die Verwendung mehrerer verschiedener Korrekturtabellen finden Sie in Kapitel 8.5 "2D-Scan-Systeme und 3D Scan-Systeme steuern", Seite 248.

(2) Weisen Sie mit **select_cor_table** die zuvor geladenen Korrekturtabelle(n) dem (oder den) Scan-Kopf-Steuerungsport(s) zu(1). Dadurch wird die vorgesehene Bildfeld-Korrektur auch für die durch **load_program_file** zuvor voreingestellte Position (0|0) wirksam (u.U. wird auch der Nullpunkt durch die Bildfeld-Korrektur verschoben).
Nach **load_program_file** erfolgt standardmäßig die folgende Zuweisung:
- Korrekturtabelle #1 an Anschluss für ersten Scan-Kopf
- *keine* Korrekturtabelle an Anschluss für zweiten Scan-Kopf

Die vorgesehene Bildfeld-Korrektur wird allerdings erst durch einen nachfolgenden **select_cor_table**, **select_cor_table_list**, Sprung-Befehl oder Markier-Befehl wirksam.

(3) Definieren Sie mit **set_delay_mode** den Scanner-Delay-Modus (u.a. variables oder konstantes Polygon-Delay).

(4) Laden Sie falls nötig(2) mit **load_varpolydelay** eine Tabelle für das "Variable Polygon-Delay".

Verbleibende Größen (wie Scanner-Delays, Sprunggeschwindigkeit und Markiergeschwindigkeit) werden mit weiteren Kontrollbefehlen oder Listenbefehlen festgelegt.

(1) **load_correction_file** ruft nach dem Laden einer Korrekturtabelle automatisch **select_cor_table** mit den zuletzt zugewiesenen Tabellennummern auf (wenn **load_correction_file** nach **load_program_file** folgt), siehe Abschnitt "Hinweise", Seite 184.

(2) Schritt 4 kann entfallen, wenn mit **create_dat_file** eine neue `RTC6DAT.dat` nach dem Laden der Tabelle erzeugt wird. Dann wird diese Tabelle beim nächsten **load_program_file** automatisch geladen.

Laser-Steuerung initialisieren

(1) Stellen Sie mit **set_laser_mode** den Laser-Modus ein.

(2) Stellen Sie mit **set_laser_control** die Polarität der Lasersteuersignale passend zu Ihrem Lasersystem ein.

(3) Stellen Sie mit **set_firstpulse_killer** die Länge des FirstPulseKiller-Signals ein (nur bei einem YAG-Modus).

(4) Stellen Sie mit **set_qswitch_delay** das Delay für das Q-Switch-Signal ein (nur bei einem YAG-Modus, insbesondere bei YAG-Modus 5).

(5) Stellen Sie mit **set_standby** Werte für die Standby-Pulse ein (insbesondere bei Laser-Mode 4 und Laser-Mode 6).

(6) Geben Sie die Lasersteuersignale frei (siehe **enable_laser**), wenn sie bei **set_laser_control** unterdrückt wurden.

Verbleibende Größen (wie Laser-Timing oder Laser-Delays) werden mit weiteren Kontrollbefehlen oder Listenbefehlen festgelegt, siehe z.B. Kapitel 7.1.4 "Beispiel-Code (C)", Seite 145 oder Abschnitt "Signale für den "Laser active"-Betrieb", Seite 194.

Listen laden und abarbeiten

(1) Laden Sie die Liste(n).

(2) Aktivieren Sie falls nötig mit **set_control_mode** die externe Start-Eingabe.

Achtung!

- Überprüfen Sie vor einem Start sorgfältig Ihr Anwenderprogramm. Programmierfehler können einen Systemabsturz verursachen. In diesem Fall lassen sich weder Laser noch Scan-Kopf steuern.

6.2.5 Beispiel-Code (C)

Der grundsätzliche Aufbau eines Anwenderprogramms zur Initialisierung der RTC6-DLL und der RTC6 PCIe-Karte ist im Folgenden beispielhaft als C-Quellcode für eine Konsolenanwendung (Win32-Umgebung) dargestellt (für komplette Demoprogramme siehe Kapitel 11 "Demo-Programme", Seite 967).

Benötigte Dateien: RTC6impl.h, RTC6DLL.lib (bei implizitem Linken) oder RTC6expl.h, RTC6expl.c (bei explizitem Linken) zum Einbinden der RTC6-DLL ins Programm, siehe Kapitel 6.2.1 "DLL-Aufrufkonvention", Seite 98. Falls das Betriebssystem die RTC6DLL.dll beim Anwenderprogramm-Startup nicht findet, antwortet es mit einer entsprechenden Fehlermeldung und beendet das Programm.

```
// System-Kopfdateien
#include <windows.h>
#include <stdio.h>
#include <conio.h>

// RTC6 Kopfdatei zum impliziten Linken zur RTC6DLL.dll (linken Sie zusätzlich mit der (Visual C++)
// Import-Bibliothek RTC6DLL.lib, um eine ausführbare Datei zu erzeugen):
#include "RTC6impl.h"
// Alternativ: RTC6 Kopfdatei zum expliziten Linken zur RTC6DLL.dll (linken Sie zusätzlich mit der Datei
// RTC6expl.obj, die Sie aus dem Quellcode der Datei RTC6expl.c generieren können, um eine ausführbare
// Datei zu erzeugen):
//#include "RTC6expl.h"

void __cdecl main( void*, void* )
{

   // nur bei explizitem Linken:
   // if ( RTC6open() ) // Fehler detektiert (RTC6open gibt 0 zurück, wenn kein Fehler auftritt)
   // {
   // printf( "Error: RTC6DLL.dll not found\n" );
   // terminateDLL();
   // return;
   // }

   printf( "Initializing the DLL\n\n" );

   UINT ErrorCode;

   // Initialisierung der RTC6-DLL (der folgende Befehl sollte als erster RTC6-Befehl aufgerufen werden)
   ErrorCode = init_rtc6_dll();

   // Direkt auf init_rtc6_dll sollte ein Programmteil eingebaut werden, mit dem ein Fehler
   // während der Initialisierung aufgefangen wird, z.B. für den Fall, dass die gewünschte Karte
   // nicht gefunden wird, wenn der Zugriff verweigert wird oder wenn ein weiterer Fehler (z.B. ein
   // VersionMismatch-Fehler) auftritt. Siehe Kapitel 6.8.3 "Beispiel-Code (C)", Seite 136.
```

```
   // Initialisierung der RTC6 PCIe-Karte:
   // - Auswählen der Karte Nummer 1 als die aktive Karte in diesem Anwenderprogramm.
   // - Falls gewünscht: Auswählen des RTC4-Kompatibilitätsmodus als Betriebsmodus
   //   (Voreinstellung: RTC6-Standardmodus).
   // - Optional: stoppen einer eventuellen Listenausführung auf der RTC6 PCIe-Karte Nummer 1
   //   (falls diese zuvor durch ein anderes Anwenderprogramm verwendet wurde, könnte noch eine Liste
   //   ausgeführt werden). load_program_file tut dies ansonsten automatisch selbst.
   //   load_correction_file kann sonst nicht vor load_program_file ausgeführt werden.
   // - Aufrufen von load_program_file zum Initialisieren der Karte, zum Laden der Programmdatei, etc.
   //   (auch hier sollten eventuell auftretende Fehler – z.B. Datei- oder System-Fehler – durch einen
   //   geeigneten Programmcode aufgefangen werden, siehe Kapitel 6.8.3 "Beispiel-Code (C)", Seite 136).
   // - Zurücksetzen aller vorherigen Fehlercodes.
   //   (stop_execution könnte einen RTC6_TIMEOUT- oder RTC6_BUSY-Fehler verursacht haben).
   // - Konfigurieren des RTC6-Listenspeichers. Voreinstellung: je 4.194.304 Speicherplätze für Liste
1 und
   //   Liste 2).
   (void) select_rtc( 1 );
   set_rtc4_mode();

   // Optional: stop_execution();

   ErrorCode = load_program_file( 0 ); // Path = 0: Pfad des aktuellen Arbeitsverzeichnisses
   if ( ErrorCode )
   {
      printf( "Program file loading error: %d\n", ErrorCode );
      free_rtc6_dll();
      return;
   }
   reset_error( -1 );
   config_list( 4000, 4000 );
   // Nachfolgend kann nun der Programmcode zur Definition des Laser-Scanprozesses
   // eingefügt werden. Ein Beispiel-Code ist in Kapitel 7.1.4 "Beispiel-Code (C)", Seite 145.

   // Ende des Hauptprogramms
   terminateDLL();
   return;
}

void terminateDLL()
{
   printf( "- Press any key to shut down \n" );
   while( !kbhit() );
   (void) getch();
   printf( "\n" );
   // Schließen der RTC6-DLL
   free_rtc6_dll();
   // nur bei explizitem Linken:
   // RTC6close();
}
```

6.3 RTC6-Listenspeicher

Der RTC6-Listenspeicher dient zum Zwischenspeichern von Listenbefehlen.

Bevor Listenbefehle an die RTC6 PCIe-Karte übergeben werden, muss mit einem Kontrollbefehl der Eingabe-Zeiger definiert werden, an den die nachfolgenden Listenbefehle übergeben werden. Dies entspricht dem Eröffnen einer Liste, siehe Kapitel 6.4.1 "Listen laden", Seite 108.

Durch weitere Kontrollbefehle kann das Abarbeiten der übergebenen Listenbefehle gestartet werden.

6.3.1 Listen und geschützter RTC6-Listenspeicher-Bereich

Der RTC6-Listenspeicher umfasst insgesamt 8 M = 8.388.608 = 2^{23} Speicherplätze.

Er kann grundsätzlich in 3 Bereiche aufgeteilt werden. Deren Größen sind frei einstellbar.

- 2 RTC6-Listenspeicher-Bereiche "Liste 1" und "Liste 2". In diesem Handbuch werden diese auch einfach als "Listen" bezeichnet.
- Benutzerdefinierbar ist zusätzlich: ein dritter "geschützter RTC6-Listenspeicher-Bereich", "Liste 3". Dieser ist vor einem versehentlichen Überschreiben (durch das Laden normaler Befehlslisten) geschützt.

"Liste 1" und "Liste 2"

Die beiden RTC6-Listenspeicher-Bereiche "Liste 1" und "Liste 2" können grundsätzlich identisch verwendet werden wie die beiden RTC6-Listenspeicher-Bereiche der RTC5-Karten oder RTC4-Karten, etwa für das kontinuierliche Laden und Abarbeiten von Befehlslisten.

Der RTC6-Listenspeicher ist zudem größer. Außerdem kann die Größe der einzelnen RTC6-Listenspeicher-Bereiche frei konfiguriert werden, siehe Kapitel 6.3.2 "RTC6-Listenspeicher konfigurieren", Seite 106.

Zum Listen-Handling siehe Kapitel 6.4 "Listen-Handling", Seite 108.

"Liste 3" – Geschützter RTC6-Listenspeicher-Bereich

"Liste 3" ist zur geschützten Speicherung immer wieder benötigter Listenbefehlssequenzen (in Form von Unterprogrammen oder Zeichensatzdefinitionen) vorgesehen. Sie ist vor einem versehentlichen Überschreiben (durch das Laden normaler Befehlslisten) geschützt.

Grundsätzlich gibt es zwei alternative Möglichkeiten, diese Schutzfunktion zu nutzen:

(1) Es werden Unterprogramme in die oberen Speicherplätze des RTC6-Listenspeicher-Bereichs geschrieben und diese Speicherplätze nachträglich dem geschützten RTC6-Listenspeicher-Bereich "Liste 3" zugewiesen. Der Aufruf eines solchen Unterprogramms erfolgt – sowohl zu Beginn im RTC6-Listenspeicher-Bereich als auch später im geschützten RTC6-Listenspeicher-Bereich "Liste 3" – mit **list_call** unter Angabe der absoluten Speicherposition.

(2) Unterprogramme oder Zeichensatzdefinitionen werden über spezielle Befehle als indizierte Unterprogramme oder Zeichensätze direkt in den geschützten RTC6-Listenspeicher-Bereich "Liste 3" geladen. Ihr Aufruf erfolgt dann unter Angabe des entsprechenden Indexes. Indizierte Zeichensätze können aber auch zusätzlich beispielsweise mit **mark_text** direkt zum Markieren von Texten verwendet werden.

SCANLAB empfiehlt dringend, die beiden Nutzungsarten *nicht* miteinander zu vermischen. Anderenfalls kann es auch im geschützten RTC6-Listenspeicher-Bereich "Liste 3" zu einem unbeabsichtigten Datenverlust durch Überschreiben kommen.

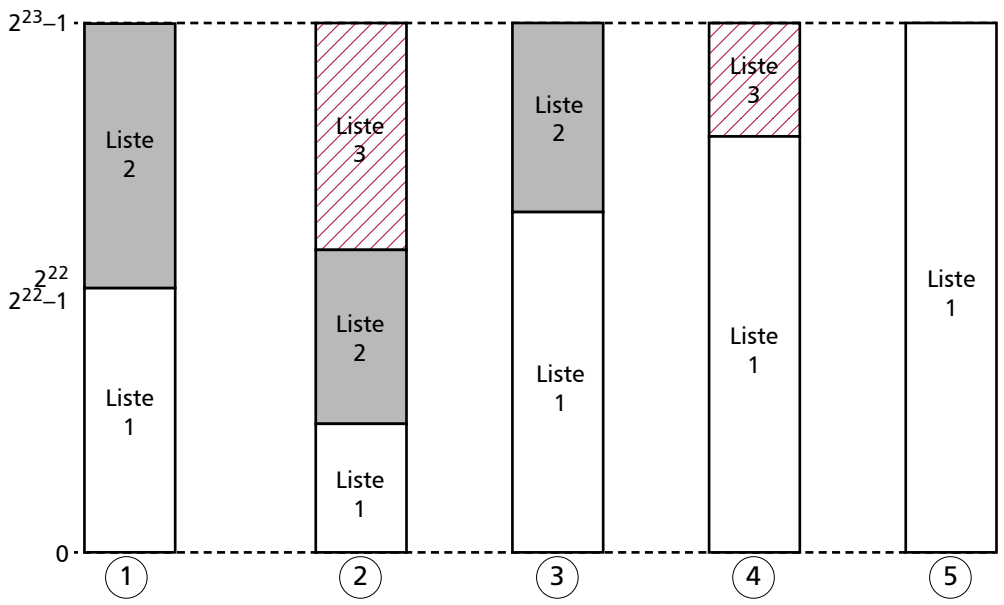
Damit Benutzer nicht dauerhaft auf die eine oder die andere Nutzungsart festgelegt sind, enthält der RTC6-Befehlssatz entsprechende Konvertierungs-Befehle. Die Definition von Unterprogrammen und Zeichensätzen, deren Verwaltung und die Möglichkeiten zur nachträglichen Konvertierung sind ausführlich in Kapitel 6.5 "Strukturierte Programmierung", Seite 115 beschrieben.

6.3.2 RTC6-Listenspeicher konfigurieren

Der RTC6-Listenspeicher umfasst insgesamt 8 M = 8.388.608 = 2^{23} Speicherplätze. Nach **load_program_file** können die RTC6-Listenspeicher-Bereiche “Liste 1” und “Liste 2” jeweils 4.194.304 Listenbefehle aufnehmen, der geschützte RTC6-Listenspeicher-Bereich “Liste 3” kann keine Listenbefehle aufnehmen, siehe Abbildung 32.

Mit **config_list** können die RTC6-Listenspeicher-Bereiche umkonfiguriert werden.

Benötigt ein Anwenderprogramm nur *eine einzige* Liste, dann kann der gesamte RTC6-Listenspeicher auch wie ein einzelner RTC6-Listenspeicher mit einer Gesamtkapazität von 2^{23} Speicherplätzen behandelt werden, siehe Konfiguration (5) in Abbildung 32.



Legende

Links: Nummern der begrenzenden Speicherplätze der zwei RTC6-Listenspeicher-Bereiche “Liste 1” und “Liste 2”.

1 Standardkonfiguration (= ohne geschützten RTC6-Listenspeicher-Bereich “Liste 3”). Entspricht:
`config_list( 4.194.304, 4.194.304 )`

2 `config_list( Mem1, Mem2 )` mit `Mem1` > 0, `Mem2` ≤ 2^{23} – `Mem1`

3 `config_list( Mem1, -1 )` mit `Mem1` > 0

4 `config_list( Mem1, 0 )` mit `Mem1` > 0

32

Beispiele für zulässige RTC6-Listenspeicher-Konfigurationen.

Allgemein werden bei der Konfiguration zugeordnet:

- niedrigere Speicherplatznummern der “Liste 1”
- mittlere Speicherplatznummern der “Liste 2”
- höhere Speicherplatznummern der “Liste 3”

Bei der Konfiguration der RTC6-Listenspeicher-Bereiche ist zu beachten:

- “Liste 1” muss mindestens *einen* Speicherplatz enthalten
- die Summe der Speicherplätze von “Liste 1” und “Liste 2” darf die Zahl 2^{23} nicht überschreiten.

Ansonsten können die Größen von “Liste 1” und “Liste 2” beliebig eingestellt werden. Beispielsweise kann die “Liste 2” auch mit 0 Speicherplätzen konfiguriert werden, siehe Konfiguration (4) und (5) in Abbildung 32.

Bei jeder Konfiguration werden dem geschützten RTC6-Listenspeicher-Bereich “Liste 3” automatisch die restlichen (nicht an “Liste 1” oder “Liste 2” vergebenen) Speicherplätze zugeteilt.

Der Speicherinhalt wird durch den Konfigurationsablauf nicht geändert. Daher ist ein wiederholter Aufruf mit verschiedenen Parametern unschädlich.

Beim Ändern der Konfiguration sollte aber auch das Folgende beachtet werden:

- Listengrenzen sollten nicht so verschoben werden, dass sie hinterher mitten durch ein Unterprogramm verlaufen.
- Der Schutz für einen “Liste 3”-Bereich wird aufgehoben, wenn dieser Bereich dem RTC6-Listenspeicher-Bereich “Liste 1” oder “Liste 2” zugeordnet wird.
- Gültige Sprungadressen in Sprung-Befehlen werden u.U. durch Ändern der Konfiguration ungültig, siehe Kapitel 6.5.3 “Sprünge”, Seite 123.
- Nachdem der geschützte RTC6-Listenspeicher-Bereich “Liste 3” vergrößert wurde, ist eventuell eine Defragmentierung erforderlich, damit der neu zugewiesene RTC6-Listenspeicher-Bereich auch nutzbar ist, siehe Abschnitt “Indexverwaltung und Defragmentierung”, Seite 118.

6.4 Listen-Handling

Die beiden RTC6-Listenspeicher-Bereiche “Liste 1” und “Liste 2” dienen als Zwischenspeicher für ein kontinuierliches Laden und Abarbeiten von Listenbefehlen.

6.4.1 Listen laden

“Liste 1” und “Liste 2” werden mit **set_start_list_pos**, **load_list** oder weiteren Kontrollbefehlen (siehe unten) zum Beschreiben mit Listenbefehlen freigegeben. Dabei wird für die ausgewählte Liste ein Eingabe-Zeiger definiert. Dieser Eingabe-Zeiger gibt die Speicherposition an, an die die nachfolgenden Listenbefehle übertragen werden.

Listen sind in sich geschlossene Blöcke von Speicherplätzen für Listenbefehle. Wird beim Laden einer Liste das Listenende erreicht und der Eingabe-Zeiger wird nicht auf die jeweils andere Liste gesetzt, dann wird der Eingabe-Zeiger automatisch auf den Anfang der aktuellen Liste zurückgesetzt. Das Laden wird dann dort fortgesetzt.

Es erfolgt niemals ein automatischer Wechsel des Eingabe-Zeigers in eine andere Liste, insbesondere nicht in den geschützten RTC6-Listenspeicher-Bereich “Liste 3”.

Grundsätzlich werden beim Laden von Listenbefehlen auf die Speicherplätze die dort zuvor gespeicherten Listenbefehle überschrieben. Das erfolgt auch, wenn diese noch nicht abgearbeitet oder sogar momentan ausgeführt werden. Benutzer sollten sich vergewissern, dass keine Befehle überschrieben werden, die noch im Anwenderprogramm gebraucht werden (s.u.).

Der PCI-Transfer der Listenbefehle in den RTC6-Listenspeicher ist gepuffert, um die Geschwindigkeit für einen kontinuierlichen Download zu erhöhen. Der Puffer ist 16 Befehle groß.

Immer wenn der Puffer voll ist oder wenn **set_end_of_list**, **list_return**, **set_input_pointer** (und verwandte Befehle), **execute_list_pos** (und verwandte Befehle), **auto_change**, **auto_change_pos**, **start_loop** oder **release_rtc** aufgerufen werden, wird automatisch ein Flush ausgeführt. Damit werden die noch gepufferten Listenbefehle in den RTC6-Listenspeicher transferiert.

Mit `set_input_pointer(get_input_pointer())` kann zu jeder Zeit auch für einen “unfertigen” Puffer ein Flush initiiert werden. Dies ist aber u.U. nur dann erforderlich, wenn Listenbefehle abgearbeitet werden sollen und die Listeneingabe noch nicht abgeschlossen ist (z.B. bei einem externen Start).

“Unbedingtes” Laden

Der Eingabe-Zeiger wird gesetzt:

- an den Anfang der ausgewählten Liste durch
  - **set_start_list**
  - **set_start_list_1**
  - **set_start_list_2**
- an die angegebene Adresse der ausgewählten Liste durch
  - **set_start_list_pos**
  - **set_input_pointer**

An diese Adresse wird – unabhängig vom aktuellen Status der angegebenen Liste, siehe Kapitel 6.4.2 “Listenstatus”, Seite 110 – der jeweils nächste Listenbefehl geschrieben.

Bei Bedarf können zuvor mit **get_input_pointer** oder **get_list_pointer** und **get_status** oder **get_out_pointer** die aktuellen Positionen von Input- und Ausgabe-Zeiger ausgelesen werden, z.B. um sicherzustellen, dass keine Listenbefehle überschrieben werden, die noch nicht abgearbeitet sind.

Mit Schutzfunktion laden

Durch **load_list** wird der Eingabe-Zeiger zur Initiierung des Lade-Vorgangs (wie bei **set_start_list_pos**) an die angegebene Adresse der ausgewählten Liste gesetzt. Das erfolgt allerdings nur dann, wenn die ausgewählte Liste nicht aktuell in Verwendung ist.

Alternativ kann man den Eingabe-Zeiger mit **load_list** auch einfach in eine gerade nicht aktive oder bereits abgearbeitete Liste setzen lassen (die jeweils geeignete Liste wird dabei von der RTC6 PCIe-Karte automatisch ermittelt). Der Returnwert von **load_list** gibt zusätzlich darüber Auskunft, ob oder in welcher Liste ein Lade-Vorgang erfolgreich initiiert wurde.

Anderenfalls wird der Eingabe-Zeiger auf eine ungültige Position gesetzt. Dann ist so lange keine Eingabe von weiteren Listenbefehlen möglich, bis der Eingabe-Zeiger wieder auf eine gültigen Position gesetzt ist (z.B. mit Hilfe eines erneuten **load_list** mit positivem Resultat oder mit **set_start_list_pos**).

Dadurch wird automatisch ein unbeabsichtigtes Überschreiben von Befehlen, die noch ausgeführt werden sollen, verhindert.

load_list hilft u.a. beim alternierenden Listenwechsel, um gezielt auf das Abarbeiten einer Liste zu warten, siehe Abschnitt "Alternierender Listenwechsel", Seite 114.

Listen abschließen

Eine Befehlsliste kann, muss aber nicht notwendigerweise mit einem **set_end_of_list** abgeschlossen werden.

Beim Abarbeiten von nicht abgeschlossenen Befehlslisten wird allerdings nach Erreichen der letztmöglichen Position des Ausgabe-Zeigers in dieser Liste automatisch mit dem Anfang derselben Liste fortgefahren.

Ein automatischer Listenwechsel nach Abarbeitung einer Liste wird nur dann durchgeführt, wenn die Liste mit **set_end_of_list** abgeschlossen wurde, siehe Kapitel 6.4.6 "Listen automatisch wechseln", Seite 113.

Durch das Laden eines **set_end_of_list** wird der Ladevorgang selbst *nicht* gestoppt. Unmittelbar auf einen **set_end_of_list** folgende Listenbefehle werden daher weiterhin in die gleiche Liste geladen.

6.4.2 Listenstatus

Listen erhalten – abhängig vom Status der Befehlseingabe und Befehlsausgabe – bestimmte Listenstatuswerte zugeordnet.

Die aktuellen Listenstatuswerte (vgl. mit Kapitel 6.4.3 "Listenausführungsstatus", Seite 111) können mit dem Kontrollbefehl **read_status** – für beide Listen getrennt – abgefragt werden.

- LOAD-Listenstatus
  Der LOAD-Listenstatus LOAD1 oder LOAD2 zeigt an, dass sich der Eingabe-Zeiger aktuell in dieser Liste befindet. Der LOAD-Listenstatus der jeweils anderen Liste ist dann in jedem Fall *nicht* gesetzt.
- READY-Listenstatus
  Der READY-Listenstatus READY1 oder READY2 wird gesetzt, wenn während des Lade-Vorgangs ein **set_end_of_list** in die Liste geschrieben wird. Der READY-Listenstatus wird zurückgesetzt, wenn der LOAD-Listenstatus der Liste neu gesetzt wird.
- BUSY-Listenstatus
  Der BUSY-Listenstatus BUSY1 oder BUSY2 zeigt an, dass sich der Ausgabe-Zeiger aktuell in dieser Liste befindet, nachdem eine Listenausführung (von "Liste 1" oder "Liste 2") gestartet wurde. Der BUSY-Listenstatus der jeweils anderen Liste ist dann in jedem Fall *nicht* gesetzt.
  Der BUSY-Listenstatus einer Liste wird beim Ausführen von **set_end_of_list** zurückgesetzt (oder wechselnd gesetzt, falls zuvor ein automatischer Listenwechsel aktiviert wurde). Wird eine Liste zum Laden geöffnet, während sie abgearbeitet wird, bleibt ihr BUSY-Listenstatus trotzdem noch gesetzt.
- USED-Listenstatus
  Der USED-Listenstatus USED1 oder USED2 wird gesetzt, wenn beim Abarbeiten der Liste ein **set_end_of_list** erreicht wird. Der USED-Listenstatus wird zurückgesetzt, wenn der LOAD-Listenstatus der Liste gesetzt wird.

Hinweise

- Wenn der Listenstatus während der Abarbeitung eines Unterprogramms im geschützten RTC6-Listenspeicher-Bereich "Liste 3" abgefragt wird, dann wird der Status derjenigen Liste "Liste 1" oder "Liste 2" zurückgegeben, in der sich der Ausgabe-Zeiger zuletzt befunden hat (in der Regel von wo aus das Unterprogramm ursprünglich aufgerufen wurde).
- Bei einer Unterbrechung der Listenausführung (mit **pause_list**, **stop_list** oder **set_wait**) bleiben die obigen Statuswerte unverändert.
- Bei einem Abbruch der Listenausführung mit **stop_execution** oder einem externen Stopp wird für beide Listen der USED-Listenstatus gesetzt (wie auch bei der Initialisierung). Siehe auch `load_list( ListNo = 3 )`.
- Wenn Sie (z.B. wegen einer Unterbrechung mit **stop_execution** oder einem externen Stopp) einer Liste `ListNo` wieder explizit den USED-Listenstatus zuweisen wollen, laden Sie an einer unbenutzten Stelle `Pos` dieser Liste durch **set_start_list_pos**(`ListNo`, `Pos`) ein **set_end_of_list** und führen Sie **execute_list_pos**(`ListNo`, `Pos`) aus. Wenn gerade keine andere Liste aktiv war, hat die Liste `ListNo` nun den USED-Listenstatus.
- Bei der Interpretation der mit **read_status** zurückgelesenen Statuswerte sollte immer der jeweils programmierte Lade- oder Ausführungsvorgang der Listen berücksichtigt werden, siehe Befehlsbeschreibung.

6.4.3 Listenausführungsstatus

Zusätzlich zu den Listenstatuswerten, siehe Kapitel 6.4.2 "Listenstatus", Seite 110, werden Listenausführungsstatuswerte bereitgestellt.

Diese können mit dem Kontrollbefehl **get_status** ausgelesen werden.

- BUSY-Listenausführungsstatus
  Der BUSY-Listenausführungsstatus ist gesetzt, wenn:
  – die RTC6 PCIe-Karte gerade eine der beiden Listen abarbeitet
  – eine Liste mit den Kontrollbefehlen **pause_list** oder **stop_list** angehalten wurde
  Der BUSY-Listenausführungsstatus ist *nicht* gesetzt, wenn eine Liste mit dem Listenbefehl **set_wait** angehalten wurde (wird erst wieder durch ein anschließendes **release_wait** gesetzt).
  BUSY-Listenausführungsstatus und INTERNAL-BUSY-Listenausführungsstatus können nicht gleichzeitig gesetzt sein.
- PAUSED-Listenausführungsstatus
  Der PAUSED-Listenausführungsstatus ist gesetzt, wenn das Abarbeiten der Liste mit **pause_list**, **stop_list** oder **set_wait** angehalten wurde. Durch ein anschließendes **restart_list** oder **release_wait** wird der PAUSED-Listenausführungsstatus wieder zurückgesetzt, siehe auch Kapitel 6.4.5 "Listen zur Synchronisation der Ausgabe unterbrechen", Seite 113.
- INTERNAL-BUSY-Listenausführungsstatus
  Der INTERNAL-BUSY-Listenausführungsstatus ist gesetzt, wenn die RTC6 PCIe-Karte mit einem Kontrollbefehl beschäftigt ist, der zur Ausführung einer Scan-Bewegung länger als 10 µs benötigt (z.B. **goto_xy** oder u.U. **set_offset**) oder während (bei **set_wait**, **set_end_of_list** oder **release_wait**) ein Home-Jump oder Home-Return ausgeführt wird (falls zuvor der Home-Jump-Modus mit **home_position** oder **home_position_xyz** aktiviert wurde).
  BUSY-Listenausführungsstatus und INTERNAL-BUSY-Listenausführungsstatus können nicht gleichzeitig gesetzt sein.
- HEAD BUSY-Listenausführungsstatus (Kontrollbefehl **get_status**, Bit #23 = 1)
  Betriebsstatus nur für SCANahead-Systeme.
  Als Voraussetzung muss **set_scanahead_params** mit gültigen Parametern ausgeführt worden sein, d.h. die RTC6-Karte ist zum Ansteuern von SCANahead-Systemen konfiguriert.
  Der HEAD BUSY-Listenausführungsstatus ist gesetzt, wenn:
  – Eine Liste abgearbeitet wird
  – Wenn die Liste beendet ist, die Scan-System-Ausgaben aber noch andauern (Zeitversatz = `PreviewTime`)
  – Das letzte noch wirksame `LaserOffDelay` der Markierung noch nicht abgelaufen ist, d.h. solange der Laser noch eingeschaltet ist
  Dadurch kann das tatsächliche Ende der Markierung abgewartet werden, bevor z.B. bei der Maschinensteuerung ein Shutter geschlossen oder ein Scan-System stromlos gemacht werden muss.
  Wenn dagegen nach dem Listen-Ende, aber noch vor dem Ablauf der `PreviewTime` der Laser manuell mit **laser_signal_on** eingeschaltet wurde, wird dessen LaserOff nicht abgewartet.

Hinweise

- Der BUSY-Listenausführungsstatus ist bei der RTC6 PCIe-Karte als BUSY OUT-Signal verfügbar an der:
  – LASER-Buchse, siehe Abbildung 17
  – EXTENSION 1-Stiftleiste, siehe Abbildung 22
  – MARKING ON THE FLY-Stiftleiste, siehe Abbildung 25
- Einige Kontrollbefehle werden ignoriert (nicht ausgeführt), wenn der BUSY-Listenausführungsstatus und/oder INTERNAL-BUSY-Listenausführungsstatus gesetzt sind (z.B. **auto_cal**, **goto_xy**, **load_correction_file**) oder werden bei gesetztem INTERNAL-BUSY-Listenausführungsstatus erst dann verzögert ausgeführt, nachdem der INTERNAL-BUSY-Listenausführungsstatus wieder zurückgesetzt wurde (z.B. **execute_list_pos**, **set_offset**).

6.4.4 Listen ausführen und beenden

Das Abarbeiten einer Liste (“Liste 1“ oder “Liste 2”) kann gestartet werden:

- Mit dem Kontrollbefehl **execute_list**
- Durch ein externes Startsignal, siehe Kapitel 9.3.1 “Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden”, Seite 314

Zum Starten der Listenausgabe an einer bestimmten Adresse kann **execute_at_pointer** verwendet werden.

Wenn ein externes Startsignal, siehe Kapitel 9.3.1 “Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden”, Seite 314, verwendet wird, ermöglicht **set_extstartpos_list** die Definition einer Startadresse für den externen Start.

Die RTC6 PCIe-Karte startet die Ausführung sofort. Während die Befehle der Liste dann der Reihe nach im 10 μs-Takt ausgeführt werden, können weiterhin Kontrollbefehle an die RTC6 PCIe-Karte gesendet werden. Diese werden sofort ausgeführt, ohne die Listenausführung zu stören.

Dies kann beispielsweise dazu genutzt werden, eine zweite Liste zu laden, während die erste abgearbeitet wird (PC und Scan-Kopf arbeiten parallel). Allerdings kann die zweite Liste erst gestartet werden, nachdem die erste Liste abgearbeitet wurde. Während der Abarbeitung einer Liste wird **execute_list** oder ein externes Startsignal ignoriert.

Die Ausführung einer Liste kann aber auch – etwa zur Implementierung eines Notstopps – abgebrochen werden. Sobald **stop_execution** aufgerufen oder ein externes Stoppsignal an die RTC6 PCIe-Karte transferiert wird, werden die aktuell ausgeführte Liste sofort abgebrochen und die Signale für den “Laser active”-Betrieb abgeschaltet (aber nicht deaktiviert).

Mit **range_checking** kann das Abarbeiten einer Liste (wie mit **stop_execution**) ebenso automatisch beendet werden.

Wenn beim Abarbeiten einer Liste das Listenende erreicht wird, ohne dass ein **set_end_of_list** gefunden wurde, dann wird mit dem Beginn der aktuellen Liste fortgefahren. Das wiederholt sich so lange, bis **stop_execution** aufgerufen oder ein externes Stoppsignal an die RTC6 PCIe-Karte transferiert wird.

Wenn beim Abarbeiten einer Liste ein **set_end_of_list** erreicht wird, dann endet die Ausführung der Liste – wurde allerdings zuvor **auto_change**, **auto_change_pos** oder **start_loop** aufgerufen, dann erfolgt ein Listenwechsel, siehe Kapitel 6.4.6 “Listen automatisch wechseln”, Seite 113. Dieser Listenwechsel erfolgt ausschließlich bei Erreichen eines **set_end_of_list**.

Hinweise

- Listen werden nicht automatisch gestartet. Sie müssen – nach wie vielen geladenen Befehlen auch immer – wie beschrieben gestartet werden, damit sie abgearbeitet werden.
- Damit die Listenausführung auch von externen Signalen gestartet und gestoppt werden kann, verfügt die RTC6 PCIe-Karte über entsprechende Signal-Eingänge, siehe Kapitel 9.3.1 “Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden”, Seite 314.
- Mit **set_pause_list_cond** oder **set_pause_list_not_cond** kann eine Bedingung für den 16-Bit-Eingang festgelegt werden, so dass statt **stop_execution** ein **pause_list** ausführt wird, wenn ein Externer Stopp anliegt. Die Liste kann dann mit **restart_list** weiter ausgeführt werden.

6.4.5 Listen zur Synchronisation der Ausgabe unterbrechen

Mit dem Listenbefehl **set_wait** können nummerierte Unterbrechungspunkte (“wait markers“) in eine Liste eingefügt werden. Jedem Unterbrechungspunkt ist eine Zahl größer als Null zugeordnet. Erreicht die RTC6 PCIe-Karte bei der Abarbeitung einer Liste, siehe Kapitel 6.5 “Strukturierte Programmierung“, Seite 115, einen Unterbrechungspunkt, dann wird die Ausgabe der Liste vorübergehend unterbrochen und der Laser ausgeschaltet.

get_wait_status überprüft, ob die Listenabarbeitung momentan an einem Unterbrechungspunkt angehalten ist. Ist das der Fall, gibt **get_wait_status** die Nummer (wait_word) des Unterbrechungspunkts zurück (anderenfalls den Wert Null).

Unterbrechungspunkte sind für Synchronisationszwecke vorgesehen. Das Anwenderprogramm sollte an jedem Unterbrechungspunkt eine Bearbeitungsroutine ausführen. Ist die Bearbeitungsroutine beendet, kann die Listenabarbeitung (am folgenden Listenbefehl) mit dem Kontrollbefehl **release_wait** fortgesetzt werden.

Durch **set_wait** wird der (mit **get_status** abrufbare) PAUSED-Listenausführungsstatus gesetzt und der BUSY-Listenausführungsstatus zurückgesetzt. Das Umgekehrte erfolgt bei einem anschließenden **release_wait**.

Die Listenausführung kann zu jedem beliebigen Zeitpunkt mit dem Kontrollbefehl **pause_list** (oder mit dem gleichbedeutenden **stop_list**) unterbrochen und durch **restart_list** wieder aufgenommen werden. Durch **pause_list** werden die Signale für den “Laser active”-Betrieb unterdrückt und das Scan-System bleibt im zuletzt vorgegebenen Zustand stehen – gegebenenfalls auch mitten in einer Zerlegung in Mikroschritte. Nach dem anschließenden **restart_list** setzt das Scan-System die vorgesehenen Bewegungen (des aktuellen Befehls) fort und die Lasersteuersignale werden wieder freigegeben (eine nahtlose Fortführung einer unterbrochenen Markierung ist im allgemeinen nicht möglich).

Durch **pause_list** wird der (mit **get_status** abrufbare) PAUSED-Listenausführungsstatus gesetzt und durch **restart_list** zurückgesetzt. Der BUSY-Listenausführungsstatus wird von beiden Befehlen unverändert belassen.

6.4.6 Listen automatisch wechseln

Wenn bei der RTC6-Listenspeicher-Konfiguration zwei RTC6-Listenspeicher-Bereiche (“Liste 1” und “Liste 2”) eingerichtet werden, siehe Kapitel 6.3.2 “RTC6-Listenspeicher konfigurieren“, Seite 106, dann kann eine zweite Liste geladen werden während die erste Liste noch abgearbeitet wird.

In der Regel dauert es wesentlich länger, eine Liste abzuarbeiten, als sie in den Speicher zu schreiben. Damit ist auch die kontinuierliche Abarbeitung von beliebig langen Listen möglich, wenn sie in Befehlsblöcke aufgeteilt sind.

Die kontinuierliche Befehlsausgabe beim notwendigen Wechsel zwischen den beiden Listen kann dabei – wie in den folgenden Abschnitten beschrieben – über einen automatischen Listenwechsel sichergestellt werden.

Die Befehle zum automatischen Listenwechsel werden erst wirksam, wenn der nächstfolgende **set_end_of_list** ausgeführt wird. Ein automatischer Listenwechsel nach Abarbeiten einer Liste wird also nur dann durchgeführt, wenn diese Liste mit einem **set_end_of_list** abgeschlossen wurde. Anderenfalls wird mit derselben Liste von vorne begonnen.

Wenn bei der RTC6-Listenspeicher-Konfiguration der zweite RTC6-Listenspeicher-Bereich (“Liste 2”) auf die Größe 0 gesetzt wird, dann führen alle automatischen Listenwechsel-Befehle wieder zu “Liste 1” an die angegebene Position.

Einmaliger Listenwechsel

auto_change oder **auto_change_pos** aktivieren einen einmaligen, automatischen Listenwechsel zwischen “Liste 1” und “Liste 2”. Nach Abarbeiten der gerade aktuellen Liste (wenn **set_end_of_list** erreicht wird) wird dadurch automatisch das Abarbeiten der nächsten Liste gestartet.

Bei Verwendung von **auto_change** wird die nächste Liste an der Position 0 gestartet, bei Verwendung von **auto_change_pos** an der angegebenen Startposition (RTC6-Listenspeicheradresse als Offset bezüglich des Listenanfangs).

Alternierender Listenwechsel

Alternativ kann eine kontinuierliche Befehlsausgabe über eine sich wiederholende alternierende Ausgabe der beiden Listen realisiert werden.

Dazu muss **start_loop** aufgerufen werden. Dies führt dann zu einer automatischen, kontinuierlichen, alternierenden Abarbeitung der beiden Listen, vorausgesetzt, beide Listen werden am Ende jeweils mit **set_end_of_list** abgeschlossen.

Die alternierende Abarbeitung wiederholt sich so lange, bis **quit_loop** aufgerufen wird. **quit_loop** bricht die kontinuierliche Abarbeitung ab, sobald die aktuelle Liste beendet ist.

Die jeweils nicht-aktive Liste kann auch während der Abarbeitung der anderen Liste neu geladen werden. Damit ist neben der alternierenden Abarbeitung zweier konstanter Listen auch eine ständig inhaltlich wechselnde Liste ausführbar.

Hinweise

- Die Befehle zum Starten eines einmaligen, automatischen Listenwechsels und **start_loop** zum Starten eines alternierenden Listenwechsels können zu einem beliebigen Zeitpunkt aufgerufen werden. Sie werden aber erst mit dem Erreichen des nächsten **set_end_of_list** wirksam.
- Beim Laden einer Liste während der Abarbeitung einer anderen Liste, sollte sichergestellt werden, dass keine Befehle überschrieben werden, die noch gebraucht werden. Dazu eignet sich **load_list**. **load_list** startet das Laden einer Liste nur dann, wenn die ausgewählte Liste nicht aktuell in Verwendung ist oder bereits abgearbeitet ist, siehe Kapitel 6.4.1 "Listen laden", Seite 108.
- Außerdem sollten die jeweils neuen Listen bereits zu einem gewissen Anteil geladen sein, bevor der Listenwechsel stattfindet. Der Eingabe-Zeiger sollte immer einen ausreichenden Vorsprung gegenüber dem Ausgabe-Zeiger haben (weil der PCI-Transfer der Listenbefehle in den RTC6-Listenspeicher gepuffert erfolgt, siehe Kapitel 6.4.1 "Listen laden", Seite 108, und sogenannte kurze Listenbefehle verwendet werden können, siehe Kapitel 6.1.2 "Kontrollbefehle und Listenbefehle", Seite 97. Anderenfalls werden u.U. "alte" Befehle ausgeführt.
- Die RTC6 PCIe-Karte unterstützt den RTC4-Ringpuffer-Modus nicht, siehe Kapitel 6.5.4 "RTC4-Ringpuffer-Modus", Seite 124. Jedoch kann dieser Betriebsmodus effektiv mit Hilfe des oben beschriebenen alternierenden Listenwechsels und **load_list** ersetzt werden.

6.5 Strukturierte Programmierung

Zur Unterstützung einer strukturierten Programmierung und Ausgabe von Listenbefehlen enthält der RTC6-Befehlssatz eine Reihe von Möglichkeiten zur Definition von Unterprogrammen und Zeichensätzen, sowie Listenbefehle zum Steuern des Programmflusses.

6.5.1 Unterprogramme

- Unterprogramme können als Listenbefehlssequenzen grundsätzlich an jeder beliebigen Stelle des RTC6-Listenspeichers stehen.
- Unterprogramme sollten bevorzugt in das obere Ende des RTC6-Listenspeicher-Bereichs geschrieben werden, siehe Abschnitt ""Liste 3" – Geschützter RTC6-Listenspeicher-Bereich", Seite 105.
- Durch ein Unterprogramm sollte keine Listengrenze verlaufen.
- Ein Unterprogramm muss mit einem **list_return** abgeschlossen werden.
- Definiert werden können:
  - Nichtindizierte Unterprogramme
  - Indizierte Unterprogramme

Nichtindizierte Unterprogramme

Nichtindizierte Unterprogramme werden wie "normale" Listenbefehlssequenzen mit den Listen-Lade-Befehlen in einen RTC6-Listenspeicher-Bereich ("Liste 1" oder "Liste 2") geladen, siehe Kapitel 6.4.1 "Listen laden", Seite 108. Ein Unterprogramm muss mit einem **list_return** abgeschlossen werden. Es wird mit **list_call** unter Angabe der absoluten Speicheradresse als Parameter aufgerufen.

Nach dem Abarbeiten eines Unterprogramms (einschließlich des abschließenden **list_return**) wird mit demjenigen Befehl fortgefahren, der der aufrufenden Stelle nachfolgt.

Hinweise

- Nichtindizierte Unterprogramme können nicht direkt in den geschützten RTC6-Listenspeicher-Bereich "Liste 3" geschrieben werden. Sie können jedoch nachträglich geschützt werden, siehe auch Abschnitt "Nachträglich nichtindizierte Unterprogramme schützen und konvertieren", Seite 119. Damit das Unterprogramm zu diesem Zweck mit **set_sub_pointer** indiziert werden kann, muss allerdings seine Anfangsadresse bekannt sein. Vor dem Laden eines nichtindizierten Unterprogramms in den RTC6-Listenspeicher-Bereich "Liste 1" oder "Liste 2" sollte daher immer die Startadresse mit **get_input_pointer** ausgelesen werden.
- Achten Sie darauf, dass im normalen Ablauf einer Liste oder im Rumpf eines Unterprogramms kein **list_return** angetroffen wird, für das es keinen entsprechenden Unterprogramm-Aufruf gegeben hat. Bei verschachtelten Unterprogramm-Aufrufen wird sonst die Integrität der Verschachtelung zerstört. Ist kein Unterprogramm-Aufruf mehr aktiv, wird die Listenabarbeitung an der absoluten Position 0 fortgesetzt.
  Beginnt ein Unterprogramm unmittelbar nach einem **list_call**, **list_call_abs**, **list_call_repeat** oder **list_call_abs_repeat**, wird die Rücksprungadresse automatisch von `Pos(list_call) + 1` auf `Pos(list_return) + 1` gesetzt. D.h. es wird derjenige Befehl abgearbeitet, der **list_return** nachfolgt und nicht der Befehl, der **list_call** nachfolgt (dieser würde das Unterprogramm erneut mit einem unkorrelierten **list_return** abarbeiten).

Indizierte Unterprogramme

load_sub versieht ein (durch nachfolgende Listenbefehle definiertes) Unterprogramm mit dem gewünschten Index und lädt es in den geschützten RTC6-Listenspeicher-Bereich “Liste 3”.

Ein indiziertes Unterprogramm muss mit einem **list_return** abgeschlossen werden (sonst wird es nicht gespeichert). Es wird mit **sub_call** unter Angabe des Indexes aufgerufen.

Es können höchstens 1024 indizierte Unterprogramme gespeichert werden.

Die Verwaltung der indizierten Unterprogramme erfolgt automatisch auf der RTC6 PCIe-Karte.

Weitere Informationen zur Indexverwaltung siehe Abschnitt “Indexverwaltung und Defragmentierung”, Seite 118.

Die Speicheradresse eines indizierten Unterprogramms kann mit **get_sub_pointer** abgefragt werden. Mit dieser kann dann das indizierte Unterprogramm (wie ein nichtindiziertes Unterprogramm auch) unter Angabe der absoluten Speicheradresse aufgerufen werden.

Ein indiziertes Unterprogramm wird mit **load_sub** nur unter folgenden Umständen gespeichert:

- Wenn vor dem Laden der geschützte RTC6-Listenspeicher-Bereich “Liste 3” in ausreichender Größe jenseits von “Liste 1” und “Liste 2” konfiguriert wurde. Wenn beispielsweise der gesamte RTC6-Listenspeicher einer oder beiden Listen zugeordnet ist, können keine indizierten Unterprogramme gespeichert werden. Mit **get_list_space**, falls dieser nach einem **load_sub** (aber vor dem abschließenden **list_return**) angewendet wird, kann der noch frei verfügbare Speicherplatz des geschützten RTC6-Listenspeicher-Bereichs “Liste 3” abgefragt werden
- Wenn das indizierte Unterprogramm mit einem **list_return** abgeschlossen wird
- Wenn vor dem **list_return** kein anderer Befehl zur Positionierung des Eingabe-Zeigers erfolgt (wie z.B. ein erneutes **load_sub**, **set_input_pointer**, oder **set_start_list_pos**)
- Wenn der Index im zulässigen Bereich (0…1023) liegt

Nach einem **list_return** wird der Eingabe-Zeiger ungültig. Direkt nachfolgende Listenbefehle werden nicht mehr gespeichert.

Indizierte Unterprogramme werden mit **load_sub**-Befehlen in der Reihenfolge der Eintragung in “Liste 3” geschrieben. Die Startadresse wird dabei automatisch hinter das Ende des letzten Unterprogramms gesetzt.

Wird ein indiziertes Unterprogramm mit einem bereits bestehenden Index abgespeichert, dann wird das vorhergehende so indizierte Unterprogramm nicht überschrieben. Es bleibt zunächst im geschützten RTC6-Listenspeicher-Bereich “Liste 3” gespeichert und ist dann aber nicht mehr indiziert. Daher kann es nicht mehr mit **sub_call** über seinen Index aufgerufen werden, allerdings weiterhin mit **list_call** über seine absolute Speicheradresse.

Mit **get_sub_pointer** wird abgefragt, ob ein Unterprogramm von einem bestimmten Index referenziert wird. Falls kein Unterprogramm referenziert ist, gibt **get_sub_pointer** den Wert “–1” (d.h. 2^{32}–1) zurück.

Damit ein indiziertes Unterprogramm in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” geladen werden kann, wenn dieser bereits mit indizierten Unterprogrammen vollgeschrieben ist, muss zuvor der geschützte RTC6-Listenspeicher-Bereich “Liste 3” mit **config_list** entsprechend vergrößert und mit **save_disk**/**load_disk** defragmentiert werden. Beachten Sie, dass das Vergrößern allein nicht ausreicht, siehe Abschnitt “Indexverwaltung und Defragmentierung”, Seite 118.

Hinweise zu nichtindizierten Aufrufen

- Bei der Indexverwaltung oder Defragmentierung, siehe Abschnitt “Indexverwaltung und Defragmentierung”, Seite 118, ändert sich u.U. die absolute Speicheradresse eines indizierten Unterprogramms. Vom Aufruf eines indizierten Unterprogramms mit **list_call** ist daher eher abzuraten.

Programmierregeln

Beim Programmieren von indizierten Unterprogrammen sollten die folgenden Regeln beachtet werden:

- In einem indizierten Unterprogramm wird **set_end_of_list** durch ein **list_nop** ersetzt.
- Absolute Sprünge in den geschützten RTC6-Listenspeicher-Bereich “Liste 3“ hinein und aus dem geschützten RTC6-Listenspeicher-Bereich “Liste 3“ heraus werden beim Abarbeiten ignoriert, siehe Kapitel 6.5.3 “Sprünge“, Seite 123. Deswegen können absolute Sprünge in indizierten Unterprogrammen nicht verwendet werden.
- Ebenso werden beim Abarbeiten des Unterprogramms ignoriert:
  - Relative Sprünge, die über die Grenzen des indizierten Unterprogramms hinausgehen
  - Sprung-Befehle, die einen Sprung auf sich selbst veranlassen

Allgemeines zum Aufruf von Unterprogrammen

Es sind verschachtelte Aufrufe mit einer Verschachtelungstiefe von bis zu 63 Aufrufen möglich.

Bei einem Aufruf mit **sub_call** oder **list_call** dürfen nur relative Sprung-Befehle und Markier-Befehle verwendet werden, wenn die Unterprogramm-Ausführung an unterschiedlichen Bildfeldstellen wiederholt werden soll.

Um die häufig leichter handhabbaren absoluten Sprung-Befehle und Markier-Befehle verwenden zu können, sind die sog. “AbsCalls“ vorgesehen.

“AbsCalls“

Wenn die Unterprogramme nur relative Vektor-Befehle und Bogen-Befehle, siehe Kapitel 7.1.1 “Mit Vektor-Befehlen und Bogen-Befehlen markieren“, Seite 139, enthalten, können die entsprechenden Prozesse an unterschiedlichen Stellen im Bildfeld wiederholt werden.

Bei “AbsCalls“ wird die momentane Position als Offset übernommen. Dieser Offset wird dann bei allen nachfolgenden Vektor-Befehlen und Bogen-Befehlen im Unterprogramm berücksichtigt.

Eine Verschachtelung der Aufrufe wird bei der Ermittlung des Offsets berücksichtigt. Dadurch können beispielsweise Zeichensätze auch mit absoluten Vektoren definiert werden.

“AbsCalls“ von Unterprogrammen erfolgen über **list_call_abs** und **sub_call_abs**.

Bedingte Aufrufe

Damit das Aufrufen von Unterprogrammen auch von externen Steuersignalen abhängig gemacht werden kann, gibt es die Möglichkeit, mit zusätzlichen Befehlen bedingte Verzweigungen in den Programmablauf einzubauen, siehe Kapitel 9.3.2 “Ausführung Bedingter Befehle“, Seite 320.

Aufrufe mit wiederholter Ausführung

Mit **sub_call_repeat**, **sub_call_abs_repeat**, **list_call_repeat** und **list_call_abs_repeat** kann der Rumpf des Unterprogramms automatisch wiederholt ausgeführt werden.

Indexverwaltung und Defragmentierung

Indexverwaltung

Damit indizierte Unterprogramme dupliziert, umnummeriert oder ineinander konvertiert werden können, wird **copy_dst_src** bereitgestellt.

Durch **copy_dst_src** wird ein indiziertes Unterprogramm ein weiteres Mal indiziert. **copy_dst_src** ändert nur die entsprechenden Einträge in der internen Verwaltungstabelle und lässt den RTC6-Listenspeicherinhalt unverändert.

Nicht mehr benötigte Indices (überflüssige Einträge in der internen Verwaltungstabelle) können mit **load_sub** zusammen mit einem unmittelbar nachfolgenden **list_return** gelöscht werden. Auch hier erfolgt das Löschen nur in der internen Verwaltungstabelle und die Listenbefehle des zuvor indizierten Unterprogramms bleiben weiterhin im RTC6-Listenspeicher stehen.

Mit **get_sub_pointer** kann abgefragt werden, ob für einen bestimmten Index ein Unterprogramm existiert. Falls kein Unterprogramm existiert, gibt **get_sub_pointer** einen Wert von "–1" (d. h. 2^{32}–1) zurück.

Ein echtes Duplikat eines indizierten Unterprogramms im geschützten RTC6-Listenspeicher-Bereich "Liste 3" kann – nachfolgend auf **copy_dst_src** – mit **save_disk**/**load_disk** erstellt werden. Mehrfach indizierte Unterprogramme werden dadurch entsprechend oft in den RTC6-Listenspeicher geschrieben. Achten Sie darauf, dass es dabei zu keinem Speicherüberlauf im geschützten RTC6-Listenspeicher-Bereich "Liste 3" kommt.

load_sub trägt ein neues indiziertes Unterprogramm immer hinter dem indizierten Unterprogramm mit der höchsten Speicheradresse ein. Deswegen blockieren nicht mehr benötigte Unterprogramme, die im geschützten RTC6-Listenspeicher-Bereich "Liste 3" liegen, u.U. Speicherplätze für weitere indizierte Unterprogramme.

Aus dem gleichen Grund schafft ein nachträgliches Vergrößern des geschützten RTC6-Listenspeicher-Bereichs "Liste 3" mit **config_list** zunächst keine weiteren nutzbaren Speicherplätze für das Abspeichern von indizierten Unterprogrammen (der geschützte RTC6-Listenspeicher-Bereich "Liste 3" kann immer nur nach unten, nicht aber nach oben vergrößert werden).

Defragmentierung

Abhilfe schafft hier eine Defragmentierung mit **save_disk**/**load_disk**.

Dabei werden alle indizierte Unterprogramme und alle über **set_sub_pointer** nachträglich indizierte Unterprogramme neu in den geschützten RTC6-Listenspeicher-Bereich "Liste 3" geschrieben (in Indexreihenfolge, beginnend beim untersten Speicherplatz des geschützten RTC6-Listenspeicher-Bereichs "Liste 3").

Die hierbei freigebliebenen oberen Speicherplätze können dann zum Speichern von weiteren indizierten Unterprogrammen genutzt werden.

Hinweise

- Vor dem Aufruf von **load_disk** sollte sichergestellt werden, dass der geschützte RTC6-Listenspeicher-Bereich "Liste 3" in ausreichender Größe jenseits von "Liste 1" und "Liste 2" konfiguriert wurde. Ein indiziertes Unterprogramm, das dort nicht vollständig Platz findet, wird von **load_disk** *nicht* gespeichert. **save_disk** liefert die Anzahl von gespeicherten Listenbefehlen zurück. Nicht mehr benötigte Unterprogramme sollten zuvor mit **load_sub** und einem unmittelbar nachfolgendem **list_return** aus der Index-Verwaltung gelöscht werden.
- Bei der Indexverwaltung oder Defragmentierung ändert sich u.U. die absolute Speicheradresse eines indizierten Unterprogramms. Vom Aufruf eines indizierten Unterprogramms mit **list_call** ist daher abzuraten.
- **save_disk** speichert die Unterprogramme ab der Start-Adresse bis zum nächstmöglichen **list_return**. Relative Sprünge werden dabei nicht ausgewertet. Verwenden Sie daher keine Verzweigungen zu unterschiedlichen **list_return**, sondern führen Sie die Verzweigungen bereits vor dem einzigen **list_return** zusammen.

Nachträglich nichtindizierte Unterprogramme schützen und konvertieren

Nichtindizierte Unterprogramme können (direkt) nur in einen RTC6-Listenspeicher-Bereich (“Liste 1” oder “Liste 2”) geschrieben werden, aber nicht in den geschützten RTC6-Listenspeicher-Bereich “Liste 3”.

Grundsätzlich gibt es zwei Methoden, um nichtindizierte Unterprogramme nachträglich zu schützen:

(1) Konfiguration ändern

Der Teil des RTC6-Listenspeicher-Bereichs, in den das nichtindizierte Unterprogramm geschrieben wurde, wird mit **config_list** dem geschützten RTC6-Listenspeicher-Bereich “Liste 3” zugeordnet. Das auf diese Weise nachträglich geschützte Unterprogramm ist weiterhin ein nichtindiziertes Unterprogramm (mit unveränderter Speicheradresse) und kann nach wie vor mit **list_call** aufgerufen werden.

(2) In indizierte Unterprogramme konvertieren

Mit **set_sub_pointer** wird ein nichtindiziertes Unterprogramm indiziert und damit in die Speicherverwaltung der indizierten Unterprogramme aufgenommen.
Mit **save_disk**/**load_disk** kann es anschließend als indiziertes Unterprogramm in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” kopiert werden, siehe Abschnitt “Defragmentierung”, Seite 118.

Bei einem nachfolgenden Aufruf über den Index mit **sub_call** wird dann das Unterprogramm im geschützten RTC6-Listenspeicher-Bereich “Liste 3” gestartet.

Falls ein nichtindiziertes Unterprogramm – sei es nach Methode 1 oder nach Methode 2 – nachträglich geschützt werden soll, so muss beachtet werden, dass absolute Sprünge in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” und aus dem geschützten RTC6-Listenspeicher-Bereich “Liste 3” heraus nicht erlaubt sind, siehe Kapitel 6.5.3 “Sprünge”, Seite 123.

Beim Konvertieren in indizierte Unterprogramme (Methode 2) müssen auch alle anderen Programmierregeln für indizierte Unterprogramme beachtet werden, siehe Abschnitt “Programmierregeln”, Seite 117.

Wenden Sie möglichst immer nur eine der beiden Methoden an. Das vermeidet, dass es zu einem unbeabsichtigten Datenverlust im geschützten RTC6-Listenspeicher-Bereich “Liste 3” durch Überschreiben kommt.

Wenn Sie zu Beginn nur mit Methode 1 arbeiten, aber zu einem späteren Zeitpunkt auch indizierte Unterprogramme verwenden wollen: dann sollten Sie alle im geschützten RTC6-Listenspeicher-Bereich “Liste 3” befindlichen nichtindizierten Unterprogramme mit Methode 2 in indizierte Unterprogramme konvertieren, bevor Sie mit **load_sub** das erste indizierte Unterprogramm definieren. Beachten Sie hierzu auch die folgenden Hinweise.

Hinweise

- Wird bei Methode 1 ein Teil des geschützten RTC6-Listenspeicher-Bereichs “Liste 3” zum Überschreiben entschützt, so riskiert man dabei, dass indizierte Unterprogramme oder bereits zuvor geschützte Unterprogramme überschrieben werden.
- Nachträglich nach Methode 1 geschützte nicht-indizierte Unterprogramme werden u.U. durch einen nachfolgendes **load_sub** oder **load_disk** überschrieben.
- **set_sub_pointer** verknüpft den angegebenen Index auch dann mit der angegebenen Start-Adresse, wenn zuvor bereits ein indiziertes Unterprogramm für diesen Index definiert wurde. Das ursprünglich so indizierte Unterprogramm ist dann nicht mehr indiziert und kann nicht mehr über den Index aufgerufen werden.

- Wenn Methode 2 angewendet wird, dann sollte sie auch vollständig angewendet werden. Wenn nämlich nur **set_sub_pointer** ausgeführt wird, dann kann zwar das Unterprogramm bereits unter Angabe des Indexes mit **sub_call** aufgerufen werden, das Unterprogramm ist aber nach wie vor nicht gegen Überschreiben geschützt.
  Letzteres erfolgt erst, wenn das Unterprogramm nachträglich mit **save_disk**/**load_disk** als indiziertes Unterprogramm in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” kopiert wird.
- **save_disk** ignoriert alle nichtindizierte Unterprogramme, also auch die über Methode 1 nachträglich geschützten nichtindizierten Unterprogramme im geschützten RTC6-Listenspeicher-Bereich “Liste 3”. Dort werden sie u.U. durch **load_disk** überschrieben.
- **save_disk**/**load_disk** ersetzt automatisch unzulässige Befehle (z.B. **set_end_of_list**) durch **list_nop**-Befehle.
- Indizierte Unterprogramme, die mit **copy_dst_src** mehrfach indiziert werden, werden bei einem anschließenden **save_disk**/**load_disk** im RTC6-Listenspeicher dupliziert. Dadurch kann es zu einem Speicherüberlauf im geschützten RTC6-Listenspeicher-Bereich “Liste 3” kommen.
- Vor Ausführen von **load_disk** sollte sichergestellt werden, dass der geschützte RTC6-Listenspeicher-Bereich “Liste 3” in ausreichender Größe jenseits von “Liste 1” und “Liste 2” konfiguriert wurde (**save_disk** liefert die Anzahl der gespeicherten Listenbefehle zurück). Ein indiziertes Unterprogramm, das dort nicht vollständig Platz findet, wird von **load_disk** *nicht* gespeichert.
- Bei der Konvertierung eines Unterprogramms gemäß Methode 2 ändert sich die absolute Speicheradresse des Unterprogramms.

Unterprogramme entschützen

Der Schutz für Unterprogramme, die im geschützten RTC6-Listenspeicher-Bereich “Liste 3” gespeichert sind, wird aufgehoben, indem der entsprechende RTC6-Listenspeicher-Bereich mit **config_list** dem RTC6-Listenspeicher-Bereich “Liste 1” oder “Liste 2” zugeordnet wird.

Das Unterprogramm kann dann nach wie vor mit unveränderten Parametern (Index oder absolute Speicheradresse) aufgerufen werden. Es ist dann aber nicht mehr gegen versehentliches Überschreiben geschützt.

6.5.2 Zeichensätze und Zeichenfolgen

Für Beschriftungsaufgaben ist es sinnvoll, Befehlslisten in Form von separaten Unterprogrammen im RTC6-Listenspeicher anzulegen, mit denen festgelegt ist, wie die benötigten Zeichen und/oder Zeichenfolgen vom Scan-System markiert werden sollen.

Um die Verwaltung von Zeichen und Zeichenfolgen sowie deren Verwendung beim Markieren zu erleichtern, bietet die RTC6 PCIe-Karte die Möglichkeit, indizierte Zeichendefinitionen und Zeichenfolgedefinitionen im geschützten RTC6-Listenspeicher-Bereich “Liste 3” abzuspeichern und mit einfachen Befehlen aufzurufen.

Indizierte Zeichendefinitionen und Zeichenfolgedefinitionen sind grundsätzlich nichts anderes als indizierte Unterprogramme, werden aber mit eigenen Befehlen definiert und aufgerufen und werden von der RTC6 PCIe-Karte mit Hilfe einer eigenen internen Verwaltungstabelle – separat von indizierten Unterprogrammen – verwaltet.

In den einzelnen Zeichen- oder Zeichenfolgedefinitionen muss die Form und die Orientierung (z.B. parallel zur x- oder y-Achse) der Zeichen oder Zeichenfolgen festgelegt werden. Dabei können sowohl relative als auch absolute Vektor-Befehle verwendet werden. Die Endposition eines Zeichens oder einer Zeichenfolge sollte so gewählt werden, dass sie der Startposition eines eventuell nachfolgenden Zeichens entspricht. Jede Zeichen- oder Zeichenfolgedefinition muss mit **list_return** abgeschlossen werden.

Indizierte Zeichensätze definieren

Eine Listenbefehlssequenz zur Definition eines Zeichens kann mit **load_char** als indiziertes Zeichen direkt in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” geladen werden (die dabei automatisch vergebene Speicheradresse kann mit **get_char_pointer** abgefragt werden). Alternativ kann aber auch ein nichtindiziertes Unterprogramm nachträglich mit **set_char_pointer** indiziert und dann anschließend mit **save_disk**/**load_disk** als indiziertes Zeichen in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” kopiert werden.

Die RTC6 PCIe-Karte verwaltet bis zu 4 Zeichensätze mit jeweils 256 indizierten Zeichen.

Ansonsten gelten die gleichen Regeln wie für indizierte Unterprogramme, siehe Abschnitt “Indizierte Unterprogramme”, Seite 116 und Abschnitt “Nachträglich nichtindizierte Unterprogramme schützen und konvertieren”, Seite 119.

Hinweise

- \0 (NUL) ist auch ein markierbares Zeichen. \0 dient ferner als Delimiter für Textausgaben (Zeichenfolgen), wird dort aber nicht markiert.
- Indizierte Zeichendefinitionen dürfen nicht **mark_text**, **mark_time**, **mark_date** und **mark_serial** enthalten. Sonst kann es bei der Ausführung des indizierten Zeichens zu einer fehlerhaften Markierung kommen.

Indizierte Zeichen aufrufen

Das Markieren eines einzelnen Zeichens wird mit **mark_char** (oder mit dem “AbsCall”-Befehl **mark_char_abs**) unter Angabe des Indexes der zugehörigen indizierten Zeichendefinition gestartet.

Zur Beschriftung von Seriennummern können indizierte Zeichen (Ziffern) auch mit **mark_serial** aufgerufen werden, siehe Kapitel 7.5 “Uhrzeit, Datum und Seriennummern markieren”, Seite 222.

Das Markieren ganzer Textpassagen kann mit **mark_text** (oder mit dem “AbsCall”-Befehl **mark_text_abs**) gestartet werden. Der gewünschte Zeichensatz kann dabei zuvor mit **select_char_set** ausgewählt werden.

Beim Laden eines **mark_text** wird der zu markierende Text, wenn er aus mehr als 12 Zeichen besteht, in Blöcke zu je 12 Zeichen zerteilt und für jeden Block ein separater **mark_text** in den RTC6-Listenspeicher geschrieben. Achten Sie darauf, dass es dabei zu keinem unerwünschten Speicherüberlauf des jeweiligen Speicher-Bereichs kommt.

Indizierte Zeichenfolgen für Uhrzeit, Datum und Seriennummer definieren

Zur Beschriftung von Uhrzeit, Datum und Seriennummern können beispielsweise Zeichenfolgen für die Monatsbezeichnungen (“Januar“…“Dezember“, “Jan.“…“Dez.”, “/01/“…“/12/” o.ä.) und Wochentagsbezeichnungen (“Sonntag“…“Samstag“, “So.“…“Sa.“, o.ä.) definiert werden.

Dabei können auch – unter Verwendung von **mark_char** und **mark_text** – zuvor definierte Zeichensätze verwendet werden.

Mit **load_text_table** kann eine Listenbefehlssequenz zur Definition einer Zeichenfolge als indizierte Zeichenfolge direkt in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” geladen werden (die dabei automatisch vergebene Speicheradresse kann mit **get_text_table_pointer** abgefragt werden).

Alternativ kann aber auch ein nichtindiziertes Unterprogramm nachträglich mit **set_text_table_pointer** indiziert und dann anschließend mit **save_disk**/**load_disk** als indizierte Zeichenfolge in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” kopiert werden.

Die RTC6 PCIe-Karte verwaltet bis zu 42 indizierte Zeichenfolgen.

Ansonsten gelten die gleichen Regeln wie für indizierte Unterprogramme, siehe Abschnitt “Indizierte Unterprogramme”, Seite 116 und Abschnitt “Nachträglich nichtindizierte Unterprogramme schützen und konvertieren”, Seite 119.

Hinweise

- **set_char_table** ist gleichbedeutend mit **set_text_table_pointer**.

Indizierte Zeichenfolgen aufrufen

Indizierte Zeichenfolgen können zur Beschriftung von Uhrzeit, Datum und Seriennummern mit **mark_time**, **mark_date** und **mark_serial** (oder mit den "AbsCall"-Befehlen **mark_time_abs**, **mark_date_abs** und **mark_serial_abs**) aufgerufen werden, siehe Kapitel 7.5 "Uhrzeit, Datum und Seriennummern markieren", Seite 222.

Indizierte Zeichen und Zeichenfolgen verwalten

Die Indexverwaltung von indizierten Zeichen und Zeichenfolgen erfolgt getrennt von der Indexverwaltung für indizierte Unterprogramme.

Die Indexverwaltung durch Benutzer (umnummerieren, duplizieren, …) erfolgt analog zur Indexverwaltung von indizierten Unterprogrammen, siehe Abschnitt "Indexverwaltung und Defragmentierung", Seite 118, mit **copy_dst_src**, **load_char**, **load_text_table**, **get_char_pointer**, **get_text_table_pointer** und **save_disk**/**load_disk**. Eine Defragmentierung des geschützten RTC6-Listenspeicher-Bereichs "Liste 3" bezieht auch die indizierten Zeichen und Zeichenfolgen mit ein.

6.5.3 Sprünge

Mit **list_jump_pos** (ist gleichbedeutend mit **set_list_jump**) und **list_jump_rel** können Sprünge zu einer angegebenen Adresse definiert werden, die im Programmablauf von der RTC6 PCIe-Karte ausgeführt werden.

Bei **list_jump_pos** kann eine absolute Speicheradresse innerhalb des konfigurierten Listen-Bereichs ("Liste 1" und "Liste 2") angegeben werden. Sprünge in den geschützten RTC6-Listenspeicher-Bereich "Liste 3" und aus dem geschützten RTC6-Listenspeicher-Bereich "Liste 3" heraus sind bei **list_jump_pos** nicht zulässig. Ein **list_jump_pos** mit einer solchen unerlaubten Sprungadresse wird beim Abarbeiten ignoriert.

Bei **list_jump_rel** können Sprungweiten (d. h. relative Speicheradressen) angegeben werden. **list_jump_rel** kann in allen RTC6-Listenspeicher-Bereichen, also auch im geschützten RTC6-Listenspeicher-Bereich "Liste 3", verwendet werden. Beim Angeben der Sprung-Adresse muss aber darauf geachtet werden, dass der Sprung nicht über die Grenzen des jeweiligen RTC6-Listenspeicher-Bereichs hinaus erfolgt. Anderenfalls wird **list_jump_rel** von der RTC6 PCIe-Karte beim Abarbeiten ignoriert.

Falls **list_jump_rel** in einem indizierten Unterprogramm eingesetzt wird, muss zusätzlich darauf geachtet werden, dass der Sprung nicht über die Grenzen des Unterprogramms hinaus geht.

Relative Sprünge, die über die Grenzen eines indizierten Unterprogramms hinausgehen, werden von der RTC6 PCIe-Karte beim Abarbeiten des Unterprogramms ignoriert.

Hinweise

- Beim Umkonfigurieren des RTC6-Listenspeichers oder bei der Konvertierung eines Unterprogramms kann es passieren, dass eine ursprünglich gültige Sprungadresse aufgrund neuer Listengrenzen oder einer veränderten Speicherposition des Unterprogramms zu einer ungültigen Sprung-adresse wird. In diesem Fall wird der entsprechende Sprung-Befehl von der RTC6 PCIe-Karte ignoriert und das Anwenderprogramm funktioniert vermutlich nicht mehr wie gewünscht. Bei der Programmierung mit Sprung-Befehlen muss daher mit einer entsprechenden Sorgfalt vorgegangen werden.
- Wenn bedingte Sprung-Befehle verwendet werden, hängt die Ausführung eines Sprungs von einem externen Steuersignal ab, siehe Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
- Sprung-Befehle wie `list_jump_rel( 0 )`, die einen Sprung auf sich selbst veranlassen, werden zur Laufzeit ignoriert. Sonst würden sie bei der Ausführung eine Endlosschleife ohne weitere Aktivitäten ergeben. Bedingte Sprung-Befehle wie `list_jump_rel_cond( Mask1, Mask0, 0 )` sind dagegen erlaubt, z.B. zum Warten auf die Bestätigung eines Signals.

6.5.4 RTC4-Ringpuffer-Modus

Bei der RTC6 PCIe-Karte gibt es den RTC4-Ringpuffer-Modus (engl.: circular queue mode) nicht.

Effektiv lässt sich dieser Betriebsmodus aber bei der RTC6 PCIe-Karte einfach mit Hilfe eines alternierenden Listenwechsels und **load_list** ersetzen.

Mit `load_list ( 3, 0 )` ist dabei sichergestellt, dass neue Befehle nur in eine bereits abgearbeitete Liste (die nicht BUSY-Listenausführungsstatus ist) geladen werden, ohne dass die Nummer der Liste explizit angegeben werden muss, siehe auch Abschnitt "Alternierender Listenwechsel", Seite 114 und Abschnitt "Mit Schutzfunktion laden", Seite 109.

6.5.5 Schleifen

Mit Listensprüngen, siehe Kapitel 6.5.3 "Sprünge", Seite 123, und bedingten Sprüngen, siehe Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320, kann zwar eine beliebige Anzahl von Listenbefehlen unbegrenzt oder extern gesteuert wiederholt werden, jedoch ist damit eine genaue Festlegung der Anzahl der Wiederholungen nicht sicher möglich.

Dies kann jedoch mit dem Befehlspaar **list_repeat** und **list_until** bewerkstelligt werden. Die Befehlsfolge zwischen diesen beiden kurzen Listenbefehlen wird exakt so oft ausgeführt, wie mit **list_until** als Parameter angegeben ist, mindestens jedoch einmal. Dabei sind bis zu 8 verschachtelte Schleifen erlaubt.

list_repeat und **list_until** müssen immer paarweise verwendet werden. Ungepaarte oder überzählige Befehle (**list_until** ohne zugehöriges **list_repeat** sowie **list_repeat**-Anweisungen, die zu einer Verschachtelungstiefe über 8 führen würden) werden ignoriert. Leere Schleifen, also ein **list_repeat** unmittelbar gefolgt von einem **list_until**, werden sofort beendet und nicht wiederholt.

Die Befehlspaare können sich sowohl innerhalb von Listen als auch innerhalb von Unterprogrammen befinden.

Innerhalb von Unterprogrammen führt **list_until** ein **list_jump_rel** zur Adresse unmittelbar nach dem zugehörigen **list_repeat** aus. Schleifen funktionieren nicht über Unterprogramm-Grenzen hinweg, da Listensprünge in ein Unterprogramm hinein oder aus einem Unterprogramm heraus nicht zulässig sind, siehe Kapitel 6.5.3 "Sprünge", Seite 123.

Innerhalb von Listen führt **list_until** dagegen ein **list_jump_pos** (zur Adresse unmittelbar nach dem zugehörigen **list_repeat**) aus. Dadurch können sich **list_repeat** und **list_until** auch in zwei verschiedenen Listen befinden, wenn der Listenwechsel dazwischen garantiert ist (entweder durch einen expliziten Listensprung oder einen automatischen Listenwechsel).

Wenn dagegen eine Liste tatsächlich beendet worden ist (wie es bei Verwendung von **auto_change_pos** der Fall sein kann), wird der **list_repeat**-Stack automatisch gelöscht und die angefangene Schleife kann nicht mehr beendet werden, weil das nächste **list_until** dann kein zugehöriges **list_repeat** mehr vorfindet.

set_end_of_list löscht die gesamte Schleifenverwaltung, wenn kein automatischer Listenwechsel ansteht, **list_return** dagegen nicht.

Explizite Listensprünge in den Rumpf einer **list_repeat**/**list_until**-Schleife oder aus diesem heraus sind zwar zulässig, weil sie nicht überwacht werden können. Bei unsachgemäßem Gebrauch davon kann jedoch die Integrität der Schleifenverwaltung so gestört werden, dass angefangene Schleifen u.U. nicht mehr programmgemäß ablaufen (Unterprogramm-Aufrufe aus einer Schleife heraus sind dagegen jederzeit problemlos möglich, wenn die Unterprogramme nicht selbst ungepaarte **list_repeat**/**list_until**-Befehle enthalten).

Wenn eine **list_repeat**/**list_until**-Schleife mit einer zunächst unbekannten Anzahl von Wiederholungen ausgeführt werden soll, kann beim Parameter `Number` von **list_until** ein hoher Wert (z.B. größer als die höchste zu erwartende Anzahl) angegeben werden. Innerhalb der Schleife kann eine (z.B. von einem externen Signal abhängige) bedingte Verzweigung zu einer Position außerhalb der Schleife springen und somit die Schleife verlassen. An dieser Stelle sollte ein **list_until**(`Number` = 0) stehen, um die gerade verlassene Schleife ordnungsgemäß zu beenden.

6.6 Mehrere RTC6 PCIe-Karten in einem PC verwenden

Es können so viele RTC6 PCIe-Karten in einem PC gleichzeitig betrieben werden, wie es der PCIe-Bus erlaubt. Die RTC6-Kartenverwaltung kann jedoch höchstens 255 RTC6 PCIe-Karten und RTC6 Ethernet-Karten gleichzeitig verwalten, siehe Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.

Alle RTC6 PCIe-Karten arbeiten unabhängig voneinander. Die Befehlslisten aller Karten können jederzeit geladen und ausgeführt werden.

Der RTC6-Befehlssatz ermöglicht zwei Methoden, Anwenderprogramme zu schreiben, wenn mehrere RTC6 PCIe-Karten verwendet werden:

- "Multi-Board-Programmierung", Seite 126
- "Single-Board-Programmierung", Seite 127

6.6.1 Multi-Board-Programmierung

Bei der Multi-Board-Programmierung werden die Multi-Board-Befehle (= Single-Board-Befehlsnamen mit Vorsatz "**n_**", siehe Multi-Board-Befehlsname, Seite 344) verwendet.

Multi-Board-Befehle dienen zur gleichzeitigen Verwendung mehrerer RTC6-Karten, siehe auch Kapitel 6.6 "Mehrere RTC6 PCIe-Karten in einem PC verwenden", Seite 126.

Bei Multi-Board-Befehlen muss im Unterschied zu Single-Board-Befehlen (= Befehlsnamen ohne dem Vorsatz "**n_**") als erster Parameter `CardNo` angeben werden. `CardNo` ist die Kartennummer(1), an die der Befehl übergeben werden soll. Alle anderen Parameter sind identisch.

Die RTC6 PCIe-Karten sind entsprechend der bei der Initialisierung gefundenen Reihenfolge (beginnend bei 1) durchnummeriert, siehe Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.

Auf welche der RTC6 PCIe-Karten sich die Nummern beziehen, lässt sich mit dem Multi-Board-Befehl **n_get_serial_number** herausfinden. Siehe auch Beispiel (3), unten.

rtc6_count_cards gibt die Anzahl der RTC6 PCIe-Karten in der RTC6-Kartenverwaltung zurück.

Hinweise

- Multi-Board-Befehle werden an die aktive Karte (Default-Karte) gesendet, wenn die angegebene Nummer > 255 oder 0 ist (reale Karten zählen ab 1).
- Der Multi-Board-Befehl wird zurückgewiesen, wenn in der RTC6-Kartenverwaltung unter der angegebenen Nummer keine reale Karte eingetragen ist.
- Alle Multi-Board-Befehle sind in Kapitel 10.1 "Übersicht", Seite 327 aufgeführt. Für fast jeden Single-Board-Befehl steht ein entsprechender Multi-Board-Befehl bereit, siehe Zeile Multi-Board-Befehlsname, Seite 344 in der jeweiligen Befehlsbeschreibung.

Beispiele (Pascal)

(1) Schreibe einen Sprung-Befehl zum Punkt (500, 500) in die aktuelle Liste der RTC6 PCIe-Karte #1:

```
n_jump_abs(1, 500, 500)
```

(2) Arbeite die Liste mit der Nummer `list_no` (1 oder 2) auf der RTC6 PCIe-Karte mit der Nummer ab, die durch die Variable `RTC6_no` angegeben wird:

```
n_execute_list(RTC6_no, list_no)
```

(3) Gebe die Seriennummer von RTC6-Karte #1 zurück:

```
sn_1 := n_get_serial_number(1)
```

(1) Als 32-Bit-Wert ohne Vorzeichen.

6.6.2 Single-Board-Programmierung

Bei der Single-Board-Programmierung wird jeweils eine der eingesteckten RTC6 PCIe-Karten mit **select_rtc** als Default-Karte festgelegt.

Alle Single-Board-Befehle, die auf **select_rtc** folgen, werden so lange an die Default-Karte gesendet, bis **select_rtc** erneut aufgerufen wird.

Multi-Board-Befehle werden durch **select_rtc** nicht beeinflusst (wenn die Kartennummer gültig ist, s.o.).

Seien Sie vorsichtig, wenn ein Prozess mehrere Karten über verschiedene Threads bedient, denn **select_rtc** ist nicht Thread-spezifisch, sondern Karten-spezifisch. Er leitet die Ausgabe aller gegenwärtig laufenden Threads eines Prozesses sofort an die angegebene RTC6 PCIe-Karte weiter.

Achtung!

- **select_rtc** definiert die aktive RTC6 PCIe-Karte für alle gegenwärtig laufenden Threads eines Prozesses (Anwenderprogramms).
  In Multi-Threaded-Anwenderprogrammen kann dies zu Programmierfehlern führen.

6.6.3 Master/Slave-Betrieb

Wenn mehrere RTC6 PCIe-Karten taktsynchronisiert betrieben werden sollen, dann müssen sie paarweise über die Master- und Slave-Anschlüsse miteinander verbunden und möglichst (empfohlen) in benachbarte PCIe-Steckplätze gesteckt werden. Verbinden Sie jeweils den Master-Anschluss einer Karte mit dem Slave-Anschluss einer anderen Karte. Geeignete Verbindungskabel, siehe auch Abbildung 8, sind bei SCANLAB erhältlich.

In einer Master/Slave-Kette ist automatisch diejenige RTC6 PCIe-Karte die Master-Karte, an deren Master-Anschluss eine weitere und an deren Slave-Anschluss keine weitere RTC6 PCIe-Karte angeschlossen ist. Alle anderen RTC6 PCIe-Karten sind Slave-Karten. Siehe auch Kapitel 4.4 "Master-Stiftleiste, Slave-Stiftleiste", Seite 68.

Mit **get_master_slave** kann für jede RTC6 PCIe-Karte der Master/Slave-Status abgefragt werden, d.h. ob sie als Master-Karte, Slave-Karte oder Einzelkarte betrieben wird.

Für ein Quellcodebeispiel auf eine Prüfung, welche RTC6 PCIe-Karte Master und welche Slave ist, siehe Abschnitt "Beispiel-Code (Delphi)", Seite 130.

Die Karten synchronisieren sich automatisch. Nach der Synchronisierung ist die Takt-Phase jeder Karte um 0…3 1/64 μs-Takte (= etwa 0 ns…50 ns) gegenüber derjenigen der vorgeschalteten Karte verzögert.

Mit **get_sync_status** kann überprüft werden, ob eine Slave-Karte synchron zur Master-Karte (oder ihrer Vorgängerkarte in der Master/Slave-Kette) ist.

In einer Master/Slave-Kette summieren sich die Verzögerungen zwischen jeweils zwei Karten auf. Dadurch ergeben sich entsprechend der Anzahl der Slaves in der Master/Slave-Kette größere Verzögerungen zwischen der ersten und der letzten Karte.

Mit **master_slave_set_offset_ctrl** lässt sich der maximale Takt-Phasenunterschied zwischen zwei Karten auf unter 20 ns verringern.

Hinweise

- Siehe auch Kontrollbefehle für Master/Slave, Seite 331.

Initialisierung mit RTC6-Software-Paket ≥ V1.5.0 (≥ RBF 619)

Auf allen RTC6 PCIe-Karten einer Master/Slave-Kette müssen ausgeführt worden sein:

- **load_program_file**

Synchronisierung der Takt-Phasen

Sobald zwei RTC6 PCIe-Karten verbunden sind, erfolgt die Synchronisierung automatisch. **sync_slaves** muss dazu nicht mehr aufgerufen werden.
Der Synchronisierungsstatus kann jederzeit mit **get_sync_status** abgefragt werden, auch ohne vorausgehenden Aufruf von **simulate_ext_start**.
Wenn eine RTC6 PCIe-Karte zwischenzeitlich auf externe Takte synchronisiert wurde (siehe Kapitel 7.4.10 "Synchronisierung von RTC6-Takt und externem Taktsignal", Seite 219), wird die Karte beim Verlassen dieses Zustands automatisch wieder mit der Master-Karte synchronisiert.
Wenn **load_program_file** ausgeführt wird, geht kurzzeitig die Synchronisierung aller nachfolgenden Slave-Karten verloren. Diese wird aber automatisch wieder hergestellt.

Hinweise

- Encodersignale werden nicht von der Masterkarte an die Slave-Karte(n) weitergeleitet. Sie müssen jeweils separat in die Slave-Karte(n) eingespeist werden. Dabei sollte auch die Takt-Phasen-Verschiebung von 0 ns…50 ns berücksichtigt werden.
- Korrekturdateien und Listen müssen auf alle RTC6 PCIe-Karten separat geladen werden.
- Mit `get_sync_status( Bit #21…Bit #30 )` kann die exakte Laufzeit in 1/64 μs-Takten zwischen zwei RTC6-Karten (Hinweg und Rückweg) ausgelesen werden.

Initialisierung mit RTC6-Software-Paket < V1.5.0 (≤ RBF 618)

Auf allen RTC6 PCIe-Karten einer Master/Slave-Kette müssen ausgeführt worden sein:

- **load_program_file**
- **load_correction_file**

Die synchrone Taktung mit stabiler Phasenlage der Master/Slave-Kette wird durch die erste nicht-initialisierte Karte unterbrochen. Eine RTC6 PCIe-Karte, die zwar mit **load_program_file** initialisiert wurde, aber als Slave an eine Karte angeschlossen ist, die nicht mit **load_program_file** initialisiert wurde, unterliegt ihrem eigenen Takt mit zufälliger Phasenlage.

Synchronisierung der Takt-Phasen

Damit die RTC6 PCIe-Karten einer Master/Slave-Kette nicht nur mit synchroner Taktung, sondern auch mit einer definierten relativen Takt-Phase betrieben werden können, müssen die Karten mit **sync_slaves** synchronisiert werden.

Dazu muss **sync_slaves** ein einziges Mal nur an die Master-Karte geschickt werden. SCANLAB empfiehlt, die Synchronisation unmittelbar nach der Initialisierung aller Karten (**load_program_file** und **load_correction_file**) durchzuführen. Es genügt, **load_correction_file**(0,1,2) aufzurufen oder vorübergehend alle Scan-Köpfe abzustecken.

Hinweise

- Encodersignale werden nicht von der Masterkarte an die Slave-Karte(n) weitergeleitet. Sie müssen jeweils separat in die Slave-Karte(n) eingespeist werden. Dabei sollte auch die Takt-Phasen-Verschiebung von 0 ns…50 ns berücksichtigt werden.
- Wenn eine Karte in einer synchronisierten Master/Slave-Kette separat durch eine Zyklus-Synchronisierung extern getaktet wird, so bezieht sich diese Taktung ausschließlich auf diese Karte. Alle anderen Karten in der Master/Slave-Kette werden weiterhin mit dem ursprünglichen Takt der Master-Karte synchronisiert. Wenn die Zyklus-Synchronisation wieder deaktiviert wird, bleibt die betroffene Karte asynchron. Sie kann nur durch nochmaliges Aufrufen von **sync_slaves** wieder synchronisiert werden.
- Korrekturdatei und Listen müssen auf alle RTC6 PCIe-Karten separat geladen werden.

Synchrone Starts und synchrone Stopps

In einer Master/Slave-Kette werden externe Starts (wenn mit **set_control_mode** freigegeben) und externe Stopps durchgereicht:

- von einer Karte zu allen Karten

Daher kann ausgelöst werden:

- Ein gleichzeitiger Start aller synchronisierter Karten einer Master/Slave-Kette (mit vorgebbaren Streckenverzögerungen) durch ein externes Start-Signal, ein **simulate_ext_start** oder **simulate_ext_start_ctrl** an einer beliebigen Karte
- Ein gleichzeitiger Stop aller synchronisierter Karten einer Master/Slave-Kette durch ein externes Stopp-Signal oder ein **simulate_ext_stop** an einer beliebigen Karte

Nicht durchgereicht werden dagegen:

- Starts mit **execute_list**
- Starts mit **execute_at_pointer**
- Stopps mit **stop_execution**

Deshalb müssen diese auch an Master/Slave-synchronisierten Karten separat ausgeführt werden.

Hinweise

- Siehe auch Kapitel 9.3.1 "Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden", Seite 314.
- Mit **master_slave_config** kann für jede RTC6-Karte individuell deren Master/Slave-Schnittstellen-Eigenschaften konfiguriert werden.

Beispiel-Code (Delphi)

Der folgende Delphi-Quellcode zeigt beispielhaft, wie geprüft werden kann, welche RTC6 PCIe-Karte Master, und welche Slave ist.

Der Code muss in ein Anwenderprogramm eingebunden werden, siehe Kapitel 6.2.5 "Beispiel-Code (C)", Seite 103.

```
if init_rtc6_dll() <> 0 then halt;      // Initialize the RTC6-DLL
if rtc6_count_cards() <> 2 then halt;   // Are 2 RTC6 in the PC?

for CardNo := 1 to 2 do                 // Load program and 3D correction files onto both boards
begin
   // Stop RTC6 if a task is currently still running. Optional:
   // n_stop_execution(CardNo);
   if n_load_program_file(CardNo, nil) <> 0 then halt;
   if n_load_correction_file(CardNo, nil, 1, 2) <> 0 then halt;
end;

// Check for Errors (Bit #12...Bit #15)
if (n_get_sync_status(1) and $F000 = 0) and (n_get_sync_status(2) and $F000 = 0) then
begin
   // Detect master board
   // Is board 1 the master and board 2 a single slave?
   if (n_get_master_slave(1) = 2) and (n_get_master_slave(2) = 1) then
   begin
      Master := 1;
      Slave := 2;
   end else
   // Is board 2 the master and board 1 a single slave?
   if (n_get_master_slave(1) = 1) and (n_get_master_slave(2) = 2) then
   begin
      Master := 2;
      Slave := 1;
   end
   else
   halt;                                    // Something wrong with master-slave configuration
end else
halt;

n_select_cor_table(Master, 1, 0);
n_select_cor_table(Slave, 1, 0);
n_set_control_mode(Master, 1 + 8);          // Master slave, activate Start Stop control
n_set_control_mode(Slave, 1 + 8);
n_sync_slaves(Master);                      // Synchronize master and slave boards
// check synchronization status at any time
// provide an external start or a simulated external start before checking
n_simulate_ext_start_ctrl(Master); // for ≤ RBF 618
Result = n_get_sync_status(Master) and $3FF; // must be 640
Result = n_get_sync_status(Slave) and $3FF; // must be <4

// MasterCfg, SlaveCfg must be defined elsewhere, see manual
n_master_slave_config(Master, MasterCfg);
n_master_slave_config(Slave, SlaveCfg);
```

6.7 RTC6 PCIe-Karten durch mehrere Anwenderprogramme verwenden

Die Verwendung von RTC6 PCIe-Karten durch mehrere Anwenderprogramme wird über die (RTC6-DLL-interne) RTC6-Kartenverwaltung koordiniert, siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008. Diese wird durch **init_rtc6_dll** initialisiert.

init_rtc6_dll vergibt einem Anwenderprogramm automatisch (mit **acquire_rtc**) die Zugriffsberechtigung für die gefundenen Karten, wenn die Zugriffsberechtigung nicht bereits an ein anderes Anwenderprogramm vergeben wurde (es können zwar mehrere RTC6 PCIe-Karten oder Anwenderprogramme gleichzeitig verwendet werden, ein und dieselbe Karte kann aber nicht gleichzeitig von mehreren Anwenderprogrammen verwendet werden).

Zugriffsberechtigungen (auch zeitweise) auf Karten werden von der RTC6-DLL exklusiv vergeben.

Mehrere Threads *eines* Anwenderprogramms können zwar ein und dieselbe Karte verwenden, aber nicht gleichzeitig Befehle an sie senden (die RTC6-DLL serialisiert die Befehlsaufrufe automatisch).

Ohne Zugriffsberechtigung ist eine Karte von einem Anwenderprogramm nur mit ausschließlich RTC6-DLL-intern wirkenden Befehlen ansprechbar, die keine Zugriffsberechtigung erfordern (z.B. **get_error**, **get_last_error** und **select_rtc**).

Hat ein Anwenderprogramm die Zugriffsberechtigung für eine Karte erhalten, dann kann diese Karte erst dann wieder von einem anderen Anwenderprogramm übernommen werden, nachdem die Karte mit **release_rtc** oder **free_rtc6_dll** explizit vom zugriffsberechtigten Anwenderprogramm freigegeben wurde.

Bei der Übernahme einer Karte mit **acquire_rtc** (oder **init_rtc6_dll** oder **select_rtc**) erfolgt eine Versionsprüfung der RTC6-DLL, `RTC6OUT.out`, `RTC6RBF.rbf`. Sind diese Dateien noch nicht geladen, kann die Versionsprüfung zwar nicht explizit durchgeführt werden, gilt aber dennoch als bestanden und verhindert damit die Übernahme der Karte nicht. Sind diese Dateien geladen und bei der Versionsprüfung wird ein Fehler festgestellt, dann wird der Zugriff verweigert (**get_last_error**-Returncode `RTC6_ACCESS_DENIED` | `RTC6_VERSION_MISMATCH`).

6.7.1 Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm

init_rtc6_dll, **acquire_rtc**, **free_rtc6_dll**, **release_rtc** und **select_rtc** beeinflussen die Zugriffsberechtigung der RTC6 PCIe-Karten. Sie initialisieren die RTC6 PCIe-Karten nicht.

Ein Anwenderprogramm, das eine Karte mit **acquire_rtc** (oder **init_rtc6_dll** oder **select_rtc**) übernimmt, erhält die Karte mit unbereinigtem Speicher und Betriebszustand. Deshalb kann dieses Anwenderprogramm die gespeicherten Daten und Einstellungen der Karte verwenden und könnte in den Programmablauf eines (durch ein vorheriges Anwenderprogramm) gestarteten Listenprogramms eingreifen.

Falls eine Karte von einem Anwenderprogramm mit **release_rtc** freigegeben und anschließend – ohne zwischenzeitliche Übernahme durch ein anderes Anwenderprogramm – wieder von demselben Anwenderprogramm beansprucht wird, so kann die RTC6 PCIe-Karte unverändert weiterbetrieben werden, da hier auch alle Konfigurationsdaten in der RTC6-DLL unverändert erhalten bleiben. Letzteres gilt aber nicht, wenn die Karte mit **free_rtc6_dll** freigegeben und nach **init_rtc6_dll** wieder übernommen wird.

Wenn eine RTC6 PCIe-Karte von einem anderen Anwenderprogramm übernommen wird, werden einige wichtige, vom vorherigen Anwenderprogramm nur in der RTC6-DLL verwaltete Informationen *nicht* (automatisch) übernommen. Dem übernehmenden Anwenderprogramm fehlen dadurch etwa Informationen bezüglich der Speicherkonfiguration, der Verwaltung des geschützten Bereichs oder dem Betriebsstatus.

Wenn nach einer Kartenübernahme ohnehin mit **load_program_file** eine Karten-Initialisierung ausgeführt wird und alle Einstellungen neu definiert werden, dann spielen solche fehlende Informationen keine Rolle.

Soll das übernehmende Anwenderprogramm die RTC6 PCIe-Karte dagegen im übernommenen Zustand weiterbetreiben, dann muss es die in der RTC6-DLL fehlenden Informationen explizit abfragen, von dem vorherigen Anwenderprogramm übernehmen und erneut explizit setzen, damit RTC6-DLL und Karte konsistent bleiben. Beachten Sie dazu die folgenden Hinweise.

Hinweise

- Für ein korrektes Verhalten des Eingabe-Zeigers an den Listengrenzen muss die in der RTC6-DLL für das übernehmende Anwenderprogramm eingetragene Speicherkonfiguration mit der aktuellen Speicherkonfiguration der übernommenen Karte konsistent sein. **get_config_list** ermittelt die aktuelle Speicherkonfiguration der Karte und stellt sie in der RTC6-DLL entsprechend ein.
- Die Verwaltungstabellen der geschützten Funktionen (indizierte Unterprogramme, Zeichensätze und Zeichenfolgen) befinden sich auf der Karte.
  Alle auf der Karte gespeicherten geschützten Funktionen sind daher weiterhin abrufbar. Verloren geht dagegen die Information, wohin die nächste geschützte Funktion geladen werden soll. Sie kann nur durch **save_disk**/**load_disk**, siehe Abschnitt "Indexverwaltung und Defragmentierung", Seite 118, wiederhergestellt werden. Eine anderweitige Übertragung ist nicht möglich. Das gemischte Laden von geschützten Funktionen durch unterschiedliche Anwenderprogramme sollte daher vermieden werden.
- Der Eingabe-Zeiger wird generell nicht übernommen (es wird die auf der RTC6-DLL für das übernehmende Anwenderprogramm aktuell gespeicherte Eingabe-Zeiger-Position verwendet, gegebenenfalls nach **get_config_list** korrigiert). Ausgabe-Zeiger von Listen können dagegen nach der Übernahme mit **get_status** abgefragt werden.
- Nach der Übernahme ist der Listenstatus (bezüglich LOAD-Listenstatus und READY-Listenstatus) gegebenenfalls bis zum nächsten **load_...**-Aufruf falsch. Das ist aber für die weitere Ausführung unwichtig, und der Status wird beim nächsten **load_...**-Aufruf neu gesetzt.
- Sonstige Einstellungen wie **start_loop** oder die Lasereinstellungen sind in der RTC6-DLL irrelevant. Die vom vorherigen Anwenderprogramm vorgenommenen Einstellungen können in der Regel nicht abgefragt werden, es können aber natürlich neue Einstellungen beliebig vorgenommen werden.
- Die Fehlerbehandlung erfolgt für jede Karte und für jedes Anwenderprogramm getrennt. Beim Austausch von Zugriffsberechtigungen werden diese Daten nicht mitgetauscht.

6.8 Fehlerbehandlung (Error-Handling)

Um Fehler im Programmablauf aufzufangen, führt die RTC6 PCIe-Karte eine allgemeine Fehlerbehandlung durch. Bei einigen Befehlen gibt es auch eine individuelle Fehlerbehandlung.

Allgemeine Fehler treten z.B. auf, wenn:

- Das Anwenderprogramm besitzt keine Zugriffsberechtigung für die Karte (`RTC6_ACCESS_DENIED`)
- Die Karte reagiert nicht auf einen Kontrollbefehl (`RTC6_TIMEOUT`)
- Beim Senden tritt ein PCI-Kommunikationsproblem auf (`RTC6_SEND_ERROR`)

Individuelle Fehler treten z.B. auf, wenn:

- Ein Befehl wird mit einem (nicht korrigierbaren) unzulässigen Parameter aufgerufen (`RTC6_PARAM_ERROR`, siehe z.B. **get_value** oder **write_da_x**)
- Das Senden eines Listenbefehls wird verweigert (`RTC6_REJECTED`, z.B. bei ungültigem Eingabe-Zeiger)
- Ein Kontrollbefehl wird zum falschen Zeitpunkt gesendet (`RTC6_BUSY`, z.B. **goto_xy**, wenn noch eine Liste abgearbeitet wird)

Kontrollbefehle werden in diesen Fällen nicht ausgeführt und für die Listenbefehle wird meist jeweils ein **list_nop** gesendet (z.B. bei `RTC6_PARAM_ERROR` oder `RTC6_IGNORED`, siehe z.B. **set_end_of_list**).

Die diesen Fehlern zugeordneten Bits werden in der RTC6-DLL bei jedem Befehl in kartenspezifischen Fehlervariablen gesetzt oder akkumuliert:

- `LastError`
  - Fehlercode
  - Wird am Anfang eines jeden Befehls automatisch zurückgesetzt
  - Ist daher eine Auflistung der Fehler, die beim zuletzt ausgeführten Befehl aufgetreten sind
  - Kann mit **get_last_error** ausgelesen werden(1)
- `AccError`
  - Akkumulierter Fehlercode
  - Wird beim Initialisieren der RTC6-DLL zurückgesetzt
  - Kann vom Anwenderprogramm selber zurückgesetzt werden mit **reset_error**
  - Sind alle akkumulierten Fehler-Bits seit dem letzten Löschen dieser Bits mit **reset_error**
  - Kann mit **get_error** ausgelesen werden(1)

Die Fehlerbehandlung erfolgt getrennt:

- für jede Karte
- für jedes Anwenderprogramm

Ein **reset_error** löscht nicht den Fehlercode eines anderen Anwenderprogramms, das gerade für die angegebene Karte zugriffsberechtigt ist.

Beim Austausch von Zugriffsberechtigungen werden diese Daten nicht mitgetauscht.

Die Fehlerbehandlung erfolgt ausschließlich beim Bearbeiten der Befehle in der RTC6-DLL und beim Senden zur RTC6 PCIe-Karte.

Die Fehlerbehandlung erfolgt nicht bei der Ausführung eines Listenprogramms.

Einen Beispiel-Code zur Einbindung der kartenspezifischen Fehlervariablen finden Sie in der Befehlsbeschreibung von **get_error**.

(1) Der beschriebene Mechanismus wirkt nur bei Befehlen, die eine Kommunikation mit der RTC6 PCIe-Karte aufbauen. Befehle, die *keine* Kommunikation mit der RTC6 PCIe-Karte aufbauen (wie z.B. **rtc6_count_cards**, **set_rtc4_mode** oder **get_serial_number**) erzeugen oder verändern weder `LastError` noch `AccError` (siehe auch Hinweis bei den entsprechenden Befehlsbeschreibungen, Seite 346).

Einige Kontrollbefehle (wie **init_rtc6_dll**, **load_correction_file** oder **load_program_file**) liefern zusätzlich einen speziellen Fehlercode als Rückgabewert. Dieser wird nicht zwischengespeichert und muss vom Anwenderprogramm sofort ausgewertet oder verworfen werden.

6.8.1 Download-Überprüfung

Insbesondere für medizintechnische Anwendungen ist eine Überprüfung der RTC6-Kommunikation erforderlich. Hierfür kann mit **set_verify** für jede Karte getrennt eine Download-Überprüfung aktiviert werden.

Dadurch verlängern sich jedoch automatisch die Download-Zeiten.

Wenn die Download-Überprüfung aktiviert ist, wird im Fehlerfall der Fehlercode `RTC6_VERIFY_ERROR` gesetzt, der mit **get_last_error** oder **get_error** abgefragt werden kann. Gewisse Operationen werden sofort abgebrochen, die Karte ist dann u.U. nicht mehr funktionsfähig (z.B. wenn **load_program_file** abgebrochen wurde).

Bei aktivierter Download-Überprüfung werden folgende Überprüfungen vorgenommen (beachten Sie auch die Hinweise in der Befehlsbeschreibung von **set_verify**):

(1) Listenbefehl laden

Bei Listenbefehl-Downloads wird jeder Download zurückgelesen und auf Gleichheit zum gesendeten Befehl geprüft. Dabei wird nur der Transfer zur RTC6 PCIe-Karte selbst geprüft, etwaige automatische Parameteranpassungen (z.B. Clipping) werden nicht berücksichtigt.

(2) Kontrollbefehl laden

Bei Kontrollbefehlen werden die zugehörigen Parameter zurückgelesen und auf Gleichheit zu den gesendeten Parametern geprüft. Etwaige automatische Parameteranpassungen werden dabei nicht berücksichtigt.

(3) load_program_file

Beim Senden von **load_program_file** wird folgendes geprüft:

- Die `RTC6DAT.dat` wird über die Check-Summe auf Korrektheit der Datei und auf Korrektheit des PCI-Tansfers geprüft.
- Die `RTC6RBF.rbf` wird nur per Handshake des bitweisen Transfers kontrolliert. Es ist keine zusätzliche Prüfung möglich.
- Die `RTC6OUT.out` wird für jede geladene Sektion unmittelbar durch Zurücklesen überprüft. Bei einem Fehler wird der Ladevorgang abgebrochen.

(4) Korrekturdatei laden
Beim Laden von Korrekturdateien mit **load_correction_file** wird die Integrität der zu ladenden Datei (über die Check-Summe) überprüft und der Transfer selbst auf Richtigkeit überprüft, indem die Korrekturtabelle sofort zurückgelesen wird. Diese Aktion erfordert eine Korrekturdatei mit Check-Summe, siehe Befehlsbeschreibung von **set_verify** und **verify_checksum**.

(5) Tabellen laden
Beim Laden von sonstigen Tabellen (z.B. mit **load_varpolydelay**) wird der Transfer auf Richtigkeit überprüft, indem die Tabelle sofort zurückgelesen wird. Neben dem **get_last_error**-Returncode `RTC6_VERIFY_ERROR` wird auch der entsprechende Fehler-Rückgabewert des Ladebefehls gesetzt.

6.8.2 Überprüfung auf Überläufe

Mit **get_overrun** kann überprüft werden, ob Laufzeitüberschreitungen des 10 μs-Takts stattgefunden haben. Siehe auch Abschnitt "Takt-Überläufe", Seite 190.

6.8.3 Beispiel-Code (C)

Der folgende C-Quellcode zeigt beispielhaft, wie ein Fehler während der Initialisierung aufgefangen werden kann. Er stellt sicher, dass das Anwenderprogramm mit einer Fehlermeldung beendet wird, wenn:

- Während der Initialisierung mit **init_rtc6_dll** ein Fehler auftritt (z.B. wenn keine RTC6 PCIe-Karte gefunden wird)
- Die gewünschte RTC6 PCIe-Karte (hier: die Karte mit der Seriennummer 12345) nicht gefunden wird
- Der Zugriff auf die gewünschte RTC6 PCIe-Karte verweigert wird
- Während **load_program_file** ein Fehler auftritt (z.B. ein VersionMismatch-Fehler, ein Datei- oder System-Fehler)

Der Code muss in ein Anwenderprogramm eingebunden werden, siehe Kapitel 6.2.5 "Beispiel-Code (C)", Seite 103.

```
UINT ErrorCode;
ErrorCode = init_rtc6_dll();
if ( ErrorCode )
{
   // Auslesen der Anzahl der RTC6-Karten, die während der Initialisierung mit init_rtc6_dll
   // gefunden wurden
   const UINT RTC6CountCards = rtc6_count_cards();
   if ( RTC6CountCards )
   {
      // Detaillierte Fehleranalyse für alle gefundenen Karten
      UINT AccError( 0 );
      for ( UINT i = 1; i <= RTC6CountCards; i++ )
      {
         // Fehler, die während der Ausführung von init_rtc6_dll aufgetreten sind
         const UINT Error = n_get_last_error( i );
         if ( Error != 0 )
         {
            AccError |= Error;
            const UINT SerialNumber = n_get_serial_number ( i );
            printf( "RTC6 board number %d (serial number %d): Error %d detected\n",
                        i, SerialNumber, Error );
            n_reset_error( i, Error );
         }
      }
      if ( AccError )
      {
         free_rtc6_dll();
         return;
      }
   }
```

```
   else
   {
      printf( "Initializing the DLL: Error %d detected\n", ErrorCode );
      free_rtc6_dll();
      return;
   }
}
else
{
   // Auslesen der internen Kartennummer der gewünschten RTC6-Karte
   const UINT SerialNumberOfDesiredBoard ( 12345 );
   const UINT RTC6CountCards = rtc6_count_cards();
   UINT InternalNumberOfDesiredBoard ( 0 );
   for ( UINT i = 1; i <= RTC6CountCards; i++ )
   {
      if ( n_get_serial_number( i ) == SerialNumberOfDesiredBoard )
      {
         InternalNumberOfDesiredBoard = i;
      }
   }
   if ( InternalNumberOfDesiredBoard == 0 )
   {
      printf( "RTC6 board with serial number %d not detected.\n", SerialNumberOfDesiredBoard);
      free_rtc6_dll();
      return;
   }
   // Auswählen der gewünschten RTC6-Karte als die aktive RTC6-Karte für dieses Anwenderprogramm
   if ( InternalNumberOfDesiredBoard != select_rtc( InternalNumberOfDesiredBoard ) )
   {
      // Fehler, die während der Ausführung von select_rtc aufgetreten sind
      ErrorCode = n_get_last_error( InternalNumberOfDesiredBoard );
      if ( ErrorCode & 256 ) // RTC6_VERSION_MISMATCH
      {
         if ( ErrorCode = n_load_program_file( InternalNumberOfDesiredBoard, 0 ) )
         {
            printf( "n_load_program_file returned error code %d\n", ErrorCode );
         }
      }
      else
      {
         printf( "No access to RTC6 board with serial number %d\n", SerialNumberOfDesiredBoard );
         free_rtc6_dll();
         return;
      }
      if ( ErrorCode )
      {
         printf( "No access to RTC6 board with serial number %d\n", SerialNumberOfDesiredBoard );
         free_rtc6_dll();
         return;
      }
      else
      {
         // falls n_load_program_file erfolgreich war: Auswählen der gewünschten Karte
         (void) select_rtc( InternalNumberOfDesiredBoard );
      }
   }
}
```

6.9 Verschiedenes

6.9.1 Freie Variablen

Es stehen 8 sogenannte freie Variablen zur Verfügung. Benutzer können diese mit dem Kontrollbefehl **set_free_variable** oder dem kurzen Listenbefehl **set_free_variable_list** frei mit Daten belegen.

Die Variablenwerte können:

- an der McBSP-Schnittstelle ausgegeben werden, siehe auch Kapitel 9.1.7 "McBSP-Schnittstelle", Seite 312
- mit **get_free_variable** und **get_value** zurückgelesen werden (siehe Befehlsbeschreibungen)
- mit **set_trigger[*]** aufgezeichnet werden (siehe Befehlsbeschreibungen)

Mithilfe der freien Variablen können z.B. Steuerbefehle über die McBSP-Schnittstelle an die Benutzer-Hardware übertragen oder Betriebszustände der Karte (z.B. bei Verzweigungen) dokumentiert werden.

Hinweise

- Durch **set_free_variable** und **set_free_variable_list** können beliebige 32-Bit-Werte ohne Vorzeichen als Variablenwerte festgelegt werden. Über die McBSP-Schnittstelle werden allerdings mit **set_mcbsp_out_ptr** nur jeweils die unteren 24 Bit und mit **set_mcbsp_out** nur jeweils die unteren 16 Bit ausgegeben (**get_free_variable**, **get_value** und **set_trigger[*]** liefern dagegen die vollen 32 Bit-Werte zurück).

7 Grundlegende Funktionen zur Scan-Kopf-Steuerung und Laser-Steuerung

7.1 Punkte, Linien und Bögen markieren

7.1.1 Mit Vektor-Befehlen und Bogen-Befehlen markieren

Wie in Kapitel 6.1 "Grundlegendes zum RTC6 Software-Konzept", Seite 96 dargestellt, wird die Positionierung der Achsen des Scan-Systems und somit des Laserstrahls bei Ansteuerung mit der RTC6 PCIe-Karte erreicht durch Aufrufen von:

- Sprung-Befehlen
- Mark-Befehlen
- Arc-Befehlen
- Ellipse-Befehlen

Jeder dieser Befehle beschreibt einen einzelnen Vektor oder Bogen(1). Mithilfe der micro_vector[*]-Befehle können auch beliebig geformte Trajektorien implementiert werden.

Auch beim Markieren von Zahlen oder Buchstaben werden die einzelnen Zeichen letztlich über einzelne Linien, Punkte und Bögen definiert, siehe Kapitel 7.5 "Uhrzeit, Datum und Seriennummern markieren", Seite 222.

Bei Vektor-Befehlen (Sprung-Befehle, Mark-Befehle) ist als Parameter jeweils der *Endpunkt* des entsprechenden Vektors anzugeben(2). Als Startpunkt eines Vektors wird der Endpunkt des zuletzt abgearbeiteten Vektors oder Bogens verwendet.

Bei Bogen-Befehlen (Arc-Befehle, Ellipse-Befehle) sind als Parameter die Koordinaten des Bogenzentrums und der/die Bogenwinkel anzugeben. Der Arc-Bogen beginnt an der aktuellen Ausgabeposition. Der Ellipsenbogen beginnt an der mit den Befehlsparametern angegebenen Position. Ein direkter Anschluss an die letzte Ausgabeposition muss vom Anwenderprogramm mit geeigneten Parametern explizit selbst sichergestellt werden. Anderenfalls gibt es einen harten Sprung (siehe "Harter Sprung") dorthin.

Nach einem Hardware-Reset der RTC6 PCIe-Karte befindet sich die Ausgabeposition in der Bildfeldmitte im Punkt (0|0). Das Koordinatensystem des Bildfelds wird in Kapitel 7.3 "Scan-Kopf-Steuerung", Seite 175 beschrieben.

Jeder Vektor oder Bogen, der vom Scan-System abgefahren werden soll, wird von der RTC6 PCIe-Karte zur Laufzeit in Mikroschritte zerlegt, siehe Kapitel 7.1.2 "Zerlegung in Mikroschritte", Seite 143(3).

Sprung-Befehle dienen dazu, die Achsen des Scan-Systems bei *ausgeschaltetem* Laser zu einer neuen Position zu bewegen.

Markier-Befehle dagegen veranlassen eine Markierbewegung bei *eingeschaltetem* Laser (siehe auch nachfolgende Beschreibung).(4)

Für das Markieren eines Punktes müssen die Signale für den "Laser active"-Betrieb nach einem Sprung-Befehl oder [*]mark[*]-Befehl für die gewünschte Zeitdauer eingeschaltet werden, siehe Kapitel 7.1.3 "Einzelne Punkte markieren", Seite 144.

Beim Markieren von Linien und Bögen wird das Anschalten der Signale für den "Laser active"-Betrieb zu Beginn eines Markier-Befehls und das Wiederausschalten z.B. bei einem nachfolgenden Sprung-Befehl automatisch von der RTC6 PCIe-Karte erledigt.

Die Synchronisation von Scan-Kopf-Steuerung und Laser-Steuerung kann dabei vom Benutzer durch Einstellen von Delays auf die jeweilige Anwendung abgestimmt werden, siehe Kapitel 7.2 "Delayeinstellungen – Scan-Kopf-Steuerung und Laser-Steuerung abstimmen", Seite 148.

(1) Verstärkte Linienbreiten können dabei über **set_wobbel_mode** eingestellt werden.

(2) Die Koordinaten sind als digitale Ansteuerwerte (einheitenlos) anzugeben. Zur Vermeidung von Verwechslungen mit Koordinatenwerten in [mm] verwendet SCANLAB den Ausdruck "Koordinatenwert [in Bits]"

(3) Nur iDRIVE-Scan-Systeme, die mit einem entsprechenden Tuning ausgestattet sind, können Sprünge auch im Sprungmodus ausführen, siehe Kapitel 8.1.5 "Sprungmodus", Seite 229.

(4) Außerhalb einer Liste ist eine Neupositionierung (auch bei eingeschalteten Lasersteuersignalen) mit **goto_xy** oder **goto_xyz** möglich.

Die Einstellung der Laserparameter wird in Kapitel 7.4 "Laser-Steuerung", Seite 192 beschrieben.

Ein ausführlich kommentierter Beispiel-Code für eine einfache Markieraufgabe ist in Kapitel 7.1.4 "Beispiel-Code (C)", Seite 145 gezeigt.

Sprung-Befehle

Ein Sprung-Befehl (**jump_abs** oder **jump_rel**(1)) veranlasst eine Bewegung der Spiegel vom Anfangs- zum Endpunkt eines Vektors.

Die Signale für den "Laser active"-Betrieb werden dazu am Anfang des Vektors automatisch ausgeschaltet und bleiben während des Sprungs ausgeschaltet, siehe auch Kapitel 7.2.1 "Laser-Delays", Seite 148 und Kapitel 7.2.2 "Scanner-Delays", Seite 150. Die Sprunggeschwindigkeit wird mit **set_jump_speed** oder **set_jump_speed_ctrl** festgelegt.

Wird ein Laser verwendet, der nicht schnell geschaltet werden kann, muss die Sprunggeschwindigkeit so groß gewählt werden, dass die Oberfläche des Werkstücks entlang des Sprung-Vektors nicht sichtbar verändert wird. Siehe auch **home_position** und **home_position_xyz**.

Mark-Befehle

Bei Ausführung eines [*]mark[*]-Befehls (**mark_abs** oder **mark_rel**(1)) wird der Laserfokus geradlinig vom Anfangs- bis zum Endpunkt eines Vektors geführt. Die RTC6 PCIe-Karte schaltet zu Beginn eines [*]mark[*]-Befehls automatisch die Signale für den "Laser active"-Betrieb ein, siehe auch Abschnitt "Polygonzüge", Seite 140.

Die Markiergeschwindigkeit wird mit **set_mark_speed** oder **set_mark_speed_ctrl** definiert. Sie ist im Allgemeinen kleiner als die Sprunggeschwindigkeit und kann mit **set_mark_speed** an jeder beliebigen Stelle innerhalb einer Liste geändert werden oder mit **set_mark_speed_ctrl**, wenn gerade keine Liste abgearbeitet wird.

Arc-Befehle

Mit den Arc-Befehlen **arc_abs** und **arc_rel** können Kreisbögen markiert werden(2). Als Parameter sind die Koordinaten des Bogenzentrums und der Bogenwinkel anzugeben. Der Kreisbogen beginnt an der aktuellen Ausgabeposition, der Winkel wird im Uhrzeigersinn positiv gezählt (entgegen der mathematischen Definition).

Bei Ausführung eines Arc-Befehls wird der Laserfokus mit der angegebenen Geschwindigkeit entlang des angegebenen Bogens geführt. Die Signale für den "Laser active"-Betrieb werden zu Beginn der Ausführung eines Arc-Befehls automatisch eingeschaltet, siehe auch Abschnitt "Polygonzüge", Seite 140.

Polygonzüge

Wenn unmittelbar auf einen [*]mark[*]-Befehl ein weiterer [*]mark[*]-Befehl (oder ein Bogen-Befehl) folgt ("Polygonzug"), dann bleiben die Signale für den "Laser active"-Betrieb eingeschaltet.

Durch direktes Aneinanderreihen von [*]mark[*]-Befehlen (und Bogen-Befehlen) kann daher kontinuierlich markiert werden.

Die Signale für den "Laser active"-Betrieb werden am Anfang des (normalen) Befehls ausgeschaltet, der dem letzten [*]mark[*]-Befehl eines Polygonzugs folgt.

Siehe auch Abschnitt "`EdgeLevel`", Seite 155.

(1) Zur Verwendung von abs- und rel-Befehlen, siehe Abschnitt ""AbsCalls"", Seite 117. Zusätzlich stehen zur Verfügung: zeitbemessene Vektor-Befehle, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286, Para-Vektor-Befehle, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217 und 3D-Vektor-Befehle, siehe Kapitel "3D-Befehle", Seite 250.

(2) Zur Verwendung von abs- und rel-Befehlen, siehe Abschnitt ""AbsCalls"", Seite 117. Zusätzlich stehen zeitbemessene Arc-Befehle zur Verfügung, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286.

Ellipse-Befehle

Der RTC6-Befehlssatz enthält auch Befehle zum Markieren ellipsenförmiger Bögen.

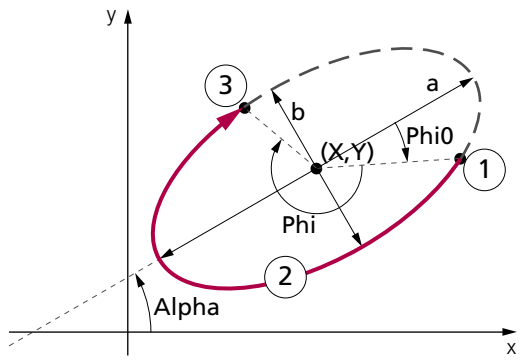
Im Gegensatz zum Markieren von Vektoren oder Kreisbögen müssen dabei allerdings in der Regel jeweils zwei Befehle aufgerufen werden: **set_ellipse** sowie **mark_ellipse_abs** oder **mark_ellipse_rel**(1).

Mit **set_ellipse** wird die Bogenform festgelegt, siehe Abbildung 33:

- die Längen `a` und `b` der Ellipsenhalbachsen
- der Anfangsphasenwinkel `Phi0` (und damit die Lage des Bogenstartpunkts relativ zum Endpunkt der Halbachse `a`)
- der Bogenwinkel `Phi` (und damit die Länge des zu markierenden Ellipsenausschnitts)

Mit **mark_ellipse_abs** oder **mark_ellipse_rel** wird die Lage und Orientierung des auszuführenden Bogens festgelegt, siehe Abbildung 33:

- die Koordinaten `(X,Y)` des Ellipsen-Mittelpunkts
- der Winkel `Alpha` zwischen der Ellipsen-Halbachse `a` und der x-Achse



Legende
1 Startpunkt
2 Markierter Bogen
3 Endpunkt

33

Markieren ellipsenförmiger Bögen.

Hinweise

- Mit `a` kann wahlweise die kurze oder lange Halbachse angegeben werden (mit `b` die jeweils andere Achse). `Phi0`, `Phi` und `Alpha` beziehen sich aber immer auf die Achse `a`.
- `Phi0` und `Phi` werden im Uhrzeigersinn positiv gezählt (entgegen der mathematischen Definition), dagegen `Alpha` im Gegenuhrzeigersinn (entsprechend der mathematischen Definition).
- Wie bei Mark-Befehlen und Arc-Befehlen wird bei Ellipse-Befehls-Ausführung der Laserfokus mit der angegebenen Geschwindigkeit entlang des angegebenen Bogens geführt. Die Signale für den "Laser active"-Betrieb werden zu Beginn der Ausführung eines Bogen-Befehls automatisch eingeschaltet, siehe auch Abschnitt "Polygonzüge", Seite 140.
- **set_ellipse** ist ein Kurzer Listenbefehl. Er kann daher zwischen einem Markier-Befehl und einem Ellipse-Befehl aufgerufen werden, ohne dass dadurch der Polygonzug unterbrochen wird (der Laser bleibt an).
- Bei Ellipse-Befehlen beginnt die Markierung immer an dem mit Hilfe der oben genannten Parameter festgelegten Startpunkt (im Gegensatz zu Mark-Befehlen und Arc-Befehlen, bei denen die Markierung automatisch an der aktuellen Ausgabeposition beginnt). Stimmen Startpunkt und aktuelle Position nicht überein, dann wird am Anfang der Markierung ein Harter Sprung an den Startpunkt ausgeführt (ohne dass ein Jump-Delay wirksam wird).

(1) Zur Verwendung von abs- und rel-Befehlen, siehe Abschnitt ""AbsCalls"", Seite 117.

- Ellipsenförmige Bögen können auch mit Kreis-Arc-Befehlen (wie **arc_abs**) markiert werden, wenn mit **set_matrix** eine geeignete Koordinatentransformation (z.B. eine in x-Richtung/y-Richtung unterschiedliche Skalierung) eingestellt ist. Hierbei variiert jedoch die effektive Markiergeschwindigkeit entlang des Bogens, siehe auch den Hinweis auf Seite 239.

  Bei **mark_ellipse_abs** und **mark_ellipse_rel** wird dagegen die Schrittweite in jedem 10 μs-Intervall an die Ellipsenform der jeweils aktuellen Position so angepasst, dass der Bogen weitgehend mit einer konstanten Markiergeschwindigkeit markiert wird.

  Wegen der im 10 μs-Takt nur stückweise angenäherten Ellipsenform kann es allerdings im Bereich der Endpunkte der großen Halbachse bei sehr starken Exzentrizitäten und auch bei hohen Bahngeschwindigkeiten zu numerischen Ungenauigkeiten kommen. Dadurch bleibt dort die effektive Markiergeschwindigkeit u.U. nicht exakt konstant (Beispiel: bei einer Exzentrizität $a/b = 2$ und bei 100 Mikroschritten pro Umfang beträgt die Geschwindigkeitsabweichung ca. 3,7%). Der ausgegebene Punkt liegt aber immer exakt auf der Ellipse.

  Da es keine geschlossene Formel zur Berechnung einer elliptischen Bogenlänge gibt, ist außerdem die Schrittweite des zuletzt markierten Mikroschritts in der Regel kürzer und die Markiergeschwindigkeit entsprechend geringer als die vorgegebene Markiergeschwindigkeit. Die Endposition ist jedoch immer exakt.

  Auch bei Verwendung von Sky-Writing kann es beim Vor- und Rücklauf im Bereich der großen Halbachse zu Unstimmigkeiten kommen. Der Benutzer muss selber sicherstellen, dass die verwendeten Parameter-Werte mit der erforderlichen Präzision im Einklang stehen.

[*]Para[*]-Befehle

- **para_jump_abs**
- **para_jump_abs_3d**
- **para_jump_rel**
- **para_jump_rel_3d**
- **para_laser_on_pulses_list** (Sonderfall)
- **para_mark_abs**
- **para_mark_abs_3d**
- **para_mark_rel**
- **para_mark_rel_3d**
- **timed_para_jump_abs**
- **timed_para_jump_abs_3d**
- **timed_para_jump_rel**
- **timed_para_jump_rel_3d**
- **timed_para_mark_abs**
- **timed_para_mark_abs_3d**
- **timed_para_mark_rel**
- **timed_para_mark_rel_3d**

Bei aktivierter Vektorkontrollierte Laser-Steuerung variieren diese Befehle gleichzeitig einen Signalparameter linear entlang des Mark-Vektors oder Sprung-Vektors, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217.

[*]para_mark[*]-Befehle berücksichtigen grundsätzlich kein Sky-Writing.

7.1.2 Zerlegung in Mikroschritte

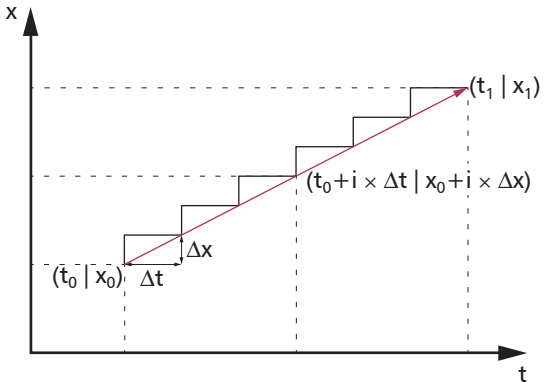
Die RTC6 PCIe-Karte zerlegt jeden

- Sprung-Befehl,
- [*]mark[*]-Befehl und
- Bogen-Befehl

in sogenannte

- Mikroschritte
  (nicht: Mikrovektoren, siehe Kapitel 8.8 "micro_vector[*]-Befehle", Seite 285).

Die Zerlegung der x-Komponente eines Vektors in Mikroschritte zeigt Abbildung 34.



x_0 x-Koordinate der aktuellen Ausgabe-Position vor der Ausgabe des Vektors.
x_1 x-Koordinate des Vektor-Endpunkts.
Δt Ausgabeintervall 10 μs.

34

Die x-Komponente eines Vektors wird in Mikroschritte zerlegt. Die Zerlegung der y-Komponente erfolgt analog.

Alle Mikroschritte werden mit konstantem Ausgabeintervall Δt von 10 μs an den Scan-Kopf übertragen und kann *nicht* verändert werden.

Es gilt:

$$\text{Länge } \Delta s \text{ eines Mikroschritts} = v \times \Delta t \ ^{(1)}$$

(v = momentane Sprunggeschwindigkeit oder Markiergeschwindigkeit)

Hinweise

- Frei definierte Kurvenformen können mit micro_vector[*]-Befehlen realisiert werden, siehe Kapitel 8.8 "micro_vector[*]-Befehle", Seite 285.
- Nur iDRIVE-Scan-Systeme, die mit einem entsprechenden Tuning ausgestattet sind, können Sprünge auch im Sprungmodus ausführen, siehe Kapitel 8.1.5 "Sprungmodus", Seite 229.

(1) Alternativ siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286.

7.1.3 Einzelne Punkte markieren

Um einen einzelnen Punkt zu markieren, müssen die Signale für den “Laser active”-Betrieb für die gewünschte Zeitdauer eingeschaltet werden, siehe **laser_on_list**, **laser_on_pulses_list**, **para_laser_on_pulses_list** und Kapitel 7.4 “Laser-Steuerung”, Seite 192.

Alternativ kann ein einzelner Punkt auch durch einen zeitbemessenen Markier-Befehl der Länge Null markiert werden, siehe Kapitel 8.9 “Zeitbemessene Befehle”, Seite 286.

7.1.4 Beispiel-Code (C)

Der folgende C-Quellcode zeigt beispielhaft die Befehle für eine einfache Laser-Scan-Anwendung.

Ein Punkt, ein Quadrat und ein Kreis werden im CO_2-Modus markiert. Es wird hier angenommen, dass der RTC4-Kompatibilitätsmodus aktiviert ist.

Der Code muss in ein Anwenderprogramm eingebunden werden, siehe Kapitel 6.2.5 "Beispiel-Code (C)", Seite 103.

```
// Scan-System-Initialisierung
   // Eine Korrekturdatei laden und zuweisen
   ErrorCode = load_correction_file( 0,   // initialisieren wie mit "D2_1to1.ct5",
                                     1,   // Tabelle (in der Standardeinstellung wird #1 verwendet)
                                     2 ); // nur 2D-Anteil verwenden
   if ( ErrorCode )
   {
      printf( "Correction file loading error: %d\n", ErrorCode );
      free_rtc6_dll();
      return;
   }
   select_cor_table( 1, 0 ); // Korrekturtabelle #1 wird Scan-Kopf-Anschluss #1 zugewiesen (Default)

// Initialisierung der Laser-Steuerung, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192.
   // CO2-Modus einstellen
   set_laser_mode( 0 );

   // Einstellen und Freigeben der Signale für den "Laser active"-Betrieb
   set_laser_control( 0x18 ); // All laser signals active-low (Bit #3 and Bit #4)
      // set_laser_control muss mindestens einmal aufgerufen werden, um die Lasersteuersignale
      // zu aktivieren. Danach kann auch enable_laser/disable_laser verwendet werden.

   // Eröffnen der Liste 1
   set_start_list( 1 );

   // Einstellen der Standby-Pulse
   set_standby_list( 800, 8 );
      // Im RTC4-Kompatibilitätsmodus werden die Standby-Parameter in Einheiten von 1/8 µs angegeben.
      // Die RTC6-Karte multipliziert die angegebenen Werte mit 8, um sie in ganzzahlige
      // Vielfache von 1/64 µs zu konvertieren. Halbe Standby-Ausgabeperiode = 100 µs.
      // Pulslänge der Standby-Pulse = 1 µs.

// Einstellen des Timings, der Delays und Geschwindigkeiten
   // Einstellen der Scanner-Delays, siehe Kapitel 7.2.2 "Scanner-Delays", Seite 150.
   set_scanner_delays( 25, 10, 5 );
      // Jump-Delay = 250 µs (angegeben in [10 µs])
      // Mark-Delay = 100 µs (angegeben in [10 µs])
      // Polygon-Delay = 50 µs (angegeben in [10 µs])
```

```
   // Einstellen der Sprunggeschwindigkeit und Markiergeschwindigkeit:
   set_jump_speed( 1000.0 );
   set_mark_speed( 250.0 );
      // Im RTC4-Kompatibilitätsmodus multipliziert die RTC6-Karte die Geschwindigkeits-Werte mit 16.
      // Sprunggeschwindigkeit = 1000.0 Bits/ms.
      // Markiergeschwindigkeit = 250.0 Bits/ms.

   // Einstellen des Laser-Timings, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192.
   set_laser_pulses( 800, 400 );
      // Im RTC4-Kompatibilitätsmodus werden die Timing-Parameter in Einheiten von 1/8 µs angegeben.
      // Die RTC6-Karte multipliziert die angegebenen Werte mit 8, um sie in
      // ganzzahlige Vielfache von 1/64 µs zu konvertieren.
      // Laser-HalfPeriod = 100 µs.
      // Laser-Pulslänge = 50 µs.

   // Einstellen der Laser-Delays, siehe Kapitel 7.2.1 "Laser-Delays", Seite 148.
   set_laser_delays( 100, 100 );
      // Im RTC4-Kompatibilitätsmodus werden die Laser-Delays in Einheiten von 1 µs angegeben.
      // Die RTC6-Karte multipliziert die angegebenen Werte mit 32, um sie in
      // ganzzahlige Vielfache von 1/64 µs zu konvertieren.
      // LaserOn-Delay = 100 µs.
      // LaserOff-Delay = 100 µs.

   // Definieren des Listenendes und des Endes der Befehlsübertragung zur RTC6-Karte
   set_end_of_list();

   // Ausführen der Listenbefehle zur Initialisierung
   execute_list( 1 );

// Markier-Prozedur
   // Warten, bis Liste 1 nicht mehr Busy ist (load_list( 1, 0 ) gibt im Erfolgsfall 1 zurück, sonst 0);
   // wenn Liste 1 nicht (mehr) Busy ist: Öffnen des RTC6-Listenspeichers zum Schreiben von Listenbefehlen
   // und Setzen des Eingabe-Zeigers auf den Anfang Liste 1
   while ( !load_list( 1, 0 ) );

   // Im Folgenden werden die Listenbefehle zum Markieren von Punkt, Quadrat und Kreis definiert
   // und an die RTC6-Karte übertragen.

   // Markieren des Bildfeld-Mittelpunkts:
   jump_abs( 0, 0 ); // Sprung zum Bildfeld-Mittelpunkt
      // Nach dem Sprung wird automatisch ein Jump-Delay eingefügt.
   // Anschalten der Lasersteuersignale für 50 µs (+ LaserOff-Delay - LaserOn-Delay):
   laser_on_list( 5 );
```

```
// Markieren eines Quadrats um den Bildfeld-Mittelpunkt:
jump_abs( -20000, -20000 ); // Sprung zur linken unteren Ecke des Quadrats
   // Nach dem Sprung wird automatisch ein Jump-Delay eingefügt.
mark_abs( -20000,  20000 ); // Markieren der linken Kante des Quadrats
mark_abs(  20000,  20000 ); // Markieren der oberen Kante des Quadrats
mark_abs(  20000, -20000 ); // Markieren der rechten Kante des Quadrats
mark_abs( -20000, -20000 ); // Markieren der unteren Kante des Quadrats
   // Die Lasersteuersignale werden automatisch mit dem ersten [*]mark[*]-Befehl
   // (nach einem LaserOn-Delay) eingeschaltet und bleiben für alle 4 [*]mark[*]-Befehle
   // eingeschaltet. Nach den ersten drei [*]mark[*]-Befehlen wird jeweils ein
   // Polygon-Delay automatisch eingefügt. Initiiert durch den nachfolgenden Nicht-Markierbefehl
   // Sprung-Befehl, siehe unten), wird nach dem letzten [*]mark[*]-Befehl automatisch ein
   // Mark-Delay eingefügt und die Lasersteuersignale werden automatisch nach
   // einem LaserOff-Delay ausgeschaltet, weil ein Sprung-Befehl folgt.

// Markieren eines Kreises rund um den Bildfeld-Mittelpunkt:
jump_abs( 0, -10000 ); // Sprung zur unteren Kante des Kreises
   // Nach dem Sprung wird automatisch ein Jump-Delay eingefügt.
arc_abs( 0, 0, 360.0 ); // Markieren des Kreises
   // Die Lasersteuersignale werden automatisch mit dem Arc-Befehl (nach einem LaserOn-Delay)
   // eingeschaltet. Initiiert durch den nachfolgenden Nicht-Markierbefehl
   // (set_end_of_list, siehe unten), wird nach dem Arc-Befehl automatisch ein Mark-Delay
   // eingefügt und die Lasersteuersignale werden automatisch nach
   // einem LaserOff-Delay ausgeschaltet, weil ein set_end_of_list folgt.

// Definieren des Listenendes und des Endes der Befehlsübertragung zur RTC6-Karte
set_end_of_list();

// Starten der übertragenen Liste
execute_list( 1 );
```

7.2 Delayeinstellungen – Scan-Kopf-Steuerung und Laser-Steuerung abstimmen

Die Scan-Kopf-Steuerung und Laser-Steuerung sollte zum dynamischen Verhalten der Systemkomponenten passen, d.h. auf:

- das Ansprechverhalten des Lasers
- das Ansprechverhalten der Galvanometerscanner (Schleppverzug)
- die Art der Wechselwirkung zwischen Laserstrahlung und Material

Dazu stehen die folgenden Delays zur Verfügung:

- Laser-Delays
  - LaserOn-Delay
  - LaserOff-Delay
- Scanner-Delays
  - Jump-Delay (wahlweise: variabel)
  - Mark-Delay
  - Polygon-Delay (wahlweise: variabel)

7.2.1 Laser-Delays

Es gibt zwei verschiedene Laser-Delays:

- LaserOn-Delay
- LaserOff-Delay

Die Laser-Delays bestimmen, wann die Signale für den "Laser active"-Betrieb ein- und ausgeschaltet werden.

Laser-Delays haben in der Regel *keinen* Einfluss auf die Gesamtmarkierungszeit. Ausnahmen:

- Ein negativer `LaserOnDelay`-Wert, siehe Abschnitt "LaserOn-Delay", Seite 149
- Künstlich eingefügte Delays, siehe Abschnitt "Automatische Delay-Anpassung", Seite 158

LaserOn-Delay und LaserOff-Delay werden durch den unverzögerten kurzen Listenbefehl **set_laser_delays** eingestellt. Ihre Einheit ist jeweils *1/64 µs*.

Um Einbrenn-Effekte an den Start- und Endpunkten einer Markierung zu vermeiden, sollte der Laserfokus mit möglichst konstanter Geschwindigkeit bewegt werden. Die Laser-Delay-Zeiten sind daher auf den Schleppverzug des Scan-Kopfs und die eingestellte Markiergeschwindigkeit abzustimmen(1), siehe auch Kapitel 7.2.3 "Hinweise zur Optimierung der Delays", Seite 158.

(1) Darüber hinaus kann zur Optimierung eine automatische Nachregelung der Laserleistung während der Markierung angewendet werden, siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208.

LaserOn-Delay

Ein LaserOn-Delay wird automatisch beim Start eines einzelnen Markier-Befehls sowie beim Start einer Abfolge von Markier-Befehlen ("Polygonzug") eingefügt und verzögert das Einschalten des Lasers.

Es kann zu verschiedenen Zwecken eingesetzt werden:

- Zu Beginn einer Markierung müssen die Spiegel (gegebenenfalls aus dem Stand heraus) auf die angegebene Markiergeschwindigkeit beschleunigt werden, siehe Abbildung 37. Diese Phase kann mit einem ausreichend großen positiven LaserOn-Delay-Wert so weit ausgeblendet werden, bis die Spiegel bereits eine gewisse Winkelgeschwindigkeit erreicht haben, bevor der Laser eingeschaltet wird. Andererseits darf das LaserOn-Delay auch nicht zu lange gewählt werden, sonst wird der erste Teil der Markierung abgeschnitten.
- Bei einigen Materialien/Anwendungen (z.B. Schweißen) dauert es eine gewisse Zeit, bis sie auf die Laserbestrahlung wie gewünscht reagieren. In diesem Fall kann es sinnvoll sein, den Startpunkt der Markierung "vorzuheizen". Das kann durch Einstellen eines *negativen* LaserOn-Delay-Werts erreicht werden. Allerdings verlängert das die Gesamtmarkierungszeit, weil das LaserOn-Delay vor dem aktuellen Markier-Befehl als Scanner-Delay eingefügt wird.(1)

LaserOff-Delay

Das Ende einer Markierung wird nicht durch die Markierung selbst definiert, sondern durch den ersten "normalen" Listenbefehl, der kein Markier-Befehl ist, z.B. ein Sprung-Befehl.

Die Beschleunigungsphase zu Beginn einer Bewegung führt zu einer Zeitdifferenz zwischen der jeweiligen Soll-Position und Ist-Position der Spiegel, nämlich dem Schleppverzug, siehe Abbildung 37.

Der Laser soll erst dann ausgeschaltet werden, wenn der *Ist-Wert* der Endposition erreicht ist (und nicht schon beim *Soll-Wert*). Deshalb wird automatisch nach dem Ende jeder Markierung durch den ersten Normaler Listenbefehl (kein Kurzer Listenbefehl) ein LaserOff-Delay eingefügt(2), siehe auch Abschnitt "Hinweise", Seite 152. Damit kann der Schleppverzug des Scan-Kopfs kompensiert werden, siehe Abbildung 37.

(1) Dieses Scanner-Delay wird automatisch verlängert, wenn ein vorausgehendes LaserOff-Delay noch nicht abgelaufen ist, siehe Abschnitt "Automatische Delay-Anpassung", Seite 158.

(2) Im DSP-Modus < 3 (siehe **set_dsp_mode**) gilt bei kurzen Markier-Vektoren: wenn ein vorangegangenes LaserOn-Delay noch nicht abgelaufen ist, wird das LaserOff-Delay vorübergehend automatisch entsprechend verlängert, siehe Abschnitt "Automatische Delay-Anpassung", Seite 158.

7.2.2 Scanner-Delays

Es gibt drei verschiedene Typen von Scanner-Delays:

- Jump-Delays (wahlweise: variabel)
- Mark-Delays
- Polygon-Delays (wahlweise: variabel)

Nach jedem Sprung-Befehl und Markier-Befehl fügt die RTC6 PCIe-Karte eines dieser Scanner-Delays ein, bevor der nächste Befehl ausgeführt wird (wenn im Einzelfall nichts anderes angegeben ist).

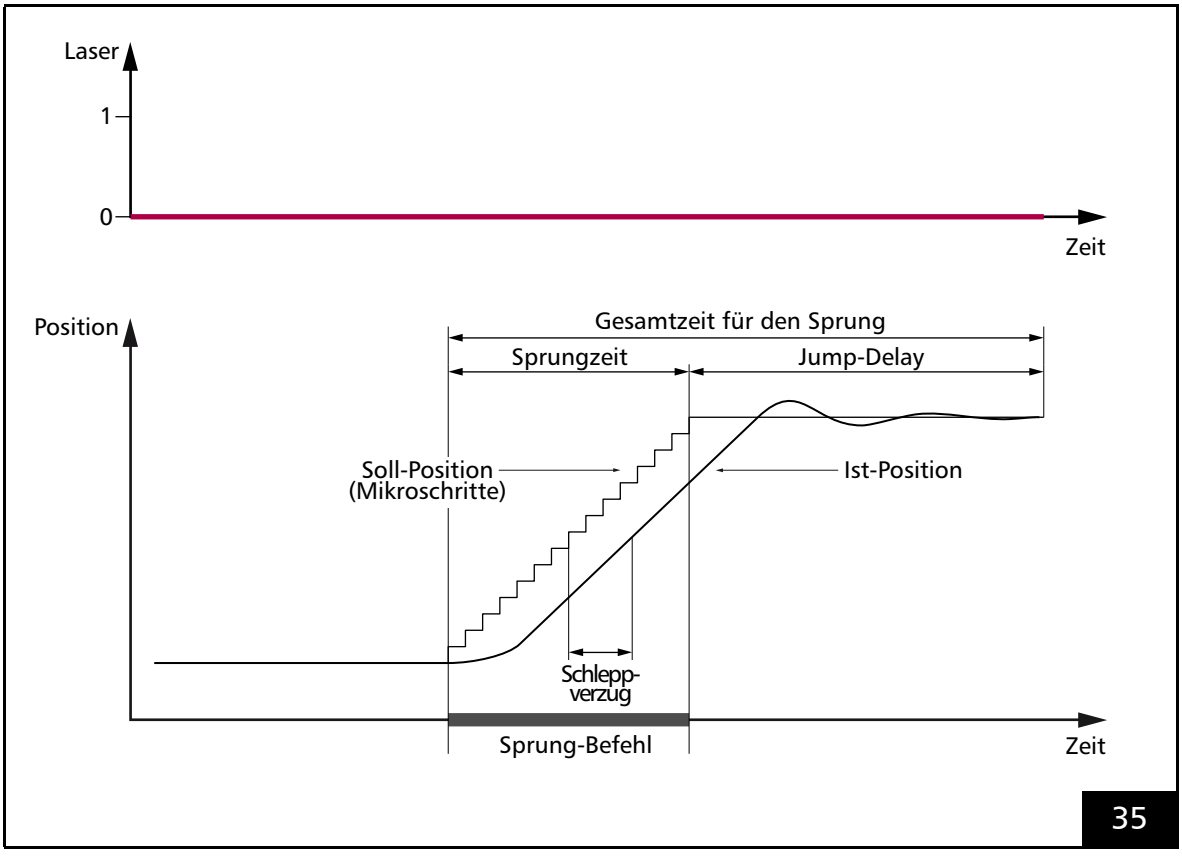
Diese Scanner-Delays werden mit **set_scanner_delays** definiert. Ihre Einheit ist jeweils *10 µs*.

Jump-Delay

Das Jump-Delay wird bei **set_scanner_delays** mit dem Parameter `Jump` angegeben.

Einen typischen Verlauf für einen einzelnen Sprung-Befehl mit dem zugehörigen Jump-Delay zeigt Abbildung 35.

Die Gesamtzeit für einen Sprung setzt sich aus der Sprungzeit und der Dauer des Jump-Delays zusammen. Die Dauer des Jump-Delays sollte zu den dynamischen Eigenschaften des Scan-Kopfs (Schleppverzug, Tuning) sowie der eingestellten Sprunggeschwindigkeit passen.



Scan-Kopf-Steuerung während eines Sprung-Befehls mit konstantem Jump-Delay.
Der Laser bleibt aus.

Variable Jump-Delays

Bei kurzen Sprüngen erreicht der Scan-Kopf oftmals nicht die volle Sprunggeschwindigkeit. Dann genügen auch *kürzere* Jump-Delays, bis die Spiegel wieder in Ruhe sind:

- "Variable Jump-Delays"

Dafür gibt es den:

- "Variable Jump-Delays"-Modus

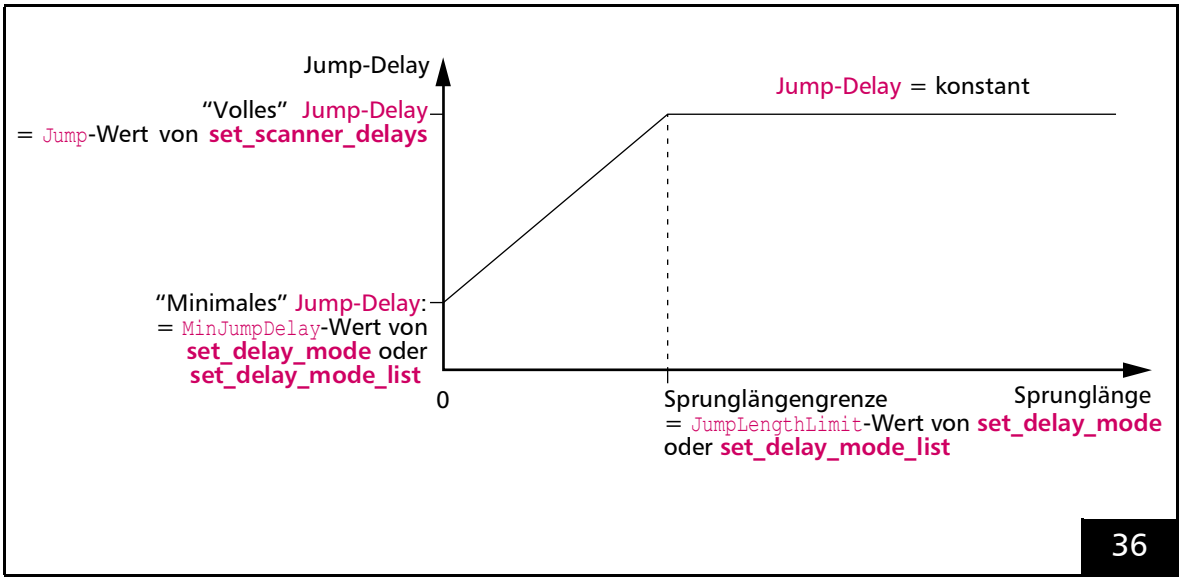
Der "Variable Jump-Delays"-Modus wird mit `JumpLengthLimit` > 0 (bei **set_delay_mode** oder **set_delay_mode_list**) eingeschaltet.

Dann fügt die RTC6 PCIe-Karte bei Sprunglängen zwischen 0 und Sprunglängengrenze ein zwischen `MinJumpDelay` (aus **set_delay_mode** oder **set_delay_mode_list**) und `Jump` (Jump-Delay aus **set_scanner_delays**) linear interpoliertes Jump-Delay ein, siehe Abbildung 36.

Bei Sprunglängen größer als `JumpLengthLimit` wird immer ein "volles" Jump-Delay eingefügt.

Hinweise

- Mit dem "Variable Jump-Delays"-Modus reduziert sich die Gesamtmarkierungszeit, insbesondere bei vielen kurzen Sprüngen.
- Der "Variable Jump-Delays"-Modus wird mit `JumpLengthLimit` = 0 (bei **set_delay_mode** oder **set_delay_mode_list**) ausgeschaltet.
- Nach Sprung-Vektoren der Länge 0 ist die Dauer von "Variable Jump-Delays" 0.
- Das "minimale" Jump-Delay sollte nicht größer als das "normale "Jump-Delay sein. Anderenfalls können "Variable Jump-Delays" sehr groß werden.



"Variable Jump-Delays"-Modus: Jump-Delay-Wert in Abhängigkeit der Sprunglänge.

Mark-Delay

Das Mark-Delay wird bei **set_scanner_delays** mit dem Parameter `Mark` angegeben.

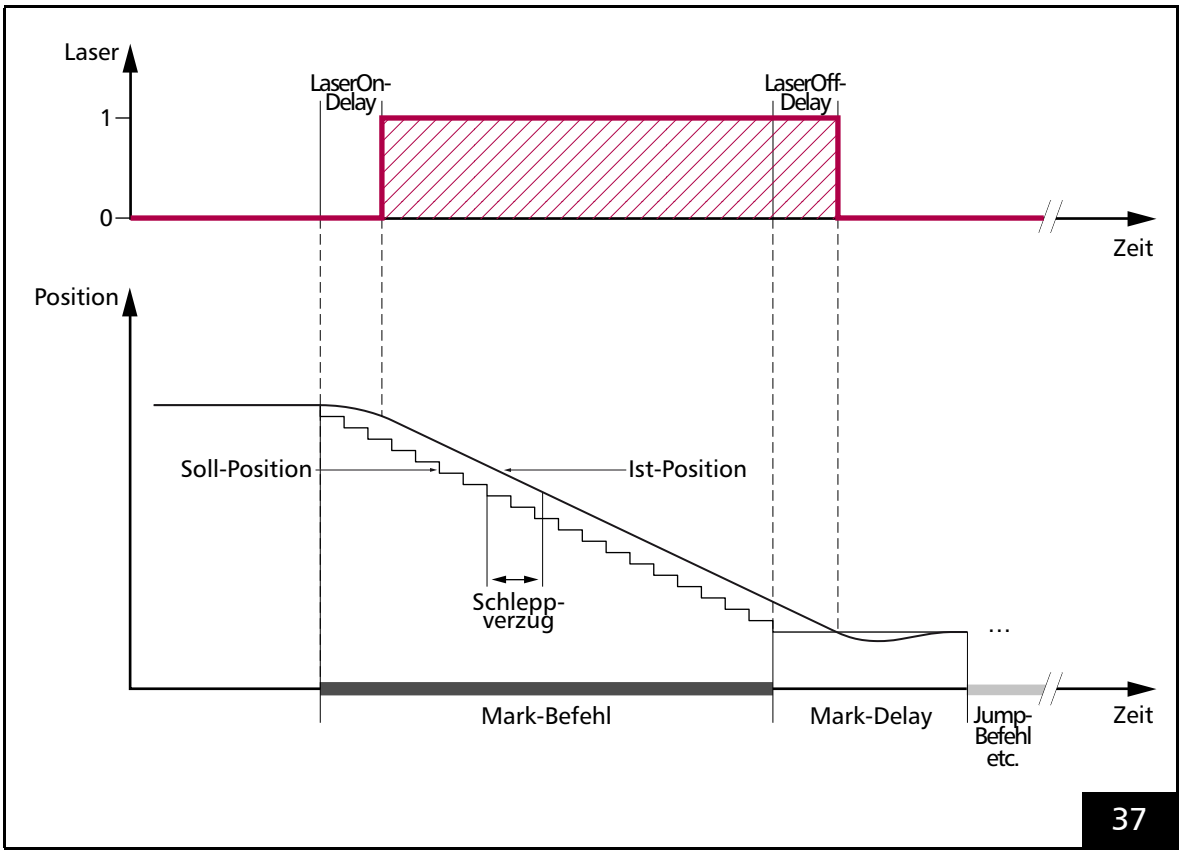
Einen typischen Verlauf für einen einzelnen Markier-Befehl mit den zugehörigen Laser-Delays zeigt Abbildung 37.

Die Dauer des Mark-Delays sollte zu den dynamischen Eigenschaften des Scan-Kopfs (Schleppverzug, Tuning) sowie der eingestellten Markiergeschwindigkeit passen.(1)

(1) Über das Zusammenwirken von Laser-Delays und Scanner-Delays im DSP-Modus < 3 (siehe **set_dsp_mode**), siehe Abschnitt "Automatische Delay-Anpassung", Seite 158.

Hinweise

- Wenn nach einem Markier-Befehl kein weiterer Markier-Befehl folgt, wird ein Mark-Delay automatisch eingefügt und der Laser nach einem LaserOff-Delay ausgeschaltet.
- Wenn nach einem Markier-Befehl ein weiterer Markier-Befehl folgt (= "Polygonzug"), wird ein (variables) Polygon-Delay eingefügt und der Laser bleibt eingeschaltet, siehe Abschnitt "Variable Polygon-Delays", Seite 154.
- Kurze Listenbefehle führen nicht zu einem Einfügen eines Mark-Delays oder Polygon-Delays und nicht zu einem Abschalten des Lasers.
- Beachten Sie, dass ein Markier-Befehl der Länge 0 einem **list_continue** entspricht, wenn er nicht zeitbemessen ist, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286.



Scan-Kopf-Steuerung und Laser-Steuerung während eines Markier-Befehls oder Bogen-Befehls mit Mark-Delay. Der Laser ist im schraffierten Zeitabschnitt an.

Polygon-Delay

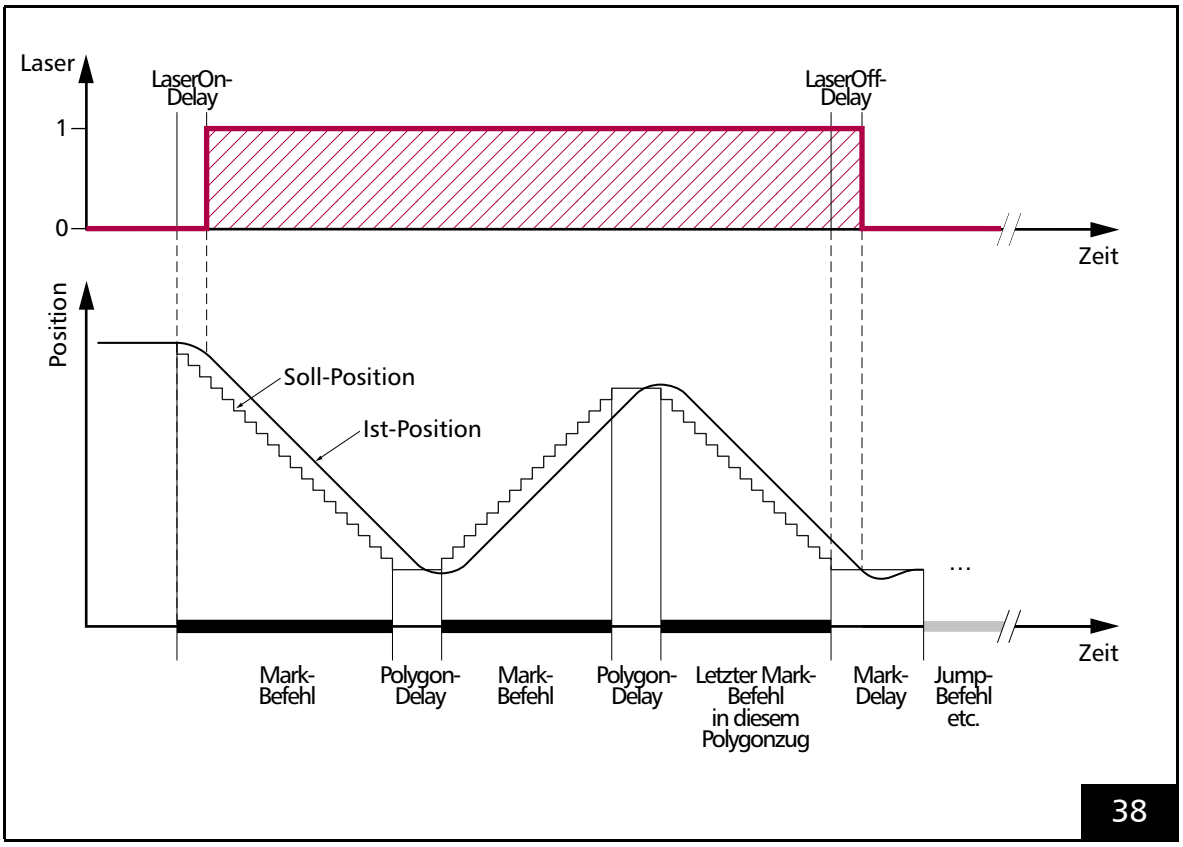
Das Polygon-Delay wird bei **set_scanner_delays** mit dem Parameter `Polygon` angegeben.

Einen typischen Verlauf für eine Abfolge mehrerer Markier-Befehle (=Polygonzug), bei denen zwischen den einzelnen Markier-Befehlen Polygon-Delays (statt Mark-Delays) eingefügt werden, zeigt Abbildung 38.

Kurze Listenbefehle unterbrechen einen Polygonzug nicht.

Wenn die (meist kleineren) Winkel zwischen den Markierungen nur wenig variieren, kann in der Regel ein Polygon-Delay-Wert angegeben werden, der kleiner als der Mark-Delay-Wert ist.

Wenn dagegen die Winkel deutlich variieren, dann wird ein "Variables Polygon-Delay" empfohlen, siehe Abschnitt "Variable Polygon-Delays", Seite 154.



Scan-Kopf-Steuerung und Laser-Steuerung während eines Polygonzugs mit konstantem Polygon-Delay.

Variable Polygon-Delays

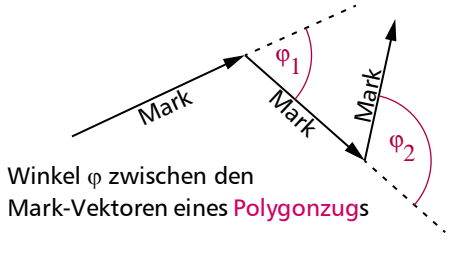
Indem (bei **set_delay_mode** oder **set_delay_mode_list**) der `VarPoly`-Wert > 0 angegeben wird, wird eingeschaltet der:

- "Variable Polygon-Delays"-Modus

Dann berechnet die RTC6 PCIe-Karte für jede Polygonzug-Ecke das "Variable Polygon-Delay" *v_delay*(φ) gemäß:

$$v_delay(\varphi) = scale(\varphi) \times polygon_delay$$

Zur Definition des Winkels φ am Beispiel von Mark-Vektoren siehe Abbildung 39.

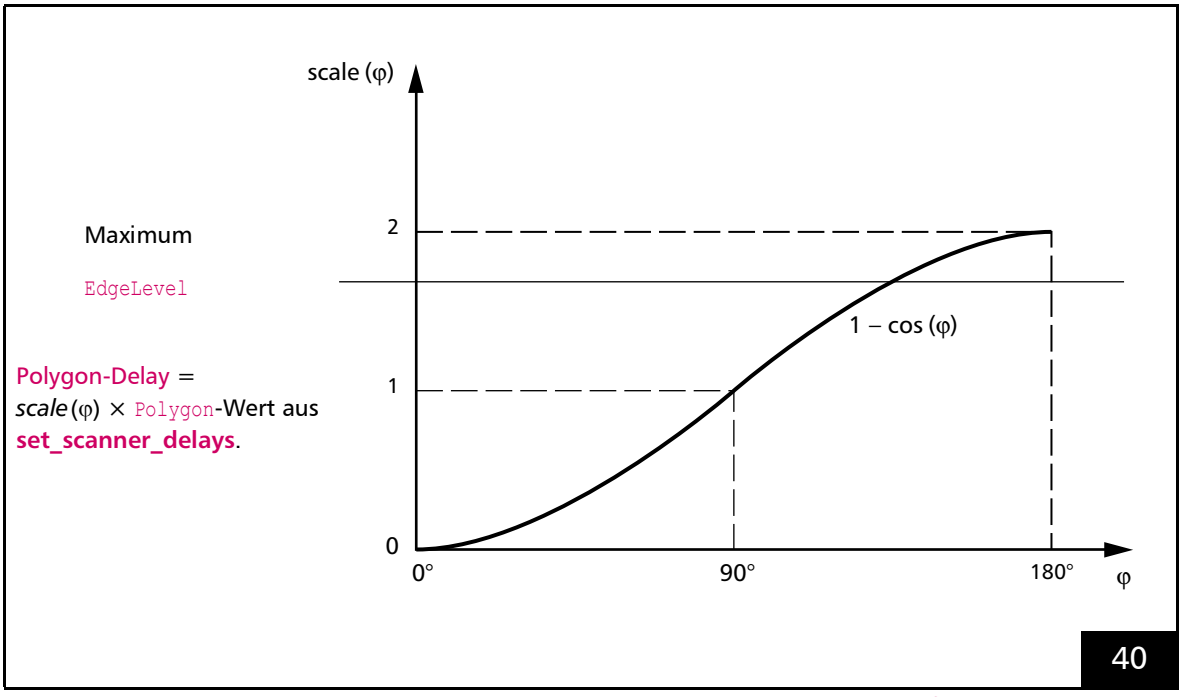


Winkel φ zwischen den Mark-Vektoren eines Polygonzugs

39

Variable Polygon-Delays. Definition des Winkels φ.

Für Kreisbögen und Ellipsen sind sinngemäß die Tangenten am Anschlusspunkt relevant. *scale*(φ) ist eine Skalierungsfunktion für den `Polygon`-Wert aus **set_scanner_delays** (Randbedingung $0 \le scale(\varphi) \le 2$). Die Standard-Funktion nach **load_program_file** ist $scale(\varphi) = 1 - \cos(\varphi)$, siehe Abbildung 40. Sie kann durch eine Benutzer-definierte Funktion ersetzt werden, siehe Abschnitt "Benutzer-definierte "Variable Polygon-Delays"", Seite 156.



40

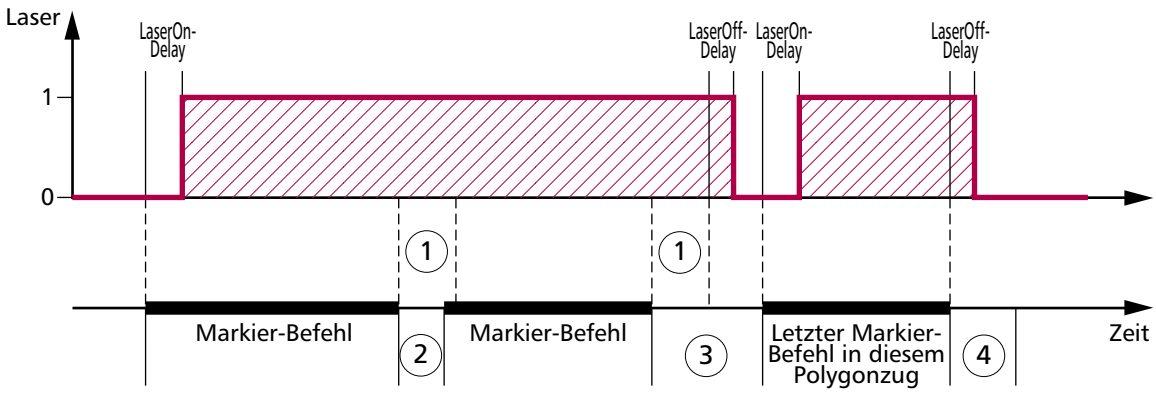
Variable Polygon-Delays. Variation des Polygon-Delays (Standard-Funktion nach **load_program_file**).

EdgeLevel

Das “Variable Polygon-Delay“ wird sehr lang, wenn der Winkel φ nahe bei 180° ist, siehe Abbildung 40. An scharfen Ecken des Polygonzugs führt das unter Umständen zu Einbrenn-Effekten.

Wenn ein EdgeLevel-Wert > 0 (bei **set_delay_mode**) angegeben ist, schaltet die RTC6 PCIe-Karte nach spätestens EdgeLevel Delay-Takten den Laser mit einem LaserOff-Delay aus und beendet damit den aktuellen Polygonzug, siehe Abbildung 41.

Der nächste Markier-Befehl startet wie üblich mit dem Ablauf des aktuell noch anliegenden “Variablen Polygon-Delays”.



Legende

1 EdgeLevel.
2 “Variables Polygon-Delay“, kürzer als der angegebene EdgeLevel-Wert.
3 “Variables Polygon-Delay“, länger als der angegebene EdgeLevel-Wert. Der Laser wird zwischen den beiden Markier-Befehlen ausgeschaltet.
4 Mark-Delay.

41

Timing der Laser-Steuerung während eines Polygonzugs mit “Variablem Polygon-Delay”. Mit **set_delay_mode** wurde ein EdgeLevel-Wert definiert.

Benutzer-definierte “Variable Polygon-Delays”

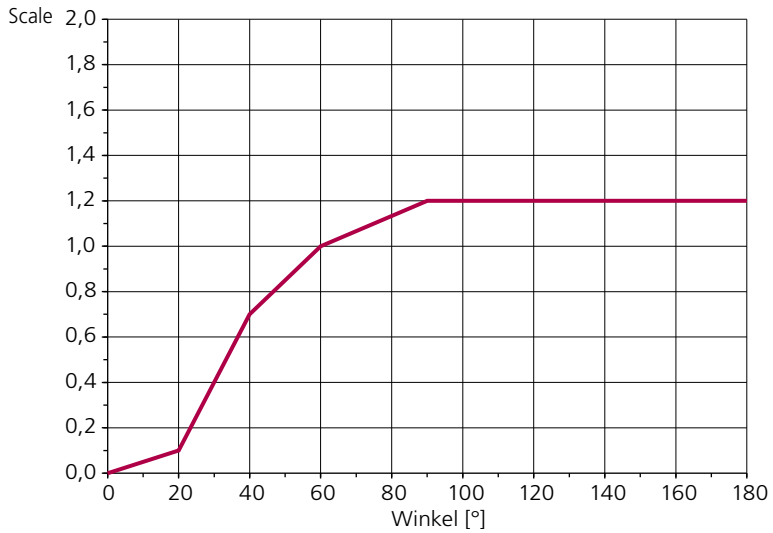
- **load_varpolydelay** lädt für die Skalierungsfunktion *scale*(φ) eine Tabelle aus einer ASCII-Textdatei. Siehe auch Abschnitt “Variable Polygon-Delays”, Seite 154.
- Die ASCII-Textdatei kann ein oder mehrere Tabellen(1) enthalten.
- Jede Tabelle kann bis zu 50 Datenpunkte (φ | *scale*(φ)) enthalten.
- Aus den Datenpunkten wird die Skalierungsfunktion *scale*(φ) linear interpoliert.
- Als Beispiel zeigt Abbildung 42 eine Tabelle mit 4 Datenpunkten und die entsprechende Skalierungsfunktion.

(1) Sogar eines anderen Typs, siehe Tabelle 1, Seite 156.

Tabelle 1: Mögliche Tabellentypen in der ASCII-Textdatei. Jeder Typ kann mehrfach vorkommen.

`[VarPolyTable<No>]`	Benutzer-definierte “Variable Polygon-Delays”, siehe Seite 156
`[PositionCtrlTable<No>]`	Skalierungsfunktion, siehe Seite 210
`[AutoLaserCtrlTable<No>]`	Nichtlinearitätskurve, siehe Seite 215
`[JumpTable<No>]`	Jump-Delay-Werte, siehe Seite 231
`[StretchTable<No>]`	2D-Stretch-Korrekturtabelle, siehe Seite 252
`[Fly2DTable<No>]`	2D-Kompensationstabelle, siehe Seite 262

```
; sample [VarPolyTable1]
Angle1 = 20
Scale1 = 0.1
Angle2 = 40
Scale2 = 0.7
Angle3 = 60
Scale3 = 1.0
Angle4 = 90
Scale4 = 1.2
```



42

Beispiel: Tabelle für Benutzer-definierte “Variable Polygon-Delays” mit 4 Datenpunkten (links) und die entsprechende Skalierungsfunktion *scale*(φ) (rechts).

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Zeile beginnen:
  `[VarPolyTable<No>]`
  `<No>` steht für die Tabellennummer.
- Enthält die Tabelle mehrere `[VarPolyTable<No>]`-Einträge zur gleichen `<No>`, werden nur die Zeilen nach dem ersten Eintrag verwendet. Es werden nur die Zeilen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (φ | *scale*(φ)) wird wie folgt definiert:
  `Angle<n> = <Value>`
  `Scale<n> = <Value>`
  Dabei entspricht `<n>` dem Index des Datenpunktes ($1 \le$ `<n>` ≤ 50). Die Werte `<Value>` für den Winkel φ (in Grad) und der Skalierungsfaktor können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Dezimal-Trennzeichen: Punkt (.).
- Enthält die Tabelle mehrere Datenpunkte mit gleichem Index `<n>`, so wird der zuletzt gelesene verwendet, die vorherigen werden ignoriert.
- Enthält die Tabelle mehrere Datenpunkte zum gleichen Winkel φ, so wird der Datenpunkt mit dem größten Index `<n>` verwendet, die anderen werden ignoriert. Die Gleichheit wird auf $\pm 0.01°$ geprüft.
- Für `<Value>` gelten die folgenden Wertebereiche: $0.0° \le \varphi \le 180.0°$ und $0.0 \le$ *scale*(φ) ≤ 2.0.
- Jede Anweisung muss in einer separaten Zeile stehen.
- Leerzeichen und Tabs in einer Zeile (z.B. zwischen '=' und `<Value>`) werden ignoriert.
- Leere Zeilen werden ignoriert.
- Datenpunkte mit unzulässigen Werten werden ignoriert.
- Datenpunkte, für die nicht `Angle<n>` *und* `Scale<n>` zum selben Index `<n>` definiert sind, werden ignoriert.
- Das Semikolon ';' dient der Kommentareinleitung. Alle Zeichen einer Zeile, die sich hinter einem Semikolon befinden, werden ignoriert.
- Die Reihenfolge der Anweisungen für die Datenpunkte in der Tabelle kann beliebig gewählt werden.
- Die Indices für die Datenpunktpaare in der Tabelle können innerhalb [1…50] beliebig gewählt werden (die Tabelle wird automatisch nach aufsteigenden Winkeln sortiert).
- Enthält die Tabelle *keinen* gültigen Datenpunkt, bleibt **load_varpolydelay** wirkungslos (Rückgabewert 1 oder 13).
- Der Winkel $\varphi = 0°$ bedeutet, dass zwei aufeinanderfolgende Vektoren parallel verlaufen und in die gleiche Richtung markiert werden. Enthält die Tabelle keinen Eintrag für $\varphi = 0°$ (die Gleichheit wird auf $\pm 0.01°$ geprüft), so wird ein Datenpunkt für $\varphi = 0°$ mit dem Skalierungsfaktor *scale*($0°$) = 0 hinzugefügt.
- Der Winkel $\varphi = 180°$ bedeutet, dass zwei aufeinanderfolgende Vektoren in entgegengesetzten Richtungen markiert werden. Enthält die Tabelle keinen Eintrag für $\varphi = 180°$ (die Gleichheit wird auf $\pm 0.01°$ geprüft), so wird ein Datenpunkt für $\varphi = 180°$ mit dem größten in der Tabelle vorhandenen Skalierungsfaktor eines gültigen Datenpunktes für *scale*($180°$) hinzugefügt.

Nach der Initialisierung mit **load_program_file** verwendet die RTC6 PCIe-Karte die interne Standard-Tabelle für das "Variable Polygon-Delay" ($1–\cos(\varphi)$, siehe Abbildung 40). Dies kann alternativ auch erreicht werden mit `Name` = NULL in **load_varpolydelay**.

Die Tabelle kann mit **create_dat_file** gespeichert werden.

7.2.3 Hinweise zur Optimierung der Delays

Die Delays werden mit **set_scanner_delays** und **set_laser_delays** eingestellt.

Sie sollten abgestimmt sein auf:

- die eingestellte Sprunggeschwindigkeit
- die eingestellte Markiergeschwindigkeit
- den Schleppverzug des verwendeten Scan-Kopfs
- die `ControlPreview`-Zeit des verwendeten Vorverarbeitendes Systems, wenn **set_controlpreview_compensation_ctrl** nicht aufgerufen wird

Nicht-optimierte Delays führen unter Umständen zu Markierergebnissen unzureichender Qualität und überhöhten Scan-Zeiten, siehe auch Abschnitt "Mögliche Fehler beim Optimieren der Delays", Seite 160.

Hinweise

- Laser-Delays werden in Einheiten von 1/64 μs angegeben.
- Scanner-Delays (Jump-Delay, Mark-Delay und Polygon-Delay) werden in Einheiten von 10 μs angegeben.

Empfohlene Optimierungsreihenfolge

(1) Laser-Delays.

Es ist empfehlenswert, Jump-Delays und Mark-Delays auf einen hohen Wert zu setzen. Die Laser-Delay-Längen beeinflussen die Gesamt-Scan-Zeit nicht, wenn positive Werte angegeben sind (siehe auch folgenden Abschnitt).

(2) Scanner-Delays.

Automatische Delay-Anpassung

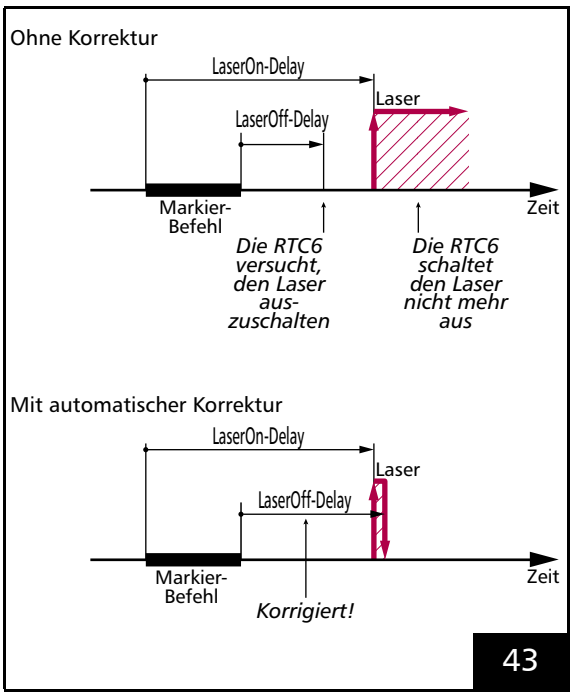
Alle Delays können innerhalb ihrer jeweils zulässigen Wertebereiche im Prinzip beliebig gewählt werden. In manchen Situationen könnte die Laser-Steuerung gestört werden, z.B. durch "Überkreuzen" von Laser-On und Laser-Off. Daher kontrolliert die RTC6 PCIe-Karte bei jedem Befehl, der den Laser schaltet, die eingestellten Laser-Delays, korrigiert diese und fügt gegebenenfalls "künstlich" ein Scanner-Delay ein. Diese Situationen, die im Folgenden aufgeführt sind, sollten eigentlich schon im Anwenderprogramm vermieden werden, weil die "korrigierte" Markierung nicht unbedingt den Anforderungen entspricht.

Überkreuzen von Laser-On und Laser-Off

Die RTC6 PCIe-Karte korrigiert das fehlerhafte Laser-Delay so, dass:

- das Laser-On 1/64 μs nach dem noch aktuell wirksamen LaserOff-Delay erfolgt
- oder das Laser-Off um 1/64 μs nach dem noch laufenden LaserOn-Delay erfolgt

In beiden Fällen wird von der zeitlich nächsten Markierung etwas abgeschnitten, siehe auch Abbildung 43.



Automatische Anpassung des LaserOff-Delays.

Hinweise

- Weitere Hinweise zum generellen Vermeiden solcher automatischer Anpassungen im RTC5-Kompatibilitätsmodus finden Sie im RTC5+Handbuch, Kapitel 7.2.3 "Hinweise zur Optimierung der Delays".

Überkreuzen von Laser-On und Laser-On

Wenn zwischen zwei Markierungen das LaserOn-Delay von einem langen Delay zu einem kurzen Delay umgeschaltet wird, könnte das kurze LaserOn-Delay den Laser schon einschalten, bevor das lange LaserOn-Delay der vorherigen Markierung schon abgelaufen ist, wenn zwischen den Markierungen der Laser ausgeschaltet wurde (anderenfalls würde der Laser an bleiben und das neue LaserOn-Delay wäre an dieser Stelle nicht wirksam).

Um dieses Überschneiden zu vermeiden, fügt die RTC6 PCIe-Karte in diesem Fall (wie die RTC5) “künstlich” ein Scanner-Delay ein. Das erhöht die Bearbeitungszeit und ändert die Dynamik der Galvanometerscanner-Bewegung.

Überkreuzen von Laser-Off und Laser-Off

Wenn das LaserOff-Delay von einem langen Delay zu einem kurzen Delay umgeschaltet wird, könnte das kurze LaserOff-Delay den Laser schon ausschalten, bevor das lange LaserOff-Delay der vorherigen Markierung schon abgelaufen ist, wenn zwischenzeitlich eine Markierung den Laser eingeschaltet hatte (anderenfalls würde der Laser ausbleiben und das neue LaserOff-Delay wäre an dieser Stelle nicht wirksam).

Um dieses Überschneiden zu vermeiden, verlängert die RTC6 PCIe-Karte in diesem Fall (wie die RTC5) das kurze LaserOff-Delay so lange, dass es erst nach dem Vorherigen abläuft (und natürlich auch erst nach dem zwischenzeitlichen LaserOn-Delay).

Mehrere gleichzeitig ablaufende Laser-Delays

Dies kann vorkommen, wenn ein Laser-Delay erst über die Länge des nächsten Befehls hinaus wirksam wird. Die RTC6 PCIe-Karte kann bis zu 256 LaserOn-Delays und LaserOff-Delays gleichzeitig in einer Pipeline verarbeiten, solange Laser-On und Laser-Off abwechselnd und Laser-On—Laser-On sowie Laser-Off—Laser-Off jeweils nacheinander ablaufen (siehe automatische Anpassungen oben). Damit können Markierungen (unterbrochene Linien) bis mindestens 2,56 ms ”vorausprogrammiert“ werden.

Hinweise

- Die RTC5 kann im Gegensatz zur RTC6 PCIe-Karte immer nur ein Delay desselben Typs gleichzeitig verarbeiten. Die Delays werden automatisch angepasst. Gegebenenfalls werden Scanner-Delays “künstlich” eingefügt. Im DSP-Modus < 3 (siehe **set_dsp_mode**) simuliert die RTC6 PCIe-Karte das Verhalten der RTC5, um gleiche zeitliche Abläufe darstellen zu können. Für weitere Details siehe RTC5+Handbuch, Kap. 7.2.3.

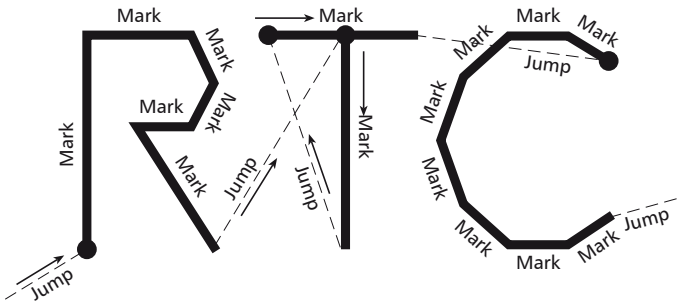
Mögliche Fehler beim Optimieren der Delays

Die folgenden Abbildungen zeigen am Beispiel des Schriftzugs “RTC”, wie sich unpassend eingestellte Delays auf das Markierergebnis auswirken.

LaserOn-Delay zu kurz

Der Laser wird eingeschaltet, bevor die Spiegel zu Beginn eines Mark-Vektors die notwendige Winkelgeschwindigkeit erreicht haben.

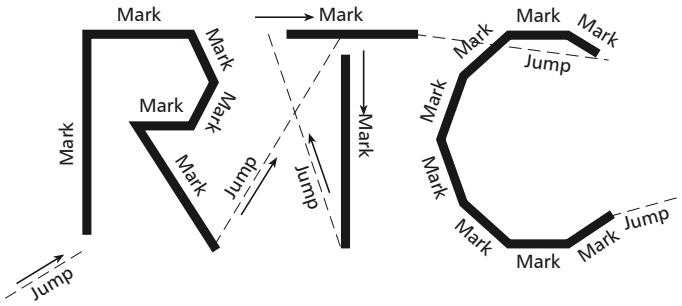
Die Folge sind Einbrenn-Effekte an den Startpunkten der entsprechenden Vektoren.



LaserOn-Delay zu lang

Der Laser wird zu Beginn eines Mark-Vektors zu spät eingeschaltet.

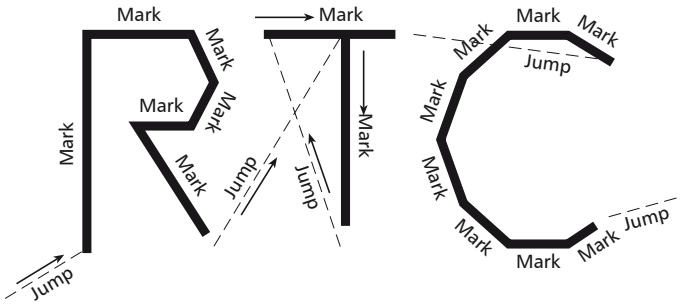
Die entsprechenden Vektoren werden nicht vom Startpunkt an markiert.



LaserOff-Delay zu kurz

Der Laser wird nach einem Mark-Befehl ausgeschaltet, bevor die Spiegel die Vektor-Endposition erreicht haben.

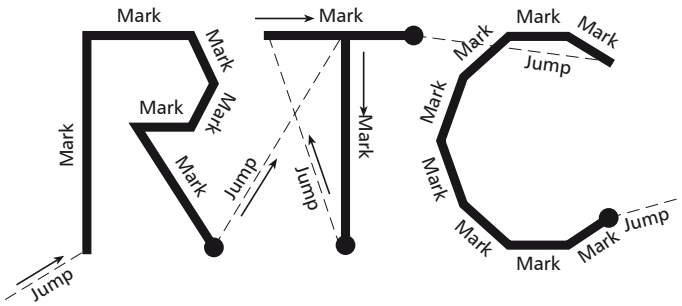
Die entsprechenden Vektoren werden nicht vollständig markiert.



LaserOff-Delay zu lang

Der Laser wird nach einem Mark-Befehl zu spät abgeschaltet. Der Laser ist noch an, obwohl die Spiegel schon deutlich abgebremst werden oder bereits stehen.

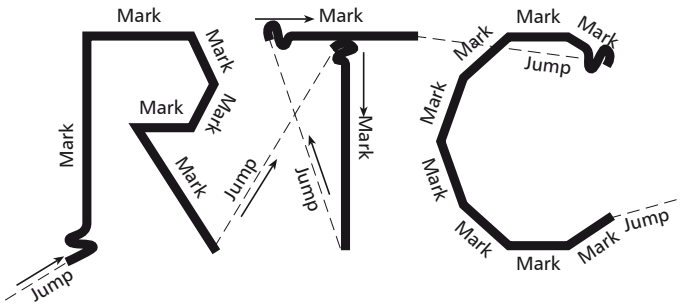
Dies führt zu Einbrenn-Effekten an den jeweiligen Vektor-Endpunkten.



Jump-Delay zu kurz

Der auf einen Sprung-Befehl folgende Mark-Vektor wird schon während der notwendigen Einschwingzeit der Spiegel um den Endpunkt des Sprung-Vektors markiert.

Ein Einlauf oder Überschwinger wird sichtbar.



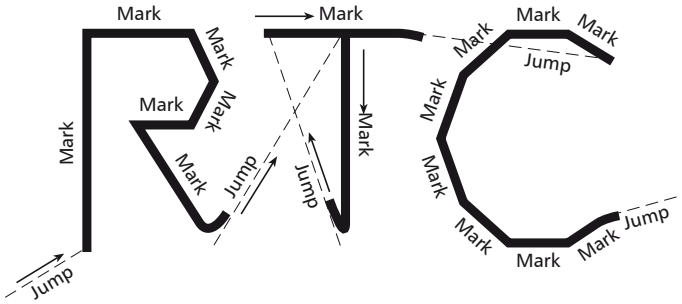
Jump-Delay zu lang

Ein zu lang gewähltes Jump-Delay wirkt sich nicht negativ auf das sichtbare Scan-Ergebnis aus, führt aber zu einer überhöhten Markierungszeit.

Mark-Delay zu kurz

Das Abfahren eines Vektors wird gestartet, bevor die Spiegel die Endposition des vorangehenden Mark-Vektors erreicht haben.

Der Mark-Vektor wird am Ende in Richtung des folgenden Vektors “abgebogen”.



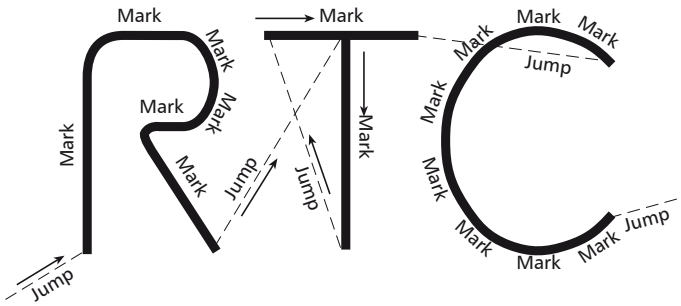
Mark-Delay zu lang

Ist das Mark-Delay zu lang, hat dies keine negativen Auswirkungen auf die Qualität der Markierung. Die Markierungszeit erhöht sich jedoch.

Polygon-Delay zu kurz

Der folgende Mark-Befehl in einem Polygonzug wird schon ausgeführt, bevor die Spiegel die Endposition des vorausgehenden Mark-Vektors erreicht haben.

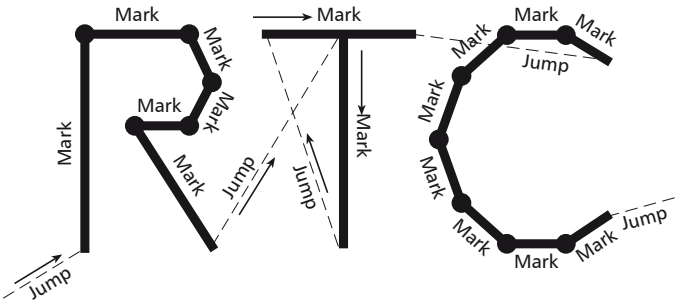
Die Ecken des Polygonzugs werden abgerundet.



Polygon-Delay zu lang

Ist das Polygon-Delay zu lang, so werden die Spiegel am Endpunkt eines Mark-Vektors stark abgebremst oder kommen sogar zum Stehen.

Da der Laser am Ende dieser Vektoren nicht ausgeschaltet wird, kommt es zu Einbrenn-Effekten an den jeweiligen Vektor-Endpunkten.



Im "Variable Polygon-Delays"-Modus kann ein Maximalwert ("`EdgeLevel`") definiert werden, siehe Abschnitt "`EdgeLevel`", Seite 155.

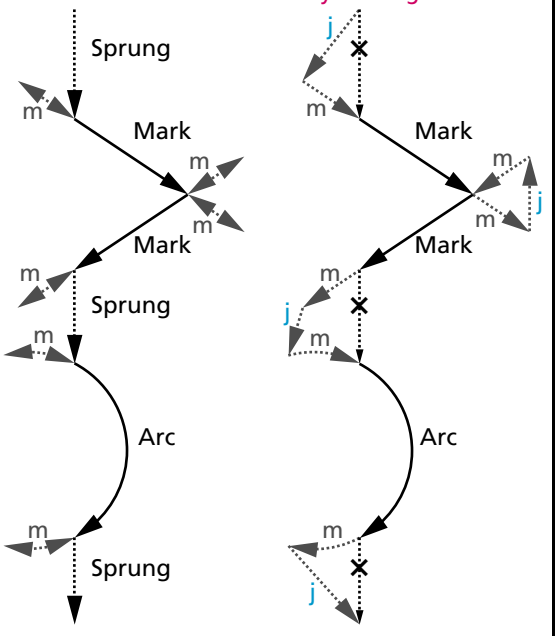
7.2.4 Sky-Writing

Für Anwendungen mit höheren Genauigkeitsanforderungen kann das sogenannte Sky-Writing eingeschaltet werden. Dann wird jeder Mark-Vektor exakt und mit – über die volle Vektorlänge – konstanter Markiergeschwindigkeit ausgeführt.

Bei eingeschaltetem Sky-Writing führt die RTC6 PCIe-Karte bei Markier-Befehlen vor und nach den auszuführenden Vektoren und Bögen automatisch nicht-zu-markierende Sky-Writing-Bewegungen aus, siehe Abbildung 44.

Immer *ohne* Sky-Writing werden ausgeführt:

- micro_vector[*]-Befehle
- [*]para[*]-Befehle



Sky-Writing-Bewegungen (grau). Für Sky-Writing-Modus 3 und Sky-Writing-Modus 4 siehe auch Abbildung 45.

Über die Parameter der Sky-Writing-Befehle können eingestellt werden:

- die Dauer dieser Vorlauf-Bewegungen
- die Dauer dieser Nachlauf-Bewegungen
- die Synchronisation zwischen Scan-Bewegung und Laser-Steuerung

Sky-Writing steht zur Verfügung als:

- Sky-Writing-Modus 1, Seite 163
- Sky-Writing-Modus 2, Seite 164
- Sky-Writing-Modus 3, Seite 165
- Sky-Writing-Modus 4, Seite 166

Sky-Writing-Modus 1

Im Sky-Writing-Modus 1 führt die RTC6 PCIe-Karte bei jedem Markier-Befehl – unabhängig von den vorausgehenden und nachfolgenden Befehlen – die folgenden Sky-Writing-Bewegungen aus:

- In der Vorlauf-Phase wird dem Vektor/Bogen eine Ausholbewegung der Galvanometerscanner mit Markiergeschwindigkeit vorangestellt, siehe Abbildung 44: Die Galvanometerscanner fahren eine kurze Strecke parallel zum Vektor (oder in Verlängerung des Bogens) zunächst in entgegengesetzter Richtung vom Anfangspunkt weg und anschließend wieder bis zum Anfangspunkt zurück.
- Nach Abarbeitung des Vektors/Bogens mit Markiergeschwindigkeit wird diesem in der Nachlauf-Phase eine kurze Abbremsbewegung und Rücklaufbewegung der Galvanometerscanner (mit Markiergeschwindigkeit) angefügt.

Sky-Writing-Modus 1 kann eingeschaltet (ausgeschaltet) werden mit:

- **set_sky_writing**
- **set_sky_writing_list**
- **set_sky_writing_para**
- **set_sky_writing_para_list**

Sky-Writing-Modus 2

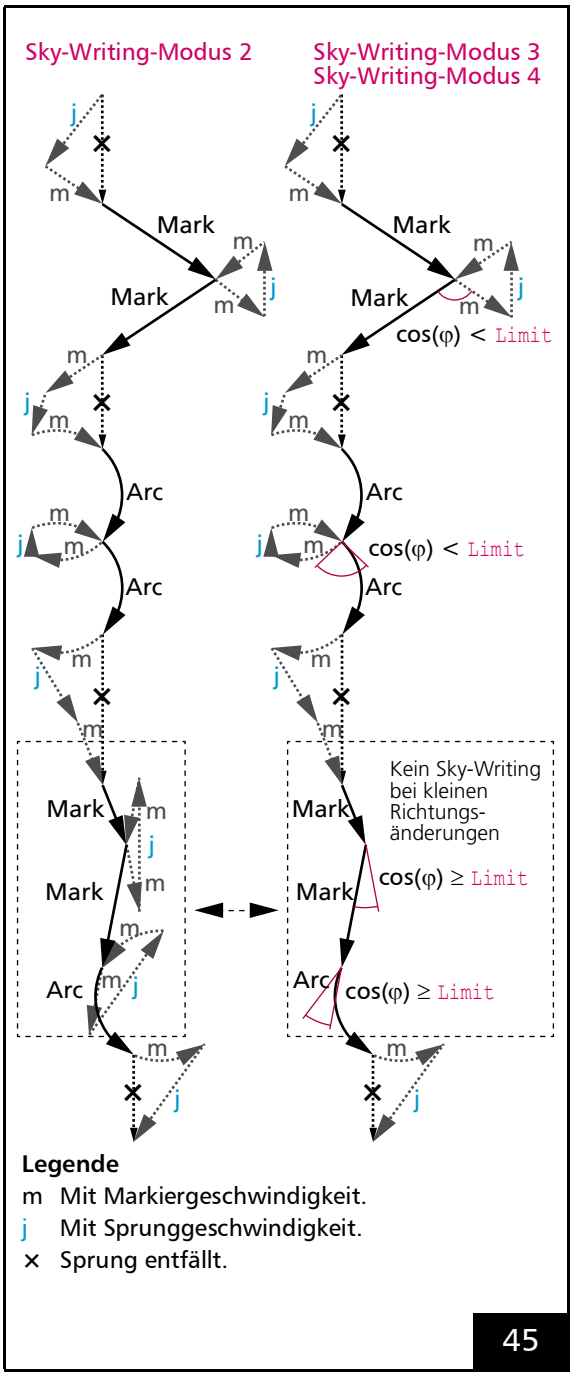
Im Sky-Writing-Modus 2 berechnet die RTC6 PCIe-Karte *zeitverkürzte* Sky-Writing-Bewegungen.

Auch hier wird jedem auszuführenden Vektor/ Bogen eine Vorlauf-Bewegung und Nachlauf-Bewegung in Verlängerung des Vektors/Bogens mit Markiergeschwindigkeit vor- oder nachgestellt. Innerhalb einer Sky-Writing-Modus 2-Markiersequenz erfolgt allerdings weder eine Ausholbewegung der Galvanometerscanner (in der Vorlauf-Phase), noch eine Rücklaufbewegung (in der Nachlauf-Phase). Stattdessen führt die RTC6 PCIe-Karte direkte Sky-Writing-Sprünge (mit der aktuell angegebenen Sprunggeschwindigkeit) von Sprung-Vektor-Startpunkten zu Vorlauf-Startpunkten, von Nachlauf-Endpunkten zu Vorlauf-Startpunkten und von Nachlauf-Endpunkten zu Sprung-Vektor-Endpunkten aus, siehe Abbildung 44.

set_sky_writing und **set_sky_writing_para** schalten immer den Sky-Writing-Modus 1 ein. Zum Sky-Writing-Modus 2 kann daher nur *danach* mit **set_sky_writing_mode** oder **set_sky_writing_mode_list** *um*geschaltet werden.

Ein aktivierter Sky-Writing-Modus 2 wirkt sich auf die gleichen Befehle aus wie Sky-Writing-Modus 1 (außer Ellipse-Befehle, s.u.). *Zeitverkürztes* Sky-Writing erfolgt allerdings nur innerhalb einer Sequenz von nicht-parametrisierten [*]mark[*]-Befehlen, Arc-Befehlen und Sprung-Befehlen (auch zeitbemessene Befehle und 3D-Befehle dürfen enthalten sein). Wird so eine Sequenz durch irgendeinen anderen "nicht-Sky-Writing-Modus 2-fähigen" Listenbefehl unterbrochen (z.B. durch einen [*]para[*]-Befehl, durch einen Ellipse-Befehl oder durch einen beliebigen kurzen Listenbefehl), so unterbricht die RTC6 PCIe-Karte den Sky-Writing-Modus 2 und beendet den vorherigen Befehl im Sky-Writing-Modus 1 (mit einer Rücklaufbewegung der Galvanometerscanner). Sky-Writing-Modus 2 wird durch die Unterbrechung nicht deaktiviert: Nachfolgende "Sky-Writing-Modus 2-fähige" Befehle werden zunächst im Sky-Writing-Modus 1 (mit einer Ausholbewegung der Galvanometerscanner) begonnen und dann wieder im Sky-Writing-Modus 2 ausgeführt.

Ellipse-Befehle werden auch bei eingeschaltetem Sky-Writing-Modus 2 immer im Sky-Writing-Modus 1 ausgeführt.



Sky-Writing-Bewegungen (grau) in Sky-Writing-Modus 2 und Sky-Writing-Modus 3 und Sky-Writing-Modus 4.

Sky-Writing-Modus 3

Für Vektoren und Bögen, die innerhalb eines Polygonzugs mit nur geringen Richtungsänderungen aufeinanderfolgen, ist möglicherweise der Zeitverlust durch die Sky-Writing-Bewegungen unverhältnismäßig hoch gegenüber dem Genauigkeitsgewinn.

Daher kann mit **set_sky_writing_mode** oder **set_sky_writing_mode_list** in den Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 gewechselt werden. Für diese kann mit **set_sky_writing_limit** oder **set_sky_writing_limit_list** ein Umschaltgrenzwinkel angegeben werden, siehe auch Abbildung 45:

- Für größere Winkelabweichungen $\cos(\varphi) < Limit$ zwischen aufeinanderfolgenden Markier-Befehlen eines Polygonzugs wird eine Sky-Writing-Bewegung wie im Sky-Writing-Modus 2 ausgeführt
- Für kleinere Winkelabweichungen $\cos(\varphi) \geq Limit$ wird ein variables Polygon-Delay ausgeführt, siehe Abschnitt “Variable Polygon-Delays“, Seite 154

Hinweise

- Es gelten die Hinweise, Seite 166 zu Sky-Writing-Modus 4.

Sky-Writing-Modus 4

- Voraussetzung: DLL 646, OUT 649
- Wie Sky-Writing-Modus 3
  - Erlaubt zudem Kurze Listenbefehle(1) zwischen Sprung-Befehlen/Mark-Befehlen

Hinweise

- Im Sky-Writing-Modus 4 wird ein Verzögerter Kurzer Listenbefehl wie ein Unverzögerter Kurzer Listenbefehl ausgeführt, also nach dem Nachlauf des vorausgehenden Vektors.
- Wenn mehrere Kurze Listenbefehle(1) zwischen 2 Vektoren stehen: Stellen Sie mit **get_overrun** sicher, dass keine Overruns entstehen.
  Um die Sky-Writing-Bewegungen zu berechnen, sucht die RTC6-Karte nach dem nächsten nicht-Kurzen Listenbefehl im RTC6-Listenspeicher (wenn kein solcher vorhanden ist: endlos).
- Rufen Sie zwischen Sprung-Vektoren und Mark-Vektoren nicht **set_mark_speed** auf. Anderenfalls wäre die berechnete Vorlauf-Position des Sprungs zum nachfolgenden Mark-Vektor falsch, weil sie noch auf der ursprünglichen Markiergeschwindigkeit basiert.
- Wenn ein Polygonzug mit einem kurzen Mark-Vektor endet (kürzer als Markiergeschwindigkeit × `Timelag`), und zuvor ein Polygon-Delay und keine Sky-Writing-Bewegung ausgeführt wurde, kann die Länge des kurzen Vektors (bedingt durch den Schleppverzug) eventuell nicht die bei Sky-Writing erwartete Präzision erreichen.
- in Sky-Writing-Modus 3 und Sky-Writing-Modus 4 erfolgt eine Sky-Writing-Modus 1-Bewegung (siehe Abschnitt "Sky-Writing-Modus 2", Seite 164):
  - Am Anfang eines Polygonzugs
  - Am Ende eines Polygonzugs
  - Bei einer Polygonzug-Unterbrechung aufgrund "nicht-Sky-Writing-Modus 2-fähiger" Listenbefehle

(1) Siehe Kurzer Listenbefehl, Seite 339.

Synchronisation

Abbildung 46 zeigt das Zeitdiagramm der Scan-Kopf-Steuerung und Laser-Steuerung für Sky-Writing-Modus 1.

Das Zeitdiagramm für Sky-Writing-Modus 2, Sky-Writing-Modus 3 und Sky-Writing-Modus 4 ist ähnlich, aber im Vorlauf und Nachlauf gibt es keine gegenläufige Bewegungen der Galvanometerscanner.

Die relevanten Parameter für die Synchronisation sind:

- `Nprev`
- `Npost`
- `Timelag`
- `LaserOnShift`

Diese werden mit **set_sky_writing_para**, **set_sky_writing** und den entsprechenden Listenbefehlen angegeben:

- Der Parameter `Nprev` wird ganzzahlig in 10 μs-Einheiten angegeben und definiert die Vorlauf-Dauer:

Modus	Dauer
Sky-Writing-Modus 1	20 × `Nprev` [μs]
Sky-Writing-Modus 2	10 × `Nprev` [μs]
Sky-Writing-Modus 3	10 × `Nprev` [μs]
Sky-Writing-Modus 4	10 × `Nprev` [μs]

- Der Parameter `Npost` wird ganzzahlig in 10 μs-Einheiten angegeben und definiert die Nachlauf-Dauer:

Modus	Dauer
Sky-Writing-Modus 1	20 × `Npost` [μs]
Sky-Writing-Modus 2	10 × `Npost` [μs]
Sky-Writing-Modus 3	10 × `Npost` [μs]
Sky-Writing-Modus 4	10 × `Npost` [μs]

- Die Parameter `Timelag` (Wert im 64-Bit-IEEE Gleitkommaformat gerundet auf ganzzahlige Vielfache von 1/64 μs) und `LaserOnShift` (ganzzahlig in 1/64 μs-Einheiten) legen die Verzögerung des Einschaltzeitpunkts oder Ausschaltzeitpunkts der Signale für den "Laser active"-Betrieb relativ zur Anfangs-Soll-Position oder End-Soll-Position fest:
  - Verzögerung des Einschaltzeitpunkts relativ zur Anfangs-Soll-Position / μs
    = `Timelag` + 1/64 μs × `LaserOnShift`
  - Verzögerung des Ausschaltzeitpunkts relativ zur End-Soll-Position / μs
    = `Timelag`

Für `LaserOnShift` = 0 werden die Signale für den "Laser active"-Betrieb mit einer Verzögerung von `Timelag` relativ zur Anfangs-Soll-Position (End-Soll-Position) angeschaltet (ausgeschaltet).

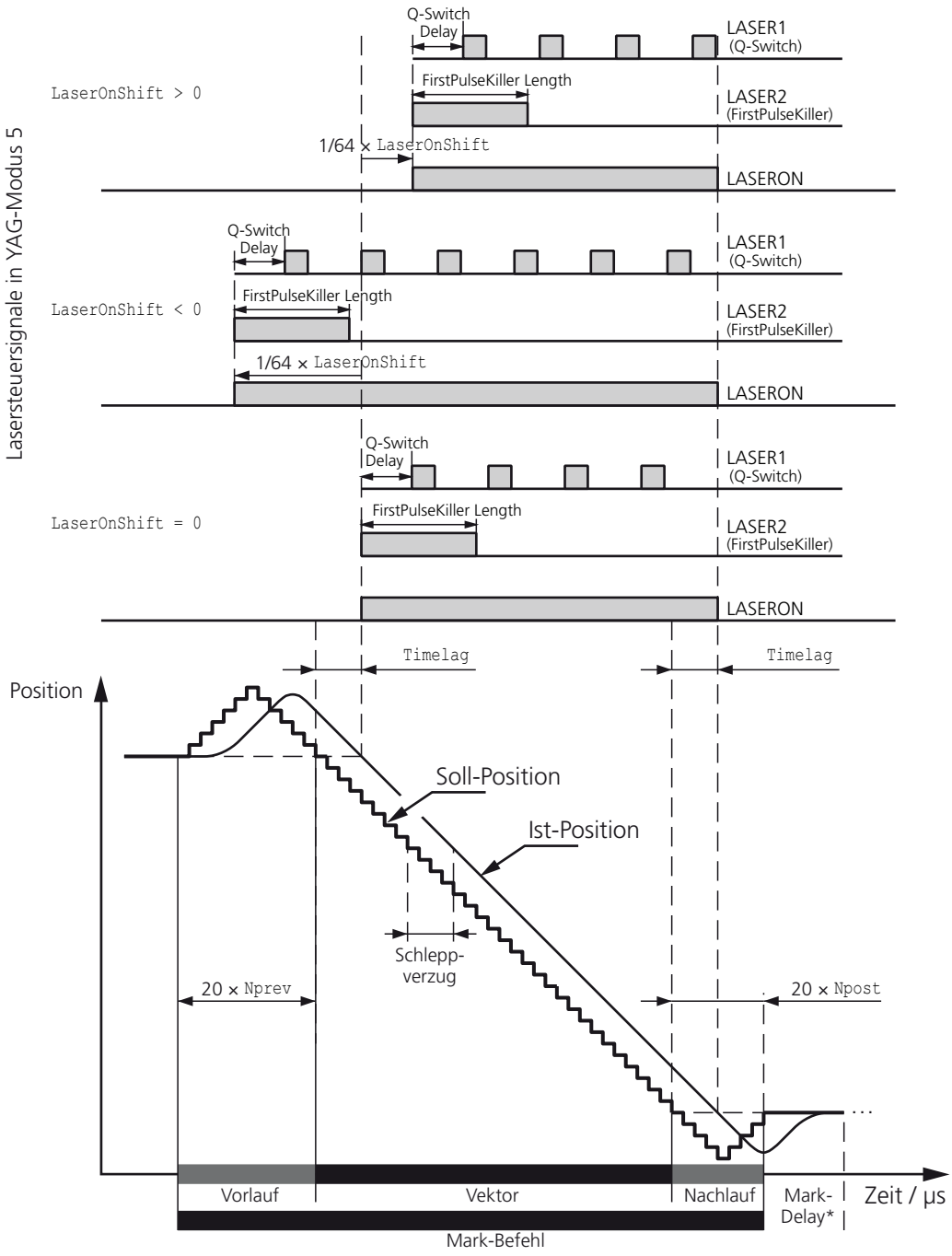
Wird der Parameter `Timelag` auf den tatsächlichen Schleppverzug eingestellt (und die Parameter `Nprev` und `Npost` auf ausreichend große Werte), so wird auf diese Weise sichergestellt, dass:

- Die Signale für den "Laser active"-Betrieb werden exakt an den Endpunkten des gewünschten Vektors eingeschaltet oder ausgeschaltet
- Es stellt sich ein konstanter Schleppverzug ein, noch bevor der Anfangspunkt des Vektors erreicht ist
- Der Vektor wird bis zu seinem Endpunkt mit konstanter Markiergeschwindigkeit abgefahren

Bei Sky-Writing sind die Laser-Delays aus **set_laser_delays** nicht wirksam.

Der Einschaltzeitpunkt der Signale für den "Laser active"-Betrieb kann mit dem Parameter `LaserOnShift` angepasst werden (z.B. an die `HalfPeriod` aus **set_laser_pulses** oder an die Reaktionszeiten des Lasers selbst), so dass der erste Laserpuls exakt mit der Anfangs-Soll-Position erfolgt:

- Positive `LaserOnShift`-Werte verzögern das Einschalten der Lasersteuersignale.
- Negative `LaserOnShift`-Werte verlegen sie vor, frühestens jedoch bis zum Beginn des Vorlaufs.
  Negative `LaserOnShift`-Werte sind auch notwendig, um bei einem YAG-Modus "Platz" für ein Q-Switch-Delay oder FirstPulseKiller-Signal zu schaffen.



Scan-Kopf- und Laser-Steuerung im Sky-Writing-Modus 1 (mit Parametern `Timelag`, `Nprev`, `Npost` und `LaserOnShift`)
(Beispielhaft sind die Lasersteuersignale LASER1 und LASER2 für den YAG-Modus 5 dargestellt)
(* Dieses Mark-Delay ist nur wirksam bei Sky-Writing-Modus 3 und Sky-Writing-Modus 4).

Hinweise

- Mit **set_sky_writing_mode** und **set_sky_writing_mode_list** kann auch zwischen Sky-Writing-Modus 1, Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 hin und her geschaltet werden.
  Auch ein vorübergehendes Ausschalten ist möglich, solange `Timelag` > 0 gesetzt ist.
- Bei der Suche nach geeigneten Sky-Writing-Parametern sollte zuerst **set_sky_writing** und Sky-Writing-Modus 1 verwendet werden: Als Startwert für den Parameter `Timelag` sollte der Schleppverzug der Galvanometerscanner eingestellt werden. Dann kann `Timelag` (mit `LaserOnShift` = 0) so feinjustiert werden, dass das *Ende* von Markier-Vektoren exakt markiert wird und anschließend der Parameter `LaserOnShift` (bei dann konstantem `Timelag`) nachjustiert werden, so dass auch der *Anfang* von Markier-Vektoren exakt markiert wird.
  In einem zweiten Schritt kann **set_sky_writing_para** verwendet werden, um die Parameter `Nprev` und `Npost` zu optimieren: `Nprev` und `Npost` können (gegenüber der Default-Einstellung von **set_sky_writing**, s.u.) verkleinert werden, um die Gesamtmarkierungszeit zu minimieren. `Nprev` kann alternativ auch vergrößert werden, um ausreichend Platz für ein entsprechend großes negatives `LaserOnShift` zu gewährleisten.
- Bei **set_sky_writing** und **set_sky_writing_list** sind `Nprev` und `Npost` vorgegeben:
  – `Nprev` = ca. 0,15 × `Timelag`
  – `Npost` = ca. 0,10 × `Timelag`
  Dies bedeutet für Vorlauf-Dauer und Nachlauf-Dauer:

	Modus	Dauer
Vorlauf	Sky-Writing-Modus 1	3 × `Timelag`
	Sky-Writing-Modus 2	1,5 × `Timelag`
	Sky-Writing-Modus 3	1,5 × `Timelag`
	Sky-Writing-Modus 4	1,5 × `Timelag`
Nachlauf	Sky-Writing-Modus 1	2 × `Timelag`
	Sky-Writing-Modus 2	1 × `Timelag`
	Sky-Writing-Modus 3	1 × `Timelag`
	Sky-Writing-Modus 4	1 × `Timelag`

- Mit **set_sky_writing_para** oder **set_sky_writing_para_list** können auch andere Vorlauf-Phasen und Nachlauf-Phasen vorgegeben werden, z.B. kürzere, um Markierzeit zu sparen (was aber gegebenenfalls auf Kosten der Genauigkeit geht).
  Kürzere Vorlauf-Phasen und Nachlauf-Phasen können insbesondere dann eingestellt werden, wenn unterbrochene Linien in nur einer Richtung markiert und Sprunggeschwindigkeit und Markiergeschwindigkeit gleich groß eingestellt werden.
  Dabei sollte beachtet werden, dass im Sky-Writing-Modus keine automatische Korrektur der Laser-Delays und Scanner-Delays durchgeführt wird:
  – Wenn zu Beginn des nachfolgenden Markier-Befehls (Anfang des Vorlaufs) das LaserOn-Delay des vorhergehenden Markier-Befehls noch nicht abgelaufen ist, geht der Laser für den vorhergehenden Befehl nicht an.
  – Wenn zum Ende des nachfolgenden Markier-Befehls (Anfang des Nachlaufs) das LaserOff-Delay des vorhergehenden Markier-Befehls noch nicht abgelaufen ist, geht der Laser für den vorhergehenden Befehl nicht aus.
  – Im Sky-Writing-Modus 1 sollte eine positive `LaserOnShift`-Zeit (in µs) nicht um mehr als (20 × `Npost` – `Timelag`) größer als die kleinste vorkommende Markierzeit eingestellt werden (im Sky-Writing-Modus 2, Sky-Writing-Modus 3 und Sky-Writing-Modus 4 nicht um mehr als (10 × `Npost` – `Timelag`) größer). Anderenfalls geht der Laser für diese Markierung nicht an.

- Zuvor für die Signale für den “Laser active”-Betrieb definierte Pulslängen und Ausgabeperioden (in einem YAG-Modus auch das Q-Switch-Delay und die Parameter des FirstPulseKiller-Signals) werden im Sky-Writing-Modus unverändert beibehalten. Zuvor definierte LaserOn-, LaserOff-, Mark-, Polygon- oder variable Polygon-Delays werden dagegen im Sky-Writing-Modus *nicht* berücksichtigt, werden aber nach einer Deaktivierung des Sky-Writing-Modus wieder voll wirksam.
  Bei der Deaktivierung des Sky-Writing-Modus wird ein vor der Aktivierung definiertes Mark-Delay eingefügt (siehe Abbildung 46).
- Vorlauf und Nachlauf im Sky-Writing-Modus 1 finden vollständig innerhalb des jeweiligen Markier-Befehls statt. Die Ausführung eines jeden Markier-Befehls verlängert sich dadurch um eine Zeitdauer (in 10 μs) von (2 × `Nprev` + 2 × `Npost`). Im Sky-Writing-Modus 2, Sky-Writing-Modus 3 und Sky-Writing-Modus 4 verlängert sich die Ausführung mindestens um eine Zeitdauer (in 10 μs) von (`Nprev` + `Npost`).
- Jump-Delays werden im Sky-Writing-Modus 1 normal ausgeführt. Der Wert des Jump-Delays (in 10 μs, siehe **set_scanner_delays**) kann allerdings um etwa (2 × `Nprev`), auch bis auf 0, verkürzt werden.
- Jump-Delays werden im Sky-Writing-Modus 2, Sky-Writing-Modus 3 und Sky-Writing-Modus 4 automatisch intern dynamisch um bis zu `Nprev` reduziert.
- Es können auch zeitbemessene [*]mark[*]-Befehle und Bogen-Befehle, siehe Kapitel 8.9 “Zeitbemessene Befehle”, Seite 286, im Sky-Writing-Modus ausgeführt werden. Eine kürzere Markierdauer ist hier allerdings auch mit einer höheren Markiergeschwindigkeit und daher auch mit längeren Strecken und Beschleunigungsphasen in der Vorlauf-Phase und Nachlauf-Phase verbunden. `Nprev` und `Npost` können dafür separat an die vorgegebene Markiergeschwindigkeit angepasst werden.
- Der Sky-Writing-Modus wird nicht berücksichtigt (aber auch nicht deaktiviert):
  – Bei der Ausführung von [*]para[*]-Befehlen, z.B. Vektorkontrollierte Laser-Steuerung ist aktiv
  – Bei der Ausführung von micro_vector[*]-Befehlen, siehe Kapitel 8.8 “micro_vector[*]-Befehle”, Seite 285
- Wenn Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 aktiviert ist, werden die folgenden Funktionalitäten der 2D-Sprung-Befehle **jump_abs** und **jump_rel** *nicht* ausgeführt:
  – Automatische Tuning-Umschaltung im Sprungmodus, siehe Kapitel 8.1.5 “Sprungmodus”, Seite 229
  – Koordinatentransformationen mit `at_once` = 2, siehe Listenpunkt “Mit `at_once` = 2…”, Seite 238

Sky-Writing mit Mindest-Markiergeschwindigkeit

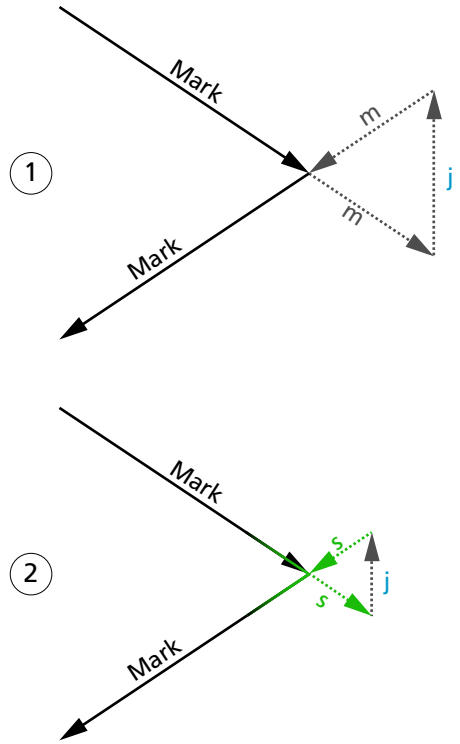
- Voraussetzungen: Nur verfügbar für SCANahead-Systeme (**set_scanahead_params**)
  - mit eingeschalteter automatischen Berechnung der Scanner-Delays und Laser-Delays (**activate_scanahead_autodelays**)
  - zusammen mit Sky-Writing-Modus 2 oder Sky-Writing-Modus 3 oder Sky-Writing-Modus 4

- Keine Voraussetzung, aber Empfehlung: **set_auto_laser_control**(`Ctrl` = 7) ("Spot Distance Control") zum Konstanthalten des Energieeintrags

Um die Prozesszeit bei Sky-Writing-Bewegungen zu verringern (= Durchsatzerhöhung), kann ein Teil der Beschleunigungsphase und Abbremsphase in den Mark-Vektor hinein verlagert werden mit:

- **set_sky_writing_min_speed_ctrl**
- **set_sky_writing_min_speed_list**

Dazu verkürzt die RTC6 PCIe-Karte `Nprev` und `Npost` automatisch und damit die Sky-Writing-Bewegungen. Zu Mark-Vektor-Beginn und Mark-Vektor-Ende wird deshalb nur noch der angegebene `Speed`-Wert (= Mindest-Markiergeschwindigkeit für Sky-Writing) erreicht, statt der vollen Markiergeschwindigkeit, siehe Abbildung 47.



Legende

m Mit Markiergeschwindigkeit.
j Mit Sprunggeschwindigkeit.
s Mit `Speed` (= Mindest-Markiergeschwindigkeit für Sky-Writing).

1. Sky-Writing-Modus 2, Sky-Writing-Modus 3, Sky-Writing-Modus 4 (siehe Abbildung 45).
2. Sky-Writing mit Mindest-Markiergeschwindigkeit.

47

Sky-Writing mit Mindest-Markiergeschwindigkeit.
Beschreibung siehe Text.

Hinweise

- Siehe **set_sky_writing_min_speed_ctrl**, Hinweise, Seite 822.
- Auf *kurzen* Mark-Vektoren kann es vorkommen, dass die volle Markiergeschwindigkeit nie erreicht wird. Die Markierergebnisse sind dann möglicherweise nicht wie erwartet, auch wenn die RTC6 PCIe-Karte die Laserschaltzeitpunkte automatisch anpasst.
  Bei optimalen Markierergebnissen sind kürzeste Mark-Vektorlänge und `Speed`-Wert in einem passenden Verhältnis.
  - Ist der `Speed`-Wert vorgegeben – wählen Sie die kürzeste Mark-Vektorlänge (l_{min}) so, dass die 100%-Markiergeschwindigkeit ($V_{100\%}$) auf dem Mark-Vektor erreicht wird, gemäß:

    $$l_{min} > (V_{100\%}{}^2 - Speed^2) / (A_{max})$$

  - Ist die kürzeste Mark-Vektorlänge (l_{min}) vorgegeben – wählen Sie den `Speed`-Wert so, dass die 100%-Markiergeschwindigkeit ($V_{100\%}$) auf dem Mark-Vektor erreicht wird, gemäß:

    $$Speed > \sqrt{V_{100\%}{}^2 - (A_{max} \times l_{min})}$$

Sky-Writing mit Skalierung der Ausholbewegung

- Voraussetzungen: Nur verfügbar für SCANahead-Systeme (**set_scanahead_params**)
  - mit eingeschalteter automatischen Berechnung der Scanner-Delays und Laser-Delays (**activate_scanahead_autodelays**)
  - zusammen mit Sky-Writing-Modus 2 oder Sky-Writing-Modus 3 oder Sky-Writing-Modus 4

Alternativ zu Sky-Writing mit Mindest-Markiergeschwindigkeit kann die Prozesszeit von Sky-Writing-Bewegungen verringert werden (= Durchsatzerhöhung) mit:

- **set_sky_writing_scale_ctrl**
- **set_sky_writing_scale_list**

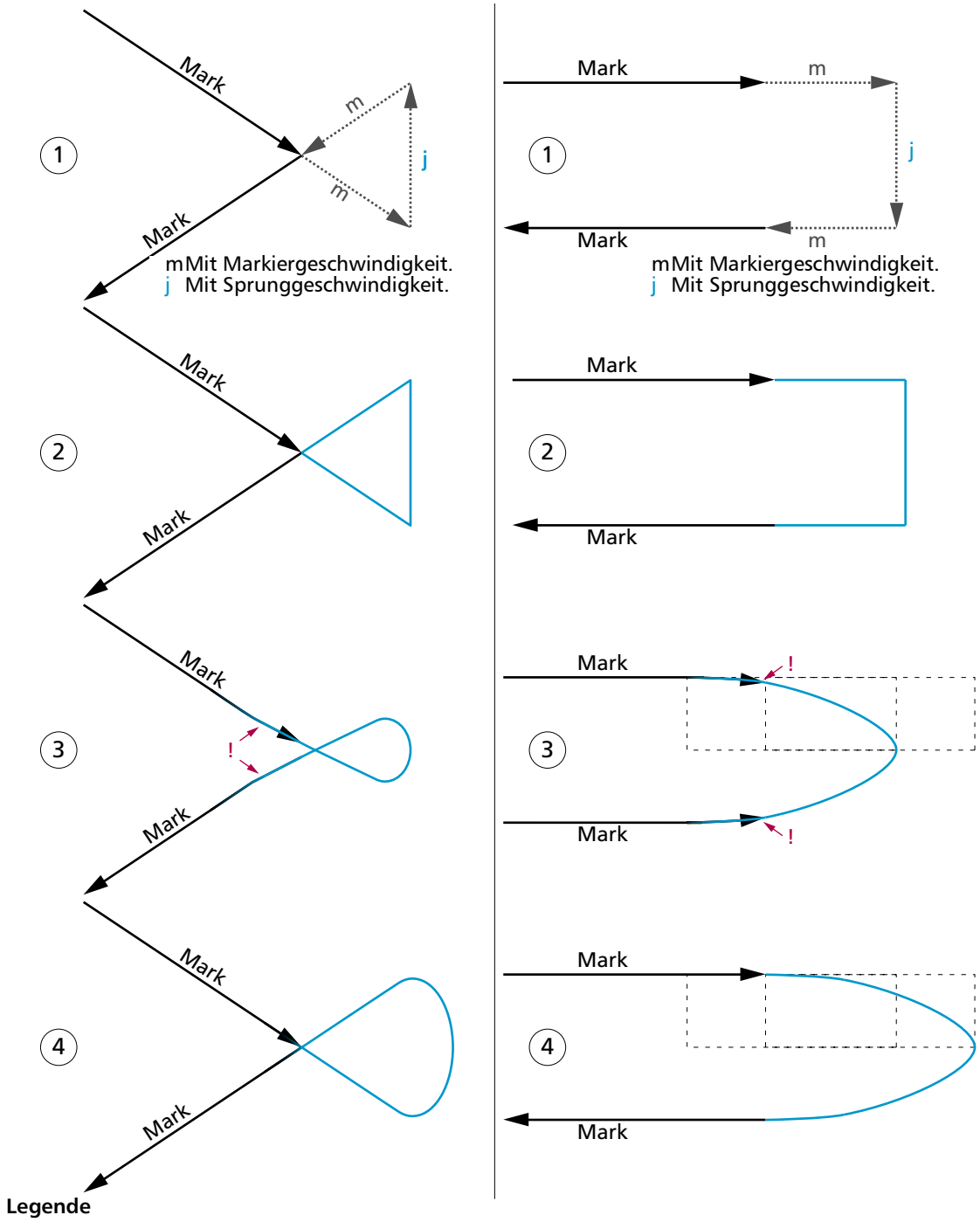
Diese verlängern die Vorlauf-Dauer `Nprev` und Nachlauf-Dauer `Npost` um den Skalierungsfaktor `Scale`. Die Verkürzung der Prozesszeit wird erzielt, indem Sie auch gleichzeitig **set_scanahead_line_params**(`EndScale`) verringern.

Abbildung 48 illustriert das Verhalten von Sky-Writing mit Skalierung der Ausholbewegung an 2 Anwendungsfällen:

- Anwendungsfall (links) “2 aufeinander folgende Mark-Vektoren“
- Anwendungsfall (rechts) “2 parallele Mark-Vektoren“
- (1) zeigt Sky-Writing-Modus 2, Sky-Writing-Modus 3, Sky-Writing-Modus 4 als Vergleichsbasis zum Sky-Writing mit Skalierung der Ausholbewegung in (2), (3), (4)
- In (2) werden alle Ecken wegen `EndScale` = 100% exakt ausgefahren. Ecken und markierte Linien haben die beste Qualität.
- In (3) führt ein verringerter `EndScale` zu einer Eckenabrundung zugunsten einer (im Vergleich zu (2)) kürzeren Prozesszeit. Unerwünschterweise erstreckt sich die Eckenabrundung bereits sichtbar in die Mark-Vektoren hinein (diese Stellen sind mit “!” gekennzeichnet). Im Anwendungsfall der parallelen Mark-Vektoren (rechts) wirkt das Konvergieren der markierten Linien besonders unpassend.
- In (4) verschiebt ein erhöhter Skalierungsfaktor `Scale` den negativen Effekt so, dass er bei den markierten Linien nicht mehr zu sehen ist.

Hinweise

- Die passenden Werte für den Skalierungsfaktor `Scale` und `EndScale` (= Optimum zwischen Prozesszeit und geforderter Genauigkeit) müssen Sie experimentell ermitteln.
- Sky-Writing mit Skalierung der Ausholbewegung benötigt kein “Spot Distance Control” für einen gleichmäßigen Energieeintrag – im Gegensatz zu Sky-Writing mit Mindest-Markiergeschwindigkeit.



Legende

1. Sky-Writing-Modus 2, Sky-Writing-Modus 3, Sky-Writing-Modus 4 (siehe Abbildung 45).
2. Sky-Writing mit Skalierung der Ausholbewegung: Trajektorie bei `EndScale` = 100%, `Scale` = 100%.
3. Sky-Writing mit Skalierung der Ausholbewegung: Trajektorie bei `EndScale` = 050%, `Scale` = 100%.
4. Sky-Writing mit Skalierung der Ausholbewegung: Trajektorie bei `EndScale` = 050%, `Scale` = 140%.

48

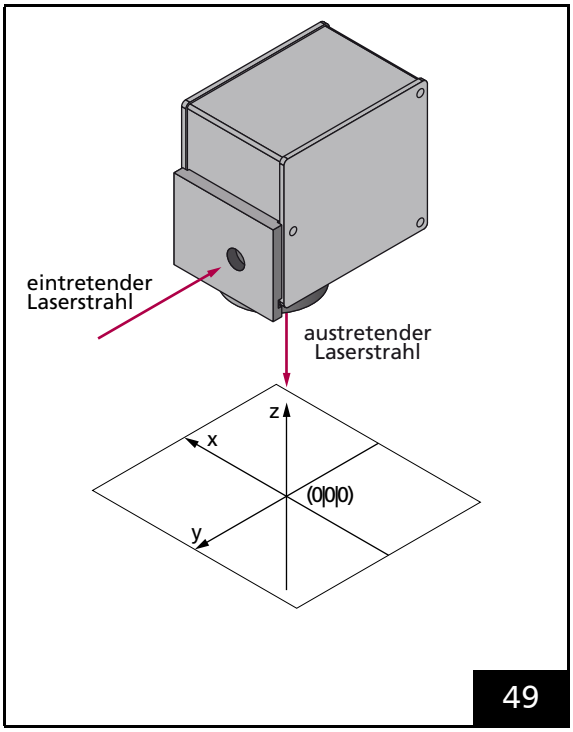
Sky-Writing mit Skalierung der Ausholbewegung. Beschreibung siehe Text.

7.3 Scan-Kopf-Steuerung

7.3.1 Referenzsystem

Das Referenzsystem des Bildfelds, das von der RTC6 PCIe-Karte genutzt wird, zeigt Abbildung 49.

Die y-Achse zeigt *entgegen* der Richtung des eintretenden Laserstrahls und die z-Achse entgegen der Richtung des austretenden Laserstrahls. x-, y- und z-Achse bilden ein rechtshändiges Referenzsystem. Der Ursprung des Referenzsystems, der Punkt (0|0|0), liegt im Zentrum des Bildfelds.



Bezugssystem für die RTC6-Koordinaten.

7.3.2 Bildfeld-Größe und Bildfeld-Kalibrierung

Die Größe des nutzbaren Bildfelds wird im wesentlichen durch den maximalen Scan-Winkel und durch die Brennweite des Objektivs, beziehungsweise durch den Arbeitsabstand (dem Abstand zwischen der Achse des einfallenden Laserstrahls und der Bildfeld-Ebene) bestimmt.

Die x-, y- und z-Koordinaten eines Vektors müssen als 20-Bit-Werte mit Vorzeichen (d. h. als Zahlen zwischen –524.288 und +524.287) angegeben werden.

Der Kalibrierfaktor *K* definiert das Verhältnis zwischen den digitalen Punktkoordinaten in Bits(1) und der tatsächlichen Position des Punktes in *Millimetern*.

Bezeichnet man die durch den maximalen Scan-Winkel gegebene Seitenlänge des Bildfelds mit a_0, dann errechnet sich ein theoretischer Kalibrierfaktor $K_0 = 2^{20}/a_0$ [Bits pro mm(1)].

In der Praxis setzt SCANLAB als Kalibrierfaktor *K* einen gerundeten Wert fest, der oberhalb, aber möglichst nahe beim theoretischen Wert liegt. Der von SCANLAB vorgegebene Kalibrierfaktor kann mit **get_table_para** oder **get_head_para** aus einer verwendeten Korrekturtabelle ausgelesen werden.

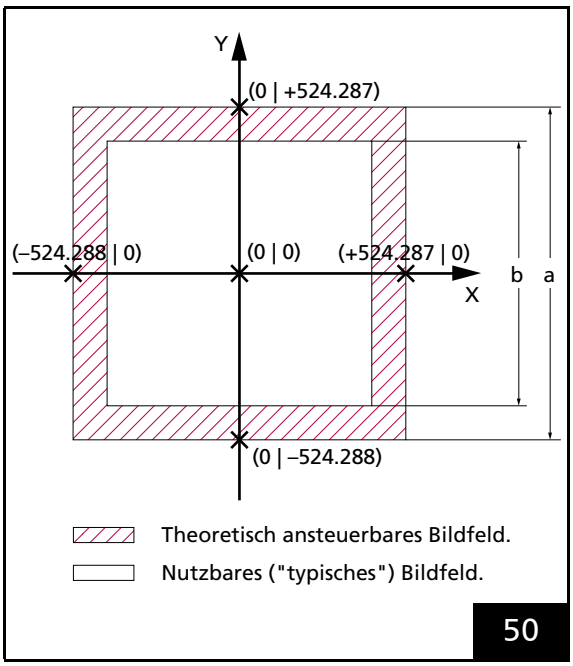
Aus dem Kalibrierfaktor K lässt sich die Seitenlänge *a* des tatsächlich erreichbaren Bildfelds in Millimetern errechnen:

$$a = \frac{2^{20}}{K}$$

(1) Die Bezeichnung “Bits” ist hier als Synonym für “digitale Ansteuerwerte” zu verstehen (siehe Fußnote (2) auf Seite 139).

Typisches Bildfeld

Im Allgemeinen ist das tatsächlich nutzbare (oder "typische") Bildfeld – abhängig vom jeweilig verwendeten Objektiv und der optischen Konfiguration des Scan-Systems – kleiner als das theoretisch ansteuerbar Bildfeld.



Bildfeld-Größe.

Werden Punkte angesteuert, die außerhalb eines für den jeweiligen Scan-Kopf nutzbaren Bildfelds liegen, so können Vignettierungen des Laserstrahls auftreten. Das Innere des Scan-Kopfs kann durch Absorption von Laserleistung beschädigt werden. Beachten Sie hierzu auch die entsprechenden Abschnitte in der Bedienungsanleitung Ihres Scan-Kopfs.

- Vergleichen Sie die errechnete Seitenlänge *a* des theoretisch ansteuerbaren Bildfelds mit der Seitenlänge *b* des nutzbaren Bildfelds, siehe Abbildung 50. Das nutzbare Bildfeld ist in der Bedienungsanleitung zum Scan-Kopf im Kapitel "Technische Spezifikationen" angegeben.
- Damit nur Punkte innerhalb des nutzbaren Bildfelds angesteuert werden, dürfen die Beträge der x- und y-Koordinaten (in Bits) einen Maximalwert *M* nicht überschreiten. *M* ist das Produkt aus dem Kalibrierfaktor *K* und der *halben* Seitenlänge des nutzbaren Bildfelds:

$$M = K \times b/2$$

Kompatibilitätsmodi

RTC6-Standardmodus

(Standard nach **load_program_file** und **set_rtc6_mode**)
Die Bildfeld-Koordinaten für x-Achse, y-Achse und z-Achse und alle diesbezüglichen Parameter (z.B. Sprunggeschwindigkeit oder Wobbelamplitude) werden als 20-Bit-Werte mit Vorzeichen angegeben.

RTC5-Kompatibilitätsmodus

(**set_rtc5_mode**)
Die Bildfeld-Koordinaten für x-Achse und y-Achse und alle damit verbundenen Parameter werden – wie im RTC6-Standardmodus – als 20-Bit-Werte mit Vorzeichen angegeben. Siehe auch Kapitel 2.12.2 "RTC5-Quellcode für die RTC6 PCIe-Karte anpassen", Seite 62.

Die Bildfeld-Koordinaten der z-Achse werden als 16-Bit-Werte mit Vorzeichen angegeben. Die RTC6 PCIe-Karte multipliziert alle z-Koordinaten automatisch mit 16(1).
Als z-Kalibrierfaktor muss ein 16-fach kleinerer Wert verwendet werden.

RTC4-Kompatibilitätsmodus

(**set_rtc4_mode**)
Die Bildfeld-Koordinaten für x-Achse, y-Achse und z-Achse und alle diesbezüglichen Parameter werden als 16-Bit-Werte mit Vorzeichen angegeben. Die RTC6 PCIe-Karte multipliziert sie automatisch mit 16(1). Als xy- und z-Kalibrierfaktor muss ein 16-fach kleinerer Wert verwendet werden. Siehe auch Kapitel 2.10.2 "RTC4-Quellcodes für die RTC6 PCIe-Karte überarbeiten", Seite 50.

(1) Entsprechend verringert sich der zulässige Wertebereich.

7.3.3 Virtuelles Bildfeld

Insbesondere für "Processing on the fly"-Anwendungen steht ein virtuelles Bildfeld mit einer Größe von 29-Bit(1) zur Verfügung.

Daher können Vektor-Befehle und Bogen-Befehle für Objekte geladen werden, die in "Processing on the fly"-Richtung bis zu 512 mal größer als das reale Bildfeld sind. Zu Details zu "Processing on the fly" im virtuellen Bildfeld siehe Kapitel 8.6.6 "Virtuelles Bildfeld mit "Processing on the fly"", Seite 264.

Zur Laufzeit werden die aktuellen Koordinaten auf die reale Bildfeld-Größe [–524.288…+524.287] geclippt, siehe #2…#6 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.

Der erweiterte Wertebereich des virtuellen Bildfelds kann aber auch dazu verwendet werden, um bei Koordinatentransformationen (wie Verdrehungen, Verkleinerungen oder Verschiebungen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236) das volle reale 20-Bit-Bildfeld auszunutzen. Anderenfalls können bei solchen Koordinatentransformationen gewisse Randbereiche des realen Bildfelds unerreichbar bleiben.

Koordinatentransformationen im virtuellen Bildfeld

"Virtuelle Koordinatentransformationen" (insbesondere Translationen und Rotationen) sind u.U. bei zusammenhängenden Markierungen, die größer als das reale Bildfeld sind, erforderlich, um eine gewisse mechanische Toleranz bei der Objektpositionierung auszugleichen.

Siehe #3 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.

"Virtuelle Koordinatentransformationen" werden definiert mit:

- `set_matrix`(`HeadNo` = 4)
- `set_offset`(`HeadNo` = 4)
- `set_offset_xyz`(`HeadNo` = 4)
- `set_angle`(`HeadNo` = 4)

Damit können Matrixkoeffizienten und 2D-Verschiebungen festgelegt werden (bei **set_offset_xyz** wird `ZOffset` ignoriert). Mit `at_once` = 1 werden sie sofort wirksam, wenn gerade keine Liste abgearbeitet wird. Anderenfalls werden sie wie auch mit `at_once` = 0 nur gespeichert.

Gespeicherte Transformationen werden automatisch aktiviert, sobald eine "Processing on the fly"-Session neu gestartet wird (die geänderte Ausgabeposition wird mit Sprunggeschwindigkeit erreicht). Sie bleiben auch nach dem Beenden dieser "Processing on the fly"-Session aktiv(2).

Solange eine "Processing on the fly"-Session aktiv ist, können die Parameter für die "virtuellen Koordinatentransformation" nicht verändert, sondern nur gespeichert werden.

Wieder deaktiviert werden kann die "virtuelle Koordinatentransformation" mit **set_matrix**(4, 0.0, 0.0, 0.0, 0.0), siehe Abschnitt "Takt-Überläufe", Seite 190.

Der Stellbereich für die Verschiebungen ist ±28 Bits. Die Matrixkoeffizienten dürfen einen Absolutwert von 2,0 nicht überschreiten. Die Verschiebung wird nach der Matrixoperation ausgeführt.

Für "virtuelle Koordinatentransformationen" können *nicht* verwendet werden:

- **set_scale**
- alle Listenbefehle für Koordinatentransformationen

"Virtuelle Koordinatentransformationen" können mit RTC6-Software-Paket ≥ V1.6.1 auch über eine "Globale Online-Positionierung", siehe Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243 definiert werden.

(1) Bis DLL 608: 24-Bit.

(2) Mit RTC6-Software-Paket < V1.5.0 sind die "virtuellen Koordinatentransformationen" nur während einer **set_fly_2d**-"Processing on the fly"-Session verfügbar.

7.3.4 3D-Bildfeld

Justierung vornehmen

SCANLAB 3D-Korrekturtabellen sind so berechnet, dass die Ebene der mittleren Fokuslage mit z = 0 angesteuert wird.

Daher muss der mechanische Abstand zwischen Scan-System und Arbeitsebene passend eingestellt sein. Bei der vario*SCAN*-Baureihe muss zusätzlich ein bestimmter Abstand zum Scan-System eingehalten werden. Die Werte sind dem Handbuch des jeweiligen 3-Achsen-Scan-Systems/vario*SCAN*s zu entnehmen.

Abweichungen hiervon sind vorzugsweise über das Anwenderprogramm mit **set_defocus**, **set_defocus_offset**, **set_offset_xyz** oder die entsprechenden Listenbefehle einzustellen.

Sind Arbeitsabstand und Abstand zum Scan-System korrekt eingestellt, sollte anschließend die Laserfokuslage feinjustiert werden. Dazu ist ein Testmuster in der Mitte der z = 0 Arbeitsebene zu markieren. Die für das Bearbeitungsergebnis optimale Laserfokuslage wird erreicht:

- Bei der vario*SCAN*-Baureihe durch manuelles Verdrehen des Fokussierrings, siehe entsprechendes Handbuch
- Beim vario*SCAN FLEX* durch Einstellen der Fokussieroptik-Position, siehe Handbuch zur Erweiterungskarte “RTC5/6 varioSCAN 40 FLEX Extension” (#0128683)
- Beim vario*SCAN* FC und intelli*WELD* FC durch Anpassen des Koeffizienten *A* der verwendeten 3D-Korrekturtabelle(1)

z-Achsen-Kalibrierung überprüfen

Optimale Ausgabewerte für die z-Achse hängen auch von Faktoren wie der Strahlendivergenz des verwendeten Lasers und den Toleranzen der optischen Komponenten ab.

Solche Informationen stehen SCANLAB in der Regel nicht zur Verfügung und fließen deswegen nicht in die Berechnung der 3D-Korrekturtabellen ein.

Daher passt die berechnete 3D-Korrekturtabelle in manchen Fällen nicht optimal zu einem individuellen Scan-System. Um dies zu überprüfen, führen Sie einen Lasermarkierungstest aus, der das ganze 3D-Bildfeld überdeckt.

Überprüfen Sie, ob der Laserfokus den Anforderungen Ihrer Anwendung entspricht. Wenn Sie eine wesentliche Variation des Spotdurchmessers feststellen, kann unter Umständen eine Neukalibrierung der z-Achsen-Korrektur helfen.

Ziel der z-Achsen-Kalibrierung ist die Bestimmung geeigneter Koeffizienten *A*, *B* und *C*. Diese können mit **load_z_table** auf die RTC6 PCIe-Karte übertragen werden.

A, *B* und *C* sind Koeffizienten der parabolischen Funktion

$$z_{out} = A + Bl + Cl^2$$

(Fokuslängenwert *l* im RTC4-Kompatibilitätsbereich [–32.768…+32.767], siehe **load_z_table_no**; Siehe auch Hinweise, Seite 179 unten).

Sie legen in 3D-Korrekturtabellen den Zusammenhang zwischen dem z-Ausgabewert z_{out} und dem Fokuslängenwert l fest. Der Fokuslängenwert l entspricht für jeden Punkt (x|y|z) im 3D-Bildfeld dem Fokuslängenunterschied zwischen dem angegebenen Punkt (x|y|z) und dem Punkt (0|0|0).

Die ABC-Koeffizientenwerte einer Korrekturdatei auf dem PC können direkt mit **read_abc_from_file** ausgelesen und mit **write_abc_to_file** hineingeschrieben werden, siehe nachfolgende Hinweise.

(1) Die Koeffizienten *A*, *B* und *C* können mit **get_head_para** ausgelesen und mit **load_z_table** neu eingestellt werden. Nur *A* sollte angepasst werden. *B* und *C* sollten unverändert – wie ausgelesen – mit **load_z_table** übergeben werden.

Hinweise

- Ab RTC6-Software-Paket V1.12.0 können Sie den Fokuslängenwert *l* im RTC6-20-Bit-Bereich [–524.288…+524.287] belassen, wenn Sie **load_z_table_20b** / **load_z_table_no_20b** sowie **read_abc_from_file_20b** / **write_abc_to_file_20b** verwenden.
- ABC-Werte aus der `*_ReadMe.txt`-Datei zur Korrekturdatei sind als 16-Bit-Werte zu verstehen.

Vorgehen

- Siehe auch Handbuch "Kalibrieren eines 3-Achsen-Laser-Scan-System".
- Siehe auch Hinweise, Seite 179.

(1) Stellen Sie den korrekten mechanischen Abstand zwischen Scan-System und z = 0 Arbeitsebene sowie den korrekten Abstand zwischen vario*SCAN* und Scan-System ein(1).

(2) Laden Sie die 3D-Korrekturdatei mit **load_correction_file**.

(3) Weisen Sie die 3D-Korrekturtabelle mit **select_cor_table** zu.

(4) Lesen Sie die zugewiesenen Koeffizienten *A*, *B* und *C* aus mit **get_head_para**.

(5) Nur vario*SCAN* FC und intelli*WELD* FC: Fahren Sie mit Schritt 10 fort. Schritt 6…9 müssen nicht ausgeführt werden.

(6) Richten Sie den Laser-Spot auf den Referenzpunkt(2) mit **goto_xyz**.

(7) Setzen Sie die z-Achse in Neutralstellung (Mittelstellung) mit **load_z_table_20b**(0, 0, 0).

(8) Platzieren Sie am Referenzpunkt ein Testobjekt.

(9) Justieren Sie den Laserfokus, siehe Seite 178.

(10) Richten Sie mit **goto_xyz** den Laserfokus auf einen beliebigen Punkt (x|y|z) im (benötigten) 3D-Bildfeld(3).

(1) Siehe entsprechendes Handbuch.

(2) Der Referenzpunkt ist ein Punkt in der z = 0 Arbeitsebene, für den ein mittlerer Fokuslängenwert zur Scharfstellung des Laserfokus benötigt wird. Der Laserstrahl sollte auf den Referenzpunkt fokussiert werden, wenn sich die z-Achse in Neutralstellung befindet (z-Ausgabewert z_{out} = 0). Die Koordinatenwerte des Referenzpunkts (in mm) sind in der der 3D-Korrekturdatei beigefügten `*_ReadMe.txt`-Datei oder im Handbuch des jeweiligen 3-Achsen-Scan-Systems oder vario*SCAN*s aufgelistet. Bei Verwendung eines F-Theta-Objektivs ist der Referenzpunkt in der Regel gleich dem Ursprung (0|0|0).

(11) Lesen Sie den von der RTC6 PCIe-Karte für diesen Punkt eingestellten Fokuslängenwert l (im RTC6-20-Bit-Bereich [–524.288…+524.287]) mit **get_z_distance** aus(4).

(12) Optimieren Sie für diesen Punkt den Laserfokus: Variieren Sie mit **load_z_table_20b**(A=z_{out}, 0, 0) den z-Ausgabewert z_{out} so, dass der Laserfokus die gewünschte Qualität erreicht(5). Ein Startwert kann jeweils gemäß $A + Bl + Cl^2$. aus dem zuvor ausgelesenen Fokuslängenwert *l* und den zuvor ausgelesenen oder verwendeten Werten A, B und C berechnet werden.

(13) Wiederholen Sie Schritt 10…12 für möglichst viele Punkte (x|y|z) (6) und notieren Sie die jeweiligen Wertepaare (l|z_{out}). Decken Sie dabei möglichst das komplette für Ihre Anwendung benötigte 3D-Bildfeld ab.

(14) Fitten Sie die Funktion $z_{out} = A + Bl + Cl^2$ an Ihre Wertepaare (l|z_{out}).

(15) Verwenden Sie die resultierenden Koeffizienten *A*, *B* und *C*, um mit **load_z_table_20b**(*A*, *B*, *C*) die 3D-Korrekturtabelle anzupassen.

Mit **load_z_table_20b** geladene Werte werden durch ein nachfolgendes **load_correction_file** überschrieben (**load_correction_file** setzt die drei Koeffizienten *A*, *B* und *C* auf die Werte der geladenen Korrekturtabelle). Nach jedem **load_correction_file** oder **select_cor_table** sollten Sie daher auch **load_z_table_20b**(*A*, *B*, *C*) erneut aufrufen.

Mit **load_z_table_no_20b** können die drei Koeffizienten `A`, `B`, `C`, der Korrekturtabelle `No` zugeordnet werden. Sie werden bei **select_cor_table** mit umgeschaltet.

Alternativ können die ABC-Werte mit **write_abc_to_file_20b** dauerhaft im Kopf der Korrekturdatei gespeichert werden, siehe Parameter 5…7 in Abschnitt "ct5-Korrekturdateikopf", Seite 186.

(3) Bei einem Scan-System mit F-Theta-Objektiv reicht es aus, verschiedene Punkte (0|0|z) auf der z-Koordinatenachse auszuwählen. Das maximal mögliche 3D-Bildfeld ist im jeweiligen Handbuch angegeben.

(4) Der Fokuslängenwert l kann positiv oder negativ sein, siehe Hinweise zu **get_z_distance**.

(5) Der passende z_{out} Ausgabewert kann positiv oder negativ sein.

(6) Beachten Sie, dass eine Vergrößerung der z-Ansteuerwerte zu einer Verringerung des Arbeitsabstandes führt und der Fokus damit Richtung Scan-System verschoben wird. Gegebenenfalls muss das Testobjekt entsprechend nachgeführt werden.

3-Achsen-Scan-Systeme mit F-Theta-Objektiv testen

- Markieren Sie mit der angepassten 3D-Korrekturtabelle und 3D-Vektor-Befehlen, siehe Kapitel "3D-Befehle", Seite 250, zwei identisch große Quadrate. Markieren Sie davon eines mit dem Werkstück in der z-Position $z = z_{min}$, das andere bei $z = z_{max}$.
  Diese zwei Quadrate sollten exakt aufeinander fallen. Ist dies nicht der Fall, so ist die Korrekturdatei nicht optimal auf das Objektiv abgestimmt. Messen Sie dann Länge und Breite (x und y) der beiden Quadrate und geben Sie die Werte an SCANLAB weiter. Sie erhalten dann eine neue 3D-Korrekturdatei.

  Siehe auch Abschnitt "Erweiterte 3D-Korrektur", Seite 252.

7.3.5 Bildfeld-Korrektur und Korrekturtabellen

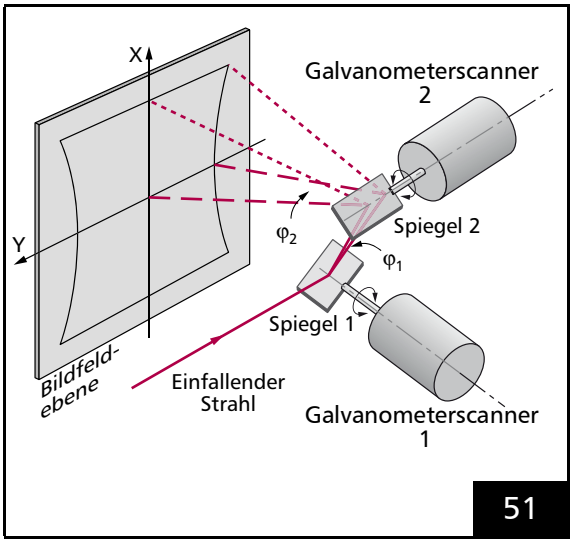
Bildfeld-Verzeichnung

Lenkt man einen Laserstrahl mit einem Zwei-Spiegel-System in einer Ebene ab, so treten die drei folgenden Effekte auf:

(1) Die Anordnung der Spiegel führt zu einer Verzeichnung des Bildfelds[1], die in Abbildung 51 dargestellt ist.

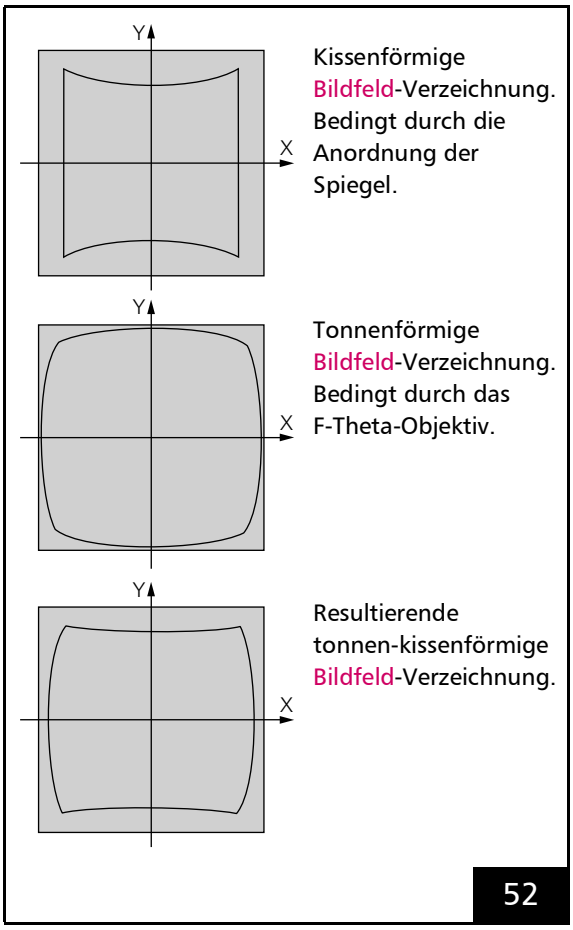
(2) Die in Folge der Spiegelbewegung auf dem Bildfeld abgefahrene Strecke ist nicht direkt proportional zum Scan-Winkel, sondern zum Tangens des Scan-Winkels. Daher ist auch die Scangeschwindigkeit des Laserstrahls auf dem Bildfeld nicht direkt proportional zur Winkelgeschwindigkeit des jeweiligen Galvanometerscanners.

(3) Fokussiert man den Laserstrahl nach der Ablenkung durch beide Spiegel mit Hilfe einer einfachen Linse, so liegt der Strahlfokus auf einer Kugelfläche. In einer planen Bildfeld-Ebene erhält man daher unterschiedliche Spot-Durchmesser.



Bildfeld-Verzeichnung bei der Ablenkung eines Strahls in einem Zwei-Spiegel-System.

Effekt 2 und Effekt 3 können vermieden werden, wenn zur Fokussierung des Strahls ein F-Theta-Objektiv eingesetzt wird. Dieses bewirkt jedoch eine "tonnenförmige" Verzeichnung des Bildfelds, siehe Abbildung 52.



Bildfeld-Verzeichnung, bedingt durch die Anordnung der Spiegel und durch das F-Theta-Objektiv.

(1) Ursache: die Strecke, die der Strahl zwischen Spiegel 1 und Bildfeld zurücklegt, hängt von den Scan-Winkeln von Spiegel 1 und Spiegel 2 ab. Ein größerer Scan-Winkel führt zu einer größeren Entfernung.

Bildfeld-Korrektur-Algorithmus

Zur Korrektur dieser Bildfeld-Verzeichnungen verwendet die RTC6 PCIe-Karte intern einen Algorithmus, der auf einer Korrekturtabelle basiert.

Ein quadratisches Gitter von 257 × 257 Punkten wird dem idealen orthogonalen, quadratischen Bildfeld überlagert. Die x- und y-Koordinaten für eine korrigierte Ausgabe dieser Gitterpunkte werden in der Korrekturtabelle gespeichert.

Soll ein beliebiger Punkt innerhalb des Bildfelds angesteuert werden, berechnet die RTC6 PCIe-Karte die korrigierten Punktkoordinaten durch Interpolation aus den angegebenen Gitterpunkten.

Die Korrektur wird für jeden Mikroschritt einzeln ausgeführt, siehe auch Kapitel 7.1.2 "Zerlegung in Mikroschritte", Seite 143.

SCANLAB erstellt für jeden ausgelieferten Systemtyp eine Korrekturdatei. Dazu wird die Korrekturtabelle aus typischen Systemdaten (wie der Spiegelgeometrie, der Kalibrierung und den Objektivdaten) berechnet. Individuelle Systemeigenheiten und Fehler bei der Justage werden dabei nicht berücksichtigt.

Für Anwendungen, die exakter auf die Eigenschaften des speziellen Systems abgestimmte Korrekturdateien erfordern, bietet SCANLAB Spezialsoftware aus der correXion-Reihe an. Diese generieren RTC-Korrekturdateien aus den Daten von Testmessungen. Weitere Informationen erhalten Sie im zugehörigen Handbuch oder bei SCANLAB.

Bildfeld-Korrektur aktivieren

Zur Aktivierung der Bildfeld-Korrektur müssen die benötigten Korrekturtabellen am Anfang eines Anwenderprogramms:

(1) aus den jeweiligen Korrekturdateien in den Speicher der RTC6 PCIe-Karte geladen werden (mit **load_correction_file**)

(2) den genutzten Scan-Kopf-Anschlüssen zugewiesen werden (mit **select_cor_table** oder **select_cor_table_list**)

Mit 2D-Korrekturtabellen wird eine Bildfeld-Korrektur für die x- und y-Koordinaten aktiviert, mit 3D-Korrekturtabellen zusätzlich auch für die z-Koordinaten.

2D- und 3D-Korrekturdateien

SCANLAB liefert 2D- und 3D-Korrekturdateien aus. Sie tragen die Dateinamenerweiterung `*.ct5` und haben die folgenden Dateinamen:

- `D2_xxx.ct5` für 2D-Korrekturdateien
- `D3_yyy.ct5` für 3D-Korrekturdateien

`xxx` und `yyy` sind dabei durchlaufende Nummern. Jede Korrekturdatei ist für eine bestimmte optische Konfiguration berechnet; für welche Konfiguration, kann der jeweils beigefügten `*_ReadMe.txt`-Datei und den Parametern im Kopf der Korrekturdatei entnommen werden, siehe auch Abschnitt "ct5-Korrekturdateikopf", Seite 186.

Korrekturtabellen laden

Mit **load_correction_file** können bis zu 8 Korrekturtabellen aus ihrer jeweiligen Korrekturdatei in den RTC6 PCIe-Karten-Speicher geladen werden, siehe auch Kapitel 8.5 "2D-Scan-Systeme und 3D Scan-Systeme steuern", Seite 248.

Ist die Option "3D" *nicht* freigeschaltet, können nur 2D-Korrekturtabellen geladen werden.

2D-Korrekturtabellen können auch aus 3D-Korrekturdateien geladen werden (durch **load_correction_file** mit `Dim` = 2). Der 3D-Anteil wird ignoriert.

Ist die Option "3D" freigeschaltet, kann eine 3D-Korrekturtabelle sogar aus einer 2D-Korrekturdatei geladen werden (durch **load_correction_file** mit `Dim` = 3). Die 2D-Korrekturtabelle wird dabei automatisch um eine lineare z-Korrektur ergänzt.

Die tatsächlich passende z-Korrektur kann nach der Zuweisung durch **select_cor_table** (s.u.) mit **load_z_table_no** oder **load_z_table** geladen werden, siehe Kapitel 7.3.4 "3D-Bildfeld", Seite 178.

Hinweise

- RTC5/RTC6-Korrekturdateien (Dateinamenerweiterung `*.ct5`) unterscheiden sich von den Korrekturdateien für die früheren RTC-Produkte (mit Dateinamenerweiterung `*.ctb`) in ihrer Größe, ihrer Struktur und in den enthaltenen Informationen.
- *`.ct5`- und *`.ctb`-Korrekturdateien, die bis auf die Dateinamenerweiterung den gleichen Dateinamen haben, sind für die gleiche optische Konfiguration berechnet.
- Zur Konvertierung älterer Korrekturdateien siehe Abschnitt "Konvertierung von Korrekturdateien", Seite 188.

Geladene Korrekturtabellen zuweisen

Geladene Korrekturtabellen werden den zwei Scan-Kopf-Anschlüssen mit **select_cor_table** oder **select_cor_table_list** zugewiesen.

Wenn weder die Option "Zweite-Scan-Kopf-Steuerung" noch die Option "3D" freigeschaltet ist, kann nur dem Anschluss für ersten Scan-Kopf eine 2D-Korrekturtabelle zugewiesen werden.

Ist die Option "Zweite-Scan-Kopf-Steuerung" freigeschaltet, kann jedem der beiden Scan-Kopf-Anschlüsse eine eigene 2D-Korrekturtabelle zugewiesen werden.

Ist die Option "3D" freigeschaltet und die Option "Zweite-Scan-Kopf-Steuerung" aber nicht, dann kann ausschließlich dem Anschluss für ersten Scan-Kopf eine 2D- oder eine 3D-Korrekturtabelle zugewiesen werden. Der Anschluss für ersten Scan-Kopf gibt Positionssignale für ein xy-Scan-System aus.

Bei einer 3D-Korrekturtabelle gibt der Anschluss für zweiten Scan-Kopf zusätzlich über beide Kanäle Positionssignale für die z-Achse (z.B. vario*SCAN*) aus.

Sind die Option "Zweite-Scan-Kopf-Steuerung" und die Option "3D" freigeschaltet, können bis zu zwei (auch unterschiedliche) 2D-Korrekturtabellen oder eine einzige 3D-Korrekturtabelle zugewiesen werden.

Zur Ansteuerung eines 3D-Systems muss die (einzige) 3D-Korrekturtabelle ausschließlich dem Anschluss für das xy-Scan-System zugewiesen werden. Es gibt keine Zuweisung für die z-Achse. Die Ausgaben für die z-Achse erfolgen automatisch auf beiden Kanälen des jeweils anderen Anschlusses (`select_cor_table( n, 0 )` oder `select_cor_table( 0, n )`).

Hinweise

- Wenn keine Korrekturtabelle in den RTC6 PCIe-Karten-Speicher geladen wurde, dann gibt die Karte unvorhergesehene Ansteuerwerte an das Scan-System aus. Deshalb muss mindestens eine 1to1-Korrekturtabelle (als 2D-Korrekturtabelle oder 3D-Korrekturtabelle) geladen werden, siehe Abschnitt "1to1-Korrekturtabellen", Seite 185, wenn keine speziell für die jeweilige optische Konfiguration berechnete 2D- oder 3D-Korrekturdatei verfügbar ist.
- Korrekturdateien sollten zugewiesen sein(1):
  - bevor erstmalig eine Liste gestartet wird oder
  - bevor mit einem Kontrollbefehl (wie **goto_xy**) die Galvanometerscanner des Scan-Systems bewegt werden
- Ist nur einem Scan-Kopf-Anschluss eine 2D-Korrekturtabelle zugewiesen, dann gibt der andere Scan-Kopf-Anschluss keine *Positions*-Signale aus.
- **load_program_file** weist bei der Initialisierung eine Korrekturtabelle gemäß `select_cor_table( 1, 0 )` zu, lässt die Galvanometerscanner jedoch auf der Position (0,0) stehen. **load_program_file** kann nicht feststellen, ob überhaupt eine Korrekturtabelle #1 geladen ist. Deswegen wird kein interner Sprung auf die in der Korrekturdatei vorgesehene Ausgabeposition durchgeführt. Deren Inhalt könnte zufällig sein (siehe oben).
- **load_program_file** initialisiert die Speicherinhalte der Korrekturtabellen nicht.
  Die Korrekturtabellenwerte sind unmittelbar nach dem Einschalten der RTC6 PCIe-Karte zufällig. Es wird empfohlen, **select_cor_table** explizit aufzurufen. Bevor **select_cor_table** zurückkehrt, wartet er selbst intern das Ende des Sprungs ab.
  - Wenn **load_correction_file** nach **load_program_file** aufgerufen wird, dann:
    - führt **load_correction_file** automatisch ein `select_cor_table( HeadA, HeadB )` mit den zuletzt verwendeten Werten für `HeadA` und `HeadB` aus
    - positioniert **load_correction_file** die Galvanometerscanner mit einem internen Sprung mit der aktuell eingestellten Sprunggeschwindigkeit auf die dort vorgesehene Ausgabeposition
  - Wenn **load_correction_file** vor **load_program_file** aufgerufen wird, dann wird die Korrekturtabelle lediglich gespeichert. Es kann kein interner Sprung auf die vorgesehene Ausgabeposition stattfinden.

(1) Korrekturdateien können auch geladen werden, bevor die RTC6-Dateien mit **load_program_file** geladen wurden.

1to1-Korrekturtabellen

1to1-Korrekturtabellen

- Sind keinem speziellen Scan-System zugeordnet
- Dienen im Allgemeinen nur Testzwecken
- Können alternativ durch **load_correction_file** mit dem Parameter `Name` = NULL geladen werden.
  - Gemäß dem weiteren Parameter `Dim` wird dabei intern eine 2D- oder 3D-1to1-Korrekturtabelle generiert.

Weil manche Anwenderprogramme einen Dateinamen erfordern und NULL als Namen nicht akzeptieren, wird die 1to1-Korrekturdatei `Cor_1to1.ct5` im RTC6-Software-Paket mitgeliefert. Dieses enthält als Kalibrierfaktor den Wert 0. Sollte ein Anwenderprogramm unbedingt einen "echten" Kalibrierfaktor benötigen, dann kann mit `CorrectionFileConverter.exe`, siehe Abschnitt "Verzeichnis CorrectionFileConverter", Seite 33, ein entsprechender Wert angegeben werden (Schaltfläche **Show File Header** > Feld 'Field Calibration [Bit/mm]').

Inverse Tabellen

Das ct5-Korrekturdatei-Format unterstützt "Inverse Tabellen". Diese werden benötigt, um Scan-Kopf-Ist-Positionen auf die zugehörigen Bildfeld-Koordinaten rückwärts zu transformieren.

Zu weiteren Informationen siehe:

- Parameter 11, siehe Seite 187
- Abschnitt "Konvertierung von Korrekturdateien", Seite 188
- Kapitel 8.1.3 "Positionierung überwachen", Seite 226

ct5-Korrekturdateikopf

Der ct5-Korrekturdatei-Kopf enthält 16 Parameter, siehe folgende Tabelle.

Diese können ausgelesen und auf diese Weise direkt in ein Anwenderprogramm eingebunden werden:

- aus aktuell geladenen Korrekturtabellen mit **get_table_para**
- aus zugewiesenen Korrekturtabellen mit **get_head_para**

Hinweise

- Bei ctb-Korrekturdateien sind manche Parameterwerte ausschließlich der mitgelieferten `*_ReadMe.txt`-Datei zu entnehmen. Diese müssen händisch in das Anwenderprogramm übertragen werden.
- Eine `ct5`-Datei, die durch Konvertierung aus einer `ctb`-Datei erzeugt wurde, siehe Abschnitt "Konvertierung von Korrekturdateien", Seite 188, enthält nicht alle Parameter-Informationen.
- Eine 1to1-Korrekturdatei enthält nicht alle Parameter-Informationen.
- Die Parameter 3…7 sind nur für 3D-Markierungen relevant. In 2D-Korrekturtabellen sind sie 0.
- Bei gleichem 3-Achsen-Scan-System sind die ABC-Koeffizienten (Parameter 5…7) in ctb-Korrekturdateien und ct5-Korrekturdateien identisch.

Parameter[(a)]	Beschreibung
0	Typ der Korrekturtabelle. • = 0,0: 2D-Korrekturtabelle • = 1,0: 3D-Korrekturtabelle
1	Kalibrierfaktor K_{xy} [Bit/mm]. Siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175.
2	Brennweite oder Arbeitsabstand [mm]. • Bei einer Konfiguration mit Scan-Objektiv: die effektive Brennweite des Objektivs [mm]. • Bei einer Konfiguration ohne Scan-Objektiv: der Arbeitsabstand A [mm]. A = Abstand zwischen der optischen Achse des auf den ersten Ablenkspiegel einfallenden Laserstrahls und der Bildfeld-Ebene.
3	Streckfaktor in x-Richtung. Kompensiert bei 3D-Markierungen die in z-Richtung vorhandene pyramidenförmige Bildfeld-Änderung.
4	Streckfaktor in y-Richtung. Siehe Parameter 3.
5	Koeffizient *A* des Polynoms zur z-Achsen-Ansteuerung, siehe Seite 178: Offset-Anteil, ±26 Bit.
6	Koeffizient *B* des Polynoms zur z-Achsen-Ansteuerung, siehe Seite 178: linearer Anteil, ±11 Bit.
7	Koeffizient *C* des Polynoms zur z-Achsen-Ansteuerung, siehe Seite 178: quadratischer Anteil, ±4 Bit.
8	Nummer der Korrekturdatei. Der Parameter entspricht bei von SCANLAB ausgelieferten Korrekturdateien der Zahl im Dateinamen (z.B. 145 für `D2_145.ct5` oder `D3_145.ct5`).

Parameter[(a)] (Forts.)	Beschreibung (Forts.)
9	Unterscheidung zwischen Korrektur mit oder ohne F-Theta-Objektiv. Es gilt: Parameter = 10 × P_{Obj} + P_{Typ} mit • P_{Obj}= 0: Korrektur ohne F-Theta-Objektiv P_{Obj}= 1: Korrektur mit F-Theta-Objektiv • Falls Korrektur mit F-Theta-Objektiv: P_{Typ} = 0,0: ohne Verzeichnungsdaten P_{Typ} = 1,0: mit Reihenentwicklung der F-Theta-Bedingung P_{Typ} = 2,0: mit Bildhöhentabelle
10	Indikator für die Herkunft der Korrekturdatei. Es gilt: Parameter = 1.000 × P_{Orig} + P_{Ver} mit • P_{Orig} = 10.000: original berechnete Datei P_{Orig} = 20.000: aus `ctb`-Datei konvertiert P_{Orig} = 30.000: aus txt-Datei rekonstruiert Durch Manipulation einer Korrekturdatei mittels bei SCANLAB erhältlichen Korrekturprogrammen wird P_{Orig} jeweils um 1 erhöht. • P_{Ver} = Versionsnummer des verwendeten Programms zur Erzeugung der Korrekturdatei
11	Information zur inversen Tabelle. Es gilt Parameter = P_{Exist} + 2 × P_{Calc} mit • P_{Exist} = 1,0: gültige inverse Tabelle vorhanden P_{Exist} = 0,0: keine gültige Tabelle vorhanden • Falls korrekte inverse Tabelle vorhanden: P_{Calc} = 0: Inverse Tabelle ab initio berechnet P_{Calc} = 1: Inverse Tabelle numerisch berechnet
12	Winkelkalibrierung des Scan-Systems. Mechanische Winkelauslenkung in [± °] bei 96% der maximalen Ansteuerung.
13	Kennzahl für die zur Berechnung verwendete Scan-Kopf-Geometrie (nur zur internen Verwendung), z.B. • = –1,0: unbekannte Geometrie (z.B. bei einer aus einer `ctb`-Datei konvertierten Tabelle) • = 0,0: Standard-Geometrie
14	Indikator, ob ein zusätzliches Schutzglas berücksichtigt wurde. Es gilt: Parameter = 1.000.000 × P_T + 1.000 × P_I mit • P_T = Schutzglas-Dicke in mm (mit höchstens 2 Nachkommastellen) • P_I = Brechungsindex (mit höchstens 3 Nachkommastellen) Beispiel: Der Wert 3.521.450,0 entspricht einer Schutzglas-Dicke von 3,52 mm und einem Brechungsindex von 1,450.
15	Indikator, ob eine Begrenzung der Bildfeld-Größe in der Korrekturdatei vorgenommen wurde. • = 0,0: ohne Begrenzung der Feldgröße • = 2,0: mit Begrenzung der Feldgröße

(a) Nummerierung gemäß **get_table_para** und **get_head_para**.

Konvertierung von Korrekturdateien

Zur Konvertierung von `ctb`-Korrekturdateien in `ct5`-Korrekturdateien und umgekehrt ist im RTC6-Software-Paket das Programm `CorrectionFileConverter.exe` enthalten. Das zugehörige Handbuch wird in Englisch und Deutsch mitgeliefert.

Bei der Konvertierung einer `ctb`-Korrekturdatei in eine `ct5`-Korrekturdatei wird im Datei-Kopf der erzeugten `ct5`-Korrekturdatei ein entsprechender Herkunftsindikator eingetragen (Parameter 10, Seite 187, P_{Orig} = 20.000).

Solche Konvertierungen liefern *keine* vollwertige `ct5`-Korrekturdateien, da im Datei-Kopf einige Parameter fehlen. Diese können aber mit `CorrectionFileConverter.exe` (über die Schaltfläche **Show File Header**) manuell eingetragen werden.

Zusätzlich kann es vorkommen, dass die bei der Konvertierung numerisch zu berechnende inverse Korrekturtabelle nicht das gesamte Bildfeld abdeckt oder nicht berechenbar ist. In diesem Fall wird stattdessen eine 1to1-Tabelle eingefügt, siehe auch Parameter 11, Seite 187.

Die Konvertierung einer `ct5`-Korrekturdatei in eine `ctb`-Korrekturdatei liefert dagegen eine vollwertige `ctb`-Korrekturdatei. Die Parameter-Informationen aus dem `ct5`-Korrekturdateikopf sind:

- nicht mehr enthalten in einer `ctb`-Korrekturdatei für 2D-Korrekturtabellen
- nur teilweise enthalten in einer `ctb`-Korrekturdatei für 3D-Korrekturtabellen

7.3.6 Ausgabewerte an das Scan-System

Berechnung

Berechnungsschritte zum Erzeugen der RTC-Ausgabewerte an das Scan-System aus den aktuellen x-, y-, x-Koordinatenwerten
(1) Zerlegen der Vektoren und Bögen in Mikroschritte[a]
(2) Wenn zutreffend: Anwenden einer Wobbelbewegung[b] – z.B. aus **set_wobbel_mode**-Aufruf
(3) Wenn zutreffend: Anwenden einer globalen Koordinatentransformation im virtuellen Bildfeld[c] – z.B. aus Aufrufen von **set_matrix**, **set_angle**, **set_offset** mit `HeadNo` = 4
(4) Wenn zutreffend: Anwenden der eingestellten "Processing on the fly"-Korrektur[d] – z.B. aus Aufrufen von **set_fly_x** oder **set_fly_x_pos**
(5) Wenn zutreffend: Anwenden von Koordinatentransformationen zum Ausrichten des 1 (der 2) Scan-Systems(e) relativ zum Bildfeld[e] (a) Transformieren der x-Koordinatenwerte und y-Koordinatenwerte gemäß Matrix × Winkel × Skalierung ("2D-Transformation") • z.B. aus Aufrufen von **set_matrix**, **set_angle**, **set_scale** und entsprechender Listenbefehle (b) Anwenden eines Offsets auf die x-Koordinatenwerte und y-Koordinatenwerte sowie z-Koordinatenwerte[f] • z.B. aus Aufrufen von **set_offset** oder **set_offset_xyz** und entsprechender Listenbefehle
(6) Clippen der x-Koordinatenwerte und y-Koordinatenwerte auf die Randwerte des realen 20-Bit-Bildfelds, sobald sie den realen Bildfeld-Bereich übersteigen (virtuelles Bildfeld)[g]
(7) Wenn zutreffend: Korrigieren der x-Koordinatenwerte und y-Koordinatenwerte entsprechend der zugewiesenen Korrekturtabelle ("2D-Bildfeld-Korrektur")[h] – Aus dem **select_cor_table**-Aufruf oder **select_cor_table_list**-Aufruf
(8) Wenn zutreffend: Anwenden der Fokusverschiebung auf z-Koordinatenwerte – z.B. aus **set_defocus**-Aufruf oder **set_defocus_list**-Aufruf
(9) Wenn zutreffend: Korrigieren der z-Koordinatenwerte entsprechend der zugewiesenen Korrekturtabelle ("3D-Bildfeld-Korrektur")[h] – Aus dem **select_cor_table**-Aufruf oder **select_cor_table_list**-Aufruf
(10) Clippen der z-Koordinatenwerte auf die Randwerte des realen 20-Bit-Bildfelds, sobald sie den realen Bildfeld-Bereich übersteigen (virtuelles Bildfeld)[i]
(11) Wenn zutreffend (nur manche ältere Scan-Systeme): Anwenden einer explizit[j] oder automatisch[k] eingestellten Gain-Korrektur und Offset-Korrektur für x-Galvanometerscanner und y-Galvanometerscanner – Aus **set_hi**-Aufruf (explizit) – Aus **auto_cal**-Aufruf (automatisch)

(a) Siehe Kapitel 7.1.2 "Zerlegung in Mikroschritte", Seite 143.

(b) Siehe Kapitel 8.4 "Wobbelmode", Seite 244.

(c) Siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 177.

(d) Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254.

(e) Siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236 und Kapitel 8.5.2 "3D-Scan-Systeme", Seite 249.

(f) Es wird immer eine vollständige 3D-Berechnung ausgeführt. Wenn die Option "3D" nicht freigeschaltet ist, werden lediglich keine z-Werte ausgegeben.

(g) Siehe Kapitel 7.3.3 "Virtuelles Bildfeld", Seite 177.

(h) Siehe Kapitel 7.3.5 "Bildfeld-Korrektur und Korrekturtabellen", Seite 181.

(i) Siehe Kapitel 7.3.3 "Virtuelles Bildfeld", Seite 177.

(j) Siehe Abschnitt "Kundeneigene Kalibrierung", Seite 290.

(k) Siehe Kapitel 8.10 "Automatische Selbstkalibrierung", Seite 287.

Hinweise

- Überlaufende Werte werden immer auf die Randwerte des maximal möglichen Werte-Bereichs geclippt.
- Es wird immer eine vollständige 3D-Berechnung ausgeführt. Wenn die Option "3D" nicht freigeschaltet ist, werden lediglich keine z-Werte ausgegeben.

Wertebereiche

Für alle Scan-System-Achsen werden 20-Bit-Werte mit Vorzeichen (–524.288…+524.287) ausgegeben. Das gilt auch für Werte, die das Scan-System and die RTC6 PCIe-Karte zurückgibt, siehe auch:

- Abschnitt "RTC4-Kompatibilitätsmodus", Seite 176
- Abschnitt "RTC5-Kompatibilitätsmodus", Seite 176
- Kapitel 4.5.2 "XY2-100-Konverter (Zubehör)", Seite 71

Vorausberechnung und Diagnose

Mit **get_galvo_controls** können die für die angegebenen Koordinaten und Einstellparameter in der momentanen Konfiguration (Korrekturtabelle, Koordinatentransformationen) resultierenden Ausgabewerte bestimmt werden, ohne dass sich die Galvanometerscanner dabei bewegen.

Takt-Überläufe

Bei den sehr rechenintensiven Sprung-Befehlen, Mark-Befehlen und Arc-Befehlen kann es vorkommen, dass der 10 μs-Takt nicht ausreicht, alle erforderlichen Daten zu berechnen, wenn sehr viele der angebotenen Möglichkeiten gleichzeitig verwendet werden, so z.B. die gleichzeitige Ansteuerung zweier Scan-Systeme, eine Wobbelbewegung, Koordinatentransformation im virtuellen Bildfeld, "Processing on the fly" in zwei Achsen (Korrektur über McBSP-Schnittstelle), "Automatische Laser-Steuerung", Para-Vektoren, Datenaufzeichnung, "Variables Polygon-Delay", kurze Listenbefehle (z.B. in einem Polygonzug), Kontrollbefehle während der Listenausführung.

Diese Überlauf-Situation kann intern erkannt und gezählt werden. Testen Sie Ihr Anwenderprogramm entsprechend, indem Sie mit **get_overrun** die Anzahl der Überläufe abrufen. Die Auswirkungen eines solchen Überlaufes sind derart, dass ein oder mehrere Peripherie-Ports nicht mehr innerhalb des aktuellen 10 μs-Taktes bedient werden können, darunter u.U. auch die Ausgabe für die Scan-Köpfe. Die Galvanometerscanner-Bewegung könnte also für 10 μs stocken.

7.3.7 Statusüberwachung und Diagnose

Zur Überwachung des Betriebsstatus und zu Diagnosezwecken können mit **get_value** oder **get_values** eine Reihe von Signalen ausgelesen werden:

- Ein 20-Bit-Statuswort, das vom Scan-System zurückgegeben wird, siehe Abschnitt "Statusinformationen vom Scan-System", Seite 191
- Das aktuelle LASERON-Signal
- Die aktuellen kartesischen Ansteuerwerte (d.h. die für beide Scan-Kopf-Anschlüsse noch gemeinsamen, sogenannten "Sample-Werte", siehe #2, #3 und #4.
- Die (für den jeweiligen Scan-Kopf-Anschluss spezifischen) Ansteuerwerte unter Berücksichtigung
  - der mit **set_matrix**, **set_scale**, **set_angle** oder den entsprechenden Listenbefehlen festgelegten Koordinatentransformationen, siehe #5a.
  - des mit **set_offset_xyz** oder **set_offset_xyz_list** festgelegten Offsets auf die z-Koordinate, siehe #5b.
  - des mit **set_defocus** oder **set_defocus_list** festgelegten Offsets auf die Fokuslänge und der geladenen Korrekturtabellen, siehe #8 und #9.

Jedes dieser Signale kann auch mit **set_trigger[*]** über einen längeren Zeitraum – mit einstellbarer Messperiode – auf der RTC6 PCIe-Karte aufgezeichnet werden. Danach können sie mit **get_waveform** zur Auswertung an den PC übertragen werden.

Mit **set_trigger**(`Period` Bit #31 = 1) (Ringpuffer-Funktionalität) können praktisch beliebige Datenmengen aufgezeichnet werden. Die aufgezeichneten Werte können während der Listenausführung mit **get_waveform_offset** blockweise zum PC übertragen werden. Nachdem der Messwertspeicher seine Kapazität erreicht hat, beginnt die Messwertaufzeichnung automatisch von vorn. Der aktuelle Status und Fortschritt einer durch **set_trigger[*]** gestarteten Messung kann mit **measurement_status** abgefragt werden. Die vom Scan-System übertragenen Werte sind immer 20-Bit-Werte.

Statusinformationen vom Scan-System

Das Scan-System überträgt alle 10 µs über das SL2-100-Protokoll die folgenden Signale an die RTC6 PCIe-Karte:

- Ein 20-Bit-Statuswort.
  Es kann folgende Datentypen bedeuten:
  - Das eigentliche Statuswort (die oberen 16 Bit sind identisch zum 16-Bit-Statuswort des XY2-100-Protokolls (XY2-100-Statuswort), zur Beschreibung siehe **get_head_status**).
  - Nur bei iDRIVE-Scan-Systeme, siehe Kapitel 8.1 "iDRIVE-Funktionen", Seite 224:
    Ein anderer Datentyp, der mit **control_command** auswählbar ist.

  Das 20-Bit-Statuswort kann:
  - ausgelesen werden mit **get_value**/**get_values**
  - aufgezeichnet werden mit **set_trigger[*]**
- 6 zusätzliche Statusbits.
  Bei iDRIVE-Scan-Systeme mit SL2-100-Protokoll werden die Statusbits (PowerOK, TempOK und PosAck; pro Achse) parallel zum und unabhängig vom 20-Bit-Statuswort übertragen. Dadurch können diese Statusbits auch dann zur Überwachung des Scan-Systems verwendet werden (siehe Abschnitt "Automatische Unterdrückung der Lasersteuersignale", Seite 195), wenn z.B. eine "Automatische Laser-Steuerung" (siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208) aktiv ist, die einen anderen Rückgabe-Datentyp erfordert.
  Die 6 zusätzlichen Statusbits können mit **get_head_status** ausgelesen werden.

Hinweise

- Bei Scan-Systemen mit XY2-100-Protokoll muss der XY2-100-Konverter (Zubehör) verwendet werden.
- Bei iDRIVE-Scan-Systeme mit XY2-100-Protokoll muss das eigentliche XY2-100-Statuswort als zu übertragender Datentyp eingestellt sein (Default-Typ nach dem Einschalten).

7.4 Laser-Steuerung

Die RTC6 PCIe-Karte gibt an bestimmten Ausgängen Signale aus, mit denen verschiedene Lasertypen gesteuert werden können ("Lasersteuersignale" LASERON, LASER1, LASER2). Diese Ausgänge sind an der:

- LASER-Buchse, Seite 75
- EXTENSION 2-Stiftleiste, Seite 83

Der Laser-Steuerungsmodus wird mit **set_laser_mode** eingestellt:

- Der CO_2-Modus (Laser-Mode 0, Default nach **load_program_file**) ist konzipiert, um CO_2-Laser zu steuern.
- YAG-Modus 1 (Laser-Modus 1), YAG-Modus 2 (Laser-Modus 2), YAG-Modus 3 (Laser-Modus 3), YAG-Modus 5 (Laser-Modus 5) sind konzipiert, um Laser aus der Nd:YAG-Familie (und verwandte) zu steuern.
- Laser-Mode 4 und Laser-Mode 6 sind konzipiert, um freilaufende Laser zu steuern.(1)

Alle Lasersteuersignale sind TTL-Signale. Mit **set_laser_control** wird eingestellt, ob sie HIGH-aktiv oder LOW-aktiv sind, siehe auch Kapitel 4.6.1 "LASER-Buchse", Seite 75. Die aktuelle Einstellung kann mit **get_startstop_info** (Bit #13) abgefragt werden.

Zu den maximalen Last-Werten siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", Seite 973.

(1) Zur Synchronisation auf extern gesteuerte freilaufende Laser siehe Kapitel 7.4.10 "Synchronisierung von RTC6-Takt und externem Taktsignal", Seite 219.

7.4.1 Lasersteuersignale freigeben, aktivieren und schalten

Alle Lasersteuersignale sind unterdrückt:

- Nach einem Hardware-Reset
- Nach der Initialisierung mit **load_program_file**

Dann befinden sich alle Lasersteuersignal-Ausgänge (LASERON, LASER1 und LASER2) im hochohmigen Zustand.

Bevor Lasersteuersignale überhaupt ausgegeben werden können, muss zuerst deren Polarität mit **set_laser_control** eingestellt werden. Damit wird gleichzeitig ihr tristate-Zustand aufgehoben (= "globale Freigabe").
Als Voreinstellung stehen dann LASERON und LASER1/LASER2 auf ihrem jeweiligen "Off"-Pegel.

Der tristate-Zustand wird erst wieder eingestellt:

- nach einem Hardware-Reset (Neustart)
- einem Aufruf von **load_program_file**

Des weiteren müssen reale Lasersteuersignale freigegeben werden:

- entweder gleichzeitig mit **set_laser_control**(Bit #2 = 0) oder
- nachfolgend mit dem separaten Befehl **enable_laser**

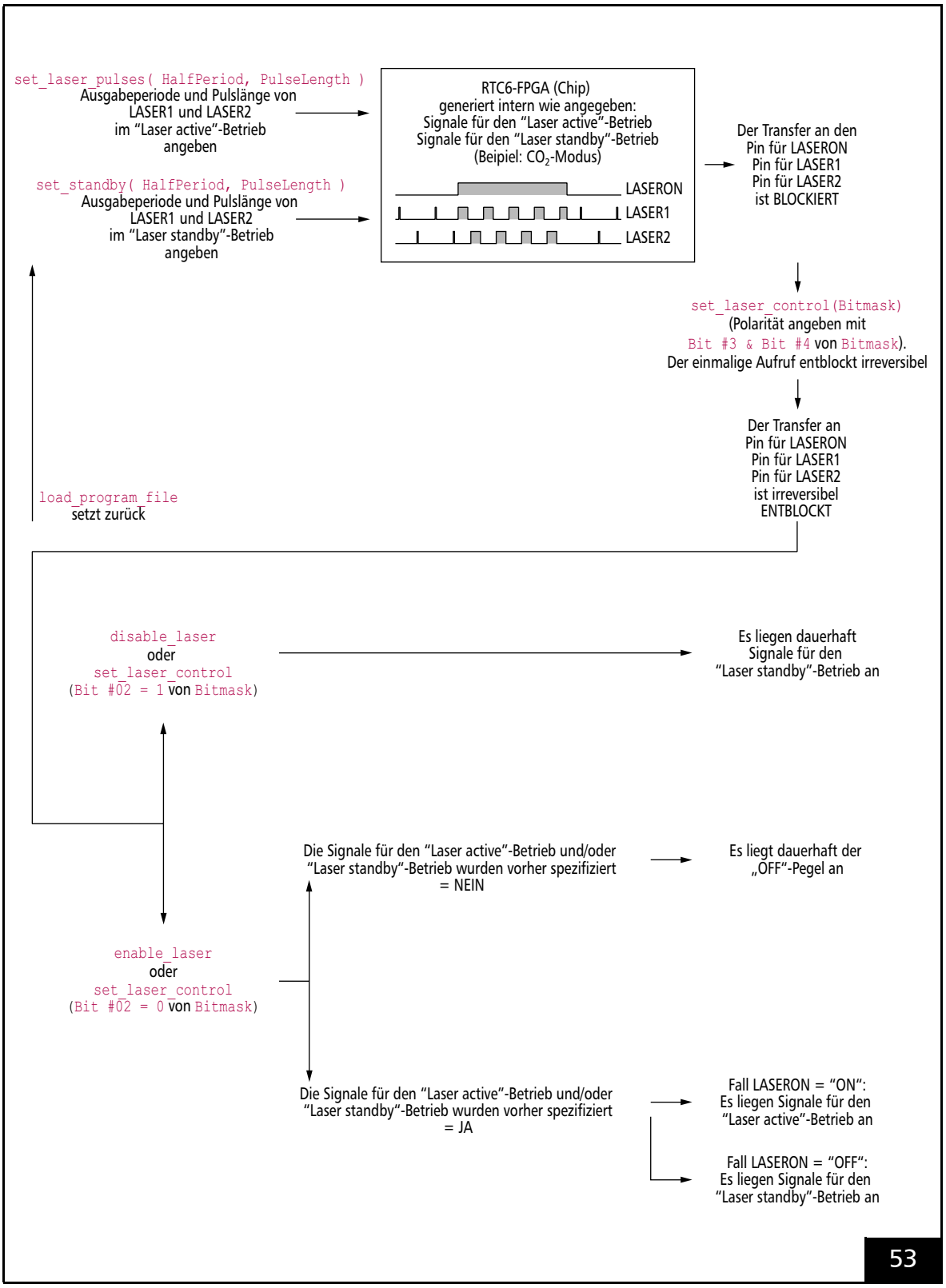
Diese können mit **disable_laser** oder **pause_list** wieder unterdrückt werden. In beiden Fällen werden alle Lasersteuersignale auf ihren jeweiligen "Off"-Pegel gestellt.

Mit **get_startstop_info** (Bit #9, Bit #10, Bit #13, Bit #14) können ausgelesen werden:

- Die "globale Freigabe" durch **set_laser_control**
- Die Polaritäten von LASERON und LASER1/LASER2
- Die Freigabe der Signale für den "Laser active"-Betrieb durch **enable_laser**

Allgemeine Hinweise zu den Lasersteuersignalen finden Sie im Abschnitt "Signale für den "Laser active"-Betrieb", Seite 194 und Abschnitt "Signale für den "Laser standby"-Betrieb", Seite 194.
Modus-spezifische Details hängen vom eingestellten Laser-Mode ab, siehe Kapitel 7.4.3 "CO_2-Modus", Seite 196 bis Kapitel 7.4.8 "Pulse-Picking-Laser-Modus", Seite 206.

Die Zuordnung der Lasersteuersignale zu den jeweiligen Ausgabe-Pins an der LASER-Buchse kann konfiguriert werden, siehe Kapitel 7.4.2 "LASER-Buchse konfigurieren", Seite 195.



RTC6-Karten-Hardware und Laser-Steuerungs-bezogene RTC6-Befehle.

Signale für den “Laser active”-Betrieb

Mit **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** können die Signale für den “Laser active”-Betrieb

- aktiviert werden: `HalfPeriod` ≠ 0 und `PulseLength` ≠ 0
- deaktiviert werden: `HalfPeriod` = 0 und/oder `PulseLength` = 0

Auch wenn die Signale für den “Laser active”-Betrieb freigegeben und aktiviert wurden, *werden sie nur dann an den Ausgängen ausgegeben, wenn sie durch weitere Befehle eingeschaltet werden*.

Sie werden automatisch eingeschaltet, wenn:

- Markier-Befehle aufgerufen werden, siehe Kapitel 7.1.1 “Mit Vektor-Befehlen und Bogen-Befehlen markieren”, Seite 139

Sie werden automatisch ausgeschaltet, wenn:

- auf einen Markier-Befehl ein normaler Nicht-Markierbefehl folgt
- eine Liste mit **set_end_of_list** oder **stop_execution** beendet wird
- eine Liste mit **set_wait**, **pause_list** oder **stop_list** vorübergehend angehalten wird

Im letzteren Fall werden die Signale für den “Laser active”-Betrieb wieder eingeschaltet, wenn die Liste mit **release_wait** oder **restart_list** fortgesetzt wird.

Sie können aber auch innerhalb einer Liste mit **laser_signal_on_list** und **laser_signal_off_list** oder außerhalb einer Liste mit **laser_signal_on** und **laser_signal_off** zeitlich unbegrenzt ein- und ausgeschaltet werden.

Pulsvervollständigung

Ob ein bei eingeschaltetem LASERON-Signal gestarteter Modulationspuls (Q-Switch-Puls) an LASER1 noch vollständig ausgeführt oder abgeschnitten wird, wenn er beim Ausschalten des LASERON-Signals noch nicht vollständig abgearbeitet ist, kann mit **set_laser_control**(Bit #0) eingestellt werden.

Signale für den “Laser standby”-Betrieb

Mit **set_standby** oder **set_standby_list** können die Signale für den “Laser standby”-Betrieb

- aktiviert werden: `HalfPeriod` ≠ 0 und `PulseLength` ≠ 0
- deaktiviert werden: `HalfPeriod` = 0 und/oder `PulseLength` = 0

Die Signale für den “Laser standby”-Betrieb werden nach deren Aktivierung (ohne weitere Befehle) ständig an den Ausgängen ausgegeben, wenn keine Signale für den “Laser active”-Betrieb eingeschaltet sind.

Wenn die Signale für den “Laser active”-Betrieb deaktiviert sind (z.B. durch `PulseLength` = 0), dann erfolgt keine Umschaltung. Dann werden die Signale für den “Laser standby”-Betrieb auch während der Ausführung von Markier-Befehlen ständig ausgegeben.

Sind die Signale für den “Laser standby”-Betrieb deaktiviert, so sind im “Laser standby”-Betrieb alle Ausgänge auf den “Off”-Pegel gesetzt.

Wenn aktivierte Signale für den “Laser standby”-Betrieb beim Anhalten einer Liste mit **pause_list** deaktiviert werden sollen (automatisch werden hier nur die Signale für den “Laser active”-Betrieb ausgeschaltet), müssen Benutzer dies durch einen Aufruf von **set_standby**(`PulseLength` = 0) explizit veranlassen.

Die aktuellen “Laser standby”-Parameter, die möglicherweise innerhalb einer Liste verändert worden sind, können mit dem Kontrollbefehl **get_standby** ausgelesen werden.

Pulsvervollständigung

Bei den Signalen für den “Laser standby”-Betrieb wird eine Pulsvervollständigung, siehe Pulsvervollständigung, Seite 194, nicht unterstützt.

Automatische Unterdrückung der Lasersteuersignale

Fall: Status-Fehler des Scan-Systems

Mit **set_laser_control**(Bit #16…Bit #27) kann eingestellt werden, dass die Lasersteuersignale automatisch unterdrückt werden, wenn das entsprechende Statussignal des Scan-Systems (PowerOK, TempOK, PosAck der Achse X/Y von Kopf A/B) einen Fehler anzeigt (d.h. 0 ist; "NOK").

Sobald mindestens eines der angegebenen Statussignale gleich 0 ist, dann:

- Die Ausgabe der Lasersteuersignale wird automatisch unterbrochen. Sie wird nur dann wieder aufgenommen, wenn alle ausgewählten Statussignale gleichzeitig 1 sind (mit **set_laser_control**, **disable_laser** oder **pause_list** unterdrückte Laser-Steuerungssignale bleiben unabhängig vom aktuellen Wert der Statussignale unterdrückt)
- Interne Fehler-Bits werden (kumulativ) gesetzt (diese können mit **get_marking_info** ausgelesen werden)
- Wenn mit **set_laser_control**(Bit #28 = 1) so festgelegt – wird automatisch ein **stop_execution** ausgeführt (die Liste hält an, Lasersteuersignale gehen dauerhaft aus)
- Wenn mit **set_laser_control**(Bit #29 = 1) außerdem so festgelegt – wird das **stop_execution** als /Master-STOP (siehe Abbildung 72) an alle Master/Slave-verbundenen RTC6-Karten weitergeleitet. Ob das Signal dort wirksam ist, kann mit **master_slave_config** für jede RTC6-Karte individuell eingestellt werden. Das dient dazu, dass der Laser auch dann automatisch ausgeschaltet wird, wenn der Fehler auf einer RTC6-Karte auftritt, die nicht für die Steuerung des Lasers vorgesehen ist.

Fall: Bereichsüberschreitung der Galvanometerscanner

Mit **range_checking** kann festgelegt werden, dass die Lasersteuersignale automatisch unterdrückt werden, wenn ein Galvanometerscanner eine vorgegebene Bereichsgrenze überschreitet.

Sie werden automatisch wieder eingeschaltet, sobald der nächste Markier-Befehl innerhalb des erlaubten Bereichs startet. Das heißt, ein unterbrochener Polygonzug bleibt für den Rest des Polygonzug weiter unterdrückt.

7.4.2 LASER-Buchse konfigurieren

Pin (01), (02) und (09) der LASER-Buchse können mit **config_laser_signals** und **config_laser_signals_list** konfiguriert werden, siehe Abbildung 17.

Bei Verwendung unterschiedlicher Laser oder Laser-Betriebsarten ersparen diese Befehle u.U. das Konfigurieren mittels Hardware (unterschiedliche Kabel und Schalter).

Während in der Standard-Einstellung (für normale Markierungen) das LASERON-Signal als Laser-Startsignal am LASERON-Kanal ausgegeben wird, kann beispielsweise für den Pixel-Mode mit Pulse-Picking das LASER2-Signal als Gate-Signal an den LASERON-Kanal gelegt werden.

Für andere Betriebsarten kann auch das FirstPulseKiller-Signal, Seite 373 als Laser-Startsignal auf den LASERON-Kanal gelegt werden. Auf diese Weise können LASER1-Signale bereits vor einem verzögerten Einschalten des Lasers ausgegeben werden (in der Standard-Einstellung ist das nicht möglich).

Die nachfolgende Beschreibung der verschiedenen Laser-Modi bezieht sich auf die Standard-Einstellung der Lasersteuersignal-Ausgabe.

7.4.3 CO_2-Modus

`set_laser_mode( Mode = 0 )` stellt den CO_2-Modus (“Laser-Mode 0”) ein.

Das Zeitdiagramm, siehe Abbildung 54, zeigt die zugehörigen Signale am Beispiel eines isolierten Mark-Befehls.

Für den “Laser active”-Betrieb werden:

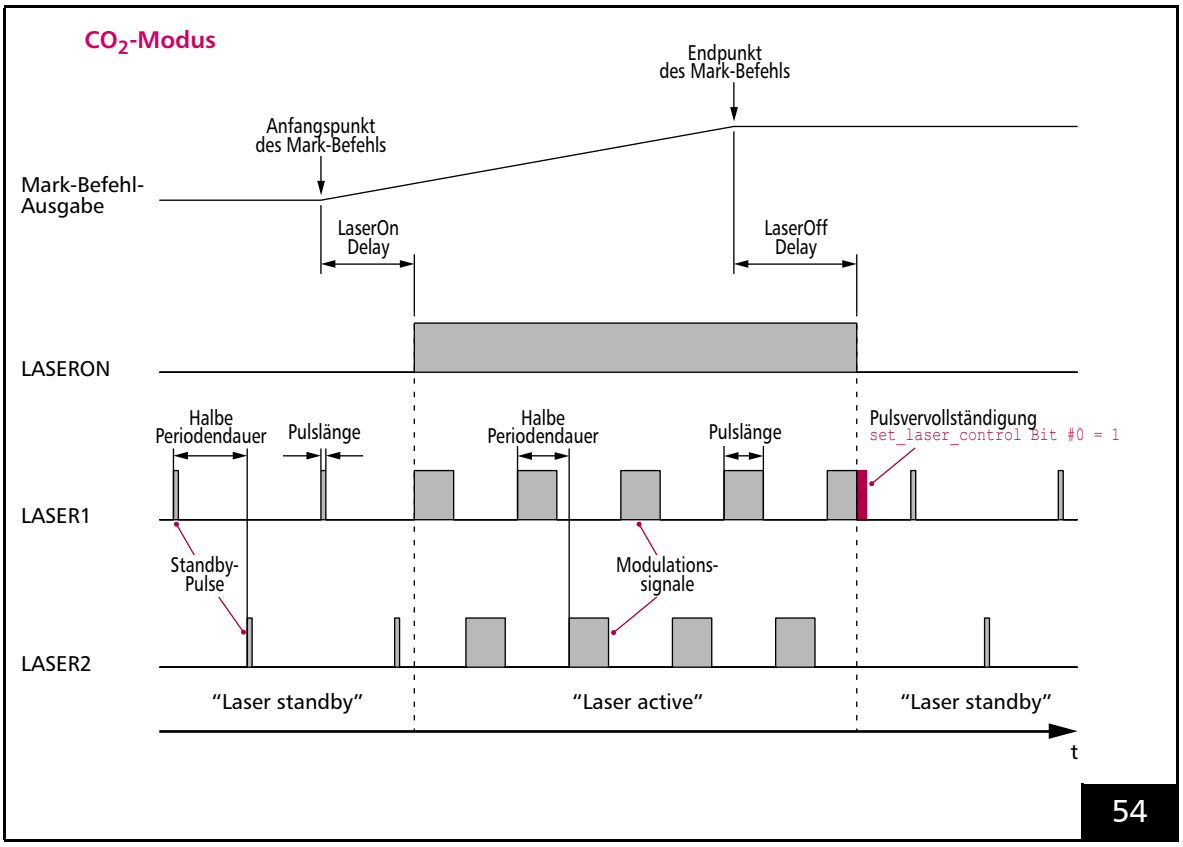
- Das LASERON-Signal eingeschaltet
- 2 alternierende Modulationssignale am LASER1-Ausgang und LASER2-Ausgang ausgegeben.
  Deren Pulslänge und Periodendauer können mit **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** eingestellt werden.

Für den “Laser standby”-Betrieb werden:

- Das LASERON-Signal ausgeschaltet
- Alternierende Standby-Pulse am LASER1-Ausgang und LASER2-Ausgang ausgegeben.
  Deren Pulslänge und Periodendauer können mit **set_standby** oder **set_standby_list** eingestellt werden.

Hinweise

- Die LASER1-Signale und LASER2-Signale werden aktiviert mit:
  `HalfPeriod` ≠ 0 und `PulseLength` ≠ 0
- Die LASER1-Signale und LASER2-Signale werden deaktiviert mit:
  `HalfPeriod` = 0 und/oder `PulseLength` = 0
  (Defaulteinstellung nach **load_program_file**).
- Das LASER2-Signal ist gegenüber dem LASER1-Signal um eine halbe Signal-Periodendauer phasenverschoben. Es kann für die Steuerung einer zweiten Laserröhre verwendet werden. Mit **set_laser_control**(Bit #1 = 1) können die beiden Signale miteinander vertauscht werden. Zum Steuern der Laserleistung kann die Pulslänge von LASER1 und LASER2 variiert werden. Pulslänge und Periodendauer der beiden Signale sind jeweils gleich groß.
- Für die LASER1-Signale und LASER2-Signale muss die *halbe* Periodendauer angegeben werden.
- **set_laser_pulses** und **set_laser_timing** sind Verzögerte Kurze Listenbefehle. Sie können auch dazu eingesetzt werden, um die Laserparameter zwischen zwei Markier-Befehlen zu verändern.
  Die Zeitbasis für die Signale ist immer:
  - 1/64 μs im RTC6-Standardmodus,
    siehe auch Abschnitt “RTC6-Standardmodus”, Seite 176
  - 1/8 μs im RTC4-Kompatibilitätsmodus,
    siehe auch Abschnitt “RTC4-Kompatibilitätsmodus”, Seite 176
- Die Pulsvervollständigung, Seite 194 wirkt auch auf die LASER2-Signale.



Zeitdiagramm der Signale im CO_2-Modus (mit HIGH-aktiven Lasersteuersignalen).
Beispiel: isolierter Mark-Befehl.

7.4.4 YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5

Mit set_laser_mode kann ein YAG-Modus eingestellt werden:

YAG-Modus	Aufruf
YAG-Modus 1	`set_laser_mode( Mode = 1 )`
YAG-Modus 2	`set_laser_mode( Mode = 2 )`
YAG-Modus 3	`set_laser_mode( Mode = 3 )`
YAG-Modus 5	`set_laser_mode( Mode = 5 )`

Das Zeitdiagramm, siehe Abbildung 55, zeigt die zugehörigen Signale am Beispiel eines isolierten Mark-Befehls.

In jedem YAG-Modus gilt für den “Laser active”-Betrieb:

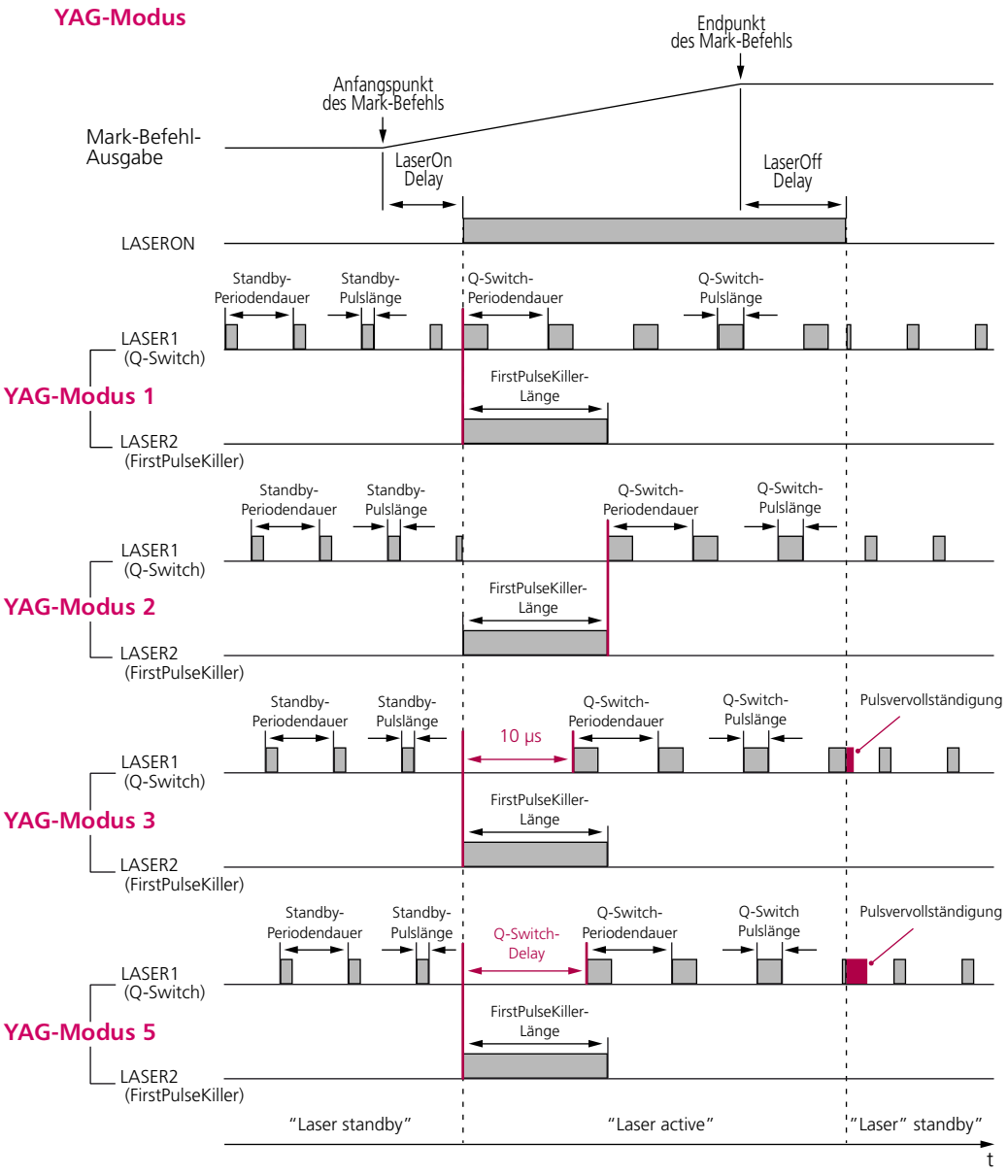
- Das LASERON-Signal wird eingeschaltet
- Ein Q-Switch-Signal wird am LASER1-Ausgang ausgegeben
- Ein einstellbares FirstPulseKiller-Signal, Seite 373 wird am LASER2-Ausgang ausgegeben, siehe auch Abschnitt “FirstPulseKiller-Signal”, Seite 200

In jedem YAG-Modus gilt für den “Laser standby”-Betrieb:

- Das LASERON-Signal wird ausgeschaltet
- Standby-Pulse werden am LASER1-Ausgang ausgegeben
  - Deren Pulslänge und Periodendauer können mit **set_standby** oder **set_standby_list** eingestellt werden.

Hinweise

- Die LASER1-Signale werden aktiviert mit: `HalfPeriod` ≠ 0 und `PulseLength` ≠ 0
- Die LASER1-Signale werden deaktiviert mit: `HalfPeriod` = 0 und/oder `PulseLength` = 0 (Defaulteinstellung nach **load_program_file**).



55

Zeitdiagramm der Signale in YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5 (mit HIGH-aktiven Lasersteuersignalen). Die Standby-Signale sind nicht mit den "Laser active"-Signalen synchronisiert und können beliebige Phasenlagen dazu haben.
Beispiel: isolierter Mark-Befehl.

Q-Switch-Signal

Das Q-Switch-Signal dient zum Steuern des Güteschalters des Lasers.

Mit **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** werden eingestellt:

- Q-Switch-Periodendauer
- Q-Switch-Pulslänge

Hinweise

- Siehe auch Abschnitt "Pulsvervollständigung", Seite 194 (Signale für den "Laser active"-Betrieb).
- Siehe auch Abschnitt "Pulsvervollständigung", Seite 194 (Signale für den "Laser standby"-Betrieb).

FirstPulseKiller-Signal

Das FirstPulseKiller-Signal, Seite 373 dient dazu, die ersten Laserpulse einer Pulsfolge zu signalisieren, wenn sie nicht der üblichen Dauerleistung entsprechen.

Das FirstPulseKiller-Signal, Seite 373 wird lediglich bereitgestellt. Der Laser selbst muss geeignet darauf reagieren.

Das FirstPulseKiller-Signal, Seite 373 wird automatisch mit dem LASERON-Signal ausgegeben.

Die Länge des FirstPulseKiller-Signal, Seite 373 wird mit **set_firstpulse_killer** oder **set_firstpulse_killer_list** eingestellt.

Siehe auch **config_laser_signals**, FirstPulseKiller-Signal, Seite 373.

YAG-Modus-Unterschiede

YAG-Modus 1, YAG-Modus 2, YAG-Modus 3 und YAG-Modus 5 unterscheiden sich nur im relativen Start-Zeitpunkt des ersten Q-Switch-Pulses bezüglich des FirstPulseKiller-Signals, siehe die Zeitdiagramme in Abbildung 55.

Nachdem das Q-Switch-Delay abgelaufen ist, startet der erste Q-Switch-Puls. Der Q-Switch-Delay-Wert kann über **set_qswitch_delay** und**set_qswitch_delay_list** festgelegt werden oder alternativ über die Auswahl des YAG-Modus:

- YAG-Modus 1 = 0
- YAG-Modus 2 = FirstPulseKiller-Signal-Länge
- YAG-Modus 3 = 10 μs
- YAG-Modus 5 = Wert aus **set_qswitch_delay** oder **set_qswitch_delay_list**

Mit **set_laser_mode** wird das Q-Switch-Delay lediglich *voreingestellt*. Daher kann in YAG-Modus 1, YAG-Modus 2 und YAG-Modus 3 das Q-Switch-Delay nachträglich mit **set_qswitch_delay** oder **set_qswitch_delay_list** geändert werden.
In YAG-Modus 2 erfolgt auch mit jedem **set_firstpulse_killer** oder **set_firstpulse_killer_list** eine entsprechende Anpassung des Q-Switch-Delays.

Lampenstrom (Laserleistung)

Zum Steuern des Lampenstroms eines YAG-Lasers können der 12-Bit-Analog-Ausgang 1 ANALOG OUT1 oder 12-Bit-Analog-Ausgang 2 ANALOG OUT2 eingesetzt werden. Sie sind an der LASER-Buchse verfügbar, siehe Abbildung 17.
12-Bit-Analog-Ausgang 2 ANALOG OUT2 ist auch an der MARKING ON THE FLY-Stiftleiste verfügbar, siehe Abbildung 25.

Zum Festlegen des analogen Ausgangssignals werden der Kontrollbefehl **write_da_x** und der Verzögerte Kurze Listenbefehl **write_da_x_list** bereitgestellt.

Alternativ kann der Lampenstrom digital über den 8-Bit-Digital-Ausgang (EXTENSION 2-Stiftleiste) gesteuert werden, siehe auch Abschnitt "8-Bit-Digital-Ausgang", Seite 84. Zum Einstellen des 8-Bit-Digital-Ausgangs stehen **write_8bit_port**, **write_8bit_port_list** oder **write_port_list** zur Verfügung.

Wenn der Lampenstrom geändert wird, kann die Ausführung der Liste mit **long_delay** angehalten werden, bis sich eine konstante Laserleistung eingestellt hat.

7.4.5 Laser-Mode 4

`set_laser_mode( Mode = 4 )` stellt den Laser-Mode 4 ein.

Das Zeitdiagramm, siehe Abbildung 56, zeigt die zugehörigen Signale am Beispiel eines isolierten Mark-Befehls.

Am LASER1-Ausgang werden sowohl für den "Laser active"-Betrieb als auch den "Laser standby"-Betrieb ständig Standby-Pulse ausgegeben. Pulslänge und Periodendauer der Standby-Pulse können mit **set_standby** oder **set_standby_list** eingestellt werden.

Für den "Laser active"-Betrieb gilt:

- Das LASERON-Signal wird eingeschaltet
- Am LASER2-Ausgang wird ein programmierbares FirstPulseKiller-Signal, Seite 373 ausgegeben, siehe auch Abschnitt "FirstPulseKiller-Signal", Seite 201

Für den "Laser standby"-Betrieb gilt:

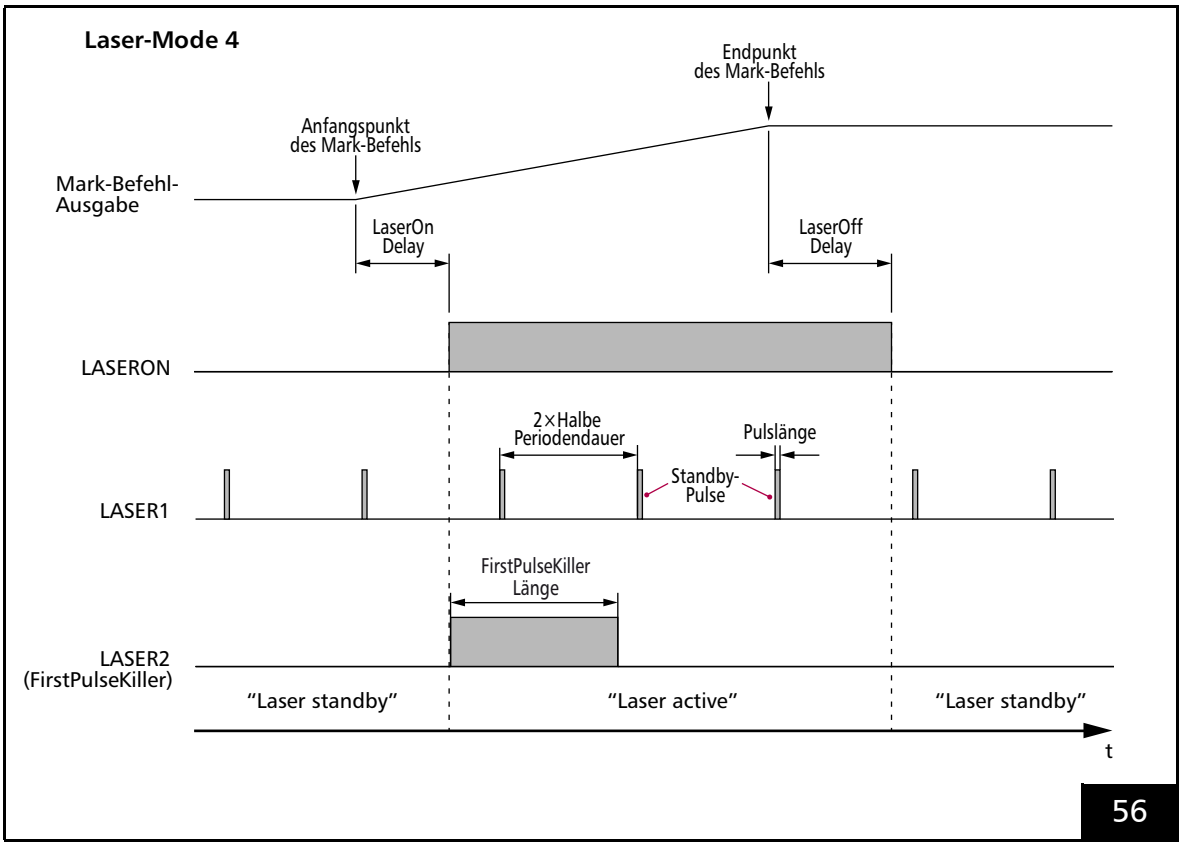
- Das LASERON-Signal wird ausgeschaltet
- Das LASER2-Signal wird ausgeschaltet

FirstPulseKiller-Signal

- Wie Abschnitt "FirstPulseKiller-Signal", Seite 200.

Hinweise

- Die LASER1-Signale werden aktiviert mit: `HalfPeriod` ≠ 0 und `PulseLength` ≠ 0
- Die LASER1-Signale werden deaktiviert mit: `HalfPeriod` = 0 und/oder `PulseLength` = 0 (Defaulteinstellung nach **load_program_file**).
- Der Laser-Mode 4 wird für manche Faserlaser verwendet.



Zeitdiagramm der Signale im Laser-Mode 4 (mit HIGH-aktiven Lasersteuersignalen).
Beispiel: isolierter Mark-Befehl.

7.4.6 Laser-Mode 6

`set_laser_mode( Mode = 6 )` stellt den Laser-Mode 6 ein.

Laser-Mode 6 ist wie Laser-Mode 4 und wird für bestimmte freilaufende Laser bereitgestellt, deren Gate-Signal (LASERON) während der Dauer eines Pulses *nicht* geändert werden darf.

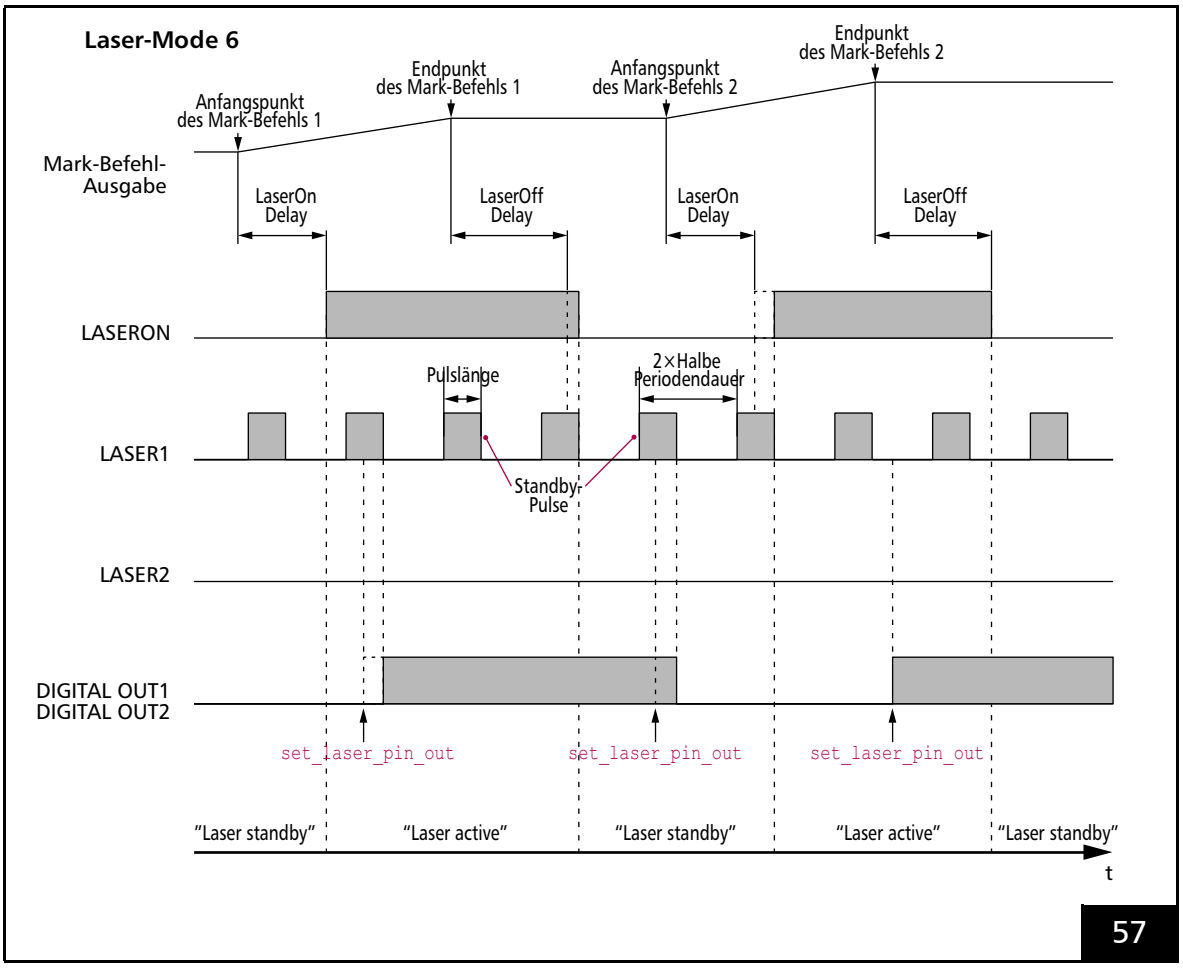
Das Zeitdiagramm, siehe Abbildung 57, zeigt die zugehörigen Signale am Beispiel eines isolierten Mark-Befehls.

Die Laser-Mode 6-Signale und Laser-Mode 4-Signale stimmen überein (siehe auch Hinweise dort, Seite 201) mit folgender Ausnahme:

- Solange ein Standby-Puls aktiv ist, wird ein Schalten des LASERON-Signals entsprechend verzögert. LASERON schaltet 5/64 μs nach dem Ende des Standby-Pulses.

Hinweise

- Es wird kein LASER2-Signal ausgegeben.
- Die Verzögerung des Schaltzeitpunkts wenn noch ein Standby-Puls aktiv ist, gilt auch für das Schalten der Ausgangswerte am 2-Bit-Digital-Ausgang durch **set_laser_pin_out**, **set_laser_pin_out_list** oder **write_port_list**, siehe Abbildung 57.



Zeitdiagramm der Signale im Laser-Mode 6 (mit HIGH-aktiven Lasersteuersignalen).
Beispiel: isolierte Mark-Befehle.

7.4.7 Softstart-Modus (noch nicht implementiert)

Noch nicht implementiert.

Ohne Funktion[1][2][3] sind daher (derzeit):

- **n_set_softstart_level**
- **n_set_softstart_level_list**
- **n_set_softstart_mode**
- **n_set_softstart_mode_list**
- **set_softstart_level**
- **set_softstart_level_list**
- **set_softstart_mode**
- **set_softstart_mode_list**

(1) In keinem der Kompatibilitätsmodi. Diese konvertieren automatisch Parameter in das RTC4/5 Format und haben keinen Einfluss darauf, welche Befehle unterstützt werden oder nicht.

(2) Siehe auch Kapitel 10.4 "Nicht mehr unterstützte RTC4-Befehle", Seite 963.

(3) Siehe auch Kapitel 10.5 "Nicht mehr unterstützte RTC5-Befehle", Seite 966.

7.4.8 Pulse-Picking-Laser-Modus

Der Pulse-Picking-Laser-Modus kann mit **set_pulse_picking** oder **set_pulse_picking_list** eingestellt werden.

Das Zeitdiagramm, siehe Abbildung 58, zeigt die zugehörigen zeigt die zugehörigen Signale am Beispiel eines isolierten Mark-Befehls.

Für den “Laser active”-Betrieb gilt:

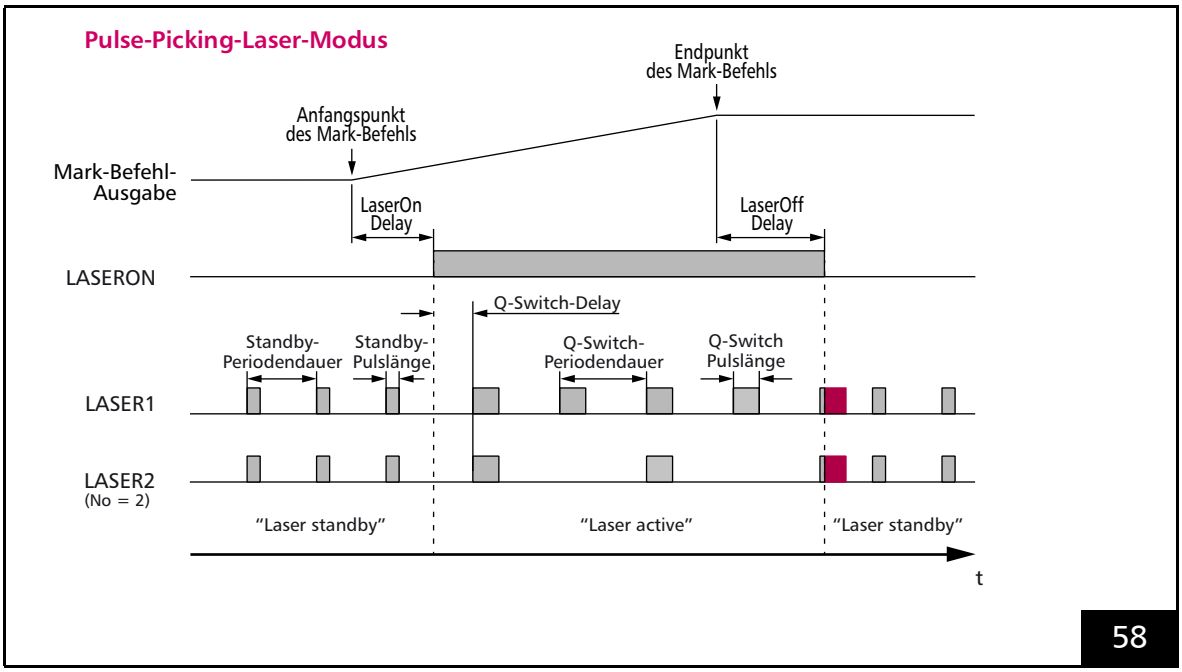
- Das LASERON-Signal wird eingeschaltet
- Am LASER1-Ausgang wird ein Modulationssignal ausgegeben
  - Dessen Pulslänge und Periodendauer kann mit **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** eingestellt werden
- Am LASER2-Ausgang wird jeder `No`-te Puls der Signale am LASER1-Ausgang (phasengleich) ausgegeben
  - `No` wird mit **set_pulse_picking** oder **set_pulse_picking_list** eingestellt

Für den “Laser standby”-Betrieb gilt:

- Das LASERON-Signal wird ausgeschaltet
- Am LASER1-Ausgang und LASER2-Ausgang werden phasengleich Standby-Pulse ausgegeben
  - Deren Pulslänge und Periodendauer können mit **set_standby** oder **set_standby_list** eingestellt werden
  - Standby-Pulse können nicht “pulse-gepickt” werden

Hinweise

- Mit **set_laser_control**(Bit #7 = 1) kann das Pulse-Picking-Signal an LASER2 auf eine vom LASER1-Signal unabhängige konstante Länge eingestellt werden. Diese Länge wird mit **set_pulse_picking_length** festgelegt.
- **set_pulse_picking** und **set_pulse_picking_list** überschreiben einen zuvor mit **set_laser_mode** eingestellten Laser-Mode. Umgekehrt schaltet **set_laser_mode** den Pulse-Picking-Laser-Modus aus.
- Für die LASER1-Signale muss die halbe Periodendauer angegeben werden.
- Ein Q-Switch-Delay ist wirksam, siehe auch Abschnitt “YAG-Modus-Unterschiede”, Seite 200.
- Ein FirstPulseKiller-Signal, Seite 373 wird nicht ausgegeben.
- Die Einstellungsreihenfolge der LASER1-Signale und des Pulse-Picking-Laser-Modus ist unerheblich.



Zeitdiagramm der Signale im Pulse-Picking-Laser-Modus (mit HIGH-aktiven Lasersteuersignalen und `No` = 2).
Beispiel: isolierter Mark-Befehl.

7.4.9 “Automatische Laser-Steuerung“

Mit **set_auto_laser_control** kann eine automatische Nachregelung der Signale für den “Laser active”-Betrieb – auch dynamisch während der Ausführung von Markier-Befehlen – erzielt werden:

- Parameter `Ctrl` bestimmt den Ausgabeport
- Parameter `Value` bestimmt den Ausgabewert für 100% Leistung bei Soll-Markiergeschwindigkeit am Bildfeld-Mittelpunkt
- Die weiteren Parameter siehe Befehlsbeschreibung

Damit können Variationen im Laser-Energieeintrag kompensiert werden, die hervorgerufen werden durch:

- Sich verändernde Leistungsdichte
- Sich verändernde Spot-Größe
- Sich verändernde Bearbeitungsgeschwindigkeit

Zur “Automatische Laser-Steuerung“ können eine positions-, geschwindigkeits- oder vektorkontrollierte Korrektur der Lasersteuersignale aktiviert und kombiniert werden:

- Siehe Tabelle “Automatische Laser-Steuerung“ Typ, Seite 209

Die “Automatische Laser-Steuerung“ passt wahlweise einen der folgenden Signalparameter an:

- 12-Bit-Ausgabewert am ANALOG OUT1-Ausgang oder ANALOG OUT2-Ausgang der LASER-Buchse, siehe auch Abschnitt “12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2”, Seite 77
- Ausgabewert am 16-Bit-Digital-Ausgang der EXTENSION 1-Stiftleiste, siehe auch Abschnitt “16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang”, Seite 81
- Ausgabewert am 8-Bit-Digital-Ausgang der EXTENSION 2-Stiftleiste, siehe auch Abschnitt “8-Bit-Digital-Ausgang”, Seite 84
- Pulslänge (`PulseLength`) oder Ausgabeperiode (`HalfPeriod`) der Lasersteuersignale LASER1 und LASER2

Der automatisch angepasste Wert kann mit **set_trigger[*]** (`Signal n = 24`) aufgezeichnet werden.

Hinweise

- Die “Automatische Laser-Steuerung“ gleicht eine explizite Änderung der Markiergeschwindigkeit zwischen zwei Markier-Befehlen aufgrund eines **set_mark_speed**-Aufrufs *nicht* aus.
- Die “Automatische Laser-Steuerung“ und der Pixelmode sollten nicht gleichzeitig auf denselben Ausgabeport wirken.
- Die “Automatische Laser-Steuerung“ kann nicht mit der variablen Laser-Steuerung über die McBSP-Schnittstelle (siehe **set_multi_mcbsp_in**) kombiniert werden.

"Automatische Laser-Steuerung" Typ	Zweck
Positions-abhängige Laser-Steuerung, Seite 210	Damit können radiale Variationen des Laser-Energieeintrags während der Ausführung von Markier-Befehlen kompensiert werden. Solche Variationen werden z.B. durch einen Objektivrandabfall oder unterschiedliche Auftreffwinkel des Laserstrahls hervorgerufen.
Geschwindigkeits-abhängige Laser-Steuerung, Seite 212	Damit können solche Variationen des Laser-Energieeintrags während der Ausführung von Markier-Befehlen kompensiert werden, die aus einer ungleichmäßigen Galvanometerscanner-Bewegung (Beschleunigungsphase und Bremsphase, zeitbemessene Markier-Befehle) resultieren. Die Galvanometerscanner-Geschwindigkeit kann zusätzlich auch positionsabhängig in eine Markiergeschwindigkeit rückwärtstransformiert werden. Geschwindigkeits-abhängige Laser-Steuerung und Encodergeschwindigkeits-abhängige Laser-Steuerung können kombiniert werden, wenn sich die Galvanometerscanner und das Werkstück beide gleichzeitig bewegen.
Encodergeschwindigkeits-abhängige Laser-Steuerung, Seite 215	Damit kann der Laser-Energieeintrag basierend auf der aktuell vorliegenden Encodergeschwindigkeit gesteuert werden. Geschwindigkeits-abhängige Laser-Steuerung und Encodergeschwindigkeits-abhängige Laser-Steuerung können kombiniert werden, wenn sich die Galvanometerscanner und das Werkstück beide gleichzeitig bewegen.
Vektorkontrollierte Laser-Steuerung, Seite 217	Damit kann ein Signalparameter entlang parametrierter Mark-Vektoren oder Sprung-Vektoren (siehe Abschnitt "[*]Para[*]-Befehle", Seite 142) linear nachgeführt werden. Dieser Signalparameter kann mit der Laserleistungssteuerung kombiniert werden, wenn der Steuerparameter `Ctrl` mit `Ctrl` aus **set_auto_laser_control** übereinstimmt (der momentane Parameter `Para` ersetzt dabei dynamisch den 100%-Wert `Value` aus **set_auto_laser_control**) oder als davon unabhängige Steuerung (z.B. als Defocus).
2D-Positions-abhängige Laser-Steuerung, Seite 218	Mit der 2D-Positions-abhängige Laser-Steuerung erfolgt eine 2D-Variation des Laser-Energieeintrags basierend auf einer Tabelle mit Skalierungsfaktor-Werten.

Allgemeine Hinweise

- Jeder einzelne Beitrag sowie die Gesamtkorrektur kann einen Faktor 4.0 nicht überschreiten (Clipping(1)).
- Wenn die Laserleistung und/oder der Energieeintrag in das zu bearbeitende Material nicht streng proportional zu den Ausgangswerten des ausgewählten Signalparameters ist, kann mit **load_auto_laser_control** eine Nichtlinearitätskurve geladen werden, in der dieser (Anwendungs-spezifische) Zusammenhang definiert wird, siehe Abschnitt "Nichtlinearitätskurve laden und ermitteln", Seite 215.
- Zusätzlich kann der ausgewählte Signalparameter weder den mit **set_auto_laser_control** (Parameter `MinValue` und `MaxValue`) noch den für den jeweiligen Ausgangs-Port zulässigen Wertebereich überschreiten. Er wird entsprechend geclippt.
- Für das Lasersteuersignal wird ein entsprechender Default-Wert ausgegeben (siehe **set_port_default** oder **set_laser_off_default**), wenn:
  - die Signale für den "Laser active"-Betrieb bei aktiver "Automatische Laser-Steuerung" ausgeschaltet werden
  - die "Automatische Laser-Steuerung" selbst deaktiviert wird

  Wurde kein Default-Wert explizit festgelegt, wird der zulässige Maximalwert ausgegeben. Für die Steuerparameter `HalfPeriod` und `PulseLength` wird ersatzweise der Parameter `Value` (100%-Wert) aus **set_auto_laser_control** verwendet.
- Wurde eine "Automatische Laser-Steuerung" einmal mit **set_auto_laser_control** eingeschaltet, dann können mit **set_auto_laser_params** oder **set_auto_laser_params_list** nachträglich verändert werden:
  - der Signalparameter
  - der 100%-Wert
  - die ihm zugeordneten Grenzwerte

  Nicht nachträglich verändert werden kann der Parameter `Mode`.

Positions-abhängige Laser-Steuerung

Um die Positions-abhängige Laser-Steuerung für den bei **set_auto_laser_control** angegebenen Ausgang zu aktivieren, muss über **load_position_control** eine Benutzer-definierte Skalierungsfunktion geladen werden, siehe Abschnitt "Hinweise zum Laden der Skalierungsfunktion", Seite 210. Diese Skalierungsfunktion stellt einen Skalierungsfaktor als Funktion des Abstands zum Bildfeld-Mittelpunkt dar.

Nach **load_program_file** ist sie für alle Abstände mit "Faktor 1,0" initialisiert. Die Positions-abhängige Laser-Steuerung ist damit faktisch deaktiviert. Die Tabelle kann mit **create_dat_file** gespeichert werden.

Bei der Berechnung der Korrektur zur Positions-abhängige Laser-Steuerung werden die aktuellen kartesischen Ansteuerwerte zu Grunde gelegt:

- Einschließlich Wobbelkorrektur (#2 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189)
- Einschließlich Koordinatentransformation im virtuellen Bildfeld (#3)
- Einschließlich "Processing on the fly"-Korrektur (#4)
- Ausschließlich kopfspezifische Koordinatentransformation (#5a und #5b)
- Ausschließlich Bildfeld-Korrektur (#7)

Hinweise zum Laden der Skalierungsfunktion

- **load_position_control** lädt für die Skalierungsfunktion *Scale(Position)* eine Tabelle aus einer ASCII-Textdatei.
- Die ASCII-Textdatei kann eine oder mehrere Tabellen enthalten.(2)
- Jede Tabelle kann bis zu 50 Datenpunkte (*Position* | *Scale(Position)*) enthalten.
- Aus den Datenpunkten wird die Skalierungsfunktion *Scale(Position)* linear interpoliert.
- 2D-Positions-abhängige Laser-Steuerung und Positions-abhängige Laser-Steuerung können nicht kombiniert werden. Das Laden einer Benutzer-definierten ASCII-Textdatei für die 2D-Positions-abhängige Laser-Steuerung deaktiviert die Positions-abhängige Laser-Steuerung und umgekehrt.

(1) < DLL 612: Überlauf.

(2) Sogar eines anderen Typs, siehe Tabelle 1, Seite 156.

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Zeile beginnen:
  `[PositionCtrlTable<No>]`
  `<No>` steht für die Tabellennummer.
- Enthält die Tabelle mehrere `[PositionCtrlTable<No>]`-Einträge zur gleichen `<No>`, werden nur die Zeilen nach dem ersten Eintrag verwendet. Es werden nur die Zeilen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (*Position* | *Scale*(*Position*)) wird wie folgt definiert:
  `Position<n> = <Value>`
  `Scale<n> = <Value>`
  Dabei entspricht `<n>` dem Index des Datenpunktes ($1 \le$ `<n>` ≤ 50). Die Werte `<Value>` können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Dezimal-Trennzeichen: Punkt (.).
- Enthält die Tabelle mehrere Datenpunkte mit gleichem Index `<n>`, so wird der zuletzt gelesene verwendet.
- Enthält die Tabelle mehrere Datenpunkte zum gleichen Positionswert *Position*, so wird der Datenpunkt mit dem größten Index `<n>` verwendet. Die Gleichheit wird auf ± 0.01 geprüft.
- Der Positionswert wird radial als Abstand des zu markierenden Punktes vom Koordinaten-Mittelpunkt ($= (x^2 + y^2)^{1/2}$) als Prozentwert der halben Bildfeld-Seitenlänge angeben.
  Beispiel: $(X_{max}|0)$ entspricht 100%, $(X_{max}|Y_{max})$ entspricht $2^{1/2} \times 100\%$.
- Für `<Value>` gelten die folgenden Wertebereiche:
  $0.0 \le$ *Position* ≤ 150.0 und
  $0.0 \le$ *Scale*(*Position*) ≤ 4.0.
- Jede Anweisung muss in einer separaten Zeile stehen.
- Leerzeichen und Tabs in einer Zeile (z.B. zwischen '=' und `<Value>`) werden ignoriert.
- Leere Zeilen werden ignoriert.
- Datenpunkte mit unzulässigen Werten werden ignoriert.
- Datenpunkte, für die nicht `Position<n>` *und* `Scale<n>` zum selben Index `<n>` definiert sind, werden ignoriert.
- Das Semikolon ';' dient der Kommentareinleitung. Alle Zeichen einer Zeile, die sich hinter einem Semikolon befinden, werden ignoriert.
- Die Reihenfolge der Anweisungen für die Datenpunkte in der Tabelle kann beliebig gewählt werden.
- Die Indices für die Datenpunktpaare in der Tabelle können innerhalb [1…50] beliebig gewählt werden (die Tabelle wird automatisch nach aufsteigenden Positionswerten sortiert).
- Enthält die Tabelle *keinen* gültigen Datenpunkt, bleibt **load_position_control** wirkungslos (Rückgabewert 1 oder 13).
- Wenn ein Eintrag zu *Position* = 0.0 fehlt, wird ein solcher mit `Scale` = Min(`Scale<i>`) hinzugefügt (der kleinste vorkommende erlaubte Wert wird nach unten aufgefüllt). Analog bei *Position* = 150.0 mit Max(`Scale<i>`).
- Wenn in der ausgewählten Text-Datei nur ein einziger gültiger Datenpunkt mit `Scale<n> = S` eingetragen ist, dann wird (für den vollen Positionsbereich) die Skalierungsfunktion *Scale*(*Position*) `= S` geladen.
  Die Korrektur wirkt multiplikativ auf das Lasersteuersignal.
  Für `S` = 1.0 ist daher die ortsabhängige Korrektur ausgeschaltet. Dies kann alternativ auch erreicht werden mit `Name` = NULL in **load_position_control**.
- Die Tabelle kann mit **create_dat_file** gespeichert werden.

Geschwindigkeits-abhängige Laser-Steuerung

Bei der Aktivierung der Geschwindigkeits-abhängige Laser-Steuerung für den über **set_auto_laser_control** angegebenen Ausgabeport wird über den Parameter `Mode` ausgewählt, welche Inputparameter für die Berechnung der Korrektur herangezogen werden.

Als Bezugswert für die 100%-Geschwindigkeit gilt immer die (über **set_mark_speed** oder **set_mark_speed_ctrl**) festgelegte Soll-Markiergeschwindigkeit (deren Änderung wird durch die "Automatische Laser-Steuerung" niemals ausgeglichen).

- `Mode` = 1 ist (aus Konsistenzgründen) insbesondere für analoge Scan-Systeme vorgesehen. Als Input dient hier die aktuelle Mikroschritt-Länge pro10 μs. Diese kann infolge der 10 μs-Takt-Rundung oder bei **[*]timed[*]**-Befehlen von der Soll-Geschwindigkeit abweichen. Variationen in den Beschleunigungsphasen und Bremsphasen können nicht ausgeglichen werden.
- `Mode` = 2 ist als Basismode für iDRIVE-Scan-Systeme vorgesehen. Er kann mit weiteren speziellen Korrekturen verknüpft werden (siehe unten).
- Erweiterungen zu `Mode` = 2 (beliebig kombinierbar):
  - +0
    Basismode für iDRIVE-Scan-Systeme
  - +4
    Kombiniert die Geschwindigkeiten aus `Mode` = 2 und `Mode` = 5 zu einer Gesamtgeschwindigkeit. Die 100%-Referenzgeschwindigkeit aus `Mode` = 5 bleibt dabei unberücksichtigt. Stattdessen werden die Encodergeschwindigkeiten mit den Fly-Skalierungsfaktoren in Galvanometerscanner-Bit-Geschwindigkeiten umgerechnet und dann vektoriell zur Galvanometerscanner-Geschwindigkeit addiert. Eine entsprechende "Processing on the fly"-Session muss aktiv sein.
  - +16
    Basismode für SCANahead-Systeme
  - +32
    Ortsabhängige Korrektur der Galvanometerscanner-Geschwindigkeit (in Winkeleinheiten) in eine solche für das Bildfeld ("inverse Geschwindigkeits-Korrekturtabelle")

Hinweise

- In der Regel erfordert **set_auto_laser_control**(`Mode` = 2) eine spezielle intelliSCAN-Firmware, die eine um die Signallaufzeiten zwischen RTC6 PCIe-Karte und Scan-Kopf korrigierte Ist-Geschwindigkeit zurückliefert. Bei Fragen, ob Ihr intelliSCAN damit ausgerüstet ist oder aufrüstbar ist, wenden Sie sich an SCANLAB.
- In der Regel ist eine Kombination von Geschwindigkeits-abhängige Laser-Steuerung und einem negativen LaserOn-Delay nicht sinnvoll.

“Spot Distance Control”

Die “Spot Distance Control”-Funktionalität steht nur für SCANahead-Systeme zur Verfügung.
Siehe auch Handbuch “excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten”.

Bei **set_auto_laser_control**(`Ctrl` = 7)
(“Spot Distance Control”) werden die zeitlichen Ausgabepulse auf einen geometrisch konstanten Pulsabstand geregelt. Dieser Pulsabstand wird über **spot_distance** oder **spot_distance_ctrl** festgelegt. Die Parameter `Value`, `MinValue` und `MaxValue` haben bei der “Spot Distance Control”-Funktionalität keine Bedeutung.

Die Geschwindigkeitskorrektur erfolgt mit `Mode` = 2 + 16 + (optional) 32, (siehe oben).(1)
“Spot Distance Control” kann nicht mit `Mode` = 1, 2 oder 5 kombiniert werden.

Die Korrektur erfolgt genauer und dynamischer als die vergleichbare Regelung mit `Ctrl` = 5 (HalfPeriod). Für eine erfolgreiche Verwendung sollte ein Laser mit “Pulse-on-Demand”-Funktionalität eingesetzt werden, der Laserpulse unverzüglich ausgibt, sobald diese vom LASER1-Signal getriggert werden. Frequenzmodulierungen werden mit einer Genauigkeit von 1/64 μs für den Pulsabstand eingestellt.

Einen Vergleich eines typischen Markierergebnisses mit automatischer Laser-Steuerung (`Ctrl` = 7) und ohne automatischer Laser-Steuerung zeigt Abbildung 59. Die in Abbildung 60 gezeigte Markierung könnte alternativ zu **set_auto_laser_control** mit `Ctrl` = 7 auch mit eingeschaltetem Sky-Writing realisiert werden. Allerdings ist die Prozesszeit bei Verwendung von **set_auto_laser_control** mit `Ctrl` = 7 häufig kürzer.

Hinweise

- **set_auto_laser_control**(`Ctrl` = 7)
  (“Spot Distance Control”) kann *nicht* kombiniert werden mit:
  - Vektorkontrollierte Laser-Steuerung mit **set_vector_control** (`Ctrl` = 5) (`HalfPeriod`)
  - Laser-Mode 4(2)
  - Laser-Mode 6(3)
- **set_trigger[*]** (Signal 24) zeichnet ein internes Steuersignal auf, *nicht* `HalfPeriod`.

(1) Siehe Seite 676: “In der Folge ist **control_command** für den Anschluss für ersten Scan-Kopf gesperrt, solange diese `Mode`-Auswahl besteht (weil die aktuelle vom Scan-System übertragene Geschwindigkeit genutzt wird).”

(2) Im Laser-Mode 4 wird nur ein Standby-Signal ausgegeben, das nicht variiert werden kann.

(3) Im Laser-Mode 6 wird nur ein Standby-Signal ausgegeben, das nicht variiert werden kann.

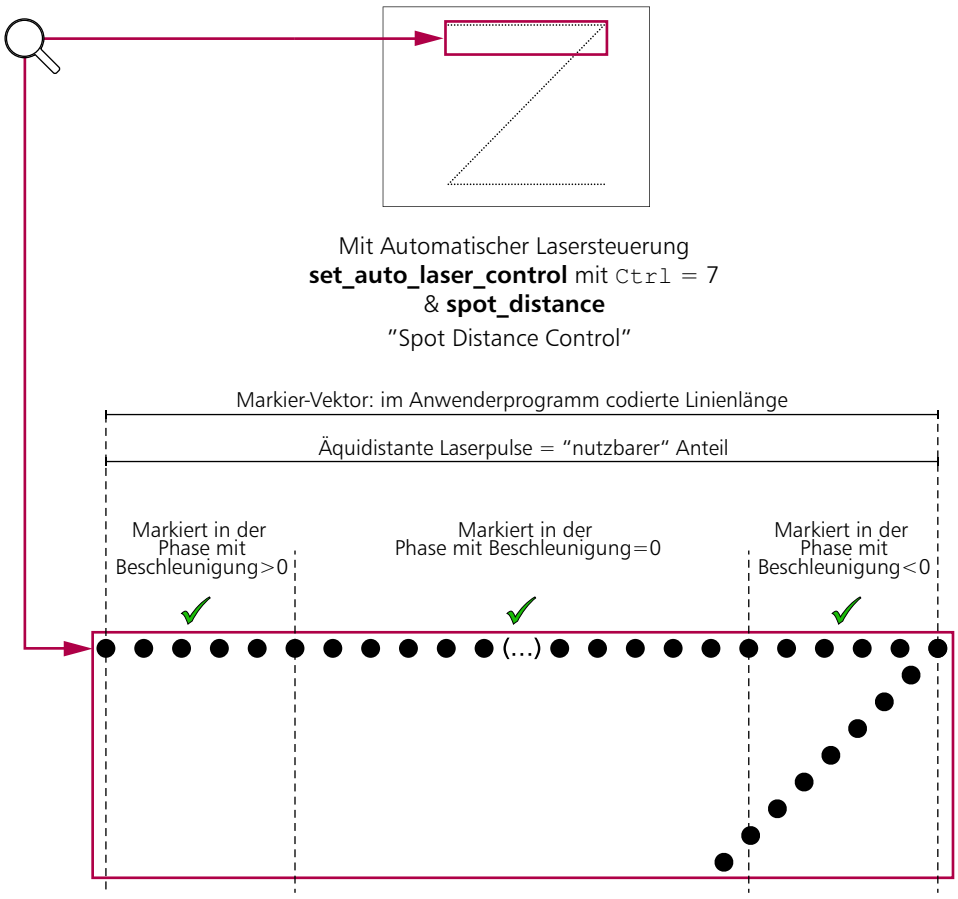


Legende

1. Mit **set_auto_laser_control** mit `Ctrl` = 7 und mit bei **spot_distance**/**spot_distance_ctrl** angegebenem äquidistant geometrischen Pulsabstand.
2. Keine "Automatische Laser-Steuerung" aktiv (Spotabstände können variieren).

59

Beispiel-Markierergebnis: "scharfe" Ecke.



60

Bei **set_auto_laser_control** mit `Ctrl` = 7 zeigt das Markierergebnis *weitgehend* äquidistante Punkte, solange sich die Beschleunigung innerhalb eines 10 μs-Takts nicht zu stark ändert.

Encodergeschwindigkeits-abhängige Laser-Steuerung

set_auto_laser_control(`Mode` = 5) ist für eine reine Encodergeschwindigkeits-abhängige Korrektur vorgesehen, wenn sich nur das Werkstück bewegt und die Galvanometerscanner (idealerweise) stillstehen.

Die 100%-Referenzgeschwindigkeit wird mit **set_encoder_speed_ctrl** oder **set_encoder_speed** festgelegt. "Processing on the fly" sollte nicht gleichzeitig aktiviert sein.

Zu einer Kombination von Galvanometerscanner-Geschwindigkeiten und Encodergeschwindigkeiten in einer "Processing on the fly"-Session siehe `Mode` = 6 = 2 + 4.

Nichtlinearitätskurve laden und ermitteln

- **load_auto_laser_control** lädt für die Nichtlinearitätskurve *Scale*(*Percent*) eine Tabelle aus einer ASCII-Textdatei.
- Die ASCII-Textdatei kann eine oder mehrere Tabellen enthalten.(1)
- Jede Tabelle kann bis zu 50 Datenpunkte *(Percent | Scale(Percent))* enthalten.
- Aus den Datenpunkten wird die Skalierungsfunktion *Scale(Percent)* linear interpoliert.

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Zeile beginnen:
  `[AutoLaserCtrlTable<No>]`
  `<No>` steht für die Tabellennummer.
- Enthält die Tabelle mehrere `[AutoLaserCtrlTable<No>]`-Einträge zur gleichen `<No>`, werden nur die Zeilen nach dem ersten Eintrag verwendet. Es werden nur die Zeilen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (*Percent* | *Scale*(*Percent*)) wird wie folgt definiert:
  `Percent<n> = <Value>`
  `Scale<n> = <Value>`
  Dabei entspricht `<n>` dem Index des Datenpunktes ($1 \le$ `<n>` ≤ 50). Die Werte `<Value>` können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Dezimal-Trennzeichen: Punkt (.).
- Enthält die Tabelle mehrere Datenpunkte mit gleichem Index `<n>`, so wird der zuletzt gelesene verwendet.
- Enthält die Tabelle mehrere Datenpunkte zum gleichen Prozentwert *Percent*, so wird der Datenpunkt mit dem größten Index `<n>` verwendet. Die Gleichheit wird auf ± 0.01 geprüft.

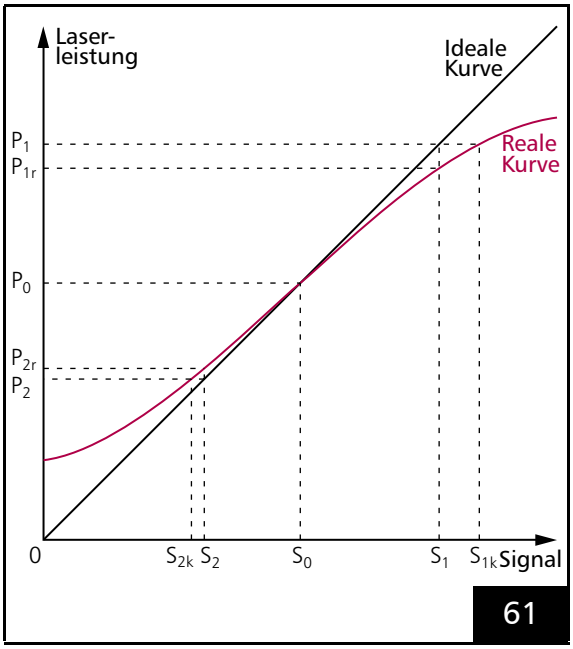
(1) Sogar eines anderen Typs, siehe Tabelle 1, Seite 156.

- Der Prozentwert ist relativ zum 100%-Wert aus **set_auto_laser_control** (Parameter `Value`) oder dynamisch aus einer Vektorkontrollierte Laser-Steuerung.
  Im folgenden Beispiel wird für das 1,5-fache des Soll-Werts ein Nichtlinearitätsfaktor von 1,2 festgelegt:
  `Percent<n> = 150`
  `Scale<n> = 1.2`
- Für `<Value>` gelten die Wertebereiche
  $0.0 \le Percent \le 400.0$ und
  $0.0 \le Scale(Percent) \le 4.0$.
- Jede Anweisung muss in einer separaten Zeile stehen.
- Leerzeichen und Tabs in einer Zeile (z.B. zwischen '=' und `<Value>`) werden ignoriert.
- Leere Zeilen werden ignoriert.
- Datenpunkte mit unzulässigen Werten werden ignoriert.
- Datenpunkte, für die nicht `Percent<n>` *und* `Scale<n>` zum selben Index `<n>` definiert sind, werden ignoriert.
- Das Semikolon ';' dient der Kommentareinleitung. Alle Zeichen einer Zeile, die sich hinter einem Semikolon befinden, werden ignoriert.
- Die Reihenfolge der Anweisungen für die Datenpunkte in der Tabelle kann beliebig gewählt werden.
- Die Indices für die Datenpunktpaare in der Tabelle können innerhalb [1…50] beliebig gewählt werden (die Tabelle wird automatisch nach aufsteigenden Prozentwerten sortiert).
- Enthält die Tabelle *keinen* gültigen Datenpunkt, bleibt **load_auto_laser_control** wirkungslos (Rückgabewert 1 oder 13).
- Wenn ein Eintrag zu *Percent* = 0.0 fehlt, wird ein solcher mit `Scale` = Min(`Scale<i>`) hinzugefügt (der kleinste vorkommende erlaubte Wert wird nach unten aufgefüllt). Analog bei *Percent* = 400.0 mit Max(`Scale<i>`).

Nach **load_program_file** ist diese Funktion für alle Prozent-Werte mit "Faktor 1,0" initialisiert, die nichtlineare Laser-Steuerung damit "deaktiviert". Dies kann alternativ auch mit `Name` = NULL in **load_auto_laser_control** erreicht werden.

Die Tabelle kann mit **create_dat_file** gespeichert werden.

Das Beispieldiagramm in Abbildung 61 zeigt, wie die Nichtlinearitätskurve ermittelt werden kann.



Laserleistungsverlauf – Beispiel zur Ermittlung der Nichtlinearitätskurve.

Die gerade Linie im Diagramm beschreibt einen idealen Zusammenhang zwischen Laserleistung und Lasersteuersignal-Parameter (der Begriff Laserleistung steht hier auch für die Pulsfrequenz = 0,5/`LaserHalfPeriod`), die gekrümmte simuliert einen realen Zusammenhang.

S_0 bezeichnet den Signalparameterwert, der als Soll-Wert definiert wurde und P_0 die zugehörige Laserleistung. Am Punkt (S_0 | P_0) (entspricht dem Datenpunkt `Percent0` = 100, `Scale0` = 1.0) sind die beiden Kurven aufeinander normiert. Die Verknüpfung einer Nichtlinearitätskurve mit einer Vektorkontrollierte Laser-Steuerung ist daher in der Regel nicht empfehlenswert.

Eine Erhöhung (Verringerung) des Signalparameters auf S_1 (S_2) führt zur idealen Laserleistung P_1 (P_2) oder zur realen Laserleistung P_{1r} (P_{2r}). Für die eigentlich gewünschte Laserleistung P_1 (P_2) ist ein korrigierter Signalparameterwert S_{1k} (S_{2k}) erforderlich. Als Datenpunkte für die Nichtlinearitätskurve sind dann die beiden folgenden Wertepaare anzugeben:

`Percent1` = $S_1/S_0 \times 100 = P_1/P_0 \times 100$
`Scale1` = S_{1k}/S_1

`Percent2` = $S_2/S_0 \times 100 = P_2/P_0 \times 100$
`Scale2` = S_{2k}/S_2

Vektorkontrollierte Laser-Steuerung

Die Vektorkontrollierte Laser-Steuerung ermöglicht es, einen Signalparameter entlang eines Mark-Vektors oder Sprung-Vektors linear zu variieren.

Zur Initialisierung der Vektorkontrollierte Laser-Steuerung muss mit **set_vector_control** festgelegt werden, welcher Signalparameter mit welchem Anfangswert variiert werden soll (Parameter `Ctrl` und `Value`).

Dann wird der Signalparameter entlang eines parametrisierten Mark-Vektor oder Sprung-Vektors linear variiert. Der Endwert ist automatisch der Startwert für einen nachfolgenden [*]para[*]-Befehl.

Hinweise

- Listenbefehle zum expliziten Ändern des Ausgabewertes des Signalparameters wirken temporär oder gar nicht.
- [*]para[*]-Befehle verwenden immer den Endwert des vorherigen [*]para[*]-Befehls als Startwert.
  Während der Abarbeitung einer Liste mit [*]para[*]-Befehlen sollten Kontrollbefehle, die denselben Ausgang beschreiben, vermieden werden.
- Wenn über **set_auto_laser_control** für `Ctrl` derselbe Ausgabeport ausgewählt ist, wirkt der Steuerparameter aus den [*]para[*]-Befehlen als 100%-Wert für die Laser-Steuerung.
  Besondere Vorsicht ist bei **set_vector_control** (`Ctrl` = 7) (Defocus) geboten: Die Signalwerte werden immer sofort eingestellt. Das kann zu harten Sprüngen(1) am vario*SCAN* führen.
- Der Sky-Writing-Modus, siehe Kapitel 7.2.4 "Sky-Writing", Seite 163, wird bei der Ausführung von [*]para[*]-Befehlen *nicht* berücksichtigt (aber auch nicht deaktiviert).

(1) Siehe Glossareintrag auf Seite 28.

2D-Positions-abhängige Laser-Steuerung

Mit der 2D-Positions-abhängige Laser-Steuerung erfolgt eine 2D-Variation des Laser-Energieeintrags basierend auf einer Tabelle mit Skalierungsfaktor-Werten.

Um die 2D-Positions-abhängige Laser-Steuerung für den bei **set_auto_laser_control** angegebenen Ausgang zu aktivieren, muss mit **load_position_control_2d_ctrl** eine Benutzer-definierte ASCII-Textdatei geladen werden, siehe:

- Anforderungen an die Benutzer-definierte ASCII-Textdatei
- Hinweise zum Laden der Benutzer-definierten ASCII-Textdatei

Bei der Berechnung der Korrektur zur 2D-Positions-abhängige Laser-Steuerung werden die aktuellen kartesischen Ansteuerwerte zu Grunde gelegt:

- Einschließlich Wobbelkorrektur (#2 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189)
- Einschließlich Koordinatentransformation im virtuellen Bildfeld (#3)
- Einschließlich "Processing on the fly"-Korrektur (#4)
- Ausschließlich kopfspezifische Koordinatentransformation (#5a und #5b)
- Ausschließlich Bildfeld-Korrektur (#6, #7)

Zur Laufzeit wird der Ausgabewert der "Automatische Laser-Steuerung" mit dem berechneten(1) Skalierungsfaktor multipliziert.

Hinweise zum Laden der Benutzer-definierten ASCII-Textdatei

- **load_position_control_2d_ctrl** lädt 257 × 257 Skalierungsfaktor-Werte aus einer Benutzer-definierten ASCII-Textdatei.
- Kommentarzeilen werden beim Einlesen ignoriert.
- Skalierungsfaktor-Werte außerhalb des Bereichs werden geclippt.
- Die RTC6-Karte prüft, ob "K = Wert" (s.u.) und der Wert aus der am Scan-Kopf 1 zugewiesenen Korrekturtabelle übereinstimmen, siehe `get_head_para( HeadNo = 1, ParaNo = 1 )`.
- 2D-Positions-abhängige Laser-Steuerung und Positions-abhängige Laser-Steuerung können nicht kombiniert werden. Das Laden einer Benutzer-definierten ASCII-Textdatei für die 2D-Positions-abhängige Laser-Steuerung deaktiviert die Positions-abhängige Laser-Steuerung und umgekehrt.

Anforderungen an die Benutzer-definierte ASCII-Textdatei

- Zeilenende-Format: \n oder \r\n
- 1 eigene Zeile (an beliebiger Stelle) mit dem Kalibrierfaktor.
  Form: "K = Wert".
  Beispiel: K = 10000.
- 257 Zeilen mit je 257 Spalten
  (= 66.049 Stützstellen) bilden die eigentliche Tabelle. Es darf nur 1 Tabelle enthalten sein(2)
  - Jede Stützstelle ist ein Skalierungsfaktor-Wert. Zulässiger Wertebereich: 0.0000… 4.0000
  - Dezimal-Trennzeichen: Punkt (.)
  - Spalten-Trennzeichen: Leerzeichen ( )
  - Zeilenhöchstlänge: 2000 Zeichen
- 0…n Kommentarzeilen. Beginn: "//" oder ";"

(1) Die Berechnung des Skalierungsfaktors erfolgt – analog zum Bildfeld-Korrektur-Algorithmus – durch bilineare Interpolation der Stützstellen aus der Benutzer-definierten ASCII-Textdatei, siehe Anforderungen an die Benutzer-definierte ASCII-Textdatei.

(2) Im Gegensatz zu den anderen Tabellentypen, siehe Tabelle 1, Seite 156.

7.4.10 Synchronisierung von RTC6-Takt und externem Taktsignal

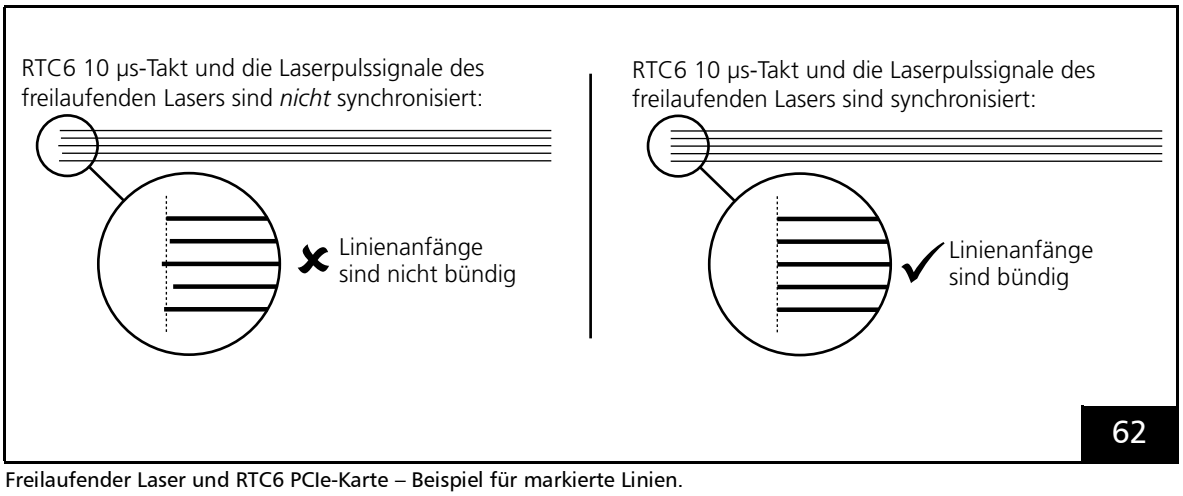
Die Laserpulssignale eines freilaufenden Lasers und die Laser-Steuerung der RTC6 PCIe-Karte können synchronisiert werden. Nichtbündige Linienanfänge kommen dadurch nicht mehr vor, siehe Abbildung 62.

Die RTC6 PCIe-Karte akzeptiert dabei das externe Taktsignal als Master-Clock. Es muss am Digital-Eingang DIGITAL IN1 der LASER-Buchse eingespeist werden.

Die Synchronisierung wird über Bit #6 von **set_laser_control** ein- und ausgeschaltet. Über Bit #5 kann eingestellt werden, ob auf die fallende oder die steigende Flanke des externen Taktes synchronisiert werden soll. Dadurch werden die RTC6-internen Lasersteuersignale automatisch auf die externen Taktsignale synchronisiert.

Hinweise

- Die Frequenz der Master-Clock muss folgende Anforderungen erfüllen:
  $f = k \times 100\ kHz$ ($k \leq 64$) und
  $f = 64\ MHz / n$ (k und n ganzzahlig).
  Damit sind nur die folgenden Frequenzen verwendbar: 100 kHz, 200 kHz, 400 kHz, 500 kHz, 800 kHz, 1 MHz, 1,6 MHz, 2 MHz, 3,2 MHz, 4 MHz, 6,4 MHz. Die maximal zulässige Abweichung der externen Frequenz gegenüber einem ganzzahligen Vielfachen von 100 kHz beträgt ±15,625 ns pro 10 µs.
- Das eingespeiste Taktsignal muss ein TTL-Signal sein. Die minimale Pulslänge oder Pulspause des Taktsignals sollte 80 ns betragen. Siehe auch Kapitel 4.6.1 "LASER-Buchse", Abschnitt "2-Bit-Digital-Eingang", Seite 77.
- Liegt bei aktivierter Synchronisierung kein gültiges Taktsignal an DIGITAL IN1 der LASER-Buchse an, erfolgt keine Synchronisierung. Die RTC6 PCIe-Karte verwendet in diesem Fall ihren internen 10 µs-Takt.
- Wird die Synchronisierung über **set_laser_control** durch Setzen von Bit #6 aktiviert, erfolgt die Synchronisierung der RTC6 PCIe-Karte schleichend auf den externen Takt (abhängig von der jeweiligen Phasenlage innerhalb von höchstens 1,2 ms).



Freilaufender Laser und RTC6 PCIe-Karte – Beispiel für markierte Linien.

7.4.11 Puls-Synchronisations-Modus

Der Puls-Synchronisations-Modus:

- Verzögert die Ausgabe eines LASER1-Pulses solange, bis ein neuer Taktsignal-Puls des Lasers am DIGITAL IN1-Pin(1) detektiert wird, siehe auch Abbildung 63
- Wird eingeschaltet mit **set_laser_pulse_sync**(`Mode` > 0)
- Wird ausgeschaltet mit **set_laser_pulse_sync**(`Mode` = 0)

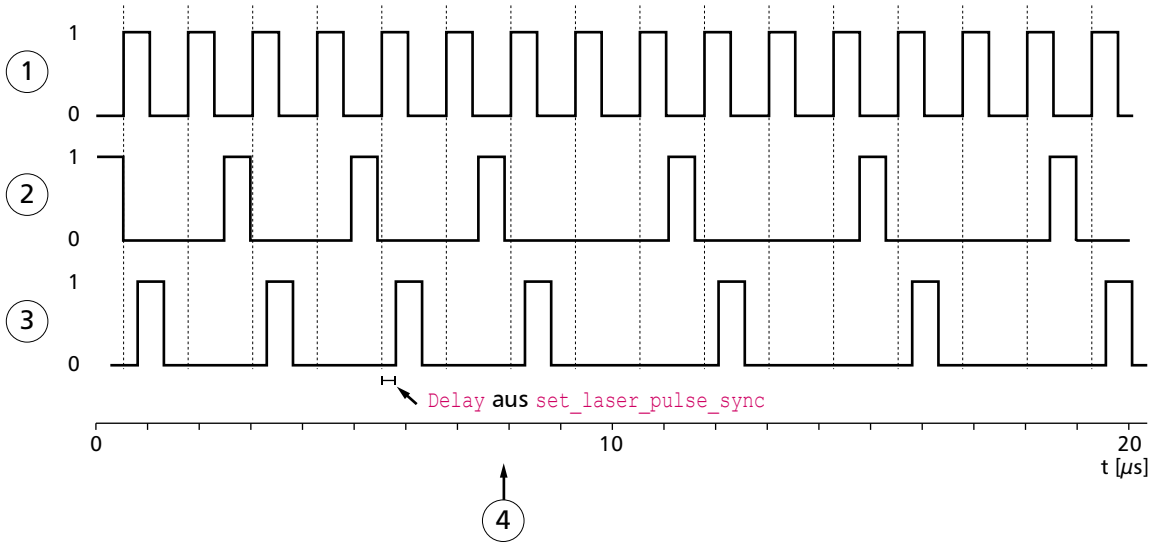
Beachten Sie Folgendes bei der Wahl der Parameterwerte:

- Mit **set_laser_pulse_sync**(`Delay`) kann das LASER1-Signal um einen festen Wert verzögert werden. Der `Delay`-Wert muss ≤ der Periode sein, die eingestellt wird mit
  - **set_laser_pulses**
  - **set_laser_timing**
  - **set_laser_pulses_ctrl**
  - **set_auto_laser_control**
- Für einen `Delay`-Wert ≥ (2 × `HalfPeriod` aus **set_laser_pulses**) gilt:
  - Der erste LASER1-Puls wird um den `Delay`-Wert verzögert
  - Der nachfolgende LASER1-Puls wird ignoriert
- Die Laser1-Pulslänge bleibt unverändert und hängt von der vorausgehenden Konfiguration ab (z.B. mit **set_laser_pulses**).

Hinweise

- Das eingespeiste Taktsignal muss ein TTL-Signal sein. Die minimale Pulslänge oder Pulspause des Taktsignals sollte 80 ns betragen. Siehe auch Kapitel 4.6.1 "LASER-Buchse", Abschnitt "2-Bit-Digital-Eingang", Seite 77.
- Über Bit #5 von **set_laser_control** kann eingestellt werden, ob auf die fallende oder die steigende Flanke des externen Taktes synchronisiert werden soll.
- Der Puls-Synchronisations-Modus kann sowohl mit der "Spot Distance Control" als auch mit den anderen "Automatische Laser-Steuerung"-Varianten wie PulseLength oder HalfPeriod kombiniert werden, siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208.
- Der Puls-Synchronisations-Modus kann mit Laser-Mode 4 oder Laser-Mode 6 kombiniert werden.
- Der Puls-Synchronisations-Modus *kann nicht* mit den Laser-Modi, die das LASER2-Signal verwenden (CO_2-Modus, Pulse-Picking-Laser-Modus), kombiniert werden, weil:
  - Er wirkt nur auf das LASER1-Signal
  - Er ändert nicht das LASERON-Signal
  - Er ändert nicht das LASER2-Signal
- Der Puls-Synchronisations-Modus *sollte nicht* mit der Zyklus-Synchronisation, siehe Kapitel 7.4.10 "Synchronisierung von RTC6-Takt und externem Taktsignal", Seite 219, kombiniert werden.

(1) An der LASER-Buchse, siehe Abbildung 17.



Legende

1. Taktsignal des Lasers@DIGITAL IN1-Pin.
2. Puls-Synchronisations-Modus ausgeschaltet: LASER1-Pulse@LASER1-Pin, unsynchronisiert mit 1.
3. Puls-Synchronisations-Modus eingeschaltet: LASER1-Pulse@LASER1-Pin, synchronisiert mit 1.
4. HalfPeriod-Änderung, z.B. wegen eines **set_laser_pulses**-Aufrufs.

63

Puls-Synchronisations-Modus. Beschreibung siehe Text.

7.5 Uhrzeit, Datum und Seriennummern markieren

Die RTC6-DLL enthält eine Reihe von Befehlen, um die aktuelle Uhrzeit, das aktuelle Datum oder die Seriennummern von Produkten zu markieren.

Bevor Uhrzeit, Datum oder Seriennummern markiert werden können, müssen die benötigten Zeichen und Zeichenfolgen als indizierte Zeichen und indizierte Zeichenfolgen definiert werden.

Für die Beschriftung von Seriennummern können separate Ziffern und Zeichenfolgen definiert werden. Siehe Abschnitt "Indizierte Zeichenfolgen für Uhrzeit, Datum und Seriennummer definieren", Seite 122.

7.5.1 Uhrzeit und Datum markieren

Mit **time_update** wird die PC-Zeit/Datum auf die RTC6 PCIe-Karte übertragen. Das ist nach jedem PC-Neustart erforderlich. Danach zählt die Karte (solange sie bestromt bleibt) intern Datum/Uhrzeit mit dem quarzgesteuerten 10 μs-Takt sekundengenau weiter.

Mit **time_fix**, **time_fix_f** oder **time_fix_f_off** wird die aktuelle Uhrzeit/Datum der Karte in einem Zwischenpuffer festgehalten.

Die Uhrzeit (Stunden, Minuten, Sekunden) kann mit **mark_time** oder **mark_time_abs** markiert werden, das Datum (Jahr, Monat, Tag, Wochentag) mit **mark_date** oder **mark_date_abs**.

Zur Markierung von Datum und Uhrzeit kann das gregorianische oder das julianische Datum eingestellt werden, sowie das 12- oder 24-Stunden-Format.

Zum Markieren von Haltbarkeitsdaten kann mit **time_fix_f_off** eine Vordatierung zum aktuellen Datum und zur aktuellen Uhrzeit fixiert werden.

7.5.2 Seriennummern markieren

Mit **mark_serial** und **mark_serial_abs** können bis zu 12-stellige Seriennummern markiert werden. Es kann festgelegt werden, wie führende Nullen dargestellt werden.

Die RTC6 PCIe-Karte verwaltet 4 Seriennummernsätze (jeweils Seriennummer und Schrittweite). Der Seriennummernsatz 0 wird bei der Initialisierung mit **load_program_file** ausgewählt.

Andere Seriennummernsätze müssen zuvor mit **select_serial_set** oder **select_serial_set_list** ausgewählt werden (siehe auch Hinweise unten).

Für jeden Seriennummernsatz kann mit **set_serial**, **set_serial_step** oder **set_serial_step_list** eine Start-Seriennummer (höchstens 10-stellig) und eine Schrittweite festgelegt werden. Bei der Initialisierung mit **load_program_file** werden alle Startseriennummern auf 0 und alle Schrittweiten auf 1 gesetzt.

Durch jeden Aufruf von **mark_serial** oder **mark_serial_abs** wird die aktuelle Seriennummer (noch vor der Ausführung der Seriennummernmarkierung) um die angegebene Schrittweite erhöht.

Wenn eine Seriennummer ausgelassen werden soll, kann auch eine Leermarkierung ausgeführt werden (`Digits` = 0), während der die Seriennummer um 1 (nicht um die vorgegebene Schrittweite) erhöht wird.

Hinweise

- Der mit **mark_serial** und **mark_serial_abs** zu markierende Seriennummernsatz kann ausschließlich mit dem Listenbefehl **select_serial_set_list** ausgewählt werden. Sowohl **mark_serial** und **mark_serial_abs**, als auch **set_serial_step_list** wirken immer auf den zuletzt mit **select_serial_set_list** ausgewählten Seriennummernsatz.
- Die Nummer des zuletzt mit **select_serial_set_list** ausgewählten Seriennummernsatzes und die aktuelle Seriennummer dieses Seriennummernsatzes kann mit dem Kontrollbefehl **get_list_serial** ausgelesen werden. Damit kann u.a. auch

festgestellt werden, ob nach einem Listenabbruch ohne Fortsetzung die aktuelle Nummer bereits hochgezählt wurde oder noch nicht.

- Mit dem Kontrollbefehl **select_serial_set** kann – unabhängig von der Auswahl mit **select_serial_set_list** – ein Seriennummernsatz für die Kontrollbefehle **set_serial_step** und **set_serial** ausgewählt werden (eine Parameteranpassung des aktuell zur Markierung verwendeten Seriennummernsatzes wird nicht verhindert). Die Kontrollbefehle **set_serial_step** und **set_serial** wirken immer auf den zuletzt mit **select_serial_set** ausgewählten Seriennummernsatz.
- **get_serial** liefert die aktuelle Seriennummer des mit **select_serial_set** ausgewählten Seriennummernsatzes (bei Verwendung mehrerer Seriennummernsätze ist dies nicht notwendigerweise die zuletzt markierte Seriennummer).
- Mit **set_max_counts** lässt sich die maximale Anzahl Externer Starts und damit die maximale Anzahl an extern gestarteten Beschriftungen einstellen. Die Anzahl bereits erfolgter Externer Starts kann mit **get_counts** ausgelesen werden. Bei Verwendung eines einzelnen Seriennummernsatzes kann mit diesen Befehlen indirekt die maximale Seriennummer eingestellt und die aktuelle Seriennummer ausgelesen werden. Bei Verwendung mehrerer Seriennummernsätze unterscheiden diese Befehle nicht zwischen den verschiedenen Seriennummernsätzen.

8 Erweiterte Funktionen zur Scan-Kopf-Steuerung und Laser-Steuerung

8.1 i*DRIVE*-Funktionen

In den SCANLAB iDRIVE-Scan-Systemen kommt die i*DRIVE*-Technologie zum Einsatz. Die i*DRIVE*-Technologie realisiert ein erweitertes Regelungs- und Ansteuerkonzept durch eine voll digitale Regelelektronik mit stark erweitertem Funktionsumfang und ein erweitertes Übertragungsprotokoll zwischen Regelelektronik und RTC-Karte (siehe auch folgender Abschnitt).

Dadurch können Benutzer eine Reihe von Einstellungen des jeweiligen Scan-Systems anpassen, z.B.:

- Einstellen, welche Daten es an die RTC-Karte übertragen soll
- Zwischen verschiedenen dynamischen Einstellungen (Tunings) auswählen
- Den PosAck-Grenzwert einstellen
- Die effektive Kalibrierung einstellen
- Das Startverhalten einstellen
- Eine Fehlerdiagnose durchführen
- Einen Funktionstest der Datenübertragung durchführen

Zu weiteren Informationen siehe das Handbuch des Scan-Systems.

i*DRIVE*-Funktionen werden mit **control_command** ausgeführt(1).

8.1.1 Übertragungsprotokoll

Die Datenübertragung zwischen RTC6 PCIe-Karte und Scan-System erfolgt nach dem SL2-100-Protokoll.

Das SL2-100-Protokoll unterstützt den vollen Funktionsumfang der i*DRIVE*-Technologie.

Bei iDRIVE-Scan-Systemen können damit z.B. die Statussignale der x-Achse und y-Achse separat und gleichzeitig ausgewertet werden. Die Statussignale der z-Achse eines 3D-Scan-Systems können gleichzeitig über den Kanal des Scan-Kopf-Anschlusses, an dem die z-Achse angeschlossen ist, ausgewertet werden.

Der XY2-100-Konverter (Zubehör) kann mit iDRIVE-Scan-Systemen verwendet werden, die mit einem Anschluss für das XY2-100-Protokoll oder XY2-100 Enhanced-Protokoll ausgestattet sind.

(1) Siehe auch Abschnitt "Verzeichnis iSCANCfg", Seite 33.

8.1.2 Datensignal-Übertragungsverhalten des Scan-Systems anpassen

Datentypen einstellen

Die digitale Regelung von iDRIVE-Scan-Systemen erlaubt eine vielfältige Rückgabe von Datensignalen des Scan-Systems an die RTC-Karte.

Jede Achse hat einen eigenen Statuskanal, auf dem alle 10 μs Daten zur RTC-Karte übertragen werden:

- STATUS-Kanal
  - ist für die x-Achse (Galvanometerscanner 2) vorgesehen
- STATUS1-Kanal
  - ist für die y-Achse (Galvanometerscanner 1) vorgesehen

Dies ermöglicht u.a. die Überwachung der Ist-Werte der Galvanometerscanner während einer Anwendung oder eine umfangreiche Fehler-Analyse im Falle einer Betriebsstörung.

Mit **control_command**(`Data` = 05nn$_H$) kann eingestellt werden, welche Daten das Scan-System an die RTC-Karte übertragen soll. Die verfügbaren Datentypen werden ausführlich im Handbuch des jeweiligen Scan-Systems beschrieben, sowie (auszugsweise) in Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083.
Der eingestellte Datentyp wird nur so lange übertragen, bis ein anderer eingestellt wird.

Nach jedem Neustart oder Reset sendet das Scan-System (nach Ablauf der Initialisierung) über alle Kanäle das XY2-100-Statuswort.

Daten auslesen

Die von der RTC6 PCIe-Karte empfangenen Daten können jederzeit:

- Asynchron mit **get_value**, **get_values** oder **get_head_status** ausgelesen werden
- Synchron mit **set_trigger[*]** aufgezeichnet werden

Siehe auch Kapitel 7.3.7 "Statusüberwachung und Diagnose", Seite 191.

Dabei ist zu berücksichtigen, dass die ersten Daten nach einem Umschalten der Datenquelle aufgrund der seriellen Übertragungszeiten erst nach einer kurzen zeitlichen Verzögerung von typischerweise 50 μs übertragen werden, siehe auch Hinweis bei **control_command**, Seite 1124.

Die Wertebereiche dieser Signale und die möglichen Statuszustände sind in Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083 beschrieben.

Hinweise

- **get_head_status** liest das XY2-100-Statuswort aus. Bei Datenübertragung gemäß SL2-100-Protokoll überträgt das Scan-System das XY2-100-Statuswort immer und parallel zu anderen Statusinformationen an die RTC6 PCIe-Karte zurück.
  Wichtig: Wenn zur Ansteuerung der XY2-100-Konverter (Zubehör) benutzt wird, dann muss explizit eingestellt werden, dass das Scan-System das XY2-100-Statuswort an die RTC6 PCIe-Karte übertragen soll. Anderenfalls liefert **get_head_status** unbrauchbare Werte.

8.1.3 Positionierung überwachen

Für manche Anwendungen ist es wichtig, die Positionierung des Scan-Systems auch während des Betriebs zu überwachen und gegebenenfalls zu dokumentieren.

Zu diesem Zweck muss die Ist-Position der Scan-Achsen mit **control_command** als vom Scan-System zu übertragender Datentyp eingestellt werden. Die zurückgegebenen Ist-Positions-Werte können dann mit **get_values** ausgelesen oder mit **set_trigger[*]** aufgezeichnet werden(1).

Wenn die zurückgegebenen Ist-Positionen der Scan-Achsen mit den kartesischen Soll-Koordinatenwerten (x, y, z) verglichen werden sollen, dann müssen diese um die auf der RTC6 PCIe-Karte durchlaufenen Korrekturen, siehe Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189, rückwärtstransformiert werden.

Die Rückwärtstransformation kann aus Laufzeitgründen nicht auf der RTC6 PCIe-Karte selbst erfolgen, sondern muss nachträglich vom PC berechnet werden. **upload_transform** überträgt dazu alle Korrektur- und Transformationseinstellungen, die dem Scan-System aktuell zugeordnet sind, von der RTC6 PCIe-Karte auf den PC. Mit **transform** kann dann ein einzelnes xy-Wertepaar oder ein einzelner z-Wert rückwärtstransformiert werden.

Mit **get_transform**/**get_transform_offset** (siehe auch **get_waveform**) kann eine ganze Serie von zuvor mit **set_trigger[*]** aufgezeichneten xy-Wertepaaren oder z-Werten rückwärtstransformiert werden.

Hinweise

- Für einige Vorwärtstransformationen ist eine Rückwärtstransformation nicht möglich:

Vorwärts-transformation	**Rückwärts-transformation möglich?**
Wobbelbewegung (#2 auf Seite 189)	nein
Globale Koordinatentransformation (#3 auf Seite 189)	nein
"Processing on the fly"-Korrektur (#4 auf Seite 189)	nein
Koordinatentransformationen (Gesamtmatrix und Offset) für die x- und y-Koordinaten (#5a und #5b auf Seite 189)	ja
Offset auf die z-Koordinate (#5b auf Seite 189)	ja
Clipping auf die Grenzen des ansteuerbaren Bildfelds (#6 auf Seite 189)	nein
2D Bildfeld-Korrektur (#7 auf Seite 189)	ja
3D-Bildfeld-Korrektur (#9 auf Seite 189)	ja
Gain- und Offset-Korrektur der automatischen Selbstkalibrierung (#11 auf Seite 189)	ja
Clipping auf maximal mögliche Ansteuerwerte	nein

- Weiterhin kann die Rückwärtstransformation nicht erfolgen, wenn bei der Vorwärtstransformation eine nicht-invertierbare Transformationsmatrix definiert wurde (Achtung: Clipping auf ±50 für jeden einzelnen Matrixkoeffizienten; siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236). Die Nicht-Invertierbarkeit wird bereits von **upload_transform** gemeldet.

(1) Aufgrund der Kommunikationslaufzeiten sind die aktuell zurückgegebenen Ist-Positionen einige Takte später als die aktuell ausgegebenen Ansteuersignale.

- Ist bei der Vorwärtstransformation ein Clipping auf die Grenzen des ansteuerbaren Bildfelds oder die maximalen Ansteuerwerte aufgetreten, dann werden bei der Rückwärtstransformation (idealerweise) die zu diesen Grenzwerten gehörenden kartesischen Koordinatenwerte statt der ursprünglichen Werte berechnet.
- Mit **get_values** können 4 beliebige Signale gleichzeitig ausgelesen werden. Beispiel:
  - die Ist-Position von Galvanometerscanner 2 über StatusAX
  - die Ist-Position von Galvanometerscanner 1 über StatusAY
  - die Ist-Position der z-Achse über StatusBX
  - ein beliebiges weiteres Signal, z.B. LaserOn

  get_value (nicht: **get_values**) eignet sich dagegen nicht zur Überwachung der xy-Positionierung, weil mit **get_value** nur ein einzelnes Signal ausgelesen werden kann und mehrere Aufrufe zwangsweise zu xyz-Werten aus unterschiedlichen Zeiten führen.
- Mit **set_trigger** können über 2 Messkanäle 2 beliebige Signale gleichzeitig aufgezeichnet werden (mit **set_trigger4** 4 Signale über 4 Messkanäle; mit **set_trigger8** 8 Signale über 8 Messkanäle).
- Die Zuordnung der über **get_values** oder **set_trigger** ausgelesenen Werte zu x, y oder z für die Rückwärtstransformation kann über den Parameter `Code` festgelegt werden bei:
  - **transform**
  - **get_transform**/**get_transform_offset**
- Außerdem kann angegeben werden, welche Teiltransformationen durchgeführt werden sollen bei:
  - **transform**
  - **get_transform**/**get_transform_offset**
- Mit **transform** können mit **get_values** ausgelesene oder beliebige synthetische Werte rückwärtstransformiert werden. Bei der Rückwärtstransformation von synthetischen Werten, die außerhalb des durch die Vorwärtstransformation möglichen Bildfelds liegen, werden u.U. aufgrund von Bereichsüberschreitungen oder anderen Fehlern nur extrapolierte Werte berechnet.
- Wenn sowohl die aufgezeichneten Werte als auch die übertragenen Transformationsdaten vom Anwenderprogramm binär gespeichert werden (siehe **get_waveform**), kann die nachträgliche Rückwärtstransformation mit **transform** (nicht **get_transform**/**get_transform_offset**) auch Offline, d.h. ohne weiteren Zugriff auf eine RTC6 PCIe-Karte ausgeführt werden.
- Wird mit **control_command** der Positionsfehler an Stelle der Ist-Position als vom Scan-System zu übertragender Datentyp ausgewählt, so kann zwar kein direkter Vergleich zwischen dem ursprünglich definierten und dem markierten Muster durchgeführt werden; es kann aber überprüft werden, ob das Scan-System die Ausgabewerte der RTC6 PCIe-Karte korrekt abarbeitet. Dies ist insbesondere dann nützlich, wenn eine Rückwärtstransformation von Ist-Positionen nicht (vollständig) möglich ist oder wenn nicht klar ist, ob eine Abweichung zwischen den rückwärtstransformierten Ist-Positionen und den ursprünglich definierten Koordinatenwerten durch einen Fehler des Scan-Systems oder durch ein Clipping während der Vorwärtstransformation verursacht ist.

8.1.4 Tuning (Dynamikeinstellung) auswählen

SCANLAB kann entsprechend den unterschiedlichen Anforderungen verschiedener Anwendungen an die Dynamik der Positionierung des Laserstrahls die Dynamikeinstellung (“Tuning”) von Scan-Systemen unterschiedlich optimieren, z.B.:

- Vektortuning
  für das Ausführen von Vektoren oder Kreisbögen mit konstanter Prozessgeschwindigkeit
- Sprungtuning
  für das Ausführen von Sprüngen mit minimaler Dauer

iDRIVE-Scan-Systeme können optional mit mehreren Tunings ausgestattet sein. Für unterschiedliche Anwendungen kann – für jede Achse separat – mit **control_command**(`Data` = 11nn$_H$) auf das jeweils geeignete Tuning umgestellt werden.

Bei Scan-Systemen, die mit einem oder mehreren Sprungtunings ausgestattet sind, kann für 2D-Sprünge zusätzlich der Sprungmodus (und damit auch ein automatischer Tuningwechsel) aktiviert werden, siehe Kapitel 8.1.5 ”Sprungmodus”, Seite 229.

Das standardmäßig eingestellte Startverhalten ist, dass das Scan-System nach jedem Neustart oder Reset mit der Tuning-Nummer 0 startet.

8.1.5 Sprungmodus

Für Anwendungen wie das Bohren von Löchern in definierten Abständen, bei denen der Laserbearbeitungsprozess eher punktuell als entlang von Linien oder Kurven ausgeführt wird, können optimierte Prozesszeiten durch Aktivierung des sogenannten "Sprungmodus" erzielt werden.

Dazu muss das Scan-System mit einem Sprungtuning ausgestattet sein, siehe auch Abschnitt "Voraussetzung und Aktivierung", Seite 230.

Funktionsweise

In der Standardeinstellung (nach **load_program_file**) werden sowohl Sprung-Befehle als auch Markier-Befehle im Vektormodus ausgeführt:

- Die Sprunglänge wird gemäß der aktuellen Sprunggeschwindigkeit in Mikroschritte zerlegt, die einzeln ausgeführt werden. Ist das Scan-System nur mit einem Sprungtuning ausgestattet, werden die einzelnen Mikroschritte mit diesem ausgeführt.
- Vor einem nachfolgenden Listenbefehl wird ein über **set_scanner_delays** festgelegtes Jump-Delay ausgeführt.

Nach Freigabe und Aktivierung des Sprungmodus mit **set_jump_mode** oder **set_jump_mode_list** wird dagegen jeder nachfolgende 2D-Sprung (s.u.) wie folgt ausgeführt:

- Die gesamte Sprunglänge des 2D-Sprungs wird als "Harter Sprung" über einen zeitbemessenen Sprung der Dauer 10 μs angesteuert. Die Zielposition wird ohne Zerlegung in Mikroschritte ausgeführt.
- Der Sprung wird mit einem Sprungtuning ausgeführt. Mit **set_jump_mode** kann das zu verwendende Sprungtuning angegeben werden. War vor dem Sprung ein anderes Tuning eingestellt, schaltet die RTC6 PCIe-Karte am Anfang des Sprungs automatisch auf das mit **set_jump_mode** angegebene um.
- Am Ende des 2D-Sprungs schaltet die RTC6 PCIe-Karte automatisch auf ein Vektortuning um (wenn das Scan-System mit einem solchen ausgestattet ist und mit **set_jump_mode** eine entsprechende Einstellung getroffen wurde).
- Am Ende des Sprungs wird außerdem ein von der Sprunglänge abhängiges Jump-Delay abgewartet. Das Jump-Delay kann mit **load_jump_table_offset** oder **set_jump_table** für die jeweilige Sprunglänge vorgegeben werden, siehe auch Abschnitt "Sprunglängen-abhängige Jump-Delays", Seite 230. Ein externes, über **set_scanner_delays** festgelegtes Jump-Delay wird dagegen *nicht* ausgeführt.

Hinweise

- Der Sprungmodus wirkt ausschließlich bei
  - **jump_abs**, **jump_rel**, **goto_xy** (*nicht* bei den entsprechenden 3D-, Para- oder zeitbemessenen Befehlen) und
  - Home-Jumps und Home-Returns (siehe **home_position**).
- Wenn bei einem 2D-Sprung nicht mindestens eine der beiden Achsen die mit **set_jump_mode** angegebene Sprunglängengrenze (Parameter `Length`) erreicht oder überschreitet, wird der Sprung auch bei freigegebenem und aktiviertem Sprungmodus im Vektormodus ausgeführt. Dadurch kann ausgenutzt werden, dass *kurze* Sprünge u.U. mit einem Vektortuning schneller ausgeführt werden als mit einem Sprungtuning. Ist jedoch kein Vektortuning installiert oder ein solches nicht angegeben, sollte der Parameter `Length` auf 0 gesetzt werden.
- Das Umschalten zwischen verschiedenen Tunings (Reglern) erfordert jeweils einen zusätzlichen 10 μs-Takt. Dies kann z.B. für reine Bohranwendungen vermieden werden, indem bei **set_jump_mode** kein Vektortuning für die Rückschaltung angegeben wird.
- Wird der Sprungmodus (mit **set_jump_mode** oder **set_jump_mode_list**) deaktiviert oder ganz unterdrückt, so werden nachfolgende Sprünge wieder im Vektormodus (zerlegt in Mikroschritte) ausgeführt (es erfolgt dann keine automatische Reglerumschaltung mehr). Dabei wird das zuletzt am Ende des Sprungmodus eingestellte Vektortuning verwendet, es sei denn, es wurde nach dem Deaktivieren ein anderes Tuning mit **control_command** ausgewählt. Außerdem wird dann auch wieder die zuletzt eingestellte Sprunggeschwindigkeit verwendet und nach dem Sprung das mit **set_scanner_delays** vorgegebene Jump-Delay abgewartet.

Voraussetzung und Aktivierung

Damit der Sprungmodus freigegeben und aktiviert werden kann, müssen die folgenden Voraussetzungen erfüllt sein:

- Mindestens einem der beiden Scan-Kopf-Anschlüsse muss eine Korrekturtabelle zugewiesen sein.
- Mindestens an einem der beiden Scan-Kopf-Anschlüsse muss ein intelli*SCAN*, intelli*cube*, intelli*WELD* oder intelli*DRILL* Scan-System angeschlossen sein.
- Ab Scan-System-Firmware ≥ 2078.
- Das angeschlossene Scan-System muss mindestens mit einem Sprungtuning ausgestattet sein. Nicht zwingend notwendig ist dagegen eine zusätzliche Ausstattung mit einem Vektortuning.
- Die bei **set_jump_mode** angegebenen Tuningnummern müssen mit den auf der Karte gespeicherten übereinstimmen.
- Die bei **set_jump_mode** angegebenen Tunings müssen den richtigen Typ – Vektortuning oder Sprungtuning – haben (der Tuningtyp ist in der Scan-System-Firmware gespeichert) und für eine schnelle Umschaltung geeignet sein.

Bevor der Sprungmodus mit **set_jump_mode_list** aktiviert werden kann, muss er mindestens einmalig mit **set_jump_mode** erfolgreich freigegeben worden sein (siehe Befehlsbeschreibung).

Der Kontrollbefehl **set_jump_mode** (nicht dagegen der Listenbefehl **set_jump_mode_list**) führt eine entsprechende Überprüfung durch, wenn der Sprungmode nicht bereits vorher schon freigegeben war.

Sprunglängen-abhängige Jump-Delays

Bei der Ausführung eines harten Sprungs (siehe "Harter Sprung") dauert es eine gewisse Zeit, bis der Scan-Kopf die vorgegebene Position eingenommen hat.

Die RTC6 PCIe-Karte berücksichtigt diese Verzögerung (auch Sprungantwort genannt) dadurch, dass am Ende des Sprungs ein Jump-Delay eingefügt wird.

Bei einer punktuellen Laserbearbeitung müssen andere Scanner-Delays nicht berücksichtigt werden, und die Laser-Delays können in der Regel auf 0 gesetzt werden.

Das exakte Sprungantwortverhalten des jeweiligen Scan-Systems (die Sprungantwortzeit in Abhängigkeit von der Sprunglänge) kann dabei in Form einer anwenderspezifischen Jump-Delay-Tabelle auf der RTC6 PCIe-Karte gespeichert werden. Bei aktiviertem Sprungmodus ermittelt die RTC6 PCIe-Karte dann bei jedem 2D-Sprung entsprechend der längeren Kante des Sprungs (also entweder gemäß x- oder y-Anteil des Sprungs) den zugehörigen Jump-Delay-Wert über die vorgegebene Jump-Delay-Tabelle.

Das Sprungantwortverhalten kann experimentell ermittelt, in Form einer Wertetabelle bereitgestellt und mit **load_jump_table_offset** auf die Karte geladen werden. Alternativ kann aber auch mit **load_jump_table_offset** (Parameter `Name` = `NULL`) eine automatische Ermittlung der Jump-Delay-Tabelle veranlasst werden. Zusätzlich kann die aktuell geladene Jump-Delay-Tabelle mit **get_jump_table** als binäre Tabelle ausgelesen und mit **set_jump_table** wieder auf die Karte geladen werden.

Da die Sprungantwortzeit (zumindest für größere Sprünge) typischerweise wurzelförmig mit der Sprunglänge skaliert, wird die interne Jump-Delay-Tabelle durch **load_program_file** wurzelförmig – mit einem Endwert von 10,24 ms für eine Sprunglänge von 2^{20} Bit – initialisiert.

Wenn nach dem Laden der Jump-Delay-Tabelle eine neue `RTC6DAT.dat`-Datei mit **create_dat_file** erzeugt wird, dann wird diese Tabelle beim nächsten **load_program_file** automatisch geladen.

Jump-Delay-Werte experimentell ermitteln

Im Manual des Scan-Systems werden üblicherweise für das jeweilige Sprungtuning Sprungantwortzeiten für ausgewählte Sprunglängen angegeben.

Für eine experimentelle Ermittlung des Sprungantwortverhaltens muss das Scan-System mit Sprüngen unterschiedlicher Sprunglängen angesteuert werden, deren resultierende Positionswerte über die Statuskanäle ausgelesen und analysiert werden können.

Die Sprünge sollten nach Aktivierung des Sprungmodus mit **jump_abs**- oder **jump_rel** ausgeführt werden. Mit **control_command** sollte zuvor der vom Scan-System zu übertragende Datentyp auf Ist-Position umgestellt werden. Letztere kann dann mit **set_trigger[*]** aufgezeichnet und per **get_waveform** ausgelesen werden.

Die ermittelten Jump-Delay-Werte müssen in einer ASCII-Textdatei bereitgestellt werden. Falls sich das Sprungantwortverhalten der beiden Achsen des Scan-Systems unterscheidet, sollte der jeweils höhere Jump-Delay-Wert der beiden Achsen in der ASCII-Textdatei angegeben werden.

Hinweise zum Laden der ermittelten Jump-Delay-Werte

- **load_jump_table_offset** lädt eine Tabelle mit Sprunglängen-Werten und Jump-Delay-Werten aus einer ASCII-Textdatei.
- Die ASCII-Textdatei kann eine oder mehrere Tabellen enthalten.(1)
- Jede Tabelle kann bis zu 50 Datenpunkte *(Length | Delay(Length))* enthalten.
- Aus den Datenpunkten wird die vollständige (interne) Jump-Delay-Tabelle *Delay*(*Length*) linear interpoliert.

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Zeile beginnen:
  `[JumpTable<No>]`
  `<No>` steht für die Tabellennummer.
- Enthält die Tabelle mehrere `[JumpTable<No>]`-Einträge zur gleichen `<No>`, werden nur die Zeilen nach dem ersten Eintrag verwendet. Es werden nur die Zeilen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (*Length* | *Delay*(*Length*)) wird wie folgt definiert:
  `Length<n> = <LengthValue>`
  `Delay<n> = <DelayValue>`
  Dabei entspricht `<n>` dem Index des Datenpunktes ($1 \leq$ `<n>` ≤ 50). Die Werte `<Value>` können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Dezimal-Trennzeichen: Punkt (.).
- Enthält die Tabelle mehrere Datenpunkte mit gleichem Index `<n>`, so wird der zuletzt gelesene verwendet, die vorherigen werden ignoriert.
- Enthält die Tabelle mehrere Datenpunkte zum gleichen Sprunglängen-Wert *Length*, so wird der Datenpunkt mit dem größten Index `<n>` verwendet, die anderen werden ignoriert. Die Gleichheit wird auf ± 0.01 geprüft.

(1) Sogar eines anderen Typs, siehe Tabelle 1, Seite 156.

- Für `<Value>` gelten die folgenden Wertebereiche:
  $0.0 \leq Length \leq 1048576.0$ und
  $0.0 \leq Delay(Length) \leq 65535.0$.
  Delay-Werte werden in Einheiten von 10 µs angegeben, Sprunglängen in Bits.
- Jede Anweisung muss in einer separaten Zeile stehen.
- Leerzeichen und Tabs in einer Zeile (z.B. zwischen '=' und `<Value>`) werden ignoriert.
- Leere Zeilen werden ignoriert.
- Datenpunkte mit unzulässigen Werten werden ignoriert.
- Datenpunkte, für die nicht `Length<n>` *und* `Delay<n>` zum selben Index `<n>` definiert sind, werden ignoriert.
- Das Semikolon ';' dient der Kommentareinleitung. Alle Zeichen einer Zeile, die sich hinter einem Semikolon befinden, werden ignoriert.
- Die Reihenfolge der Anweisungen für die Datenpunkte in der Tabelle kann beliebig gewählt werden.
- Die Indices für die Datenpunktpaare in der Tabelle können innerhalb [1…50] beliebig gewählt werden (die Tabelle wird automatisch nach aufsteigenden Positionswerten sortiert).
- Enthält die Tabelle *keinen* gültigen Datenpunkt, bleibt **load_jump_table_offset** wirkungslos (Rückgabewert 1 oder 13).
- Wenn ein Eintrag zu *Length* = 0.0 fehlt, wird ein solcher mit `Delay` = Min(`Delay<i>`) hinzugefügt (der kleinste vorkommende erlaubte Wert wird nach unten aufgefüllt). Analog bei *Length* =524288.0 mit Max(`Delay<i>`).
- Wenn in der ausgewählten Text-Datei nur ein einziger gültiger Datenpunkt mit `Delay<n> = D` eingetragen ist, dann wird (für den vollen Sprunglängenbereich) die Jump-Delay-Tabelle *Delay*(*Length*) `= D` geladen.

Jump-Delay-Tabelle automatisch ermitteln

Wenn zuvor mit **set_jump_mode**(`Flag` = 1) der Sprungmodus erfolgreich freigegeben und aktiviert wurde, kann eine automatische Ermittlung der Jump-Delay-Tabelle mit **load_jump_table_offset**(`Name` = NULL) gestartet werden.

Für die automatische Ermittlung wird die "Automatische Laser-Steuerung" deaktiviert, der vom Scan-System zu übertragende Datentyp auf Ist-Position umgestellt und das Tuning auf das mit **set_jump_mode** definierte Sprungtuning gestellt (die ursprünglichen Einstellungen werden am Ende von **load_jump_table_offset** wieder hergestellt).

Bei der automatischen Ermittlung werden mehrere diagonale Sprünge unterschiedlicher Länge ausgeführt. Für jeden Sprung wird die vom Scan-System zurückgegebene Ist-Position über **set_trigger** (nicht: **set_trigger4**, **set_trigger8**) aufgezeichnet und mit **get_waveform** ausgelesen. Die Daten werden daraufhin analysiert, zu welchem Zeitpunkt letztmals die angegebene Positionstoleranz (Parameter `PosAck`) überschritten wurde, d.h. ab wann die Ist-Position dauernd im Bereich Sprungzielposition ±`PosAck` bleibt. Dieser Wert wird dann für die jeweilige Sprunglänge als zugehöriges Jump-Delay festgehalten.

Hinweise

- Vor der automatischen Ermittlung sollte der Laser unbedingt ausgeschaltet werden.
- Die Datenaufzeichnung erfordert das Ausführen einer Liste mit insgesamt sechs Befehlen. Die Befehle werden automatisch in den RTC6-Listenspeicher geschrieben und dann abgearbeitet. Der Parameter `ListPos` gibt an, an welcher Stelle im RTC6-Listenspeicher ("Liste 1" oder "Liste 2") diese Speicherung erfolgen soll. Es sollte eine Position angegeben werden, an der eventuell zuvor eingetragene Listenbefehle schadlos überschrieben werden dürfen.
- Bei der automatischen Ermittlung werden zuerst der größte, dann zunehmend kürzere Sprünge (mit insgesamt maximal 16 unterschiedlichen Sprunglängen) ausgeführt. Die Messdauer ist für die erste (längste) Sprunglänge (Sprung über die gesamte Bilddiagonale) durch den Parameter `MaxDelay` [10 µs] festgelegt.
  `MaxDelay` sollte hinreichend, aber nicht wesentlich größer als das zum längsten Sprung gehörige Jump-Delay gewählt werden. Durch ein größeres `MaxDelay` vergrößert sich die Gesamtausführungszeit. Für `MaxDelay` = 500 beträgt die Gesamtausführungszeit der automatischen Ermittlung in der Regel wenige Sekunden.
- Zur statistischen Reduzierung des Rauschens werden für jede Sprunglänge 4 identische Sprünge ausgeführt und die Ergebnisse gemittelt. Außerdem werden bei jeder Einzelmessung die Messwerte Tiefpass-gefiltert (2-Punkt-Glättung). Dadurch kann die Positionstoleranz `PosAck` auch etwas kleiner (aber nicht deutlich) unter dem zu erwartenden Rauschen gewählt werden (zu beachten: mit XY2-100-Konverter (Zubehör) werden nur ganzzahlige Vielfache von 16 zurückgegeben. Hier ist ein Rauschen von ±3 × 16 Bit zu erwarten).
- Wird innerhalb der Messdauer das PosAck-Band nicht dauerhaft erreicht, so gilt `MaxDelay` als das ermittelte Jump-Delay.
- Ist das ermittelte Jump-Delay kleiner als `MinDelay`, so gilt `MinDelay` als das ermittelte Jump-Delay und die Messung wird abgebrochen. Kürzere Sprünge werden dann nicht mehr ausgeführt und `MinDelay` gilt dann auch für diese kürzeren Sprunglängen als das ermittelte Jump-Delay. Durch ein größeres `MinDelay` verringert sich die Gesamtausführungszeit für die automatische Ermittlung.

- Für die automatische Ermittlung kann ein Offset angegeben werden, der auf alle ermittelten Delay-Werte aufaddiert wird, bevor durch lineare Interpolation die gesamte Jump-Delay-Tabelle berechnet und auf die Karte geladen wird (die endgültigen Delays werden außerdem auf Werte zwischen 0 und 65.535 geclippt).
  Durch Angabe eines Offsets können messtechnisch bedingte Laufzeiten (z.B. durch einen XY2-100-Konverter (Zubehör), durch Tuning-Umschaltung oder Laufzeiten des vom Scan-System übertragenen Signals) bei der Ermittlung der Jump-Delay-Tabelle herausgerechnet werden oder eine gewisse Sicherheitsspanne zum Ausgleich eventueller statistischer Fluktuationen (Rauschen) hinzugefügt werden. **load_jump_table** und **load_jump_table_offset**(`Offset` = 0) sind identisch.
- Bei einer gleichzeitigen Ansteuerung zweier Scan-Systeme, sollten die Jump-Delay-Werte für beide Scan-Systeme ermittelt und nach Vergleich die des langsameren Systems verwendet werden.
- Die resultierende Tabelle kann über **get_jump_table** binär ausgelesen und mit **set_jump_table** wieder auf die Karte geladen werden.

8.1.6 PosAck-Grenzwert anpassen

control_command(`Data` = $15nn_H$) kann dazu verwendet werden, den PosAck-Grenzwert nn zu verändern. Das Startverhalten ist standardmäßig so voreingestellt, dass das Scan-System nach jedem Neustart einen Grenzwert von 0,28% des vollen Positions-Bereichs einstellt.

Wenn ein anderer Grenzwert gewünscht wird, dann muss er – für jede Achse separat – angepasst werden.

8.1.7 Effektive Kalibrierung anpassen

Mit **control_command**(`Data` = $12nn_H$) kann die Regelelektronik des Scan-Systems so eingestellt werden, dass sie die von der RTC6 PCIe-Karte empfangenen Positionswerte um einen bestimmten Faktor herunterskaliert. Der Skalierungsfaktor kann auf die Werte 1, 1/2, 1/4 oder 1/8 eingestellt werden.

Die vom Scan-System (optional) an die RTC6 PCIe-Karte zurückgesandten Positionswerte werden mit dem Kehrwert des Skalierungsfaktors rückwärtstransformiert.

Die effektive Kalibrierung dagegen kann dadurch reduziert und der Scan-Bereich – bei erhöhter Winkelauflösung – auf einen kleineren Winkelbereich eingeschränkt werden.

Das Startverhalten ist standardmäßig so voreingestellt, dass das Scan-System nach jedem Neustart mit dem Skalierungsfaktor 1 startet.

Mit **control_command**(`Data` = $053F_H$) kann der den aktuell eingestellte Skalierungsfaktor ausgelesen werden.

8.1.8 Startverhalten anpassen

In der Standard-Ausführung sind iDRIVE-Scan-Systeme wie folgt eingestellt:

- Tuning Nummer 0, siehe auch Kapitel 8.1.4 "Tuning (Dynamikeinstellung) auswählen", Seite 228
- PosAck-Grenzwert ist $B8_H$ (entspricht 0,28% des vollen Positions-Bereichs von 2^{16} Bit), siehe auch Kapitel 8.1.6 "PosAck-Grenzwert anpassen", Seite 234
- Skalierungsfaktor = 1, siehe auch Kapitel 8.1.7 "Effektive Kalibrierung anpassen", Seite 234

Diese Einstellungen können mit **control_command** geändert werden. Die geänderten Einstellungen sind zunächst temporär, können nachträglich mit **control_command**(Data = $0A00_H$) als Starteinstellungen für nachfolgende Neustarts gespeichert werden.

Das Übertragungsverhalten des Scan-Systems kann nur temporär geändert werden, siehe auch Abschnitt "Datensignal-Übertragungsverhalten des Scan-Systems anpassen", Seite 225. Nach einem Neustart überträgt das Scan-System das XY2-100-Statuswort.

8.1.9 Fehlerdiagnose und Funktionstest

Die vielfältigen Rückmeldungen von iDRIVE-Scan-Systemen können zur Diagnose im Fehlerfall verwendet werden.

Beispielsweise kann ein Ereignis-Code ausgelesen werden, der darüber Auskunft gibt, welches Ereignis für den Wechsel in einen Fehlerzustand verantwortlich war.

Damit überprüft werden kann, ob die Datenübertragung zwischen RTC6 PCIe-Karte und Scan-System funktioniert, kann mit **control_command**(Data = $21nn_H$) – für jede Achse separat – ein 8-Bit-Wert nn an das Scan-System übertragen werden. Im Folgenden wird am entsprechenden Statuskanal ein 20-Bit-Wert zurückgesendet: Bei fehlerfreier Datenübertragung sind die oberen 8 Bit des zurückgesendeten 20-Bit-Werts identisch mit dem hingesendeten 8-Bit-Wert und die nächstniedrigeren 8 Bit identisch mit dem Komplementärwert des hingesendeten 8-Bit-Wertes.

Dieser 20-Bit-Wert wird so lange zurückgegeben, bis mit **control_command**(Data = $05nn_H$) wieder ein anderer Datentyp zum Übertragen durch das Scan-System ausgewählt wird, siehe Abschnitt "Datensignal-Übertragungsverhalten des Scan-Systems anpassen", Seite 225.

Vor einem Übertragungstest kann mit **control_command**(Data = $17FF_H$) der aktuell zur Übertragung ausgewählte Datentyp zwischengespeichert werden. Nach dem Test wird mit **control_command**(Data = 1700_H) das gleiche Übertragungsverhalten wie vor dem Test wiederhergestellt.

8.2 Koordinatentransformationen

Damit das Scan-System relativ zum Bildfeld exakt ausgerichtet werden kann (oder – wenn die Option "Zweite-Scan-Kopf-Steuerung" freigeschaltet ist – *zwei* Scan-Köpfe relativ zum *gemeinsamen* Bildfeld ausgerichtet werden können), kann für alle durch Vektor-Befehle oder Bogen-Befehle definierten x- und y-Ausgabekoordinaten (x|y) (für Anschluss für ersten Scan-Kopf und Anschluss für zweiten Scan-Kopf getrennt) eine lineare Koordinatentransformation festgelegt werden:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

Die (2 x 2)-Gesamtmatrix M wird dabei von der RTC6 PCIe-Karte automatisch als Produkt einer Skalierungsmatrix M_S, einer Drehmatrix M_R und einer allgemeinen Transformationsmatrix M_T berechnet:

$$M = M_T \times M_R \times M_S$$

Die Koeffizienten der drei Matrizen (M_T, M_R und M_S) und die Offsetwerte ($x_0|y_0$) können für den Anschluss für ersten Scan-Kopf und Anschluss für zweiten Scan-Kopf unterschiedlich definiert werden.

Der Offset ($x_0|y_0$) kann mit **set_offset** oder **set_offset_list** festgelegt werden.

Bei einem 3-Achsen-Scan-System kann mit **set_offset_xyz** oder **set_offset_xyz_list** auch ein Offset z_0 für die z-Koordinate festgelegt werden (z_0 wirkt entgegengesetzt zu **set_defocus** oder **set_defocus_list**). Es gilt dann:

$$z' = z + z_0$$

Die Koeffizienten der Skalierungsmatrix M_S werden mit **set_scale** oder **set_scale_list** durch Angabe eines für beide Achsen gemeinsamen Skalierungsfaktors *k* festgelegt:

$$M_S = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

Die Koeffizienten der Drehmatrix M_R werden mit **set_angle** oder **set_angle_list** durch Angabe eines Drehwinkels α festgelegt (entsprechend der mathematischen Definition: positive Winkel drehen im Gegenuhrzeigersinn):

$$M_R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

Die Koeffizienten $m_{11}…m_{22}$ der allgemeinen Transformationsmatrix M_T werden mit **set_matrix** oder **set_matrix_list** festgelegt:

$$M_T = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

Mit der allgemeinen Transformationsmatrix M_T können als Spezialfall die beiden obigen Matrizen (M_S und M_R), aber auch weitere Transformationen zum Skalieren, Drehen, Spiegeln oder Abschrägen von Objekten definiert werden:

- Skalierung um Faktoren k_x und k_y:

$$M_T = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$$

- Drehung um den Winkel α:

$$M_T = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

  Beispiel:
  `set_matrix( 1, 0.5, -0.866, 0.866, 0.5, 1 )`
  definiert für den Anschluss für ersten Scan-Kopf eine Rotation um 60° im Gegenuhrzeigersinn um den Mittelpunkt des Bildfelds.
  Dasselbe kann auch mit `set_angle( 1, 60 )` erreicht werden.

- Spiegelung an der y-Achse (Umklappen in x-Richtung):

$$M_T = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Spiegelung an der x-Achse (Umklappen in y-Richtung):

$$M_T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- Spiegelung an der 1. Raumdiagonalen (Vertauschen von x- und y-Koordinate):

$$M_T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Querabschrägung in x-Richtung um einen Winkel α:

$$M_T = \begin{bmatrix} 1 & -\sin\alpha \\ 0 & 1 \end{bmatrix}$$

Beispiel: `set_matrix( 1, 1, -0.25, 0, 1, 0 )`

Eine über **set_matrix** oder **set_matrix_list** definierte allgemeine Transformation kann aber auch eine Kombination verschiedener Transformationen darstellen (M_T ergibt sich dann durch Multiplikation der entsprechenden Einzel-Matrizen in korrekter Reihenfolge).

Hinweise

- Die beschriebenen Koordinatentransformationen sind in erster Linie für kleinere Korrekturen zur Ausrichtung des Scan-Systems relativ zum Bildfeld vorgesehen. Die separat einstellbaren Skalierungen und Drehungen ermöglichen dabei eine flexiblere Handhabung gegenüber einer alleinigen Matrixeinstellung.
- Nach der Initialisierung mit **load_program_file** sind der Offset auf den Wert (0|0|0) und die Matrizen M_S, M_R und M_T jeweils auf die Einheitsmatrix voreingestellt.
- Bei jeder Matrix- oder Offset-Definition werden ältere Definitionen überschrieben.
- Die von der RTC6 PCIe-Karte berechnete Gesamtmatrix M ist unabhängig davon, in welcher Reihenfolge die einzelnen Transformationsmatrizen definiert werden.
- Der Wertebereich für den Skalierungsfaktor *k* der Skalierungsmatrix M_S ist [–16…+16]. Der Wertebereich für die Koeffizienten der allgemeinen Transformationsmatrix M_T ist [–50…+50]. Zusätzlich muss darauf geachtet werden, dass der Wertebereich [–50…+50] für jeden einzelnen Koeffizienten der Gesamtmatrix M nicht überschritten wird, sonst kommt es bei der Berechnung der korrigierten Koordinaten u.U. zu einem Überlauf.
- Drehungen erfolgen ausschließlich um den Mittelpunkt des Bildfelds, Spiegelungen relativ zu den Achsen.
- Bei jeder Definition kann über den Parameter `at_once` ausgewählt werden, ob sich die neuen Einstellungen unmittelbar auf die aktuelle Position auswirken (`at_once` = 1 oder 3) oder zunächst nur gesammelt und zwischengespeichert werden (`at_once` = 0 oder 2). Wann eine (angesammelte) Transformation wirksam wird, entscheidet immer der zuletzt gesendete `at_once`-Parameterwert.

- Vor Anwendung der Gesamttransformation auf die aktuelle Position werden die Signale für den "Laser active"-Betrieb bei `at_once` = 0…2 ausgeschaltet. Bei `at_once` = 3 bleiben sie unverändert.
- Mit `at_once` = 1 oder 3 werden alle bis dahin (nur) angesammelten Einstellungen sofort und gleichzeitig abgearbeitet. Dabei werden die Achsen des Scan-Systems von der aktuellen Position mit der eingestellten Sprunggeschwindigkeit in die korrigierte Position bewegt. Dies kann daher eine gewisse Anzahl von Taktzyklen benötigen.
  Das ist insbesondere dann zu beachten, wenn RTC6 PCIe-Karten Master/Slave synchronisiert sind und unterschiedliche Sprunglängen ausführen sollen. In diesem Anwendungsfall sollten die Koordinatentransformationen noch vor dem synchronisierten Start mit `at_once` = 1 ausgeführt und deren Ende abgewartet werden. Etwaige Scanner-Delays werden nicht initiiert. Für die Dauer des Sprungs auf die korrigierte Position ist der INTERNAL-BUSY-Listenausführungsstatus gesetzt.
- Mit `at_once` = 2 werden die angesammelten Einstellungen erst bei der Ausführung des nächsten **jump_abs**, **jump_rel**, **goto_xy** oder **goto_xyz** wirksam (jedoch nicht bei anderen Sprung-Befehlen wie **jump_abs_3d** oder **jump_rel_3d**), wenn nicht ein anderer Aufruf danach eine sofortige Ausführung veranlasst.
  Die Korrektur der aktuellen Ausgabeposition wird dann gemeinsam mit dem angegebenen Koordinatensprung ausgeführt. Dadurch werden überflüssige Galvanometerscanner-Bewegungen (inkl. Delays) eingespart.
  Beispiel: Die folgende Befehlsabfolge führt – nach dem ersten Sprung auf (0, 0) – für `at_once` = 1 oder `at_once` = 3 einen zweiten Sprung von (0, 0) nach (1000, 500) und einen dritten von (1000, 500) nach (0, 500) aus, dagegen für `at_once` = 2 nur einen zweiten Sprung von (0, 0) nach (0, 500).

```
jump_abs( 0, 0 );
set_offset_list( 1000, 500, at_once );
jump_abs( -1000, 0 );
```

  Hinweise zur Datenaufzeichnung:
  - Die Galvanometerscanner-Bewegung von **jump_abs** oder **jump_rel** am Listenbeginn kann mit **set_trigger** (Signaltyp 7, 8, 9) aufgezeichnet werden, wenn zuvor *keine* Koordinatentransformation mit `at_once` = 2 angewendet wurde.
  - Im Fall von Bewegungen *nach* einer Koordinatentransformation (z.B. durch **set_offset_list** mit `at_once` = 2) werden die "Sample-Werte" (**set_trigger** Signaltyp 7, 8, 9) sofort auf die Ziel-Koordinaten gesetzt und bleiben für die Dauer der Bewegung unverändert. Die hier aufgezeichnete Bewegung erscheint nicht wie in Mikroschritte zerlegt(1), sondern wie ein Harter Sprung. Die echte Galvanometerscanner-Bewegung kann mit **set_trigger**, Signaltyp 25 und 26 (transformierte Ansteuerwerte), aufgezeichnet werden.
  - Im Fall von Bewegungen, die durch Koordinatentransformation in einer Liste hervorgerufen wurden (z.B. durch **set_offset_list** mit `at_once` = 1), bleiben die "Sample-Werte" (**set_trigger** Signaltyp 7, 8, 9) für die Dauer der Bewegung unverändert. Die echte Galvanometerscanner-Bewegung kann mit **set_trigger**, Signaltyp 25 und 26 (transformierte Ansteuerwerte) aufgezeichnet werden.
- Einstellungen über Kontrollbefehle mit `at_once` = 0 werden nur gespeichert, solange keine Liste läuft. Beim Ausführungsstart oder schon laufender Liste werden die Einstellungen unmittelbar vor dem nächsten Listenbefehl wirksam.
  Einstellungen über Listenbefehle mit `at_once` = 0 werden nur gespeichert, bis sie anderweitig abgerufen werden (`at_once` > 0 oder durch einen Kontrollbefehl).
- Ist dem jeweiligen Scan-Kopf-Anschluss keine Korrekturtabelle zugewiesen, so werden die neuen Einstellungen für die Koordinatentransformationen lediglich auf der RTC6 PCIe-Karte gespeichert. Sie werden erst dann wirksam, wenn eine Korrekturtabelle zugewiesen wird.

(1) See Kapitel 7.1.2 "Zerlegung in Mikroschritte", Seite 143.

- Koordinatentransformationen werden auf alle auszugebenden Koordinaten aus allen Vektor-Befehlen (**[*]jump[*]**- oder **[*]mark[*]**-Listenbefehle, aber auch **goto_xy** oder **goto_xyz**) und Bogen-Befehlen angewendet. Siehe auch Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.
- Bei einer skalierenden Matrix ändern sich effektive Sprunggeschwindigkeit und Markiergeschwindigkeit.
- Bei 3D-Vektor-Befehlen wirkt:
  - die Transformationsmatrix nur auf die x- und y-Komponenten
  - der Offset auf alle drei Komponenten
- Um bei Koordinatentransformationen (wie Verdrehungen, Verkleinerungen oder Verschiebungen) das volle reale Bildfeld auszunutzen, kann der erweiterte Wertebereich des virtuellen Bildfelds auch ohne "Processing on the fly" verwendet werden, siehe Kapitel 7.3.3 "Virtuelles Bildfeld", Seite 177.
- Es können Koordinatentransformationen für das virtuelle Bildfeld definiert werden, siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 177.
  Siehe auch 4 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.

Hinweise für RTC4-Benutzer

- Mit RTC6 PCIe-Karten können Koordinatentransformationen nicht mehr beim Laden von Korrekturdateien mit **load_correction_file** eingestellt werden. Stattdessen werden Koordinatentransformationen für den Anschluss für ersten Scan-Kopf und Anschluss für zweiten Scan-Kopf getrennt definiert und erfüllen denselben Zweck wie diejenigen bei **load_correction_file** der RTC4.
- Die bei der RTC4 möglichen (für beide Scan-Köpfe gemeinsamen) Koordinatentransformationen vor der Zerlegung in Mikroschritte werden von RTC6 PCIe-Karten nicht unterstützt.
  Die globalen Koordinatentransformationen wirken etwas anders als die obigen Koordinatentransformationen, siehe #3, #5a und #5b in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.

8.3 Online-Positionierung

Im vorangehenden Kapitel 8.2 "Koordinatentransformationen", Seite 236 wird ausführlich beschrieben, wie ein Scan-System relativ zum Bildfeld exakt ausgerichtet werden kann, siehe auch Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 177.

Das Anwenderprogramm kann die zu verwendenden Transformationswerte beispielsweise aus einer automatischen Positionsanalyse für ein Werkstück auf einem Förderband ermitteln und dann die zugehörigen Transformationen ausführen.

Ein gut kontrollierbares Timing (im 10 μs-Takt der RTC6 PCIe-Karte) zwischen der Positionierung des Werkstücks und der Ausrichtung des Scan-Systems ist allerdings mit den in Kapitel 8.2 "Koordinatentransformationen", Seite 236 beschriebenen Kontrollbefehlen nicht leicht möglich. Für Anwendungen, bei denen solch ein Timing wichtig ist, gibt es die Befehle zur sogenannten Online-Positionierung. Hierbei können die Daten für eine Offset- und/oder Rotations-Koordinatentransformation oder einer allgemeinen Matrixoperation über die McBSP-Schnittstelle eingegeben werden.

Für unterschiedliche Anwendungsfälle werden die folgenden Varianten bereitgestellt:

- Die (bisherige) "Lokale Online-Positionierung" betrifft die Scan-System-spezifischen Koordinatentransformationen im realen Bildfeld analog zu **set_offset**, **set_angle** und **set_matrix** mit Parameter `HeadNo` = 1 oder 2 gemäß #5a und #5b in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189. Sie wird in Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240 beschrieben.
- Die "Globale Online-Positionierung" betrifft die globalen Koordinatentransformationen im virtuellen Bildfeld analog zu **set_offset**, **set_angle** und **set_matrix** jeweils mit `HeadNo` = 4 gemäß #3 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189. Sie wird in Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243 beschrieben.

Online-Positionierung kann nicht mit "Processing on the fly"-Anwendungen mit Positionsangaben kombiniert werden, jedoch mit Encoderbasierten "Processing on the fly"-Anwendungen.

Hinweise

- Da die Koordinatentransformationen #5a und #5b nach der "Processing on the fly"-Korrektur #4 berechnet werden, sind größere Bildfeld-Drehungen nicht gut mit linearen "Processing on the fly"-Korrekturen vereinbar. Für solche Fälle ist die "Globale Online-Positionierung" vorzuziehen.

8.3.1 "Lokale Online-Positionierung"

Das Einlesen der Daten für die "Lokale Online-Positionierung" über die McBSP-Schnittstelle muss mit **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** oder **set_mcbsp_matrix** (oder mit den entsprechenden Listenbefehlen) konfiguriert werden.

Mit **apply_mcbsp** oder **apply_mcbsp_list** können dann die zuletzt vollständig übertragenen Werte abgerufen und die entsprechenden Koordinatentransformationen (wie mit **set_offset** und/oder **set_angle** oder **set_matrix** oder den entsprechenden Listenbefehlen) definiert werden. Wie bei den in Kapitel 8.2 "Koordinatentransformationen", Seite 236 beschriebenen Befehlen kann dabei über einen Parameter `at_once` vorgegeben werden, wann die neudefinierte (Gesamt-) Transformation wirksam wird.

Für ein exaktes Timing kann die Ausführung desjenigen Listenbefehls, der die Transformation auslöst (je nach Parameter `at_once` ist dies **apply_mcbsp_list** oder **jump_abs** oder **jump_rel** oder auch irgendein anderer Listenbefehl), von der Eingabe eines externen Steuersignals abhängig gemacht werden (zur bedingten Befehlsausführung siehe Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320).

Zur Beschreibung der McBSP-Schnittstelle siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87.

"Lokale Online-Positionierung" konfigurieren

Mit **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** oder **set_mcbsp_matrix** (oder mit den entsprechenden Listenbefehlen) wird festgelegt, wie die an der McBSP-Schnittstelle eingegebenen Werte von der RTC6 PCIe-Karte interpretiert werden und über welche internen Speicherstellen die Werte eingelesen werden:

- Je nachdem, welche der obigen Befehle aufgerufen werden, werden die eingegebenen Werte von der RTC6 PCIe-Karte als Offset in x-Richtung und/oder y-Richtung und/oder als Rotationswerte oder als Matrixkoeffizienten interpretiert. Als Befehlsparameter muss jeweils der gewünschte Skalierungsfaktor angegeben werden (außer bei **set_mcbsp_matrix**, siehe Befehlsbeschreibung).
  Die drei Optionen **x**,**y** und **rot** können einzeln oder beliebig kombiniert gemeinsam verwendet werden. Jede dieser drei Optionen kann unabhängig von den jeweils beiden anderen durch den entsprechenden Befehl aktiviert oder deaktiviert werden. Die Option **matrix** dagegen kann nicht mit den anderen Optionen gemeinsam verwendet werden.
- Sobald eine der 4 Optionen aktiviert wird, werden alle Werte, die nachfolgend an der McBSP-Schnittstelle eingegeben werden, an die internen Speicherstellen 1 oder 2 übertragen (s.u.). Übertragene Werte können anschließend mit **read_mcbsp** ausgelesen oder mit **apply_mcbsp** oder **apply_mcbsp_list** für Koordinatentransformationen verwendet werden.

- **x oder y *oder* rot**
  Wird nur *eine* der drei Optionen aktiviert, so können x- oder y-Offset-Korrekturwerte als 32-Bit-Werte mit Vorzeichen oder Rotations-Korrekturwerte als 32-Bit-Werte ohne Vorzeichen eingegeben werden.
  Die McBSP-Eingabewerte werden an die interne Speicherstelle 1 übertragen.
- **x und y (ohne rot)**
  Werden x- und y-Offset-Korrekturen, aber keine Rotations-Korrektur aktiviert, so müssen die beiden Offset-Korrekturwerte jeweils als 16-Bit-Wert mit Vorzeichen gemeinsam über einen 32-Bit-Wert eingegeben werden (der x-Wert in den unteren 16 Bit, der y-Wert in den oberen 16 Bit).
  Die McBSP-Eingabewerte werden an die interne Speicherstelle 1 übertragen.
- **x oder y und rot**
  Wird entweder eine x- oder eine y-Offset-Korrektur zusammen mit einer Rotations-Korrektur aktiviert, so sollten die Offset- und Rotations-Korrekturwerte alternierend jeweils als 32-Bit-Werte eingegeben werden. Die McBSP-Eingabewerte werden auch alternierend an die internen Speicherstellen 1 und 2 übertragen.
  Die RTC6 PCIe-Karte identifiziert den Datentyp über das Codierungs-Bit #31 (Bit #31 = 0 für Offset-Werte, Bit #31 = 1 für Rotations-Korrekturwerte).
  Für Offset-Werte stehen effektiv 31 Bit *mit* Vorzeichen zur Verfügung, für Rotations-Korrekturwerte 31 Bit *ohne* Vorzeichen.

- **x und y und rot**
  Werden alle drei Optionen gemeinsam aktiviert, so müssen die beiden Offset-Korrekturwerte jeweils gemeinsam über einen 32-Bit-Wert eingegeben werden und sollten zu den Rotations-Korrekturwerten jeweils in einem zweiten 32-Bit-Wert alternierend übertragen werden. Die McBSP-Eingabewerte werden ebenfalls alternierend an die internen Speicherstellen 1 und 2 übertragen.
  Die Identifizierung des Datentyps erfolgt über das Codierungs-Bit #31 (Bit #31 = 0 für Offset-Werte, Bit #31 = 1 für Rotations-Korrekturwerte).
  Für Offset-Werte stehen effektiv 15 Bit mit Vorzeichen zur Verfügung (für den x-Wert in den unteren 16 Bit, für den y-Wert in den oberen 16 Bit), für Rotations-Korrekturwerte 31 Bit *ohne* Vorzeichen.
  In den beiden letzten Fällen werden die Datentypen durch das Codierungs-Bit #31 unterschieden. Die Reihenfolge der Übertragung ist dadurch zwar unerheblich; es sollten aber immer beide Datentypen übertragen werden und möglichst immer abwechselnd. Stehen zum Zeitpunkt der Anforderung durch **apply_mcbsp** (oder **apply_mcbsp_list**) zwei gleiche Datentypen auf den beiden Speicherstellen, dann wird jeweils der zuletzt übertragene Wert verwendet. Zum Zeitpunkt der Anforderung sollte die letzte Übertragung bereits beendet sein, wenn zwei identische Bit #31-Codierungen vorliegen.
- **matrix**
  Wird diese Option aktiviert, können anschließend Matrixkoeffizienten als
  32-Bit-Werte mit Vorzeichen übertragen werden. Die Indices werden dem Datenwort codiert mitgegeben (siehe Befehlsbeschreibung). Die McBSP-Eingabewerte werden an die interne Speicherstelle 1 übertragen.

Hinweise

- Eine "Lokale Online-Positionierung" kann in Kombination mit einer über Encodersignale gesteuerten "Processing on the fly"-Anwendung genutzt werden, dagegen *nicht* in Kombination mit einer über McBSP-Signale gesteuerten "Processing on the fly"-Anwendung:
  – Durch die Befehle zur Konfigurierung der "Lokale Online-Positionierung" wird eine mit **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos**, **set_mcbsp_in**, **set_mcbsp_in_list**, **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktivierte "Processing on the fly"-Korrektur automatisch deaktiviert. Nachfolgend werden McBSP-Eingabewerte an die internen Speicherstellen 1 oder 2 übertragen (s.o.) und stehen dann für eine "Lokale Online-Positionierung" zur Verfügung.
  – Umgekehrt wird durch **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos**, **set_mcbsp_in**, **set_mcbsp_in_list**, **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** eine zuvor aktivierte "Lokale Online-Positionierung" deaktiviert. Nachfolgend werden McBSP-Eingabewerte je nach Befehl an die internen Speicherstellen 0 bis 3 übertragen und stehen dann für die "Processing on the fly"-Anwendung zur Verfügung.
  – Wird die "Lokale Online-Positionierung" mit **set_mcbsp_x**, **set_mcbsp_y** oder **set_mcbsp_rot** abgeschaltet (absichtlich oder mit einem unzulässigen Skalierungsfaktor), werden die Daten zwar weiterhin auf die internen Speicherstellen 1 oder 1 und 2 übertragen (wenn weitere Daten gesendet werden), jedoch nicht mehr verwendet (**apply_mcbsp** bleibt wirkungslos).

8.3.2 “Globale Online-Positionierung“

Die “Globale Online-Positionierung“ (verfügbar ab RTC6-Software-Paket ≥ V1.6.1) muss durch einen der folgenden Befehle aktiviert werden:

- **set_mcbsp_global_matrix**
- **set_mcbsp_global_rot**
- **set_mcbsp_global_x**
- **set_mcbsp_global_y**
- **set_mcbsp_global_matrix_list**
- **set_mcbsp_global_rot_list**
- **set_mcbsp_global_x_list**
- **set_mcbsp_global_y_list**

Einmal aktiviert, werden alle anderen McBSP-Verarbeitungen deaktiviert (durch McBSP-Signale gesteuerte “Processing on the fly”-Anwendungen, “Lokale Online-Positionierung“ wie in Kapitel 8.3.1 ““Lokale Online-Positionierung““, Seite 240 beschrieben, durch **set_mcbsp_in** oder **set_multi_mcbsp_in** aktivierte Verarbeitungen) und umgekehrt.

Alle nachfolgend über McBSP übertragenen Daten werden intern wie bei der “Lokale Online-Positionierung“ behandelt (kopiert auf die internen Speicherplätze 1 und gegebenenfalls 2; lesbar durch **read_mcbsp**), jedoch anstelle der Scan-System-spezifischen Koordinatentransformationen im realen Bildfeld (siehe Kapitel 8.3.1 ““Lokale Online-Positionierung““, Seite 240) automatisch für Koordinatentransformationen im virtuellen Bildfeld verwendet.

Der Aufruf von **apply_mcbsp** oder **apply_mcbsp_list** ist nicht mehr notwendig und sogar wirkungslos.

Koordinatentransformationen im virtuellen Bildfeld (siehe auch Kapitel 8.6.4 “2D-Bewegungen ausgleichen“, Seite 261) werden automatisch angewendet, sobald *danach* eine “Processing on the fly”-Anwendung aktiviert wird.

Während einer “Processing on the fly”-Anwendung können neue Daten nur gesendet und gespeichert, aber nicht angewendet werden, siehe auch:

- `set_matrix( HeadNo = 4 )`
- `set_offset_xyz( HeadNo = 4 )`
- `set_angle( HeadNo = 4 )`

Die “Globale Online-Positionierung“ ist kompatibel mit “Processing on the fly”-Anwendungen, die durch Encodersignale gesteuert werden.

Achtung!

- Der zuletzt übertragene Datenwert wird sofort entsprechend dem aktuellen “Globale Online-Positionierung“-Modus verwendet. Stellen Sie daher sicher, dass nach dem Wechsel dieses Modus und vor der Anwendung der Daten korrekte Werte übertragen werden, *bevor* Sie eine “Processing on the fly”-Session starten.
- Beispiel: **set_mcbsp_global_x** und **set_mcbsp_global_y** fassen die xy-Offsets im unteren und oberen Halbwort der übertragenen Daten zusammen. **set_mcbsp_global_y**`( Scale = 0.0 )` deaktiviert den y-Offset und der zuletzt gesendete Datenwert wird insgesamt als x-Offset verwendet. Achten Sie darauf, dass Sie wieder den korrekten x-Offset übertragen.

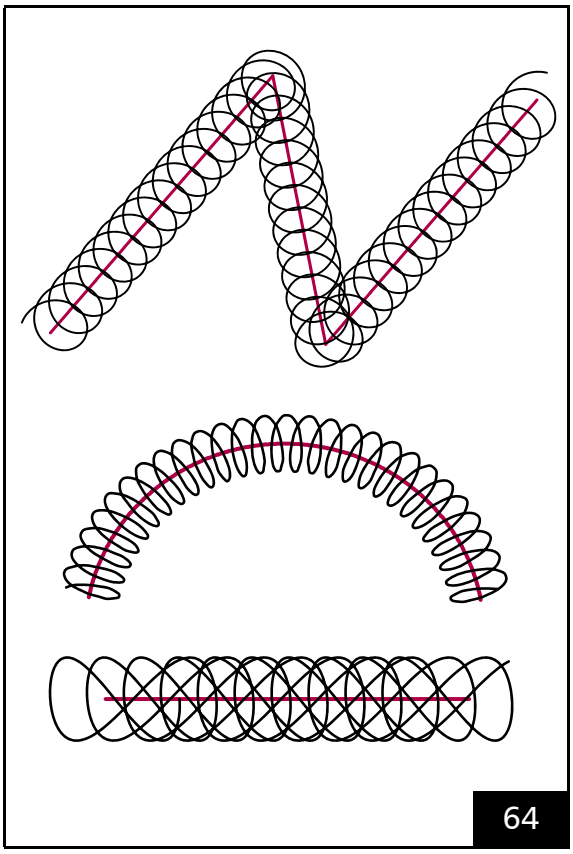
“Globale Online-Positionierung“ konfigurieren

Die “Globale Online-Positionierung“ wird wie die “Lokale Online-Positionierung“ konfiguriert, siehe Kapitel 8.3.1 ““Lokale Online-Positionierung““, Seite 240.

8.4 Wobbelmode

Der Wobbelmode ermöglicht eine Variation der Linienstrichbreite für eine Lasermarkierung.

Dazu wird die gewöhnliche, lineare Bewegung der Ausgabeposition z.B. mit einer ellipsenförmigen Bewegung kombiniert. Der Laserstrahl wird also spiralförmig durch das Bildfeld geführt, siehe Abbildung 64. Alternativ kann auch die Bewegung mit einer stehenden oder liegenden Acht kombiniert werden.



Funktionsweise der Wobbelmodes. Oben: kreisförmiger Wobbel. Mitte: elliptischer Wobbel. Unten: achtförmiger Wobbel (liegende Acht).

Durch Wahl geeigneter Werte für die transversale und longitudinale Amplitude und die Frequenz der Wobbelbewegung erhält man auf diese Weise eine Linie mit vergrößerter Strichbreite. Mit einer Achterbewegung kann je nach eingestellten Parametern eine stärkere Bearbeitung in der Linienmitte realisiert werden. Wenn transversale und longitudinale Amplituden identisch gewählt werden, bleibt die Wobbelfigur fest im Raum stehen, ansonsten folgt sie der momentanen Bewegungsrichtung. Liegt keine Bewegungsrichtung vor, wird keine Wobbelbewegung ausgeführt.

Bei einer "Processing on the fly"-Korrektur über die McBSP-Schnittstelle oder die Encoderschnittstelle, bei der der Scan-Kopf lediglich die Differenz der externen Ist-Bewegung zur gewünschten Gesamt-Bewegung auskorrigiert, siehe Kapitel 8.6.2 "Lineare Bewegungen ausgleichen", Seite 255, kann es insbesondere bei exakter externer Bahnbewegung auch zu einem kleinen Jitter in der Galvanometerscanner-Bewegungsrichtung kommen. Dabei jittert auch die an die Bewegungsrichtung angehängte Wobbelbewegung entsprechend mit. Dieser Jitter kann vermieden werden, indem der Wobbelfigur mit dem Listenbefehl **set_wobbel_direction** anstelle der momentanen Bewegungsrichtung eine konstante Bewegungsrichtung aufgeprägt wird. Das ist auch insbesondere dann wichtig, wenn die komplette Translationsbewegung außerhalb erfolgt und die Galvanometerscanner nur noch die eigentliche Wobbelbewegung auszuführen haben.

Nach **set_wobbel** oder **set_wobbel_mode** wird der Wobbel-Startpunkt immer auf denselben Wert relativ zum Vektor/Bogen-Startpunkt oder dessen Richtung gesetzt. Die Wobbelphase wird dann sowohl innerhalb eines ununterbrochenen Polygonzugs als auch nach Unterbrechungen (z.B. durch einen Sprung-Befehl) kontinuierlich fortgesetzt, bis **set_wobbel** oder **set_wobbel_mode** erneut aufgerufen wird.

Der Wobbelmode kann nicht kombiniert werden mit:

- Sky-Writing
- Pixelausgabemodus
- Jumps(1)
- **laser_on_list**

Weitere Details finden Sie unter **set_wobbel** und **set_wobbel_mode**.

(1) Siehe auch **set_wobbel_mode_phase**.

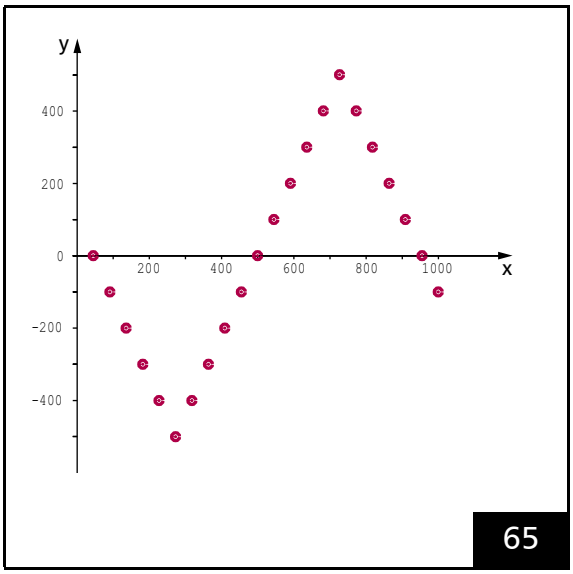
Für optimale Markierergebnisse müssen Wobbelfrequenz und Markiergeschwindigkeit aufeinander abgestimmt sein, siehe auch Kapitel 8.4.1 "Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte", Seite 246.

Für viele Schweißanwendungen führt die vorgegebene Auswahl an klassischen Wobbelfiguren (Kreis, Ellipse, Sinus, Acht) nicht zu einem optimalen Ergebnis im Bereich der Schweißnaht. Beispielsweise erfolgen die Bewegungen parallel zur Translationsbewegung mit erhöhter Geschwindigkeit, und Bewegungen in entgegengesetzter Richtung mit erniedrigter Geschwindigkeit.

Mit **set_wobbel_vector** kann das Anwendungsprogramm eine individuell geformte Wobbelfigur definieren, bestehend aus maximal 1023 stückweise linearen Abschnitten, und gleichzeitig auch eine Variation der Laserleistung entlang der Figur festlegen (siehe auch **set_wobbel_control**). Mit **create_dat_file** wird diese "Frei definierbare Wobbelfigur" gespeichert, siehe Hinweis auf Seite 381.

Jedoch kann **set_wobbel_vector** nicht mit der automatischen Laser-Steuerung oder der Vektorkontrollierte Laser-Steuerung kombiniert werden.

Die momentane Wobbelauslenkung aus **set_wobbel** oder **set_wobbel_mode** kann mit **set_trigger[*]** (Signal 53) aufgezeichnet werden. Das Format der Daten ist ((Transversal << 16) + Longitudinal).



Siehe Abschnitt "Beispiel-Code (C++)", Seite 245.

Beispiel-Code (C++)

Der folgende C++-Quellcode zeigt beispielhaft ein Zickzack-Muster, siehe Abbildung 65. Der Code muss in ein Anwenderprogramm eingebunden werden, siehe Kapitel 6.2.5 "Beispiel-Code (C)", Seite 103.

```
// Zigzag pattern

// Transversal micro step
   const double dTrans( 100.0 );
// Longitudinal micro step
   const double dLong( 0.0 );

// Number of steps
   const uint16 Period( 5 );

// Start a new wobbel shape
   set_wobbel_vector( 0, 0, 0, 0);

// 1st wobbel vector
   set_wobbel_vector( dTrans, dLong,
                      Period, 0.0 );
// 2nd wobbel vector
   set_wobbel_vector( -dTrans, dLong,
                      Period * 2, 0.0 );
// 3rd wobbel vector
   set_wobbel_vector( dTrans, dLong,
                      Period, 0.0 );
// if laser power variation is needed,
include here
// set_wobbel_control( Ctrl, Value,
                       MinValue, MaxValue );

// Activate freely definable wobbel shapes
   set_wobbel_mode( 100, 0, 100, 2 );

// Mark a vector
   timed_mark_rel( 1000, 0, 22*10 );
```

8.4.1 Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte

"Klassische" Wobbelfiguren

"Klassische" Wobbelfiguren werden mit **set_wobbel_mode** oder **set_wobbel** definiert. Weisen Sie den Parametern `Transversal`, `Longitudinal` und `Freq` nur solche Werte zu, die der Hardware auch angemessen sind.

Achtung!

- Zu große Werte stellen eine Ansteuersituation dar, die zu sehr hoher Abwärme führt. Dadurch können bereits im Kurzzeitbetrieb die Galvanometerscanner und Regler-/Verstärkerkarten überhitzen und sogar dauerhaft beschädigt werden (Überlast!). Berücksichtigen Sie die maximal mögliche Dynamik des Scan-Kopfs und des Lasers.
- Bei zu hohen Frequenzwerten können die Galvanometerscanner der Sollkurve gegebenenfalls nur noch eingeschränkt folgen. Das kann zu unerwarteten Markierungsergebnissen führen.

Faustregeln zur Abschätzung von angemessenen Höchstwerten:

(1) Höchstfrequenz: $F = 1/(10 \times S)$
mit S = Schleppverzug(1)

(2) Wobbelamplitude(2): $A = S \times V$
mit V = typische Positioniergeschwindigkeit(1)

Hinweise

- Beachten Sie auch die Bahngeschwindigkeit auf der Wobbelfigur. Bei kreisförmigen Wobbelfiguren errechnet sich diese wie folgt:

(3) Bahngeschwindigkeit: $V_{Bahn} = 2 \times \pi \times A \times F$.

- Stellen Sie sicher, dass die verwendeten Werte in Kombination mit der Trajektoriengeschwindigkeit für einen schadlosen Langzeitbetrieb geeignet sind (Prozesssicherheit!).
- Kontrollieren Sie schon während einer Testphase den Temperaturstatus (siehe **get_head_status**), um mögliche Überlastungssituationen frühzeitig erkennen zu können. Bei iDRIVE-Scan-Systeme können Sie dazu auch die aktuellen Temperaturen der Galvanometerscanner und/oder die Verstärkerkarten auslesen (siehe **control_command**).

Anwendungsbeispiel

Systemspezifikation

- Schleppverzug S = 0,33 ms.
- Kalibrierung $\pm 0{,}349\ rad_{optisch}$ ($= 10°_{mechanisch}$) bei ±503.316 Bit.
  - Das ist eine Winkelansteuerung WA $\approx 1{,}44 \times 10^6$ Bit/$rad_{optisch}$ (≈ 50.300 Bit/$°_{mechanisch}$).
- Kalibrierfaktor(3) K = 5.000 Bit/mm.
- Typische Positioniergeschwindigkeit $V_{rad} = 100\ rad_{optisch}/s$.
  - Umgerechnet in Ansteuerwerte entspricht dies einer typischen Positioniergeschwindigkeit von $V = V_{rad} \times WA \approx 1{,}44 \times 10^8$ Bit/s = 1.440 Bit/10 µs.
  - Im Bildfeld entspricht das einer typischen Positioniergeschwindigkeit von V/K =28,8 m/s.

(1) Zu finden bei den technischen Spezifikationen im Scan-Kopf Handbuch.

(2) Bei der sich aus (1) ergebenden Höchstfrequenz

(3) Zu entnehmen aus der `*_ReadMe.txt`-Datei zur Korrekturdatei oder **get_table_para**.

Wobbelparameter

- Höchstfrequenz nach Faustregel 1:
  $F = 1/(10 \times 0{,}33 \times 10^{-3}\ s) \approx 300$ Hz.
- Wobbelamplitude nach Faustregel 2:
  $A = 0{,}33 \times 10^{-3}\ s \times 1{,}44 \times 10^{8}$ Bit/s
  $\approx$ 47.500 Bit.
  - Im Bildfeld entspricht das einer Wobbelamplitude von A/K $\approx$ 9,5 mm.
- Bei kreisförmigem Wobbel ist die Bahngeschwindigkeit nach Faustregel 3:
  $V_{Bahn} \approx 895$ Bit/10 μs.
- Die *maximale* Markiergeschwindigkeit ergibt sich aus V_{Bahn} /K (in diesem Beispiel also 89.500.000 Bit/s / 5.000.000 Bit/m $\approx$ 17,9 m/s. Beachten Sie, dass dieser Wert nur eine grobe Schätzung und darüber hinaus über die (nicht-lineare) Korrekturdatei auch noch ortsabhängig ist.
  Die für Wobbel allein (!) errechnete Bahngeschwindigkeit
  - sollte niedriger als die spezifizierte, maximale Positioniergeschwindigkeit sein,
  - kann aber u.U. um einiges höher als die *typische* Markiergeschwindigkeit sein.

  Stellen Sie sicher, dass Ihre Werte für den Betrieb tauglich sind (Temperaturstatus etc., wie oben).

"Frei definierbare Wobbelfiguren"

Achten Sie bei der Definition von "Frei definierbaren Wobbelfiguren" (siehe **set_wobbel_vector**) auf die Dynamik des Scan-Kopfs.

- Die maximale Wiederholrate für "Frei definierbare Wobbelfiguren" entspricht der mit Faustregel 1, Seite 246, abgeschätzten maximalen Wobbelfrequenz.
- Die Summe aus Bahngeschwindigkeit und der Geschwindigkeit auf der frei definierten Wobbelfigur sollte die maximale Positioniergeschwindigkeit nicht überschreiten.
- Die Beschleunigung sollte den Wert 2,5 V/S (mit V = typische Positioniergeschwindigkeit und S= Schleppverzug) nicht wesentlich überschreiten.
- Beachten Sie auch bei "Frei definierbaren Wobbelfiguren" die Erwärmung des Scan-Systems, siehe Sicherheitshinweis auf Seite 246, und Faustregel 3.

8.5 2D-Scan-Systeme und 3D Scan-Systeme steuern

8.5.1 2D-Scan-Systeme

1 RTC6 PCIe-Karte kann 1…2 2D-Scan-Systeme steuern.

Das zweite 2D-Scan-System erfordert die Option "Zweite-Scan-Kopf-Steuerung", Seite 39.

Beim Steuern von 2 2D-Scan-Systemen kann an beiden jeweils eine eigene Bildfeld-Korrektur durchgeführt werden.

Mit **select_cor_table** oder **select_cor_table_list** werden die zwei Korrekturtabellen dem jeweiligen Scan-Kopf-Anschluss, siehe auch Abschnitt "2. SCANHEAD-Stiftleiste", Seite 70, zugewiesen.

Zwei Korrekturtabellen in einer Doppel-Scan-System-Konfiguration verwenden

(1) Laden Sie die 2D-Korrekturdateien jeweils mit `load_correction_file(Name, n, 2)` (`n` = 1…8).
Siehe auch Kapitel 8.5.3 "Mehrere Korrekturtabellen verwenden", Seite 253.

(2) Weisen Sie mit `select_cor_table(HeadA, HeadB)` jeweils eine der geladenen 2D-Korrekturtabellen dem ersten und dem zweiten Scan-System zu (`HeadA` und `HeadB` = 1…8).

(3) Geben Sie mit **set_offset**, **set_scale**, **set_angle** oder den entsprechenden Listenbefehlen Offset, Skalierungsfaktor und Drehung an, um die beiden Scan-Systeme und die beiden Bildfelder exakt aufeinander auszurichten.

Hinweise

- Wenn Sie einen der Scan-Kopf-Anschlüsse nicht benutzen: weisen Sie diesem die Korrekturtabelle #0 zu.
- Die Standardeinstellung für **select_cor_table** und **select_cor_table_list** ist `(1,0)`:
  - für den Anschluss für ersten Scan-Kopf wird Korrekturtabelle #1 verwendet
  - am Anschluss für zweiten Scan-Kopf erfolgen *keine Positions-Ausgaben*
- Die RTC6 PCIe-Karte kehrt nach jedem **load_program_file** zur Standardeinstellung zurück. Wenn eine andere Einstellung verwendet werden soll, muss **select_cor_table** oder **select_cor_table_list** erneut aufgerufen werden.
- Den Scan-Kopf-Anschlüssen können *nicht* gleichzeitig zugewiesen werden:
  - zwei 3D-Korrekturtabellen
  - eine 2D-Korrekturtabelle *und* eine 3D-Korrekturtabelle

8.5.2 3D-Scan-Systeme

Mit einer RTC6 PCIe-Karte kann auch ein 3-Achsen-Scan-System[1] gesteuert werden. Das erfordert die Option "3D", Seite 39.

Vorgesehene Verwendung

Mit 3-Achsen-Scan-Systemen kann der Laserfokus innerhalb eines ebenen Bildfelds positioniert werden, ohne dass dafür ein Flachfeld-Objektiv gebraucht wird. Sie kommen daher häufig in Anwendungen zum Einsatz, für die Flachfeld-Objektive nicht verfügbar sind.

Andererseits können 3-Achsen-Scan-Systeme auch als 3D-Strahlablenksysteme genutzt werden. Hier wird der Laserfokus entlang der Kontur des zu bearbeitenden Werkstücks geführt. Dies ermöglicht die Bearbeitung von Werkstücken in 3 Dimensionen.

SCANLAB bietet dynamische Fokussiereinheiten[2] an, mit denen xy-Scan-Systeme zu 3-Achsen-Scan-Systeme erweitert werden können.

Achtung!

- Die Standard-RTC6-DLL (`RTC6DLL.dll`) unterstützt alle RTC6-Befehle für den Betrieb eines 3-Achsen-Scan-Systems.
  Allerdings steht die volle Funktionalität der RTC6-Befehle – dass die z-Koordinaten *ausgegeben* werden – nur mit freigeschalteter Option "3D", Seite 39, zur Verfügung.

Anschluss und Initialisierung

Damit mit einer RTC6 PCIe-Karte ein 3-Achsen-Scan-System angesteuert werden kann:

- Die Option "3D", Seite 39, der RTC6 PCIe-Karte muss freigeschaltet sein (das kann mit **get_rtc_version** überprüft werden)
- Das xy-Scan-System muss am Anschluss für ersten Scan-Kopf angeschlossen sein
- Die z-Achse muss am Anschluss für zweiten Scan-Kopf angeschlossen sein
- Eine 3D-Korrekturtabelle (`D3_*.CT5`) muss ausschließlich dem Anschluss für ersten Scan-Kopf zugewiesen sein (mit **select_cor_table** oder **select_cor_table_list**)
- Ist zusätzlich die Option "Zweite-Scan-Kopf-Steuerung", Seite 39, freigeschaltet, kann alternativ angeschlossen werden:
  - das xy-Scan-System an den Anschluss für zweiten Scan-Kopf
  - die z-Achse an den Anschluss für ersten Scan-Kopf
  - In dieser Konstellation muss die 3D-Korrekturtabelle dem Anschluss für zweiten Scan-Kopf zugewiesen sein

Siehe auch Kapitel 4.5 "Schnittstellen zum Scan-System", Seite 69 und Abschnitt "2D- und 3D-Korrekturdateien", Seite 182.

Ansonsten werden für den Betrieb des 3-Achsen-Scan-Systems weder zusätzliche Treiber noch weitere Software-Dateien benötigt. Auch Initialisierung und Programmstart sind unverändert, siehe Kapitel 6.2 "Initialisierung und Programmstart", Seite 98.

Mehrere 3-Achsen-Systeme können (mit Multi-Board-Befehlen) gleichzeitig von einem PC aus angesteuert werden. Dazu müssen entsprechend viele RTC6 PCIe-Karten mit freigeschalteter Option "3D" in den PC eingebaut sein.

(1) Bestehend aus einem xy-Scan-System und einer dynamischen Fokussiereinheit (siehe Fußnote 2) als z-Achse.

(2) Siehe Glossareintrag auf Seite 28.

3D-Befehle

Das sind z.B. die folgenden RTC6-Befehle:

- **[*]3d[*]**
- **goto_xyz**
- **set_offset_xyz** und **set_offset_xyz_list**

Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktionieren die 3D-Befehle wie ihre entsprechenden 2D-Befehle:

- Die angegebenen Vektoren und Bögen werden in Mikroschritte zerlegt, siehe Kapitel 7.1.2 "Zerlegung in Mikroschritte", Seite 143
- Sprunggeschwindigkeit und Markiergeschwindigkeit werden mit **set_jump_speed** und **set_mark_speed** angegeben, siehe Kapitel 7.1.1 "Mit Vektor-Befehlen und Bogen-Befehlen markieren", Seite 139, oder sie werden bei zeitbemessenen Vektor-Befehlen aus der angegebenen Sprung- oder Markierdauer automatisch ermittelt
- Entsprechend der mit **select_cor_table** oder **select_cor_table_list** zugewiesenen 3D-Korrekturtabelle wird eine 3D-Bildfeld-Korrektur durchgeführt, siehe Kapitel 7.3.5 "Bildfeld-Korrektur und Korrekturtabellen", Seite 181
- Bei Sprung-Befehlen und [*]mark[*]-Befehlen werden, unter Berücksichtigung der eingestellten Delays, die Lasersteuersignale ein- und ausgeschaltet, siehe Kapitel 7.2 "Delayeinstellungen – Scan-Kopf-Steuerung und Laser-Steuerung abstimmen", Seite 148

Zum Wertebereich der Daten siehe Abschnitt "Kompatibilitätsmodi", Seite 176.

Der Kalibrierfaktor K_{xy} kann mit **get_table_para** aus der verwendeten Korrekturtabelle ausgelesen werden.

Im RTC4-Kompatibilitätsmodus und RTC5-Kompatibilitätsmodus werden die drei Koordinatenwerte intern immer auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bezüglich der Sprunggeschwindigkeit oder Markiergeschwindigkeit gleich behandelt werden.

Beachten Sie bei großen z-Sprüngen die unterschiedliche Dynamik von vario*SCAN* und 2D-Scan-System.

Hinweise

- Nach der Aktivierung der Vektorkontrollierte Laser-Steuerung mit **set_vector_control** (`Ctrl` = 7) kann mit den Para-Markier-Befehlen und Para-Sprung-Befehlen eine dynamische Fokusverschiebung linear entlang des Mark-Vektors oder Sprung-Vektors, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217, eingestellt werden. Siehe auch **set_defocus** oder **set_defocus_list**.
- Die Größe des nutzbaren Bildfelds(1) und der z-Hub(2) (Höhe des nutzbaren 3D-Bildfelds) kann der `*_ReadMe.txt`-Datei, die der 3D-Korrekturdatei beigefügt ist, sowie dem Handbuch des jeweiligen 3-Achsen-Scan-Systems/vario*SCAN*s (Kapitel "Technische Spezifikationen") entnommen werden.
- Wenn eine z-Achse dazu benutzt werden soll, den Laserfokus nur in einer bestimmten Ebene zu halten (insbesondere bei 3D-Systemen ohne Objektiv), dann können auch 2D-Vektor-Befehle verwendet werden. 2D-Vektor-Befehle lassen die zuvor mit einem 3D-Vektor-Befehl eingestellte z-Position unverändert und regeln nur die Fokuslänge nach.
- Wenn die Option "3D", Seite 39, nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist:
  - Die Zerlegung in Mikroschritte wird unter Einschluss der z-Achse berechnet (was die effektive Sprunggeschwindigkeit und Markiergeschwindigkeit in der xy-Ebene beeinflusst)
  - Es findet lediglich keine Ausgabe für die z-Achse statt

(1) Zeile "Max. Field Size (z=0): nnn.nnn mm".

(2) Zeile "Max. Z-Range: +/- nn.n mm".

- Mit **set_offset_xyz** oder **set_offset_xyz_list** kann für die z-Koordinate ein Offset definiert werden. Zusätzlich kann mit **set_defocus** oder **set_defocus_list** ein Offset auf die berechnete Fokuslänge festgelegt werden, der sich dann auch (in zum Offset entgegengesetzter Richtung) auf den z-Ausgabewert auswirkt.
- Während eines Markierungsprozesses wird die Fokuslänge des Scan-Systems kontinuierlich nachgestellt. Während der Ausführung eines Sprung-Befehls (oder **goto_xyz**) kann dieses Nachstellen ausgeschaltet werden.
  Mit `DirectMove3D` = 1 in **set_delay_mode** oder **set_delay_mode_list** wird der z-Ausgabewert während des Sprungs direkt (linear) zu seinem Endwert geführt.

Schleppverzüge anpassen

Wenn bei einem 3D-Scan-System der xy-Scan-Kopf und die Dynamische Fokussiereinheit unterschiedliche Schleppverzüge (siehe Glossareintrag Seite 30) haben, kann mit **set_timelag_compensation** die Ausgabe der schnelleren Komponente an diejenige der langsameren Komponente *verzögert* werden.

Die Lasersteuersignal-Ausgabe wird mit der langsameren Komponente synchronisiert. Dadurch kann z.B. ein Sky-Writing mit 3D-Befehlen ohne dauernde Defokussierung realisiert werden.

Im Falle eines SCANahead-Systems wird der Parameter `PreviewTime` aus **set_scanahead_params** automatisch als "Schleppverzug" verwendet.

Erweiterte 3D-Korrektur

- Weitere Informationen zum konkreten Vorgehen finden Sie im Handbuch "Kalibrieren eines 3-Achsen-Laser-Scan-System", Kapitel 5.4 "Schritt 4: Streckung korrigieren".

Bei 3D-Scan-Systemen ohne Objektiv oder mit einem nicht-telezentrischen F-Theta-Objektiv, hängt die Bildfeld-Größe vom Abstand zum Scan-Kopf-Objektiv ab (z-Koordinate).

Typischerweise wird die Bildfeld-Größe linear mit dem Z-Wert gestreckt oder gestaucht. SCANLAB 3D-Korrekturdateien enthalten daher sogenannte Stretch-Faktoren, siehe Abschnitt "ct5-Korrekturdateikopf", Seite 186, die diesen Effekt kompensieren.

Bei manchen Objektiven kann es aber auch vorkommen, dass mit der typischen Streckung auch eine Änderung der Bildgeometrie einhergeht. Dann sind zusätzliche Stretch-Korrekturen erforderlich, die diese Änderungen xy-ortsabhängig, aber linear in Z kompensieren (2D-Stretch-Korrekturtabelle). Diese Stretch-Korrekturwerte bedeuten den Z-Gradienten für die Ortsabweichung an diesem Punkt.

Angenommen, in einer beliebig, aber ungleich 0 wählbaren Z-Ebene sei (X, Y) die gewünschte Position und (X', Y') die gemessene Position. Dann berechnen sich die `<StretchX>`- und `<StretchY>`-Korrekturen wie folgt:

```
<StretchX> = (X-X')/Z
<StretchY> = (Y-Y')/Z
```

Mit **load_stretch_table** kann dann die erweiterte 3D-Korrektur aus einer ASCII-Textdatei auf die RTC6 PCIe-Karte geladen und einer bereits geladenen 3D-Korrekturtabelle zugeordnet werden. Die ASCII-Textdatei muss die Korrekturen zu einem vollständigen, rechtwinkligen Gitter enthalten. Die Anzahl der Gitterlinien und deren Abstände sind beliebig wählbar, für X und Y auch unterschiedlich. Wenn die Absolut-Werte der Korrekturen 0,03125 überschreiten, werden sie auf diese Grenzwerte geclippt. Die Korrekturen werden für Datenpunkte innerhalb des angegebenen Gitters bilinear interpoliert und für Datenpunkte außerhalb linear extrapoliert.

Die erweiterte 3D-Korrektur ist nach **load_program_file** standardmäßig nicht aktiv. Sie wird aktiv, sobald eine gültige Tabelle auf die RTC6 PCIe-Karte geladen ist. Sie kann mit **load_stretch_table** durch Angabe eines NULL-Zeigers anstelle des Dateinamens wieder deaktiviert werden.

Für die 2D-Stretch-Korrekturtabellen gelten die folgenden Regeln:

- Die ASCII-Textdatei kann eine oder mehrere Tabellen enthalten.(1)
- Jede 2D-Stretch-Korrekturtabelle muss mit der Zeile beginnen:
  `[StretchTable<No>]`
  `<No>` steht für die Tabellennummer, die bei **load_stretch_table** anzugeben ist.
- Unmittelbar anschließend folgt ein Block mit den Datenpunkten.
- Wenn mehrere Tabellen mit derselben Nummer existieren, werden nur die Daten des ersten Vorkommens eingelesen, die anderen werden ignoriert.
- Das Einlesen der Textdatei wird beendet, sobald das Dateiende oder eine Zeile mit einer Caption erreicht wird.
- Alle Zeichen rechts von einem Semikolon werden als Kommentar interpretiert und ignoriert.
- Die Reihenfolge der Datenpunkte ist beliebig.
- Eine Datenzeile darf maximal 255 Zeichen lang sein.
- Eine Datenzeile enthält zwei Positionskoordinaten (in Bits, Ganzzahl mit Vorzeichen) und zwei Korrekturwerte (dimensionslos, Gleitkommazahl mit Vorzeichen, Punkt oder Komma als Dezimaltrennzeichen), jeweils durch Leerzeichen oder Tabulatoren getrennt:
  `<Xpos> <Ypos> <StretchX> <StretchY>`
- Wenn ein Datenpunkt mehrfach vorkommt, wird der zuletzt eingelesene Wert übernommen.
- Leere oder unvollständige Zeilen sind ungültig und werden ignoriert.

(1) Sogar eines anderen Typs, siehe Tabelle 1, Seite 156.

8.5.3 Mehrere Korrekturtabellen verwenden

Der RTC6 PCIe-Kartenspeicher kann 8 verschiedene Korrekturtabellen gleichzeitig aufnehmen.

Auch wenn nur ein einziges Scan-System (2D oder 3D) verwendet wird, kann es nützlich sein, mit mehreren verschiedenen Korrekturtabellen zu arbeiten. Beispiel: es werden ein Pilotlaser und ein Arbeitslaser mit unterschiedlicher Wellenlänge eingesetzt.

Spezielle Anwendungen benötigen u. U. auch Korrekturtabellen in schneller Abfolge. Dieser Wechsel kann z.B. mit `select_cor_table( n, 0 )` oder `select_cor_table_list( n, 0 )` erfolgen (`n`= 1…`number_of_correction_tables`).

number_of_correction_tables dient dazu, andere Befehle (z.B. **load_correction_file** und **select_cor_table**) vor unerwünschten Tabellennummern zu schützen.

Bestehende Anwenderprogramme müssen nicht geändert werden, außer wenn künftig Benutzer-Eingaben (mittels expliziter RTC6-Fehlermeldungen) zurückgewiesen werden sollen.

8.6 “Processing on the fly”

- *“Processing on the fly”* bedeutet die Verknüpfung von Werkstückbewegung und Scan-Systembewegung.
- Die Nutzung der *“Processing on the fly”*-Funktionalität erfordert die Option “Processing on the fly”.
- Ob die Option “Processing on the fly” freigeschaltet ist, kann mit **get_rtc_version** überprüft werden.
- Verfügbar sind (nicht mischbar):
  – “Klassische” “Processing on the fly”-Befehle(1)
  – “Fly-Erweiterung”-Befehle, Seite 271(2)

8.6.1 Verwendung und Initialisierung

Wenn die Option “Processing on the fly” der RTC6 PCIe-Karte freigeschaltet ist, können auch bewegte Werkstücke (z.B. auf einem Förderband, auf einer rotierenden Scheibe oder auf einem xy-Verfahrtisch), aber auch feststehende Werkstücke mit einem bewegten Scan-System (z.B. mit einem Roboterarm) bearbeitet werden.

Zur Abstimmung des Laser-Scan-Prozesses auf die aktuelle Position des Werkstücks relativ zum Scan-System kann die Position des Werkstücks oder des Scan-Systems an die RTC6 PCIe-Karte übergeben werden:

- indirekt über Encoderzähler
- direkt über die McBSP-Schnittstelle

Ist die “Processing on the fly”-Korrektur aktiviert, werden die Koordinatenwerte aller Vektor-Befehle und Bogen-Befehle entsprechend der übergebenen Positionswerte transformiert.

Bei Übergabe der Bewegung als Encoderpulse werden RTC6-interne Encoderzähler getriggert. Die Zählwerte entsprechen der aktuellen Position. Sie werden mit einem Skalierungsfaktor (aus bestimmten RTC6-Befehlen) versehen und dann als *“Processing on the fly”*-Korrektur verwendet.

(1) Siehe “Klassische” “Processing on the fly”-Kontrollbefehle, Seite 331 und “Klassische” “Processing on the fly”-Listenbefehle, Seite 337.

(2) Siehe “Fly-Erweiterung”-Listenbefehle, Seite 337.

Siehe auch Kapitel 9.3.3 “Synchronisation durch Encodersignale”, Seite 323.

Bei Übergabe der Positionswerte über die McBSP-Schnittstelle wird deren Eingabewert mit einem Skalierungsfaktor versehen und dann als “Processing on the fly”-Korrektur verwendet.

Siehe auch Kapitel 9.3.4 “Synchronisation und Online-Positionierung mit McBSP-Signalen”, Seite 325.

Die beiden Übergabemöglichkeiten können nicht gleichzeitig zur “Processing on the fly”-Korrektur zweier unabhängiger Bewegungen verwendet werden, siehe Abschnitt “Übersicht”, Seite 254.

Die McBSP-Schnittstelle kann nicht gleichzeitig für eine “Processing on the fly”-Anwendung und eine Online-Positionierung verwendet werden, siehe Abschnitt “Hinweise”, Seite 242.

Die “Processing on the fly”-Korrektur wird über Listenbefehle aktiviert und deaktiviert. Die zur Aktivierung einzustellenden Parameter müssen gegebenenfalls zuvor durch einen Kalibrierungsprozess ermittelt werden (s.u.).

Ist beiden Scan-Kopf-Anschlüssen je eine 2D-Korrekturtabelle zugewiesen, wirkt sich eine “Processing on the fly”-Korrektur an beiden Scan-Kopf-Anschlüssen gleich aus.

Auch für die z-Achse kann eine “Processing on the fly”-Korrektur aktiviert werden, siehe Kapitel 8.6.11 ““Processing on the fly”-Korrektur für die z-Achse”, Seite 269.

Übersicht

Die folgenden encoderbasierten “Processing on the fly”-Korrekturen von Bewegungen sind möglich:

- **set_fly_x**, **set_fly_y**, auch in Kombination Lineare Bewegungen des Werkstücks (z.B. mit Förderband oder xy-Verfahrtisch) ausgleichen, siehe Kapitel 8.6.2 “Lineare Bewegungen ausgleichen”, Seite 255.
- **set_fly_rot**
  Rotationsbewegungen des Werkstücks (z.B. mit Drehteller) ausgleichen, siehe Kapitel 8.6.3 “Rotationsbewegungen ausgleichen”, Seite 259.
- **set_fly_2d**
  2D-Bewegungen des Werkstücks (z.B. xy-Verfahrtisch) ausgleichen, siehe Kapitel 8.6.4 “2D-Bewegungen ausgleichen”, Seite 261.

Die folgenden McBSP-basierten "Processing on the fly"-Korrekturen von Bewegungen sind möglich:

- **set_fly_x_pos**, **set_fly_y_pos**, auch in Kombination
  1D-Bewegungen oder 2D-Bewegungen des Scan-Systems (z.B. Roboterarm) ausgleichen, siehe Kapitel 8.6.2 "Lineare Bewegungen ausgleichen", Seite 255.
- **set_fly_rot_pos**
  Rotationsbewegungen des Scan-Systems (z.B. Drehtisch) ausgleichen, siehe Kapitel 8.6.3 "Rotationsbewegungen ausgleichen", Seite 259.
- **set_mcbsp_in**, **set_mcbsp_in_list**
  1D- oder 2D-Bewegungen oder Rotationsbewegungen (z.B. Roboterarm, Drehtisch) ausgleichen, siehe Kapitel 8.6.2 "Lineare Bewegungen ausgleichen", Seite 255 und Kapitel 8.6.3 "Rotationsbewegungen ausgleichen", Seite 259.

Folgende "Processing on the fly"-Korrekturen können miteinander kombiniert werden:

- **set_fly_x** mit **set_fly_y**
- **set_fly_x_pos** mit **set_fly_y_pos**
- Zum Sonderfall **set_fly_z** siehe Kapitel 8.6.11 ""Processing on the fly"-Korrektur für die z-Achse", Seite 269
- Zu linearen 3D-"Processing on the fly"-Korrekturen mit Positionsangaben siehe Abschnitt "Korrektur über McBSP-Schnittstelle mit erweiterter McBSP-Eingabe", Seite 258.

Folgende "Processing on the fly"-Korrekturen können *nicht* miteinander kombiniert werden:

- Encoderbasierte mit McBSP-basierten
- lineare mit rotierende

Der letzte Befehl bestimmt immer die gesamte Korrektur, wenn sie nicht mit vorherigen Einstellungen kombinierbar ist.

Eine gegenseitige Synchronisation beliebiger "Processing on the fly"-Korrekturen über **wait_for_encoder_mode**, **wait_for_encoder_in_range** und **wait_for_mcbsp** ist dagegen möglich, siehe Kapitel 8.6.7 ""Processing on the fly"-Anwendungen synchronisieren", Seite 264.

8.6.2 Lineare Bewegungen ausgleichen

Die "Processing on the fly"-Korrektur linearer Bewegungen (Translationen) kann aktiviert werden(1) mit:

- **set_fly_x** und/oder **set_fly_y**
- **set_fly_x_pos** und/oder **set_fly_y_pos**
- **set_mcbsp_in** (`Mode` = 1…3)
- **set_mcbsp_in_list** (`Mode` = 1…3)

Dabei muss jeweils ein Skalierungsfaktor angegeben werden.

Mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** kann eine weitere "Processing on the fly"-Korrektur mit Positionsangaben für lineare Bewegungen in allen drei Koordinatenrichtungen ohne Beschränkung der Bit-Auflösung aktiviert werden, wodurch kein Skalierungsfaktor erforderlich ist.

Die "Processing on the fly"-Korrektur kann (für beide Raumrichtungen gleichzeitig) mit **fly_return** deaktiviert werden. Siehe hierzu auch die Hinweise in Kapitel 8.6.5 ""Processing on the fly"-Korrektur deaktivieren", Seite 263.

Korrektur über Encoderzähler

Um Positionswerte über die Karten-internen Encoderzähler(2) übergeben zu können, muss die "Processing on the fly"-Korrektur aktiviert werden mit:

- **set_fly_x** und/oder **set_fly_y**

Der Skalierungsfaktor [in Bits/Count] definiert hier das Verhältnis zwischen der Verschiebung der aktuellen Ausgabeposition im Bildfeld [in Bits] und einem Zählschritt [Count] des jeweiligen Encoderzählers. Siehe auch Abschnitt "Bestimmen der Skalierungsfaktoren", Seite 256.

Durch **set_fly_x** wird der Encoderzähler "Encoder0" auf Null zurückgesetzt, durch **set_fly_y** der Encoderzähler "Encoder1".(3)

(1) Unterschiedliche "Processing on the fly"-Korrekturen können nicht beliebig kombiniert werden, siehe Abschnitt "Übersicht", Seite 254.

(2) Siehe Achtung!, Seite 59.

(3) Mit **set_control_mode** `Bit #9` kann zuvor eingestellt werden, ob der Zähler mit dem jeweiligen "Processing on the fly"-Befehl oder mit dem Start-Trigger zurückgesetzt wird.

Danach wird der Ausgabewert aus der aktuellen Ausgabeposition berechnet, indem (für alle Raumrichtungen) das Produkt aus Skalierungsfaktor und aktuellem Zählerstand aufaddiert wird.

Diese Korrektur erfolgt alle 10 μs.(1)

Hinweise

- Die Kombination aus **set_fly_x** und **set_fly_y** kann sowohl für 1D- als auch für 2D-"Processing on the fly"-Anwendungen (z.B. für einen xy-Verfahrtisch) verwendet werden, vornehmlich für separate oder separierbare Markierungen im realen Bildfeld.
- Für zusammenhängende Markierungen im virtuellen Bildfeld empfiehlt SCANLAB eher die Verwendung von **set_fly_2d**, siehe Kapitel 8.6.4 "2D-Bewegungen ausgleichen", Seite 261.

Bestimmen der Skalierungsfaktoren

Die Skalierungsfaktoren können experimentell bestimmt werden:

(1) Lesen Sie den Zähler-Startwert mit **get_encoder** aus.

(2) Starten Sie die Bewegung.

(3) Stoppen Sie die Bewegung.

(4) Lesen Sie den Zähler-Endwert mit **get_encoder** aus.(2)

(5) Messen Sie die zurückgelegte Distanz in mm.

(6) Berechnen Sie das Encoderinkrement i wie folgt:

$$i = \frac{(\text{Zaehler-Endwert} - \text{Zaehler-Startwert})}{\text{zurueckgelegte Distanz}}$$

(7) Berechnen Sie die Skalierungsfaktoren `Scalex` und `Scaley` [in Bits/Count] wie folgt:

$$\texttt{Scalex} = \frac{K}{i_x}$$

$$\texttt{Scaley} = \frac{K}{i_y}$$

Dabei ist K ist der Kalibrierfaktor [in Bits/mm], siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175 und i das Encoderinkrement aus Schritt 6.

Falls das Werkstück sich mit konstanter Geschwindigkeit bewegt (v_x oder v_y [in mm/s]) und mit **simulate_encoder** eine Encodersimulation aktiviert wurde, berechnen Sie die Skalierungsfaktoren wie folgt:

$$\texttt{Scalex} = \frac{K \times v_x}{(1.000.000 \text{ counts} / \text{s})}$$

$$\texttt{Scaley} = \frac{K \times v_y}{(1.000.000 \text{ counts} / \text{s})}$$

(8) Passen Sie die Vorzeichen der Skalierungsfaktoren entsprechend der Bewegungsrichtung an.

(1) Die Encoderzähler sind vorzeichenbehaftete 32-Bit-Zähler mit Überlauf (= nach Erreichen des maximalen (minimalen) Zählerwerts wird die Zählung mit dem minimalen (maximalen) Zählerwert fortgesetzt).

(2) Alternativ können Zähler-Start- und Endwert mit dem Listenbefehl **store_encoder** in einen Zwischenspeicher der RTC6 PCIe-Karte gespeichert und dann mit dem Kontrollbefehl **read_encoder** ausgelesen werden.

Korrektur über McBSP-Schnittstelle

Um Positionswerte über die McBSP-Schnittstelle übergeben zu können, muss die "Processing on the fly"-Korrektur aktiviert werden mit:

- **set_fly_x_pos** und/oder **set_fly_y_pos**
- **set_mcbsp_in** für Positionsangaben in 2 Raumrichtungen
- **set_multi_mcbsp_in** für Positionsangaben in allen 3 Raumrichtungen (benötigt keine Skalierungsfaktoren)

Der Skalierungsfaktor [in Bits/Bits] definiert hier das Verhältnis zwischen der Verschiebung der aktuellen Ausgabeposition im Bildfeld [in Bits] und dem eingelesenen Wert an der McBSP-Schnittstelle [in Bits]. Siehe auch Abschnitt "Bestimmen der Skalierungsfaktoren", Seite 257.

Der Ausgabewert wird aus der aktuellen Ausgabeposition berechnet, indem (für alle Raumrichtungen) das Produkt aus Skalierungsfaktor und aktuellem Eingabewert der McBSP-Schnittstelle aufaddiert wird.

Diese Korrektur erfolgt alle 10 μs.

An der McBSP-Schnittstelle, siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87, stehen zur Verfügung:

- Für 1D-Korrekturen – 32-Bit-Werte mit Vorzeichen
- Für 2D-Korrekturen – 16-Bit-Werte mit Vorzeichen für jede Achse (der x-Wert steht in den unteren 16 Bit und der y-Wert in den oberen 16 Bit des Wertes an der McBSP-Schnittstelle)

Hinweise

- Nach **set_fly_x_pos** und **set_fly_y_pos** werden Eingabewerte, die an der McBSP-Schnittstelle empfangen werden, automatisch an die interne Speicherstelle 0 übertragen. Das gilt weiterhin auch noch nach Abschalten der "Processing on the fly"-Korrektur mit **fly_return**, **set_fly_x_pos**, **set_fly_y_pos** oder **set_fly_rot_pos** sowie nach **load_program_file**. Die aktuellen Daten an der Speicherstelle 0 können mit `read_mcbsp(0)` ausgelesen werden.
- Nach einer mit **set_mcbsp_in** oder **set_mcbsp_in_list** aktivierten "Processing on the fly"-Korrektur werden McBSP-Eingabewerte dagegen auch an die internen Speicherstellen 1, 2 und/oder 3 übertragen, siehe Abschnitt "Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe", Seite 258.
- Nach einer über **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktivierten "Processing on the fly"-Korrektur werden die übertragenen Daten nacheinander an die Speicherstellen 0…3 geschrieben. Von dort können sie mit **read_mcbsp** unsortiert oder mit **read_multi_mcbsp** sortiert ausgelesen werden.

Bestimmen der Skalierungsfaktoren

Die Skalierungsfaktoren können experimentell bestimmt werden:

(1) Lesen Sie den Start-Eingabewert mit **read_mcbsp** aus.

(2) Starten Sie die Bewegung.

(3) Stoppen Sie die Bewegung.

(4) Lesen Sie den End-Eingabewert mit **read_mcbsp** aus.

(5) Messen Sie die zurückgelegte Distanz in mm.

(6) Berechnen Sie das Positions-Inkrement i wie folgt:

$$i = \frac{(\text{End-Eingabewert} - \text{Start-Eingabewert})}{\text{zurueckgelegte Distanz}}$$

(7) Berechnen Sie die Skalierungsfaktoren `Scalex` und `Scaley` [in Bits/Bits] wie folgt:

$$\texttt{Scalex} = \frac{K}{i_x}$$

$$\texttt{Scaley} = \frac{K}{i_y}$$

Dabei ist K ist der Kalibrierfaktor [in Bits/mm], siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175 und i das Positions-Inkrement aus Schritt 6.

Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe

Eine “Processing on the fly”-Korrektur für lineare Bewegungen mit McBSP-Eingabewerten kann (alternativ zu **set_fly_x_pos** oder **set_fly_y_pos**) auch mit **set_mcbsp_in** oder **set_mcbsp_in_list** (`Mode` = 1…3) aktiviert werden.

Der Vorteil dieser Befehle liegt darin, dass die McBSP-Schnittstelle bei aktivierter “Processing on the fly”-Korrektur zusätzlich zur Eingabe von beliebigen anderen Signalen (die nicht für die “Processing on the fly”-Korrektur berücksichtigt werden sollen) verwendet werden kann.

Alle McBSP-Eingabewerte müssen dazu über Bit #31 codiert werden:

- Bit #31 = 0: Der Eingabewert wird an die interne Speicherstelle 0 übertragen und zur “Processing on the fly”-Korrektur verwendet.
- Bit #31 = 1: Der Eingabewert wird an die interne Speicherstelle 3 übertragen, aber nicht zur “Processing on the fly”-Korrektur verwendet.

Hinweise

- Alle Eingabewerte werden immer alternierend an die internen Speicherstellen 1 und 2 übertragen und anschließend gemäß der Bit #31-Codierung auch an die Speicherstellen 0 oder 3, nach Deaktivieren der Korrektur mit `set_mcbsp_in(0)` oder `set_mcbsp_in_list(0)` dagegen nur an die Speicherstellen 1 und 2.
- Die aktuellen Daten der internen Speicherstellen 0 - 3 können mit **read_mcbsp** ausgelesen werden.
- Bei **set_mcbsp_in** oder **set_mcbsp_in_list** wird ein Skalierungsfaktor angegeben. Für 2D-Korrekturen (`Mode` = 3) gilt der Skalierungsfaktor für beide Achsen gleichzeitig. Die Kalibrierung zur Ermittlung des Skalierungsfaktors erfolgt wie für **set_fly_x_pos** und **set_fly_y_pos** (s.o.).
- Für die Übertragung von 1D-Korrekturwerten stehen effektiv 31 Bit mit Vorzeichen zur Verfügung (Bit #31 ist als Codierungs-Bit reserviert), für die Übertragung von 2D-Korrekturwerten dagegen nur jeweils 15 Bit (mit Vorzeichen) pro Achse. Dabei steht der x-Wert in den unteren 16 Bit, der y-Wert in den oberen 16 Bit des Wertes an der McBSP-Schnittstelle. Zur Beschreibung der Schnittstelle siehe Kapitel 4.6.6 “McBSP/ANALOG-Stiftleiste“, Seite 87.

Korrektur über McBSP-Schnittstelle mit erweiterter McBSP-Eingabe

Eine “Processing on the fly”-Korrektur für lineare Bewegungen kann alternativ zu den Methoden der vorherigen Kapitel auch mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktiviert werden.

Der Vorteil dieser Befehle gegenüber **set_mcbsp_in** oder **set_mcbsp_in_list** liegt darin, dass neben einer “Processing on the fly”-Korrektur in x-Richtung und y-Richtung auch eine Korrektur in z-Richtung sowie eine Anpassung der Laserleistung möglich ist. Außerdem können bis zu 4 weitere Signale zur freien Verwendung übertragen werden.

Die asynchron in den McBSP-Speicher auf die Speicherstellen 0 bis 3 übertragenen Daten werden einmal pro 10 μs auf einen weiteren internen Speicher sortiert. Von dort aus werden sie endgültig verwendet und können nach Signaltypen sortiert mit **read_multi_mcbsp** ausgelesen werden.

8.6.3 Rotationsbewegungen ausgleichen

Vor der Aktivierung einer "Processing on the fly"-Korrektur für xy-Rotationsbewegungen sollte das Rotationszentrum mit **set_rot_center** oder **set_rot_center_list** angegeben werden. Dieses kann auch außerhalb des Bildfelds liegen.

Die "Processing on the fly"-Korrektur kann dann aktiviert werden mit:

- **set_fly_rot**
- **set_fly_rot_pos**
- **set_mcbsp_in** (`Mode` = 4)
- **set_mcbsp_in_list** (`Mode` = 4)

Mit **fly_return** kann die "Processing on the fly"-Korrektur deaktiviert werden (siehe hierzu auch die Hinweise auf Kapitel 8.6.5 ""Processing on the fly"-Korrektur deaktivieren", Seite 263).

Hinweise

- Eine "Processing on the fly"-Rotationskorrektur:
  - Kann *nicht* mit anderen "Processing on the fly"-Korrekturen kombiniert werden, siehe Abschnitt "Übersicht", Seite 254
  - Ist gleichzeitig für beide Scan-Kopf-Anschlüsse nur sinnvoll, wenn die zwei angeschlossenen Scan-Systeme exakt auf dasselbe Rotationszentrum justiert sind

Korrektur über Encoderzähler

Um Drehwinkel über die Karten-internen Encoderzähler(1) übergeben zu können, muss der Encodereingang ENCODER X(2) angeschlossen werden, siehe auch Abschnitt "Eingänge für externe Encodersignale", Seite 324.

Die "Processing on the fly"-Korrektur muss dann mit **set_fly_rot** aktiviert werden. Dabei muss der Parameter `Resolution` [in Counts/Umdrehung] angegeben werden. Siehe auch Abschnitt "Bestimmen des Parameters Resolution", Seite 259.

Durch **set_fly_rot** wird der Encoderzähler "Encoder0" auf Null zurückgesetzt.(3)

Der Ausgabewert wird aus der aktuellen Ausgabeposition über eine Drehmatrix um das angegebene Rotationszentrum mit Drehwinkel / 360° = aktueller "Encoder0"-Zählerstand / `Resolution`) berechnet.

Diese Korrektur erfolgt alle 10 µs.(4)

Bestimmen des Parameters Resolution

Der Parameter `Resolution` kann experimentell bestimmt werden:

(1) Lesen Sie den Zähler-Startwert mit **get_encoder** aus.

(2) Starten Sie die Drehbewegung.

(3) Zählen Sie die Zahl der Umdrehungen.

(4) Stoppen Sie die Drehbewegung.

(5) Lesen Sie den Zähler-Endwert aus mit **get_encoder**.(5)

(6) Berechnen Sie den Parameter `Resolution` [in Counts/Umdrehung] wie folgt:

$$\texttt{Resolution} = \frac{(\text{Zaehler-Endwert} - \text{Zaehler-Startwert})}{\text{Zahl der Umdrehungen}}$$

Falls sich das Werkstück mit konstanter Rotationsgeschwindigkeit ω [in Umdrehungen/s] bewegt und mit **simulate_encoder** eine Encodersimulation aktiviert wurde, dann kann der Parameter `Resolution` wie folgt berechnet werden:

$$\texttt{Resolution} = \frac{(1.000.000 \text{ counts / s})}{\omega}$$

(7) Passen Sie das Vorzeichen von `Resolution` entsprechend der Drehrichtung an.

(1) Siehe Achtung!, Seite 59.

(2) ENCODER X = Encoder0.

(3) Mit **set_control_mode** `Bit #9` kann zuvor eingestellt werden, ob der Zähler mit dem jeweiligen "Processing on the fly"-Befehl oder mit dem Start-Trigger zurückgesetzt wird.

(4) Die Encoderzähler sind vorzeichenbehaftete 32-Bit-Zähler mit Überlauf (= nach Erreichen des maximalen (minimalen) Zählerwerts wird die Zählung mit dem minimalen (maximalen) Zählerwert fortgesetzt).

(5) Alternativ können Zähler-Start- und Endwert mit dem Listenbefehl **store_encoder** in einen Zwischenspeicher der RTC6 PCIe-Karte gespeichert und dann mit dem Kontrollbefehl **read_encoder** ausgelesen werden.

Korrektur über McBSP-Schnittstelle

Erfolgt die Übergabe der Winkelpositionswerte für die “Processing on the fly”-Korrektur von Rotationsbewegungen über die McBSP-Schnittstelle, so muss die “Processing on the fly”-Korrektur mit **set_fly_rot_pos** aktiviert werden.

Der anzugebende Parameter `Resolution` hat die gleiche Bedeutung wie bei **set_fly_rot** und wird gleich ermittelt, siehe Abschnitt “Bestimmen des Parameters Resolution”, Seite 259.

McBSP-Eingangswerte werden mit **read_mcbsp** ausgelesen.

Hinweise

- McBSP-Eingabewerte werden an die Karten-interne Speicherstelle 0 übertragen, siehe Hinweise in Abschnitt “Korrektur über McBSP-Schnittstelle”, Seite 257.

Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe

Eine “Processing on the fly”-Korrektur für Rotationsbewegungen mit McBSP-Eingabewerten kann aktiviert werden mit:

- **set_mcbsp_in** (`Mode` = 4)
- **set_mcbsp_in_list** (`Mode` = 4)

Diese Befehle haben den Vorteil, dass die McBSP-Schnittstelle auch bei aktivierter “Processing on the fly”-Korrektur zusätzlich zur Eingabe von beliebigen anderen Signalen verwendet werden kann, die bei der “Processing on the fly”-Korrektur nicht berücksichtigt werden sollen.

Dazu müssen alle McBSP-Eingabewerte über Bit #31 codiert werden.

Hinweise

- Alle Eingabewerte werden immer alternierend an die internen Speicherstellen 1 und 2 übertragen und anschließend gemäß der Bit #31-Codierung auch an die Speicherstellen 0 oder 3, nach Deaktivieren der Korrektur mit `set_mcbsp_in(0)` oder `set_mcbsp_in_list(0)` nur an die Speicherstellen 1 und 2.
- Die aktuellen Daten der internen Speicherstellen können mit **read_mcbsp** ausgelesen werden.
- Mit **set_mcbsp_in** oder **set_mcbsp_in_list** wird auch eine Rotationsauflösung angegeben. Die Kalibrierung zur Ermittlung der Rotationsauflösung erfolgt wie für **set_fly_rot_pos**, siehe Abschnitt “Bestimmen des Parameters Resolution”, Seite 259.
- Für die Übertragung von Rotations-Korrekturwerten stehen effektiv 31 Bit mit Vorzeichen zur Verfügung.

8.6.4 2D-Bewegungen ausgleichen

Zum Aktivieren einer encoderbasierten 2D-“Processing on the fly”-Korrektur (z.B. für einen xy-Verfahrtisch), insbesondere mit zusammenhängenden Markierungen im virtuellen Bildfeld, kann **set_fly_2d** verwendet werden.

Gegenüber einer Aktivierung mit **set_fly_x** und **set_fly_y** hat dies folgende Vorteile:

- **set_fly_2d** setzt die beiden Encoder gleichzeitig zurück (**set_fly_x** und **set_fly_y** dagegen als separate Befehle immer mit einer kleinen Zeitverzögerung zwischen den beiden Kanälen)
- **set_fly_2d** erlaubt eine Kompensation von nicht linearen Zusammenhängen zwischen den Encoderwerten und den tatsächlichen xy-Verfahrtisch-Bewegungen, siehe Abschnitt “2D-Encoderkompensation für xy-Verfahrtische”, Seite 261
- Auch während der Unterbrechung einer **set_fly_2d**-Markierung mit **wait_for_encoder_mode** oder **wait_for_encoder_in_range** werden die Positionen der Galvanometerscanner kontinuierlich entsprechend der aktuellen Encoderwerte des xy-Verfahrtischs “Processing on the fly”-korrigiert (der Laserfokus bleibt relativ zum xy-Verfahrtisch stehen). Dadurch werden nach xy-Verfahrtisch-Bewegungsfahrten unnötige Sprünge vermieden, siehe Kapitel 8.6.6 “Virtuelles Bildfeld mit “Processing on the fly””, Seite 264

Hinweise

- Eine **set_fly_2d**-Korrektur kann *nicht* mit anderen “Processing on the fly”-Korrekturen kombiniert werden, siehe Abschnitt “Übersicht”, Seite 254.
- Koordinatentransformationen im virtuellen Bildfeld, siehe Abschnitt “Koordinatentransformationen im virtuellen Bildfeld”, Seite 177, können bei Start einer “Processing on the fly”-Session mit geänderten Werten aktiviert werden, wenn sie zuvor mit den entsprechenden Kontrollbefehlen geladen wurden.

2D-Encoderkompensation für xy-Verfahrtische

Bei erhöhten Anforderungen an die Markiergenauigkeit kann es notwendig sein, auch die mechanischen Abweichungen eines xy-Verfahrtischs zu kompensieren.

Zu diesem Zweck können mit **load_fly_2d_table** bis zu zwei 2D Kompensationstabellen, siehe Seite 262, auf die RTC6 PCIe-Karte geladen werden.

Dann werden die Encoderwerte während einer mit **set_fly_2d** eingeleiteten “Processing on the fly”-Anwendung wie bei einer Feld-Korrekturtabelle 2D-interpoliert und kompensiert.

Um überflüssige Initialisierungsfahrten einzusparen, kann mit **init_fly_2d** eine beliebige Anfangsposition des Verfahrtisches als Referenzwert für die 2D-Encoderkompensation festgelegt (auf der RTC6 PCIe-Karte gespeichert) und gleichzeitig eine der beiden Korrekturtabellen ausgewählt werden.

Bei jedem nachfolgenden Zurücksetzen der Encoder mit **set_fly_2d** wird die aktuelle Position automatisch als neue Referenzposition übernommen. So bleibt die Beziehung zwischen den aktuellen Encoderwerten und der absoluten Position des Verfahrtischs für die Kompensation erhalten. Diese Beziehung bleibt auch beim Beenden mit **fly_return** oder bei einem erneuten Starten mit **set_fly_2d** erhalten. Sie bleibt dagegen nicht erhalten, wenn zwischendurch andere “Processing on the fly”-Korrekturen aktiviert werden, und auch nicht beim Zurücksetzen der Encoder mit einem /START (siehe **set_control_mode** (`Bit #9` = 1)).

Die aktuell gültigen Referenzwerte können jederzeit mit **get_fly_2d_offset** ausgelesen werden.

Die Encoderkompensation wirkt ausschließlich auf die “Processing on the fly”-Korrektur. Die originalen, nicht kompensierten Encoderwerte verwenden:

- **get_encoder**
- **store_encoder**
- **read_encoder**
- **wait_for_encoder**
- **wait_for_encoder_mode**
- **wait_for_encoder_in_range**
- die Aufzeichnung mit **set_trigger** (`Signal1`/`Signal2` = 43 oder 44)
- die Aufzeichnung mit **get_value**

Für **load_fly_2d_table** muss eine ASCII-Textdatei mit einer 2D-Kompensationstabelle bereitgestellt werden.

Eine 2D-Kompensationstabelle muss Encoderkompensationen zu Stützpunkten aus einem rechtwinkligen Gitter enthalten. Die Anzahl der Gitterlinien und deren Abstände sind beliebig wählbar, für x und y auch unterschiedlich. Fehlende Stützpunktdaten werden automatisch durch solche mit einer Kompensation von 0 ergänzt. Die jeweils größten vorkommenden Encoderstützpunkte bilden einen Rahmen, innerhalb dessen die Encoderkompensationswerte bilinear interpoliert werden. Encoderwerte außerhalb dieses Rahmens werden vor der Interpolation auf denselben geclippt. Daher sollten die Stützpunkte zumindest den für die jeweilige Anwendung geforderten Stellbereich des xy-Verfahrtischs überdecken. Wenn die Stützpunkte die Werte ±524.288 übersteigen, kann etwas an Genauigkeit verloren gehen.

Die 2D-Encoderkompensation ist nach **load_program_file** per Default nicht aktiv.

Sie ist aktiv, sobald eine gültige 2D-Kompensationstabelle aus einer ASCII-Textdatei auf die RTC6 PCIe-Karte geladen ist und sie kann mit **load_fly_2d_table** durch Angabe eines NULL-Zeigers anstelle des Dateinamens auch wieder deaktiviert werden.

Für die 2D-Kompensationstabellen gelten die folgenden Regeln:

- Die ASCII-Textdatei kann eine oder mehrere Tabellen enthalten.(1)
- Jede 2D-Kompensationstabelle muss mit der Zeile beginnen:
  `[Fly2DTable<No>]`
  `<No>` steht für die Tabellennummer `No`, die bei **load_fly_2d_table** anzugeben ist. Mit `No` = `<No>` wird die erste der beiden Tabellen geladen, mit `No` = `<No>` + 65.536 die Zweite.
- Unmittelbar anschließend folgt ein Block mit den Datenpunkten.
- Wenn mehrere Tabellen mit derselben Nummer existieren, werden nur die Daten des ersten Vorkommens eingelesen, die anderen werden ignoriert.
- Das Einlesen der ASCII-Textdatei wird beendet, sobald das Dateiende oder eine Zeile mit einer Caption erreicht wird.
- Alle Zeichen rechts von einem Semikolon werden als Kommentar interpretiert und ignoriert.
- Die Reihenfolge der Datenpunkte ist beliebig.
- Eine Datenzeile darf maximal 255 Zeichen lang sein.
- Eine Datenzeile enthält zwei Stützpunkt-Koordinaten für “Encoder0” und “Encoder1” (Ganzzahl mit Vorzeichen) und zwei Kompensationswerte (Ganzzahl mit Vorzeichen), jeweils durch Leerzeichen oder Tabulatoren getrennt:
  `<Encoder0> <Encoder1> <Encoder0-delta> <Encoder1-delta>`
- Wenn ein Datenpunkt mehrfach vorkommt, wird der zuletzt eingelesene Wert übernommen, die anderen werden ignoriert.
- Leere oder unvollständige Zeilen sind ungültig und werden ignoriert.

(1) Sogar eines anderen Typs, siehe Tabelle 1, Seite 156.

8.6.5 “Processing on the fly”-Korrektur deaktivieren

Alle “Processing on the fly”-Korrekturen können (für beide Raumrichtungen gleichzeitig) mit **fly_return** deaktiviert werden.

fly_return erfordert die Angabe einer neuen Ausgabeposition, die dann mit einem normalen Sprung erreicht wird.

Mit **set_multi_mcbsp_in** aktivierte “Processing on the fly”-Korrekturen sollten mit **fly_return_z** beendet werden.

In allen anderen Fällen (s.u.) kann es zu einem harten Sprung (siehe “Harter Sprung”) auf eine neue Ausgabeposition kommen.

Hinweise

- Wird die “Processing on the fly”-Korrektur nicht explizit deaktiviert(1), dann:
  - wirkt sie auch bei der Ausführung nachfolgender Listen
  - wirkt sie jedoch nicht in der Pause zwischen zwei Listen(2)
- Eine mit **set_fly_x**, **set_fly_y**, **set_fly_2d**, **set_fly_rot**, **set_fly_x_pos**, **set_fly_y_pos** oder **set_fly_rot_pos** aktivierte “Processing on the fly”-Korrektur wird auch dann deaktiviert, wenn der jeweilige Befehl erneut, aber mit einem unzulässigen Parameterwert aufgerufen wird. Hierdurch findet u.U. ein Sprung auf eine unkorrigierte Ausgabeposition statt. Siehe auch Befehlsbeschreibungen.
- Eine mit **set_fly_x** aktivierte “Processing on the fly”-Korrektur wird durch **set_fly_y** *nicht* deaktiviert und umgekehrt. Das gleiche gilt für **set_fly_x_pos** und **set_fly_y_pos**. Ansonsten deaktiviert aber jeder “Processing on the fly”-Befehl automatisch eine durch einen anderen “Processing on the fly”-Befehl aktivierte “Processing on the fly”-Korrektur, siehe auch Abschnitt ”Übersicht”, Seite 254. Hierdurch findet u.U. ein “Harter Sprung” auf eine neue Ausgabeposition statt.
- Eine mit **set_mcbsp_in** oder **set_mcbsp_in_list** aktivierte “Processing on the fly”-Korrektur wird auch durch **set_mcbsp_in** mit `Mode` = 0 oder **set_mcbsp_in_list** mit `Mode` = 0 deaktiviert. Hierdurch findet u.U. ein “Harter Sprung” auf eine unkorrigierte Ausgabeposition statt.
- Eine mit **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos**, **set_mcbsp_in** oder **set_mcbsp_in_list** aktivierte “Processing on the fly”-Korrektur wird auch durch die Befehle zum Konfigurieren einer Online-Positionierung deaktiviert, siehe Abschnitt ”Hinweise”, Seite 242. Hierdurch findet u.U. ein “Harter Sprung” auf eine neue Ausgabeposition statt.
- Eine Deaktivierung der “Processing on the fly”-Korrektur erfolgt erst nach dem Scanner-Delay.
- Bei einer mit **set_mcbsp_in** oder **set_mcbsp_in_list** aktivierten “Processing on the fly”-Korrektur deaktiviert **fly_return** zwar die “Processing on the fly”-Korrektur, aber nicht das Kopieren an interne Speicherstellen (wirkt hier also wie `set_mcbsp_in(5)`). Um hier auch das Kopieren an interne Speicherstellen nachträglich zu deaktivieren, kann `set_mcbsp_in(0)` oder `set_mcbsp_in_list(0)` verwendet werden.
- “Processing on the fly”-Korrekturen werden auch durch **stop_execution** und durch einen externen Stopp (für beide Raumrichtungen gleichzeitig) deaktiviert.

(1) **set_end_of_list** deaktiviert die “Processing on the fly”-Korrektur nicht.

(2) Die Korrektur wirkt daher nicht bei den Kontrollbefehlen **goto_xy** und **goto_xyz**.

8.6.6 Virtuelles Bildfeld mit “Processing on the fly”

Bei “Processing on the fly”-Anwendungen kann der volle Wertebereich (29-Bit) des virtuellen Bildfelds dazu genutzt werden, siehe Kapitel 7.3.3 “Virtuelles Bildfeld”, Seite 177 um Befehlslisten mit Objekten, die bis zu 512 mal größer als das reale 20-Bit-Bildfeld sind, zu laden und abzuarbeiten.

Bei 1D-“Processing on the fly”-Anwendungen (z.B. Werkstücke auf einem Förderband) dürfen die Objekte nur in der Dimension parallel zur “Processing on the fly”-Richtung größer sein als das reale Bildfeld.

Bei 2D-“Processing on the fly”-Anwendungen (z.B. mit einem xy-Verfahrtisch) können die zu markierenden Objekte einer Befehlsliste über das komplette virtuelle Bildfeld verteilt werden.

Falls eine gesamte Markieraufgabe aus mehreren Teilmarkierungen (“Kacheln”) besteht, deren Ausdehnung das reale Bildfeld nicht überschreitet, kann die “Processing on the fly”-Funktionalität auch nur dazu verwendet werden, die jeweils zu markierende Teilmarkierung in das reale Bildfeld zu bewegen und sie dann dort abzuarbeiten.

Koordinatenwerte, die auch *nach* “Processing on the fly”-Korrektur noch außerhalb des realen Bildfelds liegen, werden auf die Randwerte des realen Bildfelds geclippt. Dabei werden die Fehler-Bits für **get_marking_info** automatisch gesetzt, siehe Kapitel 8.6.9 ““Processing on the fly”-Korrektur überwachen”, Seite 267.

Wenn bei der Listenausführung eine Überschreitung des realen 20-Bit-Bildfelds droht, sollte gezielt eine Bewegungsfahrt des Förderbands oder xy-Verfahrtischs ausgeführt werden. Diese kann mit **wait_for_encoder**, **wait_for_encoder_mode**, **wait_for_encoder_in_range** oder **wait_for_mcbsp** abgewartet werden, indem die Listenausführung so lange unterbrochen wird, bis bestimmte Encoder-Werte oder McBSP-Werte erreicht sind, siehe Kapitel 8.6.7 ““Processing on the fly”-Anwendungen synchronisieren”, Seite 264.

Die geeignete Unterteilung der gesamten Markieraufgabe und die Synchronisation mit der 1D- oder 2D-“Processing on the fly”-Bewegung ist Aufgabe des Anwenderprogramms.

8.6.7 “Processing on the fly”-Anwendungen synchronisieren

Das Abarbeiten von Befehlslisten kann durch einen RTC6-Befehl oder ein externes Startsignal gestartet werden, siehe Kapitel 6.4.4 “Listen ausführen und beenden”, Seite 112.

Falls die Befehlsliste nicht unmittelbar mit dem Startsignal gestartet werden soll, kann eine entsprechende Verzögerung wie folgt realisiert werden:

- Bei Positionsübergabe über die McBSP-Schnittstelle durch ein **wait_for_mcbsp** am Beginn der Befehlsliste.
- Bei Positionsübergabe über Encoderzähler durch **set_fly_x**/**set_fly_y**, **set_fly_2d** oder **set_fly_rot** (zum Zurücksetzen der/des Zähler/s) und einem anschließenden **wait_for_encoder_mode** oder **wait_for_encoder_in_range** am Beginn der Befehlsliste. **wait_for_encoder_in_range** empfiehlt sich für 2D-encoderbasierte “Processing on the fly”-Anwendungen. **wait_for_encoder_in_range** wartet bis beide Encoder gleichzeitig innerhalb des angegebenen Bereichs liegen und ist damit unabhängig von der tatsächlichen Bahnkurve, mit der dieser Bereich erreicht wird(1).

  Bei Positionsübergabe über Encoderzähler kann (auch unabhängig von einer aktiven “Processing on the fly”-Session) eine Streckenverzögerung konfiguriert werden, um die Ausführung des tatsächlichen Starts relativ zum auslösenden Startsignal oder RTC6-Befehl zu verzögern, siehe Abschnitt “Externer Start”, Seite 315.

(1) **wait_for_encoder** oder **wait_for_encoder_mode** müssten für jeden Encoder einzeln und nacheinander abgefragt werden. Dadurch wäre die gleichzeitige Erfüllung beider Encoderkriterien von der expliziten Bahnkurve der xy-Verfahrtisch-Bewegung abhängig und man müsste schon vor dem Laden der Listen eine geeignete Bahnkurve für den xy-Verfahrtisch festlegen und reproduzieren.

Damit Totzeiten zwischen der Ausführung mehrerer ("Processing on the fly"-) Listenprogramme vermieden werden können, werden externe Starts, die mit einem externen Startsignal, mit **simulate_ext_start** oder **simulate_ext_start_ctrl** angetriggert, aber entsprechend der eingestellten Streckenverzögerung nicht sofort ausgeführt werden, in einer Warteschleife von bis zu 8 Starts gehalten (der jeweilige Start-Trigger wird automatisch generiert, wenn die Streckenverzögerung abgelaufen ist, siehe auch Abschnitt "Externer Start", Seite 315).

Außerdem kann der Listenbefehl **simulate_ext_start** verwendet werden, um mehrere Listen in definierten Abständen auszuführen.

Wenn eine Liste für zwischengeschaltete Bewegungsfahrten mit **wait_for_encoder**, **wait_for_encoder_mode**, **wait_for_encoder_in_range** oder **wait_for_mcbsp** unterbrochen wird, siehe Kapitel 8.6.6 "Virtuelles Bildfeld mit "Processing on the fly"", Seite 264, dann werden die Signale für den "Laser active"-Betrieb vor der Bewegungsfahrt nicht verändert.

Bei encoderbasierten "Processing on the fly"-Anwendungen kann der Laserfokus mit **park_position** vor einer Unterbrechung an eine sichere Parkposition und nach der Unterbrechung mit **park_return** wieder zurück zur Ausgangsposition oder an irgendeine andere Position bewegt werden. Zu weiteren Details siehe Befehlsbeschreibung.

Das Verhalten der Galvanometerscanner während einer solchen Bewegungsfahrt hängt von der verwendeten "Processing on the fly"-Korrektur ab:

- Bei McBSP-basierten "Processing on the fly"-Korrekturen sowie bei den encoderbasierten "Processing on the fly"-Korrekturen **set_fly_x**, **set_fly_y** und **set_fly_rot** bleiben die Galvanometerscanner während der Listenunterbrechung relativ zum realen Bildfeld stehen.
  I. d. R. muss danach ein Sprung zur nächsten Markierung ausgeführt werden.
- Bei der encoderbasierten **set_fly_2d**-"Processing on the fly"-Korrektur werden die Positionen der Galvanometerscanner kontinuierlich "Processing on the fly"-korrigiert (der Laserfokus bleibt dadurch relativ zum Markierobjekt stehen). Der nachfolgende Sprung zur nächsten Markierung oder Fortsetzung derselben ist i. d. R. sehr klein.
- Bei **set_fly_2d** kann es bei einer längeren Bewegungsfahrt zu einem Clipping kommen, wenn die "Processing on the fly"-korrigierten Koordinatenwerte während der Bewegungsfahrt die Grenzen des realen Bildfelds überschreiten würden. Um dies zu vermeiden, kann es hier sinnvoll sein, die "Processing on the fly"-Korrektur vor der Bewegungsfahrt (mit **fly_return**, siehe Kapitel 8.6.5 ""Processing on the fly"-Korrektur deaktivieren", Seite 263) auszuschalten (die Galvanometerscanner bleiben dann stehen). Zum Wiedereinschalten nach der Bewegungsfahrt kann hier **activate_fly_2d** statt **set_fly_2d** (oder **activate_fly_xy** statt **set_fly_x** und **set_fly_y**) verwendet werden. Diese Befehle setzen die Encoder nicht zurück, sondern rechnen die letzten nicht-"Processing on the fly"-korrigierten Koordinatenwerte so um, dass die "Processing on the fly"-korrigierte Ausgabe mit der aktuellen Ausgabe übereinstimmt. Tritt beim Wiedereinschalten mit diesen Befehlen ein Fehler auf (z.B. weil die umgerechneten Koordinatenwerte außerhalb des virtuellen 29-Bit-Bildfelds zu liegen kämen oder es wurde in der Zwischenzeit ein anderer "Processing on the fly"-Modus aktiviert), wird ein Fehler-Bit gesetzt. Dieses Fehler-Bit kann mit **get_marking_info**(`Bit #9`) ausgelesen werden.

Falls es notwendig ist, bei der Listenabarbeitung bereits unmittelbar auf diesen Fehler zu reagieren, kann der Listenbefehl **if_not_activated** aufgerufen werden, um gegebenenfalls in eine Fehlerbehandlungsroutine zu verzweigen.

- Wenn beim Wiedereinschalten an einer anderen Verfahrtischposition fortgefahren werden soll, können **activate_fly_2d_encoder** oder **activate_fly_xy_encoder** verwendet werden. Sie setzen die Encoder zurück und simulieren eine Verfahrtischposition über die als Parameter übergebenen Encoder-Werte. Dadurch können größere Bewegungsfahrten zur Initialisierung vermieden werden.

8.6.8 Encoderresets

Bei vielen encoderbasierten "Processing on the fly"-Anwendungen (z.B. bei einem Förderband, das endlos immer in die gleiche Richtung läuft) muss der Encoder von Zeit zu Zeit (z.B. vor einer neuen Markierungssequenz) zurückgesetzt werden. Dazu kann in die Liste jeweils eine erneute "Processing on the fly"-Aktivierung mit **set_fly_x**, **set_fly_y**, **set_fly_rot** oder **set_fly_2d** eingebaut werden.

Bei encoderbasierten "Processing on the fly"-Anwendungen mit kontinuierlicher Markierung (z.B. mit einem xy-Verfahrtisch im virtuellen Bildfeld) soll dagegen meist der Zusammenhang zwischen den Encoderwerten und der absoluten Position des xy-Verfahrtisches erhalten bleiben.

Für diesen Zweck sollte **set_fly_2d** für die "Processing on the fly"-Aktivierung verwendet werden. Vor einer längeren Unterbrechung kann die "Processing on the fly"-Korrektur mit **fly_return** deaktiviert und nach der Unterbrechung mit **activate_fly_2d** oder **activate_fly_2d_encoder** wieder zugeschaltet werden. Siehe auch Kapitel 8.6.4 "2D-Bewegungen ausgleichen", Seite 261 und Kapitel 8.6.7 ""Processing on the fly"-Anwendungen synchronisieren", Seite 264. Mit **activate_fly_xy** oder **activate_fly_xy_encoder** kann die "Processing on the fly"-Korrektur ebenfalls wieder zugeschaltet werden, jedoch sind dann einige Funktionalitäten nicht mehr verfügbar, insbesondere diejenige, für die der Zusammenhang zwischen aktuellen Encoderwerten und der absoluten xy-Verfahrtisch-Position notwendig ist, siehe Abschnitt "2D-Encoderkompensation für xy-Verfahrtische", Seite 261.

Mit **set_control_mode** `Bit #9` kann zuvor eingestellt werden, ob der Zähler mit dem jeweiligen "Processing on the fly"-Befehl oder mit dem Start-Trigger zurückgesetzt wird.

8.6.9 “Processing on the fly”-Korrektur überwachen

Unter Umständen wird der Randbereich des Bildfelds erreicht, wenn:

- Ein “Processing on the fly”-Anwenderprogramm nicht auf die vorgesehene Bewegung des Werkstücks oder Scan-Systems optimiert ist
- Sich die Geschwindigkeit des Werkstücks oder Scan-Systems während einer “Processing on the fly”-Anwendung unvorgesehenerweise stark ändert

Da ein Überschreiten des zulässigen Werte-Bereichs von der RTC6 PCIe-Karte durch Clippen der Koordinatenwerte auf den jeweiligen Randbereich verhindert wird, werden dann unter Umständen nicht alle Teile des zu markierenden Musters gescannt.

Damit “Processing on the fly”-Anwendungen durch das Anwenderprogramm überwacht werden können, setzt die RTC6 PCIe-Karte bei Überschreitung des Bildfeldrandes (und Clippen der Koordinatenwerte) interne Fehler-Bits (Bit #0…Bit #3). Diese können mit **get_marking_info** ausgelesen werden.

Um Bereichsüberschreitungen zu vermeiden, sollte **get_marking_info** während des Test- oder Normalbetriebs innerhalb der “Processing on the fly”-Anwendung wiederholt aufgerufen und das Anwenderprogramm entsprechend angepasst werden.

Hinweise

- Über die Fehler-Bits Bit #0…Bit #3 kann festgestellt werden, *welche* Kante des Bildfelds überschritten wurde. Durch jede Bereichsüberschreitung wird das entsprechende Fehler-Bit gesetzt.
- Die Fehler-Bits Bit #0…Bit #3 werden bei der Initialisierung mit **load_program_file** und durch **get_marking_info** zurückgesetzt. **get_marking_info** liefert also Informationen über Fehler, die seit der letzten Initialisierung oder dem letzten Aufruf von **get_marking_info** aufgetreten sind. Nachfolgende Transformationen vom Typ #5a und #5b…#11, siehe Kapitel 7.3.6 “Ausgabewerte an das Scan-System”, Seite 189 werden hier also nicht berücksichtigt.
- Während der Einjustierphase einer “Processing on the fly”-Korrektur sollten daher unbedingt Koordinatenpunkte am Bildfeldrand angefahren und auf mögliche Begrenzungen überprüft werden.
- Über das Fehler-Bit Bit #9 kann festgestellt werden, ob während des Wiedereinschaltens einer “Processing on the fly”-Korrektur mit **activate_fly_2d** oder **activate_fly_xy** eine Bereichsüberschreitung bezogen auf das virtuelle 29-Bit-Bildfeld stattgefunden hat, siehe Kapitel 8.6.7 ““Processing on the fly”-Anwendungen synchronisieren”, Seite 264.

Kundenspezifischer Überwachungsbereich

Bei "Processing on the fly"-Anwendungen prüft die RTC6 PCIe-Karte auch auf Überschreitung eines zweiten Werte-Bereichs. Dessen Bereichsgrenzen können mit **set_fly_limits** vorgegeben werden.

Auch bei solchen Bereichsüberschreitungen setzt die RTC6 PCIe-Karte interne Fehler-Bits (Bit #4…Bit #7). Diese können mit **get_marking_info** ausgelesen werden.

Darüber hinaus kann mit den folgenden bedingten Befehlen die Ausführung beliebiger Listenbefehle davon abhängig gemacht werden, ob eine Bereichsüberschreitung des kundenspezifischen Überwachungsbereichs stattgefunden hat oder nicht:

- **if_fly_x_overflow**
- **if_fly_y_overflow**
- **if_not_fly_x_overflow**
- **if_not_fly_y_overflow**

Wenn die im Befehlsparameter angegebene Bedingung an Fehler-Bits Bit #4…Bit #7 erfüllt ist, wird der unmittelbar nachfolgende Listenbefehl ausgeführt. Anderenfalls wird er übersprungen.

Hinweise

- Eine Überschreitung des kundenspezifischen Überwachungsbereichs führt nicht unbedingt zu einem Clippen der Ausgabe-Koordinatenwerte. Ein Clippen (und Setzen der Fehler-Bits Bit #0…Bit #3) erfolgt nur dann, wenn das maximale Bildfeld (–524.288…+524.287 Bit) überschritten wird (der kundenspezifische Überwachungsbereich ist in der Regel kleiner).
- Die Fehler-Bits Bit #4…Bit #7 werden bei der Initialisierung mit **load_program_file** zurückgesetzt, aber *nicht* durch **get_marking_info**. Einzelne Fehler-Bits werden implizit durch die bedingten Befehle zurückgesetzt und können auch explizit mit **clear_fly_overflow** oder **clear_fly_overflow_ctrl** zurückgesetzt werden (siehe Befehlsbeschreibung).
- Die Fehler-Bits Bit #4…Bit #7 berücksichtigen *nicht* die Transformationen vom Typ #5a und #5b…#11, siehe Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.
- Auch für Rotations-Fly-Anwendungen ist die Überwachung eines rechteckigen Bereichs möglich, jedoch keine Überwachung eines Drehwinkel-Bereichs.

8.6.10 Schleppverzugskompensation von Encoderwerten für “Processing on the fly”-Anwendungen

- Voraussetzung:
  RTC6-Software-Paket ≥ 1.18.0

Insbesondere bei “Processing on the fly”-Anwendungen mit variabler Encodergeschwindigkeit führt der Schleppverzug der Scan-System-Galvanometerscanner zu gewissen Positionsfehlern.

Für Anwendungen mit höheren Genauigkeitsanforderungen kann deshalb mit **set_fly_tracking_error** eine Schleppverzugskompensation eingeschaltet werden.

8.6.11 “Processing on the fly”-Korrektur für die z-Achse

Die encodergesteuerte “Processing on the fly”-Korrektur für die z-Achse (hier als “FlyZ-Korrektur” bezeichnet) kann mit **set_fly_z** aktiviert werden.

Die “FlyZ-Korrektur” kann einer mit **set_fly_x**/**set_fly_y**/**set_fly_rot** aktivierten “Processing on the fly”-Korrektur für die x-Achse und y-Achse zugeschaltet werden oder auch allein wirksam sein.

Dadurch wird eine dynamische Markierung möglich, wenn sich die Werkstück-Ebene (z = 0) und das Scan-System nicht parallel zueinander bewegen.

Bei aktivierter “FlyZ-Korrektur” wird die z-Komponente der Werkstücksbewegung durch Nachregelung der z-Achse (siehe Dynamische Fokussiereinheit) gemäß den aktuellen Encodersignalen kompensiert.

Voraussetzungen

- Sowohl die Option “Processing on the fly” als auch die Option “3D” muss freigeschaltet sein.
- Die gewünschte Markierung muss statisch, also bei fest stehendem Werkstück, ordnungsgemäß funktionieren:
  - Die Schräge des Werkstücks muss durch Verwendung passender 3D-Vektor-Befehle im Anwenderprogramm abgebildet sein.
  - Die Dynamische Fokussiereinheit muss in der Lage sein, den xy-Galvanometerscanner-Bewegungen des Scan-Systems zu folgen.
- Bei bewegten Werkstücken muss die Dynamische Fokussiereinheit auch in der Lage sein, der Werkstücksbewegung zu folgen. Es ist darauf zu achten, dass der maximale z-Hub nicht überschritten wird, dass die Genauigkeit der Korrekturtabelle bezüglich Randunschärfe, Stretching etc. ausreicht und dass der Tiefenschärfebereich des Objektivs nicht überschritten wird. Bei einer Kombination mit **set_fly_rot** ist auch darauf zu achten, dass der Drehwinkel über dem Bildfeld klein genug und das Rotationszentrum genügend weit entfernt ist. Entsprechende Tests liegen in der

Verantwortung der Benutzer. SCANLAB stellt diesbezüglich keine weiteren Funktionalitäten bereit. Ein virtuelles Bildfeld für die z-Koordinate existiert *nicht*.

- Siehe auch Voraussetzungen zum Betrieb eines 3-Achsen-Scan-Systems in Abschnitt "Anschluss und Initialisierung", Seite 249.

Hinweise zur Verwendung

- Zur allgemeinen Verwendung der "Processing on the fly"-Korrektur und zur Ermittlung von Skalierungsfaktoren siehe Kapitel 8.6 ""Processing on the fly"", Seite 254.
- Beim Aktivieren der "FlyZ-Korrektur" mit **set_fly_z** kann angegeben werden, welcher der beiden Encoderzähler hierzu verwendet werden soll. Da für **set_fly_x** bereits Encoderzähler "Encoder0" und für **set_fly_y** Encoderzähler "Encoder1" verwendet wird (für **set_fly_rot** "Encoder0"), muss gegebenenfalls einer der beiden Encoderzähler doppelt verwendet werden.
- Die "FlyZ-Korrektur" kann **set_fly_x**/**set_fly_y**/**set_fly_rot** zugeschaltet werden.
- Die "FlyZ-Korrektur" kann aber auch allein verwendet werden (nur encodergesteuerte z-Variation).
- Eine aktivierte "FlyZ-Korrektur" kann mit **fly_return_z** oder **fly_return** deaktiviert werden. Dadurch werden alle "Processing on the fly"-Korrekturen für alle drei Raumrichtungen gleichzeitig deaktiviert. Als Befehlsparameter kann jeweils eine neue Ausgabeposition angegeben werden (bei **fly_return** nur die x- und y-Koordinate, bei **fly_return_z** auch die z-Koordinate).
- Eine aktivierte "FlyZ-Korrektur" kann auch mit **set_fly_z** und einem unzulässigem Parameterwert (z.B. `set_fly_z( ScaleZ=0 )`) deaktiviert werden.
  - Wenn nur eine z-Korrektur aktiviert war, findet dabei u.U. ein Sprung auf einen unkorrigierten Ausgabepunkt statt.
  - Waren (mit **set_fly_x**/**set_fly_y**/**set_fly_rot**) "Processing on the fly"-Korrekturen für andere Raumrichtungen aktiviert, so werden diese dadurch nicht deaktiviert (und es findet dann auch kein Sprung auf einen unkorrigierten Ausgabepunkt statt).
- Wird mit **set_fly_x**, **set_fly_y** und **set_fly_z** eine Korrektur für alle drei Raumrichtungen aktiviert, so führt ein anschließendes `set_fly_x( ScaleX=0 )` nur zur Abschaltung der x-Korrektur und die y- und z-Korrekturen bleiben bestehen (analoges gilt für `set_fly_y(ScaleY=0)`). Wird dagegen nur eine z-Korrektur (mit **set_fly_z**) oder eine 2D-Korrektur (mit **set_fly_z** und **set_fly_x** *oder* **set_fly_y**) aktiviert, so führt ein anschließendes **set_fly_x**, **set_fly_y** oder **set_fly_rot** mit unzulässigem Parameterwert (z.B. `set_fly_x( ScaleX=0 )`) auch zur Deaktivierung der z-Korrektur. Im letzteren Fall findet u.U. ein Sprung auf eine (teilweise) unkorrigierte Ausgabeposition statt.
- **set_fly_z** deaktiviert eine "Processing on the fly"-Korrektur mit Positionsangaben über McBSP.
- Umgekehrt wird eine mit **set_fly_z** aktivierte "Processing on the fly"-Korrektur durch eine solche deaktiviert. Hierdurch findet u.U. ein "Harter Sprung" auf eine neue Ausgabeposition statt.
- Um harte Sprünge (siehe "Harter Sprung") zu vermeiden, sollte eine "Processing on the fly"-Korrektur nur mit **fly_return_z** deaktiviert werden.
- **get_marking_info** liefert auch Informationen über mögliche Bereichsüberschreitungen während einer "FlyZ-Korrektur" (Bit #22…Bit #25).
- Zur Definition eines kundenspezifischen Überwachungsbereichs für die "FlyZ-Korrektur" und eine zugehörige bedingte Befehlsausführung, siehe Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320, werden die Befehle **set_fly_limits_z**, **if_fly_z_overflow** und **if_not_fly_z_overflow** bereitgestellt. Fehler-Bits (Bit #24…Bit #25 von **get_marking_info**) für die kundenspezifische Überwachung von FlyZ-Anwendungen können mit **clear_fly_overflow** und **clear_fly_overflow_ctrl** zurückgesetzt werden.
- Ein 3D-"Processing on the fly"-Korrektur mit Positionsangaben über McBSP kann mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktiviert werden.

8.6.12 “Fly-Erweiterung”-Befehle

- Um die “Processing on the fly”-Funktionalität flexibel und generisch nutzen zu können, stehen in RTC6-Software-Paket ≥ V1.6.1 (DLL 617, OUT 617, RBF 623) die “Fly-Erweiterung”-Befehle zur Verfügung, siehe Tabelle 2.

Tabelle 2: Anzahl der unterstützten Achsen in “Fly-Erweiterung”-Befehlen.

“Fly-Erweiterung”-Befehl[a]	Achsen
set_fly_1_axis	1
fly_return_1_axis	1
activate_fly_1_axis	1
park_position_1_axis	1
park_return_1_axis	1
wait_for_1_axis	1
set_fly_2_axes	2
fly_return_2_axes	2
activate_fly_2_axes	2
park_position_2_axes	2
park_return_2_axes	2
wait_for_2_axes	2
set_fly_3_axes	3
fly_return_3_axes	3

(a) Nur Listenbefehle, keine Kontrollbefehle.

- Die “Processing on the fly”-Funktionalität[1] selber bleibt unverändert. Lediglich deren Steuerung ist flexibler gestaltet.
- Die “Fly-Erweiterung”-Befehle sind ein flexibles System von “Processing on the fly”-Anwendungen.
  - Die meisten wirken auf:
    - nur 1, nur 2 oder alle 3 Raumachsen
    - alternativ 1 Dreh-Achse
  - Gleichzeitig können frei zugeordnet werden (Tabelle 4)
    - Encoderzähler “Encoder0” oder “Encoder1”
    - und/oder eine McBSP-Eingabe auf jede dieser Achsen, auch gemischt

(1) Wie beschrieben in Kapitel 8.6.1 “Verwendung und Initialisierung”, Seite 254…Kapitel 8.6.11 ““Processing on the fly”-Korrektur für die z-Achse”, Seite 269.

- Wichtig: “Fly-Erweiterung”-Befehle sind in keinem Fall mit den “klassischen” “Processing on the fly”-Befehlen[2] kompatibel.
- Eine “Processing on the fly”-Anwendung mit “klassischen” “Processing on the fly”-Befehlen[2] muss explizit beendet worden sein, bevor die Funktionalität einer der “Fly-Erweiterung”-Befehle aktiviert werden kann, und umgekehrt.
- Bei “Fly-Erweiterung”-Befehlen werden die Achsen gemäß Tabelle 3 angegeben.

Tabelle 3: Parameter `Axis` der “Fly-Erweiterung”-Befehle.

`Axis`-Parameterwert	Achse
1	x-Achse
2	y-Achse
3	z-Achse
4	Dreh-Achse

- Bei den “Fly-Erweiterung”-Befehlen können die Linear-Achsen `1`, `2` und `3` nicht gleichzeitig mit einer Dreh-Achse (`4`) kombiniert werden.
- Die “Fly-Erweiterung”-Befehle unterstützen 1, 2 oder 3 Achsen, siehe Tabelle 2.

(2) Siehe Fußnote, Seite 254.

Hinweise zur Verwendung der "Fly-Erweiterung"-Befehle

- `set_fly_1_axis( Axis, Mode, Scale )` ersetzt einen der "klassischen" "Processing on the fly"-Befehle:
  - **set_fly_x**
  - **set_fly_y**
  - **set_fly_z**
  - **set_fly_rot**
  - **set_fly_x_pos**
  - **set_fly_y_pos**
  - **set_fly_rot_pos**
- **set_fly_2_axes** fasst die Aktivierung von 2 beliebigen Linear-Achsen zusammen und ersetzt z.B. auch **set_fly_2d**.
- **fly_return_1_axis**, **fly_return_2_axes** und **fly_return_3_axes** erweitern die Funktionalität der "klassischen" "Processing on the fly"-Befehle **fly_return** und **fly_return_z**.
- **activate_fly_2_axes** ersetzt die "klassischen" "Processing on the fly"-Befehle
  - **activate_fly_2d**
  - **activate_fly_2d_encoder**
  - **activate_fly_xy**
  - **activate_fly_xy_encoder**
- **wait_for_1_axis** kann auf einen beliebigen Encoderwert oder einen über McBSP übertragenen Wert "warten" und ersetzt so **wait_for_encoder_mode** und **wait_for_mcbsp**.
- **wait_for_2_axes** steht für **wait_for_encoder_in_range** und **wait_for_encoder_in_range_mode** und kann zusätzlich auf McBSP-Werte warten.
- Die Laser-Steuerung (Laser ausschalten oder unverändert lassen) kann bei **wait_for_1_axis** (**wait_for_2_axes**) mit dem Parameter `LaserMode` (`LaserMode`) individuell eingestellt werden.
- Die Galvanometerscanner-Bewegung (stehen bleiben oder mitlaufen) kann bei **wait_for_1_axis** (**wait_for_2_axes**) mit dem Parameter `WaitMode` (`WaitMode`) festgelegt werden.
  Somit ist sie nicht mehr an **set_fly_2d** und **set_fly_x**/**set_fly_y** gebunden.
- Die "Fly-Erweiterung"-Befehle sind auch mit den McBSP-Übertragungen aus **set_mcbsp_in** und **set_multi_mcbsp_in** kompatibel. Die "Processing on the fly"-Korrekturen können nachträglich überschrieben werden, z.B. mit Encoder-gestützten Korrekturen. Die McBSP-Übertragung auf die Speicherstellen selbst kann jedoch nicht verändert werden.
- "Fly-Erweiterung"-Befehle sind – wie die "klassischen" "Processing on the fly"-Befehle(1) – *nicht* mit einer Online-Positionierung kompatibel.
- "Fly-Erweiterung"-Befehle können auf McBSP-Werte "warten", siehe Tabelle 4.
- Die bei "Fly-Erweiterung"-Befehlen unterstützen Modi und Encoder zeigt Tabelle 4.

(1) Siehe Fußnote auf Seite 271.

Tabelle 4: Unterstütze Modi und Encoder bei "Fly-Erweiterung"-Befehlen.

Modus (`Mode`-Parameter-wert)	**Encoder ("Quelle")**	**Auswirkung**
0	Reserviert.	Erzeugt **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
1	Encoder0.	Nur für "Processing on the fly"-Korrektur. Encoderzähler "Encoder0"-skaliert. `PreviewTime`-korrigiert.
2	Encoder1.	Nur für "Processing on the fly"-Korrektur. Encoderzähler "Encoder1"-skaliert. `PreviewTime`-korrigiert.
3	Encoder2.	Nur für "Processing on the fly"-Korrektur. Verknüpft mit Encoderzähler "Encoder0"-skaliert. `PreviewTime`-korrigiert.
4	Encoder2.	Nur für "Processing on the fly"-Korrektur. Verknüpft mit Encoderzähler "Encoder1"-skaliert. `PreviewTime`-korrigiert.
5	Reserviert.	Erzeugt **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
6	McBSP 32 Bit.	"Processing on the fly"-Korrektur mit Positionsangaben, **read_mcbsp**(0).
7	McBSP 31 Bit.	**set_mcbsp_in** (`Mode` 1, 2 oder 4) und **read_mcbsp**(0).
8	McBSP 29 Bit.	Speicherstelle für die x-Koordinate der "Processing on the fly"-Anwendung aus **set_multi_mcbsp_in**, **read_multi_mcbsp**(0).
9	Reserviert.	Erzeugt **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
10	McBSP LOW 16 Bit.	"Processing on the fly"-Korrektur mit Positionsangaben bei 2 Achsen.
11	McBSP LOW 15 Bit.	Aus **set_mcbsp_in** (`Mode`= 3).
12	McBSP 29 Bit.	Speicherstelle für die y-Koordinate der "Processing on the fly"-Anwendung aus **set_multi_mcbsp_in**, **read_multi_mcbsp**(1).
13	Reserviert.	Erzeugt **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
14	McBSP HIGH 16 Bit.	"Processing on the fly"-Korrektur mit Positionsangaben bei 2 Achsen.
15	McBSP HIGH 15 Bit.	Aus **set_mcbsp_in** (`Mode`= 3).
16	McBSP 29 Bit.	Speicherstelle für die z-Koordinate der "Processing on the fly"-Anwendung aus **set_multi_mcbsp_in**, **read_multi_mcbsp**(2).
17	Encoder0.	Nur für **wait_for_1_axis** und **wait_for_2_axes**. Nicht für "Processing on the fly"-Korrekturen. Encoderzähler "Encoder0"-unskaliert und `PreviewTime`-korrigiert.
18	Encoder1.	Nur für **wait_for_1_axis** und **wait_for_2_axes**. Nicht für "Processing on the fly"-Korrekturen. Encoderzähler "Encoder1"-unskaliert und `PreviewTime`-korrigiert.
19	Encoder0.	Wie 17, aber nicht `PreviewTime`-korrigiert.
20	Encoder1.	Wie 18, aber nicht `PreviewTime`-korrigiert.

Hinweise zu Tabelle 4

- Jeder dieser Quellen kann über **set_trigger[*]** aufgezeichnet werden.
- Modus 1…4 sowie 17…18 eignen sich auch für SCANahead-Systeme.
- Bei **wait_for_1_axis** und **wait_for_2_axes** muss Modus 17…20 angegeben werden, nicht 1…4.
- Bei den anderen "Fly-Erweiterung"-Befehlen mit Encoderangabe muss dagegen Modus 1…4 angegeben werden, nicht 17…20.
- Bei den McBSP-Modi:
  - muss der Benutzer sicherstellen, dass die Daten tatsächlich im richtigen Format übertragen werden
  - muss der Benutzer (Ausnahme: `Mode` = 6) bei den **wait_for[*]**-Befehlen zusätzlich eine entsprechende "Processing on the fly"-Korrektur einstellen (mit **set_fly[*]**, **set_mcbsp_in** oder **set_multi_mcbsp_in**)
  - "wartet" die RTC6-Karte auf die Encoderwerte auch ohne explizite Aktivierung der "Processing on the fly"-Korrektur
- Ist ein Modus unzulässig, wird ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt.
- Weitere Details finden Sie in den jeweiligen Befehlsbeschreibungen.

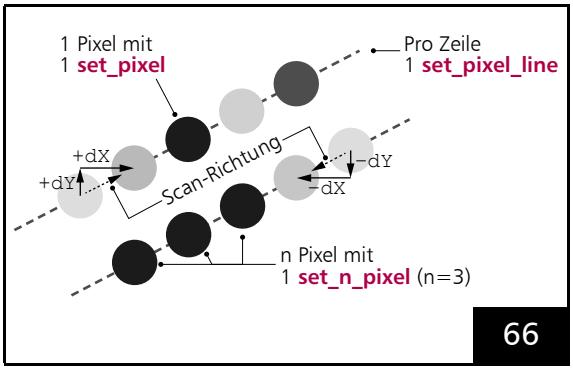
8.7 Pixelausgabemodus

Die in Kapitel 7.1 "Punkte, Linien und Bögen markieren", Seite 139 beschriebenen Vektor-Befehle dienen zum Markieren von vektorbasierten Grafikobjekten. Mit der RTC6 PCIe-Karte können außerdem Pixelbilder (Bitmaps) markiert werden. Mit geeignet eingestellten Lasern erhält man Schwarzweißbilder oder Graustufenbilder. Pixelbasierte und vektorbasierte Bilder können beliebig miteinander kombiniert werden.

8.7.1 Funktionsprinzip

Pixelbilder werden Bildzeile für Bildzeile erstellt, siehe Abbildung 66. Jede Zeile besteht aus einer Zahl von äquidistanten Pixeln und wird in einem einzelnen Durchgang erzeugt. Während dieses Durchgangs bewegt sich der Laserfokus – wie bei einem gewöhnlichen [*]mark[*]-Befehl – mit (annähernd) konstanter Geschwindigkeit über die ganze Bildzeile (die Bewegung wird in Mikroschritte zerlegt). Die einzelnen Pixel werden im Vorübergehen markiert, indem jedem Pixel an der entsprechenden Stelle ein Laserpuls zugeordnet wird. Durch Variation der Laserenergie von Pixel zu Pixel erhält man Graustufenbilder (also auch Schwarzweißbilder).

Zum Steuern des Lasers können je nach Modus und Pixelausgabefrequenz unterschiedliche Ausgänge verwendet werden, siehe Befehlsbeschreibungen.



Für jede Bildzeile wird ein eigenes **set_pixel_line** benötigt. Über nachfolgende **set_pixel**/**set_n_pixel** werden dann die einzelnen Pixel dieser Zeile definiert.

- Mit **set_pixel_line** wird eine einzelne Zeile konfiguriert:
  - Der *räumliche* Abstand zwischen den einzelnen Pixeln mit den Parametern `dX`, `dY` (bei **set_pixel_line_3d** zusätzlich `dZ`)
  - Der *zeitliche* Abstand zwischen den einzelnen Pixeln mit dem Parameter `HalfPeriod`
  - Parameter `Channel` = Modus + Port, wobei der Pixelausgabemodus über den Wert Modus und der Ausgang über den Wert Port definiert wird, siehe Tabelle 5 und 6

Hinweise

- Der Pixelausgabemodus kann mit "Processing on the fly" kombiniert werden, siehe Kapitel 8.6 ""Processing on the fly"", Seite 254.
- Der Pixelausgabemodus *sollte nicht* zusammen mit der "Automatischen Laser-Steuerung" (siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208) verwendet werden, wenn dort der Port-Ausgang, die Pulslänge (`PulseLength`) oder Ausgabeperiode (`HalfPeriod`) der Lasersteuersignale LASER1 und LASER2 angepasst werden.
- Der Pixelausgabemodus *kann nicht* mit Sky-Writing kombiniert werden.(1)
- Der Pixelausgabemodus *kann nicht* mit Wobbelmode kombiniert werden.

8.7.2 RTC6-Befehle

Bevor eine Bildzeile geschrieben wird, sollte mit einem Sprung-Befehl ein Sprung zum Zeilenanfang ausgeführt werden.

Am Anfang einer jeder Bildzeile werden dann mit **set_pixel_line** oder **set_pixel_line_3d** der Pixelausgabemodus aktiviert und zugleich Pixelabstand und Pixelausgabeperiode eingestellt (und damit auch effektiv die Geschwindigkeit, mit der die Bildzeile abgefahren wird).

(1) Der Pixelausgabemodus kann aber auch so ausgeführt werden, dass automatisch generierte (Sky-Writing-Modus 1-ähnliche) Vor- und Nachbewegungen enthalten sind. Dies erleichtert die Ausgabe von Pixelbildern mit genauer Positionierung, siehe Kapitel 8.7.4 "Synchronisation", Seite 280.

Die Pixelausgabeperiode wird über den Parameter `HalfPeriod` (halbe Pixelausgabeperiode) definiert. Das ist gleichzeitig auch die halbe Laser-Periode. Der Pixelabstand zwischen zwei benachbarten Pixeln in der Zeile und damit auch die Laufrichtung wird definiert über einen:

- 2D-Vektor (`dX`,`dY`) bei **set_pixel_line**
- 3D-Vektor (`dX`,`dY`,`dZ`) bei **set_pixel_line_3d**

Unmittelbar nach **set_pixel_line**/**set_pixel_line_3d** muss dann für jedes Pixel der Zeile separat **set_pixel** aufgerufen werden. Damit werden die Laserenergien festgelegt, die an den entsprechenden Pixelpositionen ausgegeben werden sollen.

Pixel-Pulse werden immer am LASER1-Ausgang ausgegeben, auch zum Zweck der Synchronisation des Lasers (Gating).
Wenn als Port nicht `PulseLength` angegeben ist, wird die zuletzt mit **set_laser_pulses** festgelegte `PulseLength` ausgegeben, mindestens jedoch 1/64 μs.

Alternativ zu einer Abfolge von `n` identischen **set_pixel**-Aufrufen kann **set_n_pixel** verwendet werden.

Dann wird auf der RTC6 PCIe-Karte nur ein Befehl gespeichert, beim Abarbeiten der Liste aber `n` mal ein entsprechender **set_pixel** ausgeführt. Dadurch kann die Länge einer Liste insbesondere bei Schwarzweißbildern u.U. drastisch verkleinert werden. **set_n_pixel** sollte nicht mit **n_set_pixel** (Multi-Board-Befehl von **set_pixel**) verwechselt werden.

Vor dem Ende einer Bildzeile darf kein anderer Befehl außer **set_pixel** oder **set_n_pixel** in die Liste eingefügt werden. Der erste auf einen **set_pixel_line**/**set_pixel_line_3d**-Befehl folgende Befehl in der Liste, der nicht ein **set_pixel**- oder **set_n_pixel**-Befehl ist, beendet den Pixelausgabemodus und damit die Bearbeitung der Bildzeile.

Jeder **set_pixel**/**set_n_pixel**-Aufruf, der nicht auf einen anderen **set_pixel**/**set_n_pixel**- oder **set_pixel_line**/**set_pixel_line_3d**-Aufruf folgt, wird bei der Abarbeitung ignoriert und ist dabei ein Kurzer Listenbefehl.

Hinweise

- Die Zahl der Pixel pro Bildzeile wird nur durch die Kapazität des RTC6-Listenspeichers begrenzt, siehe Kapitel 15 "Technische Spezifikationen – RTC6 PCIe-Karte", Seite 973. Insbesondere bei großen Bitmaps ist es empfehlenswert, für jede Zeile eine eigene Liste zu erstellen, damit ein Listenwechsel während der Abarbeitung einer Zeile vermieden wird.
- Jede Bildzeile muss mit **set_pixel_line** oder **set_pixel_line_3d** gestartet werden.
- Der Pixelabstand (`dX`, `dY`) in x-Richtung und y-Richtung (in Bits) (bei **set_pixel_line_3d** auch `dZ`) kann im 64-Bit-IEEE-Gleitkommaformat angegeben werden. Dies erlaubt eine Skalierung und eine Rotation des Bildes ohne Rundungsfehler.
- Die halbe Pixelausgabeperiode kann je nach Pixelausgabemodus ein beliebiges ganzzahliges Vielfaches von 1/64 μs sein. Sie ist unabhängig von der Positions-Ausgabeperiode von 10 μs, mit der die (in Mikroschritte zerlegte) Bewegung der Galvanometerscanner des Scan-Systems erfolgt. Bei sehr niedrigen Pixelausgabefrequenzen erfolgen mehrere Galvanometerscanner-Schritte je Pixel, bei hohen dagegen mehrere Pixelpulse je Galvanometerscanner-Schritt.
- Ein Pixelausgabemodus im 10 μs-Raster mit variabler Geschwindigkeit und/oder krummliniger Bahn kann mit Hilfe der micro_vector[*]-Befehle realisiert werden, siehe Kapitel 8.8 "micro_vector[*]-Befehle", Seite 285.

8.7.3 Laser-Steuerung

Je nach verwendetem Lasertyp kann die Laserenergie, die an jeder Pixelposition ausgegeben wird, variiert werden über die:

- Laserpulsdauer
- Laserleistung pro Pixel

"Classic Mode"

Der "Classic Mode" (Modus = 0, siehe `Channel`) entspricht dem Pixelausgabemodus der RTC4. Maximale Pixelausgabefrequenz: 400 kHz.

- Variation der Laserpulsdauer: **set_pixel** definiert für jedes Pixel die Länge des Pixelpulses (in Einheiten von 1/64 μs), der am LASER1-Ausgang ausgegeben wird. Zur Synchronisation siehe Kapitel 8.7.4 "Synchronisation", Seite 280.
- Variation der Laserleistung: **set_pixel** erlaubt die Angabe eines 12-Bit-wertigen Analogspannungswerts für jedes einzelne Pixel. Der Wert wird entweder am ANALOG OUT1- oder ANALOG OUT2-Ausgang ausgegeben (s.o.). Zur Synchronisation siehe Kapitel 8.7.4 "Synchronisation", Seite 280.
  set_n_pixel definiert den Analogspannungswert für `n` unmittelbar aufeinander folgende Pixel einer Zeile.

"Extended Mode"

Im "Extended Mode" (Modus = 16, siehe Seite 792) werden bei jedem **set_pixel** zwei Pixel (32-Bit-Werte) am angegebenen Port ausgegeben. Maximale Pixelausgabefrequenz: 800 kHz.

"Fast Mode"

Im "Fast Mode" (Modus = 32, siehe Seite 792) werden bei jedem **set_pixel** 4 Pixel zu je 16 Bit am angegebenen Port ausgegeben.
Maximale Pixelausgabefrequenz: 1,6 MHz.

"Ultra Fast Mode"

Im "Ultra Fast Mode" (Modus = 64, siehe Seite 792) werden bei jedem **set_pixel** acht Pixel zu je 8 Bit am angegebenen Port ausgegeben.
Maximale Pixelausgabefrequenz: 3,2 MHz.

Hinweise

- Pixelausgabefrequenzen > 800 kHz ("Fast Mode" und "Ultra Fast Mode") erfordern die Option "UFPM". Anderenfalls wird die Pixelausgabefrequenz auf 800 kHz geclippt.
- Es ist empfehlenswert, einige Experimente durchzuführen, um eine passende Gradationskurve für eine gleichmäßige Grauskala zu bestimmen. Die erhaltenen Pixel-Graustufenwerte ("Farben") hängen stark vom eingesetzten Material und vom Laser ab.
- Das LASERON-Signal geht – wie bei einem normalen [*]mark[*]-Befehl – nach Ablauf des LaserOn-Delays auf "On" und bleibt für die gesamte Pixelzeile auf "On". Nachdem das letzte Pixel ausgegeben wurde, geht es automatisch auf "Off". Siehe Kapitel 7.4 "Laser-Steuerung", Seite 192.
- Das LASER1-Signal hängt vom jeweiligen Laser-Modus und den zugehörigen Einstellungen ab, sowie vom mit **set_pixel_line**/**set_pixel_line_3d** eingestellten Modus (siehe Seite 792): Es wird ein periodisches Signal mit `HalfPeriod` aus **set_pixel_line** und einer `PulseLength` aus **set_laser_pulses** (mindestens jedoch 1/64 μs lang) ausgegeben. Es gelten die folgenden Ausnahmen:
  - Im Laser-Mode 4 und Laser-Mode 6 wird nur ein Standby-Signal ausgegeben, das nicht variiert werden kann. Leistungsänderungen sind nur möglich über die Ausgabeports ANALOG OUT1, ANALOG OUT2, 8-Bit-Digital-Ausgang (EXTENSION 2-Stiftleiste) oder 16-Bit-Digital-Ausgang (EXTENSION 1-Stiftleiste).
  - Wenn als Modus eine Ausgabe an `PulseLength` angegeben wurde, wird `PulseLength` pro Pixel variiert.
- Das LASER2-Signal folgt den Spezifikationen des jeweiligen Laser-Modus.
- Es wird kein Softstart ausgeführt, siehe Kapitel 7.4.7 "Softstart-Modus (noch nicht implementiert)", Seite 205.
- Pixelausgabemodus und Sky-Writing können nicht kombiniert werden. Jedoch kann eine separate Galvanometerscanner-Bewegung eingefügt werden, die Sky-Writing-Modus 1-ähnlich ist, siehe Abbildung 67.
- Die vor **set_pixel_line** eingestellten Werte für `HalfPeriod` und `PulseLength` werden nicht wiederhergestellt. **set_laser_pulses** muss explizit nochmals aufgerufen werden.

Tabelle 5:

<table>
<tr><th>Modus</th><th>Max. Frequenz</th><th>Anzahl Pixel pro set_pixel/ set_n_pixel</th><th>Erster Parameter von set_pixel/set_n_pixel
(Name: PortOutValue1[a])</th><th>Zweiter Parameter von set_pixel/set_n_pixel
(Name: PortOutValue2[b])</th></tr>
<tr><td>0, 256[c]</td><td>400 kHz</td><td>1</td><td colspan="2">Pixel 1 (definiert durch die Pulslänge).
Die Ausgabe erfolgt als Signal LASER1.
Port-Ausgabe (aber nicht Port 5) für Pixel 1

Pixel 1
(erster Parameter: 32-Bit; zweiter Parameter: bis 16 Bit, je nach Port)</td></tr>
<tr><td>16</td><td>800 kHz</td><td>2</td><td>Port-Ausgabe für Pixel 1
(bis zu 32 Bit pro Pixel, je nach Port)

Pixel 1</td><td>Port-Ausgabe für Pixel 2
(bis zu 32 Bit pro Pixel, je nach Port)

Pixel 2</td></tr>
<tr><td>32[d]</td><td>1,6 MHz</td><td>4</td><td>Port-Ausgabe für Pixel 1, 2
(bis zu 16 Bit pro Pixel, je nach Port)

Pixel 2 | Pixel 1</td><td>Port-Ausgabe für Pixel 3, 4
(bis zu 16 Bit pro Pixel, je nach Port)

Pixel 4 | Pixel 3</td></tr>
<tr><td>64[d]</td><td>3,2 MHz</td><td>8</td><td>Port-Ausgabe für Pixel 1, 2, 3, 4
(8 Bit pro Pixel, für alle Ports)

Pixel 4 | Pixel 3 | Pixel 2 | Pixel 1</td><td>Port-Ausgabe für Pixel 5, 6, 7, 8
(8 Bit pro Pixel, für alle Ports)

Pixel 8 | Pixel 7 | Pixel 6 | Pixel 5</td></tr>
<tr><td colspan="3"></td><td>Bit 31.....Bit 0</td><td>Bit 31.....Bit 0</td></tr>
</table>

(a) PulseLength in RTC6-Software-Paketen < V1.3.3.

(b) AnalogOut in RTC6-Software-Paketen < V1.3.3.

(c) Die Galvanometerscanner-Bewegung ist im Modus = 0 *nicht* kontinuierlich, im Modus = 256 kontinuierlich.

(d) > 800 kHz nur mit Option "UFPM" möglich.

Tabelle 6:

Ausgang	Wo an der RTC6 PCIe-Karte	
1	12-Bit-Analog-Ausgang 1	LASER-Buchse, Seite 75, ANALOG OUT1 Siehe Abbildung 17.
2	12-Bit-Analog-Ausgang 2	LASER-Buchse, Seite 75, ANALOG OUT2 Siehe Abbildung 17. MARKING ON THE FLY-Stiftleiste, Seite 85, ANALOG OUT2 Siehe Abbildung 25.
3	8-Bit-Digital-Ausgang	EXTENSION 2-Stiftleiste, Seite 83 Siehe Abbildung 24.
4	16-Bit-Digital-Ausgang	EXTENSION 1-Stiftleiste, Seite 81 Siehe Abbildung 22.
5[(a)(b)]	Pulslänge	LASER-Buchse, Seite 75, LASER1 Siehe Abbildung 17.

(a) Port = 5 kann nicht mit Modus = 0 kombiniert werden.

(b) Port = 5 kann nicht mit Modus = 256 kombiniert werden.

8.7.4 Synchronisation

Das Zeitdiagramm der Pixelausgabe für eine Bildzeile mit 3 Pixeln zeigt Abbildung 68.

Zur Vorbereitung der Laser-Steuerung schaltet **set_pixel_line**/**set_pixel_line_3d** die Signale für den "Laser active"-Betrieb aus einem vorherigen Markier-Befehl nach einem LaserOff-Delay ab (wie bei einem Sprung-Befehl) und wartet, bis der Laser tatsächlich aus ist.

Während dieser Wartezeit bleiben die Galvanometerscanner nur im "Classic Mode" (Modus = 0, siehe Seite 792) stehen. In allen anderen Modi bewegen sie sich mit der über `HalfPeriod` und Pixelabstand definierten Geschwindigkeit in äquidistanten Mikroschritten weiter. Dadurch wird es möglich, ruckfreie Ausholbewegungen und Auslaufbewegungen programmieren zu können. Anfängliche Beschleunigungsphasen können (wie bei einem normalen [*]mark[*]-Befehl) mit einem LaserOn-Delay oder einer entsprechenden Anzahl von "Leerpixeln" ausgeblendet werden (s.u.).

Danach beginnt die Pixelausgabe je nach Lasermode sofort oder nach einem Q-Switch-Delay, siehe Abbildung 68. Analogsignale an ANALOG OUT1 oder ANALOG OUT2 wechseln synchron mit der vorderen Flanke eines jeden Pixelpulses. Der Digital-Analog-Wandler braucht etwa 1,5 µs…3 µs, um ein stabiles analoges Ausgabesignal zu produzieren. Bei Pixelausgabefrequenzen über ca. 100 kHz (`HalfPeriod` < ca. 320) wird die Digital-Analog-Wandlung nicht in jedem Fall vollständig abgeschlossen. Bei solchen Pixelausgabefrequenzen muss sorgfältig überprüft werden, ob die Funktionalität für den beabsichtigten Zweck ausreicht.

Mit Bit #1 in `set_laser_control( Ctrl )` kann der Laserpuls und damit der Beginn der Digital-Analog-Wandlung um eine halbe Pixelperiode verschoben werden.

Die Pixelzeile endet mit dem ersten Listenbefehl, der kein **set_pixel** oder **set_n_pixel** ist.

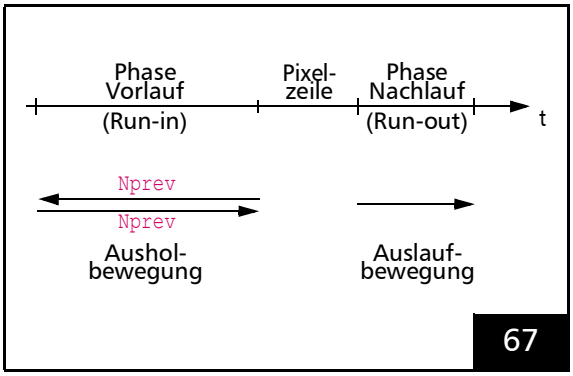
Damit der Laser auch mitten in einem 10 µs-Takt ausgeschaltet werden kann, wird automatisch nach dem letzten Pixel zusätzlich ein Default-Pixel ausgegeben. Dieses wird so oft wiederholt, bis ein angefangener 10 µs-Takt beendet ist. Dann wird der Laser endgültig ausgeschaltet.

Das Default-Pixel sollte vor **set_pixel_line**/**set_pixel_line_3d** geeignet definiert werden, um im Auslauf (run out) eine nicht-sichtbare Lasermarkierung (= "Leerpixel") zu erzielen, siehe **set_port_default** und **set_default_pixel**.

Die RTC6-Karte wartet – insbesondere bei Pixelausgabefrequenzen < 100 kHz – bis das Default-Pixel ausgegeben ist.

Die Galvanometerscanner laufen in dieser Zeit weiter, um (benutzerseitig programmierte) ruckfreie Anschlussbewegungen gewährleisten zu können. Es wird kein Scanner-Delay automatisch eingefügt. Im "Classic Mode" (Modus = 0, siehe Seite 792) bleiben die Galvanometerscanner wenige Takte stehen. Mit Modus + 256 kann dies (auch für die Vorlauf-Phase) unterdrückt werden.

Der Schleppverzug und die ausgeblendete Beschleunigungsphase bedeuten eine Pixelzeilenverschiebung, siehe Abbildung 68. Diese muss normalerweise durch einen angepassten Vorlauf (Run-in) kompensiert werden. Um dies zu erleichtern, können mit Modus + 512 Sky-Writing-Modus 1-ähnliche Bewegungen in Vorlauf-Phase und Nachlauf-Phase eingeschaltet und damit die Pixelzeile punktgenau platziert werden, siehe Abbildung 67.



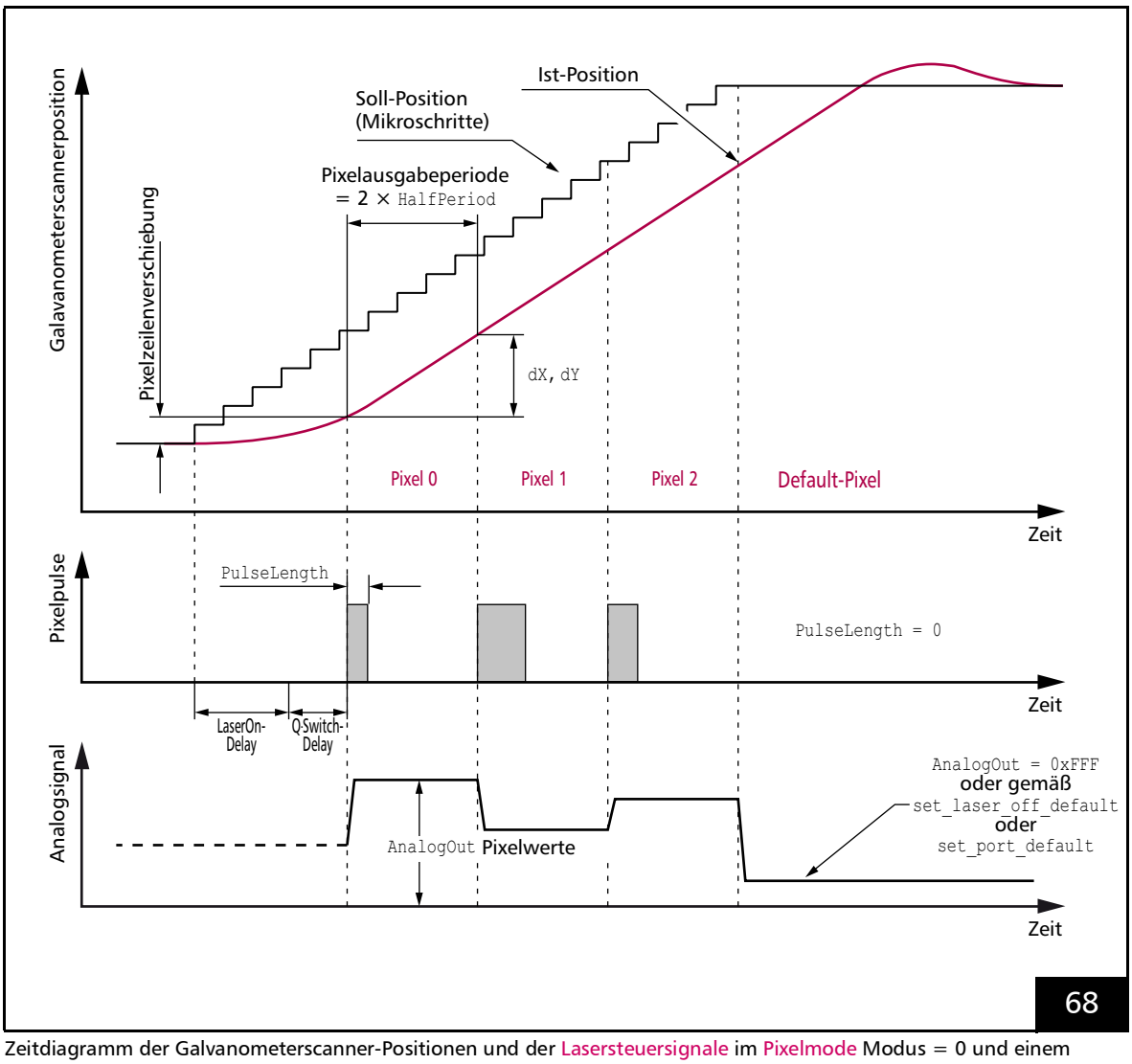
Modus + 512

Diese Bewegungen können nicht mit Sky-Writing-Modus 2 kombiniert werden. Die Dauer der Ausholbewegung muss zuvor über `Nprev` von **set_sky_writing_para** explizit festgelegt werden.(1)

(1) Bei SCANahead-Systemen müssen Sky-Writing und automatische Delayberechnung eingeschaltet sein, siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten".

Hinweise

- Mit `HalfPeriod` < `PulseLength` / 2 geht der Laser zwischen den Pixeln nicht aus.
- Beachten Sie beim Festlegen von `HalfPeriod` und Pixelabstand die Dynamik des Scan-Systems (Markiergeschwindigkeit) und die Eigenschaften des Lasersystems (Leistungsmodulation).



68

Zeitdiagramm der Galvanometerscanner-Positionen und der Lasersteuersignale im Pixelmode Modus = 0 und einem YAG-Modus, siehe Kapitel 7.4.4 "YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5", Seite 198.
Die Pixelausgabeperiode ist in diesem Beispiel etwa 4,5 Mikroschritte.

Beispiel-Code

Der Beispiel-Code muss in ein Anwenderprogramm eingebunden werden.

```
//  UINT = uint32_t
//  LONG =  int32_t

// laser parameters
UINT LaserMode = 0;
UINT LaserControl = 0x0;
LONG LaserOnDelay  = (LONG)round(100.0*64.0);  // LaserOnDelay = 100 µs
UINT LaserOffDelay = (UINT)round(100.0*64.0);  // LaserOffDelay = 100 µs
UINT HalfPeriod  = (UINT)round(5.0 * 32.0);    // Period = 5 µs, PixelFrequency = 200 kHz
UINT PulseLength = (UINT)round(0.5 * 64.0);    // PulseLength = 0.5 µs

// Pixelausgabemodus variables
enum E_PixelModes
{
    STANDARD      = 0,    // like RTC5-Mode; up to 400 kHz; 1 pixel per set_pixel command; 2 values of 32 bit per pixel
    ENHANCED      = 16,   // up to 800 kHz; 2 pixel per set_pixel command; 1 value of 32 bit per pixel
    FAST          = 32,   // up to 1.6 MHz; 4 pixel per set_pixel command; 1 value of 16 bit per pixel
    ULTRAFAST     = 64,   // up to 3.2 MHz; 8 pixel per set_pixel command; 1 value of 8 bit per pixel
    STANDARD_MOVE = 256   // like STANDARD, but with continuous galvanometer scanner movement
} Mode;

enum E_PixelPorts
{
    NO_OUT_PORT   = 0,
    ANALOG_OUT1   = 1,    // LASER-Buchse ANALOG OUT1
    ANALOG_OUT2   = 2,    // LASER-Buchse ANALOG OUT2
    DIGITAL_8Bit  = 3,    // EXTENSION 2 socket connector (8-bit)
    DIGITAL_16Bit = 4,    // EXTENSION 1-Stiftleiste (16-bit)
    PULSE_LENGTH  = 5     // not allowed for mode = 0 or mode = 256
} Port;

Port = DIGITAL_8Bit;          // pixel output port number
Mode = STANDARD;              // Pixelausgabemodus
UINT Channel = Port + Mode;   // pixel channel

UINT Number = 1;              // set_pixel repetition number

UINT PixelArray[17] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };   // port output values
UINT PortOutValue1 = PixelArray[0];
UINT PortOutValue2 = PixelArray[0];

LONG dx = (LONG)round(0.01 * CalibrationFactorXY);   // 10 um spot distance in x direction
LONG dy = (LONG)round(0.0  * CalibrationFactorXY);   //  0 um spot distance in y direction

LONG PixelLineStartPosX = (LONG)round(0.0 * CalibrationFactorXY);
LONG PixelLineStartPosY = (LONG)round(0.0 * CalibrationFactorXY);

// basic configuration
set_laser_mode(LaserMode);
set_laser_control(LaserControl);

// RTC-List
set_start_list_pos(1, 0);
set_laser_delays(LaserOnDelay, LaserOffDelay);
set_default_pixel_list(0);     // sets pixel default PulseLength
UINT PortDefault = Port - 1;   // CAUTION: port numbers differ between set_port_default/set_port_default_list and
                               //                                     set_pixel_line
set_port_default_list(PortDefault, PixelArray[0]); // sets default value for specified port
```

```
switch (Mode)
{
case STANDARD:
case STANDARD_MOVE:
{
    // standard Pixelausgabemodus up to 400 kHz (Mode = 0, 256)
    // 2 values of 32 bit per pixel
    // 1 pixel per set_pixel command
    //--------------------------------
    // |           pixel1            |
    // |   32 Bit    ||    32 Bit    |
    // | PulseLength || PortOutValue |
    //--------------------------------
    HalfPeriod  = (UINT)round(5.0 * 32.0);   // Period = 5 µs, PixelFrequency = 200 kHz
    PulseLength = (UINT)round(0.5 * 64.0);   // PulseLength = 0.5 µs

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);
    set_n_pixel(1 * PulseLength, PixelArray[1], Number);        // pixel 1
    set_n_pixel(2 * PulseLength, PixelArray[2], Number);        // pixel 2
    set_n_pixel(3 * PulseLength, PixelArray[3], Number);        // pixel 3
    set_n_pixel(4 * PulseLength, PixelArray[4], Number * 2);    // (pixel 4)*2

}
break;

case ENHANCED:
{
    // enhanced Pixelausgabemodus up to 800 kHz (Mode = 16)
    // 1 value of 32 bit per pixel
    // 2 pixel per set_pixel command
    //-----------------------------------
    // |    pixel1     ||    pixel2     |
    // |    32 Bit     ||    32 Bit     |
    // | PortOutValue1 || PortOutValue2 |
    //-----------------------------------
    HalfPeriod  = (UINT)round(2.0 * 32.0);   // Period = 2 µs, PixelFrequency = 500 kHz
    PulseLength = (UINT)round(0.5 * 64.0);   // PulseLength = 0.5 µs
    set_laser_pulses(HalfPeriod, PulseLength);

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);
    set_n_pixel(PixelArray[1], PixelArray[2], Number);        // pixel 1, pixel 2
    set_n_pixel(PixelArray[3], PixelArray[4], Number);        // pixel 3, pixel 4
    set_n_pixel(PixelArray[5], PixelArray[6], Number * 2);    // (pixel 5, pixel 6)*2
    //--> pixel output: pixel 5, pixel 6, pixel 5, pixel 6

}
break;
```

```
case FAST:
{
    // fast Pixelausgabemodus up to 1.6 MHz (Mode = 32) - needs UFPM-Option
    // 1 value of 16 bit per pixel
    // 4 pixel per set_pixel command
    //-------------------------------------------------------
    // |   pixel2   |   pixel1   ||   pixel4   |   pixel3   |
    // |   16 Bit   |   16 Bit   ||   16 Bit   |   16 Bit   |
    // |      PortOutValue1      ||      PortOutValue2      |
    //-------------------------------------------------------
    HalfPeriod  = (UINT)round(1.0 * 32.0);   // Period = 1 µs, PixelFrequency = 1 MHz
    PulseLength = (UINT)round(0.5 * 64.0);   // PulseLength = 0.5 µs
    set_laser_pulses(HalfPeriod, PulseLength);

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);
    // low 16 bit - first pixel, high 16 bit - second pixel
    PortOutValue1 = PixelArray[1] | (PixelArray[2] << 16);
    PortOutValue2 = PixelArray[3] | (PixelArray[4] << 16);
    set_n_pixel(PortOutValue1, PortOutValue2, Number);        // pixel 2 + pixel 1, pixel 4 + pixel 3

    PortOutValue1 = PixelArray[5] | (PixelArray[6] << 16);
    PortOutValue2 = PixelArray[7] | (PixelArray[8] << 16);
    set_n_pixel(PortOutValue1, PortOutValue2, Number * 2);    // (pixel 6 + pixel 5, pixel 8 + pixel 7)*2
    //--> pixel output: pixel 5, pixel 6, pixel 7, pixel 8, pixel 5, pixel 6, pixel 7, pixel 8

}
break;
case ULTRAFAST:
{
    // ultra fast Pixelausgabemodus up to 3.2 MHz (Mode = 64) - needs UFPM-Option
    // 1 value of 8 bit per pixel
    // 8 pixel per set_pixel command
    //-----------------------------------------------------------------------------------------------------------
    //|   pixel4   |   pixel3   |   pixel2   |   pixel1   ||   pixel8   |   pixel7   |   pixel6   |   pixel5   |
    //|    8 Bit   |    8 Bit   |    8 Bit   |    8 Bit   ||    8 Bit   |    8 Bit   |    8 Bit   |    8 Bit   |
    //|                 PortOutValue1                    ||                 PortOutValue2                    |
    //-----------------------------------------------------------------------------------------------------------
    HalfPeriod  = (UINT)round(0.5  * 32.0);   // Period = 0.5 µs, PixelFrequency = 2 MHz
    PulseLength = (UINT)round(0.25 * 64.0);   // PulseLength = 0.25 µs
    set_laser_pulses(HalfPeriod, PulseLength);

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);

    // lowest 8 bit - first pixel, highest 8 bit - fourth  pixel
    PortOutValue1 = PixelArray[1] | (PixelArray[2] << 8) | (PixelArray[3] << 16) | (PixelArray[4] << 24);
    PortOutValue2 = PixelArray[5] | (PixelArray[6] << 8) | (PixelArray[7] << 16) | (PixelArray[8] << 24);

    // pixel 4 + pixel 3 + pixel 2 + pixel 1, pixel 8 + pixel 7 + pixel 6 + pixel 5
    set_n_pixel(PortOutValue1, PortOutValue2, Number);

    PortOutValue1 = PixelArray[ 9] | (PixelArray[10] << 8) | (PixelArray[11] << 16) | (PixelArray[12] << 24);
    PortOutValue2 = PixelArray[13] | (PixelArray[14] << 8) | (PixelArray[15] << 16) | (PixelArray[16] << 24);

    // (pixel 12 + pixel 11 + pixel 10 + pixel 9, pixel 16 + pixel 15 + pixel 14 + pixel 13)*2
    set_n_pixel(PortOutValue1, PortOutValue2, Number * 2);
    // --> pixel output: pixel 9, pixel 10, pixel 11, pixel 12, pixel 13, pixel 14, pixel 15, pixel 16,
    //                   pixel 9, pixel 10, pixel 11, pixel 12, pixel 13, pixel 14, pixel 15, pixel 16

}
break;

default:
break;

}

set_end_of_list();
```

8.8 micro_vector[*]-Befehle

micro_vector[*]-Befehle bewegen die Galvanometerscanner mit hartem Sprung (siehe "Harter Sprung") direkt an die angegebene Position

- im 2D-Bildfeld:
  - **micro_vector_abs**
  - **micro_vector_rel**
- im 3D-Bildfeld:
  - **micro_vector_abs_3d**
  - **micro_vector_rel_3d**

Die Signale für den "Laser active"-Betrieb können individuell für jeden micro_vector[*]-Befehl über die angegebenen Laser-Delays (Parameter `LasOn` und `LasOff`) ein- und ausgeschaltet werden. Dadurch werden die mit **set_laser_delays** angegebenen Laser-Delays nicht überschrieben. Diese bleiben für normale Sprung-Befehle und [*]mark[*]-Befehle weiter gültig.

Mit den micro_vector[*]-Befehlen können Bahnkurven (Trajektorien) beliebiger Form und mit variabler Geschwindigkeit markiert werden, solange die dynamischen Grenzen des Scan-Systems nicht überschritten werden.

Hinweise

- micro_vector[*]-Befehle warten ein vorhergehendes Scanner-Delay ab, triggern selbst aber keine neuen Scanner-Delays.
- Benutzer sind selber dafür verantwortlich, die micro_vector[*]-Befehle passend zu parametrisieren:
  - Einhalten der dynamischen Grenzen des Scan-Systems
  - Vermeiden von Überkreuzungen und Überholungen bei LaserOn und LaserOff, siehe Kapitel 7.2.3 "Hinweise zur Optimierung der Delays", Seite 158
- Eine mit **set_wobbel** oder **set_wobbel_mode** festgelegte Wobbelbewegung, siehe Kapitel 8.4 "Wobbelmode", Seite 244 wird *nicht* berücksichtigt (aber auch nicht deaktiviert).
- Der Sky-Writing-Modus wird *nicht* berücksichtigt (aber auch nicht deaktiviert).
- Die Ausgabewerte werden berechnet gemäß Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189 ohne 1 und 2.

8.9 Zeitbemessene Befehle

"Normale" Vektor-Befehle und Bogen-Befehle werden so ausgeführt, dass sich der Laserfokus mit einer definierten Geschwindigkeit(1) bewegt.

Zu Beginn eines Sprungs werden die Signale für den "Laser active"-Betrieb ausgeschaltet, die Signale für den "Laser standby"-Betrieb je nach Einstellung eingeschaltet, siehe Kapitel 7.4.1 "Lasersteuersignale freigeben, aktivieren und schalten", Seite 192.

Manche Anwendungen erfordern jedoch, dass jeder Vektor-Befehl oder Bogen-Befehl ungeachtet seiner räumlichen Länge exakt dieselbe Zeit in Anspruch nimmt.

In diesem Fall ist es notwendig, eine Sprung*dauer* oder Markier*dauer* anzugeben (statt einer Sprunggeschwindigkeit oder Markiergeschwindigkeit).

Die zeitbemessenen Befehle erlauben die Angabe der Dauer eines Vektor-Befehls oder Bogen-Befehls mit einer Genauigkeit von 10 μs (der Ausgabeperiode der Mikroschritte) in einem Bereich von 10 μs bis 167.772.160 μs ($\approx$ 2,8 min).

Ein Vektor oder Bogen wird in die angegebene Anzahl (Parameter $\mathtt{T}$) von Mikroschritten aufgeteilt.

Für $\mathtt{T} \geq 5$ gilt:

$$\text{Länge } \Delta s \text{ eines Mikroschritts} = L / t$$
$$\text{mit } t = \text{Ganzzahl}((\mathtt{T} + 5) / 10)^{(2)}$$

Dadurch hängen Sprunggeschwindigkeit und Markiergeschwindigkeit automatisch von der Länge der Vektoren und Bögen ab.

Folgende zeitbemessene Befehle stehen zur Verfügung:

- Vektoren und Bögen
  - **timed_jump_abs**
  - **timed_jump_rel**
  - **timed_mark_abs**
  - **timed_mark_rel**
  - **timed_arc_abs**
  - **timed_arc_rel**
- Parametrisierte Vektoren
  - **timed_para_jump_abs**
  - **timed_para_jump_rel**
  - **timed_para_mark_abs**
  - **timed_para_mark_rel**
- 3D-Vektoren
  - **timed_jump_abs_3d**
  - **timed_jump_rel_3d**
  - **timed_mark_abs_3d**
  - **timed_mark_rel_3d**
- Parametrisierte 3D-Vektoren
  - **timed_para_jump_abs_3d**
  - **timed_para_jump_rel_3d**
  - **timed_para_mark_abs_3d**
  - **timed_para_mark_rel_3d**

Hinweise

- Nach einem zeitbemessenen Befehl wird ein "normales" Jump-Delay, Mark-Delay oder Polygon-Delay eingefügt.
- Ellipsen sind grundsätzlich nicht als zeitbemessene Befehle verfügbar.
- Mit $\mathtt{T} < 5$ μs ist ein zeitbemessener Befehl gleichbedeutend mit seinem "normalen" (= ohne **[timed_*]**) Befehl.
- Die Gesamtausführungszeit eines zeitbemessenen Befehls ist die Summe aus der angegebenen Zeit und dem zugehörigen Delay, siehe auch Kapitel 7.2.2 "Scanner-Delays", Seite 150.

(1) Sprunggeschwindigkeit oder Markiergeschwindigkeit.

(2) Für $\mathtt{T} < 5$ wird der Befehl als nicht-zeitbemessener Befehl ausgeführt, siehe Kapitel 7.1.2 "Zerlegung in Mikroschritte", Seite 143.

8.10 Automatische Selbstkalibrierung

- Diese Funktionalität betrifft keine aktuellen SCANLAB Scan-Systeme
- Vormalig wurden manche SCANLAB Scan-Systeme mit einem zusätzlichen, internen Sensorsystem zur automatischen Selbstkalibrierung (ASC-Sensorsystem, Home-In-Sensoren) ausgestattet, um höhere Anforderungen an die Langzeit-Wiederholgenauigkeit erfüllen zu können

8.10.1 Verwendung zur Drift-Kompensation

In vielen Anwendungen von Scan-Systemen ist die Langzeit-Wiederholgenauigkeit sehr wichtig, beispielsweise im Rapid Prototyping, wo sich der Bearbeitungsprozess oft über mehrere Stunden erstreckt. In solchen Laseranwendungen können die Langzeit-Drift und die Temperatur-Drift der Galvanometerscanner des Scan-Systems, die sich in einer Verschiebung (Offset-Drift) und in einer Vergrößerung oder Verkleinerung (Gain-Drift) des Bildfelds äußern, die tolerierbaren Werte überschreiten.

Bei solchen Anwendungen ist es hilfreich, diese erst nach Erreichen der Galvanometerscanner-Betriebstemperatur zu starten. Außerdem sollte das Scan-System möglichst geringen Schwankungen der Umgebungsbedingungen ausgesetzt und mit möglichst konstanter Belastung betrieben werden.

Mit dem ASC-Sensorsystem (s.o.) können die Positionsdetektoren der Galvanometerscanner kalibriert und Gain-Drift und Offset-Drift zuverlässig kompensiert werden. Die Positioniergenauigkeit bleibt so über lange Zeiträume erhalten. Verbleibende Langzeit-Drifteffekte sind von gleicher Größenordnung wie die kurzfristigen Wiederholgenauigkeiten.

8.10.2 Funktionsweise

Mit **auto_cal** kann eine Messroutine gestartet werden, mit der die exakten Ansteuerwerte für Referenzpositionen (Home-In-Positionen) bestimmt werden, die durch das interne Sensorsystem definiert sind.

Zur *Vermessung* der Drift sollte die Messroutine aufgerufen werden:

- Beim Einrichten der Maschine – um Referenzwerte für die Home-In-Positionen zu ermitteln, siehe Kapitel 8.10.3 "Referenzwerte bestimmen", Seite 289
- Während der Anwendung in geeigneten Zeitabständen – um festzustellen, ob und wie sich die Home-In-Positionen verändert haben, siehe Kapitel 8.10.4 "Kalibrierung während der Anwendung", Seite 289

Zur *Kompensation* der Drift können dann aus der ermittelten Abweichung zwischen der aktuellen Home-In-Position und dem Referenzwert – für beide Galvanometerscanner separat – passende Gain-Werte und Offset-Werte berechnet und eingestellt werden. Siehe auch #11 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.

Die Einstellung kann auch mit **set_hi** manuell erfolgen, wenn kundeneigene Messprozeduren verwendet werden sollen, siehe Abschnitt "Kundeneigene Kalibrierung", Seite 290.

Hinweise

- Vor der Ausführung einer Messroutine kann mit **auto_cal**(`Command` = 4) überprüft werden, ob:
  - das angeschlossene Scan-System überhaupt mit einem internen Sensorsystem zur automatischen Selbstkalibrierung (Home-In-Sensoren) ausgestattet ist
  - das Sensorsystem funktionsfähig ist

  Diese ASC-Hardware-Überprüfung wird auch für **auto_cal**(`Command` = 0) und falls erforderlich für **auto_cal**(`Command` = 1 und 3) automatisch durchgeführt, siehe auch **auto_cal**-Befehlsbeschreibung und **get_auto_cal**.
- Während der Ausführung der Messroutine zur Ermittlung der Home-In-Positionen sollte(n):
  - der Laser ausgeschaltet sein
  - keine anderen Befehle an das Scan-System gesendet werden
- Zur Gain/Offset-Korrektur:
  Siehe #11 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.
  Nach einer Initialisierung mit **load_program_file** oder **auto_cal**(`Command` = 2) ist gesetzt:
  - Gain = 1,0
  - Offset = 0
- Für iDRIVE-Scan-Systeme gilt zusätzlich:
  - Zu Beginn der Messroutine wird das XY2-100-Statuswort als zu übertragender Datentyp automatisch eingestellt. Am Ende der Messroutine wird der zuvor eingestellte Datentyp wieder hergestellt.
  - **auto_cal** bricht mit einem Fehler-Rückgabewert 7 ab, wenn
    - **auto_cal** für den Anschluss für ersten Scan-Kopf ausgeführt werden soll und
    - es wurde eine "Automatische Laser-Steuerung" im `Mode` = 2 (Basismode) aktiviert, siehe Abschnitt "Geschwindigkeits-abhängige Laser-Steuerung", Seite 212.

    Die "Automatische Laser-Steuerung" muss vor der automatischen Selbstkalibrierung explizit deaktiviert werden.
  - **auto_cal** bricht auch ab (Rückgabewert 1, 10 oder 11), wenn der Skalierungsfaktor mit **control_command**(`Data` = $12xx_H$) auf einen Wert < 1 eingestellt wurde, weil dann die Sensorpositionen nicht angesteuert werden können, siehe auch **control_command**(`Data` = $053F_H$). Vor der automatischen Selbstkalibrierung muss der Skalierungsfaktor auf den Wert 1 (Standardeinstellung) eingestellt worden sein mit
    - **control_command**(`Data` = 1283_H) und
    - **control_command**(`Data` = 1200_H).

8.10.3 Referenzwerte bestimmen

Die Referenzwerte werden mit **auto_cal**(`Command` = 0) bestimmt. Dadurch wird die Messroutine zur Bestimmung der aktuellen Home-In-Positionen gestartet. Diese sind dann die Referenzwerte für spätere Kalibrierungen mit **auto_cal**(`Command` = 1 oder 3). Unmittelbar nach der Bestimmung können sie mit **get_hi_pos** ausgelesen werden und stehen für eine kundeneigene Kalibrierung, siehe Abschnitt "Kundeneigene Kalibrierung", Seite 290, zur Verfügung. Die Referenzwerte werden im Flash-Speicher der RTC6-Karte gespeichert. Dadurch stehen die Referenzwerte auch nach einem Neustart wieder zur Verfügung.

Hinweise

- Nach **auto_cal**(`Command` = 0) stehen die Galvanometerscanner in der gleichen Position im realen Bildfeld wie vor dem Befehl.
- Die Referenzwerte sollten beim Einrichten der Maschine bestimmt werden, z.B. wenn das Scan-System installiert oder getauscht wird. Bei einem Tausch des Scan-Systems, für das bereits Referenzwerte ermittelt und mit **get_hi_pos** ausgelesen wurden, können diese Referenzwerte mit **write_hi_pos** direkt auf die RTC6-Karte übertragen werden.
- Während der Referenzwertbestimmung sollten anwendungstypische Bedingungen (Umgebungstemperatur, Last) herrschen und die gesamte Anlage die Endtemperatur erreicht haben. Die Referenzwertbestimmung sollte immer erst nach einer Aufwärmphase von mehr als 20 Minuten und nicht bevor das TempOK-Signal aktiviert wurde, erfolgen.
- Die Ausführung von **auto_cal**(`Command` = 0) dauert typischerweise bis zu 10 Sekunden.
- Das Abspeichern der Referenzwerte dauert mehrere ms und unterbricht dabei den 10 μs-Takt des DSP. Bei Scan-Systemen mit Takt-Überwachung (Interlock) sollte diese vorübergehend deaktiviert werden.

8.10.4 Kalibrierung während der Anwendung

Automatische Selbstkalibrierung

Eine automatische Selbstkalibrierung des Scan-Systems während der Anwendung kann mit **auto_cal**(`Command` = 1) durchgeführt werden.

Dadurch wird die Messroutine zur Bestimmung der aktuellen Home-In-Positionen gestartet. Aus den dabei ermittelten Abweichungen zu den gespeicherten Referenzwerten (siehe Kapitel 8.10.3 "Referenzwerte bestimmen", Seite 289) werden (für die x-Achse und y-Achse separat) Gain-Werte und Offset-Werte berechnet und eingestellt.

Diese Drift-Kompensation ist sofort wirksam und gilt bis:

- **auto_cal**(`Command` = 1) erneut aufgerufen wird
- sie abgeschaltet wird
  - mit **auto_cal**(`Command` = 2)
  - nach **load_program_file**

Hinweise

- Nach **auto_cal**(`Command` = 1) stehen die Galvanometerscanner in derselben Position im realen Bildfeld wie vor dem Befehl.
- Die mit **auto_cal**(`Command` = 1) gestartete Kalibrierroutine dauert (abhängig von der Stärke der Drift) typischerweise 1…2 Sekunden. Eine ASC-Hardware-Prüfung wird automatisch durchgeführt, wenn **auto_cal**(`Command` = 0 oder 4) nicht vor dem erstmaligen Aufruf von **auto_cal**(`Command` = 1) ausgeführt wurde. Dadurch verlängert sich die Ausführung von **auto_cal** um einige Sekunden.

Kundeneigene Kalibrierung

Die automatische Selbstkalibrierung erfolgt mittenzentriert, d. h. die Bildfeldmitte bleibt stabil.

Wird etwas anderes gewünscht (z.B. linker Rand soll stabil bleiben, Größe ist egal), kann die Kalibrierung auch extern berechnet und eingestellt werden.

Für eine solche kundeneigene Kalibrierung können die mit **auto_cal**(`Command` = 0) ermittelten Home-In-Positionen mit **get_hi_pos** ausgelesen werden. Daraus können kompensierende Gain-Werte und Offset-Werte berechnet und mit **set_hi** gesetzt werden. Die so eingestellte Drift-Kompensation ist sofort wirksam.

Hinweise

- **get_hi_pos** gibt die zuletzt gemessenen Home-In-Positionen zurück. Unmittelbar nach
  – **auto_cal**(`Command` = 0)
  – der Initialisierung mit **load_program_file**
  sind das die gespeicherten Home-In-Referenzwerte.
- Nach **set_hi** wird die aktuelle Position mit Sprunggeschwindigkeit korrigiert.
- Gain- und Offset-Korrekturen können auch für Systeme *ohne* Home-In-Sensoren mit **set_hi** eingestellt werden.
- Nach **auto_cal**(`Command` = 3) stehen die Galvanometerscanner exakt in der gleichen Position wie vor dem Befehl.

Weitere Hinweise zur Kalibrierung

- Die Passgenauigkeit der eingestellten Drift-Kompensation kann mit fortschreitender Zeit abnehmen. Die Kalibrierung sollte daher in geeigneten Zeitintervallen wiederholt werden. Je kürzer der Abstand zwischen den einzelnen Kalibrierungen, desto höher die erreichte Langzeit-Wiederholgenauigkeit. Typische Intervalle liegen im Minutenbereich. Gelegenheiten wie Werkstückwechsel, Zeilenvorschub o.ä. sollten zur Neukalibrierung genutzt werden.
- Die Passgenauigkeit der Kalibrierung und die dadurch erzielte Langzeit-Wiederholgenauigkeit verbessern sich zusätzlich durch konstante Umgebungs- und Lastbedingungen.
- Die Messroutine ermittelt die Home-In-Positionen mit mehreren Messungen. Wenn die Abweichungen zwischen den individuellen Messungen zu groß sind (Maximum – Minimum > 96 Bits), dann wird die Messroutine gestoppt und der Fehler 2 zurückgegeben. In diesem Fall werden für die betroffene Achse keine Referenzwerte gespeichert (`Command` = 0) und die Gain-Werte und Offset-Werte bleiben mit (`Command` = 1) unverändert(1). **get_hi_pos** liefert dann 0 für die fehlerhaften Werte zurück.
- Ein Fehler bei einem einzelnen Messzyklus kann durch kurzzeitige mechanische oder elektrische Störungen verursacht sein. Tritt bei einem einzelnen Messzyklus eine größere Streubreite auf, wird folgendes vorgeschlagen:
  – entweder es wird direkt im Anschluss noch ein weiterer Messzyklus gestartet oder
  – der Fehler wird zunächst ignoriert und die bestehenden Gain-Werte und Offset-Werte so lange weiter verwendet, bis bei einem nachfolgenden (erfolgreichen) Messzyklus neue Korrekturwerte ermittelt werden.
  Treten größere Streubreiten über viele Messzyklen hinweg auf, so kann dies auf einen Defekt im internen Sensorsystem oder auf einen anderweitigen Defekt im Scan-System hindeuten. Da aber auch dauerhafte (mechanische oder elektrische) externe Störungen oder etwa eine Verschmutzung des Scan-Systems zu einer Beeinträchtigung der automatischen Selbstkalibrierung führen können, sollte in einem solchen Fall zunächst das Scan-System auf seine allgemeine Funktionsfähigkeit und die Umgebung des Scan-Systems entsprechend überprüft werden.
- Gewisse Hardware-Fehlerbedingungen, wie Signal defekt oder nicht gefunden, werden dauerhaft gespeichert. Nachdem der Fehler behoben worden ist, muss **auto_cal** mit `Command` = 0 oder 4 aufgerufen werden, um die Fehlerbedingung wieder zu löschen. Vorher ist keine Korrektur mit `Command` = 1 oder 3 möglich.

(1) (`Command` = 0) setzt immer Gain = 1,0 und Offset = 0.

8.11 Camming

camming (von engl. cam = Nockenwelle) führt eine Markierung aus, die das Funktionsprinzip einer klassischen Nockenwelle zur Bewegung eines Ventilstößels – oder allgemeiner einer Kurvenscheibe zur Bewegung eines Hebels – simuliert. Ein Beispiel für einen Camming-Prozess zeigt Abbildung 69.

Die Galvanometerscanner-Bewegung entspricht dabei der Hebelbewegung. Sie wird als eine punktweise definierte Kurve in 2 Dimensionen als eine geschlossene Abfolge von **mark_abs**-Befehlen in eine Liste geschrieben.

Die gesamte Kurve muss in einen zusammenhängenden Listenbereich passen, siehe **config_list**. Ein Listenwechsel mit Nachladen von weiteren Kurventeilen ist hier nicht möglich.

Der "Antrieb" wird durch extern eingespeiste oder intern simulierte Encoderpulse bewerkstelligt.

Der jeweils ausgegebene Punkt ergibt sich aus den xy-Koordinaten desjenigen **mark_abs**, der sich an der Stelle `FirstPos + Index` befindet, mit `Index = Round((EncoderCurrent - EncoderStart) × Scale)`. `FirstPos` bedeutet die Listenposition des ersten **mark_abs**. `Scale` ist ein beliebig wählbarer Skalierungsfaktor. `EncoderStart` wird beim Aufruf von **camming** automatisch ermittelt. Es gibt keinen automatischen Encoderreset. Der erste ausgegebene Punkt ist immer `Index = 0`.

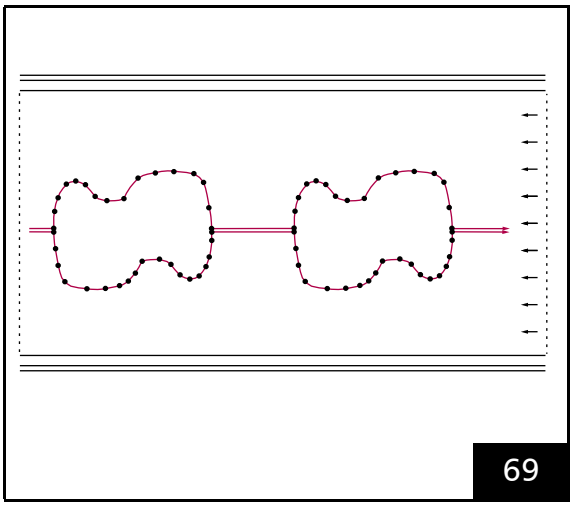
Die einzelnen Punkte werden im 10 μs-Takt als "Harter Sprung" ohne Scanner-Delay ausgeführt.

Der Abstand zwischen zwei Punkten (= "Auflösung") ist frei wählbar.

`Scale` bestimmt, wie präzise die Kurve abgetastet wird. Je größer `Scale` ist, desto grober wird die Kurve stückweise linear angenähert.

Die Anzahl der Encoderpulse pro 10 μs-Takt bestimmt zusammen mit dem Abstand der Punkte die eigentliche Markiergeschwindigkeit.

"Auflösung", `Scale` und Encodergeschwindigkeit sollten an die dynamischen Eigenschaften des angeschlossenen Scan-Systems angepasst sein.



Beispiel für einen Camming-Prozess. Ein Endlos-Werkstück wird mittels Transportsystem bewegt. Zwei Scan-Köpfe sind am Markiervorgang der Konturen beteiligt. Schematische Darstellung.

Der Camming-Prozess kann auf verschiedene Art und Weise gesteuert werden, siehe **camming** Befehlsbeschreibung:

- `Ctrl` > 0
  Der Laser wird extern gesteuert (mit **laser_signal_on_list** vor **camming** und **laser_signal_off_list** nach **camming**)
- `Ctrl` = 0
  Der Laser wird RTC6-Karten-intern automatisch gesteuert (wie bei einem normalen Polygonzug mit Berücksichtigung der Laser-Delays)

Die Kurve kann einmalig durchlaufen und dann automatisch beendet werden (`Ctrl = 0` oder `Ctrl = 1`). Die Liste wird dann mit dem nächsten Befehl nach dem Ende der Punkteliste fortgesetzt (`NPos` legt die Länge der Punkteliste fest).

Je nach Encoderrichtung kann die Punkteliste auch rückwärts durchlaufen werden. Dann wird bei Index = 0 die Kurve beendet.

Die Kurve kann unbegrenzt wiederholt werden (`Ctrl = 2` oder `Ctrl = 3`). Um harte Sprünge (siehe Harter Sprung) beim Start einer Wiederholung zu vermeiden, sollte die Punkteliste eine geschlossene Kurve darstellen. Eine unbegrenzte Wiederholung muss mit **stop_execution** oder /STOP abgebrochen werden.

Die Kurve kann jederzeit mit einem /START bei der mit **set_extstartpos** oder **set_extstartpos_list** festgelegten Adresse (üblicherweise, aber nicht notwendigerweise `FirstPos`) neu begonnen werden (unabhängig vom aktuellen Index, gegebenenfalls mit einem harten Sprung an die erste Markierposition).

Hinweise

- Ein /START während der Listenausführung wird:
  - unterdrückt im normalen Betrieb
  - nicht unterdrückt beim Camming-Prozess mit unbegrenzter Wiederholung

Mit `Ctrl = 2` wartet der Camming-Prozess am Ende der Punkteliste auf einen neuen Start. Mit `Ctrl = 3` wird automatisch von vorne begonnen (die Punkteliste wird als Ringpuffer abgearbeitet).

Der Camming-Prozess kann auch mit "Processing on the fly" kombiniert werden (mit separatem Skalierungsfaktor, was gewisse Skalierungen ermöglicht, wenn er unterschiedlich zum **camming** Parameter `Scale` gewählt wird).

Die Laser-Steuerung und Vektorkontrollierte Laser-Steuerung (**set_vector_control**) können kombiniert werden. Dann müssen anstelle der **mark_abs[*]**-Befehle **para_mark_abs[*]**-Befehle verwendet werden.
Der dort angegebene Wert wird sofort ausgegeben. Dadurch kann die Laserleistung entlang der Kurve systematisch variiert werden. Wenn die Vektorkontrollierte Laser-Steuerung nicht aktiviert ist, erfolgt keine Ausgabe der Laserleistung aus den **para_mark_abs[*]**-Aufrufen.

Zusätzlich ist eine Kombination mit der Encodergeschwindigkeits-abhängige Laser-Steuerung möglich.

Alternativ können auch **mark_abs_3d** oder **para_mark_abs_3d**, sowie **timed_mark_abs_3d** verwendet werden. Jedoch werden die z-Koordinate und der Parameter `T` bei der Ausgabe ignoriert. Andere Befehle innerhalb der Punkteliste werden ebenfalls ignoriert.
Die Punkt-Ausgabe bleibt dann für einen Takt unverändert.

Hinweise zum Testen

- Wenn die Punkteliste der Kurve mit **set_end_of_list** abgeschlossen ist und nicht die Grenze zwischen Liste 1 und Liste 2 überschreitet (siehe Kapitel 6.4 "Listen-Handling", Seite 108), kann die Form der Kurve auch mit einem normalen Ausführungsstart z.B. durch `execute_at_pointer( Pos )` getestet werden (gewissermaßen als Polygonzug, hier jedoch mit der vorgegebenen Markiergeschwindigkeit und mit Berücksichtigung des aktuell eingestellten variablen Polygon-Delays).
- Die Punkteliste sollte mit **list_return** abgeschlossen sein, wenn sie sich in einem Unterprogramm befindet und mit einem **sub_call**-Aufruf getestet werden soll.
- Wenn der harte Sprung (siehe "Harter Sprung") mitgetestet werden soll, kann auch **timed_mark_abs_3d** (mit `Time` = 10 μs) verwendet werden, nicht jedoch **timed_para_mark_abs_3d**.

8.12 Zeitmessungen

8.12.1 RTC6-Timer

RTC6 PCIe-Karten sind mit einem integrierten Timer ausgestattet, im Folgenden als "RTC6-Timer" bezeichnet. Der RTC6-Timer zählt nur Takte von Listenbefehlen. Die Zählung pausiert während durch **set_wait** oder **pause_list** initiierten Unterbrechungen.

Um die für einen bestimmten Markierungsprozess benötigte Markierungszeit zu messen, wird **save_and_restart_timer** jeweils vor und nach dem Markierungsprozess aufgerufen. **save_and_restart_timer** speichert den aktuellen RTC6-Timer-Wert und setzt ihn auf Null zurück. Die abgelaufene Zeit kann dann über **get_time** ausgelesen werden, der den RTC6-Timer-Wert zurückgibt, der beim letzten Aufruf von **save_and_restart_timer** gespeichert wurde.

Mit **get_lap_time** kann der momentane RTC6-Timer-Wert ausgelesen werden. **get_lap_time** gibt die seit dem letzten Aufruf von **save_and_restart_timer** vergangene Zeit zurück, aber ohne den RTC6-Timer zurückzusetzen. Bei länger andauerenden Markierprozessen kann so die zwischenzeitliche Laufzeit überwacht werden.

8.12.2 Zeitstempel

32-Bit-"Zeitstempel-Zähler"

Außerdem gibt es bei RTC6-Karten einen 32-Bit-"Zeitstempel-Zähler". Er beginnt mit **load_program_file** bei 0 an zu zählen und zählt unterbrechungsfrei alle 10 μs-Takte. Mit dem 32-Bit-"Zeitstempel-Zähler" kann auch bei durch **save_and_restart_timer** zurückgesetztem RTC6-Timer ein absoluter Bezug zu den einzelnen Zeitabschnitten hergestellt werden. Der 32-Bit-"Zeitstempel-Zähler" kann mit **set_trigger[*]** (Signal 52) aufgezeichnet und mit **get_value** `( 52 )` ausgelesen werden.

store_timestamp_counter / **store_timestamp_counter_list** liest den 32-Bit-"Zeitstempel-Zähler" aus der RTC6-Karte aus und speichert ihn dort als Zeitreferenz `TimeStampStorage` zwischen.

Nachfolgend kann mit **wait_for_timestamp_counter** oder **wait_for_timestamp_counter_mode** in einer Liste auf einen festen Zeitunterschied zum gespeicherten 32-Bit-"Zeitstempel-Zähler" gewartet werden. 32-Bit-"Zeitstempel-Zähler"-Überläufe werden erkannt und behoben.

Etwa 12 Stunden ist die längste Wartezeit, die mit dem 32-Bit-Offset angegeben werden kann. Wartezeiten darüber hinaus müssen in kürzere Blöcke zerlegt und mit Folgen von **store_timestamp_counter_list** und **wait_for_timestamp_counter** / **wait_for_timestamp_counter_mode** nacheinander abgearbeitet werden. Bei langen Prozesszeiten ist das nicht immer praktikabel.

64-Bit-"Zeitstempel-Zähler"

Ab DLL 624, OUT 624 stehen alternativ dazu **get_timestamp_long** mit 64-Bit-"Zeitstempel-Zähler" und **wait_for_timestamp_counter_long** mit 64-Bit-Offset zur Verfügung.

get_timestamp_long liest den 64-Bit-"Zeitstempel-Zähler" (= `TimeStampCounterLong`) aus der RTC6-Karte aus. Dessen unterer Teil (= `Parameterrückgabe TimeStampL`) und der 32-Bit-"Zeitstempel-Zähler" sind identisch. **wait_for_timestamp_counter_long** wartet auf einen absoluten 64-Bit-"Zeitstempel-Zähler"-Wert(1) (64-Bit-Offset). Das Anwenderprogramm muss die Synchronisation kontrollieren.

Hinweise

- **get_time** und **get_lap_time** berücksichtigen nur Listenausführungszeiten.
- Falls die RTC6-Karten-interne **save_and_restart_timer**-Zeitmessung mit einer externen Zeitmessung über den BUSY-Pin verglichen werden soll, dann sollte zwischen **save_and_restart_timer** und **set_end_of_list** ein **list_nop** eingefügt werden. So wird ein Scanner-Delay noch vor **set_end_of_list** abgearbeitet. Ohne **list_nop** misst **save_and_restart_timer** das Scanner-Delay mit, obwohl es erst nach **set_end_of_list** abgearbeitet wird (der BUSY-Pin also schon LOW ist).

(1) Analog zu **wait_for_timestamp_counter_mode**, jedoch wird **store_timestamp_counter** / **store_timestamp_counter_list** nicht verwendet.

8.13 *SCAN*ahead-Funktionalität

SCANahead-Systeme zeigen wegen der SCANahead-Regelung ein anderes Dynamikverhalten als Schleppverzugs-Systeme. Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten".

Bei der synchronen Ansteuerung von SCANahead-System, Laser und Peripheriegeräten ist vor allem das unterschiedliche Timing zu berücksichtigen.

Zu diesem Zweck enthält der RTC6-Befehlssatz eigens [*]scanahead[*]-Befehle zum Einschalten der SCANahead-Funktionalität auf der RTC6-Karte. Mit diesen können vorhandene RTC6-Anwenderprogramme für Schleppverzugs-Systemen sehr einfach für SCANahead-Systeme umgeschrieben werden.

Dazu müssen Sie im Wesentlichen nur wenige Programmzeilen hinzufügen:

```
set_scanahead_params( 1, 1, 1, 0, 0, 0 );
activate_scanahead_autodelays( 1 );
```

Wahlweise zusätzlich:

```
set_scanahead_line_params(n, n, n);
```

Allerdings setzt die Ausführung dieses Programmcodes voraus, dass die RTC6-Karte mit der Option "SCANa" ausgestattet ist.

8.13.1 RTC6–*SCAN*ahead-Funktionen

SCANahead-Systeme berechnen intern eine Soll-Trajektorie mit begrenzten, konstanten Sollbeschleunigungen, siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten". Dadurch variiert die Beschleunigungszeit abhängig von der Geschwindigkeitsänderung. Daraus ergibt sich, dass auch die für die Berücksichtigung der Beschleunigungszeit benötigten Scanner-Delays und Laser-Delays von der Geschwindigkeit abhängen. Die RTC6-Karte kann die benötigten Scanner-Delays und Laser-Delays automatisch berechnen.

Dazu muss mit 2 RTC6-Befehls-Aufrufen das Einschalten der SCANahead-Funktionalität auf der RTC6-Karte und damit der zu SCANahead-Systemen passenden Ansteuerung erfolgen.

(1) set_scanahead_params`( Mode =1 )`

Die RTC6-Karte liest die relevanten Informationen aus dem SCANahead-System aus, bearbeitet diese (Korrekturdatei) und hält diese zur weiteren Verwendung vor. Außerdem wird der Laser-Steuerungsteil der RTC6-Karte darauf vorbereitet, die `PreviewTime` zu berücksichtigen. Das betrifft z.B. LASER DIGITAL OUT1/2 etc., siehe Hinweise zum Ansteuern von Peripherie, Seite 304.

(2) activate_scanahead_autodelays`(1)`

Die RTC6-Karte wird instruiert:

- Die Ergebnisse aus (1) zu verwenden
- Im Quellcode mit **set_scanner_delays** und **set_laser_delays** gesetzte Delays zu ignorieren
- Automatisch die (für eine hohe Konturtreue) optimalen, geschwindigkeitsabhängigen Scanner-Delays und Laser-Delays zu berechnen(1)
- Diese dann auch dynamisch in Echtzeit während der Listenausführung einzusetzen
- Mit Sky-Writing werden ebenso alle benötigte Parameter automatisch berechnet (z.B. Vorlauf-Bewegungen und Nachlauf-Bewegungen; englisch run-in and run-out motions), siehe auch Hinweise zu Sky-Writing, Seite 304

(1) Bei allen RTC6-Befehlen die Delays erfordern: Sprung-Befehle, Markier-Befehle, Bogen-Befehle.

So können ohne Aufwand sofort Markierungen mit hoher Konturtreue erzeugt werden[1] (ohne selber Scanner-Delays und Laser-Delays zu ermitteln und anzupassen).

Selbstverständlich kann das Markierergebnis mittels entsprechender RTC6-Befehle weiter beeinflusst werden:

- Auf die Scanner-Delays wirken die Parameter `CornerScale` und `EndScale`[2] von **set_scanahead_line_params** und **set_scanahead_line_params_list**.
  Die Wirkung der Parameter zeigt Abbildung 70.
  Es kann zwischen optimaler Konturtreue und optimaler Prozessgeschwindigkeit auf Kosten der Konturtreue gewählt werden (*Prozessoptimierung*).
- Auf die Laser-Delays wirkt der Parameter `AccScale`[3] von **set_scanahead_line_params** und **set_scanahead_line_params_list**.
  Die Wirkung des Parameters zeigt Abbildung 71 am Beispiel einer Linie bei drei unterschiedlichen Parameterwerten: Der Energieeintrag an Linienenden wird beeinflusst, indem die Beschleunigungsphasen teilweise oder ganz ein- oder ausgeblendet werden.
- Zur *Feinabstimmung* der Laser-Delays gibt es **set_scanahead_laser_shifts** und **set_scanahead_laser_shifts_list**. Damit können eine Zeitverschiebung der Laserschaltzeitpunkte eingestellt werden, z.B. zur Kompensation von Signallaufzeiten[4].

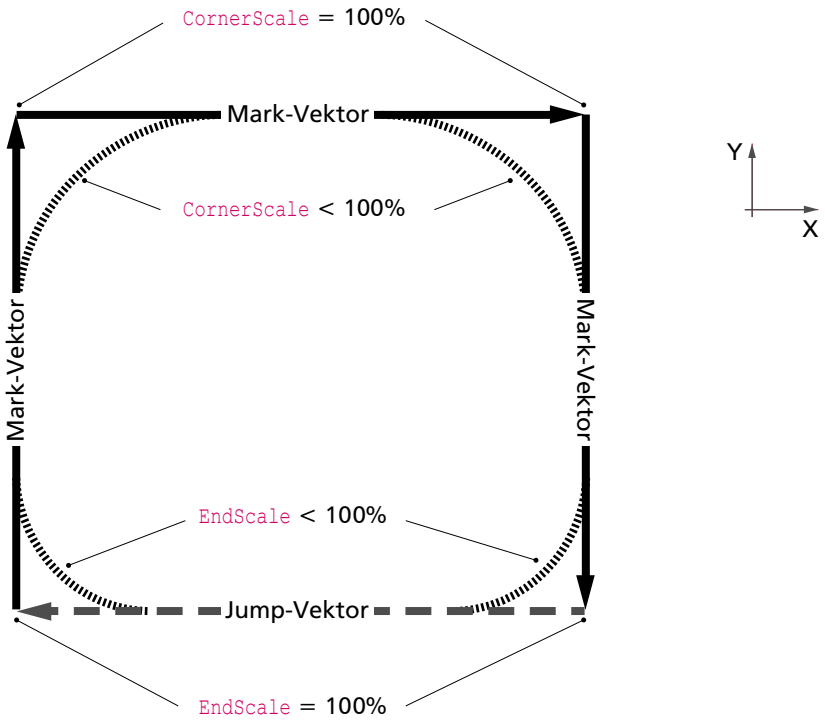
(1) Vorausgesetzt dass die angegebene Markiergeschwindigkeit sinnvoll ist. Die Sprunggeschwindigkeit wird hingegen an die Sprungweite angepasst.

(2) Diese Parameter wirken nur auf die Positionierung der Galvanometerscanner. Sie wirken *nicht* auf die Laser-Schaltzeitpunkte!

(3) Dieser Parameter wirkt nur auf die Laser-Schaltzeitpunkte.
Er wirkt *nicht* auf die Positionierung der Galvanometerscanner!

(4) Für ≥ DLL 605 gilt: Die Übertragungsverzögerung von RTC6-Karte zum Scan-System (20 µs) wird automatisch berücksichtigt.

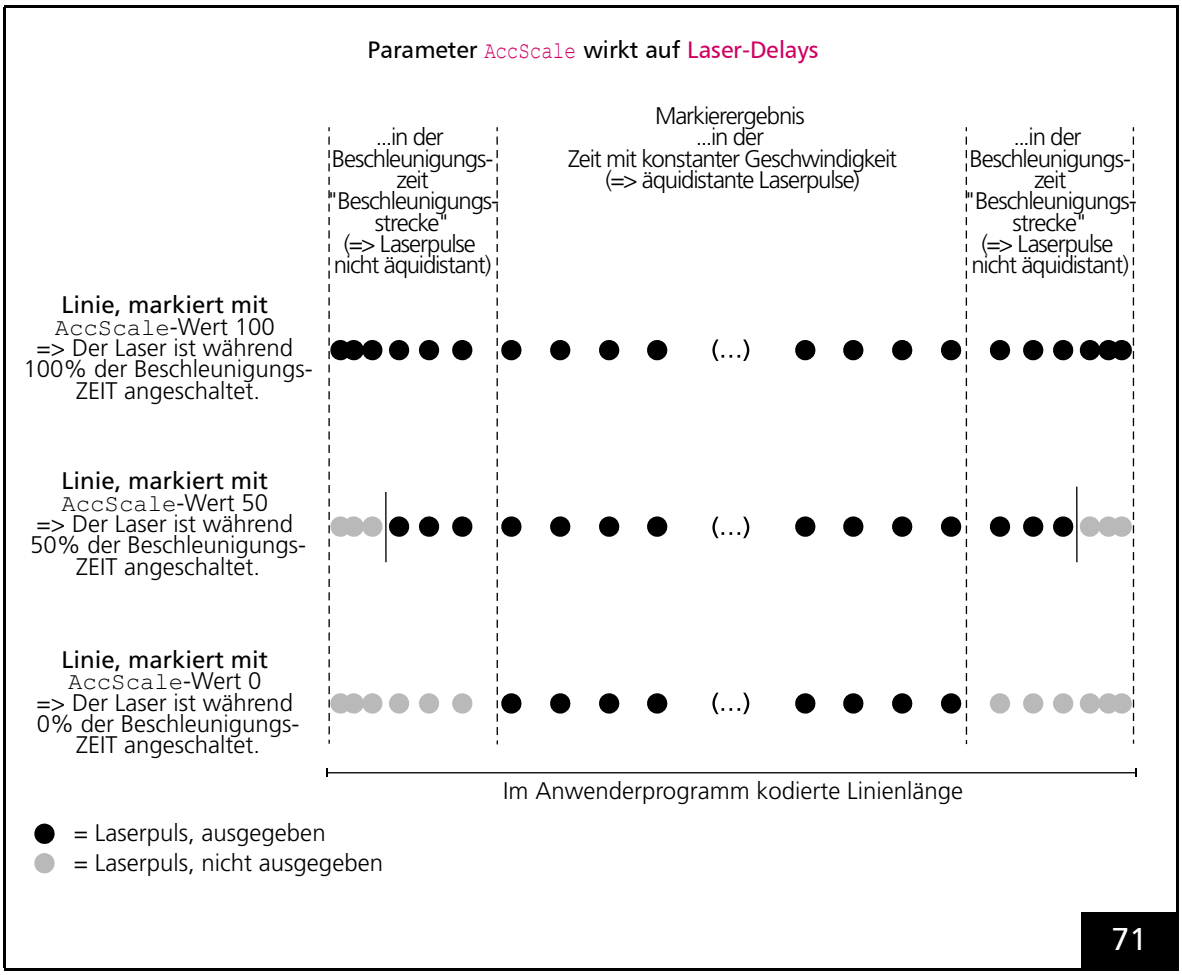
Parameter CornerScale und EndScale wirken auf Scanner-Delays



Parameter CornerScale	Steuert die Markiergenauigkeit bei Wechseln von Mark-Vektor zu Mark-Vektor (“Eckenabrundung”). 100% = scharfe Ecken.
Parameter EndScale	Steuert die Markiergenauigkeit bei Wechseln von Mark-Vektor zu Sprung-Vektor (“Linienende”) und bei Wechseln von Sprung-Vektor zu Mark-Vektor (“Linienanfang”). 100% = gerade Linienenden/Linienanfänge.

70

set_scanahead_line_params und **set_scanahead_line_params_list**: Die Parameter CornerScale und EndScale wirken auf Scanner-Delays.



set_scanahead_line_params und **set_scanahead_line_params_list**: Der Parameter AccScale wirkt auf Laser-Delays.

Bei AccScale-Werten < 100 sind die markierten Linienlängen kürzer, als sie im Anwenderprogramm codiert sind. Bei diesen Werten ist der Laser während der Beschleunigungszeiten nicht immer an, d.h. Teile der Beschleunigungsstrecken werden nicht markiert. Die Längen der Beschleunigungsstrecken sind proportional zum Quadrat der Beschleunigungszeit. Der *nicht markierte* Anteil je Beschleunigungsstrecke entspricht (1 – AccScale-Wert / 100)² × 100%. Bei einem AccScale-Wert 50 werden also je 25% der Beschleunigungsstrecken nicht markiert.

Hinweis: Um die im Anwenderprogramm codierte Linienlänge *vollständig und mit äquidistanten Laserpulsabständen* zu erhalten, müssen Sie Sky-Writing verwenden.

8.13.2 Schnelleinstieg für erfahrene Benutzer

In diesem Kapitel:

- Beispiel-Code (C++), Seite 298
- Bestehende Anwenderprogramme für SCANahead-Systeme anpassen – Grundlegende Schritte, Seite 300
- Bestehende Anwenderprogramme für SCANahead-Systeme anpassen – Einschränkungen berücksichtigen, Seite 301
- Anwenderprogramme anpassen – Weitere Schritte, Hintergrundinformationen zu SCANahead-System-bezogenen Funktionalitäten, Seite 301

Beispiel-Code (C++)

Der grundsätzliche Aufbau eines RTC6-Anwenderprogramms für ein SCANahead-System zum Initialisieren der `RTC6DLL.dll` und der RTC6-Karte ist im Folgenden beispielhaft in Form von Quellcodeabschnitten dargestellt (d.h. es ist kein vollständiger Programmcode).

In diesem Beispiel wird implizites Linken verwendet, daher werden die Dateien `RTC6impl.h` und `RTC6DLL.lib` benötigt. Zusätzlich muss die `RTC6DLL.dll` vom Programm geladen werden können. Falls das Betriebssystem die `RTC6DLL.dll` beim Programm-Startup nicht findet, antwortet es mit einer entsprechenden Fehlermeldung und beendet das Programm.

```
// BEISPIEL-QUELLCODE-SCHNIPSEL: EIN QUADRAT MARKIEREN.
//
// RTC6-Headerdatei zum impliziten Linken mit RTC6DLL.dll.
// Außerdem mit der Visual C++-Importbibliothek RTC6DLL.lib linken,
// um eine ausführbare Datei zu kompilieren
#include "RTC6impl.h"

// scanahead line params
// Polyline: 100%: max Genauigkeit, 0% min Ausführzeit
CornerScale = 100;
// Mark, Jump: 100%: max Genauigkeit, 0% min Ausführzeit
EndScale = 100;
```

```
// 100%: Laser ist an während Beschleunigung und Abbremsen
//   0%: Laser ist aus während Beschleunigung und Abbremsen
AccScale = 100;

// Variables Polygon-Delay oder Sky-Writing einschalten/auschalten
VariablePolygonDelay = 0;
SkyWriting = 0;
int main()
{

// Initialisieren
init_rtc6_dll();
// lade RTC Firmware
if( 0 != load_program_file( NULL ) ) { return -1; }
if( 0 != load_correction_file( NULL, 1, 2 ) ) { return -2; }
set_mark_speed_ctrl( 25000 );
set_jump_speed_ctrl( 50000 );

// Laser-Steuerung
// high-aktive Laser-Signale
set_laser_control( 0 );
// Modus 1 unterstützt YAG
set_laser_mode( 1 );
set_firstpulse_killer( 640 );
set_laser_pulses_ctrl( 320, 128 );

// SCANahead: Berechne Delays automatisch.
// SCANahead-System muss angeschlossen und mit Strom versorgt sein.
//
// platziere irgendwo nach load_correction_file
set_scanahead_params( 1, 1, 1, 0, 0, 0 );
activate_scanahead_autodelays( 1 );
set_scanahead_line_params( CornerScale, EndScale, AccScale );

// Fine-Tuning LaserON und LaserOFF. 1 Bit = 1/64 µs
set_scanahead_laser_shifts( 1600, 1600 );

// Sky-Writing und Variables Polygon-Delay
set_delay_mode( VariablePolygonDelay, 0, 2^30, 0, 0 );
set_sky_writing_para( SkyWriting, 0, 0, 0 );
set_sky_writing_mode( 3 );
set_sky_writing_limit( 0.5 );

// Erzeuge Liste und führe sie aus
set_start_list( 1 );
jump_abs( 50000, -50000 );
mark_abs( 50000, 50000 );
mark_abs( -50000, 50000 );
mark_abs( -50000, -50000 );
mark_abs( 50000, -50000 );
jump_abs( 0, 0 );
set_end_of_list();
execute_list( 1 );
}
// ENDE BEISPIEL-QUELLCODE-SCHNIPSEL.
```

Bestehende Anwenderprogramme für SCANahead-Systeme anpassen – Grundlegende Schritte

Es wird davon ausgegangen, dass es sich um ein vollständiges RTC6-Anwenderprogramm oder RTC5-Anwenderprogramm zum Ansteuern eines Schleppverzugs-Systems, z.B. intelli*SCAN* III, handelt.

Mit diesem Vorgehen sollten die vorgenommenen Änderungen für einen erfolgreichen Funktionsablauf Ihres Anwenderprogramms bereits ausreichend sein.

Entsprechende Quellcodeabschnitte für "Variables Polygon-Delay" und Sky-Writing sollten damit auch lauffähig sein.

(1) Fügen Sie am Anfang des Anwenderprogramms, jedoch unbedingt nach **load_correction_file**, den **set_scanahead_params**-Aufruf ein (Parameter siehe Befehlsbeschreibung).

Damit aktivieren Sie die SCANahead-Funktionalität der RTC6-Karte (inkl. Laser-Steuerung). Mit `Mode = 1` werden die SCANahead-System-spezifischen Informationen `PreviewTime`, `Vmax` und `Amax` direkt vom Scan-System ausgelesen. Deswegen muss zur Laufzeit des Anwenderprogramms ein SCANahead-System an die RTC6-Karte angeschlossen und eingeschaltet sein.

(2) Fügen Sie **activate_scanahead_autodelays** `(1)` ein.
Damit schalten Sie die automatische Berechnung der Scanner-Delays und Laser-Delays ein.
Alle Aufrufe von **set_scanner_delays** und **set_laser_delays** in den nachfolgenden Quellcodezeilen bleiben wirkungslos, siehe Hinweise, Seite 355 unter **activate_scanahead_autodelays**.

(3) Wenn der Ausgangspunkt ein RTC5-Anwenderprogramm ist, passen Sie das Anwenderprogramm zusätzlich wie folgt an:
- `init_rtc5_dll` ändern zu `init_rtc6_dll`
- `rtc5_count_cards` ändern zu `rtc6_count_cards`
- `free_rtc5_dll` ändern zu `free_rtc6_dll`
- *Wichtig*
  - `set_rtc4_mode` *NICHT* zu `set_rtc6_mode` ändern
  - `set_rtc5_mode` *NICHT* zu `set_rtc6_mode` ändern

  Wenn weder **set_rtc4_mode** noch **set_rtc5_mode** vorhanden ist, dann ist bei RTC5-Karten **set_rtc5_mode** der Default-Zustand. Fügen Sie **set_rtc5_mode** in das RTC6-Anwenderprogramm ein, wenn bestimmte Parameterwerte automatisch von RTC5-Einheiten in RTC6-Einheiten umgerechnet werden sollen, siehe Abschnitt "RTC5-Kompatibilitätsmodus", Seite 176.

(4) Prüfen Sie, ob Ihr Code von den Einschränkungen betroffen ist, die in Kapitel "Bestehende Anwenderprogramme für SCANahead-Systeme anpassen – Einschränkungen berücksichtigen", Seite 301 genannt sind.

Bestehende Anwenderprogramme für SCANahead-Systeme anpassen – Einschränkungen berücksichtigen

Bei der weiteren Anpassung des Anwenderprogramms für das SCANahead-System müssen Sie die im Folgenden genannten Punkte berücksichtigen.

- Nicht nutzbare Funktionalitäten von RTC6-Karte und RTC6-DLL, Seite 301
- Nicht nutzbare iDRIVE-Funktionalitäten, Seite 301

Nicht nutzbare Funktionalitäten von RTC6-Karte und RTC6-DLL

- Folgende RTC6-Funktionalitäten können *nach der Aktivierung der* SCANahead-Funktionalität *mit* **set_scanahead_params** *nicht* genutzt werden (Stand ≤ DLL 615):
  - **load_varpolydelay**
  - **laser_on_pulses_list** kann nicht für SCANahead-Systeme verwendet werden, wenn externe Signalpulse am DIGITAL IN1 erforderlich sind (**laser_on_list** kann dagegen mit SCANahead-Systemen verwendet werden)

Nicht nutzbare i*DRIVE*-Funktionalitäten

- Folgende i*DRIVE*-Funktionen stehen bei SCANahead-Systemen nicht zur Verfügung:
  - Die Änderung der effektiven Kalibrierung (**SetPositionScale**)
    `control_command( Data = 12nn`$_H$ `)`
- Folgende i*DRIVE*-Funktionen stehen bei SCANahead-Systemen mit Firmware-Version ≤5.05.4 (siehe Hinweise, Seite 24) nicht zur Verfügung:
  - Das Umschalten zwischen Tunings (**SelectControlDefinition**)
    `control_command( Data = 11nn`$_H$ `)`

Anwenderprogramme anpassen – Weitere Schritte, Hintergrundinformationen zu SCANahead-System-bezogenen Funktionalitäten

Zur weiteren Anpassung des Anwenderprogramms für das SCANahead-System sind zusätzlich die folgenden Punkte zu berücksichtigen:

- Hinweise bei Verwendung der RTC6-Befehle für Scanner-Delays und Laser-Delays, Seite 302
- Hinweise zu [*]arc[*]-Befehlen, Seite 303
- Hinweise zu variablen Polygon-Delays, Seite 303
- Hinweise zu Sky-Writing, Seite 304
- Hinweise zu Koordinatentransformationen, Seite 304
- Hinweise zum Ansteuern von Peripherie, Seite 304
- Hinweise zu gesendeten Datensignalen, Seite 305

Hinweise bei Verwendung der RTC6-Befehle für Scanner-Delays und Laser-Delays

Bei Verwendung der RTC6-Befehle zum Synchronisieren von Scan-Bewegung und Lasersteuersignale muss beim SCANahead-System Folgendes beachtet werden:

- Alle relevanten *SCAN*ahead-Parameter sind werksseitig (Artikelnummer-spezifisch) auf dem SCANahead-System gespeichert. Obwohl bei **set_scanahead_params**`( Mode = 2 )` auch andere Werte zugelassen sind, ist es empfehlenswert, keine anderen als die ausgelesenen Werte aus dem SCANahead-System zu verwenden. `Mode = 2` ist hauptsächlich für Software-Entwicklung ohne direkt angeschlossenen SCANahead-System vorgesehen.
- Beim SCANahead-System hängt die Ist-Position der Scan-Bewegung den Ansteuerwerten zeitlich um die `PreviewTime` hinterher(1). Für eine synchrones Steuern des Lasers muss die `PreviewTime` auch bei den Lasersteuersignalen berücksichtigt werden.
  Das könnte händisch erfolgen, indem die Parameter von **set_scanner_delays** und **set_laser_delays** entsprechend angepasst werden. Allerdings müssten Sie dabei die geschwindigkeitsabhängige Beschleunigungszeit des SCANahead-Systems berücksichtigen.
  Alternativ und vorzugsweise verwenden Sie einfach **activate_scanahead_autodelays** zur automatischen Berechnung der Scanner-Delays und Laser-Delays. Dann gilt:
  - Die bisherigen Parameter von **set_scanner_delays** und **set_laser_delays** werden durch **`activate_scanahead_autodelays`**`( Mode = 1 )` unwirksam und nicht verändert. Sie sind sofort wieder wirksam, sobald die automatische Delay-Berechnung mit **`activate_scanahead_autodelays`**`( Mode = 0 )` deaktiviert wird.
  - `set_scanahead_params( Mode = 1 oder 2, HeadNo, TableNo, PreviewTime, Vmax, Amax )` definiert die Parameter zur automatischen Berechnung der Scanner-Delays und Laser-Delays zusammen mit **`activate_scanahead_autodelays`**`( Mode = 1)`.
- Bei Bedarf kann der Durchsatz der lasermarkierten Werkstücke bei gleichzeitiger Verminderung der Qualität gesteigert werden (“Prozessoptimierung”), siehe:
  - `set_scanahead_line_params( CornerScale, EndScale, AccScale )`
  - Abbildung 70
  - Abbildung 71
- Zur *Feinabstimmung* der Laser-Delays (z.B. um Signallaufzeiten zum Laser auszugleichen oder das Schaltverhalten des Lasersystems zu berücksichtigen), siehe:
  `set_scanahead_laser_shifts( dLasOn, dLasOff )`

(1) Dieser Zeitversatz unterscheidet sich sowohl quantitativ als auch qualitativ vom Schleppverzug bei Schleppverzugs-Systemen. Siehe Handbuch “excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten”.

Hinweise zu [*]arc[*]-Befehlen

- SCANahead-Systeme haben einen nominellen Schleppverzug von 0[(1)]. Diesbezügliche Effekte (z.B. Einschnürung bei kleinen, schnellen Kreisbewegungen) gibt es nicht, solange die Maximalbeschleunigung des SCANahead-Systems nicht überschritten wird (z.B. wenn im Quellcode die Geschwindigkeit im Verhältnis zum Kreisdurchmesser zu groß angegeben wurde).
- Zum Abschätzen der Maximalbeschleunigung im Bildfeld kann die Maximalbeschleunigung in Programmier-Bits (Bildfeld-Koordinaten) mit `get_scanahead_params( HeadNo = 256 )` abgefragt werden. Zum Umrechnen in m/s² siehe Hinweise, Seite 469.
- Der zulässige Geschwindigkeitsbereich für einen gegebenen Kreisradius kann abgeschätzt werden mit:
  $v^2/r <$ `Amax`[(2)(3)(4)]
  - Beachten Sie, dass **arc_abs**, **arc_abs_3d**, **arc_rel**, **arc_rel_3d** bei der Zerlegung in Mikroschritte runden. Dadurch kann die von der RTC6 an das SCANahead-System kommandierte Geschwindigkeit von der (in **set_mark_speed** angegebenen) Markiergeschwindigkeit abweichen. Wenn die aus der kommandierten Geschwindigkeit resultierende Beschleunigung die Maximalbeschleunigung `Amax` des SCANahead-Systems übersteigt, dann sind Kreise kleiner als programmiert.
  - Bei **timed_arc_abs** und **timed_arc_rel** müssen Sie – um eine Rundung zu verhindern – die Ausführungszeit in Vielfachen von 10 μs angegeben.

Hinweise zu variablen Polygon-Delays

- Variable Polygon-Delays wirken bei SCANahead-Systemen wie bei Schleppverzugs-Systemen.
  Zum Aktivieren der Variablen Polygon-Delays muss ebenso **set_delay_mode**(`VarPoly` > 0) aufgerufen werden. Dessen Parameter `EdgeLevel`, `MinJumpDelay` und `JumpLengthLimit` bleiben ohne Wirkung, wenn die automatische Delay-Berechnung (**activate_scanahead_autodelays**) eingeschaltet ist. Sie werden aber sofort wirksam, wenn die automatische Delay-Berechnung ausgeschaltet wird. Der Parameter `DirectMove3D` ist dagegen immer wirksam.
- **load_varpolydelay** hat bei einem SCANahead-System keine Wirkung.
- Es stehen keine Benutzer-definierten "Variable Polygon-Delay"-Tabellen zur Verfügung.

(1) Die um die `PreviewTime` verzögerte Ausgabe ist kein Schleppverzug im üblichen Sinne.

(2) `Amax` mit `get_scanahead_params`(`HeadNo` = 256) zurücklesen. Beachten Sie die Einheiten.

(3) Gilt für Frequenzen ≤ 6,25 kHz (excelli*SCAN* 14) sowie ≤ 4,00 kHz (excelli*SCAN* 20, excelli*SCAN* 30);
$f = v / (2\pi \times r)$.

(4) Gilt für excelli*SCAN* mit beliebigem Objektiv. Setzen Sie entweder Ansteuer-Bits oder Programmier-Bits in die Formel ein. Siehe auch zweiter Listenpunkt.

Hinweise zu Sky-Writing

- Sky-Writing wirkt bei SCANahead-Systemen wie bei Schleppverzugs-Systemen. Alle 3 Modi sind verfügbar.
  Zum Aktivieren von Sky-Writing muss ebenso **set_sky_writing_para(** `Timelag` > 1/4 µs) oder **set_sky_writing(** `Timelag` > 1/4 µs) aufgerufen werden.
  Ist die automatische Delay-Berechnung (**activate_scanahead_autodelays**) eingeschaltet, haben `Timelag` (außer für die eigentliche Aktivierung von Sky-Writing) und die Parameter `Nprev` und `Npost` keine Wirkung. Sie werden aber sofort wirksam, sobald die automatische Delay-Berechnung ausgeschaltet wird. Dagegen ist der Parameter `LaserOnShift` immer wirksam.
- Wenn Sky-Writing eingeschaltet ist, wirkt nur der Parameter `EndScale`(1).
- Die anderen Parameter(1) `CornerScale` und `AccScale` werden nicht angewendet, außer es findet *keine* Sky-Writing-Bewegung statt:
  – Wenn Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 aktiviert ist und der Grenzwinkel unterschritten ist, wirkt `CornerScale`.

Hinweise zu Koordinatentransformationen

Beachten Sie beim Verwenden von RTC6-Befehlen zu Koordinatentransformationen das Folgende:

- Eine Koordinatentransformation soll Form und Größe(2) der Markierung *nicht* verändern, d.h. idealerweise ist die (2 x 2)-Gesamtmatrix M (siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236) eine reine Drehmatrix.

Hinweise zum Ansteuern von Peripherie

Beachten Sie beim Verwenden von RTC6-Befehlen zur Ansteuerung von Peripherie (Pins für ANALOG OUT1 und ANALOG OUT2 an der LASER-Buchse, EXTENSION 1-Stiftleiste und EXTENSION 2-Stiftleiste) das Folgende.

Wenn die SCANahead-Funktionalität der RTC6-Karte mit **set_scanahead_params** aktiviert ist (`PreviewTime` > 0), gilt:

(1) Die Kontrollbefehle **write_io_port**, **write_8bit_port**, **write_da_x**, **write_da_1** und **write_da_2** werden sofort ausgeführt ("asynchron")

(2) Die Listenbefehle **write_io_port_list**, **write_8bit_port_list**, **write_da_x_list**, **write_da_1_list** und **write_da_2_list** werden um die `PreviewTime` verzögert ausgeführt, d.h. gleichzeitig mit den Galvanometerscannern ("achsensynchron")

(3) Für eine "lasersynchrone" Ausgabe einer Laserleistungssteuerung müssen Sie **set_laser_power** statt der Listenbefehle aus (2) verwenden.

Wenn die SCANahead-Funktionalität der RTC6-Karte *nicht* aktiviert ist (`PreviewTime` = 0), gilt:

- asynchrones Ausführen (1) und achsensynchrones Ausführen (2) sind identisch

Mit der RTC6-Laser-Steuerung (**set_dsp_mode**`( 3 )`) können die LaserOn-Delays bei einer Abfolge von kurzen Vektoren auch über einen oder sogar mehrere der nachfolgenden Vektoren hinausreichen. Die RTC5-kompatiblen künstlichen Scanner-Delays im DSP-Mode 2 entfallen. Die Ausgabe erfolgt dann synchron mit den dem jeweiligen Vektor zugeordneten LaserOn-Delay.

Mit der RTC5-kompatiblen Laser-Steuerung (**set_dsp_mode**`( 2 )`) sind achsensynchrones Ausführen und lasersynchrones Ausführen identisch.

(1) Siehe **set_scanahead_line_params**, **set_scanahead_line_params_list**.

(2) Bei Größenänderungen von mehr als 5% sind Qualitätseinbußen wahrscheinlich.



Hinweise zu gesendeten Datensignalen

Beachten Sie beim Anpassen des Anwenderprogramms auch, dass sich einige von SCANahead-Systemen gesendeten Datensignalen inhaltlich von Schleppverzugs-Systemen (z.B. intelli*SCAN* III), unterscheiden.

- **TrAck-Signal**
  (statt **PosAck-Signal**)
- **Trajektorienfehler**
  (statt Positionsfehler)

9 Peripherie-Schnittstellen programmieren

Scan-Systeme kommen häufig in Maschinen zum Einsatz, in denen Bearbeitungsprozesse des Lasers und des Scan-Systems mit weiteren Prozessschritten (wie Werkstückzuführung, Roboterbewegungen, Prozessüberwachung, etc.) synchronisiert werden müssen.

Zu diesem Zweck ist die RTC6 PCIe-Karte mit einer Reihe von Peripherie-Schnittstellen ausgestattet, siehe Kapitel 4.6 "Schnittstellen zum Laser und zur Peripherie", Seite 75.

Mit Hilfe der Befehle zur Programmierung dieser Schnittstellen können damit zusätzlich und/oder synchron zur Ansteuerung von Lasern und Scan-Systemen

- Signale zur Peripherie-Ansteuerung ausgegeben werden,
- Peripherie-Signale eingelesen und ausgewertet werden,
- Laser-Scan-Prozess und Peripherie-Ansteuerung mit externen Steuersignalen gesteuert und synchronisiert werden.

9.1 Signal-Ausgabe

Zur Peripherie-Ansteuerung (beispielsweise zur Ansteuerung eines Werkstück-Transportsystems, eines Shutters, etc.) können geeignete Signale über die im Folgenden beschriebenen Schnittstellen ausgegeben werden. Die Ausgabe-Werte können zu einem beliebigen Zeitpunkt mit Kontrollbefehlen oder – während der Ausgabe einer Liste – mit Listenbefehlen geändert werden.

9.1.1 16-Bit-Digital-Ausgang

An der EXTENSION 1-Stiftleiste steht ein gepufferter 16-Bit-Digital-Ausgang (DIGITAL OUT 0…DIGITAL OUT 15) zur Verfügung. Sein Ausgangssignal-Pegel ist über einen Jumper einstellbar, siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81.

Mit **write_io_port_list**, **write_io_port**, **write_io_port_mask_list**, **write_io_port_mask** oder **write_port_list** werden dem Ausgang 16-Bit-Digital-Werte zugewiesen. Vor der ersten Wertezuweisung befindet er sich im hochohmigen Zustand.

Zusätzlich kann mit **set_port_default** (`Port` = 3) festgelegt werden, auf welchen Wert der Ausgang gesetzt wird, sobald das Abarbeiten einer Liste mit **stop_execution** oder durch ein externes Stoppsignal beendet wird.

Der Default-Wert wird auch wirksam:

- Bei Positions-abhängige Laser-Steuerung und Geschwindigkeits-abhängige Laser-Steuerung (siehe **set_port_default**)
- Beim Beenden des Pixelmode

Bei aktivierter automatischer Laser-Steuerung mit `Ctrl=6` aus **set_auto_laser_control** wird der Wert am 16-Bit-Digital-Ausgang automatisch angepasst, siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208. Dies kann mit **set_trigger[*]** (Signal 24) aufgezeichnet werden.

Bei der Ausgangswert-Ausgabe wird an der EXTENSION 1-Stiftleiste ein LATCH-Signal als Triggersignal zur Synchronisation der Datenübergabe mit ausgegeben.

Mit **get_io_status** können die aktuellen Werte des 16-Bit-Digital-Ausgangs ausgelesen werden.

9.1.2 8-Bit-Digital-Ausgang

An der EXTENSION 2-Stiftleiste steht ein (über einen Jumper konfigurierbarer) gepufferter 8-Bit-Digital-Ausgang zur Verfügung, siehe Abschnitt "8-Bit-Digital-Ausgang", Seite 84.

Seine Ausgangswerte können mit **write_8bit_port**, **write_8bit_port_list** oder **write_port_list** gesetzt werden. Vor der ersten Wertezuweisung befindet er sich im hochohmigen Zustand. Zusätzlich kann mit **set_port_default** (`Port` = 2) oder **set_laser_off_default** festgelegt werden, auf welchen Wert der 8-Bit-Digital-Ausgang gesetzt wird, sobald das Abarbeiten einer Liste mit **stop_execution** oder durch ein externes Stoppsignal beendet wird.

Der Default-Wert wird auch wirksam:

- Bei der Positions-abhängige Laser-Steuerung und Geschwindigkeits-abhängige Laser-Steuerung (siehe **set_port_default**)
- Beim Beenden des Pixelmode

Bei aktivierter automatischer Laser-Steuerung mit `Ctrl` = 3 aus **set_auto_laser_control** wird der Wert am 8-Bit-Digital-Ausgang automatisch angepasst, siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208. Dies kann mit **set_trigger[*]** (Signal 24) aufgezeichnet werden.

Bei der Ausgangswert-Ausgabe wird an der EXTENSION 2-Stiftleiste ein LATCH-Signal als Triggersignal zur Synchronisation der Datenübergabe mit ausgegeben, wenn Pin (17) durch entsprechende Jumperung dafür konfiguriert wurde, siehe Abschnitt "8-Bit-Digital-Ausgang", Seite 84.

9.1.3 2-Bit-Digital-Ausgang

An der LASER-Buchse steht ein gepufferter 2-Bit-Digital-Ausgang zur Verfügung, siehe Abschnitt "2-Bit-Digital-Ausgang", Seite 77.

Mit **set_laser_pin_out**, **set_laser_pin_out_list** oder **write_port_list** werden dem Ausgang Werte zugewiesen.
Vor der ersten Wertezuweisung ist der Wert 0 (Pins sind LOW).

Zusätzlich kann mit **set_port_default** (`Port` = 4) festgelegt werden, auf welchen Wert der Ausgang gesetzt wird, sobald das Abarbeiten einer Liste mit **stop_execution** oder durch ein externes Stoppsignal beendet wird.

9.1.4 12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2

Die LASER-Buchse stellt den 12-Bit-Analog-Ausgang 1 ANALOG OUT1 und 12-Bit-Analog-Ausgang 2 ANALOG OUT2 bereit, siehe Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77.

Der 12-Bit-Analog-Ausgang 2 ANALOG OUT2 steht zusätzlich auch an der MARKING ON THE FLY-Stiftleiste zur Verfügung, siehe Abschnitt "Analog-Ausgang", Seite 85.

Die Ausgangswerte von 12-Bit-Analog-Ausgang 1 ANALOG OUT1 und 12-Bit-Analog-Ausgang 2 ANALOG OUT2 können unabhängig voneinander gesetzt werden mit:

- **write_da_x**
- **write_da_x_list**
- **write_port_list**

Vor der ersten Wertezuweisung sind die Werte 1 (entspricht etwa 0 V).

Zusätzlich kann mit **set_port_default** (`Port` = 0 oder 1) oder **set_laser_off_default** festgelegt werden, auf welche Werte 12-Bit-Analog-Ausgang 1 ANALOG OUT1 und 12-Bit-Analog-Ausgang 2 ANALOG OUT2 gesetzt werden, sobald das Abarbeiten einer Liste mit **stop_execution** oder durch ein externes Stoppsignal beendet wird.

Der Default-Wert wird auch bei der Positions-abhängige Laser-Steuerung und Geschwindigkeits-abhängige Laser-Steuerung wirksam (siehe **set_port_default**). Der Wert an 12-Bit-Analog-Ausgang 1 ANALOG OUT1 und 12-Bit-Analog-Ausgang 2 ANALOG OUT2 wird – falls entsprechend eingestellt – auch geändert:

- Durch einen Softstart, siehe Kapitel 7.4.7 "Softstart-Modus (noch nicht implementiert)", Seite 205
- Im Pixelausgabemodus, siehe Kapitel 8.7 "Pixelausgabemodus", Seite 275

Bei aktivierter automatischer Laser-Steuerung mit `Ctrl` = 1 oder `Ctrl` = 2 aus **set_auto_laser_control** wird der Wert an 12-Bit-Analog-Ausgang 1 oder 12-Bit-Analog-Ausgang 2 automatisch angepasst, siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208. Dies kann mit **set_trigger[*]** (Signal 24) aufgezeichnet werden.

9.1.5 Schrittmotoren ansteuern

Ausgangssignale

Die Signale (ENABLE, DIRECTION und CLOCK) zum Ansteuern von 1…2 Schrittmotoren werden an der STEPPER MOTOR-Stiftleiste ausgegeben:

- Das ENABLE-Signal (zum Ein- und Ausschalten des Motorstroms) kann während der Initialisierung mit **stepper_init** oder nachträglich mit **stepper_enable** oder **stepper_enable_list** geeignet verändert werden.
- Als CLOCK-Signal generiert die RTC6 PCIe-Karte (bei Referenzfahrten mit **stepper_init** und Zielpositionsfahrten mit **stepper_abs** etc.) periodisch Pulse. Mit jedem CLOCK-Puls führt der Schrittmotor einen einzelnen Schritt aus. Die Pulsperiode für die CLOCK-Signale (und damit die Geschwindigkeit der Schrittmotorbewegung) kann während der Initialisierung mit **stepper_init** oder nachträglich mit **stepper_control** oder **stepper_control_list** eingestellt werden (die Periode wird in Einheiten von 10 μs-Takten angegeben).
- Das DIRECTION-Signal (und damit die Richtung der Schrittmotorbewegung) kann bei Referenzfahrten mit **stepper_init** explizit festgelegt werden. Bei Zielpositionsfahrten mit **stepper_abs** etc. wird das DIRECTION-Signal dagegen intern gesteuert: das Signal wird (auf HIGH) gesetzt, wenn der nächste CLOCK-Puls (gemäß vorgegebenem Zielpositionswert) zu einer Vergrößerung der internen Positionsvariablen führen würde. Das DIRECTION-Signal bleibt auch für die Takte zwischen zwei CLOCK-Pulsen gesetzt, ebenso, wenn der Schrittmotor seine Zielposition erreicht hat. Erst bei einem echten Richtungswechsel wird das DIRECTION-Signal anstelle eines CLOCK-Pulses entsprechend anders gesetzt. Der nächste CLOCK-Puls wird dabei erst um eine volle CLOCK-Pulsperiode später ausgegeben (CLOCK-Puls-Perioden werden also nie undefiniert abgeschnitten).

Hinweise

- Zur Signal-Spezifikation siehe Kapitel 4.6.7 "STEPPER MOTOR-Stiftleiste", Seite 90.
- Zum Auslesen der Signale, siehe Abschnitt "Signale und Statuswerte abfragen", Seite 311.
- Die Schrittmotorsignale werden unabhängig von einer etwaigen Listenausführung ausgegeben. Ein **set_end_of_list**, **pause_list**, **set_wait**, **stop_execution** und externe Stopps beenden oder unterbrechen eine Bewegungsfahrt nicht.
- Sowohl bei Richtungswechsel als auch bei einem Periodenwechsel werden bereits angefangene Perioden noch abgewartet, bevor die neuen Werte aktiv werden. Dadurch werden niemals kürzere als die aktuell eingestellten Pulsabstände erzeugt. Bei einem Richtungswechsel wird zusätzlich eine leere Periode (ohne CLOCK-Puls) eingefügt.

Referenzfahrten und Positions-Initialisierung

Mit **stepper_init** kann eine Referenzfahrt zum jeweiligen Endschalter ausgelöst werden. Die gewünschte Fahrtrichtung kann dabei über den Parameter `Dir` vorgegeben werden. Um sicherzustellen, dass die erreichte Endposition auch bei mechanischem Spiel oder erhöhten Signallaufzeiten nicht jenseits des Endschalters liegt, kann ein Toleranzwert `Tol` angegeben werden, um den der Schrittmotor nach Erreichen des Endschalters in die entgegengesetzte Richtung zurückfährt.

SCANLAB empfiehlt, zuerst eine (schnelle) Referenzfahrt mit kleiner CLOCK-Pulsperiode `Period` und anschließend eine weitere (kürzere, aber langsamere) Referenzfahrt mit größerer CLOCK-Pulsperiode auszuführen.

Ist die Referenzfahrt erfolgreich abgeschlossen, wird als Referenz für nachfolgende Zielpositionsfahrten die Positionsvariable (für die aktuelle Position) auf den über den Parameter `Pos` festgelegten Wert gesetzt. Die angefahrene Referenzposition ist gegenüber der Endschalterposition um ±`Tol` (richtungsabhängig) verschoben.

Während der Referenzfahrt ist der Status “Init” (Bit #5 = 1), siehe Abschnitt “Signale und Statuswerte abfragen”, Seite 311. Die Endschalterposition wird 4 mal überfahren und daraus eine mittlere Endschalterposition berechnet. Danach fährt der Steppermotor um `Tol` Schritte in der zu **stepper_init** `Dir` entgegengesetzten Richtung vom Endschalter weg. `Tol` muss groß genug sein, damit eine mögliche Hysterese überwunden wird. Während dieser Zielpositionsfahrt, siehe Abschnitt “Zielpositionsfahrten”, Seite 310, ist der Status nicht mehr “Init”, sondern “Busy” (Bit #4 = 1). Wählt man bei **stepper_init** `Pos` = `Tol`, so befindet sich der Endschalter im Mittel bei Position 0. Die einzig resultierende Ungenauigkeit ist die Schwankung der Endschalterpositionsmessung beim 4-maligen Überfahren der Endschalterposition.

Für `Period` = 0 und/oder `Dir` < 0 wird keine Referenzfahrt ausgeführt. Stattdessen wird die dann aktuelle Schrittmotor-Position als neue Referenzposition festgelegt (mit dem über `Pos` definierten neuen Wert für die Positionsvariable).

Da **stepper_init** auf jeden Fall eine zuvor gestartete Schrittmotor-Bewegung stoppt, kann **stepper_init** bei Bedarf auch für einen Not-Halt für die Schrittmotoren genutzt werden.

Es können auch für beide Schrittmotoren parallel Referenzfahrten ausgeführt, jedoch nicht mit einem Befehl gleichzeitig gestartet werden.

Zielpositionsfahrten

Zielpositionsfahrten werden mit **stepper_abs**, **stepper_rel**, **stepper_abs_no** oder **stepper_rel_no** oder den entsprechenden Listenbefehlen ausgelöst.

Bei **_abs**-Befehlen werden absolute, bei **_rel**-Befehlen relative Zielpositionswerte (jeweils in CLOCK-Puls-Einheiten) angegeben. **_no**-Befehle führen Zielpositionsfahrten nur für *einen* Schrittmotor-Ausgang aus, die anderen Zielpositions-Befehle für beide Schrittmotor-Ausgänge gleichzeitig.

Mit **stepper_wait** kann die weitere Ausführung einer Liste so lange unterbrochen werden, bis eine gestartete Schrittmotor-Bewegung beendet ist.

Die Listenbefehle für Zielpositionsfahrten sind kurze Listenbefehle, so dass eine Schrittmotorbewegung auch synchron zu einer Galvanometerscanner-Bewegung ausgeführt werden kann.

Wird während einer Zielpositionsfahrt der Endschalter aktiv, bricht die Bewegung sofort ab und kann mit einer normalen Zielpositionsfahrt nicht wieder gestartet werden. Es muss entweder eine Referenzfahrt angefordert oder der Endschalter mechanisch oder über die Software, siehe unten, deaktiviert werden. Bei einer einmal abgebrochenen Schrittmotorbewegung (z.B. auch durch `Period` = 0) wird die bisherige Zielposition durch den aktuellen Positionswert überschrieben. Die Schrittmotorbewegung zur ursprünglichen Zielposition kann daher nicht wieder aufgenommen werden, indem die Unterbrechungsursache (Endschalter oder CLOCK-Pulsperiode = 0) beseitigt wird. Sie muss neu ausgelöst werden.

Um dieses Verhalten zu umgehen, kann die Berücksichtigung des Endschalters mit **stepper_disable_switch** deaktiviert werden. Dann kann auch bei einem vorliegenden Endschalter-Stopp die “Freigabe” auch ohne erneute Initialisierungsfahrt erfolgen. Die Deaktivierung ist insbesondere dann sinnvoll, wenn durch den Schrittmotor eine Rotationsachse gesteuert wird, die sich kontinuierlich dreht. Während einer Initialisierungsfahrt bleibt die etwaige Deaktivierung eines Endschalters unberücksichtigt.

Signale und Statuswerte abfragen

Der aktuelle Status der Schrittmotor-Signale (ENABLE, DIRECTION, CLOCK und SWITCH), die aktuell eingestellte CLOCK-Pulsperiode und die aktuellen Werte der internen Positionsvariablen der beiden Schrittmotoren können mit **get_stepper_status** abgefragt werden.

Zusätzlich liefert **get_stepper_status** auch den Busy- und Init-Status der beiden Schrittmotoren. Während einer Referenzfahrt ist der Init-Status gesetzt, bei Zielpositionsfahrten der Busy-Status.
Solange der Init-Status gesetzt ist, sind keine Zielpositions-Befehle (**stepper_abs**, **stepper_rel**, etc.) zulässig, Kontrollbefehle (außer **stepper_init**) werden mit **get_error**-Returncode `RTC6_BUSY` zurückgewiesen und Listenbefehle warten, bis der Init-Status zurückgesetzt wird. Gegebenenfalls muss die Liste selbst oder der Bewegungsvorgang abgebrochen werden.

Endlosfahrten anhalten

Abhängig von den gewählten Parametern können mit **stepper_init**, **stepper_abs** etc. auch sehr lange Schrittmotor-Bewegungen oder gar Endlosfahrten initiiert werden, z.B. wenn in der angegebenen Richtung kein Endschalter vorhanden oder der Zielpositionswert sehr groß ist bei gleichzeitig langer Pulsperiode.

- Wird eine Endlosfahrt mit einem Kontrollbefehl (z.B. **stepper_abs**) gestartet, dann wird dieser Kontrollbefehl spätestens dann beendet, wenn die im Parameter `WaitTime` angegebene positive Zeit (in Sekunden) abgelaufen ist. Die Endlosfahrt des Schrittmotors selbst läuft aber weiter und muss separat abgebrochen werden, indem die CLOCK-Pulsperiode (z.B. mit dem Kontrollbefehl **stepper_control**) auf 0 gesetzt (Not-Halt) oder eine geeignete neue Zielposition definiert wird.
- Wird eine Endlosfahrt mit einem Listenbefehl (z.B. **stepper_abs_list**) gestartet und mit **stepper_wait** auf deren Ende gewartet, stagniert die weitere Listenausführung, solange die Endlosfahrt nicht über einen Kontrollbefehl wie **stepper_control** mit `Period` = 0 oder einer sinnvollen neuen Zielposition beendet wird. Die Liste kann aber auch mit **stop_execution** oder einem externen Stopp beendet werden. Die Endlosfahrt des Schrittmotors muss auch hier mit **stepper_control**(`Period` = 0) separat gestoppt werden.

Parameter WaitTime

Mit dem Parameter `WaitTime` der Kontrollbefehle kann eingestellt werden, dass sie nach Ablauf der angegeben Zeit (in Sekunden) zum aufrufenden Programm zurückkehren, unabhängig davon, ob die Steppermotorbewegung fertig ist oder nicht.

Mit `WaitTime` = 0 kehrt der Befehl sofort zurück. In diesem Fall muss der Benutzer sicherstellen, dass keine unzulässigen Befehle aufgerufen werden. Insbesondere sollte die Initialisierung nicht abgebrochen werden, bevor sie fertig ist. Anderenfalls führt das zu fehlerhaften Referenzpositionen.

9.1.6 RS-232-Schnittstelle

An der RS232-Stiftleiste wird eine RS-232-Schnittstelle bereitgestellt, siehe Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86.

Zum Konfigurieren der RS-232-Schnittstelle, siehe Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86.

Mit **rs232_write_data** können einzelne Daten-Bytes, mit **rs232_write_text** oder **rs232_write_text_list** Texte an die RS-232-Schnittstelle gesendet werden.

9.1.7 McBSP-Schnittstelle

An der McBSP-Schnittstelle, siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87, wird am DX0 Pin (07) ständig ein 32-Bit-Datenwort im 10 μs-Takt ausgegeben.

Mit **set_mcbsp_out** und **set_mcbsp_out_ptr** kann eingestellt werden, welche Signaltypen (analog zu **set_trigger**) dort ausgegeben werden:

- Mit **set_mcbsp_out** können zwei Signaltypen zur gleichzeitigen Ausgabe an der McBSP-Schnittstelle ausgewählt werden. Ein 16-Bit-Anteil des ersten Signaltyps wird mit einem 16-Bit-Anteil des zweiten Signaltyps zu einem 32-Bit-Datenwort zusammengefasst und dieses alle 10 μs ausgegeben. Für eine genaue Beschreibung siehe **set_mcbsp_out**.
- Mit **set_mcbsp_out_ptr** kann eine Liste von bis zu 8 Signaltypen definiert werden. Die Signalausgabe erfolgt sequenziell in der vorgegebenen Reihenfolge. In jedem 10 μs-Takt werden vom jeweiligen Datensignal die unteren 24 Bit und die zugehörige Signaltypnummer (8 Bit) zu einem 32-Bit-Datenwort zusammengefasst und an der McBSP-Schnittstelle ausgegeben. Für eine genaue Beschreibung siehe **set_mcbsp_out_ptr**.

Hinweise

- Zu Signalen und Betriebsbedingungen siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87.
- In der Voreinstellung werden an der McBSP-Schnittstelle jeweils Bit #4…Bit #19 der kartesischen Ansteuerwerte für die x-Achse und y-Achse (SampleX und SampleY) im gemeinsamen 32-Bit-Datenwort ausgegeben. Das entspricht einer Auswahl gemäß `set_mcbsp_out(7, 8)`.
- Die mit **set_mcbsp_out** ausgewählten Signale werden so lange (bei **set_mcbsp_out_ptr** sequenziell) ausgegeben, bis einer der beiden Befehle erneut aufgerufen wird.
- Der OIE-Ausgabemodus, Seite 773 kann mit **set_mcbsp_out_oie_ctrl** und **set_mcbsp_out_oie_list** an der McBSP-Schnittstelle eingestellt werden.

9.2 Signal-Eingabe

Peripherie-Signale (z.B. Signale eines Transportsystems, eines Werkstück-Erkennungssystems, einer Prozessüberwachungskamera, etc.) können zu einem beliebigen Zeitpunkt mit Kontrollbefehlen oder – während der Ausgabe einer Liste – mit Listenbefehlen über die im Folgenden beschriebenen Schnittstellen eingelesen werden.

9.2.1 16-Bit-Digital-Eingang

Zum Einlesen von 16-Bit-Digital-Werten steht an der EXTENSION 1-Stiftleiste ein geschützter 16-Bit-Digital-Eingang (DIGITAL IN 0…DIGITAL IN 15) zur Verfügung, siehe Kapitel "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81.

Zur Synchronisation der Datenübergabe wird an der EXTENSION 1-Stiftleiste ein SYNC-Signal ausgegeben.

Mit **read_io_port** oder mit **read_io_port_buffer** und **read_io_port_list** werden die aktuellen Werte des Digital-Eingangs ausgelesen.

Für eine bedingte Befehlsausführung stehen weitere Befehle zur Verfügung, siehe Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.

9.2.2 2-Bit-Digital-Eingang

Zum Einlesen von 2-Bit-Digital-Werten steht an der LASER-Buchse ein 2-Bit-Digital-Eingang zur Verfügung, siehe Abschnitt "2-Bit-Digital-Eingang", Seite 77.

Der Eingang kann mit **get_laser_pin_in** ausgelesen werden.

Für eine bedingte Befehlsausführung stehen weitere Befehle zur Verfügung, siehe Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.

9.2.3 RS-232-Schnittstelle

An der RS232-Stiftleiste können Signale über die RS-232-Schnittstelle eingelesen werden, siehe Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86.

Zum Konfigurieren der RS-232-Schnittstelle, siehe Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86.

Mit **rs232_read_data** können Daten eingelesen werden.

9.2.4 McBSP-Schnittstelle

An der McBSP-Schnittstelle, siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87, kann mit **read_mcbsp** das zuletzt vollständig an die angegebene Speicherstelle übertragene 32 Bit-Datenwort ausgelesen werden.

Die Interpretation als ein 32 Bit-Datenwort oder zwei 16 Bit-Datenworte liegt in der Verantwortung des Benutzers.

Die an der McBSP-Schnittstelle empfangenen Signale (Positionswerte) können aber auch direkt in die "Processing on the fly"-Korrektur einer Werkstück- oder Scan-System-Bewegung eingebunden (siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Kapitel 9.3.4 "Synchronisation und Online-Positionierung mit McBSP-Signalen", Seite 325) oder für eine Online-Positionierung verwendet werden, siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240 und Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243.

Hinweise

- Zu Signalen und Betriebsbedingungen siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87.

9.3 Steuerung mit externen Signalen

Die oben beschriebene Eingabe und Ausgabe von Peripherie-Signalen kann wie folgt mit der Scan-System- und Laser-Steuerung synchronisiert werden:

- Die zugehörigen Listenbefehle können an geeigneten Stellen in Befehlslisten eingebaut werden.
- Die Ausführung zugehöriger Kontrollbefehle kann vom aktuellen Status der Listenausführung abhängig gemacht werden. Dazu kann mit **read_status** der Listenstatus, siehe Kapitel 6.4.2 "Listenstatus", Seite 110, und mit **get_status** der Listenausführungsstatus, siehe Kapitel 6.4.3 "Listenausführungsstatus", Seite 111, abgefragt werden.
- Zusätzlich wird der BUSY-Listenausführungsstatus-Listenausführungsstatus über das BUSY OUT-Signal bereitgestellt:
  - an der LASER-Buchse, siehe Abschnitt "BUSY-Listenausführungsstatus", Seite 76
  - an der EXTENSION 1-Stiftleiste, siehe Abschnitt "BUSY-Listenausführungsstatus", Seite 82
  - an der MARKING ON THE FLY-Stiftleiste, siehe Abschnitt "BUSY-Listenausführungsstatus", Seite 85

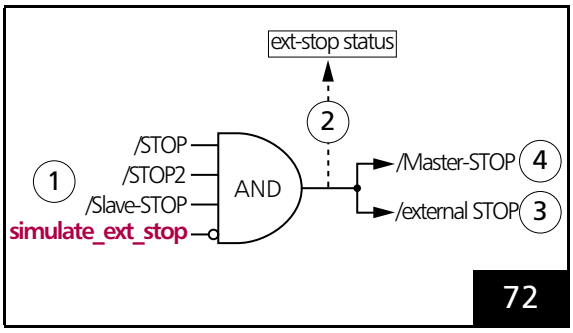
Darüber hinaus stellt die RTC6 PCIe-Karte Schnittstellen bereit, mit denen die Listenausführung direkt mit externen Signalen gesteuert und synchronisiert werden kann.

9.3.1 Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden

Externer Stopp

Ein Externer Stopp, siehe (1) und (3) in Abbildung 72, kann ausgelöst werden durch:

- ein Signal am Eingang /STOP
- ein Signal am Eingang /STOP2
- ein Signal an /Slave-STOP
- einen **simulate_ext_stop**-Aufruf



Externer Stopp. Beschreibung siehe Text.

Dadurch werden:

- die aktuell ausgeführte Liste sofort abgebrochen
- die Signale für den "Laser active"-Betrieb abgeschaltet (aber nicht deaktiviert)

Wie auch nach Aufruf von **stop_execution** (interner Stopp) werden dann die folgenden Ausgänge auf die zuvor – mit **set_port_default** – für den Stopp-Fall festgelegten Werte gesetzt, wenn diese als ≠ "–1" definiert wurden:

- 16-Bit-Digital-Ausgang (EXTENSION 1-Stiftleiste)
- 8-Bit-Digital-Ausgang (EXTENSION 2-Stiftleiste)
- 2-Bit-Digital-Ausgang (LASER-Buchse)
- 12-Bit-Analog-Ausgang 1 ANALOG OUT1 (LASER-Buchse)
- 12-Bit-Analog-Ausgang 2 ANALOG OUT2 (LASER-Buchse)

Die Eingänge für externe Stoppsignale (1) sind immer entriegelt, so dass ein Externer Stopp zu *jedem* Zeitpunkt erfolgen kann (Nothalt).

Der /STOP-Eingang ist an der LASER-Buchse, siehe Abschnitt "Externe Steuersignale", Seite 76, der /STOP2-Eingang an der MARKING ON THE FLY-Stiftleiste, siehe Abschnitt "Externe Steuersignale", Seite 85, verfügbar. Beide Eingänge sind intern über Pull-up-Widerstände (4,7 kΩ) mit einer Versorgungsspannung von +3,3 V verbunden. Sie sind TTL-LOW-aktiv und Pegel-sensitiv. Ein Listenabbruch wird ausgelöst, sobald mindestens einer der beiden Eingänge für mindestens 10 µs auf LOW (oder 0 V oder Massepotential) gesetzt wird.

Wenn eine RTC6 PCIe-Karte über den Master- oder Slave-Anschluss, siehe Abbildung 7, mit weiteren Karten verbunden ist, dann werden externe *Stopps* von der Karte, die das Signal empfängt, an *alle* mit ihr verbundenen Karten durchgereicht, wenn die jeweilige Master/Slave-Schnittstelle entsprechend (mit **master_slave_config**) konfiguriert wurde. Siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.

Durch **stop_execution** ausgelöste Stopps werden *nicht* durchgereicht. Dagegen werden durchgereicht durch **simulate_ext_stop** ausgelöste externe Stopps.

Mit **get_startstop_info** kann der momentane Stopp-Status (d. h. ob einer der Eingänge momentan auf LOW gesetzt ist) ausgelesen werden (2) und ob seit dem letzten Auslesen ein neuer Externer Stopp erfolgt ist.

Externer Start

Über ein Signal an den Eingängen /START, /START2 oder/Slave-START oder über **simulate_ext_start** oder **simulate_ext_start_ctrl** kann ein Externer Start ausgelöst werden (siehe (1), (5) und (7) in Abbildung 73).

Dadurch wird die Ausführung in der Standardeinstellung am Anfang von "Liste 1" gestartet. Mit **set_extstartpos** oder **set_extstartpos_list** kann aber auch zuvor eine andere Startadresse ausgewählt werden. Eine Liste wird nur dann gestartet, wenn weder der BUSY-Listenausführungsstatus (wie bei Listenausgabe) noch der INTERNAL-BUSY-Listenausführungsstatus (wie z.B. bei **goto_xy**) oder der PAUSED-Listenausführungsstatus (nach **pause_list**, **stop_list** oder **set_wait**) gesetzt ist.

Bevor die Eingänge /START, /START2 oder /Slave-START (1) verwendet werden können, müssen sie mit **set_control_mode** aktiviert werden (3).

Der Kontrollbefehl **simulate_ext_start_ctrl** kann mit **set_control_mode**(`Bit #4` = 1) deaktiviert werden. Der Listenbefehl **simulate_ext_start** bleibt weiterhin aktiv.

Der /START-Eingang ist an der LASER-Buchse, siehe Abschnitt "Externe Steuersignale", Seite 76, der /START2-Eingang an der MARKING ON THE FLY-Stiftleiste, siehe Abschnitt "Externe Steuersignale", Seite 85, verfügbar. Beide Eingänge sind intern über Pull-up-Widerstände (4,7 kΩ) mit einer Versorgungsspannung von +3,3 V verbunden. Sie sind TTL-LOW-aktiv und Flanken-sensitiv (auf den Übergang von HIGH nach LOW). Ein Start wird – nach Aktivierung mit **set_control_mode** – ausgelöst, sobald eines der drei Eingangssignale von HIGH auf LOW (oder 0 V oder Massepotential) wechselt.

Wenn eine RTC6 PCIe-Karte über den Master- oder Slave-Anschluss, siehe Abbildung 7, mit weiteren Karten verbunden ist, dann werden externe *Starts* von der Karte, die das Signal empfängt, an *alle* mit ihr verbundenen Karten durchgereicht, wenn die jeweilige Master/Slave-Schnittstelle entsprechend (mit **master_slave_config**) konfiguriert wurde. Siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.

Nicht durchgereicht werden durch **execute_list** oder **execute_at_pointer** ausgelöste interne Starts.

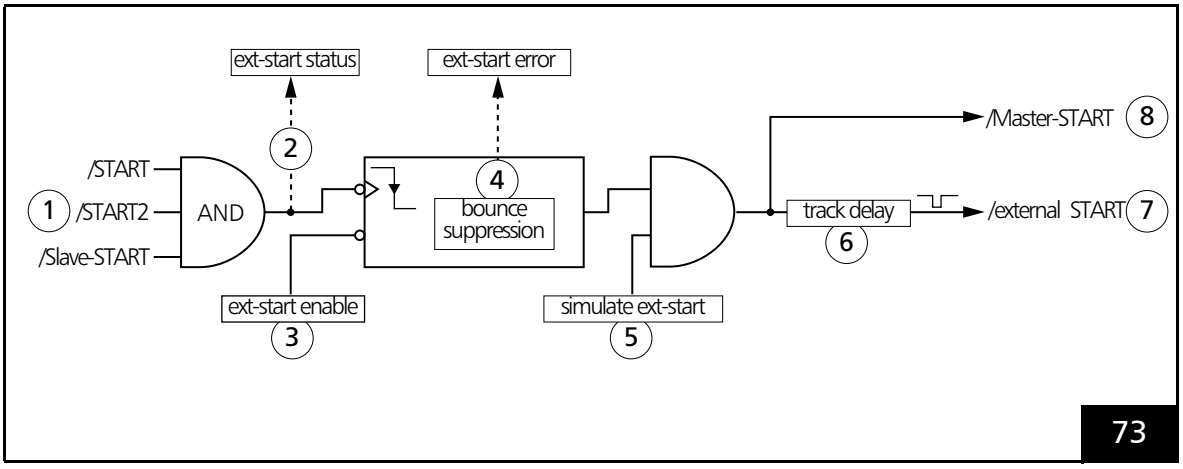
Durchgereicht werden durch **simulate_ext_start** oder **simulate_ext_start_ctrl** ausgelöste externe Starts, siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.

Mit **get_startstop_info** kann der momentane Start-Status (d. h. ob einer der Eingänge auf LOW gesetzt ist) ausgelesen werden (2) und ob eine Liste seit dem letzten Auslesen erfolgreich gestartet wurde.

Mit **bounce_supp** kann eine Entprellung für die über die Eingänge /START, /START2 oder /Slave-START eingespeisten Startsignale eingestellt werden (4). Startsignale, die innerhalb der eingestellten Entprellungszeit nach einem erfolgreichen Startsignal erfolgen, werden dadurch unterdrückt. Ob ein Startsignal unterdrückt (4) wurde, kann mit **get_marking_info** ausgelesen werden.

simulate_ext_start_ctrl kann dazu verwendet werden, einen synchronen Start Master/Slave-synchronisierter Karten mit Kontrollbefehl zu starten.

Der Listenbefehl **simulate_ext_start** kann dazu verwendet werden, nach einem einmalig erfolgten externen Start weitere Starts in definierten Abständen auszulösen (s.u.).



Externer Start. Beschreibung siehe Text.

Externer Start mit Streckenverzögerung

Bei vielen Anwendungen muss der Start relativ zum auslösenden Start-Signal um einen wohldefinierten Abstand verzögert werden, z.B. weil das Werkstück zwischen einer Lichtschranke und dem Scan-System erst noch eine bestimmte Strecke transportiert werden muss.

Zu diesem Zweck kann mit **set_ext_start_delay**, **set_ext_start_delay_list** oder **simulate_ext_start** eine Streckenverzögerung (track delay) konfiguriert werden (siehe (6) in Abbildung 73), so dass die Ausführung des Starts relativ zum auslösenden Eingangssignal oder Befehl entsprechend verzögert wird. Die Streckenverzögerung wird dabei in Zähleinheiten eines internen Zählers (Encoderzählers) angegeben, der wiederum durch ein externes oder simuliertes Encodersignal getriggert wird, siehe Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323.

Externe Starts, die mit einem externen Startsignal oder mit **simulate_ext_start** oder **simulate_ext_start_ctrl** angetriggert, aber entsprechend der eingestellten Streckenverzögerung nicht sofort ausgeführt werden, werden in einer Warteschleife von bis zu 8 Starts gehalten (der jeweilige Start-Trigger wird automatisch generiert, wenn das Delay abgelaufen ist). Dies kann z.B. bei der Bearbeitung von mehreren Werkstücken, die dem Scan-System auch in unregelmäßigen Abständen zugeführt werden, nützlich sein: Hier dürfen sich innerhalb der Streckenverzögerung (Abstand zwischen Lichtschranke und Scan-System) bis zu 8 Werkstücke gleichzeitig befinden. Werden mehr externe Starts angetriggert als gleichzeitig in der Warteschleife gehalten werden können, so wird ein Fehler-Bit gesetzt, das mit **get_startstop_info** (`Bit #11`) ausgelesen werden kann. Wird eine Streckenverzögerung gesetzt, so wird die bisherige Warteschlange vernichtet (siehe **set_ext_start_delay** und **simulate_ext_start**).

Mit **set_control_mode** (`Bit #1` = 1) kann eingestellt werden, dass die Warteschlange mit den externen Start-Eingaben durch einen externen Stopp explizit gelöscht wird. Mit **set_control_mode** (`Bit #1` = 0) bleibt die Warteschlange nach einem externen Stopp bestehen.

Hinweise

- Die Eingänge /START, /START2 und /Slave-START sind (auf den *Übergang* von HIGH nach LOW) *Flanken*-sensitiv.
- Die Eingänge /STOP, /STOP2 und /Slave-STOP sind *Pegel*-sensitiv.
- Ein **stop_execution**-Aufruf deaktiviert die Eingänge /START, /START2 und /Slave-START. Auch durch ein externes Stoppsignal werden diese Eingänge zumindest vorübergehend deaktiviert. Sie bleiben auch deaktiviert, solange einer der Eingänge /STOP, /STOP2 oder /Slave-STOP auf LOW liegt. Ob die Eingänge /START, /START2 und /Slave-START auch dann deaktiviert bleiben wenn das Stoppsignal nicht mehr anliegt, kann mit **set_control_mode** eingestellt werden.
- Mit **set_control_mode** können zusätzlich die Eingänge /START, /START2 oder /Slave-START aktiviert oder deaktiviert und die Streckenverzögerung deaktiviert werden.
- Externe Starts sind auch nach **pause_list**, **stop_list** oder **set_wait** unterdrückt (PAUSED-Listenausführungsstatus gesetzt). **restart_list**, **stop_execution**, **release_wait** oder ein Externer Stopp heben diese Unterdrückung wieder auf.
- Falls die Listeneingabe noch nicht abgeschlossen ist, sollte vor einem externen Start z.B. mit `set_input_pointer( get_input_pointer() )` ein Puffer-Flush initiiert werden, damit noch gepufferte Listenbefehle vollständig in den RTC6-Listenspeicher transferiert werden, siehe Kapitel 6.4.1 "Listen laden", Seite 108.
- Wird eine Master-Karte intern (z.B. mit **execute_list_pos**) gestartet und eine Slave-Karte über **simulate_ext_start** nachgestartet, so laufen Master- und Slave-Karte nicht synchron, falls zuvor über **home_position** oder **home_position_xyz** ein Home-Jump aktiviert wurde: Auf der Master-Karte wird der Home-Return ausgeführt, *bevor* **simulate_ext_start** die Slave-Karte startet, auf der Slave-Karte erst *danach*. Während auf der Slave-Karte der Home-Return abläuft, läuft die Master-Karte bereits weiter. Diese Asynchronität tritt nicht auf, wenn alle Karten mit externem Startsignal (oder mit **simulate_ext_start** oder **simulate_ext_start_ctrl**) gestartet werden oder wenn kein Home-Jump aktiviert ist.

Regelmäßige (periodische) externe Starts

Mit **set_control_mode** oder **set_control_mode_list** (Bit #10) können äquidistante externe Starts realisiert werden, die unabhängig vom Zeitpunkt der Start-Trigger sind, solange diese nur innerhalb der angegebenen Streckenverzögerung erfolgen.

Diese streng periodische Abarbeitung von Listen ist – unabhängig von der tatsächlichen Ausführungsdauer der jeweiligen Liste und des genauen Zeitpunkts der externen Starts – exakt auf den 10 μs-Takt der RTC6 PCIe-Karte synchronisiert.

Durch Setzen von Bit #10 = 1 (`Mode|Bit #10`) kann dazu die Verarbeitung der internen Encoderzähler so eingestellt werden, dass die Streckenverzögerung eines externen Starts nicht erst ab dem Zeitpunkt des auslösenden externen Startsignals oder **simulate_ext_start** gezählt wird (Bit #10 = 0), sondern bereits ab dem *zuletzt* (ebenfalls mit einem externen Startsignal oder mit **simulate_ext_start**) ausgeführten externen *Start*, siehe Abbildung 74. Der Abstand aufeinander folgender Externer Starts (in Encoderpulsen) wird dadurch konstant.

Um diesen Modus zu aktivieren, muss ein Externer Start (einmalig) noch im Modus Bit #10 = 0 (`Mode &~Bit #10`) erfolgt sein. Jeder weitere externe Start muss dann innerhalb der angegebenen Streckenverzögerung angefordert werden.

Beispiel in Pascal für eine typische Befehlsfolge ohne Verwendung eines externen Startsignals:

```
set_control_mode(Mode &~Bit #10);
// einmaliges Zurücksetzen von Bit #10
// (Initialisierung)
set_start_list_pos(ListNo, Pos);
// öffne irgend eine Liste
// nachfolgend: sonstige Befehle
simulate_ext_start(Delay,EncoderNo);
// startet einmalig im Modus Bit #10 = 0,
// sonst im Modus Bit #10 = 1
set_control_mode_list(Mode|Bit #10);
// Setzen von Bit #10
// nachfolgend: weitere Befehle
set_end_of_list;
// schließe die Liste
execute_list_pos(ListNo,Pos);
// starte einmalig die Liste
```

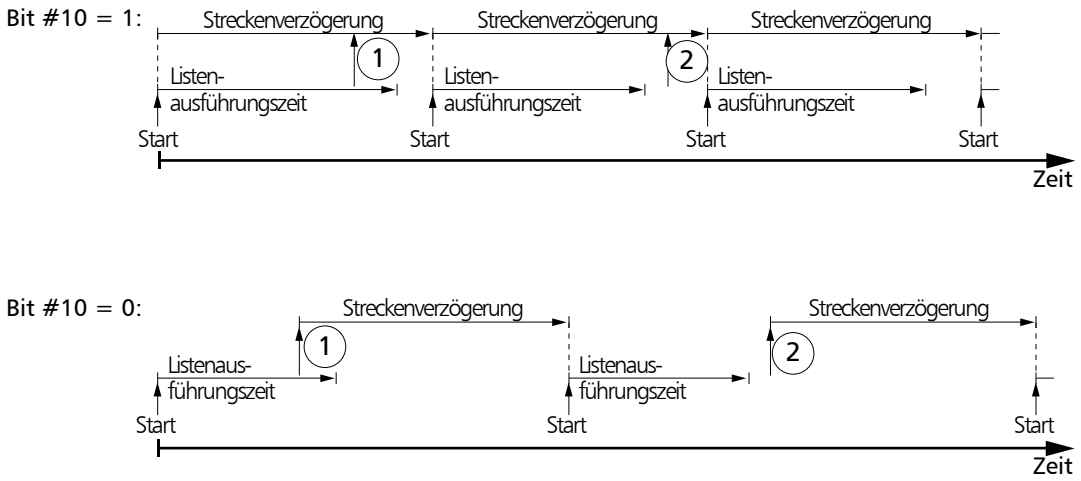
Soll der erste Start statt über einen **execute_list_pos**-Befehl extern erfolgen (z.B. mit /START oder mit **simulate_ext_start_ctrl**), alle weiteren Starts aber mit **simulate_ext_start**, so muss **set_control_mode_list** in obigem Beispiel vor **simulate_ext_start** aufgerufen werden.

Mit **set_control_mode**(`Bit #1` = 1) kann eingestellt werden, dass die Warteschlange der externen Start-Eingabe durch einen externen Stopp explizit gelöscht wird (externe Starts können dadurch auch mit externem Stopp dauerhaft abgebrochen werden).

Für **set_control_mode**(`Bit #1` = 0) (Standardeinstellung) sollte – nachdem eine ansonsten endlos wiederholte Startserie (z.B. mit **set_control_mode**(`Bit #0` = 0) beendet wurde – mit **set_ext_start_delay**(`Delay` = 0) die Streckenverzögerung deaktiviert und die Warteschlange noch nicht ausgeführter Externer Starts vernichtet werden. Anderenfalls ist der nächste "äquidistante" externe Start nicht im richtigen Abstand. **set_control_mode**(`Bit #2` = 1) allein reicht zum Beenden nicht aus, da etwaige noch nicht ausgeführte **simulate_ext_start**-Aufrufe die Streckenverzögerung wieder aktivieren.

Wird ein erneuter Externer Start innerhalb der Streckenverzögerung verpasst, muss die Warteschleife gelöscht werden (anderenfalls benötigt der Encoderzähler einen kompletten 31-Bit-Umlauf, bis wieder ein Start erfolgreich ausgelöst wird). Das Löschen kann durch ein erneutes Setzen einer Streckenverzögerung mit **set_ext_start_delay** erfolgen.

In jedem Fall muss zur Initialisierung Bit #10 zunächst zurückgesetzt und dann (einmalig) ein Externer Start ausgelöst werden (für externe Startsignale muss Bit #0 gesetzt werden), bevor Bit #10 wieder gesetzt werden kann (siehe Beispiel). Anderenfalls ist die erste Streckenverzögerung in der Warteschleife undefiniert.



Legende

1 **simulate_ext_start**.

2 /START oder /START2 oder /Slave-START.

74

Regelmäßige und unregelmäßige externe Starts (Beschreibung siehe Text).

9.3.2 Ausführung Bedingter Befehle

Damit das Ausführen einzelner Listenbefehle von externen Steuersignalen abhängig gemacht werden kann, stehen die sogenannten bedingten Befehle(1) zur Verfügung.

Die bedingten Befehle lesen den aktuellen Wert am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste aus, siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81.
Ihre Ausführung ist *abhängig vom ausgelesenen Wert*:

- Bedingte Sprünge
  list_jump_pos_cond (gleichbedeutend mit **list_jump_cond**) und **list_jump_rel_cond** führen einen Sprung innerhalb eines RTC6-Listenspeicher-Bereichs aus oder nicht aus. Für die Sprungadressen gelten dieselben Bedingungen wie für **list_jump_pos** und **list_jump_rel**
- Sprung mit variabler Sprungweite
  switch_ioport führt einen relativen Listensprung aus
- Bedingte Aufrufe von nichtindizierten Unterprogrammen
  list_call_cond und **list_call_abs_cond** rufen ein nichtindiziertes Unterprogramm am angegebenen Speicherplatz auf oder nicht auf
- Bedingte Aufrufe von indizierten Unterprogrammen
  sub_call_cond und **sub_call_abs_cond** rufen ein indiziertes Unterprogramm mit dem angegebenen Index auf oder nicht auf
- Bedingte Ausgabe von Peripherie-Signalen
  set_io_cond_list und **clear_io_cond_list** verknüpfen die Ausgabe am 16-Bit-Digital-Ausgang der EXTENSION 1-Stiftleiste, siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81, direkt mit den Signalen am Digital-Eingang: es werden einzelne Bits des Digital-Ausgangs gesetzt oder zurückgesetzt
- Bedingte Ausführung beliebiger Listenbefehle:
  if_cond oder **if_not_cond** bleiben wirkungslos, wenn die Bedingung für den ausgelesenen Wert erfüllt oder nicht erfüllt ist. Ansonsten sorgen sie dafür, dass der unmittelbar nachfolgende Listenbefehl übersprungen wird. Auf diese Weise können sämtliche Listenbefehle bedingt ausgeführt werden.
  Beispielsweise ist die Befehlsfolge
  `if_cond(…)`
  `list_call(…)`
  wirkungsgleich mit
  `list_call_cond(…)`.

Die Ausführung beliebiger Listenbefehle kann auch vom aktuellen Wert am 2-Bit-Digital-Eingang der LASER-Buchse abhängig gemacht werden. Dazu stehen **if_pin_cond** und **if_not_pin_cond** zur Verfügung. Sie funktionieren analog zu **if_cond** und **if_not_cond**. Für ein sicheres Funktionieren dieser bedingten Befehle sollten die Signale am 2-Bit-Digital-Eingang jeweils für mindestens 10 μs unverändert anliegen.

Mit dem Kontrollbefehl **set_pause_list_cond** kann eine Bedingung für den 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste definiert werden. Ist diese Bedingung erfüllt, wird eine gerade ausgeführte Liste durch einen automatischen Aufruf von **pause_list** angehalten. Die Liste kann dann nur noch mit **restart_list** weiter ausgeführt werden. Die Bedingung wird einmal pro 10 μs überprüft. Ein bedingtes **pause_list** hat Vorrang vor einem gleichzeitig anliegenden /STOP-Signal.
set_pause_list_not_cond führt dasselbe aus wie **set_pause_list_cond**, wenn die definierte Bedingung nicht erfüllt ist.

(1) Synonym: konditionale Befehle.

Beispiel-Code (Pascal)

(1) Bestätige ein Signal:

```
set_start_list(1);
   …
   // set Bit #0 of the 16-Bit-Digital-Ausgang
   set_io_cond_list(0, 0, 1);
   // loop until the signal is confirmed (that is, Bit #0 of the digital input turns HIGH)
   list_jump_rel_cond(0, 1, 0);
   // clear Bit #0 of the 16-bit output
   clear_io_cond_list(0, 0, 1);
   // loop until the signal is confirmed
   list_jump_rel_cond(1, 0, 0);
   …
   set_end_of_list;
execute_list(1);
```

(2) Wenn die 4 niederwertigen Bits des Digital-Eingangs den Wert (0110) haben, dann setze Bit #1 des 16-Bit-Digital-Ausgangs. Sonst setze Bit #1 zurück:

```
set_start_list(2);
   …
   // RTC4 style: list_jump_cond($0006, $0009, get_input_pointer + 3);
   // this command uses absolute addresses and is not relocatable
   // the following RTC6 command uses relative addresses and is relocatable:
   // skip the next two commands, if the state
   // of the 16-bit input port is (xxxx xxxx xxxx 0110)
   // list_jump_rel_cond($0006, $0009, 3);

   // clear Bit #1 of the 16-bit output port and…
   clear_io_cond_list(0, 0, 2);
   //RTC4 style: set_list_jump(get_input_pointer + 2);
   //this command uses absolute addresses and is not relocatable
   //the following RTC6 command uses relative addresses and is relocatable
   //…skip the next command
   list_jump_rel(2);

   // set Bit #1 of the 16-bit output port
   set_io_cond_list(0, 0, 2);

// (continue)
   …
   set_end_of_list;
execute_list(2);
…
bit1 := (get_io_status AND $0002)        // returns the current state of Bit #1
```

(3) Wähle unter 15 kleinen Unterprogrammen an definierten Speicheradressen aus:

```
…
for i := 1 to 15 do
     // call subroutine at address i*100, if [Bit #3…Bit #0] (binary) = i
     list_call_cond(i, 15-i, i*100);
…
```

(4) Wähle unter 15 indizierten Unterprogrammen aus:

```
…
for i := 1 to 15 do
     // call subroutine with index i, if [Bit #3…Bit #0] (binary) = i
     sub_call_cond(i, 15-i, i);
…
```

9.3.3 Synchronisation durch Encodersignale

Vorgesehene Verwendung

Bei der Bearbeitung von bewegten Werkstücken müssen Laser-Scan-Prozesse auf die momentane Position des Werkstücks abgestimmt werden.

Zur Einbindung der aktuellen Werkstück-Position können von der RTC6 die Signale von bis zu zwei kundeneigenen Inkremental-Encodern eingelesen werden. Inkremental-Encoder registrieren allerdings nicht die aktuelle Position des Werkstücks, sondern die Bewegungen des Transportsystems (Förderband, rotierende Scheibe, etc.)(1): Bei jeder Bewegung des Transportsystems liefern sie – von der Bewegungsrichtung abhängige – Signale an die RTC6, die dort zu einer Zählwert-Erhöhung oder -Verringerung zweier interner Encoderzähler(2) führen. Der Zählerstand der RTC6-Encoderzähler entspricht dann direkt der Position des Werkstücks(3).

Falls das Werkstück immer bei einer konstanten Geschwindigkeit bearbeitet wird und daher ein Encoder nicht zwingend erforderlich ist, können die Encodersignale auch mit **simulate_encoder** simuliert werden, so dass die Encoderzähler mit einer konstanten Zählrate von 1 MHz hochgezählt werden.

Die aktuellen Zählerstände der beiden Encoderzähler können mit dem Kontrollbefehl **get_encoder** ausgelesen werden. Sie können auch mit dem Listenbefehl **store_encoder** zwischengespeichert und mit dem Kontrollbefehl **read_encoder** aus dem Zwischenspeicher ausgelesen werden.

Außerdem werden die aktuellen Zählerstände von der RTC6 automatisch ausgewertet, wenn die Ausführung von Laser-Scan-Prozessen wie folgt gesteuert wird:

- Bei “Processing on the fly”-Anwendungen (siehe Kapitel 8.6 ““Processing on the fly””, Seite 254) werden die Koordinatenwerte aller Vektor-Befehle und Bogen-Befehle entsprechend der aktuellen Zählerstände der Encoderzähler transformiert.
- Mit dem Listenbefehl **wait_for_encoder_mode** kann die weitere Ausführung einer Liste so lange unterbrochen werden, bis der ausgewählte Encoderzähler einen vorgegebenen Wert über- oder unterschritten hat.
- Bei 2D-Bewegungen wartet **wait_for_encoder_in_range**, bis die Encoderwerte innerhalb eines vorgegebenen Rechtecks liegen.
- Bei externen Starts kann mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** eine Streckenverzögerung zur Verzögerung der Ausführung des Starts relativ zum auslösenden Eingangssignal oder Befehl konfiguriert werden, siehe Abschnitt “Externer Start”, Seite 315.
- Bei der Encodergeschwindigkeits-abhängige Laser-Steuerung kann ein Lasersteuersignal-Parameter entsprechend der aktuell vorliegenden Encodergeschwindigkeit (Zählerpulse des letzten 10 μs-Intervalls) eines Encoderzählers gesteuert werden.

(1) Die *tatsächliche* Position des Werkstücks kann auch über die McBSP-Schnittstelle an die RTC6 PCIe-Karte übergeben werden, siehe Kapitel 9.3.4 “Synchronisation und Online-Positionierung mit McBSP-Signalen”, Seite 325.

(2) Siehe Achtung!, Seite 59.

(3) Die Encoderzähler sind 32-Bit-Zähler mit Vorzeichen. Wird der maximale (minimale) Zählerwert erreicht, wird beim minimalen (maximalen) Zählerwert weitergezählt. Ein Reset der Zähler wird nur durch “Processing on the fly”-Befehle ausgelöst (siehe **set_fly_x**, **set_fly_y**, **set_fly_rot**).
Mit **set_control_mode**(Bit #9 = 1) kann der Encoderreset exakt mit externen Startsignalen synchronisiert werden.

Eingänge für externe Encodersignale

Zur Eingabe von Encodersignalen stehen an der MARKING ON THE FLY-Stiftleiste 2 Encodereingänge zur Verfügung, siehe Abschnitt "Encodereingänge", Seite 85:

- ENCODER X
- ENCODER Y

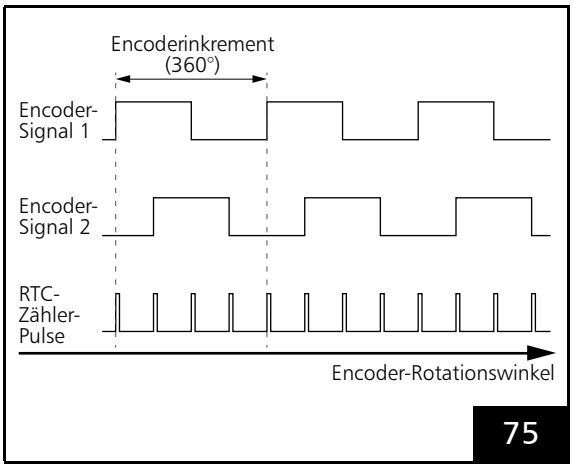
Bei linearen Werkstück-Bewegungen können daran bis zu 2 kundeneigene Inkremental-Encoder angeschlossen werden, die unabhängig voneinander die Werkstück-Bewegung in x-Richtung und y-Richtung definieren.

Bei Rotationsbewegungen wird nur *ein* Inkremental-Encoder benötigt. Für "Processing on the fly"-Anwendungen muss dieser an ENCODER X angeschlossen werden.

ENCODER X und ENCODER Y sind jeweils für ein Paar (1, 2) von standardisierten differenziellen Eingangssignalen (RS-422) vorgesehen.

Technische Spezifikationen siehe Seite 976 (RTC6 PCIe-Karte) und Seite 1068 (RTC6 Ethernet-Karte).

Das Zeitdiagramm eines typischen Encodersignalpaares zeigt Abbildung 75. Das zweite Signal ist üblicherweise um 90° gegenüber dem ersten Signal phasenverschoben. Der entsprechende RTC6-interne Encoderzähler(1) wird durch jede Flanke beider Signale getriggert. *Ein* Encoderinkrement führt also zu *4* Zählerpulsen (Zählschritten). Durch die relative 90°-Phasenverschiebung der beiden Signale kann die RTC6 PCIe-Karte auch die Bewegungsrichtung des Werkstücks ermitteln. Abhängig von der Bewegungsrichtung, wird der jeweilige Zählerwert erhöht oder erniedrigt.



Zeitdiagramm. Typisches Encodersignalpaar und entsprechende RTC6 Zähler-Pulse.

Die Signale am Encodereingang:

- ENCODER X triggern den Encoderzähler "Encoder0"
- ENCODER Y triggern den Encoderzähler "Encoder1"

Die Encodersignale sollten die zulässige Höchstfrequenz von 4 MHz (16 Encodersignalflanken / μs) nicht überschreiten. Siehe `set_encoder_filter_ctrl( Mode = 1 )` (Encodereingangsfilter).

Ein Rauschen in den Encodersignalen kann zu unerwünschten Effekten in Markierergebnissen führen. Mit **set_encoder_filter_ctrl** können diese verringert oder sogar beseitigt werden: Die RTC6 PCIe-Karte glättet dann die Encodergeschwindigkeiten mit einem gleitenden Mittelwertfilter. Dessen Länge kann mit dem Parameter `Length` konfiguriert werden. Ein höherer `Length`-Wert verbessert die Filterwirkung auf Kosten einer gewissen Verzögerung der verwendeten Encodergeschwindigkeiten.

Encodersimulation

Die Encodersimulation kann mit **simulate_encoder** aktiviert (deaktiviert) werden. Die RTC6 PCIe-Karte stellt ein 1 MHz-Taktsignal (Zählerpulse) bereit, das die Signale eines externen Inkremental-Encoders ersetzt.

(1) Siehe Achtung!, Seite 59.

9.3.4 Synchronisation und Online-Positionierung mit McBSP-Signalen

Anstelle von Inkremental-Encodern kann die Bewegung zwischen Werkstück und Scan-System über absolute Positionsangaben über die McBSP-Schnittstelle synchronisiert werden.

Alternativ erlaubt diese Schnittstelle auch die Ausrichtung des Werkstücks relativ zum (feststehenden) Scan-System ("Online-Positionierung").

Der zuletzt vollständig übertragene Eingangswert an der McBSP-Schnittstelle kann mit **read_mcbsp** ausgelesen werden. Er wird von der RTC6 PCIe-Karte automatisch ausgewertet, wenn die Ausführung von Laser-Scan-Prozessen wie folgt gesteuert wird:

- Bei "Processing on the fly"-Anwendungen (siehe Kapitel 8.6 ""Processing on the fly"", Seite 254) werden die Koordinatenwerte aller Vektor-Befehle und Bogen-Befehle entsprechend des aktuellen Eingangswerts transformiert.
- Mit dem Listenbefehl **wait_for_mcbsp** kann die weitere Ausführung einer Liste so lange unterbrochen werden, bis der Eingangswert einen vorgegebenen Wert erreicht, über- oder unterschritten hat.
- Bei der Online-Positionierung (siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240 und Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243) wird das Scan-System entsprechend der aktuellen Eingangswerte ausgerichtet.

Hinweise

- Zur Kompensation linearer Bewegungen können kartesische Koordinatenwerte an der McBSP-Schnittstelle übergeben werden. Zur Kompensation von Rotationsbewegung müssen dagegen Winkelpositionen übergeben werden. Bei der Auswertung werden volle Umdrehungen ausgeblendet, solange die Eingangswerte den zulässigen Wertebereich (20 Vollkreise, siehe **set_mcbsp_rot**) nicht übersteigen.
- Die Signale an der McBSP-Schnittstelle haben keinen Einfluss auf die Streckenverzögerung, die für externe Starts (mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list**) konfiguriert werden kann.

9.4 Periodische I/O-Signale

Mit **periodic_toggle** und **periodic_toggle_list** können periodisch wiederholte Signale an einem auswählbaren IO-Port ausgegeben werden:

- ANALOG OUT1-Ausgang, siehe Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77
- ANALOG OUT2-Ausgang, siehe Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77
- 8-Bit-Digital-Ausgang, siehe Abschnitt "8-Bit-Digital-Ausgang", Seite 84
- 16-Bit-Digital-Ausgang, siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81
- 2-Bit-Digital-Ausgang, siehe Abschnitt "2-Bit-Digital-Ausgang", Seite 77

Dadurch können z.B. externe Peripheriegeräte synchron zur Listenausführung getriggert werden.

Hinweise

- Bei **set_end_of_list**, **stop_execution** oder /STOP laufen die periodischen Signale weiter, selbst wenn sie über den Listenbefehl **periodic_toggle_list** aktiviert worden sind.
- **periodic_toggle** und **periodic_toggle_list** toggeln bei `Count` = 2^{32}–1 endlos.

10 RTC6-Befehle

- Kapitel 10.1 "Übersicht", Seite 327
- Kapitel 10.2 "RTC6-Befehlssatz", Seite 342
- Kapitel 10.4 "Nicht mehr unterstützte RTC4-Befehle", Seite 963
- Kapitel 10.5 "Nicht mehr unterstützte RTC5-Befehle", Seite 966

10.1 Übersicht

- Kontrollbefehle, Seite 328
- Listenbefehle, Seite 334

10.1.1 Kontrollbefehle

- Bedeutung(en):

(n_)	Entsprechender Multi-Board-Befehl, siehe Multi-Board-Befehlsname, Seite 344.

RTC6-Karten-BIOS Flashen

(n_) upgrade_bios 930

Initialisierung der RTC6-DLL

free_rtc6_dll 433
get_rtc_mode 466
init_rtc6_dll 517
set_rtc4_mode 802
set_rtc6_mode 804

Mehrere RTC6 PCIe-Karten in einem PC verwenden

acquire_rtc 346
release_rtc 652
rtc6_count_cards 659
select_rtc 668

RTC6-Listenspeicher-Befehle

(n_) config_list 375
(n_) get_config_list 437
(n_) get_list_space 458
(n_) load_disk 554
(n_) save_disk 661

Initialisierung der Karte und Bildfeld-Korrektur

(n_) get_head_para 448
(n_) get_sync_status 479
(n_) get_table_para 481
(n_) load_correction_file 550
(n_) load_program_file 566
(n_) load_stretch_table (1) 569
(n_) load_z_table (1) 575
(n_) load_z_table_20b (1) 576
(n_) load_z_table_no (1) 577
(n_) load_z_table_no_20b (1) 578
(n_) number_of_correction_tables 612
read_abc_from_file 638
read_abc_from_file_20b 639
(n_) select_cor_table 664
(n_) set_dsp_mode 697
(n_) sync_slaves 902
write_abc_to_file 949
write_abc_to_file_20b 950

Laser-Modus und Laser-Parameter

(n_) config_laser_signals 373
(n_) get_standby 472
(n_) load_auto_laser_control 547
(n_) load_position_control 563
(n_) load_position_control_2d_ctrl 565
(n_) set_auto_laser_control 674
(n_) set_auto_laser_params 679
(n_) set_encoder_speed_ctrl 703
(n_) set_firstpulse_killer 709
(n_) set_laser_control 740
(n_) set_laser_mode 745
(n_) set_laser_pulse_sync 749
(n_) set_laser_pulses_ctrl 751
(n_) set_pulse_picking 798
(n_) set_pulse_picking_length 799
(n_) set_qswitch_delay 800
(n_) set_softstart_level 831
(n_) set_softstart_mode 832
(n_) set_standby 833
(n_) spot_distance_ctrl 877

(1) Nur mit Option "3D".



Festlegen der Galvanometerscannerparameter

(n_) load_varpolydelay ... 573
(n_) set_delay_mode ... 694
(n_) set_jump_speed_ctrl ... 738
(n_) set_mark_speed_ctrl ... 756
(n_) set_sky_writing ... 819
(n_) set_sky_writing_limit ... 820
(n_) set_sky_writing_min_speed_ctrl ... 822
(n_) set_sky_writing_mode ... 824
(n_) set_sky_writing_para ... 826
(n_) set_sky_writing_scale_ctrl ... 829

Koordinatentransformationen

(n_) set_angle ... 672
(n_) set_defocus (1) ... 689
(n_) set_defocus_offset (1) ... 692
(n_) set_matrix ... 757
(n_) set_offset ... 785
(n_) set_offset_xyz (1) ... 786
(n_) set_scale ... 805

Online-Positionierung

(n_) apply_mcbsp ... 357
(n_) set_mcbsp_global_matrix ... 762
(n_) set_mcbsp_global_rot ... 764
(n_) set_mcbsp_global_x ... 765
(n_) set_mcbsp_global_y ... 766
(n_) set_mcbsp_matrix ... 770
(n_) set_mcbsp_rot ... 778
(n_) set_mcbsp_x ... 779
(n_) set_mcbsp_y ... 780

Statusüberwachung und Diagnose

(n_) get_head_status ... 449
(n_) get_overrun ... 464
(n_) get_value ... 492
(n_) get_values ... 494
(n_) get_waveform ... 496
(n_) get_waveform_offset ... 497
(n_) measurement_status ... 604
(n_) stop_trigger ... 891

i*DRIVE*-Befehle

(n_) control_command ... 377
(n_) get_transform ... 486
(n_) get_transform_offset ... 487
(n_) read_user_data ... 650
(n_) send_user_data ... 671
transform ... 926
(n_) upload_transform ... 931

Pixelausgabemodus

(n_) set_default_pixel ... 688

I/O-Befehle

(n_) get_free_variable ... 445
(n_) get_io_status ... 454
(n_) get_mcbsp ... 463
(n_) mcbsp_init ... 602
(n_) mcbsp_init_spi ... 603
(n_) periodic_toggle ... 633
(n_) read_analog_in ... 640
(n_) read_io_port ... 643
(n_) read_io_port_buffer ... 644
(n_) read_mcbsp ... 646
(n_) read_multi_mcbsp ... 647
(n_) rs232_config ... 655
(n_) rs232_read_data ... 656
(n_) rs232_write_data ... 657
(n_) rs232_write_text ... 657
(n_) set_free_variable ... 729
(n_) set_laser_off_default ... 746
(n_) set_mcbsp_freq ... 761
(n_) set_mcbsp_out_ptr ... 775
(n_) set_port_default ... 796
(n_) uart_config ... 929
(n_) write_8bit_port ... 948
(n_) write_da_1 ... 951
(n_) write_da_2 ... 952
(n_) write_da_x ... 953
(n_) write_io_port ... 957
(n_) write_io_port_mask ... 958

(1) Ohne Option "3D" nur eingeschränkte Funktionalität.

Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden

(n_) get_counts ... 437
(n_) get_startstop_info ... 473
(n_) set_control_mode ... 683
(n_) set_extstartpos ... 706
(n_) set_max_counts ... 760
(n_) simulate_ext_start_ctrl ... 875

Listen-Handling und Listen-Status

(n_) auto_change ... 366
(n_) auto_change_pos ... 367
(n_) get_lap_time ... 455
(n_) get_status ... 475
(n_) get_wait_status ... 495
(n_) pause_list ... 632
(n_) quit_loop ... 635
(n_) read_status ... 648
(n_) release_wait ... 653
(n_) restart_list ... 655
(n_) set_pause_list_cond ... 789
(n_) set_pause_list_not_cond ... 790
(n_) start_loop ... 878
(n_) stop_execution ... 890
(n_) stop_list ... 891

Eingabe-Zeiger-Befehle

(n_) get_input_pointer ... 454
(n_) get_list_pointer ... 457
(n_) load_list ... 561
(n_) set_input_pointer ... 731
(n_) set_start_list ... 835
(n_) set_start_list_1 ... 835
(n_) set_start_list_2 ... 835
(n_) set_start_list_pos ... 836

Ausgabe-Zeiger-Befehle

(n_) execute_at_pointer ... 423
(n_) execute_list ... 424
(n_) execute_list_1 ... 424
(n_) execute_list_2 ... 424
(n_) execute_list_pos ... 425
(n_) get_out_pointer ... 464

Unterprogramm-Befehle

(n_) copy_dst_src ... 379
(n_) get_char_pointer ... 436
(n_) get_sub_pointer ... 478
(n_) get_text_table_pointer ... 483
(n_) load_char ... 549
(n_) load_sub ... 571
(n_) load_text_table ... 572
(n_) set_char_pointer ... 681
(n_) set_char_table ... 682
(n_) set_sub_pointer ... 838
(n_) set_text_table_pointer ... 839

Direkte Laser- und Scan-Kopf-Steuerung

(n_) disable_laser ... 385
(n_) enable_laser ... 386
(n_) get_laser_pin_in ... 456
(n_) get_z_distance (1) ... 498
(n_) goto_xy ... 500
(n_) goto_xyz (2) ... 501
(n_) laser_signal_off ... 528
(n_) laser_signal_on ... 529
(n_) set_laser_pin_out ... 747

Versionsbefehle

(n_) get_bios_version ... 435
get_dll_version ... 438
(n_) get_hex_version ... 451
(n_) get_rtc_version ... 467
(n_) get_serial_number ... 471

Error-Befehle

(n_) get_error ... 439
(n_) get_last_error ... 456
(n_) reset_error ... 654
(n_) set_verify ... 856
verify_checksum ... 933

Datum, Uhrzeit, Seriennummern

(n_) get_list_serial ... 458
(n_) get_serial ... 470
(n_) select_serial_set ... 670
(n_) set_serial ... 816
(n_) set_serial_step ... 817
(n_) time_update ... 907

(1) Nur mit Option "3D".

(2) Ohne Option "3D" nur eingeschränkte Funktionalität.

Festlegen der Laserparameter
(n_) set_laser_delays_ctrl 744

Festlegen der Galvanometerscannerparameter
(n_) set_scanner_delays_ctrl 815

"Klassische" "Processing on the fly"-Kontrollbefehle
(n_) clear_fly_overflow_ctrl 371
(n_) get_encoder 438
(n_) get_fly_2d_offset 445
(n_) get_marking_info 459
(n_) init_fly_2d 515
(n_) load_fly_2d_table 557
(n_) read_encoder 641
(n_) set_ext_start_delay 707
(n_) set_fly_tracking_error 721
(n_) set_mcbsp_in (1) 767
(n_) set_multi_mcbsp_in (1) 781
(n_) set_rot_center 801
(n_) simulate_encoder 873
(n_) simulate_ext_stop 876

Schrittmotoren ansteuern
(n_) get_stepper_status 477
(n_) stepper_abs 879
(n_) stepper_abs_no 880
(n_) stepper_control 882
(n_) stepper_disable_switch 883
(n_) stepper_enable 884
(n_) stepper_init 885
(n_) stepper_rel 887
(n_) stepper_rel_no 888

Sprungmodus
(n_) get_jump_table 455
(n_) load_jump_table 558
(n_) load_jump_table_offset 559
(n_) set_jump_mode 733
(n_) set_jump_table 739

Kontrollbefehle für Master/Slave
(n_) get_master_slave 462
(n_) get_sync_status 479
(n_) master_slave_config 600
(n_) master_slave_set_offset_ctrl 601
(n_) sync_slaves 902

Kontrollbefehle nur für SCANahead-Systeme
(n_) activate_scanahead_autodelays 355
(n_) get_scanahead_params 468
(n_) set_scanahead_laser_shifts 806
(n_) set_scanahead_line_params 808
(n_) set_scanahead_params 810
(n_) set_scanahead_speed_control 813

(1) Ohne Option "Processing on the fly" nur eingeschränkte Funktionalität.

Kontrollbefehle für RTC6 Ethernet-Karten

eth_assign_card ... 387
eth_assign_card_ip ... 388
(n_) eth_check_connection ... 391
(n_) eth_config_waveform_streaming_ctrl ... 392
(n_) eth_configure_link_loss ... 393
eth_convert_ip_to_string ... 394
eth_convert_string_to_ip ... 395
eth_count_cards ... 396
eth_found_cards ... 397
eth_full_reset ... 398
eth_get_card_info ... 399
eth_get_card_info_search ... 400
(n_) eth_get_com_timeouts ... 401
(n_) eth_get_com_timeouts_auto ... 402
(n_) eth_get_error ... 403
eth_get_ip ... 405
eth_get_ip_search ... 405
(n_) eth_get_last_error ... 406
(n_) eth_get_port_numbers ... 408
eth_get_serial_search ... 409
(n_) eth_get_standalone_status ... 410
(n_) eth_get_static_ip ... 411
eth_max_card ... 412
eth_remove_card ... 413
eth_search_cards ... 414
eth_search_cards_range ... 415
(n_) eth_set_com_timeouts ... 416
(n_) eth_set_com_timeouts_auto ... 417
(n_) eth_set_high_performance_mode ... 418
(n_) eth_set_port_numbers ... 419
(n_) eth_set_remote_tgm_format ... 420
eth_set_search_cards_timeout ... 421
(n_) eth_set_static_ip ... 422
(n_) time_control_eth ... 904

Standalone-Funktionalität für RTC6 Ethernet-Karten

(n_) eth_boot_dcmd ... 389
(n_) eth_boot_timeout ... 389
(n_) set_eth_boot_control ... 705
(n_) read_image_eth ... 642
(n_) store_program ... 893
(n_) write_image_eth ... 956

Erweiterter Scan-Kopf-Status

(n_) get_extended_head_status_block_ctrl .. 443
(n_) get_extended_head_status_value_ctrl .. 444

Rückkanal-Multiplexing

demux_rcm_ctrl 383
(n_) get_rcm_value_ctrl 465

Kontrollbefehle für Vorverarbeitendes System

(n_) set_controlpreview_compensation_ctrl 687

Kontrollbefehle nur für einen begrenzten Benutzerkreis(1)

(n_) set_fly_tracking_error 721

Sonstige Kontrollbefehle

(n_) auto_cal 363
(n_) bounce_supp 368
(n_) create_dat_file 381
(n_) get_auto_cal 434
(n_) get_card_type 435
(n_) get_galvo_controls 446
(n_) get_hi_data 452
(n_) get_hi_pos 453
(n_) get_temperature 482
(n_) get_time 484
(n_) get_timestamp_long 485
(n_) home_position 502
(n_) home_position_xyz (2) 503
(n_) load_zoom_correction_file 579
(n_) move_to 610
(n_) set_encoder_filter_ctrl 700
(n_) set_hi 730
(n_) set_mcbsp_out_oie_ctrl 773
(n_) set_pause_list_cond 789
(n_) set_short_cmd_mode_ctrl 818
(n_) set_timelag_compensation 841
(n_) set_zoom 872
(n_) store_timestamp_counter 894
(n_) write_hi_pos 955

(1) Diese Benutzer wurden im Umgang mit diesen RTC6-Befehlen separat informiert. Deswegen wird nur die Befehlstabelle aufgenommen und auf weitere Informationen in diesem Handbuch verzichtet.

(2) Ohne Option "3D" nur eingeschränkte Funktionalität.

10.1.2 Listenbefehle

- Bedeutung(en):

(n_)	Entsprechender Multi-Board-Befehl, siehe Multi-Board-Befehlsname, Seite 344.
nor	Normaler Listenbefehl, Seite 339
uk	Unverzögerter Kurzer Listenbefehl, Seite 340
vk	Verzögerter Kurzer Listenbefehl, Seite 340
var	Variabler Listenbefehl, Seite 341
mul	Multipler Listenbefehl, Seite 341

Initialisierung der Karte und Bildfeld-Korrektur

(n_) select_cor_table_list var 667

2D-Sprungbefehle

(n_) jump_abs nor 519
(n_) jump_rel nor 522
(n_) para_jump_abs nor 613
(n_) para_jump_rel nor 615
(n_) timed_jump_abs nor 910
(n_) timed_jump_rel nor 912
(n_) timed_para_jump_abs nor 918
(n_) timed_para_jump_rel nor 920

3D-Sprungbefehle(1)

(n_) jump_abs_3d nor 520
(n_) jump_rel_3d nor 523
(n_) para_jump_abs_3d nor 614
(n_) para_jump_rel_3d nor 616
(n_) timed_jump_abs_3d nor 911
(n_) timed_jump_rel_3d nor 913
(n_) timed_para_jump_abs_3d mul 919
(n_) timed_para_jump_rel_3d mul 921

micro_vector[*]-Befehle

(n_) micro_vector_abs nor 605
(n_) micro_vector_abs_3d nor 607
(n_) micro_vector_rel nor 608
(n_) micro_vector_rel_3d nor 609

2D-Markierbefehle

(n_) arc_abs nor 359
(n_) arc_rel nor 361
(n_) mark_abs nor 581
(n_) mark_ellipse_abs nor 588
(n_) mark_ellipse_rel nor 589
(n_) mark_rel nor 590
(n_) para_mark_abs nor 619
(n_) para_mark_rel nor 622
(n_) set_ellipse uk 699
(n_) timed_arc_abs nor 908
(n_) timed_arc_rel nor 909
(n_) timed_mark_abs nor 914
(n_) timed_mark_rel nor 916
(n_) timed_para_mark_abs nor 922
(n_) timed_para_mark_rel nor 924

3D-Markierbefehle(1)

(n_) arc_abs_3d nor 360
(n_) arc_rel_3d nor 362
(n_) mark_abs_3d nor 582
(n_) mark_rel_3d nor 591
(n_) para_mark_abs_3d nor 621
(n_) para_mark_rel_3d nor 623
(n_) timed_mark_abs_3d nor 915
(n_) timed_mark_rel_3d nor 917
(n_) timed_para_mark_abs_3d mul 923
(n_) timed_para_mark_rel_3d mul 925

Textbefehle

(n_) mark_char uk 583
(n_) mark_char_abs uk 584
(n_) mark_text var 595
(n_) mark_text_abs var 596
(n_) select_char_set uk 663

(1) Ohne Option "3D" nur eingeschränkte Funktionalität.

Datum, Uhrzeit, Seriennummern

(n_) mark_date nor 585
(n_) mark_date_abs nor 587
(n_) mark_serial nor 592
(n_) mark_serial_abs nor 594
(n_) mark_time nor 597
(n_) mark_time_abs nor 599
(n_) select_serial_set_list uk 670
(n_) set_serial_step_list nor 817
(n_) time_fix nor 905
(n_) time_fix_f nor 905
(n_) time_fix_f_off nor 906

Statusüberwachung und Diagnose

(n_) set_trigger vk 842
(n_) set_trigger4 vk 852
(n_) set_trigger8 vk 853

Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden

(n_) set_control_mode_list nor 686
(n_) set_extstartpos_list uk 706

Listen-Handling und Strukturierte Programmierung

(n_) list_continue nor 537
(n_) list_jump_pos uk 539
(n_) list_jump_rel uk 541
(n_) list_next uk 543
(n_) list_nop nor 543
(n_) list_repeat uk 544
(n_) list_until uk 546
(n_) long_delay nor 580
(n_) set_end_of_list nor 704
(n_) set_list_jump uk 754
(n_) set_wait nor 857

Unterprogramm-Befehle

(n_) list_call uk 530
(n_) list_call_abs uk 532
(n_) list_call_abs_repeat uk 534
(n_) list_call_repeat uk 536
(n_) list_return uk 545
(n_) sub_call uk 895
(n_) sub_call_abs uk 896
(n_) sub_call_abs_repeat uk 898
(n_) sub_call_repeat uk 900

Festlegen der Laserparameter

(n_) config_laser_signals_list vk 374
(n_) set_auto_laser_params_list vk 680
(n_) set_encoder_speed vk 702
(n_) set_firstpulse_killer_list uk 709
(n_) set_laser_delays uk 743
(n_) set_laser_pin_out_list uk 747
(n_) set_laser_pulses vk 750
(n_) set_laser_timing vk 752
(n_) set_pulse_picking_list uk 799
(n_) set_qswitch_delay_list uk 800
(n_) set_softstart_level_list nor 831
(n_) set_softstart_mode_list var 832
(n_) set_standby_list vk 834
(n_) set_vector_control uk 854
(n_) spot_distance uk 877

Festlegen der Galvanometerscannerparameter

(n_) set_delay_mode_list mul 696
(n_) set_jump_speed uk 737
(n_) set_mark_speed vk 755
(n_) set_scanner_delays vk 814
(n_) set_sky_writing_limit_list uk 820
(n_) set_sky_writing_list nor 821
(n_) set_sky_writing_min_speed_list uk 823
(n_) set_sky_writing_mode_list nor 825
(n_) set_sky_writing_para_list nor 828
(n_) set_sky_writing_scale_list uk 830

Koordinatentransformationen

(n_) set_angle_list var 673
(n_) set_defocus_list var (1) 690
(n_) set_defocus_2_list uk (1) 691
(n_) set_defocus_offset_list var (1) 693
(n_) set_matrix_list var 759
(n_) set_offset_list var 785
(n_) set_offset_xyz_list var (2) 788
(n_) set_scale_list var 805

Online-Positionierung

(n_) apply_mcbsp_list nor 358
(n_) set_mcbsp_global_matrix_list uk 763
(n_) set_mcbsp_global_rot_list uk 764
(n_) set_mcbsp_global_x_list uk 765
(n_) set_mcbsp_global_y_list uk 766
(n_) set_mcbsp_matrix_list uk 771
(n_) set_mcbsp_rot_list uk 778
(n_) set_mcbsp_x_list uk 779
(n_) set_mcbsp_y_list uk 780

Direkte Laser- und Scan-Kopf-Steuerung

(n_) laser_on_list var 525
(n_) laser_on_pulses_list var 526
(n_) laser_signal_off_list nor 528
(n_) laser_signal_on_list nor 529
(n_) para_laser_on_pulses_list var 617

Pixelausgabemodus

(n_) set_default_pixel_list uk 688
(n_) set_n_pixel var 784
(n_) set_pixel var 791
(n_) set_pixel_line nor 792
(n_) set_pixel_line_3d mul (2) 795

I/O-Befehle

(n_) clear_io_cond_list uk 372
(n_) periodic_toggle_list uk 634
(n_) read_io_port_list uk 645
(n_) rs232_write_text_list var 658
(n_) set_free_variable_list uk 729
(n_) set_io_cond_list uk 732
(n_) set_laser_power uk 748
(n_) set_mcbsp_out uk 772
(n_) set_mcbsp_out_ptr_list mul 777
(n_) set_port_default_list uk 797
(n_) write_8bit_port_list vk 948
(n_) write_da_1_list vk 951
(n_) write_da_2_list vk 952
(n_) write_da_x_list vk 954
(n_) write_io_port_list vk 957
(n_) write_io_port_mask_list vk 958
(n_) write_port_list uk 959

Bedingte Befehle

(n_) if_cond uk 504
(n_) if_not_cond uk 509
(n_) if_not_pin_cond uk 513
(n_) if_pin_cond uk 514
(n_) list_call_abs_cond uk 533
(n_) list_call_cond uk 535
(n_) list_jump_cond uk 538
(n_) list_jump_pos_cond uk 540
(n_) list_jump_rel_cond uk 542
(n_) sub_call_abs_cond uk 897
(n_) sub_call_cond uk 899
(n_) switch_ioport uk 901

(1) Nur mit Option "3D".

(2) Ohne Option "3D" nur eingeschränkte Funktionalität.

“Klassische” “Processing on the fly”-Listenbefehle

(n_) activate_fly_2d var (1) 351
(n_) activate_fly_2d_encoder mul (1) 352
(n_) activate_fly_xy var (1) 354
(n_) activate_fly_xy_encoder mul (1) 354
(n_) clear_fly_overflow uk 371
(n_) fly_return nor 428
(n_) fly_return_z nor 432
(n_) if_fly_x_overflow uk 505
(n_) if_fly_y_overflow uk 506
(n_) if_fly_z_overflow uk 507
(n_) if_not_activated uk 508
(n_) if_not_fly_x_overflow uk 510
(n_) if_not_fly_y_overflow uk 511
(n_) if_not_fly_z_overflow uk 512
(n_) init_fly_2d_list nor 512
(n_) park_position var (1) 624
(n_) park_return var (1) 628
(n_) set_ext_start_delay_list nor 708
(n_) set_fly_2d nor (1) 713
(n_) set_fly_limits uk 717
(n_) set_fly_limits_z uk 718
(n_) set_fly_rot nor (1) 719
(n_) set_fly_rot_pos nor (1) 720
(n_) set_fly_x nor (1) 723
(n_) set_fly_x_pos nor (1) 724
(n_) set_fly_y nor (1) 726
(n_) set_fly_y_pos nor (1) 727
(n_) set_fly_z nor (1) 728
(n_) set_mcbsp_in_list nor (2) 769
(n_) set_multi_mcbsp_in_list nor (2) 783
(n_) set_rot_center_list vk 801
(n_) simulate_ext_start nor 874
(n_) store_encoder uk 892
(n_) wait_for_encoder nor 938
(n_) wait_for_encoder_in_range mul 939
(n_) wait_for_encoder_in_range_mode mul 940
(n_) wait_for_encoder_mode nor 941
(n_) wait_for_mcbsp nor 943

”Fly-Erweiterung”-Listenbefehle

(n_) activate_fly_1_axis var 348
(n_) activate_fly_2_axes var 349
(n_) fly_disable_list nor 426
(n_) fly_return_1_axis var 429
(n_) fly_return_2_axes var 430
(n_) fly_return_3_axes var 431
(n_) park_position_1_axis var 626
(n_) park_position_2_axes var 627
(n_) park_return_1_axis var 630
(n_) park_return_2_axes var 631
(n_) set_fly_1_axis nor 710
(n_) set_fly_2_axes nor 711
(n_) set_fly_3_axes mul 715
(n_) wait_for_1_axis nor 934
(n_) wait_for_2_axes mul 936

Schrittmotoren ansteuern

(n_) stepper_abs_list uk 880
(n_) stepper_abs_no_list uk 881
(n_) stepper_control_list uk 882
(n_) stepper_enable_list uk 884
(n_) stepper_rel_list uk 887
(n_) stepper_rel_no_list uk 888
(n_) stepper_wait nor 889

Sprungmodus

(n_) set_jump_mode_list nor 736

Camming

(n_) camming nor 369

Wobbelmode

(n_) set_wobbel vk 858
(n_) set_wobbel_control uk 860
(n_) set_wobbel_direction uk 862
(n_) set_wobbel_mode vk 863
(n_) set_wobbel_mode_phase vk 866
(n_) set_wobbel_offset vk 867
(n_) set_wobbel_vector uk 868
(n_) set_wobbel_vector_2 uk 872

(1) Nur mit Option “Processing on the fly”.

(2) Ohne Option “Processing on the fly” nur eingeschränkte Funktionalität.

Listenbefehle nur für SCANahead-Systeme

(n_) activate_scanahead_autodelays_list uk . 356
(n_) set_scanahead_laser_shifts_list uk 807
(n_) set_scanahead_line_params_list uk 809

Listenbefehle nur für einen begrenzten Benutzerkreis(1)

(n_) fly_prediction uk 427
(n_) regulation3 nor 651
(n_) set_duty_cycle_table nor 698
(n_) set_laser_timing_table nor 753

Sonstige Listenbefehle

(n_) jump_abs_drill nor 521
(n_) jump_abs_drill_2 nor 521
(n_) jump_rel_drill nor 524
(n_) jump_rel_drill_2 nor 524
(n_) range_checking uk 636
(n_) save_and_restart_timer vk 660
(n_) set_mcbsp_out_oie_list uk 872
(n_) set_short_cmd_mode_list nor 818
(n_) set_zoom_list uk 872
(n_) store_timestamp_counter_list uk 894
(n_) wait_for_timestamp_counter nor 944
(n_) wait_for_timestamp_counter_long nor .. 945
(n_) wait_for_timestamp_counter_mode nor 946

(1) Diese Benutzer wurden im Umgang mit diesen RTC6-Befehlen separat informiert. Deswegen wird nur die Befehlstabelle aufgenommen und auf weitere Informationen in diesem Handbuch verzichtet.

Normaler Listenbefehl, Kurzer Listenbefehl, Variabler Listenbefehl und Multipler Listenbefehl

Die Listenbefehle des RTC6-Befehlssatzes unterscheiden sich teilweise in ihrer Befehlsausführungsdauer. Entsprechend werden sie in der Befehlsbeschreibung (in Kapitel 10.2 “RTC6-Befehlssatz”, Seite 342) unterschiedlich bezeichnet:

Normaler Listenbefehl, Seite 339
Kurzer Listenbefehl, Seite 339 • Unverzögerter Kurzer Listenbefehl, Seite 340 • Verzögerter Kurzer Listenbefehl, Seite 340
Variabler Listenbefehl, Seite 341
Multipler Listenbefehl, Seite 341

Normaler Listenbefehl

Ein Normaler Listenbefehl benötigt einen vollen 10 μs-Takt zur Befehlsausführung.

Kurzer Listenbefehl

Ein Kurzer Listenbefehl benötigt weniger als einen vollen 10 μs-Takt zur Befehlsausführung.

Daher kann ein Kurzer Listenbefehl zusammen mit dem jeweils nachfolgenden Listenbefehl unmittelbar nacheinander innerhalb eines 10 μs-Takts ausgeführt werden.

Dagegen wird ein Kurzer Listenbefehl, der einem Normalen Listenbefehl unmittelbar nachfolgt, erst im nachfolgenden 10 μs-Takt ausgeführt.

Durch die kürzere Befehlsausführung der Kurzen Listenbefehle verringert sich die gesamte Listenausführungszeit.

Zusätzlich kann bei einem Polygonzug auch zwischen den einzelnen Markier-Befehlen beispielsweise die Laserleistung variiert werden (siehe z.B. **set_laser_pulses**) oder einer der IO-Ports angesprochen werden (siehe z.B. **write_da_x_list**), ohne dass der Polygonzug unterbrochen wird (der Laser bleibt an).
Wenn dagegen ein definiertes Zeitverhalten (10 μs-Takt) erforderlich ist, kann nach jedem kurzen Listenbefehl ein zusätzlicher **list_nop**-Aufruf oder **list_continue**-Aufruf eingefügt werden. Diese beiden Befehle sorgen dafür, dass der jeweils nachfolgende Befehl garantiert erst im nächsten 10 μs-Takt ausgeführt wird. Durch **list_nop** wird ein Polygonzug unterbrochen (die “Laser active”-Lasersteuersignale werden ausgeschaltet), nicht dagegen durch **list_continue**.
Es sind bis zu 8 kurze Listenbefehle innerhalb eines 10 μs-Taktes möglich. Die maximale Anzahl wird jedoch in Abhängigkeit von der Belastung der RTC6-Karte und des DSP-Modus verringert. Kurze Listenbefehle, die den Ausgabe-Zeiger verändern (z.B. **sub_call**, **list_return** oder **list_jump_pos**) zählen dabei als 2 Befehle. Bei Überschreiten der Höchstanzahl wird ein 10 μs-Takt eingefügt (identisch zu einem zusätzlich eingefügten **list_continue**, der Laser bleibt im Polygonzug an).

Vor einem Normalen Listenbefehl sind innerhalb desselben 10 μs-Takts maximal 2 Kurze Listenbefehle erlaubt. Folgt der Normale Listenbefehl auf mehr als 2 Kurze Listenbefehle, dann werden die Kurzen Listenbefehle sofort und der Normale Listenbefehl um einen 10 μs-Takt verzögert ausgeführt.

Die maximale Anzahl von bis zu 8 Kurzen Listenbefehlen kann sich zukünftig ändern. Für unbedingt zukunftssichere Anwendungen sollte nur ein einziger Kurzer Listenbefehl vor einem Normalen Listenbefehl stehen, gegebenenfalls sollte explizit ein **list_continue**-Aufruf oder **list_nop**-Aufruf eingefügt werden (**list_nop** unterbricht den Polygonzug).

Hinweise

- Bei Normalen Listenbefehlen und Variablen Listenbefehlen setzt sich die Gesamtausführungsdauer aus der Befehlsausführungsdauer und der Ausführungsdauer des durch den Befehl veranlassten Prozesses zusammen. Ein nachfolgender Listenbefehl wird dann erst nach der Gesamtausführungsdauer ausgeführt.

  Beispiel: Die Gesamtausführungsdauer des Normalen Listenbefehls **mark_abs** beträgt 10 μs Befehlsausführungsdauer + die Ausführungsdauer der Markierung. Letztere hängt von den eingestellten Parametern für die Markierlänge, Markiergeschwindigkeit, Delays, etc. ab.

Unverzögerter Kurzer Listenbefehl, Verzögerter Kurzer Listenbefehl

Die meisten Kurzen Listenbefehle (wie **list_jump_pos**, **list_call**, **sub_call** oder **list_return**) werden vor dem jeweils nachfolgenden Listenbefehl und noch vor einem eventuellen Scanner-Delay (Jump-Delay, Mark-Delay, Polygon-Delay) ausgeführt. Diese werden in der jeweiligen Befehlsbeschreibung (in Kapitel 10.2 "RTC6-Befehlssatz", Seite 342) bezeichnet als:

- Unverzögerter Kurzer Listenbefehl

Einige Kurze Listenbefehle werden jedoch erst nach dem jeweiligen Scanner-Delay, also unmittelbar vor dem nachfolgenden Befehl ausgeführt. Diese werden in der jeweiligen Befehlsbeschreibung (in Kapitel 10.2 "RTC6-Befehlssatz", Seite 342) bezeichnet als:

- Verzögerter Kurzer Listenbefehl

Zu diesen zählen Befehle, die:

- Sich unmittelbar auf die Ausgabe oder auf die Laserleistung auswirken (z.B. **set_rot_center_list**, **set_wobbel_mode**, **write_da_x_list**, **set_laser_pulses**, **set_standby_list**, **set_mark_speed**, **set_encoder_speed**)
- Sich auf die Aufzeichnung oder Zeitmessung auswirken (z.B. **set_trigger** oder **save_and_restart_timer**)

Ohne diese verzögerte Ausführung würden solche Befehle z.B. dazu führen, dass eine Laserleistungsänderung bereits am Ende eines Markier-Befehls (also noch während des Mark-Delays oder Polygon-Delays) wirksam wäre und nicht erst zum Beginn des nachfolgenden Markier-Befehls.

Wenn ein solcher Befehl zur Leistungsänderung, etwa eine Peripherie-Ausgabe oder **set_laser_power**, unmittelbar vor einem **set_end_of_list** steht, wird er *nicht* ausgeführt, da **set_end_of_list** die Lasersteuerung "aufräumt". Fügen Sie ein **list_nop** ein, damit der Befehl tatsächlich ausgeführt wird.

set_trigger oder **save_and_restart_timer** würden bereits das Delay des vorangehenden Befehls mitberücksichtigen anstatt richtigerweise dasjenige des nachfolgenden Befehls.

Wenn diese Befehle unmittelbar vor einem **set_end_of_list** stehen, fügen Sie einen **list_nop**-Aufruf ein, damit sie noch innerhalb der Liste ausgeführt werden.

Bei mehreren unmittelbar aufeinander folgenden kurzen Listenbefehlen wird auch ein Verzögerter Kurzer Listenbefehl nur dann verzögert ausgeführt, wenn kein weiterer Verzögerter Kurzer Listenbefehl unmittelbar nachfolgt.

Es kann bei mehreren Verzögerten Kurzen Listenbefehlen in einer Abfolge von Kurzen Listenbefehlen immer nur der letzte "verzögerte" Befehl auch tatsächlich verzögert ausgeführt werden. Alle anderen davor werden sofort ausgeführt (noch vor einem Scanner-Delay).

Achten Sie bei der Reihenfolge der Befehle darauf, insbesondere innerhalb eines Polygonzugs, den wichtigsten "verzögerten" Befehl ans Ende zu stellen.

Alternativ kann auch (z.B. mit **list_nop**) explizit die Abarbeitung des Scanner-Delays initiiert werden, damit alle nachfolgenden kurzen Listenbefehle tatsächlich in jedem Fall "verzögert" ausgeführt werden.

Folgt ein Normaler Listenbefehl auf mehr als 2 kurze Listenbefehle, so er um einen 10 μs-Takt verzögert ausgeführt.

Variabler Listenbefehl

Bei einem Variablen Listenbefehl hängt die Befehlsausführungsdauer von weiteren Parametern oder vom Anwenderprogramm ab, siehe jeweilige Befehlsbeschreibungen.

Hinweise

- Bei Normalen Listenbefehlen und Variablen Listenbefehlen setzt sich die Gesamtausführungsdauer aus der Befehlsausführungsdauer und der Ausführungsdauer des durch den Befehl veranlassten Prozesses zusammen. Ein nachfolgender Listenbefehl wird dann erst nach der Gesamtausführungsdauer ausgeführt.

Multipler Listenbefehl

Ein Multipler Listenbefehl besteht aus mehreren Teilen und belegt entsprechend auch mehrere RTC6-Listenspeicherplätze.

Die vorderen Teile sind jeweils Unverzögerte Kurze Listenbefehle. Der letzte Teil ist jeweils ein Kurzer Listenbefehl, Normaler Listenbefehl oder Variabler Listenbefehl. Alle Teile werden unmittelbar nacheinander ausgeführt. Eventuell anliegende Verzögerte Kurze Listenbefehle werden davor ausgeführt.

Derzeit sind nur zweiteilige multiple Listenbefehle im RTC6-Befehlssatz enthalten, z.B. **wait_for_encoder_in_range** oder **set_pixel_line_3d**.

10.2 RTC6-Befehlssatz

In diesem Kapitel:

- Allgemeiner Aufbau der Befehlsbeschreibungen, Seite 343
- Datentypen, Seite 345
- Zeiger auf Arbeitsspeicherbereiche, Seite 345
- Befehlsbeschreibungen, Seite 346

10.2.1 Allgemeiner Aufbau der Befehlsbeschreibungen

Der allgemeine Aufbau der Befehlstabellen ist wie folgt[1]:

(1) Es wird eine programmiersprachenneutrale Form gewählt. Jede echte Programmiersprache hat ihre eigene individuelle Benennung.

Kategorie des Befehls	**beispiel_befehlsname_eins**
Funktion	Kurze Beschreibung wofür der RTC6-Befehl geschrieben wurde.
Einschränkung	Nennt beispielsweise das Verhalten des RTC6-Befehls, wenn eine bestimmte Option nicht freigeschaltet ist, siehe Kapitel 2.6 "Optionen", Seite 39. • Ohne Option "Processing on the fly" haben Befehle zur Aktivierung einer "Processing on the fly"-Korrektur keine Wirkung • Ohne Option "3D" werden 3D-Vektor-Befehle zwar als solche ausgeführt, es werden jedoch keine z-Achsen-Signale ausgegeben • Ohne Option "Zweite-Scan-Kopf-Steuerung" wirken sich Befehle, bei denen der Scan-Kopf-Ausgang explizit angegeben werden kann, nicht auf den Anschluss für zweiten Scan-Kopf aus (z.B. **set_matrix**)

Aufruf	Nennt die richtigen Schreibweisen sowie die Reihenfolge der Parameter. Kein Strichpunkt als Zeilenabschluss. Ein '&' (Adressoperator) wird nur in dieser Tabellenzeile verwendet und kennzeichnet einen Zeiger. Beispiele:
	`beispiel_befehlsname_eins( parameter_A, &parameter_B, parameter_C )`
	`beispiel_variable = beispiel_befehlsname_eins( parameter_A )`

Parameter	`A` Kurztext. Datentyp. Parameter sind in manchen C/C++-Codebeschreibungen mit `__in` gekennzeichnet.
	`C` Kurztext. Datentyp.
Parameter-Rückgaben	`B` Kurztext. Datentyp. Parameter-Rückgaben sind in manchen C/C++-Codebeschreibungen mit `__out` gekennzeichnet.
Rückgabe	Wenn implementiert: Nennt Rückgabewert und den Datentyp in generischer Form (Beispiel: Fehlercode #. Als 32-Bit-Wert ohne Vorzeichen.) Ein Wertebereich wird hier nur dann angegeben, wenn der tatsächlich verwendbare kleiner ist als der Wertebereich des Datentyps. Wenn nicht implementiert: "Keiner." Listenbefehle haben grundsätzlich keine Rückgabewerte.

Kategorie des Befehls	beispiel_befehlsname_eins
Multi-Board-Befehlsname	• Wenn vorhanden: "**n_** + Single-Board-Befehlsname", hier z.B. **n_beispiel_befehlsname_eins** • Wenn nicht vorhanden: "Nicht als Multi-Board-Befehl verfügbar." Siehe auch Kapitel 6.6.1 "Multi-Board-Programmierung", Seite 126 und Kapitel 6.6 "Mehrere RTC6 PCIe-Karten in einem PC verwenden", Seite 126.
Hinweise	• Weitere Informationen zu diesem RTC6-Befehl. • Verweise auf andere Kapitel und Dokumente.
RTC4→RTC6	Nennt die Unterschiede zum gleichnamigen RTC4-Befehl. Siehe auch RTC5-Kompatibilitätsmodus und Kapitel 2.11 "Hinweise an RTC5-Benutzer", Seite 60 sowie Kapitel 10.4 "Nicht mehr unterstützte RTC4-Befehle", Seite 963.
RTC5→RTC6	Nennt die Unterschiede zum gleichnamigen RTC5-Befehl. Siehe auch RTC5-Kompatibilitätsmodus und Kapitel 2.11 "Hinweise an RTC5-Benutzer", Seite 60 sowie Kapitel 10.5 "Nicht mehr unterstützte RTC5-Befehle", Seite 966.
Versionsinfo	Nennt z.B. die Mindest-Versionen von DLL, OUT, RBF, um den RTC6-Befehl benutzen zu können, letzte Änderung. Siehe auch RTC6_Aenderungshistorie.pdf.
Verweise	Links zu verwandten RTC6-Befehlen: **befehlsname_zwei**, **befehlsname_drei**

10.2.2 Datentypen

Die folgende Tabelle definiert die Formate und die Wertebereiche der verschiedenen Datentypen, die in den RTC6-Befehlen verwendet werden.

Datenformat	Wertebereich	Pascal	C, C++	C#
32-Bit-Wert ohne Vorzeichen	[0; (2^{32}–1)]	longword	unsigned long	uint
32-Bit-Wert mit Vorzeichen	[–2^{31}; +(2^{31}–1)]	longint	long	int
Wert im 64-Bit-IEEE Gleitkommaformat		double	double	double
Zeiger auf einen nullterminierten ANSI-String (1 Byte pro Zeichen)	4 Byte für Win32-Anwenderprogramme 8 Byte für Win64-Anwenderprogramme	pchar	char*	string

10.2.3 Zeiger auf Arbeitsspeicherbereiche

Einige RTC6-Befehle (wie **get_transform**/**get_transform_offset**, **get_values**, **get_waveform**, **transform** oder **upload_transform**) haben Zeiger auf Arbeitsspeicherbereiche als Parameter. In C# und Pascal werden dafür passende Zeigertypen verwendet (siehe Importdeklarationen). In C und C++ wird dafür der Datentyp ULONG_PTR verwendet. Dieser ist in den Importdeklarationen für C und C++ wie folgt definiert (ULONG_PTR = 32-Bit-Wert ohne Vorzeichen für Win32- Anwenderprogramme, ULONG_PTR = 64-Bit-Wert ohne Vorzeichen für Win64-Anwenderprogramme):

```
#if !defined(ULONG_PTR)
   #if !defined(_WIN64)
      #define ULONG_PTR UINT
   #else
      #define ULONG_PTR UINT64
   #endif // !defined(_WIN64)
#endif // !defined(ULONG_PTR)
```

Üblicherweise ist der Datentyp ULONG_PTR auch entsprechend in der Windows-Kopfdatei `BaseTsd.h` definiert.

10.2.4 Befehlsbeschreibungen

Die RTC6-Befehle sind alphabetisch geordnet.

Kontrollbefehl	**acquire_rtc**
Funktion	Übernimmt die angegebene RTC6-Karte für ein Anwenderprogramm.
Aufruf	`NoOfAcquiredCard = acquire_rtc( CardNo )`
Parameter	`CardNo` RTC6-DLL-interne Nummer (= RTC6-Kartenverwaltungsindex) der gewünschten Karte. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Der Rückgabewert ist: • `CardNo`, wenn die Übernahme erfolgreich war • 0, wenn die Karte bereits von einem anderen Anwenderprogramm übernommen ist oder die Versionsprüfung einen Fehler feststellt Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• **acquire_rtc** ist auch in einem Single-Board-System hilfreich, wenn die Verwendung der RTC6-Karte durch unterschiedliche Anwenderprogramme koordiniert werden muss. • **acquire_rtc** bleibt wirkungslos (Rückgabewert 0, **get_last_error**-Returncode `RTC6_PARAM_ERROR`), wenn: – `CardNo` > Zahl der bei der Initialisierung gefundenen RTC6 PCIe-Karten (siehe **rtc6_count_cards**) und dort keine RTC6 Ethernet-Karte eingetragen ist – `CardNo` = 0 (reale Karten zählen ab 1) • Zugriffsberechtigungen auf die vorhandenen Karten werden exklusiv (gleichzeitig jeweils immer nur an *ein* Anwenderprogramm) vergeben. **acquire_rtc** bleibt deshalb auch wirkungslos, wenn die angegebene Karte bereits durch ein anderes Anwenderprogramm in Besitz genommen ist (Rückgabewert 0, **get_last_error**-Returncode `RTC6_ACCESS_DENIED`). Dieses muss die RTC6-Karte (mit **release_rtc** oder **free_rtc6_dll**) explizit wieder freigeben, bevor die RTC6-Karte von einem anderen Anwenderprogramm mit **acquire_rtc** in Besitz genommen werden kann. Wenn die Karte andererseits bereits vor der Initialisierung eines Anwenderprogramms freigegeben wurde, so wird das Anwenderprogramm bei der Initialisierung mit **init_rtc6_dll** als für die Karte zugriffsberechtigt in die RTC6-Kartenverwaltung aufgenommen. Ein explizites **acquire_rtc** ist in diesem Fall nicht erforderlich und bleibt dann auch wirkungslos. Rückgabewert ist dann aber trotzdem `CardNo`. • Die RTC6-DLL und die Dateien `RTC6OUT.out`, `RTC6RBF.rbf` und `RTC6DAT.dat` können nicht in beliebigen Versionen miteinander kombiniert werden. **acquire_rtc** führt eine Versionskompatibilitätsprüfung durch. Sind `RTC6OUT.out`, `RTC6RBF.rbf` und `RTC6DAT.dat` noch nicht geladen, kann diese Prüfung zwar nicht explizit durchgeführt werden (**get_last_error**-Returncode `RTC6_TIMEOUT`), gilt aber dennoch als bestanden und verhindert damit die Übernahme nicht. Sind `RTC6OUT.out`, `RTC6RBF.rbf` und `RTC6DAT.dat` geladen und bei der Versionsprüfung wird ein Fehler festgestellt, dann wird der Zugriff verweigert (Rückgabewert 0, **get_last_error**-Returncode `RTC6_ACCESS_DENIED` | `RTC6_VERSION_MISMATCH`). Bei RTC6 Ethernet-Karten ist `RTC6OUT.out` durch `RTC6ETH.out` zu ersetzen. **init_rtc6_dll** nimmt RTC6 Ethernet-Karten nicht automatisch in Besitz.

Kontrollbefehl	acquire_rtc
Hinweise (Forts.)	• Bei RTC6 Ethernet-Karten kann **acquire_rtc** bis zu 1 s lang dauern. • Eine mit **acquire_rtc** erfolgreich übernommene Karte wird nicht automatisch zur aktiven Karte. Die Aktivierung einer Karte erfolgt nur mit **select_rtc** oder **init_rtc6_dll**. • Laufende Karten werden durch **acquire_rtc** nicht angehalten und insbesondere auch nicht initialisiert. • **acquire_rtc** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Siehe auch Kapitel 6.7.1 "Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm", Seite 131.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**init_rtc6_dll**, **select_rtc**, **free_rtc6_dll**, **release_rtc**, **rtc6_count_cards**

Variabler Listenbefehl	**activate_fly_1_axis**
Funktion	"Fly-Erweiterung"-Befehl: Aktiviert eine 1-Achsen-"Processing on the fly"-Anwendung.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **activate_fly_1_axis** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_1_axis( Axis, Mode, Scale, Offset )`
Parameter	`Axis` Achse aus Tabelle 3, Seite 271. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: 1…2. `Mode` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: 1…4. `Scale` Skalierungsfaktor. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: • 1/256 ≤ |`Scale`| ≤ 16.000,0 bei Linear-Achse (1 oder 2) `Scale` kann + oder – annehmen. Nur der Absolutbetrag ist eingeschränkt. `Offset` Offset auf den Encoderwert. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_activate_fly_1_axis**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **activate_fly_1_axis** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • **activate_fly_1_axis** belegt zwei RTC6-Listenspeicherplätze. • **activate_fly_1_axis** benötigt zwei 10 μs-Takte zur Ausführung. • Bei der angegebenen Achse darf weder eine "Processing on the fly"-Korrektur noch eine Rotationskorrektur aktiv sein. Die Achse kann jedoch als Linear-Achse mit anderen Linear-Achsen kombiniert werden. • Eine xy-Verfahrtischkompensation wird automatisch ausgeführt, wenn nach der erfolgreichen **activate_fly_1_axis**-Ausführung Achse 1 und Achse 2 mit zwei physikalisch verschiedenen Encodermodi aktiviert sind (z.B. bedeuten Modus 1 und 3 den gleichen physikalischen Encoder) und eine xy-Verfahrtischkompensation definiert und aktiviert ist (siehe **load_fly_2d_table**). • Bei einem unzulässigen Parameter-Wert wird **activate_fly_1_axis** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Siehe auch Hinweise zu **activate_fly_2_axes**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**activate_fly_2_axes**

Variabler Listenbefehl	**activate_fly_2_axes**
Funktion	"Fly-Erweiterung"-Befehl: Aktiviert eine 2-Achsen-"Processing on the fly"-Anwendung.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **activate_fly_2_axes** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_2_axes( ModeX, ScaleX, OffsetX, ModeY, ScaleY, OffsetY)`
Parameter	`ModeX` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. `ScaleX` Skalierungsfaktor. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: • 1/256 ≤ |`Scale`| ≤ 16.000,0 bei Linear-Achse (`1` oder `2`) `Scale` kann + oder – annehmen. Nur der Absolutbetrag ist eingeschränkt. `OffsetX` Offset auf den Encoderwert. Als 32-Bit-Wert mit Vorzeichen. `ModeY` Wie `ModeX`. `ScaleY` Wie `ScaleX`. `OffsetY` Wie `OffsetX`.
Multi-Board-Befehlsname	**n_activate_fly_2_axes**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **activate_fly_2_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • Die Achsen sind automatisch `1` und `2`. Die Modi müssen zwei physikalisch verschiedene Encodermodi sein (Modus 1 und 3 oder 2 und 4 sind nicht erlaubt). • **activate_fly_2_axes** belegt zwei RTC6-Listenspeicherplätze. • **activate_fly_2_axes** benötigt zwei 10 µs-Takte zur Ausführung. • Bei Achse `1` und `2` darf weder eine "Processing on the fly"-Korrektur noch eine Rotationskorrektur aktiv sein. Die Achsen können jedoch als Linear-Achsen mit der dritten Linear-Achse kombiniert werden. • Bei einem unzulässigen Parameter-Wert wird **activate_fly_2_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Wenn eine xy-Verfahrtischkompensation definiert und aktiviert ist (siehe **load_fly_2d_table**), dann wird diese automatisch ausgeführt. Siehe auch Hinweise zu **activate_fly_2d** (Encoderreset, Fehler-Bit, **get_marking_info**).

Variabler Listenbefehl	activate_fly_2_axes

Hinweise (Forts.)	• Folgende Befehlsaufrufe werden gleichartig ausgeführt:
	– `activate_fly_2_axes( 1, ScaleX, 0, 2, ScaleY, 0 )` =
	`activate_fly_2d( ScaleX, ScaleY )`
	oder
	`activate_fly_xy( ScaleX, ScaleY )`
	– `activate_fly_2_axes( 1, ScaleX, EncX, 2, ScaleY, EncY )` =
	`activate_fly_2d_encoder( ScaleX, ScaleY, EncX, EncY )`
	oder
	`activate_fly_xy_encoder( ScaleX, ScaleY, EncX, EncY )`
	"xy" und "2d" wird bei `wait_for_1_axis` und `wait_for_2_axes` unterschieden!
	– `activate_fly_2_axes( 1, ScaleX, OffsetX, 2, ScaleY, OffsetY )` =
	`{`
	`activate_fly_1_axis( 1, 1, ScaleX, OffsetX );`
	`activate_fly_1_axis( 2, 2, ScaleY, OffsetY );`
	`}`

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**activate_fly_1_axis**

Variabler Listenbefehl	**activate_fly_2d**
Funktion	Aktiviert eine **set_fly_2d**-"Processing on the fly"-Anwendung ohne Encoderreset.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **activate_fly_2d** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_2d( ScaleX, ScaleY )`
Parameter	`ScaleX` Skalierungsfaktor wie bei **activate_fly_2d_encoder**. `ScaleY` Wie `ScaleX`.
Multi-Board-Befehlsname	**n_activate_fly_2d**
Hinweise	• Wenn keine "Processing on the fly"-Korrektur aktiv ist, aktiviert **activate_fly_2d** eine **set_fly_2d**-"Processing on the fly"-Korrektur, siehe Kapitel 8.6.4 "2D-Bewegungen ausgleichen", Seite 261. Im Gegensatz zu **set_fly_2d** setzt dabei **activate_fly_2d** die Encoder nicht zurück, sondern rechnet die Koordinatenwerte so aus, dass die "Processing on the fly"-korrigierte Ausgabe mit der aktuellen Ausgabe übereinstimmt. Falls die dann aktuell umgerechneten Koordinatenwerte außerhalb des virtuellen 29-Bit-Bildfelds zu liegen kommen würden, wird "Processing on the fly" nicht aktiviert. Dann wird ein Fehler-Bit gesetzt, das mit **get_marking_info** (`Bit #9`) ausgelesen werden kann. • Bei *aktiver* "Processing on the fly"-Korrektur ist **activate_fly_2d** ein Kurzer Listenbefehl ohne weitere Auswirkungen und setzt lediglich ein Fehler-Bit, welches mit **get_marking_info** (`Bit #9`) ausgelesen werden kann. **activate_fly_2d** kann damit auch nicht dafür verwendet werden, den "Processing on the fly"-Modus selbst oder die Skalierungsfaktoren beim selben Modus zu ändern. • Das Fehler-Bit kann auch mit dem kurzen Listenbefehl **if_not_activated** abgefragt werden, um in eine geeignete Fehlerbehandlungsroutine zu verzweigen. • Eine erfolgreiche Aktivierung mit **activate_fly_2d** löscht ein gesetztes Fehler-Bit *nicht*. Es bleibt für **get_marking_info** stehen, bis **get_marking_info** aufgerufen wurde. • Für unzulässige Parameterwerte (z.B. `ScaleX` = 0) wird **activate_fly_2d** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • **activate_fly_2d** lässt die Lasersteuersignale unangetastet (Signale für den "Laser active"-Betrieb bleiben ein/aus, wenn sie ein/aus sind).
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_fly_2d**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_2d**, **activate_fly_2d_encoder**, **activate_fly_xy**

Multipler Listenbefehl	**activate_fly_2d_encoder**
Funktion	Aktiviert eine **set_fly_2d**-"Processing on the fly"-Anwendung mit Encoderreset und Encoderoffset.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **activate_fly_2d_encoder** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_2d_encoder( ScaleX, ScaleY, EncX, EncY )`
Parameter	`ScaleX` Skalierungsfaktor wie bei **set_fly_2d**. `ScaleY` Wie `ScaleX`. `EncX` Encoderoffset. Als 32-Bit-Wert mit Vorzeichen. `EncY` Wie `EncX`.
Multi-Board-Befehlsname	**n_activate_fly_2d_encoder**
Hinweise	• **activate_fly_2d_encoder** belegt zwei RTC6-Listenspeicherplätze und benötigt auch zwei 10 *µs*-Takte zur Ausführung (der erste Teil des Befehls ist *kein* Kurzer Listenbefehl). • **activate_fly_2d_encoder** ist eine Kombination aus **set_fly_2d** (Encoderreset) und **activate_fly_2d** (siehe auch Hinweise dort). Allerdings werden in **activate_fly_2d_encoder** zur Berechnung der nicht-"Processing on the fly"-korrigierten virtuellen Bildfeld-Koordinaten nicht die aktuellen (zurückgesetzten) Encoderwerte, sondern die Parameterwerte `EncX` und `EncY` verwendet. Zur Fehlerbehandlung siehe Hinweise zu **activate_fly_2d**. • Wegen dieser Kombination spart **activate_fly_2d_encoder** die (oft lange, und für die "Processing on the fly"-Aktivierung ohne diesen Befehl unter Umständen notwendige) Verfahrtischbewegung von der Initialisierungsposition (z.B. in der linken, unteren Ecke) zur Mitte und wieder zurück, ein. • In der Folge werden alle Encoderwerte mit den Offsets `EncX` und `EncY` beaufschlagt, bevor die "Processing on the fly"-Korrektur angewendet wird. Die übrigen Encoder-bezogenen Befehle beziehen sich auf die tatsächlichen Encoderwerte und funktionieren wie bisher. • Ist der Wert von `EncX` oder `EncY` unzulässig (d. h. die "Processing on the fly"-korrigierten virtuellen Bildfeld-Koordinaten liegen außerhalb der Grenzen des virtuellen Bildfelds), wird **activate_fly_2d_encoder** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Ist der Wert von `ScaleX` oder `ScaleY` unzulässig (siehe **set_fly_2d**), wird zwar der erste Teil auf die Karte übertragen (und führt damit den Encoderreset aus), jedoch wird der zweite Teil durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Bei *aktiver* "Processing on the fly"-Korrektur ist **activate_fly_2d_encoder** ein Kurzer Listenbefehl ohne weitere Auswirkungen und setzt lediglich ein Fehler-Bit, das mit **get_marking_info** (`Bit #9`) ausgelesen werden kann. **activate_fly_2d_encoder** kann damit auch nicht dafür verwendet werden, den "Processing on the fly"-Modus selbst, die Skalierungsfaktoren oder die Encoderoffsets beim selben Modus zu ändern.

Multipler Listenbefehl	**activate_fly_2d_encoder**
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_fly_2d**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 610, OUT 610, RBF 615.
Verweise	**set_fly_2d**, **activate_fly_2d**

Variabler Listenbefehl	**activate_fly_xy**
Funktion	Aktiviert eine **set_fly_x**/**set_fly_y**-"Processing on the fly"-Anwendung ohne Encoderreset.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **activate_fly_xy** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_xy( ScaleX, ScaleY )`
Parameter	`ScaleX` Skalierungsfaktor wie bei **set_fly_x**. `ScaleY` Skalierungsfaktor wie bei **set_fly_y**.
Multi-Board-Befehlsname	**n_activate_fly_xy**
Hinweise	• Wie **activate_fly_2d** mit dem Unterschied, dass eine **set_fly_x**/**set_fly_y**-"Processing on the fly"-Session aktiviert wird.
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_fly_x**/**set_fly_y**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_x**, **set_fly_y**, **activate_fly_xy_encoder**

Multipler Listenbefehl	**activate_fly_xy_encoder**
Funktion	Aktiviert eine **set_fly_x**/**set_fly_y**-"Processing on the fly"-Anwendung mit Encoderreset und Encoderoffset.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **activate_fly_xy_encoder** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_xy_encoder( ScaleX, ScaleY, EncX, EncY )`
Parameter	`ScaleX` Skalierungsfaktor wie bei **set_fly_x**. `ScaleY` Skalierungsfaktor wie bei **set_fly_y**. `EncX` Encoderoffset. Als 32-Bit-Wert mit Vorzeichen. `EncY` Encoderoffset. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_activate_fly_xy_encoder**
Hinweise	• Wie **activate_fly_2d_encoder** mit dem Unterschied, dass eine **set_fly_x**/**set_fly_y**-"Processing on the fly"-Session aktiviert wird.
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_fly_2d**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 610, OUT 610, RBF 615.
Verweise	**set_fly_x**, **set_fly_y**, **activate_fly_xy**, **activate_fly_2d_encoder**

Kontrollbefehl	**activate_scanahead_autodelays**
Funktion	Nur für SCANahead-Systeme. Schaltet die automatische (dynamische) Berechnung der Scanner-Delays und Laser-Delays ein oder aus. Gibt den gegenwärtigen Modus nach Ausführung von **activate_scanahead_autodelays** zurück.
Aufruf	`CurrentMode = activate_scanahead_autodelays( Mode )`
Parameter	`Mode` Als 32-Bit-Wert mit Vorzeichen. `-1`: Gibt nur den gegenwärtigen Modus zurück. Der bestehende Mode wird nicht geändert. `0`: Schaltet die automatische Berechnung aus. `1`: Schaltet die automatische Berechnung ein (`Mode = 1`), wenn zuvor **set_scanahead_params** aufgerufen wurde. Ansonsten bleibt der `Mode` nach Ausführung des Befehls `0`.
Rückgabe	Modus. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_activate_scanahead_autodelays**
Hinweise	• Im `Mode = 0` werden die normalen Scanner-Delays und Laser-Delays verwendet (siehe **set_scanner_delays** und **set_laser_delays**). Im `Mode = 1` werden die Scanner-Delays und Laser-Delays automatisch berechnet und gesetzt. Die Werte aus `Mode = 0` werden nicht überschrieben und können nach Abschaltung im `Mode = 0` weiter verwendet werden (wie bei nicht-SCANahead-Systemen). • **activate_scanahead_autodelays** wirkt auch auf Sky-Writing: Die Parameter der Sky-Writing-Befehle `Timelag` (außer für die eigentliche Aktivierung von Sky-Writing), `Nprev` und `Npost` bleiben ohne Wirkung. Sie werden aber sofort wirksam, wenn die automatische Delay-Berechnung ausgeschaltet wird. Der Parameter `LaserOnShift` ist dagegen immer wirksam.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600.
Verweise	**activate_scanahead_autodelays_list**, **set_scanner_delays**, **set_laser_delays**

Unverzögerter Kurzer Listenbefehl	**activate_scanahead_autodelays_list**
Funktion	Nur für SCANahead-Systeme. Wie **activate_scanahead_autodelays**, aber ein Listenbefehl.
Aufruf	`CurrentMode = activate_scanahead_autodelays_list( Mode )`
Parameter	`Mode` Als 32-Bit-Wert ohne Vorzeichen. Werte siehe **activate_scanahead_autodelays**.
Multi-Board-Befehlsname	**n_activate_scanahead_autodelays_list**
Hinweise	• **activate_scanahead_autodelays_list** wird mit dem nächsten zu berechnenden Delay wirksam. • `Mode = -1` ist bei **activate_scanahead_autodelays_list** nicht verfügbar.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600.
Verweise	**activate_scanahead_autodelays**

Kontrollbefehl	**apply_mcbsp**
Funktion	Ruft die zuletzt für die "Lokale Online-Positionierung" vollständig über die McBSP-Schnittstelle übertragenen Werte ab und definiert einen Offset und/oder die Drehmatrix M_R oder eine allgemeine Transformationsmatrix M_T für nachfolgende Koordinatentransformationen.
Aufruf	`apply_mcbsp( HeadNo, at_once )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. = 1: Die Definition wirkt nur auf den Anschluss für ersten Scan-Kopf. = 2: Die Definition wirkt nur auf den Anschluss für zweiten Scan-Kopf. = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`at_once` Legt fest, wann die definierte Transformation wirksam wird. Als 32-Bit-Wert ohne Vorzeichen. = 0: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten Listenbefehls berechnet und auf die dann aktuelle Position angewendet. = 1: Die neue Gesamttransformation wird sofort berechnet und auf die aktuelle Position angewendet (oder vor dem nächsten Listenbefehl, wenn gerade der BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus gesetzt ist). = 2: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten **jump_abs**, **jump_rel**, **goto_xy** oder **goto_xyz** berechnet und auf die dann aktuelle Position angewendet. > 2: Wie `at_once` = 2.
Multi-Board-Befehlsname	**n_apply_mcbsp**
Hinweise	• Das Einlesen der Daten zur "Lokale Online-Positionierung" über die McBSP-Schnittstelle muss zuvor mit **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** oder **set_mcbsp_matrix** (oder mit den entsprechenden Listenbefehlen) aktiviert werden. Je nach Konfigurierung definiert **apply_mcbsp** nur (wie mit **set_offset**) einen x-Offset und/oder y-Offset oder auch (wie mit **set_angle**) eine Drehmatrix oder (wie mit **set_matrix**) eine allgemeine Matrixoperation (siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240). Wie bei den in Kapitel 8.2 "Koordinatentransformationen", Seite 236 beschriebenen Befehlen bestimmt der Parameter `at_once`, wann die neudefinierte Gesamttransformation wirksam wird. • Zuvor mit **set_angle**, **set_offset** oder **set_matrix** definierte Transformationen werden durch **apply_mcbsp** überschrieben. Zuvor mit **set_scale** oder **set_defocus** definierte Transformationen und Fokusverschiebungen, sowie mit **set_offset_xyz** definierte z-Offsets werden dagegen bei der Neuberechnung der Gesamttransformation weiterhin unverändert berücksichtigt. • Nach einer Datenübertragung mit **set_angle**, **set_offset** oder **set_matrix** erfolgte Neudefinitionen überschreiben die über die McBSP-Schnittstelle definierten Koordinatentransformationen. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Abschnitt "RTC6 PCIe-Karte als Empfänger", Seite 88.

Kontrollbefehl	apply_mcbsp
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**apply_mcbsp_list**, **set_mcbsp_x**, **set_mcbsp_y**, **set_mcbsp_rot**, **set_mcbsp_matrix**

Normaler Listenbefehl	apply_mcbsp_list
Funktion	Wie **apply_mcbsp**, aber ein Listenbefehl.
Aufruf	`apply_mcbsp_list( HeadNo, at_once )`
Parameter	`HeadNo` Wie **apply_mcbsp**.
	`at_once` Legt fest, wann die definierte Transformation wirksam wird. Als 32-Bit-Wert ohne Vorzeichen. = 0: Die Transformationswerte werden nur gesammelt und zwischengespeichert, aber die Transformation wird nicht ausgeführt, solange dies nicht durch eine andere Koordinatentransformation (z.B. mit einem Listenbefehl mit `at_once` = 1 oder einem entsprechenden Kontrollbefehl) aktiviert wird. = 1: Die Transformation wird zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und noch vor dem nächsten Listenbefehl ausgeführt. = 2: Die Transformationswerte werden (wie bei `at_once` = 0) nur gesammelt und zwischengespeichert, die Transformation wird aber bei der Ausführung des nächsten **jump_abs** oder **jump_rel** (nur wenn keine Liste ausgeführt wird: auch **goto_xy** oder **goto_xyz**) zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewendet. > 2: Wie `at_once` = 2.
Multi-Board-Befehlsname	**n_apply_mcbsp_list**
Hinweise	• Siehe **apply_mcbsp**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**apply_mcbsp**

Normaler Listenbefehl	**arc_abs**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit von der aktuellen Position entlang eines Kreisbogens mit dem angegebenen Winkel um den angegebenen Mittelpunkt (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`arc_abs( X, Y, Angle )`
Parameter	`X` Absolute x-Koordinate des Bogenzentrums. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y` Wie `X` (analog).
	`Angle` Bogenwinkel. In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat. Ein positives Vorzeichen bedeutet "im Uhrzeigersinn". Zulässiger Wertebereich: [–3.600,0°…+3.600,0°] (±10 Vollkreise). Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_arc_abs**
Hinweise	• Wenn die Markiergeschwindigkeit zuvor nicht explizit über **set_mark_speed** oder **set_mark_speed_ctrl** festgelegt wurde, wird die Markierung mit der voreingestellten Markiergeschwindigkeit von 1.000 *Bits/ms* ausgeführt. • Die Signale für den "Laser active"-Betrieb werden automatisch am Anfang der Markierung eingeschaltet (oder bleiben nach einem unmittelbar vorhergehenden [*]mark[*]-Befehl oder Bogen-Befehl eingeschaltet). Die eingestellten Scanner-Delays und Laser-Delays werden dabei berücksichtigt, siehe Kapitel 7.2 "Delayeinstellungen – Scan-Kopf-Steuerung und Laser-Steuerung abstimmen", Seite 148. Beachten Sie, dass im Sky-Writing-Modus andere Delays ausgeführt werden. Ausnahme: Bogen-Befehle der Bogenlänge 0, siehe Abschnitt "Hinweise", Seite 152.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mark_speed**, **set_scanner_delays**, **arc_rel**, **timed_arc_abs**, **mark_abs**, **arc_abs_3d**, **mark_ellipse_abs**

Normaler Listenbefehl	**arc_abs_3d**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit von der aktuellen Position helixförmig um eine Achse parallel zur z-Achse. Die x- und y-Komponenten beschreiben dabei einen Kreisbogen mit dem angegebenen Winkel um die angegebene Achse und die z-Komponente gleichzeitig eine lineare Bewegung von der aktuellen Position zum angegebenen Endwert. Die Lage der Helixachse und der z-Endwert werden als absolute Koordinatenwerte angegeben.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **arc_abs_3d** wie **arc_abs**.
Aufruf	`arc_abs_3d( X, Y, Z, Angle )`
Parameter	`X` Lage der Helixachse (parallel zur z-Achse) als absolute x-Koordinate. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y` Wie `X` (analog).
	`Z` Absolute z-Endkoordinate. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Angle` Bogenwinkel. In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat. Ein positives Vorzeichen bedeutet "im Uhrzeigersinn". Zulässiger Wertebereich: [–3.600,0°…+3.600,0°] (±10 Vollkreise). Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_arc_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **arc_abs_3d** gleich wie **arc_abs** (siehe Hinweise dort). • Die z-Bewegung wird für die Berechnung der Anzahl der Mikroschritte nicht berücksichtigt.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X`, `Y` und `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**arc_abs**, **arc_rel_3d**

Normaler Listenbefehl	**arc_rel**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit von der aktuellen Position entlang eines Kreisbogens mit dem angegebenen Winkel um den angegebenen Mittelpunkt (relative Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`arc_rel( dX, dY, Angle )`
Parameter	`dX` Relative x-Koordinate des Bogenzentrums. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`dY` Wie `dX` (analog).
	`Angle` Bogenwinkel. In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat. Ein positives Vorzeichen bedeutet “im Uhrzeigersinn”. Zulässiger Wertebereich: [–3.600,0°…+3.600,0°] (±10 Vollkreise). Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_arc_rel**
Hinweise	• Die Koordinaten des Bogenzentrums werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **arc_rel** identisch mit **arc_abs** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mark_speed**, **set_scanner_delays**, **arc_abs**, **timed_arc_rel**, **mark_rel**, **arc_rel_3d**, **mark_ellipse_rel**

Normaler Listenbefehl	**arc_rel_3d**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit von der aktuellen Position helixförmig um eine Achse parallel zur z-Achse. Die x- und y-Komponenten beschreiben dabei einen Kreisbogen mit dem angegebenen Winkel um die angegebene Achse und die z-Komponente gleichzeitig eine lineare Bewegung von der aktuellen Position zum angegebenen Endwert. Die Lage der Helixachse und der z-Endwert werden als relative Koordinatenwerte angegeben.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **arc_rel_3d** wie **arc_rel**.
Aufruf	`arc_rel_3d( dX, dY, dZ, Angle )`
Parameter	`dX` Lage der Helixachse (parallel zur z-Achse) als relative x-Koordinate. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`dY` Wie `dX` (analog).
	`dZ` Relative z-Endkoordinate. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Angle` Bogenwinkel. In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat. Ein positives Vorzeichen bedeutet "im Uhrzeigersinn". Zulässiger Wertebereich: [–3.600,0°…+3.600,0°] (±10 Vollkreise). Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_arc_rel_3d**
Hinweise	• Die Lage der Helixachse (`dX`, `dY`) und der z-Endwert (`dZ`) werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **arc_rel_3d** identisch mit **arc_abs_3d** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **arc_abs_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**arc_abs_3d**, **arc_rel**

Kontrollbefehl	**auto_cal**
Funktion	Nur für einige abgekündigte Systeme. Steuert die Funktionen zur (automatischen Selbst-) Kalibrierung des am angegebenen Scan-Kopf-Anschluss angeschlossenen Scan-Systems.
Aufruf	`ErrorCode = auto_cal( HeadNo, Command )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf. Erfordert Option "Zweite-Scan-Kopf-Steuerung".
	`Command` Kontrollparameter. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…4]. = 0: Die RTC6 ermittelt die aktuellen Home-In-Positionen, speichert sie als Home-In-Referenzwerte in der RTC6-DLL und im Flash-Speicher und initialisiert die Gain-Werte und Offset-Werte (Gain = 1.0, Offset = 0). = 1: Die RTC6 ermittelt die aktuellen Home-In-Positionen, berechnet und setzt neue Gain-Werte und Offset-Werte und aktiviert dadurch eine Drift-Kompensation. = 2: Die RTC6 deaktiviert die Drift-Kompensation durch Initialisierung der Gain-Werte und Offset-Werte (Gain = 1.0, Offset = 0). = 3: Die RTC6 ermittelt die aktuellen Home-In-Positionen (lässt aber im Gegensatz zu `Command` = 1 die Gain-Werte und Offset-Werte unverändert, aktiviert also auch keine Drift-Kompensation). = 4: Die RTC6 überprüft die ASC-Hardware ob am angegebenen Scan-Kopf-Anschluss ein Scan-System angeschlossen ist, das mit einem internen Sensorsystem zur automatischen Selbstkalibrierung – Home-In-Sensoren – ausgestattet ist) und gibt gegebenenfalls den Typ oder den Status des Sensorsystems zurück. Der ermittelte Typ wird auch im Flash-Speicher gespeichert.
Rückgabe	Fehlercode oder Typ des Sensorsystems. Als 32-Bit-Wert ohne Vorzeichen. 3 **auto_cal** kann nicht ausgeführt werden, weil gerade der BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus gesetzt ist. 6 Parameterfehler. Die folgenden Fehlercodes werden nur nach `Command` = 0…3 zurückgegeben: 0 Kein Fehler. 1, 10, 11 Home-In-Sensor nicht gefunden (dies kann auch bedeuten, dass ein Home-In-Sensor defekt ist) (1: bei x-Achse (Galvanometerscanner 2) 10: bei y-Achse (Galvanometerscanner 1) 11: bei beiden Achsen). 2, 20, 22 Streuung der Messwerte in einem Messzyklus zu hoch (2: bei x-Achse / 20: bei y-Achse / 22: bei beiden Achsen).

Kontrollbefehl	**auto_cal**
Rückgabe (Forts.)	4, 40, 44 Referenzwerte nicht gefunden (nur bei `Command` = 1 und 3) (4: bei x-Achse / 40: bei y-Achse / 44: bei beiden Achsen). 5, 50, 55 Kalibrierungsfehler (Fehler während der Kalibrierung oder Fehler bei den Referenzwerten) (5: bei x-Achse / 50: bei y-Achse / 55: bei beiden Achsen). 9, 90 Sensor für x-Achse oder y-Achse ist defekt. Wird nur nach `Command` = 0, 1 oder 3 zurückgegeben. Der folgende Wert wird nur nach `Command` = 0 oder 4 zurückgegeben, und auch nur dann, wenn kein anderer Fehler vorliegt: 8 Downloadfehler. Die Werte sind möglicherweise nicht gespeichert. Bei diesem Fehler wird in jedem Fall auch der **get_last_error**-Returncode `RTC6_FLASH_ERROR` erzeugt. Die folgenden Werte werden nur nach `Command` = 4 zurückgegeben: 100 Bei beiden Achsen: Sensorsystem vom Typ1 eingebaut, funktioniert. 200 Bei beiden Achsen: Sensorsystem vom Typ2 eingebaut, funktioniert. 19 x-Achse (Galvanometerscanner 2): Sensorsystem vom Typ1 eingebaut, funktioniert. y-Achse (Galvanometerscanner 1): Sensorsystem defekt. 29 x-Achse (Galvanometerscanner 2): Sensorsystem vom Typ2 eingebaut, funktioniert. y-Achse (Galvanometerscanner 1): Sensorsystem defekt. 91 x-Achse (Galvanometerscanner 2): Sensorsystem defekt. y-Achse (Galvanometerscanner 1): Sensorsystem vom Typ1 eingebaut, funktioniert. 92 x-Achse (Galvanometerscanner 2): Sensorsystem defekt. y-Achse (Galvanometerscanner 1): Sensorsystem vom Typ2 eingebaut, funktioniert. 99 Bei beiden Achsen: Sensorsystem defekt. 255 Bei beiden Achsen: kein Sensorsystem eingebaut.
Multi-Board-Befehlsname	**n_auto_cal**
Hinweise	• Zur Verwendung von **auto_cal** siehe Kapitel 8.10 "Automatische Selbstkalibrierung", Seite 287. • Die RTC6-Karte fährt die Galvanometerscanner am Ende der Ausführung von **auto_cal** immer wieder in die gleiche Stellung wie vor dem Aufruf, gegebenenfalls in eine um geänderte Gain-Werte und Offset-Werte korrigierte Position. • Bei der Ermittlung der aktuellen Home-In-Positionen mit **auto_cal**(`Command` = 0, 1 oder 3) werden die aktuellen Gain- und Offset-Korrekturen der Galvanometerscanner nicht berücksichtigt, ebenso wenig irgendwelche Kopfkorrekturen oder die momentanen Positionen der Galvanometerscanner. • Nach dem erstmaligen oder erneuten Anschließen eines Scan-Systems sollte eine Referenzwertbestimmung mit **auto_cal**(`Command` = 0) durchgeführt werden. Die RTC6 verwendet sonst gegebenenfalls bei einer späteren automatischen Selbstkalibrierung die gespeicherten Referenzwerte für ein zuvor betriebenes (gegebenenfalls anderes) Scan-System.

Kontrollbefehl	auto_cal
Hinweise (Forts.)	• Nach der Initialisierung der RTC6 ist die Drift-Kompensation abgeschaltet (Gain = 1.0, Offset = 0). Zuvor bestimmte Referenzwerte sind jedoch weiterhin verfügbar. • Sind keine geeigneten Home-In-Referenzwerte gespeichert oder ist das Scan-System nicht mit Home-In-Sensoren ausgestattet, bricht die Messroutine für **auto_cal**(`Command` = 1) (achsenspezifisch) automatisch ab und stellt den Ausgangszustand wieder her. • **auto_cal** wird nicht ausgeführt für einen unzulässigen Wert von `HeadNo` oder `Command` (Rückgabewert 6, **get_last_error**-Returncode `RTC6_PARAM_ERROR`). Dies gilt auch für `HeadNo` = 2, falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist (Rückgabewert 6). • **auto_cal** wird nicht ausgeführt (Rückgabewert 3, **get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **auto_cal** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Bei einem `RTC6_PARAM_ERROR` wird der BUSY-Listenausführungsstatus nicht überprüft, d.h. Returncode `RTC6_BUSY` und `RTC6_PARAM_ERROR` kommen nicht gleichzeitig vor. • Für `Command` = 0, 1 oder 2 muss eine gültige Korrekturtabelle geladen und zugewiesen sein. Es kann sonst zu unvorhersehbaren Sprüngen kommen, wenn neue Gain/Offet-Werte gesetzt werden. • Die Gain- und Offset-Korrekturen können auch direkt über **set_hi** eingestellt werden (auch für Systeme *ohne* Home-In-Sensoren). • Die mit **auto_cal**(`Command` = 0, 1 oder 3) ermittelten und gespeicherten Home-In-Positionen können mit **get_hi_pos** ausgelesen werden. • Eine Überprüfung der ASC-Hardware erfolgt nicht nur mit **auto_cal**(`Command` = 4), sondern wird auch automatisch durchgeführt – bei **auto_cal**(`Command` = 0) und – beim erstmaligen Aufruf von **auto_cal**(`Command` = 1) und **auto_cal**(`Command` = 3), falls bis dahin weder **auto_cal**(`Command` = 0) noch **auto_cal**(`Command` = 4) ausgeführt wurde. In allen Fällen wird der ermittelte ASC-Hardware-Typ im Flash-Speicher gespeichert und kann nachträglich mit **get_auto_cal** ausgelesen werden. Bei automatischer `Command` = 4-Ausführung werden jedoch keine entsprechenden Rückgabewerte ausgegeben, sondern nur diejenigen des primären `Command`-Aufrufs (siehe oben). Ist ein Fehler aufgetreten, wird ein entsprechender Fehlercode im Flash-Speicher gespeichert (d.h. **get_auto_cal** liefert nicht 100 oder 200). Sobald die Hardware wieder funktioniert, kann der Fehler durch expliziten Aufruf von **auto_cal** (`Command = 0`) oder **auto_cal** (`Command = 4`) aus dem Flash-Speicher gelöscht werden. **auto_cal** (`Command = 1`) und **auto_cal** (`Command = 3`) können nicht ausgeführt werden, solange der Fehler noch im Flash-Speicher steht. Der Fehler wird durch diese Aufrufe aus dem Flash-Speicher *nicht* gelöscht.

Kontrollbefehl	auto_cal
RTC4→RTC6	Die Funktionen zur automatischen Selbstkalibrierung (`Command` = 0…2) sind (größtenteils) unverändert. Neu: `Command` = 3 und `Command` = 4.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_hi**, **get_hi_pos**, **get_auto_cal**, **write_hi_pos**

Kontrollbefehl	auto_change
Funktion	Aktiviert einen einmaligen automatischen Listenwechsel.
Aufruf	`auto_change()`
Multi-Board-Befehlsname	**n_auto_change**
Hinweise	• **auto_change** ist gleichbedeutend mit `auto_change_pos( 0 )`. Damit wird die nachfolgende Liste an ihrem Listenanfang gestartet. • Siehe auch Hinweise unter **auto_change_pos**.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**auto_change_pos**, **get_status**, **read_status**

Kontrollbefehl	**auto_change_pos**
Funktion	Aktiviert einen einmaligen, automatischen Listenwechsel und legt gleichzeitig die Listenposition fest, an der die Ausführung fortgesetzt werden soll.
Aufruf	`auto_change_pos( Pos )`
Parameter	`Pos` Startposition (RTC6-Listenspeicheradresse) als Offset bezüglich des Listenanfangs der durch den automatischen Listenwechsel zu startenden Liste. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_auto_change_pos**
Hinweise	• **auto_change_pos** löst nur einen *einmaligen* nachfolgenden Listenwechsel oder Listenneustart aus. Für weitere Listenwechsel oder Listenneustarts muss **auto_change_pos** erneut aufgerufen werden. • **auto_change_pos** kann zu jedem beliebigen Zeitpunkt aufgerufen werden. • Wird der automatische Listenwechsel während der Abarbeitung einer Liste aktiviert, fährt die Ausführung bei Erreichen von **set_end_of_list** ohne Verzögerung mit der jeweils anderen Liste an der angegebenen Startposition fort. Gibt es nur *eine* Liste (`Mem2` = 0, siehe **config_list**), wird die Ausführung der Liste bei Erreichen von **set_end_of_list** an der angegebenen Startposition fortgesetzt. • Während eine Liste abgearbeitet wird, kann die jeweils andere (aber auch dieselbe) Liste neu geladen werden, siehe Kapitel 6.4.6 "Listen automatisch wechseln", Seite 113. • Damit **auto_change_pos** überhaupt wirkt, muss die gegebenenfalls gerade aktive Liste unbedingt mit **set_end_of_list** abgeschlossen worden sein, die neue Liste sollte bereits geladen sein oder der Eingabe-Zeiger einen gewissen Vorsprung vor dem Ausgabe-Zeiger haben (anderenfalls werden "alte" Befehle ausgeführt). Wird bei der Listenausführung das Ende der Liste erreicht, ohne dass ein **set_end_of_list** angetroffen wird, so wird automatisch mit dem Anfang der aktuellen Liste fortgefahren. • Wird der automatische Listenwechsel aktiviert, wenn momentan keine Liste abgearbeitet wird, so wird überprüft, ob bereits eine Liste abgearbeitet wurde und gegebenenfalls die jeweils andere Liste (an der angegebenen Startposition) gestartet. Ist zuvor noch keine Liste ausgeführt worden, gilt "Liste 1" als bereits ausgeführt (Initialisierung) und "Liste 2" wird gestartet. • Wird eine RTC6-Listenspeicheradresse außerhalb des jeweiligen Listen-Bereichs angegeben (je nachdem welche Liste gestartet werden soll: `Pos` ≥ `Mem1` oder `Pos` ≥ `Mem2`), so wird die Startposition auf den Listenanfang (`Pos` = 0) gesetzt. • Falls während der Abarbeitung einer Liste **auto_change_pos**(`Pos` > 0) *und* **start_loop** aufgerufen werden, so wird beim unmittelbar nächsten **set_end_of_list** der Befehl **auto_change_pos**(`Pos` > 0) ausgeführt, bei den nachfolgenden der **start_loop**-Befehl.

Kontrollbefehl	auto_change_pos
Hinweise (Forts.)	• Die aktuellen Listenstatuswerte können mit **read_status** gelesen werden. • Die aktuellen Listenausführungsstatuswerte können mit **get_status** gelesen werden. • **auto_change_pos** löst einen Flush der gepufferten Listeneingabe aus, siehe Kapitel 6.4.1 "Listen laden", Seite 108.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Die RTC6-Listenspeicher-Adresse (`Pos`) wird bei der RTC6 als relative Speicheradresse bezüglich des Listenanfangs der jeweiligen Liste, bei der RTC4 dagegen als absolute Speicheradresse (0…7999) angegeben. Wird mit **config_list** der "Liste 2" kein RTC6-Listenspeicher-Bereich zugewiesen (`Mem2` = 0), wirkt der RTC6-Befehl wie bei der RTC4.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**auto_change**, **get_status**, **read_status**

Kontrollbefehl	bounce_supp
Funktion	Entprellt das externe Startsignal.
Aufruf	`bounce_supp( Length )`
Parameter	`Length` Entprellungszeit. In ms. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023]. Die 22 höherwertigeren Bits werden ignoriert.
Multi-Board-Befehlsname	**n_bounce_supp**
Hinweise	• Mit **bounce_supp** kann eine Entprellung für die über die Eingänge /START, /START2 oder /Slave-START eingespeisten Startsignale eingestellt werden, siehe Abschnitt "Externer Start", Seite 315. Startsignale, die innerhalb der eingestellten Entprellungszeit nach einem erfolgreichen Startsignal erfolgen, werden dadurch unterdrückt. • Vorgehensweise: Starten Sie eine Liste, die länger als eine Sekunde läuft. Wenn /START, /START2 oder /Slave-START prellt, wird ein zusätzliches Triggerfehlersignal erzeugt, das über **get_marking_info** (Bit #8) detektiert werden kann. Erhöhen Sie die Entprellungszeit so lange, bis dieses zusätzliche Signal nicht mehr detektiert wird. • Als Defaultwert ist eine Entprellungszeit von 0 ms eingestellt. • **bounce_supp** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_startstop_info**

Normaler Listenbefehl	**camming**
Funktion	Aktiviert die Camming-Funktionalität.
Aufruf	`camming ( FirstPos, NPos, EncoderNo, Ctrl, Scale, Code )`
Parameter	`FirstPos` Anfangsadresse der Camming-Befehlsliste (absolute Adresse im RTC6-Listenspeicher). Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [0…(2^{23}–1)].
	`NPos` Länge der Camming-Befehlsliste (ohne abschließendes **set_end_of_list** oder **list_return**). Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [1…(2^{23}–`FirstPos`)].
	`EncoderNo` Nummer des Encoderzählers, dessen Pulse zum Steuern des Camming-Prozesses ausgewertet werden,. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoderzähler “Encoder0”. = 1: Encoderzähler “Encoder1”. Höherwertige Bits werden ignoriert.
	`Ctrl` Camming-Steuerungsmodus. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: RTC6-Karte steuert Laser wie bei einem [*]mark[*]-Befehl. **camming** endet automatisch. = 1: Benutzer steuert Laser. **camming** endet automatisch. = 2: Benutzer steuert Laser. **camming** endet *nicht* automatisch. Die Camming-Befehlsliste wird bis zum Endpunkt (0 oder `NPos`–1) ausgeführt und wartet dann auf einen neuen /START. = 3: Benutzer steuert Laser. **camming** endet *nicht* automatisch. Die Camming-Befehlsliste wird kontinuierlich umlaufend (Ausgabe-Index modulo `NPos`) abgearbeitet.
	`Scale` Übersetzung (Umrechnungsfaktor) zwischen Encoderzählwert und Befehls-Index. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässige Werte: $2^{-60} < \lvert$`Scale`$\rvert < 2^{60}$.
	`Code` Wird nicht verwendet. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_camming**

Normaler Listenbefehl	**camming**
Hinweise	• Siehe auch Kapitel 8.11 "Camming", Seite 291. • Bei unzulässigen Parameterangaben wird **camming** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Bei jedem Aufruf von **camming** wird der aktuelle Encoderzählwert als neuer Referenzwert ermittelt. Zu einem späteren Zeitpunkt wird für den dann aktuellen Encoderzählwert (unter Verwendung des Referenzwerts und des Parameters `Scale`) der zugehörige Befehls-Index der Camming-Befehlsliste berechnet. Bei jedem Aufruf von **camming** wird also immer zuerst der erste Befehl in der Camming-Befehlsliste (Index = 0) abgearbeitet. • **camming** wartet ein Scanner-Delay ab, setzt aber selbst kein Delay. • Für `Ctrl` = 0 wird der Laser (wie bei einem normalen [*]mark[*]-Befehl) zu Beginn des Camming-Prozesses eingeschaltet und beim Beenden ausgeschaltet (eingestellte Laser-Delays werden berücksichtigt). Für `Ctrl` > 0 wird der Zustand des Lasers nicht verändert; dessen Kontrolle liegt dann vollständig in der Verantwortung des Benutzers. • Für `Ctrl` = 0 oder 1 endet **camming** automatisch (im nächsten Zyklus), sobald der Index erstmalig 0 unter- oder `NPos`–1 überschritten hat (als letzter Camming-Befehl wird dann der Befehl mit Index 0 oder `NPos`–1 ausgeführt). In diesen beiden Modi sind (wie immer, wenn eine Liste BUSY-Listenausführungsstatus ist) keine externen Starts erlaubt, solange **camming** noch nicht beendet ist. • Die Steuerungsmodi `Ctrl` = 2 und 3 sind dagegen endlos. Der Camming-Prozess kann hier nur durch **stop_execution** oder durch einen externen Stopp beendet werden. In diesen Modi ist aber (ausnahmsweise) ein Externer Start auch dann erlaubt, wenn die Liste (oder **camming**) noch aktiv ist. Für `Ctrl` = 2 wird der Index bis zum Endpunkt (0 oder `NPos`–1) ausgeführt und wartet dann auf einen neuen /START. Für `Ctrl` = 3 wird der Index modulo `NPos` gesetzt (dabei wird vorausgesetzt, dass die Encodergeschwindigkeit nicht so groß ist, dass mehr als ein kompletter Umlauf übersprungen wird). `Ctrl` = 3 wirkt also wie ein Ringpuffer. • **camming** ist ein Normaler Listenbefehl, aber variabel in der Ausführungsdauer. • **camming** funktioniert auch dann, wenn die Option "Processing on the fly" nicht freigeschaltet ist. • Während der Ausführung von **camming** liefert **get_out_pointer** immer die Position von **camming**, nicht diejenige des aktuellen Indexes.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_encoder_speed**, **set_auto_laser_control**

Unverzögerter Kurzer Listenbefehl	**clear_fly_overflow**
Funktion	Setzt die angegebenen Fehler-Bits (von **get_marking_info**) für die kundenspezifische Überwachung von "Processing on the fly"-Anwendungen zurück.
Aufruf	`clear_fly_overflow( Mode )`
Parameter	`Mode` Die Fehler-Bits, die zurückgesetzt werden sollen. Als 32-Bit-Wert ohne Vorzeichen. Bit #0 = 1: Fehler-Bit Bit #4 (Underflow X). Bit #1 = 1: Fehler-Bit Bit #5 (Overflow X). Bit #2 = 1: Fehler-Bit Bit #6 (Underflow Y). Bit #3 = 1: Fehler-Bit Bit #7 (Overflow Y). Bit #4 = 1: Fehler-Bit Bit #24 (Underflow Z). Bit #5 = 1: Fehler-Bit Bit #25 (Overflow Z). `Bit #0...Bit #5` können beliebig kombiniert werden. Höherwertige Bits werden ignoriert.
Multi-Board-Befehlsname	**n_clear_fly_overflow**
Hinweise	• Zur Verwendung von **clear_fly_overflow** siehe Abschnitt "Kundenspezifischer Überwachungsbereich", Seite 268. • Alle 6 Fehler-Bits werden zurückgesetzt für: – `Mode` = 0 – `Mode` = 63
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**, **clear_fly_overflow_ctrl**

Kontrollbefehl	**clear_fly_overflow_ctrl**
Funktion	Wie **clear_fly_overflow**, aber ein Kontrollbefehl.
Aufruf	`clear_fly_overflow_ctrl( Mode )`
Parameter	`Mode` Wie **clear_fly_overflow**.
Multi-Board-Befehlsname	**n_clear_fly_overflow_ctrl**
Hinweise	• Siehe **clear_fly_overflow**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 628, OUT 629.
Verweise	**clear_fly_overflow**

Unverzögerter Kurzer Listenbefehl	**clear_io_cond_list**
Funktion	Löscht die Bits des 16-Bit-Digital-Ausgangs der EXTENSION 1-Stiftleiste, die im Parameter `MaskClear` gesetzt sind, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind).
Aufruf	`clear_io_cond_list( Mask1, Mask0, MaskClear )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
	`MaskClear` Siehe `Mask1`.
Multi-Board-Befehlsname	**n_clear_io_cond_list**
Hinweise	• **clear_io_cond_list** löscht nur diejenigen Bits des Digital-Ausgangs, die im Parameter `MaskClear` gesetzt sind und lässt die anderen Bits unverändert. • Siehe auch Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81 und Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
Beispiele (Pascal)	• Lösche Bit #4 des Digital-Ausgangs (DIGITAL OUT4), falls Bit #0 des Digital-Eingangs (DIGITAL IN0) gesetzt und die Bit #1 bis Bit #3 (DIGITAL IN1…3) des Digital-Eingangs nicht gesetzt sind: `clear_io_cond_list`($0001, $000E, $0010) • Lösche immer Bit #15 des Digital-Ausgangs (und lasse die anderen Bits unverändert): `clear_io_cond_list`(0, 0, $8000)
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_io_cond_list**, **write_io_port**, **write_io_port_mask**, **get_io_status**, **read_io_port**

Kontrollbefehl	**config_laser_signals**
Funktion	Konfiguriert die Laserausgabesignal-Typen, die an Pin (01) (LASER1), Pin (02) (LASERON) und Pin (09) (LASER2) der LASER-Buchse ausgegeben werden.
Aufruf	`config_laser_signals( Config )`
Parameter	`Config` Gewünschte Signalkonfiguration. Als 32-Bit-Wert ohne Vorzeichen. Es konfigurieren: Bit #0…Bit #1: Pin (02) (LASERON-Kanal) Bit #2…Bit #3: Pin (01) (LASER1-Kanal) Bit #4…Bit #5: Pin (09) (LASER2-Kanal) Bit #6: Wird ignoriert … Bit 31: Wird ignoriert Es bedeuten: = 0 = 00_b: LASERON-Signal = 1 = 01_b: LASER1-Signal = 2 = 10_b: LASER2-Signal = 3 = 11_b: FirstPulseKiller-Signal (relevant für YAG-Modus, Laser-Mode 4; siehe Abschnitt "FirstPulseKiller-Signal", Seite 200 und **set_firstpulse_killer**) Die Standard-Einstellung nach **load_program_file** ist: `Config` = 100100_b = 0x24 = 36.
Multi-Board-Befehlsname	**n_config_laser_signals**
Hinweise	• Die angegebene Konfiguration wird wirksam, sobald der Laser das nächste Mal eingeschaltet wird. **config_laser_signals** sollte also nicht während einer aktiven Markierung aufgerufen werden. • Siehe auch **set_laser_control**, Bit #3 und Bit #4. • **config_laser_signals** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
Beispiele	• `Config` = 000111_b: FirstPulseKiller-Signal an LASERON-Kanal, LASER1-Signal an LASER1-Kanal, LASERON-Signal an LASER2-Kanal. In dieser Konfiguration können die LASER1-Signale, die synchron mit dem LASERON-Signal generiert werden, sofort ausgegeben und der Laser selbst dagegen erst um das FirstPulseKiller-Signal verzögert eingeschaltet werden. • `Config` = 100100_b (Standardeinstellung): LASERON-Signal an LASERON-Kanal, LASER1-Signal an LASER1-Kanal, LASER2-Signal an LASER2-Kanal. In dieser Konfiguration können die LASER1-Signale gegenüber dem Laser nur verzögert eingeschaltet werden, nicht schon vorher (hier ist das LASERON-Signal das Laser-Startsignal; da LASER1-Signale von der RTC6 nur dann erzeugt werden, wenn das LASERON-Signal an ist, werden hier LASER1-Signale erst frühestens mit dem Laser-Start ausgegeben).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**config_laser_signals_list**, **set_laser_control**

Verzögerter Kurzer Listenbefehl	**config_laser_signals_list**
Funktion	Wie **config_laser_signals**, aber ein Listenbefehl.
Aufruf	`config_laser_signals_list( Config )`
Parameter	`Config` Wie **config_laser_signals**.
Multi-Board-Befehlsname	**n_config_laser_signals_list**
Hinweise	• Siehe **config_laser_signals**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**config_laser_signals**

Kontrollbefehl	config_list
Funktion	Konfiguriert den RTC6-Listenspeicher, d.h. weist den RTC6-Listenspeicher-Bereichen bestimmte Speicherplätze zu.
Aufruf	`config_list( Mem1, Mem2 )`
Parameter	`Mem1` Speicherplätze für den RTC6-Listenspeicher-Bereich “Liste 1”. Als 32-Bit-Wert ohne Vorzeichen.
	`Mem2` Speicherplätze für den RTC6-Listenspeicher-Bereich “Liste 2”. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_config_list**
Hinweise	• Der RTC6-Listenspeicher bietet insgesamt 8.388.608 (= 2^{23}) Speicherplätze. Sie können mit **config_list** auf drei Bereiche (“Listen”) aufgeteilt werden. Die Größen von “Liste 1” und “Liste 2” werden mit den Parametern `Mem1` und `Mem2` vorgegeben. Die restlichen, nicht an “Liste 1” oder “Liste 2” vergebenen, Speicherplätze teilt **config_list** automatisch dem geschützten RTC6-Listenspeicher-Bereich “Liste 3” zu. • Für die beiden Parameter `Mem1` und `Mem2` gelten die folgenden Regeln (unzulässige Werte werden automatisch in der angegebenen Reihenfolge korrigiert): – `Mem1` > 0 (“Liste 1” darf nicht leer sein). `Mem1` = 0 wird zu `Mem1` = 1 korrigiert. – `Mem1` ≤ 2^{23} (“Liste 1” kann max. 2^{23} Speicherplätze umfassen). `Mem1` > 2^{23} wird zu `Mem1` = 2^{23} korrigiert. – `Mem1` = ”–1” wird als `Mem1` = (2^{32}–1) interpretiert und zu `Mem1` = 2^{23} korrigiert. Beispiel: Mit `config_list( -1, x )` und `x` beliebig (auch für `x` = –1) wird “Liste 1” automatisch der gesamte RTC6-Listenspeicher zugeordnet (`Mem1` = 2^{23}, `Mem2` = 0, keine Speicherplätze für “Liste 3”). – `Mem2` ≤ 2^{23} – `Mem1` (“Liste 2” darf maximal den “Rest” einnehmen). `Mem2` = 0 ist erlaubt. `Mem2` > 2^{23} – `Mem1` wird zu `Mem2` = 2^{23} – `Mem1` korrigiert. – `Mem2` = ”–1” wird als `Mem2` = (2^{32}–1) interpretiert und zu `Mem2` = 2^{23} – `Mem1` korrigiert. Beispiel: Mit `config_list( Mem1, -1 )`, wird “Liste 2” automatisch der “Rest” des RTC6-Listenspeichers zugeordnet (`Mem2` = 2^{23} – `Mem1`, keine Speicherplätze für “Liste 3”). – Speicherplätze für “Liste 3”: 2^{23} – `Mem1` – `Mem2`. • Der RTC6-Listenspeicher bietet insgesamt 8.388.608 (= 2^{23}) Speicherplätze. Er ist standardmäßig so vorkonfiguriert, dass “Liste 1” und “Liste 2” jeweils 4.194.304 (= 2^{22}) Listenbefehle aufnehmen können (`Mem1` = `Mem2` = 4.194.304). Der geschützte RTC6-Listenspeicher-Bereich “Liste 3” umfasst keine Speicherplätze, weil 2^{23} – `Mem1` – `Mem2` = 0. • **config_list** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt)

Kontrollbefehl	config_list
Hinweise (Forts.)	• Bei der Konfiguration mit **config_list** erfolgt keine Änderung des Speicherinhalts, ein wiederholter Aufruf mit verschiedenen Parametern ist insofern unschädlich. Zuvor geladene Listenbefehle werden allerdings nach einer Konfigurationsänderung gemäß der neuen Konfiguration abgearbeitet. Außerdem wirkt sich eine Konfigurationsänderung u.U. auf den Eingabe-Zeiger (liegt dieser vor der Konfiguration auf einer Speicherposition, die bei der Konfiguration der “Liste 3” zugewiesen wird, so wird der Eingabe-Zeiger auf den Anfang der “Liste 1” versetzt) oder auf einen zuvor gestarteten automatischen Listenwechsel aus. Dies sollte beim weiteren Laden oder Ausführen von Befehlslisten berücksichtigt werden. Beachten Sie auch die Hinweise im Kapitel 6.3.2 ”RTC6-Listenspeicher konfigurieren”, Seite 106. • Sind die aktuellen Konfigurationsdaten des RTC6-Listenspeichers (`Mem1` und `Mem2`) nicht bekannt, können sie bei Bedarf nach `load_list( ListNo, 0 )` oder `set_start_list_pos( ListNo, 0 )` mit **get_list_space** ermittelt werden (bei einem “Besitzerwechsel” muss zuvor **get_config_list** aufgerufen werden). • **config_list** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 ”Automatisches Booten – Ablauf im Detail”, Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: erhöhter Wertebereich und geänderte Initialisierungswerte.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_config_list**

Kontrollbefehl	control_command	
Funktion	Setzt einen Steuerbefehl ab. Ob und wie das angesprochene Scan-System reagiert, hängt von dessen Eigenschaften ab (unter anderem der Scan-System-Firmware). Siehe auch Hinweise, Seite 378.	
Aufruf	`control_command( Head, Axis, Data )`	
Parameter	`Head`	Nummer des Scan-Kopf-Anschlusses der RTC-Karte. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf.
	`Axis`	Nummer der Achse. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: x-Achse (STATUS-Kanal, Galvanometerscanner 2). = 2: y-Achse (STATUS1-Kanal, Galvanometerscanner 1).
	`Data`	Befehlscode mit optionalem Parameter. Siehe Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083. Als 32-Bit-Wert ohne Vorzeichen. Die obere Hälfte (Bit #16…Bit #31) wird *nicht* ausgewertet. Die untere Hälfte (Bit #0…Bit #15) wird wie folgt ausgewertet: • $Code_{HIGH}$ Das höherwertigere Daten-Byte (= Bit #8…Bit 15). Gibt einen Befehlscode an. In Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083 als Hexadezimalzahl "(hex)" angegeben. • $Code_{LOW}$ Das niederwertigere Daten-Byte (= Bit #0…Bit #7). Gibt einen optionalen Parameter an. In Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083 als Hexadezimalzahl "(hex)" angegeben. Beispiel: Mit `Data` = 0501_H, d. h. $Code_{HIGH} = 05_H$ (**SetMode**) und $Code_{LOW} = 01_H$ wird die Ist-Position als zu übertragender Datentyp eingestellt.
Multi-Board-Befehlsname	**n_control_command**	

Kontrollbefehl	control_command
Hinweise	• **control_command** kann bei iDRIVE-Scan-Systemen verwendet werden. • **control_command** wird von Scan-Systemen ohne i*DRIVE*-Technologie nicht ausgewertet. • **control_command** wird für unzulässige Werte von `Head` und/oder `Axis` nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Der Befehlscode `Data` wird anstelle der sonst üblichen Positionsdaten an das Scan-System ausgegeben. Wird **control_command** während der Abarbeitung einer Liste aufgerufen, fällt der entsprechende Galvanometerscanner-Mikroschritt daher aus. • **control_command** wird u.U. durch Aktivierung der Geschwindigkeits-abhängige Laser-Steuerung mit `set_auto_laser_control( Mode = 2 )` für den Anschluss für ersten Scan-Kopf gesperrt. • **control_command** ist in Boot-Phase 2 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Unveränderte Funktionalität. Beachten Sie, dass alle mit **control_command** zur Übertragung ausgewählten Daten immer im 20-Bit-Wertebereich liegen, auch im RTC4-Kompatibilitätsmodus (siehe auch Hinweise bei **get_value**).
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_value**, **get_values**, **get_head_status**, **set_trigger**, **set_trigger4**, **set_trigger8**, **get_waveform**

Kontrollbefehl	copy_dst_src	
Funktion	Erstellt in der internen Verwaltungstabelle für ein indiziertes Zeichen, eine indizierte Zeichenfolge oder ein indiziertes Unterprogramm mit dem angegebenen Index (`Dst`) Tabelleneinträge durch Kopieren der Tabelleneinträge zu einem anderen Index (`Src`).	
Aufruf	`copy_dst_src( Dst, Src, Mode )`	
Parameter	`Dst`	Index des indizierten Zeichens, der indizierten Zeichenfolge oder des indizierten Unterprogramms, dessen Einträge aus `Src` kopiert werden sollen. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023] für indizierte Zeichen oder Unterprogramme, [1024+0…1024+41] für indizierte Zeichenfolgen.
	`Src`	Index des indizierten Zeichens, der indizierten Zeichenfolge oder des indizierten Unterprogramms, dessen Einträge nach `Dst` kopiert werden sollen. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023] für indizierte Zeichen oder Unterprogramme, [1024+0…1024+41] für indizierte Zeichenfolgen.
	`Mode`	Legt fest, welche Verwaltungstabellen geändert werden sollen. Als 32-Bit-Wert ohne Vorzeichen. Bit #0 = 0: `Dst` ist der Index eines indizierten Zeichens oder der Index einer indizierten Zeichenfolge. Bit #0 = 1: `Dst` ist der Index eines indizierten Unterprogramms. Bit #1 = 0: `Src` ist der Index eines indizierten Zeichens oder einer indizierten Zeichenfolge. Bit #1 = 1: `Src` ist der Index eines indizierten Unterprogramms. Bit #2…Bit #31: Nicht berücksichtigt.
Multi-Board-Befehlsname	**n_copy_dst_src**	
Hinweise	• Bei ungültigen Indexwerten (`Dst` und/oder `Src`) wird **copy_dst_src** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Mit **copy_dst_src** wird ein indiziertes Zeichen, eine indizierte Zeichenfolge oder ein indiziertes Unterprogramm ein weiteres Mal referenziert (kann also dann mit einem weiteren Index aufgerufen werden). **copy_dst_src** ändert nur die entsprechenden Einträge in der internen Verwaltungstabelle und lässt den Inhalt des RTC6-Listenspeichers unverändert. Damit können indizierte Zeichen, Zeichenfolgen und Unterprogramme dupliziert, umnummeriert oder ineinander konvertiert werden, ohne dass sie jeweils neu geladen werden müssen. • Ein echtes Duplikat eines indizierten Zeichens, einer indizierten Zeichenfolge oder eines indizierten Unterprogramms im geschützten RTC6-Listenspeicher-Bereich “Liste 3” des RTC6-Listenspeichers kann – nachfolgend auf **copy_dst_src** – mit **save_disk**/**load_disk** erstellt werden. Mehrfach referenzierte Zeichen, Zeichenfolgen und/oder Unterprogramme werden dadurch entsprechend oft in den RTC6-Listenspeicher geschrieben. Achten Sie darauf, dass es dabei zu keinem Speicherüberlauf im geschützten RTC6-Listenspeicher-Bereich “Liste 3” kommt. • Siehe auch Abschnitt “Indizierte Zeichen und Zeichenfolgen verwalten”, Seite 123.	

Kontrollbefehl	copy_dst_src
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**save_disk**, **load_disk**

Kontrollbefehl	**create_dat_file**
Funktion	Generiert eine neue `RTC6DAT.dat`-Datei der aktuellen Version.
Aufruf	`Version = create_dat_file ( Flag )`
Parameter	`Flag` Als 32-Bit-Wert mit Vorzeichen. 1: Generiert neue `RTC6DAT.dat`-Datei und liefert die aktuelle Version zurück. –1: Gibt nur die aktuelle Version zurück.
Rückgabe	`Version` Als 32-Bit-Wert ohne Vorzeichen. > 600: Aktuelle Version. 0: **create_dat_file** kann nicht ausgeführt werden (**get_last_error**-Returncode `RTC6_BUSY` oder auf der RTC6 ist kein Programm geladen). 1: Nicht genügend Windows-Speicher (**get_last_error**-Returncode `RTC6_OUT_OF_MEMORY`). 2: Die Ausgabedatei kann nicht geöffnet werden (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). Stellen Sie sicher, dass Sie Schreibrechte für den Ausgabepfad mit der `RTC6DAT.dat` besitzen.
Multi-Board-Befehlsname	**n_create_dat_file**
Hinweise	• Frühere `RTC6DAT.dat`-Versionen können nicht mehr verwendet werden. Anderenfalls liefert **load_program_file** den Fehlercode 7 zurück (**get_last_error**-Returncode `RTC6_VERSION_MISMATCH`). • Die `RTC6DAT.dat`-Versionsinformation steht nur zur Verfügung, nachdem mit der aktuellen RTC6-DLL-Instanz auch **load_program_file** ausgeführt wurde. • Die `RTC6DAT.dat`-Versionsinformation wird beim in Besitz nehmen einer RTC6-Karte nicht mit übernommen (egal ob diese bereits initialisiert ist oder nicht). • Ein neu generiertes `RTC6DAT.dat` (einer bestimmten Version) kann jederzeit für alle RTC6-Karten verwendet werden, die dieselbe Version benötigen. • Jedes neu generierte `RTC6DAT.dat` enthält (neben statischen Initialisierungen) auch: – benutzerdefinierbare Tabellen, die zuvor mit **load_varpolydelay**, **load_auto_laser_control**, **load_jump_table**, **load_jump_table_offset**, **load_position_control** geladen worden sind – eine “Frei definierbare Wobbelfigur”, die zuvor mit **set_wobbel_vector** festgelegt wurde Die benutzerdefinierbaren Tabellen stehen nach dem nächsten **load_program_file** automatisch zur Verfügung, für die “Frei definierbare Wobbelfigur” muss zusätzlich noch `set_wobbel_mode( Mode > 1 )` aufgerufen werden. • Beachten Sie, dass **create_dat_file** eine bereits vorhandene `RTC6DAT.dat` ohne Warnung überschreibt. Bewahren Sie eine Kopie Ihrer originalen `RTC6DAT.dat` an einem sicheren Ort auf. • **create_dat_file** kann nicht ausgeführt werden, wenn gerade eine Liste abgearbeitet wird (**get_last_error**-Returncode `RTC6_BUSY`).

Kontrollbefehl	create_dat_file
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 610, OUT 610, RBF 615. Letzte Änderung DAT 603.
Verweise	**load_program_file**

Kontrollbefehl	demux_rcm_ctrl	
Funktion	Relevant für Rückkanal-Multiplexing: Demultiplex von Daten, die bei aktivem Rückkanal-Multiplexing aufgezeichnet wurden.	
Aufruf	`demux_rcm_ctrl( Input, Input_length, N_signals, Index, Flags, Output )`	
Parameter	`Input`	Die von **get_waveform**/**get_waveform_offset** zurückgegebenen Daten. Zeiger auf einen Array von 32-Bit-Werten.
	`Input_length`	Anzahl der Datenpunkte in `Input` Als 32-Bit-Wert ohne Vorzeichen.
	`N_signals`	Die Rückkanal-Multiplexing-Blocklänge. Als 32-Bit-Wert ohne Vorzeichen.
	`Index`	Rückkanal-Multiplexing-Index, dessen Daten extrahiert werden sollen. Als 32-Bit-Wert ohne Vorzeichen.
	`Flags`	Demultiplex-Einstellung. Bit #1 = 1: Überspringt den ersten Eintrag in `Output`, falls der erste Rückkanal-Multiplexing-Block in `Input` unvollständig ist, und der gewünschte Index dort nicht enthalten ist. Bit #1 = 0: Überspringt den ersten Rückkanal-Multiplexing-Block in `Input`, falls dieser unvollständig ist. Bit #0 = 1: Überspringt den letzten Eintrag in `Output`, falls der letzte Rückkanal-Multiplexing-Block in `Input` unvollständig ist, und der gewünschte Index dort nicht enthalten ist. Bit #0 = 0: Überspringt den letzten Rückkanal-Multiplexing-Block in `Input`, falls dieser unvollständig ist. Als 32-Bit-Wert ohne Vorzeichen.
	`Output`	Zeiger auf einen Array von 32-Bit-Werten der Dimension n. n ergibt sich aus den anderen Parametern.
Rückgabe	• Wenn in `Flags` Bit #0 = 0 und Bit #1 = 0: Anzahl der vollständigen Rückkanal-Multiplexing-Blöcke. • Wenn in `Flags` Bit #0 = !0 oder Bit #1 != 0: Der auf `Output` zu addierende Offset für den nächsten Aufruf zum kontinuierlichen Demultiplexing.	
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.	

Kontrollbefehl	demux_rcm_ctrl
Hinweise	• **demux_rcm_ctrl** ist nicht als Multi-Board-Befehl verfügbar. • `Flags` ist für ein kontinuierliches Demultiplexing vorgesehen (Zum demultiplexen einer einmaligen Datenaufzeichnung rufen Sie `Flags` = 0 auf). Rückkanal-Multiplexing-Blöcke werden abgeschnitten, wenn sie (am Anfang oder Ende) unvollständig sind. • Um einen kontinuierlichen Datenstrom zu demultiplexen, der sich nicht mit **get_waveform**/**get_waveform_offset** als Ganzes auslesen lässt: – Beim 1. Aufruf setzen Sie Bit #0 = 1 und Bit #1 = 0 in `Flags`. Ist der 1. Rückkanal-Multiplexing-Block unvollständig, wird er verworfen. Ist am Ende der Daten ein unvollständiger Rückkanal-Multiplexing-Block, dann wird er – so weit wie möglich – demultiplext. Sie müssen den Rückgabewert speichern. – Bei allen nachfolgenden Aufrufen setzen Sie Bit #0 = 1 und Bit #1 = 1 in `Flags`. Versetzen Sie außerdem `Output` um dessen Rückgabewert des vorigen Aufrufs. Sollen keine weiteren Daten mehr demultiplext werden, bewirkt Bit #0 = 0, dass ein (möglicherweise unvollständiger) Rückkanal-Multiplexing-Block am Ende verworfen wird. • Beim Kopieren nach `Output` wird nur das Synchronisations-Bit entfernt. Danach gilt: – Bit #31 = 0…Bit #20 = 0 – Bit #19…Bit #0 enthalten die Nutzdaten • Der Rückgabe-Wert ist 0, wenn: – `N_signals` ist kein Teiler von 192 – In den ersten 192 Einträgen von `Input` wird kein Synchronisations-Bit gefunden • **demux_rcm_ctrl** greift nicht auf eine RTC6-Karte zu. Daher kann **demux_rcm_ctrl** auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte benutzt werden. Sind die Daten auf dem PC binär gespeichert, kann **demux_rcm_ctrl** auch Offline (= am PCIe-Bus ist keine RTC6-Karte vorhanden) benutzt werden. • Das Verhalten von **demux_rcm_ctrl** ist undefiniert, wenn die Daten nicht aus einer Aufzeichnung von Signal 63, 64, 65, 66 stammen.
Beispiel (C/C++)	–
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	**get_rcm_value_ctrl**

Kontrollbefehl	disable_laser
Funktion	Unterdrückt die Signale für den “Laser active”-Betrieb.
Aufruf	`disable_laser()`
Multi-Board-Befehlsname	**n_disable_laser**
Hinweise	• **disable_laser** unterdrückt die Signale für den “Laser active”-Betrieb an den Ausgängen LASER1, LASER2 und LASERON (die Signale werden auf ihren jeweiligen “Off”-Pegel gesetzt). Pin (02) der LASER-Buchse ist dann statisch. Mit **set_standby** oder **set_standby_list** aktivierte Standby-Signale werden aber weiterhin an den Ausgängen LASER1 und LASER2 ausgegeben. Bei deaktivierten Standby-Signalen sind nach **disable_laser** auch Pin (01) und Pin (09) der LASER-Buchse und Pin (19) und Pin (22) der EXTENSION 2-Stiftleiste statisch. • Die Signale für den “Laser active”-Betrieb können auch mit **set_laser_control** unterdrückt und mit **set_laser_control** oder **enable_laser** wieder freigegeben werden. • Nach einer Initialisierung der RTC6 mit **load_program_file** ist die Laser-Steuerung *deaktiviert* und benötigt unbedingt **set_laser_control** zur Aktivierung. • Wenn **disable_laser** einen `RTC6_TIMEOUT`-Fehler erzeugt (z.B. wenn noch keine Programmdatei geladen worden ist) können die Ausgänge LASER1, LASER2 und LASERON dann nur mit **set_laser_control** nach einem **load_program_file** reaktiviert werden. • Mit **get_startstop_info** (`Bit #9`) kann der aktuelle Status der Lasersteuersignale (global freigegeben, ja oder nein) ausgelesen werden. • Mit **get_startstop_info** (`Bit #14`) kann der Status “laser enabled” (ja oder nein) ausgelesen werden.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung: DLL 617, OUT 617: **get_startstop_info** (`Bit #14`).
Verweise	**enable_laser**, **set_laser_control**, **get_startstop_info**

Kontrollbefehl	enable_laser
Funktion	Gibt die Signale für den “Laser active”-Betrieb frei.
Aufruf	`enable_laser()`
Multi-Board-Befehlsname	**n_enable_laser**
Hinweise	• Nach einem Hardware-Reset (und nach **load_program_file**) muss zur Aktivierung der Ausgänge LASER1, LASER2 und LASERON und zum Festlegen der Signalpegel **set_laser_control** aufgerufen werden, siehe Kapitel 7.4 “Laser-Steuerung”, Seite 192. Sonst bleibt **enable_laser** wirkungslos. • Nach einer Initialisierung der RTC6 mit **load_program_file** sind die Lasersteuersignale *deaktiviert*. Zur erstmaligen Aktivierung muss **set_laser_control** aufgerufen werden (s.o.). Nach einer anschließenden Unterdrückung mit **disable_laser** (oder **set_laser_control**) kann **enable_laser** (oder **set_laser_control**) zur erneuten Freigabe verwendet werden. • Auch wenn die Lasersteuersignale mit **enable_laser** oder **set_laser_control** freigegeben wurden, werden sie ohne weitere Befehle noch nicht an den Ausgängen ausgegeben, siehe Kapitel 7.4 “Laser-Steuerung”, Seite 192. • Mit **get_startstop_info** (`Bit #9`) kann der aktuelle Status der Lasersteuersignale (global freigegeben, ja oder nein) ausgelesen werden. • Mit **get_startstop_info** (`Bit #14`) kann der Status “laser enabled” ausgelesen werden.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. U.U. muss **set_laser_control** vor **enable_laser** aufgerufen werden (s.o.).
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**disable_laser**, **set_laser_control**, **get_startstop_info**, **set_laser_mode**

Kontrollbefehl	eth_assign_card
Funktion	Trägt die RTC6 Ethernet-Karte mit dem Index `SearchNo` aus der Suchergebnisliste am RTC6-Kartenverwaltungsindex `CardNo` ein.
Aufruf	`Result = eth_assign_card( SearchNo, CardNo )`
Parameter	`SearchNo` Index der Suchergebnisliste. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…(Rückgabewert > 0 von **eth_found_cards**)]. `CardNo` Index der RTC6-Kartenverwaltung, bei dem die RTC6 Ethernet-Karte eingetragen werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [(**rtc6_count_cards** + 1)…255] oder 0.
Rückgabe	Fehlercode. Als 32-Bit-Wert mit Vorzeichen. –2 Fehler: Der Eintrag kann nicht vorgenommen werden. An diesem Index ist bereits eine RTC6 Ethernet-Karte eingetragen. –1 Fehler: Der Eintrag kann nicht vorgenommen werden. Dieser Index ist bereits mit einer RTC6 PCIe-Karte besetzt. 0 Fehler: Der Eintrag kann nicht vorgenommen werden. `SearchNo` ist ungültig (0, > **eth_found_cards**). Oder: `CardNo` ist ungültig (1 ≤ `CardNo` ≤ **rtc6_count_cards**, > 255). n (= `CardNo`) Erfolg: Der Kartennummern-Eintrag wurde vorgenommen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zum Erzeugen einer Suchergebnisliste werden **eth_search_cards** und **eth_search_cards_range** verwendet. Siehe auch Kapitel 16.5.2 "Über die Suche nach RTC6 Ethernet-Karten", Seite 1007. • Am angegebenen Index `CardNo` darf noch keine RTC6 Ethernet-Karte oder RTC6 PCIe-Karte eingetragen sein. Eine RTC6 Ethernet-Karte muss zuvor explizit mit **eth_remove_card** entfernt werden. RTC6 PCIe-Karten können nicht entfernt werden. • Im Erfolgsfall wird **eth_count_cards** automatisch um 1 erhöht. • Mit `CardNo` = 0 wird die RTC6 Ethernet-Karte an **rtc6_count_cards** + **eth_count_cards** + 1 eingetragen. Dadurch kann automatisch eine lückenlose Kartennummerierung erzeugt werden (siehe auch **eth_remove_card**). `CardNo` = 0 und freigewählte Zuweisung von RTC6 Ethernet-Karten zu Indices sollten nicht zusammen verwendet werden. • Bei allen Fehlern wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_assign_card_ip**, **eth_count_cards**, **eth_found_cards**, **eth_max_card**, **eth_search_cards**, **eth_search_cards_range**, **eth_remove_card**, **rtc6_count_cards**, **eth_set_com_timeouts_auto**

Kontrollbefehl	eth_assign_card_ip
Funktion	Trägt eine RTC6 Ethernet-Karte mit der IP-Adresse `Ip` am Index `CardNo` in die RTC6-Kartenverwaltung ein.
Aufruf	`Result = eth_assign_card_ip( Ip, CardNo )`
Parameter	`Ip` IP-Adresse der RTC6 Ethernet-Karte. Als 32-Bit-Wert ohne Vorzeichen. `CardNo` Index der RTC6-Kartenverwaltung, an dem die RTC6 Ethernet-Karte eingetragen werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [(**rtc6_count_cards** + 1)…255] oder 0.
Rückgabe	Fehlercode. Als 32-Bit-Wert mit Vorzeichen. –2 Fehler: Der Eintrag kann nicht vorgenommen werden. An diesem Index ist bereits eine RTC6 Ethernet-Karte eingetragen. –1 Fehler: Der Eintrag kann nicht vorgenommen werden. An diesem Index ist bereits eine RTC6 PCIe-Karte eingetragen. 0 Fehler: Der Eintrag kann nicht vorgenommen werden. `CardNo` ist ungültig (1 ≤ `CardNo` ≤ **rtc6_count_cards**, > 255). n (= `CardNo`) Erfolg: Der Eintrag wurde vorgenommen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Am angegebenen Index `CardNo` darf noch keine RTC6 Ethernet-Karte oder RTC6 PCIe-Karte eingetragen sein. Eine RTC6 Ethernet-Karte muss zuvor explizit mit **eth_remove_card** entfernt werden. RTC6 PCIe-Karten können nicht entfernt werden. • Im Erfolgsfall wird **eth_count_cards** automatisch um 1 erhöht. • Mit `CardNo` = 0 wird die RTC6 Ethernet-Karte an **rtc6_count_cards** + **eth_count_cards** + 1 eingetragen. Dadurch kann automatisch eine lückenlose Kartennummerierung erzeugt werden (siehe auch **eth_remove_card**). `CardNo` = 0 und freigewählte Zuweisung von RTC6 Ethernet-Karten zu Indices sollten nicht zusammen verwendet werden. • Bei allen Fehlern wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Die RTC6 Ethernet-Karte wird auch dann in die RTC6-Kartenverwaltung eingetragen, wenn bei deren Initialisierung ein Ethernet-Fehler aufgetreten ist. Vor einem erneuten Initialisierungsversuch muss diese RTC6 Ethernet-Karte wieder explizit aus der RTC6-Kartenverwaltung entfernt werden. • Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_assign_card**, **eth_count_cards**, **eth_remove_card**, **eth_set_com_timeouts_auto**

Kontrollbefehl	**eth_boot_dcmd**
Funktion	Standalone-Funktionalität: Definiert den nächsten folgenden Kontrollbefehl als einen Boot-Befehl für den Standalone-Betriebsmodus.
Voraussetzung	RTC6-Software-Paket ≥ V1.7.0 und BIOS-ETH ≥ 26.
Aufruf	`eth_boot_dcmd()`
Parameter	Keine.
Rückgabe	Keine.
Multi-Board-Befehlsname	**n_eth_boot_dcmd**
Hinweise	• Die Definition bezieht sich auf den nachfolgenden Kontrollbefehl, d.h. Listenbefehle dazwischen stören nicht. • Die Boot-Definition entfällt ersatzlos, wenn der definierte Kontrollbefehl zum Booten unzulässig ist, siehe Kapitel 16.8.4 "Für das automatische Booten zulässige Kontrollbefehle", Seite 1016. • **eth_boot_dcmd** ist nur bei RTC6 Ethernet-Karten erlaubt. Anderenfalls wird ein **get_last_error**-Returncode `RTC6_TYPE_REJECTED` erzeugt. • Siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 618, OUT 618, RBF 623.
Verweise	**eth_boot_timeout**

Kontrollbefehl	**eth_boot_timeout**
Funktion	Standalone-Funktionalität: Setzt die Wartezeit für den automatischen Übergang von Boot-Phase 1 nach Boot-Phase 2.
Voraussetzung	RTC6-Software-Paket ≥ V1.7.0 und BIOS-ETH ≥ 26.
Aufruf	`eth_boot_timeout( TimeOut )`
Parameter	`TimeOut` Wartezeit. In s. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Keine.
Multi-Board-Befehlsname	**n_eth_boot_timeout**
Hinweise	• **eth_boot_timeout** ist nur bei RTC6 Ethernet-Karten erlaubt. Anderenfalls wird ein **get_last_error**-Returncode `RTC6_TYPE_REJECTED` erzeugt. • `TimeOut` = 0 deaktiviert die Wartezeit. Der Boot-Vorgang wartet endlos auf einen /START. • Boot-Phase 2 beginnt sofort, wenn innerhalb der Wartezeit ein /START ausgelöst wird. • Siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012. • **eth_boot_timeout** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.

Kontrollbefehl	eth_boot_timeout
Versionsinfo	Verfügbar ab DLL 618, OUT 618, RBF 623.
Verweise	**eth_boot_dcmd**

Kontrollbefehl	**eth_check_connection**
Funktion	Prüft, ob die RTC6 Ethernet-Karte antwortet.
Aufruf	`CheckOK = eth_check_connection()`
Parameter	Keine.
Rückgabe	Antwortverhalten. Als 32-Bit-Wert ohne Vorzeichen. 0: Nicht OK. Die RTC6 Ethernet-Karte hat keine Ethernet-Verbindung, sie antwortet nicht oder es handelt sich nicht um eine RTC6 Ethernet-Karte. 1: OK. Die RTC6 Ethernet-Karte hat eine Ethernet-Verbindung und sie antwortet.
Multi-Board-Befehlsname	**n_eth_check_connection**
Hinweise	• Bei `CheckOK = 0` kann mit **get_last_error**, **eth_get_last_error** und **eth_get_error** die mögliche Fehlerursache abgefragt werden. • Wenn die Karte keine RTC6 Ethernet-Karte ist, dann wird: – **eth_check_connection** nicht ausgeführt – der **get_last_error**-Returncode `RTC6_TYPE_REJECTED` gesetzt • Siehe auch Kapitel 16.5.4 "Verbindung zur RTC6 Ethernet-Karte prüfen", Seite 1009.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 608, OUT 608, RBF 611.
Verweise	**eth_get_error**, **eth_get_last_error**, **get_last_error**

Kontrollbefehl	**eth_config_waveform_streaming_ctrl**	
Funktion	Relevant für Data-Streaming: Konfiguriert die Schnittstelle der RTC6 Ethernet-Karte für das Data-Streaming.	
Voraussetzung	RTC6-Software-Paket ≥ 1.16.3 und BIOS-ETH ≥ 36.	
Aufruf	`eth_config_waveform_streaming_ctrl( size, flags )`	
Parameter	`size`	Nutzdaten-Größe in einem einzelnen Data-Streaming-Datenpaket. In Bytes. Zulässige Werte: 256, 512, 1024, 2048, 4096, 8192, 16384, 32768. Als 32-Bit-Wert ohne Vorzeichen.
	`flags`	Bit #31: Reserviert. … Bit #01: Reserviert. Bit #0 = 0: Data-Streaming ist ausgeschaltet. Eine bestehende Verbindung von StreamParser-DLL mit der RTC6 Ethernet-Karte wird sofort getrennt. Bit #0 = 1: Data-Streaming ist eingeschaltet. Eine StreamParser-DLL kann eine Verbindung mit der RTC6 Ethernet-Karte aufbauen. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Keine.	
Multi-Board-Befehlsname	**n_eth_config_waveform_streaming_ctrl**	
Hinweise	• **eth_config_waveform_streaming_ctrl** wird bei einem unzulässigen `size`-Wert nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Standardmäßig sind 1024 Byte voreingestellt. • Mit jedem Data-Streaming-Datenpaket werden neue Metadaten geschickt. Ein kleiner `size`-Wert bedeutet: – Häufigere Aktualisierung der Metadaten – Höhere Netzwerklast • Siehe Handbuch "StreamParser-DLL – Schnittstelle zur Anwendungsprogrammierung". • **eth_config_waveform_streaming_ctrl** verwendet dieselbe Aufzeichnungsperiode wie **set_trigger[*]**. Die RTC6 Ethernet-Karte sendet mit jedem Data-Streaming-Datenpaket Metadaten: entsprechend ergibt sich die Metadaten-Rate aus Aufzeichnungsperiode und Paket-Größe. Die gesendeten Metadaten sind fix und können von Benutzern nicht konfiguriert werden. Die RTC6 Ethernet-Karte schickt die Data-Streaming-Datenpakete an diejenige IP-Adresse, von der aus sich die StreamParser-DLL verbindet. Die RTC6 Ethernet-Karte hört starr auf Port 63751. Die RTC6 Ethernet-Karte unterstützt nur Data-Streaming über TCP (das Protokoll wird automatisch erkannt).	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 635, OUT 636.	
Verweise	–	

Kontrollbefehl	**eth_configure_link_loss**
Funktion	Stellt das Verhalten der RTC6 Ethernet-Karte bei einem Ethernet Link Loss ein.
Voraussetzung	RTC6-Software-Paket ≥ V1.7.6 und BIOS-ETH ≥ 28.
Aufruf	`eth_configure_link_loss( Mode )`
Parameter	`Mode` Zu treffende Maßnahme. Als 32-Bit-Wert ohne Vorzeichen. 0: Die Signale für den "Laser active"-Betrieb werden sofort unterdrückt. Die Listenausführung läuft weiter. 1: Die Signale für den "Laser active"-Betrieb werden sofort ausgeschaltet. Die Listenausführung wird sofort beendet (wie **stop_execution** oder /STOP). 2: Ein **simulate_ext_stop** wird an alle Slave-Karten weitergeleitet. 3: Ethernet Link Loss-Erkennung ausgeschaltet. Default nach **load_program_file**.
Rückgabe	Keine.
Multi-Board-Befehlsname	**n_eth_configure_link_loss**
Hinweise	• Für `Mode` > 3 wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt und **eth_configure_link_loss** nicht ausgeführt. • Die RTC6 Ethernet-Karte erkennt einen Ethernet Link Loss nach spätestens 100 ms. • Ob ein Ethernet Link Loss erkannt wurde, kann mit **get_startstop_info** abgefragt werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 622, OUT 622.
Verweise	**get_startstop_info**

Kontrollbefehl	eth_convert_ip_to_string
Funktion	Wandelt eine IP-Adresse in Big-Endian Byte-Reihenfolge in die übliche Dezimalpunktschreibweise um (Beispiel: “192.168.250.1”).
Aufruf	`eth_convert_ip_to_string( Ip, IpString )`
Parameter	`Ip` Umzuwandelnde IP-Adresse in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen. `IpString` Umgewandelte IP-Adresse. Als Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von 4 32-Bit-Werten ohne Vorzeichen, in dem die umgewandelte IP-Adresse als Zeichenkette in üblicher Dezimalpunktschreibweise zu finden ist (gleichbedeutend mit einem Zeichen-Array der Länge 16, Low Byte first, mit abschließendem \0).
Rückgabe	Keine.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zum Speichern der umgewandelten IP-Adresse muss vom Anwenderprogramm an der über `IpString` angegebenen Adresse ein Speicherbereich der Größe 4 × 4 Byte bereitgestellt werden. • Beispiel: `Ip` = 33204416, `IpString`[0] = “192.”, `IpString`[1] = “168.”, `IpString`[2] = “250.”, `IpString`[3] = “1\0xx”. • Die Verwendung von **eth_convert_ip_to_string** bietet sich z.B. bei **eth_get_static_ip** an. • Siehe auch Kapitel 16.5.1 “Hinweise zum Arbeiten mit IP-Adressen”, Seite 1007.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_convert_string_to_ip**, **eth_get_static_ip**

Kontrollbefehl	eth_convert_string_to_ip
Funktion	Wandelt eine IP-Adresse in üblicher Dezimalpunktschreibweise (Beispiel: "192.168.250.1") in die Big-Endian Byte-Reihenfolge um.
Aufruf	`ResultIp = eth_convert_string_to_ip( &IpString )`
Parameter	`IpString` Umzuwandelnde IP-Adresse in üblicher Dezimalpunktschreibweise. Als String (char array). Als Zeiger auf einen nullterminierten ANSI-String. Max. 16 Bytes.
Rückgabe	IP-Adresse. Als 32-Bit-Wert ohne Vorzeichen. 0: `IpString` ist fehlerhaft. > 0: Umgewandelte IP-Adresse in Big-Endian Byte-Reihenfolge.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Beispiel: Beim Parameter `IpString` = "192.168.250.1" ist der Rückgabewert `ResultIp` = 33204416. Siehe auch Hinweis bei **eth_search_cards**. • Die Verwendung von **eth_convert_string_to_ip** bietet sich z.B. bei **eth_search_cards**, **eth_search_cards_range** und **eth_set_static_ip** an, sowie bei **eth_get_card_info** und **eth_get_card_info_search**. • Siehe auch Kapitel 16.5.1 "Hinweise zum Arbeiten mit IP-Adressen", Seite 1007.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_convert_ip_to_string**, **eth_get_card_info**, **eth_get_card_info_search**, **eth_search_cards**, **eth_search_cards_range**, **eth_set_static_ip**

Kontrollbefehl	eth_count_cards
Funktion	Gibt die Anzahl der RTC6 Ethernet-Karten zurück, die in der *RTC6-Kartenverwaltung* eingetragen sind.
Aufruf	`Result = eth_count_cards()`
Parameter	Keine.
Rückgabe	Anzahl der RTC6 Ethernet-Karten, die in der RTC6-Kartenverwaltung eingetragen sind. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Im Gegensatz zu **rtc6_count_cards** kann aus dem Rückgabewert nicht unbedingt auf den höchsten Index aller RTC6-Karten geschlossen werden (siehe **eth_assign_card** und **eth_remove_card** mit `CardNo` = 0, siehe auch **eth_max_card**). • Es ist nur sichergestellt, dass jede RTC6 Ethernet-Karten-Nummer größer ist als die größte RTC6 PCIe-Kartennummer, siehe **rtc6_count_cards**. • **eth_count_cards** kann durchaus größer als **eth_found_cards** sein. Das ist der Fall, wenn RTC6 Ethernet-Karten explizit über ihre IP-Adressen in die RTC6-Kartenverwaltung eingetragen worden sind, siehe **eth_assign_card_ip**. • Nach erfolgreicher Ausführung von **eth_assign_card** wird **eth_count_cards** automatisch um 1 erhöht. • Nach erfolgreicher Ausführung von **eth_remove_card** wird **eth_count_cards** automatisch um 1 verringert. • Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_assign_card**, **eth_assign_card_ip**, **eth_found_cards**, **eth_max_card**, **eth_remove_card**, **rtc6_count_cards**

Kontrollbefehl	eth_found_cards
Funktion	Gibt die Anzahl der RTC6 Ethernet-Karten in der Suchergebnisliste zurück.
Aufruf	`Result = eth_found_cards()`
Parameter	Keine.
Rückgabe	Anzahl der RTC6 Ethernet-Karten in der Suchergebnisliste. Als 32-Bit-Wert ohne Vorzeichen. 0: Es sind keine RTC6 Ethernet-Karten in der Suchergebnisliste enthalten. Möglicherweise wurde die Suche noch nicht durchgeführt. n: Anzahl der RTC6 Ethernet-Karten, die in der Suchergebnisliste enthalten sind.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• **eth_found_cards** stimmt *nicht* unbedingt überein mit – der RTC6 Ethernet-Karten-Anzahl in der RTC6-Kartenverwaltung – der Gesamtzahl aus RTC6 Ethernet-Karten und RTC6 PCIe-Karten • Die Suchergebnisliste wird mit **eth_search_cards** oder **eth_search_cards_range** erzeugt. Deren Rückgabewert stimmt mit dem von **eth_found_cards** überein. Siehe auch Kapitel 16.5.2 "Über die Suche nach RTC6 Ethernet-Karten", Seite 1007. • Informationen zu einer bestimmten RTC6 Ethernet-Karte in der Suchergebnisliste können mit **eth_get_card_info_search** abgefragt werden. • Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_get_card_info_search**, **eth_search_cards**, **eth_search_cards_range**

Kontrollbefehl	eth_full_reset
Funktion	Löst einen (softwaregesteuerten) Hardware-Reset der RTC6 Ethernet-Karte aus.
Aufruf	`eth_full_reset( CardNo )`
Parameter	`CardNo` Wie **eth_get_card_info**. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Erfolg. 1 Ethernet-Fehler. Details: **eth_get_error**. 2 Keine RTC6 Ethernet-Karte (**get_last_error**-Returncode `RTC6_TYPE_REJECTED`).
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	–

Kontrollbefehl	eth_get_card_info
Funktion	Gibt Informationen zu einer bestimmten RTC6 Ethernet-Karte zurück, die in der RTC6-Kartenverwaltung eingetragen und einmal in Besitz genommen war oder ist.
Aufruf	`eth_get_card_info( CardNo, Ptr )`
Parameter	`CardNo` Index der RTC6 Ethernet-Karte in der RTC6-Kartenverwaltung. Als 32-Bit-Wert ohne Vorzeichen. `Ptr` Karteninformationen. Als Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von 16 32-Bit-Werten ohne Vorzeichen.
Rückgabe	Keine.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zum Speichern der Karteninformationen muss vom Anwenderprogramm (an der über `Ptr` angegebenen Adresse) ein Speicherbereich der Größe 16 × 4 Byte bereitgestellt werden. • Karteninformationen [0] = Firmwareversion der Ethernet-Kommunikationssoftware [1] = Seriennummer [2] = IP-Adresse (Big-Endian-Format) [3] = MAC-Adresse (untere 4 Bytes) [4] = MAC-Adresse (obere 2 Bytes) [5] = Ist in Besitz genommen (1 oder 0) [6] = IP-Adresse des verbundenen PCs (Big-Endian-Format) [7] = Force DHCP (1 oder 0) [8] = Statische IP-Adresse (Big-Endian-Format) [9] = Statische Netzmaske (Big-Endian-Format) [10] = Statisches Gateway (Big-Endian-Format) [11] = UPD-Port für Kartensuchvorgänge [12] = UDP-Port für exklusive Verbindungen [13] = TCP-Port für eine exklusive Verbindung [14] = ≥ BIOS-ETH 43: Status – Bit #9 = 1: Remote-Interface-Modus ist aktiv (dauerhaft eingeschaltet über Standalone-Betriebsmodus) – Bit #10 = 1: Remote-Interface-Modus ist aktiv (temporär eingeschaltet über **eth_set_remote_tgm_format**) ≤ BIOS-ETH 42: Reserviert [15] = Reserviert • Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_count_cards**, **eth_get_card_info_search**, **eth_max_card**, **get_card_type**

Kontrollbefehl	eth_get_card_info_search
Funktion	Gibt Informationen zu einer bestimmten RTC6 Ethernet-Karte in der Suchergebnisliste zurück.
Aufruf	`eth_get_card_info_search( SearchNo, Ptr )`
Parameter	`SearchNo` Index der Suchergebnisliste. Als 32-Bit-Wert ohne Vorzeichen. `Ptr` Karteninformationen. Als Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von 16 32-Bit-Werten ohne Vorzeichen.
Rückgabe	Keine.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Die Suchergebnisliste wird mit **eth_search_cards** oder **eth_search_cards_range** erzeugt. Deren Rückgabewert stimmt mit dem von **eth_found_cards** überein. Siehe auch Kapitel 16.5.2 "Über die Suche nach RTC6 Ethernet-Karten", Seite 1007. • Für `SearchNo` = 0 und `SearchNo` > **eth_found_cards** wird eine Default-Information zurückgegeben. Außerdem wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Wurde noch gar keine Suche ausgeführt, dann gilt automatisch: `SearchNo` > **eth_found_cards**. • Zur Speicherung der Karteninformationen muss vom Anwenderprogramm (an der über `Ptr` angegebenen Adresse) ein Speicherbereich der Größe 16 × 4 Byte bereitgestellt werden. • Karteninformationen: Wie **eth_get_card_info**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_found_cards**, **eth_get_card_info**, **eth_search_cards**, **eth_search_cards_range**

Kontrollbefehl	**eth_get_com_timeouts**
Funktion	Veraltet. Gibt Timing-Informationen für eine RTC6 Ethernet-Karte zurück, die in der RTC6-Kartenverwaltung eingetragen ist.
Aufruf	`eth_get_com_timeouts( &AcquireTimeout, &AcquireMaxRetries, &SendRecvTimeout, &SendRecvMaxRetries, &KeepAlive, &KeepInterval )`
Rückgabe	Keine.
Parameter-Rückgaben	`AcquireTimeout` Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. `AcquireMaxRetries` Siehe `AcquireTimeout`. `SendRecvTimeout` Siehe `AcquireTimeout`. `SendRecvMaxRetries` Siehe `AcquireTimeout`. `KeepAlive` Siehe `AcquireTimeout`. `KeepInterval` Siehe `AcquireTimeout`. Alle Werte sind jeweils 0, wenn die Karte keine RTC6 Ethernet-Karte ist. Alle Zeitangaben in μs.
Multi-Board-Befehlsname	**n_eth_get_com_timeouts**
Hinweise	• Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008. • Mit **eth_set_com_timeouts** können die Einstellungen geändert werden (normalerweise nur im Kontext einer Problemklärung und nur auf Aufforderung durch SCANLAB). • Die Parameter `AcquireTimeout`, `AcquireMaxRetries`, `SendRecvTimeout` und `SendRecvMaxRetries` beziehen sich nur auf die RTC6-DLL. Daher können sie auch ohne Zugriffsberechtigung für eine RTC6 Ethernet-Karte ausgelesen werden. • Die zurückgegebenen Parameter `KeepAlive` und `KeepInterval` haben den Wert 0, wenn es für die angesprochene RTC6 Ethernet-Karte – keine Zugriffsberechtigung gibt (≤ DLL 618: **get_last_error**-Returncode `RTC6_ACCESS_DENIED` und `RTC6_ETH_ERROR`; ≥ DLL 619: keine Fehlermeldung mehr) – zwar eine Zugriffsberechtigung gibt, jedoch keine Verbindung besteht (**get_last_error**-Returncode `RTC6_ETH_ERROR`)
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611. Letzte Änderung DLL 619: keine Fehlermeldung mehr, wenn ohne Zugriffsberechtigung.
Verweise	**eth_set_com_timeouts**, **eth_set_com_timeouts_auto**

Kontrollbefehl	eth_get_com_timeouts_auto	
Funktion	Gibt Timing-Informationen (für einen automatischen Mechanismus) für eine RTC6 Ethernet-Karte zurück, die in der RTC6-Kartenverwaltung eingetragen ist.	
Aufruf	`eth_get_com_timeouts_auto( &InitialTimeout, &MaxTimeout, &Multiplier, &Mode )`	
Rückgabe	Keine.	
Parameter-Rückgaben	`InitialTimeout`	Timeout, mit dem gestartet wird. In ms. Als Zeiger auf einen Wert im 64-Bit-IEEE Gleitkommaformat.
	`MaxTimeout`	Maximales Timeout (= Summe der Timeouts der einzelnen Versuche). Nach Verstreichen von `MaxTimeout` wird die Netzwerkverbindung als fehlerhaft betrachtet (`udp_recv_timeout` Fehler, siehe **eth_get_last_error**). In ms. Als Zeiger auf einen Wert im 64-Bit-IEEE Gleitkommaformat.
	`Multiplier`	Multiplikationsfaktor, um den der Timeout bei jedem Versuch erhöht wird. Als Zeiger auf einen Wert im 64-Bit-IEEE Gleitkommaformat.
	`Mode`	Reserviert. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_eth_get_com_timeouts_auto**	
Hinweise	• Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008. • Mit **eth_set_com_timeouts_auto** können die Einstellungen geändert werden. • **eth_get_com_timeouts_auto** liefert die Werte des automatischen Mechanismus (zum Setzen geeigneter Timeout-Werte) zurück. • Alle Zeitangaben sind in ms, nicht μs. • Default-Werte nach **eth_assign_card** oder **eth_assign_card_ip**: `InitialTimeout` 0,75, `MaxTimeout` 20,0, `Multiplier` 1,3 • Der Parameter `Mode` ist für zukünftige Erweiterungen reserviert. Gibt 1 zurück. • **eth_get_com_timeouts_auto** wirkt nur auf die RTC6-DLL. Es muss keine Verbindung zur RTC6 Ethernet-Karte bestehen.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 620, OUT 620.	
Verweise	**eth_set_com_timeouts_auto**, **eth_get_com_timeouts**	

Kontrollbefehl	**eth_get_error**		
Funktion	Gibt einen akkumulierten Fehlercode zurück. Dieser beinhaltet alle Fehler, die bei den zuletzt ausgeführten RTC6 Ethernet-Karten-Befehlen aufgetreten sind.		
Aufruf	`EthAccError = eth_get_error()`		
Parameter	Keine.		
Rückgabe	Fehlercode. Identisch zu **eth_get_last_error**. Als 32-Bit-Wert ohne Vorzeichen. Sind mehrere Fehler gleichzeitig aufgetreten, so sind mehrere Bits gesetzt.		
	`no_error = 0. Kein Fehler.`	Fehler	Zahlenwert ($2^{[Bit]}$)
	`wsasstartup_failed`	Bit 0	1
	`wrong_winsock_version`	Bit 1	2
	`Reserviert.`	Bit 2	4
	`udp_create_socket_error`	Bit 3	8
	`udp_bind_socket_error`	Bit 4	16
	`udp_connect_socket_error`	Bit 5	32
	`udp_excl_add_use_error`	Bit 6	64
	`udp_bc_ena_error`	Bit 7	128
	`Reserviert.`	Bit 8	256
	`tcp_create_socket_error`	Bit 9	512
	`tcp_excl_add_use_error`	Bit 10	1.024
	`tcp_bind_socket_error`	Bit 11	2.048
	`Reserviert.`	Bit 12	4.096
	`tcp_connect_socket_error`	Bit 13	8.192
	`tcp_connect_sel_error`	Bit 14	16.384
	`tcp_connect_fds_error`	Bit 15	32.768
	`tcp_nodelay_error`	Bit 16	65.536
	`create_thread_failed`	Bit 17	131.072
	`udp_recv_error`	Bit 18	262.144
	`udp_send_error`	Bit 19	524.288
	`udp_recv_timeout`	Bit 20	1.048.576
	`already_acquired`	Bit 21	2.097.152
	`not_acquired`	Bit 22	4.194.304
	`access_denied`	Bit 23	8.388.608
	`send_tgm_timeout`	Bit 24	16.777.216
	`card_not_found`	Bit 25	33.554.432
	`core1_timeout`	Bit 26	67.108.864
	`bootmode_set_failed`	Bit 27	134.217.728
Multi-Board-Befehlsname	**n_eth_get_error**		

Kontrollbefehl	eth_get_error
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133. • **eth_get_error** und **n_eth_get_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC6 Ethernet-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **eth_get_error** weder erzeugt noch verändert. • Wenn bei einem RTC6 Ethernet-Karten-Befehlsaufruf (intern) mehrere Fehler nacheinander auftreten, gibt **eth_get_last_error** möglicherweise nur den wirklich letzten Fehler (z.B. nur "not_acquired") zurück. Die tatsächlichen Fehlerursachen können mit **eth_get_error** festgestellt werden, weil dieser die akkumulierten Fehler zurück gibt. • Der akkumulierte Fehler wird automatisch mit jeder Inbesitznahme der RTC6 Ethernet-Karte zurückgesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 608, OUT 608, RBF 611.
Verweise	**acquire_rtc**, **eth_get_last_error**

Kontrollbefehl	eth_get_ip
Funktion	Gibt für einen angegebenen RTC6-Kartenverwaltungsindex die zugehörige IP-Adresse zurück.
Aufruf	`IP = eth_get_ip( CardNo )`
Parameter	`CardNo` Index der RTC6 Ethernet-Karte in der RTC6-Kartenverwaltung. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	IP-Adresse. In Big-Endian Byte-Reihenfolge. Der Wert ist 0, wenn die Karte keine RTC6 Ethernet-Karte ist.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zur Konvertierung von der Big-Endian Byte-Reihenfolge in die übliche Dezimalpunktschreibweise dient **eth_convert_ip_to_string**. Siehe auch Kapitel 16.5.1 "Hinweise zum Arbeiten mit IP-Adressen", Seite 1007. • **eth_get_ip** erlaubt eine schnellere Abfrage der IP-Adresse aus der RTC6-Kartenverwaltung als über **eth_get_card_info**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_convert_ip_to_string**, **eth_get_card_info**

Kontrollbefehl	eth_get_ip_search
Funktion	Gibt für eine bestimmte RTC6 Ethernet-Karte in der Suchergebnisliste die zugehörige IP-Adresse zurück.
Aufruf	`IP = eth_get_ip_search( SearchNo )`
Parameter	`SearchNo` Index in der Suchergebnisliste. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	IP-Adresse. In Big-Endian Byte-Reihenfolge.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Die Suchergebnisliste wird mit **eth_search_cards** oder **eth_search_cards_range** erzeugt. Deren Rückgabewert stimmt mit dem von **eth_found_cards** überein. Siehe auch Kapitel 16.5.2 "Über die Suche nach RTC6 Ethernet-Karten", Seite 1007. • Der Rückgabewert ist 0 für `SearchNo` = 0 und `SearchNo` > **eth_found_cards**. Gleichzeitig wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Wurde noch gar keine Suche ausgeführt, dann gilt automatisch: `SearchNo` > **eth_found_cards**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 608, OUT 608, RBF 611.
Verweise	**eth_get_ip**

Kontrollbefehl	eth_get_last_error		
Funktion	Gibt einen Fehlercode zurück, der nur die beim zuletzt ausgeführten RTC6 Ethernet-Karten-Befehl aufgetretenen Fehler beinhaltet.		
Aufruf	`EthLastError = eth_get_last_error()`		
Parameter	Keine.		
Rückgabe	Fehlercode. Identisch zu **eth_get_error**. Als 32-Bit-Wert ohne Vorzeichen. Sind mehrere Fehler gleichzeitig aufgetreten, dann sind mehrere Bits gesetzt.		
	`no_error = 0. Kein Fehler.`	Fehler	Zahlenwert ($2^{[Bit]}$)
	`wsasstartup_failed`	Bit 0	1
	`wrong_winsock_version`	Bit 1	2
	`Reserviert.`	Bit 2	4
	`udp_create_socket_error`	Bit 3	8
	`udp_bind_socket_error`	Bit 4	16
	`udp_connect_socket_error`	Bit 5	32
	`udp_excl_add_use_error`	Bit 6	64
	`udp_bc_ena_error`	Bit 7	128
	`Reserviert.`	Bit 8	256
	`tcp_create_socket_error`	Bit 9	512
	`tcp_excl_add_use_error`	Bit 10	1.024
	`tcp_bind_socket_error`	Bit 11	2.048
	`Reserviert.`	Bit 12	4.096
	`tcp_connect_socket_error`	Bit 13	8.192
	`tcp_connect_sel_error`	Bit 14	16.384
	`tcp_connect_fds_error`	Bit 15	32.768
	`tcp_nodelay_error`	Bit 16	65.536
	`create_thread_failed`	Bit 17	131.072
	`udp_recv_error`	Bit 18	262.144
	`udp_send_error`	Bit 19	524.288
	`udp_recv_timeout`	Bit 20	1.048.576
	`already_acquired`	Bit 21	2.097.152
	`not_acquired`	Bit 22	4.194.304
	`access_denied`	Bit 23	8.388.608
	`send_tgm_timeout`	Bit 24	16.777.216
	`card_not_found`	Bit 25	33.554.432
	`core1_timeout`	Bit 26	67.108.864
	`bootmode_set_failed`	Bit 27	134.217.728
Multi-Board-Befehlsname	**n_eth_get_last_error**		

Kontrollbefehl	eth_get_last_error
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133. • **eth_get_last_error** und **n_eth_get_last_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC6 Ethernet-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **eth_get_last_error** weder erzeugt noch verändert. • Jeder **eth_get_last_error**-Fehler führt auch zu einem **get_last_error**-Fehler `RTC6_ETH_ERROR`. • Wenn bei einem RTC6 Ethernet-Karten-Befehlsaufruf intern mehrere Fehler nacheinander auftreten, gibt **eth_get_last_error** möglicherweise nur den wirklich letzten Fehler (z.B. nur "not_acquired") zurück. Die tatsächlichen Fehlerursachen können mit **eth_get_error** festgestellt werden, weil dieser die akkumulierten Fehler zurück gibt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_get_error**, **get_error**, **reset_error**, **set_verify**

Kontrollbefehl	eth_get_port_numbers
Funktion	Gibt die UDP-Ports für Kartensuchvorgänge (`UDPsearch`) und exklusive Verbindungen (`UDPexcl`) sowie den TCP-Port einer RTC6 Ethernet-Karte zurück.
Aufruf	`Result = eth_get_port_numbers( &UDPsearch, &UDPexcl, &TCP )`
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0: Erfolg: OK. 1: Fehler: Keine RTC6 Ethernet-Karte.
Parameter-Rückgaben	`UDPsearch` UPD-Port für Kartensuchvorgänge. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. `UDPexcl` UDP-Port für exklusive Verbindungen. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. `TCP` TCP-Port für exklusive Verbindungen. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. Alle Werte sind jeweils 0, wenn die Karte keine RTC6 Ethernet-Karte ist.
Multi-Board-Befehlsname	**n_eth_get_port_numbers**
Hinweise	• Die Werte können auch mit **eth_get_card_info** abgefragt werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_get_card_info**, **eth_set_port_numbers**

Kontrollbefehl	eth_get_serial_search
Funktion	Gibt für eine bestimmte RTC6 Ethernet-Karte in der Suchergebnisliste die zugehörige Seriennummer zurück.
Aufruf	`Serialnumber = eth_get_serial_search( SearchNo )`
Parameter	`SearchNo` Index in der Suchergebnisliste. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Seriennummer. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Die Suchergebnisliste wird mit **eth_search_cards** oder **eth_search_cards_range** erzeugt. Deren Rückgabewert stimmt mit dem von **eth_found_cards** überein. Siehe auch Kapitel 16.5.2 "Über die Suche nach RTC6 Ethernet-Karten", Seite 1007. • Der Rückgabewert ist 0 für `SearchNo` = 0 und `SearchNo` > **eth_found_cards**. Gleichzeitig wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Wurde noch gar keine Suche ausgeführt, dann gilt automatisch: `SearchNo` > **eth_found_cards**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 608, OUT 608, RBF 611.
Verweise	**get_serial_number**, **eth_get_card_info_search**

Kontrollbefehl	**eth_get_standalone_status**
Funktion	Gibt Standalone-Funktionalitäts-bezogene Informationen über die RTC6 Ethernet-Karte zurück.
Voraussetzung	RTC6-Software-Paket ≥ V1.9.0 und BIOS-ETH ≥ 30.
Aufruf	`eth_get_standalone_status( &Status, &Error, &Mode )`
Rückgabe	Keine.
Parameter	Keine.
Parameter-Rückgaben	`Status` Status des automatischen Bootens, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. = 0: Beendet. = 1: Läuft. `Error` Fehler während des automatischen Bootens. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. = 0: Nein. = 1: Ja. `Mode` Ziel-Zustand nach dem automatischen Booten (nach dem nächsten Hardware-Reset). Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. = 0: "Normaler PC-Betriebs-Zustand". = 1: "Standalone-Basis-Zustand". = 2: "Standalone-Voll-Zustand".
Multi-Board-Befehlsname	**n_eth_get_standalone_status**
Hinweise	• **eth_get_standalone_status** erfordert die Zugriffsberechtigung zur RTC6 Ethernet-Karte. Anderenfalls hat **eth_get_standalone_status** keine Wirkung (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`). • **eth_get_standalone_status** hat keine Wirkung, wenn die Karte keine RTC6 Ethernet-Karte ist (**get_last_error**-Returncode `RTC6_TYPE_REJECTED`). • Auch während des automatischen Bootens kann mit **acquire_rtc** die Zugriffsberechtigung angefordert werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 628, OUT 629.
Verweise	–

Kontrollbefehl	**eth_get_static_ip**
Funktion	Gibt die auf der RTC6 Ethernet-Karte gespeicherte statische IP-Adresse, Subnetzmaske und Gateway-Adresse zurück.
Aufruf	`Result = eth_get_static_ip( &Ip, &NetMask, &Gateway )`
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0: Erfolg: OK. 1: Fehler: Keine RTC6 Ethernet-Karte.
Parameter	Keine.
Parameter-Rückgaben	`Ip` Statische IP-Adresse in Big-Endian Byte-Reihenfolge. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. `NetMask` Subnetzmaske in Big-Endian Byte-Reihenfolge. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. `Gateway` Adresse des Gateways in Big-Endian Byte-Reihenfolge. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. Alle Werte sind jeweils 0, wenn die Karte keine RTC6 Ethernet-Karte ist oder noch keine statische IP-Adresse programmiert wurde.
Multi-Board-Befehlsname	**n_eth_get_static_ip**
Hinweise	• Zur Konvertierung von der Big-Endian Byte-Reihenfolge in die übliche Dezimalpunktschreibweise dient **eth_convert_ip_to_string**. Siehe auch Kapitel 16.5.1 "Hinweise zum Arbeiten mit IP-Adressen", Seite 1007. • Die Werte können auch mit **eth_get_card_info** abgefragt werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_convert_ip_to_string**, **eth_get_card_info**, **eth_set_static_ip**

Kontrollbefehl	eth_max_card
Funktion	Gibt den höchsten Index in der RTC6-Kartenverwaltung zurück, an dem eine RTC6 Ethernet-Karte eingetragen ist.
Aufruf	`Result = eth_max_card()`
Parameter	Keine.
Rückgabe	Höchster Index einer RTC6 Ethernet-Karte in der RTC6-Kartenverwaltung. Als 32-Bit-Wert ohne Vorzeichen. 0: In der RTC6-Kartenverwaltung ist keine RTC6 Ethernet-Karte eingetragen. n: Höchster Index, an dem eine RTC6 Ethernet-Karte in der RTC6-Kartenverwaltung eingetragen ist.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.

Hinweise	• **eth_max_card** liefert nicht die Gesamtzahl der RTC6 Ethernet-Karten in der RTC6-Kartenverwaltung zurück. Dazu steht **eth_count_cards** zur Verfügung.
	• Mit **eth_max_card** können alle RTC6 Ethernet-Karten aus der RTC6-Kartenverwaltung durch eine einfache Schleife entfernt werden:
	`while ( eth_max_card() )`
	`{ release_rtc( eth_max_card() ); eth_remove_card( eth_max_card() ); }`
	• Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_count_cards**, **eth_remove_card**

Kontrollbefehl	eth_remove_card
Funktion	Löscht den RTC6 Ethernet-Karten-Eintrag aus der RTC6-Kartenverwaltung am angegebenen Index.
Aufruf	`Result = eth_remove_card( CardNo )`
Parameter	`CardNo` Index der RTC6 Ethernet-Karte in der RTC6-Kartenverwaltung. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Fehlercode. Als 32-Bit-Wert mit Vorzeichen. –2 Fehler: Der Eintrag kann nicht gelöscht werden. An diesem Index ist eine RTC6 Ethernet-Karte eingetragen, die aber noch in Besitz genommen ist. –1 Fehler: Der Eintrag kann nicht gelöscht werden. An diesem Index ist eine RTC6 PCIe-Karte eingetragen. 0 Fehler: Der Eintrag kann nicht gelöscht werden. An diesem Index ist "Keine Karte" eingetragen oder `CardNo` ist ungültig (> 255). n (= `CardNo`) Erfolg: Der Eintrag wurde entfernt.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Bei allen Fehlern wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Einträge für "Keine Karte" und RTC6 PCIe-Karten können nicht gelöscht werden. • Vor dem Löschen eines RTC6 Ethernet-Karten-Eintrags muss eine in Besitz genommene Karte mit **release_rtc** explizit freigegeben werden. • Mit `CardNo` = 0 wird der RTC6 Ethernet-Karten-Eintrag am Index (**rtc6_count_cards** + **eth_count_cards**) gelöscht (siehe auch lückenlose Kartennummerierung mit **eth_assign_card** oder **eth_assign_card_ip** und `CardNo` = 0). `CardNo` = 0 und freigewählte Zuweisung von RTC6 Ethernet-Karten zu Indices sollten nicht zusammen verwendet werden. • Alle RTC6 Ethernet-Karten-Einträge können aus der RTC6-Kartenverwaltung durch eine einfache Schleife entfernt werden (vorausgesetzt, alle RTC6 Ethernet-Karten sind *nicht* in Besitz genommen): `while ( eth_remove_card(0)> 0 );` • Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_assign_card**, **eth_assign_card_ip**, **eth_count_cards**, **release_rtc**, **rtc6_count_cards**

Kontrollbefehl	eth_search_cards
Funktion	Führt eine Suche nach RTC6-Karten in Form eines Broadcasts aus und gibt die Anzahl der RTC6 Ethernet-Karten zurück, die im angegebenen Adressbereich geantwortet haben.
Aufruf	`Result = eth_search_cards( Ip, Netmask )`
Parameter	`Ip` IP-Adresse in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen. `Netmask` Subnetzmaske in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Anzahl der RTC6 Ethernet-Karten, die im angegebenen Adressbereich (innerhalb von TimeOut) geantwortet haben. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zur Umwandlung aus der üblichen Dezimalpunktschreibweise in die Big-Endian Byte-Reihenfolge kann **eth_convert_string_to_ip** verwendet werden. Siehe auch Kapitel 16.5.1 "Hinweise zum Arbeiten mit IP-Adressen", Seite 1007. • Der Timeout-Wert kann mit **eth_set_search_cards_timeout** eingestellt werden. • Ein Broadcast erreicht zuverlässig nur Adressen innerhalb des angegebenen Netzwerksegmentes. Für andere Anwendungsfälle kann eine Kartensuche mit IP-Scan (**eth_search_cards_range**) durchgeführt werden. • Die Anzahl der gefundenen RTC6 Ethernet-Karten kann auch zu einem späteren Zeitpunkt mit **eth_found_cards** abgefragt werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_convert_string_to_ip**, **eth_found_cards**, **eth_search_cards_range**, **eth_set_search_cards_timeout**

Kontrollbefehl	eth_search_cards_range
Funktion	Führt eine Suche in einem konkret angegebenen IP-Adressbereich aus und gibt die Anzahl der RTC6 Ethernet-Karten zurück, die geantwortet haben.
Aufruf	`Result = eth_search_cards_range( StartIP, EndIp )`
Parameter	`StartIP` Start-IP-Adresse in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen. `EndIp` Ende-IP-Adresse in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Anzahl der RTC6 Ethernet-Karten, die (innerhalb von TimeOut) geantwortet haben. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zur Umwandlung aus der üblichen Dezimalpunktschreibweise in die Big-Endian Byte-Reihenfolge kann **eth_convert_string_to_ip** verwendet werden. Siehe auch Kapitel 16.5.1 "Hinweise zum Arbeiten mit IP-Adressen", Seite 1007. • **eth_search_cards_range** führt die Suche durch Versenden von UDP-Paketen an jede IP-Adresse im angegebenen Adressbereich aus. Sie umfasst verlässlich den angegebenen Adressbereich (vgl. Broadcast bei **eth_search_cards**). • Der TimeOut-Wert kann mit **eth_set_search_cards_timeout** eingestellt werden. • Die Anzahl der gefundenen RTC6 Ethernet-Karten kann auch zu einem späteren Zeitpunkt mit **eth_found_cards** abgefragt werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_convert_string_to_ip**, **eth_found_cards**, **eth_search_cards**, **eth_set_search_cards_timeout**

Kontrollbefehl	**eth_set_com_timeouts**	
Funktion	Veraltet. Stellt Timing-Informationen (für die Ethernet-Kommunikation) für eine RTC6 Ethernet-Karte ein, die in der RTC6-Kartenverwaltung eingetragen ist (und falls `KeepAlive` und `KeepInterval` geändert werden sollen, außerdem in Besitz genommen sein muss).	
Aufruf	`eth_set_com_timeouts( AcquireTimeout, AcquireMaxRetries, SendRecvTimeout, SendRecvMaxRetries, KeepAlive, KeepInterval )`	
Parameter	`AcquireTimeout`	Als 32-Bit-Wert ohne Vorzeichen. Bei 0 wird der Parameter nicht verändert.
	`AcquireMaxRetries`	Siehe `AcquireTimeout`.
	`SendRecvTimeout`	Siehe `AcquireTimeout`.
	`SendRecvMaxRetries`	Siehe `AcquireTimeout`.
	`KeepAlive`	Siehe `AcquireTimeout`.
	`KeepInterval`	Siehe `AcquireTimeout`.
Rückgabe	Keine.	
Multi-Board-Befehlsname	**n_eth_set_com_timeouts**	
Hinweise	• Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008. • Die Einstellungen können mit **eth_get_com_timeouts** abgefragt werden. • Alle Zeitangaben in μs. • Normalerweise ist es nicht erforderlich, diese Parameterwerte zu ändern. Sollten Timing-Probleme auftreten, wenden Sie sich an SCANLAB, um Ihre lokalen Besonderheiten zu klären. • `AcquireTimeout`, `AcquireMaxRetries`, `SendRecvTimeout` und `SendRecvMaxRetries` beziehen sich nur auf die RTC6-DLL. Daher können sie auch ohne Zugriffsberechtigung für eine RTC6 Ethernet-Karte gesetzt werden. • `KeepAlive` und `KeepInterval` werden nicht gesetzt, wenn es für die angesprochene RTC6 Ethernet-Karte – keine Zugriffsberechtigung gibt (**get_last_error**-Returncode `RTC6_ACCESS_DENIED` und `RTC6_ETH_ERROR`) – zwar eine Zugriffsberechtigung gibt, jedoch keine Verbindung besteht (**get_last_error**-Returncode `RTC6_ETH_ERROR`) • ≥ DLL 619: Bei `KeepAlive` = `KeepInterval` = 0 wird kein **get_last_error**-Returncode gesetzt, da eine Verbindung zu einer RTC6 Ethernet-Karte nicht notwendig ist. • **eth_set_com_timeouts_auto** stellt einen automatischen Mechanismus ein, siehe dort.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611. Letzte Änderung DLL 619: Keine Fehlermeldung mehr, wenn keine RTC6 Ethernet-Karten-Verbindung besteht.	
Verweise	**eth_get_com_timeouts**, **eth_set_com_timeouts_auto**	

Kontrollbefehl	eth_set_com_timeouts_auto	
Funktion	Stellt Timing-Informationen (für die Ethernet-Kommunikation) für eine RTC6 Ethernet-Karte ein, die in der RTC6-Kartenverwaltung eingetragen ist.	
Aufruf	`eth_set_com_timeouts_auto( InitialTimeout, MaxTimeout, Multiplier, Mode )`	
Parameter	`InitialTimeout`	Timeout, mit dem gestartet wird. In ms. Als Wert im 64-Bit-IEEE Gleitkommaformat.
	`MaxTimeout`	Maximales Timeout (= Summe der Timeouts der einzelnen Versuche). Nach Verstreichen von `MaxTimeout` wird die Netzwerkverbindung als fehlerhaft betrachtet (`udp_recv_timeout` Fehler, siehe **eth_get_last_error**). In ms. Als Wert im 64-Bit-IEEE Gleitkommaformat.
	`Multiplier`	Multiplikationsfaktor, um den der Timeout bei jedem Versuch erhöht wird. Als Wert im 64-Bit-IEEE Gleitkommaformat.
	`Mode`	Reserviert. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Keine.	
Multi-Board-Befehlsname	**n_eth_set_com_timeouts_auto**	
Hinweise	• Siehe auch Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008. • Jeder Kontrollbefehl und Listenbefehl wird in Form von Netzwerk-Paketen zur RTC6 Ethernet-Karte gesendet. Auf jedes gesendete Paket muss das Antwort-Paket innerhalb eines bestimmten Timeout empfangen werden. Wird dieser Timeout überschritten, dann wird der Vorgang wiederholt. Um eine Überlastung zu vermeiden, wird mit jedem weiteren Versuch der nächste Timeout-Wert um den Faktor `Multiplier` erhöht. • Die Einstellungen können mit **eth_get_com_timeouts_auto** abgefragt werden. • Alle Zeitangaben in ms, nicht µs. • Wird für einen Parameter 0 angegeben, wird sein aktueller Wert nicht verändert. Parameter-Default-Werte (nach **eth_assign_card** und **eth_assign_card_ip**): siehe **eth_get_com_timeouts_auto**. • Der Parameter `Mode` ist für zukünftige Erweiterungen reserviert. • **eth_set_com_timeouts_auto** wirkt nur auf die RTC6-DLL. Es muss keine Verbindung zur RTC6 Ethernet-Karte bestehen. • **eth_set_com_timeouts** stellt einen statischen Mechanismus ein.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 620, OUT 620.	
Verweise	**eth_get_com_timeouts_auto**, **eth_set_com_timeouts**	

Kontrollbefehl	**eth_set_high_performance_mode**
Funktion	Schaltet für eine RTC6 Ethernet-Karte den "High-Performance-Modus" ein (aus).
Voraussetzung	Mindestanforderung: RTC6-Software-Paket V1.14.1 und BIOS-ETH 35.
Aufruf	`ErrorCode = eth_set_high_performance_mode( Mode )`
Parameter	`Mode` Modus. Als 32-Bit-Wert ohne Vorzeichen. = 0: Schaltet den "High-Performance-Modus" aus. = 1: Schaltet den "High-Performance-Modus" ein.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0: Erfolg: OK. 1: Fehler: Modus unzulässig. **get_last_error**-Returncode `RTC6_PARAM_ERROR`. 2: Fehler: BIOS nicht unterstützt (< BIOS-ETH 35). **get_last_error**-Returncode `RTC6_UNSUPPORTED_BIOS`. 3: Fehler: Kein Zugriff auf die RTC6 Ethernet-Karte. **get_last_error**-Returncode `RTC6_ACCESS_DENIED`. 4: Fehler: Keine RTC6 Ethernet-Karte. **get_last_error**-Returncode `RTC6_TYPE_REJECTED`.
Multi-Board-Befehlsname	**n_eth_set_high_performance_mode**
Hinweise	• Im "High-Performance-Modus" kann die Netzwerkauslastung deutlich steigen. – **eth_set_high_performance_mode** ruft zum Ausgleich intern `eth_set_com_timeouts_auto( 0.75, 1000.0, 1.3, 1 )` auf. – Falls **eth_set_com_timeouts** zuvor mit größeren Werten aufgerufen wurde, werden diese nicht verändert. • Default-Einstellung nach **eth_assign_card** und **eth_assign_card_ip**: – `Mode` = 0 • **load_program_file** setzt die Einstellung nicht zurück. • Siehe auch Kapitel 16.9 ""High-Performance-Modus"", Seite 1020.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 633.
Verweise	–

Kontrollbefehl	**eth_set_port_numbers**
Funktion	Speichert die UDP-Ports für Kartensuchvorgänge (`UDPsearch`) und exklusive Verbindungen (`UDPexcl`) sowie den TCP-Port in den Flash-Speicher der RTC6 Ethernet-Karte.
Aufruf	`Result = eth_set_port_numbers( UDPsearch, UDPexcl, TCP )`
Parameter	`UDPsearch` UPD-Port für Kartensuchvorgänge. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…65.535]. Überlaufende Werte werden geclippt. `UDPexcl` UDP-Port für exklusive Verbindungen. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…65.535]. Überlaufende Werte werden geclippt. `TCP` TCP-Port für exklusive Verbindungen. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…65.535]. Überlaufende Werte werden geclippt.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0: Erfolg: OK. 1: Fehler: Kein Zugriff auf die RTC6 Ethernet-Karte. **get_last_error**-Returncode `RTC6_ACCESS_DENIED`. 2: Fehler: Keine RTC6 Ethernet-Karte. **get_last_error**-Returncode `RTC6_TYPE_REJECTED`. 3: Fehler: Programmierung des Flash-Speichers ist nicht möglich. Möglicherweise ist die RTC6 Ethernet-Karte beschäftigt. **get_last_error**-Returncode `RTC6_BUSY` oder `RTC6_FLASH_ERROR`.
Multi-Board-Befehlsname	**n_eth_set_port_numbers**
Hinweise	• Voraussetzungen für **eth_set_port_numbers**: – Auf der RTC6 Ethernet-Karte darf gerade keine Liste ausgeführt werden. – Die RTC6 Ethernet-Karte muss in der RTC6-Kartenverwaltung eingetragen sein. – Die RTC6 Ethernet-Karte muss in Besitz genommen sein. • Wird für einen Parameter 0 angegeben, so wird dieser Parameter nicht überschrieben. • Gespeicherte Parameter werden erst nach einem Neustart der RTC6 Ethernet-Karte verwendet.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_get_port_numbers**

Kontrollbefehl	**eth_set_remote_tgm_format**
Funktion	Schaltet den Remote-Interface-Modus ein (aus).
Voraussetzung	Mindestanforderung: RTC6-Software-Paket V1.16.1 und BIOS-ETH 39.
Aufruf	`eth_set_remote_tgm_format( Format )`
Parameter	`Format` Telegramm-Format. Als 32-Bit-Wert ohne Vorzeichen. = 0: Remote-Interface-Modus wird nicht eingeschaltet. = 1: Remote-Interface-Modus wird eingeschaltet. > 1: Reserviert.
Rückgabe	Keine.
Multi-Board-Befehlsname	**n_eth_set_remote_tgm_format**
Hinweise	• **eth_set_remote_tgm_format** ist nur bei RTC6 Ethernet-Karten erlaubt. Anderenfalls wird ein **get_last_error**-Returncode `RTC6_TYPE_REJECTED` erzeugt. • Ungültiger `Format`-Wert: siehe `Format`= 0. • Im "normalen" Betrieb (d.h. ohne vorherigen **eth_boot_dcmd**-Aufruf) schaltet **eth_set_remote_tgm_format** den Remote-Interface-Modus unverzüglich und temporär (bis zum nächsten Hardware-Reset) ein. Siehe auch Seite 399. • **eth_set_remote_tgm_format** wirkt in Boot-Phase 1: siehe Schritt 3 in Kapitel 16.8.2 ""Standalone-Voll-Zustand" herstellen", Seite 1014. • **eth_set_remote_tgm_format** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 643, OUT 639. Letzte Änderung DLL 646, OUT 649. Wirkt jetzt im "normalen Betrieb" (vorher: nicht).
Verweise	–

Kontrollbefehl	eth_set_search_cards_timeout
Funktion	Setzt einen Timeout-Wert.
Aufruf	`eth_set_search_cards_timeout( TimeOut )`
Parameter	`TimeOut` Timeout-Wert. In μs. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Keine.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Der voreingestellte Timeout-Wert ist 5 ms. • Der Timeout-Wert ist relevant bei **eth_search_cards** und **eth_search_cards_range**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_search_cards**, **eth_search_cards_range**

Kontrollbefehl	**eth_set_static_ip**
Funktion	Schreibt eine statische IP-Adresse, eine Subnetzmaske sowie eine Gateway-Adresse auf die RTC6 Ethernet-Karte.
Aufruf	`Result = eth_set_static_ip( Ip, NetMask, Gateway )`
Parameter	`Ip` Statische IP-Adresse in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen. `NetMask` Subnetzmaske in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen. `Gateway` Adresse des Gateways in Big-Endian Byte-Reihenfolge. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0: Erfolg: OK. 1: Fehler: Kein Zugriff auf die RTC6 Ethernet-Karte. **get_last_error**-Returncode `RTC6_ACCESS_DENIED`. 2: Fehler: Keine RTC6 Ethernet-Karte. **get_last_error**-Returncode `RTC6_TYPE_REJECTED`. 3: Fehler: Programmierung des Flash-Speichers ist nicht möglich. Möglicherweise ist die RTC6 Ethernet-Karte beschäftigt. **get_last_error**-Returncode `RTC6_BUSY` oder `RTC6_FLASH_ERROR`.
Multi-Board-Befehlsname	**n_eth_set_static_ip**
Hinweise	• Voraussetzungen für **eth_set_static_ip**: – Auf der RTC6 Ethernet-Karte darf gerade keine Liste ausgeführt werden. – Die RTC6 Ethernet-Karte muss in der RTC6-Kartenverwaltung eingetragen sein. – Die RTC6 Ethernet-Karte muss in Besitz genommen sein. • Soll kein Gateway verwendet werden, dann muss `Gateway` auf 0 gesetzt werden. • Zur Umwandlung aus der üblichen Dezimalpunktschreibweise in die Big-Endian Byte-Reihenfolge kann **eth_convert_string_to_ip** verwendet werden. Siehe auch Kapitel 16.5.1 "Hinweise zum Arbeiten mit IP-Adressen", Seite 1007.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_convert_string_to_ip**, **eth_get_static_ip**

Kontrollbefehl	**execute_at_pointer**
Funktion	Startet die Listenausführung (“Liste 1” oder “Liste 2”) an der angegebenen Adresse des RTC6-Listenspeichers.
Aufruf	`execute_at_pointer( Pos )`
Parameter	`Pos` Absolute Speicheradresse des ersten auszuführenden Listenbefehls. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_execute_at_pointer**
Hinweise	• **execute_at_pointer** wirkt grundsätzlich wie **execute_list_pos** (siehe Hinweise dort). Die Startadresse wird allerdings bei **execute_at_pointer** als *absolute* Speicheradresse, bei **execute_list_pos** dagegen unter Angabe der Listennummer und der *relativen* Speicheradresse angegeben. • Für `Pos` ≥ `Mem1` + `Mem2` (siehe **config_list**) wird `Pos` auf 0 gesetzt.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**execute_list_pos**, **get_out_pointer**

Kontrollbefehl	execute_list
Funktion	Startet die Ausführung der gewünschten Liste (“Liste 1” oder “Liste 2”) an deren Listenanfang.
Aufruf	`execute_list( ListNo )`
Parameter	`ListNo` Nummer der auszuführenden Liste. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [ungerade: “Liste 1”, gerade: “Liste 2”].
Multi-Board-Befehlsname	**n_execute_list**
Hinweise	• **execute_list ist gleichbedeutend mit execute_list_pos** mit `Pos` = 0. • Alternativ können **execute_list_1** und **execute_list_2** (ohne Parameter) verwendet werden.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**execute_list_pos**, **execute_at_pointer**, **set_start_list**

Kontrollbefehl	execute_list_1
Funktion	Siehe **execute_list**.
Aufruf	`execute_list_1()`
Multi-Board-Befehlsname	**n_execute_list_1**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**execute_list**

Kontrollbefehl	execute_list_2
Funktion	Siehe **execute_list**.
Aufruf	`execute_list_2()`
Multi-Board-Befehlsname	**n_execute_list_2**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**execute_list**

Kontrollbefehl	execute_list_pos
Funktion	Startet die Ausführung der gewünschten Liste (“Liste 1” oder “Liste 2”) an der angegebenen Position.
Aufruf	`execute_list_pos( ListNo, Pos )`
Parameter	`ListNo` Nummer der auszuführenden Liste. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [ungerade: “Liste 1”, gerade: “Liste 2”].
	`Pos` Adresse des ersten auszugebenden Listenbefehls (Offset bezüglich des Listenanfangs der jeweiligen Liste). Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_execute_list_pos**
Hinweise	• **execute_list_pos** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Wenn der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist, wird **execute_list_pos** gegebenenfalls erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Listenausführungsstatus wieder zurückgesetzt wurde). Es erfolgt keine Prüfung, ob die Liste momentan geladen wird. Während der Ausgabe einer Liste kann die andere, aber auch dieselbe Liste bereits wieder neu geladen werden, siehe auch Kapitel 6.4 “Listen-Handling”, Seite 108. • Programme im geschützten RTC6-Listenspeicher-Bereich “Liste 3” können nicht direkt mit **execute_list_pos** ausgeführt werden. Sie können nur als Unterprogramme von einer Liste aus (“Liste 1” oder “Liste 2”) aufgerufen werden. Alternativ kann der entsprechende Bereich mit **config_list** der “Liste 1” oder “Liste 2” zugeschlagen werden. • Für ungerade Werte von `ListNo` wird “Liste 1”, anderenfalls “Liste 2” ausgeführt. Dadurch kann ein ständiger Listenwechsel durch fortlaufende Zählung automatisch generiert werden. • Wenn “Liste 2” kein Speicher zugewiesen ist (`Mem2` = 0, siehe **config_list**) wird immer “Liste 1” ausgeführt. • Wird `Pos` größer als der RTC6-Listenspeicher-Bereich der jeweiligen Liste gewählt (`Pos` ≥ `Mem1` oder `Pos` ≥ `Mem2`), so wird `Pos` auf 0 gesetzt. • Der BUSY-Listenstatus der gewählten Liste wird gesetzt und der BUSY-Listenstatus der jeweils anderen Liste zurückgesetzt (siehe **read_status**). Der BUSY-Listenausführungsstatus-Listenausführungsstatus (siehe **get_status**) wird gesetzt. • Die Ausgabe stoppt, wenn ein **set_end_of_list** angetroffen wird. Wenn das Ende des jeweiligen Listen-Bereichs erreicht ist, ohne dass ein **set_end_of_list** angetroffen wurde, wird am Anfang desselben Listen-Bereichs fortgefahren, nicht mit der nächsten Liste. Der Ausgabe-Zeiger bleibt immer im aktiven Listen-Bereich, es sei denn, ein **set_end_of_list** wird angetroffen und es wurde zuvor ein **auto_change_pos** oder **start_loop** aufgerufen. Sollen beide Listen wie eine einzige behandelt werden, muss die Konfiguration entsprechend angepasst werden: z.B. `config_list( Mem1+Mem2, 0 )`.

Kontrollbefehl	execute_list_pos
Hinweise (Forts.)	• Wurde mit **set_end_of_list** ein (mit **home_position** oder **home_position_xyz** definierter) Home-Jump ausgeführt, so führt **execute_list_pos** einen entsprechenden Home-Return aus (für die Dauer des Home-Returns ist der INTERNAL-BUSY-Listenausführungsstatus gesetzt). • **execute_list_pos** löst einen Flush der gepufferten Listeneingabe aus (siehe Kapitel 6.4.1 "Listen laden", Seite 108), auch dann, wenn der Start nicht erfolgreich war. • **execute_list_pos** deckt auch die Spezialisierungen **execute_list_1**, **execute_list_2**, **execute_list** oder **execute_at_pointer** ab.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**execute_list**, **execute_at_pointer**, **set_start_list_pos**

Normaler Listenbefehl	fly_disable_list
Funktion	Deaktiviert eine zuvor eingestellte "Processing on the fly"-Korrektur und speichert deren Einstellungen (Skalierungsfaktoren und Modus) für eine Encodergeschwindigkeits-abhängige Laser-Steuerung zwischen.
Aufruf	`fly_disable_list( )`
Parameter	–
Multi-Board-Befehlsname	**n_fly_disable_list**
Hinweise	• **fly_disable_list** führt im Gegensatz zu **fly_return** keinen Sprung aus. • Die zuvor eingestellte "Processing on the fly"-Korrektur wird zwischengespeichert und für die Encodergeschwindigkeits-abhängige Laser-Steuerung (**set_auto_laser_control** mit `Mode` = 2-Erweiterung M = +4, Seite 675) verwendet. Siehe auch Hinweis, Seite 676. • Ist gerade keine "Processing on the fly"-Korrektur aktiv, löscht ein **fly_disable_list**-Aufruf die zwischengespeicherten Einstellungen. • Siehe auch Kapitel 8.6 ""Processing on the fly"", Seite 254.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 644, OUT 647.
Verweise	**fly_return**

Unverzögerter Kurzer Listenbefehl	**fly_prediction**
Funktion	Kategorie: Listenbefehle nur für einen begrenzten Benutzerkreis. Setzt die “Processing on the fly”-Gesamt-Prädiktionswerte.
Aufruf	`fly_prediction( PredictionX, PredictionY )`
Parameter	`PredictionX` “Processing on the fly”-Gesamtprädiktionswert für x. In [10/16 µs]. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…4.095].
	`PredictionY` Wie `PredictionX` (analog).
Multi-Board-Befehlsname	**n_fly_prediction**
Hinweise	• Siehe **set_fly_tracking_error**. • Für Listenbefehle nur für einen begrenzten Benutzerkreis gilt: Diese Benutzer wurden im Umgang mit diesen RTC6-Befehlen separat informiert. Deswegen wird nur die Befehlstabelle aufgenommen und auf weitere Informationen in diesem Handbuch verzichtet.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 646, OUT 649, RBF 641.
Verweise	**set_fly_tracking_error**, **set_controlpreview_compensation_ctrl**

Normaler Listenbefehl	**fly_return**
Funktion	Deaktiviert eine zuvor eingestellte “Processing on the fly”-Korrektur und führt anschließend einen Sprung auf die angegebene neue Ausgabeposition aus.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, führt **fly_return** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`fly_return( X, Y )`
Parameter	`X` Absolute x-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y` Wie `X` (analog).
Multi-Board-Befehlsname	**n_fly_return**
Hinweise	• Der Sprung auf die neue Ausgabeposition wird wie mit **jump_abs** ausgeführt (siehe Hinweise dort). • Wird eine “Processing on the fly”-Korrektur innerhalb eines mit einem “AbsCall” aufgerufenen Unterprogramms aktiviert und mit **fly_return** deaktiviert, so werden die mit **fly_return** angegebenen Koordinatenwerte (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt, siehe auch Abschnitt “”AbsCalls””, Seite 117. • Siehe auch Kapitel 8.6 “”Processing on the fly””, Seite 254.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: erhöhter Wertebereich und “AbsCall”, s.o. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_x**, **set_fly_y**, **set_fly_rot**, **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos**, **fly_disable_list**

Variabler Listenbefehl	**fly_return_1_axis**
Funktion	"Fly-Erweiterung"-Befehl: Deaktiviert 1 Achse einer "Processing on the fly"-Anwendung und führt anschließend einen Sprung auf die angegebene neue Ausgabeposition aus.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, führt **fly_return_1_axis** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`fly_return_1_axis( Axis, RetPos1 )`
Parameter	`Axis` Achse aus Tabelle 3, Seite 271. Als 32-Bit-Wert ohne Vorzeichen. `RetPos1` Absolute Achsen-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_fly_return_1_axis**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **fly_return_1_axis** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • `Axis` muss eine Linear-Achse (`1`, `2` oder `3`) sein. Zum Deaktivieren einer Dreh-Achse ist **fly_return_2_axes** zu verwenden. • **fly_return_1_axis** deaktiviert nur dann 1 Achse einer "Processing on the fly"-Anwendung, wenn diese mit einem "Fly-Erweiterung"-Befehl aktiviert wurde. • Wenn bei der angegebenen Achse keine "Processing on the fly"-Funktionalität aktiviert ist, wird nur der Sprung ausgeführt. • Bei einem unzulässigen Parameter-Wert wird **fly_return_1_axis** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Siehe auch Hinweise zu **fly_return**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**fly_return_2_axes**, **fly_return_3_axes**

Variabler Listenbefehl	**fly_return_2_axes**
Funktion	"Fly-Erweiterung"-Befehl: Deaktiviert 2 Achsen einer "Processing on the fly"-Anwendung und führt anschließend einen Sprung auf die angegebene neue Ausgabeposition aus.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, führt **fly_return_2_axes** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`fly_return_2_axes( Axis1, RetPos1, Axis2, RetPos2)`
Parameter	`Axis1` Achse aus Tabelle 3, Seite 271. Als 32-Bit-Wert ohne Vorzeichen. `RetPos1` Absolute Achsen-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt. `Axis2` Wie `Axis1`. `RetPos2` Wie `RetPos1`.
Multi-Board-Befehlsname	**n_fly_return_2_axes**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **fly_return_2_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • `Axis1` und `Axis2` müssen 2 verschiedene Linear-Achsen (`1`, `2` oder `3`) oder beide Dreh-Achse (`4`) sein. Im letzteren Fall bedeuten `RetPos1`/`RetPos2` die Returnkoordinaten der Achse `1`/`2`. • **fly_return_2_axes** deaktiviert nur dann Achsen einer "Processing on the fly"-Anwendung, wenn diese mit einem "Fly-Erweiterung"-Befehl aktiviert wurden. • Wenn für eine der angegebenen Achsen keine "Processing on the fly"-Funktion aktiviert ist, wird nur der Sprung ausgeführt. • Bei einem unzulässigen Parameter-Wert wird **fly_return_2_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Siehe auch Hinweise zu **fly_return**. • Folgende Befehlsaufrufe werden gleichartig ausgeführt: – `fly_return_2_axes( 1, X, 2, Y )` = `fly_return( X, Y )` – `fly_return_2_axes( 4, X, 4, Y )` = `fly_return( X, Y )`
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**fly_return_1_axis**, **fly_return_3_axes**

Variabler Listenbefehl	**fly_return_3_axes**
Funktion	"Fly-Erweiterung"-Befehl: Deaktiviert alle 3 Achsen einer "Processing on the fly"-Anwendung und führt anschließend einen Sprung auf die angegebene neue Ausgabeposition aus.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, führt **fly_return_3_axes** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`fly_return_3_axes( RetPosX, RetPosY, RetPosZ )`
Parameter	`RetPosX` Absolute Achsen-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt. `RetPosY` Wie `RetPosX`. `RetPosZ` Wie `RetPosX`.
Multi-Board-Befehlsname	**n_fly_return_3_axes**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **fly_return_3_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • **fly_return_3_axes** deaktiviert nur dann Achsen einer "Processing on the fly"-Anwendung, wenn diese mit einem "Fly-Erweiterung"-Befehl aktiviert wurden. • Wenn für eine der angegebenen Achsen keine "Processing on the fly"-Funktion aktiviert ist, wird nur der Sprung ausgeführt. • Bei einem unzulässigen Parameter-Wert wird **fly_return_3_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Siehe auch Hinweise zu **fly_return**. • Folgende Befehlsaufrufe werden gleichartig ausgeführt: – `fly_return_3_axes( X, Y, Z )` = `fly_return_z( X, Y, Z )`
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**fly_return_1_axis**, **fly_return_2_axes**

Normaler Listenbefehl	**fly_return_z**
Funktion	Deaktiviert eine zuvor eingestellte “Processing on the fly”-Korrektur. Führt anschließend einen Sprung auf die angegebene neue Ausgabeposition aus.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, führt **fly_return_z** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`fly_return_z( X, Y, Z )`
Parameter	`X` Absolute x-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y` Wie `X` (analog).
	`Z` Wie `X` (analog).
Multi-Board-Befehlsname	**n_fly_return_z**
Hinweise	• Wie **fly_return**, jedoch mit zusätzlicher `Z`-Ausgabeposition.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X`, `Y` und `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_z**

Kontrollbefehl	free_rtc6_dll
Funktion	Gibt alle von der RTC6-DLL für ein Anwenderprogramm belegten Ressourcen frei.
Aufruf	`free_rtc6_dll()`
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zu den von der RTC6-DLL belegten Ressourcen zählt vor allem der in der RTC6-DLL belegte Speicherbereich für die RTC6-Kartenverwaltung (die mit **init_rtc6_dll** aufgebaut wird). Mit **free_rtc6_dll** wird die RTC6-Kartenverwaltung der RTC6-DLL gelöscht. Danach ist vom Anwenderprogramm keine Karte mehr ansprechbar (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`). • Der Aufruf von **free_rtc6_dll** ist nicht zwingend erforderlich, da die von der RTC6-DLL belegten Ressourcen auch automatisch freigegeben werden, wenn das Anwenderprogramm beendet wird und Microsoft Windows dabei die RTC6-DLL entlädt. Einige Entwicklungsumgebungen melden jedoch im Debugmodus “Memory leaks detected!“ noch bevor die RTC6-DLL entladen wird. Der Aufruf von **free_rtc6_dll** beseitigt diese Irritation.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl. Die Funktionalitäten von **free_rtc6_dll** und dem RTC5-Befehl **free_rtc5_dll** sind identisch.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**init_rtc6_dll**, **release_rtc**

Kontrollbefehl	**get_auto_cal**
Funktion	Nur für einige abgekündigte Systeme. Gibt den Typ der im angeschlossenen Scan-System integrierten ASC-Hardware zurück, der zuvor mit **auto_cal** ermittelt wurde.
Aufruf	`ASCtype = get_auto_cal( HeadNo )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf.
Rückgabe	ASC-Hardware-Typ. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_auto_cal**
Hinweise	• Wurde der ASC-Hardware-Typ zuvor mit **auto_cal** ermittelt, so liefert **get_auto_cal** die gleichen Rückgabewerte wie **auto_cal**(`Command` = 4), siehe Hinweise dort. • Falls der ASC-Hardware-Typ *nicht* zuvor mit **auto_cal** ermittelt wurde, so liefert **get_auto_cal** den Wert 255 (Initialisierungswert) zurück.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**auto_cal**

Kontrollbefehl	get_bios_version
Funktion	Gibt die BIOS-Versionsnummer der RTC6-Karte zurück.
Aufruf	`BiosVersion = get_bios_version()`
Rückgabe	BIOS-Versionsnummer in BCD-Codierung. Als 32-Bit-Wert ohne Vorzeichen. Beispiel: `BiosVersion = 33` = 0x21 bedeutet BIOS-Version 21.
Parameter	Keine.
Multi-Board-Befehlsname	**n_get_bios_version**
Hinweise	• **get_bios_version** liefert nur ab BIOS-Version 21 zuverlässige Ergebnisse, sonst 0.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**get_dll_version**, **get_hex_version**, **get_rtc_version**

Kontrollbefehl	get_card_type
Funktion	Gibt den Typ der RTC6-Karte zurück.
Aufruf	`Result = get_card_type()`
Rückgabe	Kartentyp. Als 32-Bit-Wert ohne Vorzeichen. 0: "Keine Karte". 1: RTC6 PCIe-Karte. 2: RTC6 Ethernet-Karte.
Parameter	Keine.
Multi-Board-Befehlsname	**n_get_card_type**
Hinweise	• Siehe Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 606, OUT 606, RBF 611.
Verweise	**eth_assign_card**, **eth_assign_card_ip**

Kontrollbefehl	**get_char_pointer**
Funktion	Gibt die absolute Startadresse eines indizierten Zeichens zurück.
Aufruf	`CharPointer = get_char_pointer( Char )`
Parameter	`Char` Index des indizierten Zeichens. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Rückgabe	Absolute Startadresse. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_char_pointer**
Hinweise	• **get_char_pointer** liest die Startadresse des indizierten Zeichens für den ausgewählten Index aus der internen Verwaltungstabelle. Je nachdem, ob das Zeichen in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” geladen wurde oder ein ungeschütztes Unterprogramm nur nachträglich referenziert wurde, liegt die zurückgelesene Speicheradresse im geschützten oder ungeschützten RTC6-Listenspeicher-Bereich. • Für `Index` > 1023 oder wenn kein Zeichen für den ausgewählten Index referenziert ist, gibt **get_char_pointer** einen Wert von “–1” (d.h. 2^{32}–1) zurück. • **get_char_pointer** ist nützlich zur Überprüfung, ob ein Zeichen bereits definiert wurde oder auch, um das indizierte Zeichen auch wie ein nichtindiziertes Unterprogramm unter Angabe der absoluten Speicheradresse aufrufen zu können, z.B. für eine bedingte Ausführung mit **list_call_cond**. Dabei sollte man allerdings berücksichtigen, dass sich die absolute Speicheradresse bei einem anschließenden **save_disk**/**load_disk** u.U. ändert. Außerdem sollte dann sichergestellt werden, dass **get_char_pointer** nicht “–1” zurückgibt, sonst wird **list_call_cond** ignoriert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_sub_pointer**, **get_text_table_pointer**

Kontrollbefehl	**get_config_list**
Funktion	Gibt die Parameter der aktuellen RTC6-Listenspeicher-Konfiguration (`Mem1`, `Mem2`) an die RTC6-Kartenverwaltung der RTC6-DLL zurück und initialisiert diese so, als wäre **config_list** aufgerufen worden.
Aufruf	`get_config_list()`
Rückgabe	–
Multi-Board-Befehlsname	**n_get_config_list**
Hinweise	• **get_config_list** ist nützlich, wenn eine Karte den "Besitzer" wechselt und die neue RTC6-Kartenverwaltung die Speicherkonfiguration nicht kennt (Karte und Verwaltung initialisieren sich unabhängig voneinander mit jeweils `Mem1` = 4.194.304 und `Mem2` = 4.194.304, die Karte mit **load_program_file** und die Verwaltung beim Start des jeweiligen Anwenderprogramms). Siehe auch Kapitel 6.7.1 "Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm", Seite 131. • **get_config_list** liefert keinen Rückgabewert an das Anwenderprogramm. Die Konfigurationsdaten des RTC6-Listenspeichers (`Mem1` und `Mem2`) können aber vom Anwenderprogramm nach `load_list( ListNo, 0 )` oder `set_start_list_pos( ListNo, 0 )` mit **get_list_space** ermittelt werden. • **get_config_list** wird unabhängig vom BUSY-Listenausführungsstatus der Karte ausgeführt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**config_list**

Kontrollbefehl	**get_counts**
Funktion	Gibt die aktuelle Anzahl erfolgter Externer Starts zurück.
Aufruf	`Counts = get_counts()`
Rückgabe	Anzahl erfolgter Externer Starts. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_counts**
Hinweise	• Die Anzahl wird aus einem internen Zähler ausgelesen, der bei jedem erfolgten Start inkrementiert wird. Dieser Zähler kann mit **set_control_mode** auf 0 zurückgesetzt werden.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_max_counts**, **set_control_mode**, **get_startstop_info**

Kontrollbefehl	**get_dll_version**
Funktion	Gibt die Versionsnummer der RTC6-DLL zurück.
Aufruf	`DLLVersion = get_dll_version()`
Rückgabe	Versionsnummer. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Die RTC6-DLL-Versionsnummern liegen im Bereich 600…699. • **get_dll_version** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 “Fehlerbehandlung (Error-Handling)”, Seite 133, werden von **get_dll_version** weder erzeugt noch verändert.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_hex_version**, **get_rtc_version**

Kontrollbefehl	**get_encoder**
Funktion	Gibt die aktuellen Zählerstände der beiden internen Encoderzähler zurück.
Aufruf	`get_encoder( &Encoder0, &Encoder1 )`
Parameter-Rückgaben	`Encoder0` Aktueller Zählerstand des Encoderzählers “Encoder0”. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen. `Encoder1` Aktueller Zählerstand des Encoderzählers “Encoder1”. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_get_encoder**
Hinweise	• Zur Verwendung von **get_encoder** siehe Kapitel 8.6 ““Processing on the fly””, Seite 254 und Kapitel 9.3.3 “Synchronisation durch Encodersignale”, Seite 323. • Wird die Werkstück-Bewegung mit einem Inkremental-Encoder registriert, so wird getriggert: – Encoderzähler “Encoder0” durch die Signale am Encodereingang ENCODER X – Encoderzähler “Encoder1” durch die Signale am Encodereingang ENCODER Y • Wurde mit **simulate_encoder** eine Encodersimulation gestartet, dann werden die Encoderzähler durch ein internes periodisches 1 MHz-Taktsignal getriggert.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**store_encoder**, **read_encoder**, **set_fly_x**, **set_fly_y**, **set_fly_rot**, **wait_for_encoder**

Kontrollbefehl	get_error
Funktion	Gibt den akkumulierten Fehlercode zurück. Dieser entspricht einer Auflistung der seit dem letzten Reset oder Error-Reset aufgetretenen Fehlertypen.
Aufruf	`AccError = get_error()`
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. Sind mehrere Fehler aufgetreten, so sind mehrere Bits gesetzt. Für die einzelnen Fehler sind die im Folgenden genannten Fehlerkonstanten vorzudefinieren.

Bit	Fehlertyp	Fehlerkonstante	
–	Kein Fehler.	`RTC6_NO_ERROR`	= 0
Bit #0 (LSB)	= 1: Keine RTC6 PCIe-Karte gefunden. Der Fehler kann nur bei **init_rtc6_dll** auftreten.	`RTC6_NO_PCIE_CARD_FOUND`	= 1
Bit #1	= 1: Zugriff verweigert. Der Fehler kann bei **init_rtc6_dll**, **select_rtc**, **acquire_rtc** oder bei allen Multi-Board-Befehlen auftreten.	`RTC6_ACCESS_DENIED`	= 2
Bit #2	= 1: Befehl nicht gesendet. Der Fehler deutet auf einen internen, RTC6-Kartentreiber- oder PCI-Fehler z.B. durch eine defekte Hardware oder einen fehlerhaften Anschluss.	`RTC6_SEND_ERROR`	= 4
Bit #3	= 1: Karte reagiert nicht. Vermutlich ist kein Programm auf die RTC6 geladen. Der Fehler kann bei allen Kontrollbefehlen auftreten, die eine Antwort erwarten, z.B. bei **get_head_para**.	`RTC6_TIMEOUT`	= 8
Bit #4	= 1: Unzulässige Parameter. Der Fehler kann bei allen Befehlen auftreten, bei denen fehlerhafte Parameter nicht automatisch auf einen zulässigen Wert korrigiert werden, z.B. bei Parametern mit begrenzter Auswahl wie **get_head_para**. Tritt der Fehler bei einem Listenbefehl auf, so wird dieser durch **list_nop** ersetzt. Tritt der Fehler bei einem Kontrollbefehl auf, so wird dieser nicht ausgeführt.	`RTC6_PARAM_ERROR`	= 16

Kontrollbefehl	get_error			
Rückgabe (Forts.)	Bit #5	= 1:	Listenausführung ist (nicht) aktiv. Beispiele: bei **execute_list**, wenn noch eine Liste läuft; bei **stop_execution**, wenn gerade keine Liste läuft; bei **restart_list**, wenn nicht zuvor **pause_list** aufgerufen wurde.	RTC6_BUSY = 32
	Bit #6	= 1:	Listenbefehl verweigert, ungültiger Eingabe-Zeiger. Z.B. bei beliebigem Listenbefehl direkt nach **load_char** + **list_return**: der Listenbefehl wird dann nicht gespeichert.	RTC6_REJECTED = 64
	Bit #7	= 1:	Listenbefehl wurde in **list_nop** konvertiert. Z.B. **set_end_of_list** in einem geschützten Unterprogramm.	RTC6_IGNORED = 128
	Bit #8	= 1:	Versions-Fehler: RTC6-DLL-Version, RTC6RBF.rbf-Version und RTC6OUT.out/RTC6ETH.out-Version passen nicht zusammen. Siehe auch **load_program_file**.	RTC6_VERSION_MISMATCH = 256
	Bit #9	= 1:	Verify-Fehler: bei der Download-Überprüfung wurde ein fehlerhafter Download festgestellt. Siehe auch Kapitel 6.8.1 "Download-Überprüfung", Seite 134.	RTC6_VERIFY_ERROR = 512
	Bit #10	= 1:	Es wurde z.B. ein **[*]eth[*]**-Befehl an eine RTC6 PCIe-Karte gesendet.	RTC6_TYPE_REJECTED = 1024
	Bit #11	= 1:	Eine RTC6-DLL-interne Windows-Speicher-Anforderung schlug fehl.	RTC6_OUT_OF_MEMORY = 2048
	Bit #12	= 1:	Downloadfehler. Die Werte sind möglicherweise nicht gespeichert. Kann bei **auto_cal** und **write_hi_pos** auftreten.	RTC6_FLASH_ERROR = 4096

Kontrollbefehl	get_error			
Rückgabe (Forts.)	Bit #13	= 1:	Allgemeiner Ethernet-Fehler. Kann z.B. beim Versuch auftreten, mit **select_rtc** oder **acquire_rtc** von RTC6 PCIe-Karte auf RTC6 Ethernet-Karte zu wechseln. Weitere Informationen liefert dann **n_eth_get_error** auf der angesprochenen RTC6 Ethernet-Karte.	RTC6_ETH_ERROR = 8192
	Bit #14	Reserviert.		–
	Bit #15	= 1:	Nicht unterstützte Windows Version. Kann nur bei **init_rtc6_dll** auftreten.	RTC6_UNSUPPORTED_WINDOWS = 32768
	Bit #16	Reserviert.		–
	Bit #17	= 1:	Nicht unterstützte BIOS Version.	RTC6_UNSUPPORTED_BIOS = 131072
	Bit #18	Reserviert.		–
	…	…		
	Bit #30	Reserviert.		
	Bit #31	= 1:	Kann ausschließlich im Remote-Interface-Modus auftreten: Befehls-Telegramm ungültig. Siehe Abschnitt "Fehlercodes für LastError, die nur im Remote-Interface-Modus vorkommen", Seite 1028.	RTC6_REMOTE_ERROR = 2147483648
Multi-Board-Befehlsname	**n_get_error**			
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133. • **get_error** und **n_get_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Die kartenspezifischen Fehlervariablen LastError und AccError (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **get_error** weder erzeugt noch verändert. • Bit #0 = 1 (Fehlerkonstante: RTC6_NO_PCIE_CARD_FOUND = 1) bedeutet, dass keine RTC6 PCIe-Karte gefunden wurde (**init_rtc6_dll** sucht nicht nach RTC6 Ethernet-Karten).			

Kontrollbefehl	get_error

Beispiel (C/C++)	Erstellung eines Arrays mit der Info, für welche Karte keine Zugriffsberechtigung vorliegt und Rücksetzen des akkumulierten Fehlercodes.
	`UINT NoAccess[MaxAnzahl+1]; // MaxAnzahl ist eine Benutzer-definierte Konstante`
	`UINT Error = init_rtc6_dll(); // Suche alle installierten RTC6-Karten`
	`if (Error & RTC6_ACCESS_DENIED) { // mindestens eine Karte ist gesperrt`
	`  UINT Anzahl = rtc6_count_cards(); // Anzahl Karten gefunden`
	`  for ( UINT Nr = 1; Nr <= Anzahl; Nr++ ) {`
	`  NoAccess[Nr] = n_get_last_error(Nr) & RTC6_ACCESS_DENIED;`
	`  n_reset_error(Nr, RTC6_ACCESS_DENIED);`
	`  }`
	`}`

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_last_error**, **reset_error**, **set_verify**

Kontrollbefehl	get_extended_head_status_block_ctrl	
Funktion	Relevant für Erweiterter Scan-Kopf-Status: Schreibt alle möglichen Erweiterter Scan-Kopf-Status-Signal-Werte einer Achse in das Array `Ptr`.	
Aufruf	`get_extended_head_status_block_ctrl( Head, Axis, Ptr )`	
Parameter	`Head`	Nummer des Scan-Kopf-Anschlusses der RTC-Karte. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Scan-System am 1. Scan-Kopf-Anschluss. = 2: Scan-System am 2. Scan-Kopf-Anschluss.
	`Axis`	Nummer der Achse. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: x-Achse (STATUS-Kanal, Galvanometerscanner 2). = 2: y-Achse (STATUS1-Kanal, Galvanometerscanner 1).
	`Ptr`	Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von 256 32-Bit-Werten mit Vorzeichen, in dem die aktuellen Werte abgelegt werden sollen.
Rückgabe	Keine.	
Multi-Board-Befehlsname	**n_get_extended_head_status_block_ctrl**	
Hinweise	• –	

Beispiel (C/C++)	`// Mit get_extended_head_status_block_ctrl`
	`// (alternativ zu get_extended_head_status_value_ctrl)`
	`// Apertur und Wellenlänge des iDRIVE-Scan-Systems abfragen`
	
	`UINT Buffer[256];`
	`get_extended_head_status_block_ctrl( 1, 1, (ULONG_PTR) Ptr );`
	`UINT Aperture = Ptr[0x0D];`
	`UINT Wavelength = Ptr[0x0E];`	

RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 647, OUT 650.	
Verweise	**get_extended_head_status_value_ctrl**	

Kontrollbefehl	get_extended_head_status_value_ctrl	
Funktion	Relevant für Erweiterter Scan-Kopf-Status: Gibt nur einen einzelnen Erweiterter Scan-Kopf-Status-Signal-Wert einer Achse zurück.	
Aufruf	`ReturnValue = get_extended_head_status_value_ctrl( Head, Axis, Value )`	
Parameter	`Head`	Nummer des Scan-Kopf-Anschlusses der RTC-Karte. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Scan-System am 1. Scan-Kopf-Anschluss. = 2: Scan-System am 2. Scan-Kopf-Anschluss.
	`Axis`	Nummer der Achse. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: x-Achse (STATUS-Kanal, Galvanometerscanner 2). = 2: y-Achse (STATUS1-Kanal, Galvanometerscanner 1).
	`Value`	ID des Erweiterter Scan-Kopf-Status-Signals, siehe Kapitel 21.2.1 "Erweiterter Scan-Kopf-Status-Signal-IDs", Seite 1135. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Erweiterter Scan-Kopf-Status-Wert. Als 32-Bit-Wert ohne Vorzeichen.	
Multi-Board-Befehlsname	**n_get_extended_head_status_value_ctrl**	
Hinweise	• –	

Beispiel (C/C++)	`// Mit get_extended_head_status_value_ctrl`
	`// (alternativ zu get_extended_head_status_block_ctrl)`
	`// Apertur und Wellenlänge des iDRIVE-Scan-Systems abfragen`
	
	`UINT Aperture = get_extended_head_status_value_ctrl( 1, 1, 0x0D );`
	`UINT Wavelength = get_extended_head_status_value_ctrl( 1, 1, 0x0E );`	

RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 647, OUT 650.	
Verweise	**get_extended_head_status_block_ctrl**	

Kontrollbefehl	**get_fly_2d_offset**
Funktion	Gibt die aktuellen Referenzwerte (Offsetwerte) für die 2D-Encoderkompensation zurück.
Aufruf	`get_fly_2d_offset( &OffsetX, &OffsetY )`
Parameter-Rückgaben	`OffsetX` x-Referenzwert. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen. `OffsetY` y-Referenzwert. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_get_fly_2d_offset**
Hinweise	• Zur 2D-Encoderkompensation siehe Abschnitt "2D-Encoderkompensation für xy-Verfahrtische", Seite 261.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**init_fly_2d**, **set_fly_2d**

Kontrollbefehl	**get_free_variable**
Funktion	Gibt den aktuellen Wert einer freien Variablen zurück.
Aufruf	`VariableValue = get_free_variable( No )`
Parameter	`No` Nummer derjenigen freien Variablen, deren Wert gelesen werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…7]. Es werden nur die 3 niederwertigsten Bits ausgewertet.
Rückgabe	Der aktuell in der freien Variablen `No` gespeicherte Wert. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_free_variable**
Hinweise	• Siehe auch Kapitel 6.9.1 "Freie Variablen", Seite 138.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_free_variable**, **set_free_variable_list**

Kontrollbefehl	**get_galvo_controls**
Funktion	Gibt die zu den Eingabewerten gehörigen Ansteuerwerte zurück.
Einschränkung	**get_galvo_controls** kann nur ausgeführt werden, wenn gerade keine Liste läuft (**get_last_error**-Returncode `RTC6_BUSY`).
Aufruf	`get_galvo_controls( InPtr, OutPtr )`
Parameter	`InPtr` Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von fünf 32-Bit-Werten ohne Vorzeichen, in dem die auszugebenden Einstellungen angegeben sind: X, Y, Z, Defocus, Zoom. Überlaufende Werte werden auf die Randwerte geclippt. `OutPtr` Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von 4 32-Bit-Werten ohne Vorzeichen, in dem die zugehörigen Ansteuerwerte abgelegt werden sollen: XA, YA, XB, YB. Wertebereiche jeweils [–524.288…+524.287].
Parameter-Rückgaben	XA, YA, XB, YB bezeichnen die Ansteuerwerte der x-Achse/y-Achse für Scan-Kopf A/B.
Multi-Board-Befehlsname	**n_get_galvo_controls**
Hinweise	• **get_galvo_controls** führt einen virtuellen Sprung auf die angegebenen Koordinaten aus. Die Galvanometerscanner werden dabei aber *nicht* bewegt. • Alle sonstigen Einstellungen, die nicht direkt in **get_galvo_controls** angegeben sind (z.B. Matrix, Offset, Angle, etc.; auch Stretch-Table) werden so berücksichtigt, wie sie gerade eingestellt sind. Stellen Sie sicher, diese vor Aufruf von **get_galvo_controls** mit `at_once > 0` anzuwenden! Bei lediglichem Speichern mit `at_once = 0` werden die Einstellungen nicht verwendet. • Bei der Berechnung der Ansteuerwerte werden nur diejenigen Kanäle berücksichtigt, für die auch Korrekturtabellen mit **select_cor_table** zugewiesen wurden, siehe folgende Beispiele. `select_cor_table(0,0)`: XA = YA = XB = YB = 0 2D-Korrekturdateien: die Inputs Z, Defocus, Zoom werden ignoriert `select_cor_table(1,0)`: XA, YA berechnet, XB, YB = 0 `select_cor_table(0,1)`: XA, YA = 0, XB, YB berechnet `select_cor_table(1,1)`: XA, YA, XB, YB berechnet (Standard-)3D-Korrekturdatei: der Input Zoom wird ignoriert `select_cor_table(1,0)`: XA, YA berechnet, XB = YB = Zout berechnet `select_cor_table(0,1)`: XA= YA = Zout berechnet, XB, YB berechnet `select_cor_table(1,1)`: wie `select_cor_table(0,0)` 3D-Zoom-Korrekturdatei (nur für intelli*WELD* II mit Zoom-Achse): `select_cor_table(1,0)`: XA, YA berechnet, XB = Zout, YB = ZoomOut berechnet `select_cor_table(0,1)`: XA= Zout, YA = ZoomOut berechnet, XB, YB berechnet • Die Rückgabewerte sind 0, falls ein **get_last_error**-Returncode `RTC6_BUSY` erzeugt wurde.

Kontrollbefehl	get_galvo_controls
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für x, y, z und Defocus mit 16. Entsprechend verringert sich der zulässige Wertebereich. Alle zurückgegebenen Werte sind auch im RTC4-Kompatibilitätsmodus im RTC6-20-Bit-Bereich.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für z und Defocus mit 16. Entsprechend verringert sich der zulässige Wertebereich. Alle zurückgegebenen Werte sind auch im RTC5-Kompatibilitätsmodus im RTC6-20-Bit-Bereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_cor_table**

Kontrollbefehl	get_head_para
Funktion	Gibt aus der Korrekturtabelle, die dem angegebenen Kopf zugewiesen ist, den gewünschten Parameterwert zurück.
Aufruf	`HeadPara = get_head_para( HeadNo, ParaNo )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf.
	`ParaNo` Nummer des Parameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: 0…15. Zuordnung: siehe Abschnitt "ct5-Korrekturdateikopf", Seite 186.
Rückgabe	Parameterwert, siehe Abschnitt "ct5-Korrekturdateikopf", Seite 186. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_get_head_para**
Hinweise	• Die Parameterwerte können mit **get_table_para** aus einer aktuell geladenen und mit **get_head_para** aus einer zugewiesenen RTC6-Korrekturtabelle ausgelesen und auf diese Weise direkt in ein Anwenderprogramm eingebunden werden, siehe Abschnitt "ct5-Korrekturdateikopf", Seite 186. • Für unzulässige Parameterwerte für `HeadNo` und `ParaNo` ist der Rückgabewert 0 (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). Der Rückgabewert ist auch 0 (kein **get_last_error**-Returncode), wenn dem angegebenen Kopf keine Korrekturtabelle zugewiesen ist (z.B. für `HeadNo` = 2, falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist) und dem anderen Kopf keine 3D-Korrekturtabelle zugewiesen ist. • Ist einem Kopf eine 3D-Korrekturtabelle zugewiesen, werden unabhängig von `HeadNo` die Parameter dieser 3D-Korrekturtabelle zurückgegeben (zwei 3D-Korrekturtabellen können nicht gleichzeitig zugewiesen sein). `HeadNo` muss dennoch 1 oder 2 sein (siehe vorheriger Hinweis).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_table_para**

Kontrollbefehl	**get_head_status**
Funktion	Gibt das XY2-100-Statuswort des angegebenen Scan-Kopf-Anschlusses zurück.
Aufruf	`get_head_status( Head )`
Parameter	`Head` = 1: Gibt den Status zurück von Anschluss für ersten Scan-Kopf (Byte #1 = Byte #0). = 2: Gibt den Status zurück von Anschluss für zweiten Scan-Kopf (Byte #1 = Byte #0). Sonst: Gibt den Status zurück von Anschluss für ersten Scan-Kopf (Byte #0) *und* Anschluss für zweiten Scan-Kopf (Byte #1).
Rückgabe	XY2-100-Statuswort. Als 32-Bit-Wert ohne Vorzeichen. Byte #0 (LSB) Bit #0 (LSB) 1. Bit #1 0. Bit #2 1 (reserviert). Bit #3 Positionsrückmeldung der x-Achse, 1 = OK. Bit #4 Positionsrückmeldung der y-Achse, 1 = OK. Bit #5 1 (reserviert). Bit #6 Temperatur-Status, 1 = OK. Siehe Hinweis, Seite 450. Bit #7 Power-Status, 1 = OK. Siehe Hinweis, Seite 450. Byte #1 Bit #8 Bit-Zuordnung wie bei Byte #0. … Bit #15 Byte #2 Bit #16 0. … … Byte #3 Bit #31
Multi-Board-Befehlsname	**n_get_head_status**
Hinweise	• **get_head_status** ist mit dem SL2-100-Protokoll auch dann verfügbar, wenn als Datentyp nicht das 20-Bit-Statuswort selbst eingestellt ist, siehe Abschnitt "Statusinformationen vom Scan-System", Seite 191. • Die Statusbits werden von **get_head_status** über Bit #3…Bit #7 und/oder Bit #11…Bit #15 zurückgegeben. Bit #0 und Bit #8 werden von **get_head_status** unabhängig vom aktuellen Zustand des Scan-Systems als 1, sowie Bit #1 und Bit #9 als 0 zurückgegeben. • Ist aktuell kein Scan-System angeschlossen oder nicht eingeschaltet, wird der Wert 0 zurückgeliefert. • Das PowerOK-Signal ist ein elektronisch generiertes Signal. Es bedeutet: Die Scan-System-Regelung ist bereit für Eingabedaten ("PowerOK", entspricht eigentlich einem "ServoOK"). Es kann daher nicht dafür eingesetzt werden, um zu prüfen, ob ein angeschlossener Scan-Kopf überhaupt eingeschaltet ist. Zur Unterscheidung kann **get_startstop_info** (Bit #17 und/oder 25) verwendet werden.

Kontrollbefehl	get_head_status
Hinweise (Forts.)	• Power-Status und Temperatur-Status sind für beide Achsen verknüpfte Signale. • Beachten Sie auf jeden Fall auch die Hinweise zu den Statussignalen im Handbuch Ihres Scan-Systems. • Bei iDRIVE-Scan-Systemen – ohne SL2-100-Schnittstelle liefert **get_head_status** nur dann brauchbare Rückgabewerte, wenn das XY2-100-Statuswort zur Übertragung ausgewählt wurde. – werden die Positionsrückmeldesignale der x- und y-Achse UND-verknüpft und als gemeinsames Signal (z.B. Bit #3,4) zurückgegeben. – kann es nach einem Neustart oder Reset des Scan-Systems ca. 5 Sekunden dauern, bis Daten vom Scan-System übertragen werden. • Statussignale können auch mit ausgelesen werden mit: – **get_value** – **get_values** – **set_trigger[*]** • Siehe auch Kapitel 8.5 "2D-Scan-Systeme und 3D Scan-Systeme steuern", Seite 248 zur Verwendung zweier Scan-Köpfe.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: • Bei der RTC6 können gleichzeitig die Statusworte beider Scan-Kopf-Anschlüsse ausgelesen werden und bei iDRIVE-Scan-Systemen mit SL2-100-Schnittstelle auch unabhängig von einem (mit **control_command** eingestellten) übertragenen Signal. • Bei iDRIVE-Scan-Systemen liefern Bit #0 und Bit #1 keine Informationen über die Betriebsbereitschaft des Scan-Systems (siehe **get_value**).
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_value**, **get_values**, **set_trigger**, **set_trigger4**, **set_trigger8**, **get_waveform**

Kontrollbefehl	**get_hex_version**
Funktion	Gibt die Versionsnummer der aktuell auf die RTC6 geladenen DSP-Programmdatei `RTC6OUT.out` zurück.
Aufruf	`HexVersion = get_hex_version()`
Rückgabe	Versionsnummer. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_hex_version**
Hinweise	• Die Versionsnummern der Programmdateien liegen im Bereich 600…699. **get_hex_version** gibt die folgenden Werte zurück: – ohne freigeschaltete Option "3D" Werte im Bereich 2.600…2.699 (Versionsnummer + 2.000) – bei freigeschalteter Option "3D" Werte im Bereich 3.600…3.699 (Versionsnummer + 3.000) • Die Dateinamenerweiterung für RTC6-DSP-Programmdateien ist `*.out`. Siehe auch **load_program_file**. • Die Software-Versionsnummer kann auch nach einem `RTC6_VERSION_MISMATCH` und `RTC6_ACCESS_DENIED`-Fehler zurückgegeben werden. Der Rückgabewert ist 0, wenn noch kein Programm geladen ist. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133, werden von **get_hex_version** weder erzeugt noch verändert.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_dll_version**, **get_rtc_version**

Kontrollbefehl	**get_hi_data**	
Funktion	Nur für einige abgekündigte Systeme. Gibt die jeweils zuletzt gemessenen Home-In-Positionen zurück, die zuvor mit **auto_cal** für das am Anschluss für ersten Scan-Kopf angeschlossene Scan-System ermittelt wurden.	
Aufruf	`get_hi_data( &X1, &X2, &Y1, &Y2)`	
Parameter-Rückgaben	`X1`	x1-Koordinate der aktuell gespeicherten (zuletzt gemessenen) Home-In-Positionen. In Bits. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen.
	`X2`	Wie `X1` (analog).
	`Y1`	Wie `X1` (analog).
	`Y2`	Wie `X1` (analog).
Multi-Board-Befehlsname	**n_get_hi_data**	
Hinweise	• **get_hi_data** ist gleichbedeutend mit **get_hi_pos** mit `HeadNo` = 1 (siehe Hinweise dort).	
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Die zurückgegebenen Werte liegen im 20-Bit-Wertebereich.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**get_hi_pos**, **write_hi_pos**	

Kontrollbefehl	get_hi_pos	
Funktion	Nur für einige abgekündigte Systeme. Gibt die jeweils zuletzt gemessenen Home-In-Positionen zurück, die zuvor mit **auto_cal** für das am angegebenen Scan-Kopf-Anschluss angeschlossene Scan-System ermittelt wurden.	
Aufruf	`get_hi_pos( HeadNo, &X1, &X2, &Y1, &Y2 )`	
Parameter	`HeadNo`	Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf.
Parameter-Rückgaben	`X1`	x1-Koordinate der aktuell gespeicherten (zuletzt gemessenen) Home-In-Positionen. In Bits. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen.
	`X2`	Wie `X1` (analog).
	`Y1`	Wie `X1` (analog).
	`Y2`	Wie `X1` (analog).
Multi-Board-Befehlsname	**n_get_hi_pos**	
Hinweise	• Zur Verwendung von **get_hi_pos** siehe Abschnitt "Kundeneigene Kalibrierung", Seite 290. • Stellen Sie sicher, dass das aktuell angeschlossene Scan-System mit demjenigen übereinstimmt, mit dem die ausgelesenen Werte auch bestimmt worden sind. Dieses sollte zur Bestimmung der Werte mit einem internen Sensorsystem zur automatischen Selbstkalibrierung (Home-In-Sensoren) ausgestattet sein. • Die zurückgegebenen Werte sind 0, wenn: – am angegebenen Scan-Kopf-Anschluss kein Scan-System angeschlossen ist, das mit Sensoren zur automatischen Selbstkalibrierung (Home-In-Sensoren) ausgestattet ist – bei der Ermittlung der Werte für ein solches System ein Fehler aufgetreten ist • Unmittelbar nach der Initialisierung (**init_rtc6_dll**), insbesondere bevor **auto_cal**(`Command` = 0, 1 oder 3) erstmals aufgerufen wurde, bedeuten die zurückgegebenen Werte die im Flash-Speicher gespeicherten Referenzwerte für die Home-In-Positionen. Sind diese noch nicht mindestens einmal mit **auto_cal**(`Command` = 0) erfolgreich bestimmt worden, gibt **get_hi_pos** 0 zurück. • **get_hi_pos** ist auch ohne (momentane) explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. Die Rückgabewerte sind jedoch 0, solange nicht zuvor zumindest einmal ein Zugriff auf die angesprochene Karte erfolgreich war. • Für unzulässige Parameterwerte wird für alle Koordinatenwerte 0 zurückgeliefert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**get_hi_data**, **auto_cal**, **set_hi**, **write_hi_pos**	

Kontrollbefehl	get_input_pointer
Funktion	Gibt die aktuelle absolute Position des Eingabe-Zeigers zurück.
Aufruf	`InputPointer = get_input_pointer()`
Rückgabe	Position des Eingabe-Zeigers [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_input_pointer**
Hinweise	• Die Position des Eingabe-Zeigers entspricht der Position im RTC6-Listenspeicher (auch im geschützten RTC6-Listenspeicher-Bereich "Liste 3"), an die der nächste Listenbefehl gespeichert wird. Die Zahl der noch freien Speicherplätze in diesem Bereich kann mit **get_list_space** ermittelt werden. • **get_input_pointer** liefert die absolute RTC6-Listenspeicheradresse (Offset bezüglich Listenanfang von "Liste 1") zurück. Die relative Position bezüglich des Listenanfangs des jeweiligen Listen-Bereichs kann mit **get_list_pointer** ausgelesen werden. • Vor dem Laden eines nichtindizierten Unterprogramms oder Zeichensatzes sollte dessen Startadresse mit **get_input_pointer** ausgelesen werden, wenn anschließend eine Referenzierung mit **set_sub_pointer** oder **set_char_pointer** durchgeführt werden soll. • Die Position des Ausgabe-Zeigers kann mit **get_status** oder **get_out_pointer** ausgelesen werden. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133, werden von **get_input_pointer** weder erzeugt noch verändert.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_list_pointer**, **set_input_pointer**, **get_list_space**, **get_status**, **get_out_pointer**

Kontrollbefehl	get_io_status
Funktion	Gibt den aktuellen Zustand des 16-Bit-Digital-Ausgangs der EXTENSION 1-Stiftleiste zurück.
Aufruf	`IOStatus = get_io_status()`
Rückgabe	16-Bit-Wert (DIGITAL OUT0…DIGITAL OUT15). Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_io_status**
Hinweise	• **get_io_status** ist zur Verwendung in Kombination mit **set_io_cond_list** und **clear_io_cond_list** vorgesehen. Siehe auch Abschnitt "Beispiel-Code (Pascal)", Seite 321. • Siehe auch Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_io_port**, **write_io_port_mask**, **set_io_cond_list**, **clear_io_cond_list**

Kontrollbefehl	get_jump_table
Funktion	Liest die aktuell auf der Karte gespeicherte Jump-Delay-Tabelle aus. Kopiert dann die 1024 zugehörigen 16-Bit-Werte ohne Vorzeichen an die angegebene PC-Adresse.
Aufruf	`ErrorCode = get_jump_table( Addr )`
Parameter	`Addr` PC-Adresse für den 2048 Byte großen Speicherbereich.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Kein Fehler. 11 RTC6-Kartentreiber-Fehler.
Multi-Board-Befehlsname	**n_get_jump_table**
Hinweise	• **get_jump_table** sollte nach Möglichkeit nicht während der Ausführung einer Liste ausgeführt werden. • Das Datenformat ist "1024 16-Bit-Werte", welche die Delay-Werte für eine stückweise lineare Interpolation an den Stützstellen (=Sprunglängen) $N \times 1024$ mit $0 \leq N < 1024$ bedeuten. Die Werte können auch von Benutzern direkt so erzeugt oder manipuliert werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_jump_table**, **load_jump_table_offset**

Kontrollbefehl	get_lap_time
Funktion	Gibt den *aktuellen* Wert des RTC6-Timers zurück (ohne ihn auf Null zurückzusetzen).
Aufruf	`TimerValue = get_lap_time()`
Rückgabe	Wert des RTC6-Timers seit dem letzten Aufruf von **save_and_restart_timer**. In Sekunden. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_get_lap_time**
Hinweise	• **get_lap_time** dient dazu, die bereits verstrichene Zeit einer länger dauernden Markierung noch während der Laufzeit abzufragen.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_time**, **save_and_restart_timer**

Kontrollbefehl	get_laser_pin_in
Funktion	Gibt den aktuellen Zustand des 2-Bit-Digital-Eingangs an der LASER-Buchse zurück, siehe auch Abschnitt "2-Bit-Digital-Eingang", Seite 77.
Aufruf	`LaserPinIn = get_laser_pin_in()`
Rückgabe	Als 32-Bit-Wert ohne Vorzeichen. Bit #0 DIGITAL IN1. (LSB) Bit #1 DIGITAL IN2. Bit #2 Reserviert. … Bit #31 Reserviert.
Multi-Board-Befehlsname	**n_get_laser_pin_in**
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_pin_out**

Kontrollbefehl	get_last_error
Funktion	Gibt einen Fehlercode als Auflistung der beim zuletzt ausgeführten Befehl aufgetretenen Fehler zurück.
Aufruf	`LastError = get_last_error()`
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. Sind mehrere Fehler gleichzeitig aufgetreten, so sind mehrere Bits gesetzt. Die Zuordnung zwischen Bitnummern, Fehlertypen und Fehlerkonstanten ist identisch zu **get_error**.
Multi-Board-Befehlsname	**n_get_last_error**
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133. • **get_last_error** und **n_get_last_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133, werden von **get_last_error** weder erzeugt noch verändert. • Jeder **eth_get_last_error**-Fehler führt auch zu einem **get_last_error**-Fehler `RTC6_ETH_ERROR`.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**eth_get_last_error**, **get_error**, **reset_error**, **set_verify**

Kontrollbefehl	**get_list_pointer**	
Funktion	Gibt die aktuelle *relative* Position des Eingabe-Zeigers sowie die Listennummer zurück.	
Aufruf	`get_list_pointer( &ListNo, &Pos )`	
Parameter-Rückgaben	`ListNo`	Listennummer der Liste, in dem sich der Eingabe-Zeiger aktuell befindet. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. [1…3].
	`Pos`	Aktuelle Position des Eingabe-Zeigers (Offset bezüglich des Listenanfangs der jeweiligen Liste). Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_list_pointer**	
Hinweise	• Die absolute RTC6-Listenspeicheradresse (Offset bezüglich Listenanfang von “Liste 1”) des Eingabe-Zeigers kann mit **get_input_pointer** ausgelesen werden (siehe auch Hinweise dort). • Die Zahl der Listenplätze bis zum Ende der jeweiligen Liste (ab dem Eingabe-Zeiger) kann mit **get_list_space** ausgelesen werden. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 “Fehlerbehandlung (Error-Handling)”, Seite 133, werden von **get_list_pointer** weder erzeugt noch verändert. • Falls der Eingabe-Zeiger beim **get_list_pointer**-Aufruf ungültig ist (z.B. nach einem **list_return**), sind die Rückgabewerte `ListNo` = 0 und `Pos` = 0xFFFFFFFF.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**get_input_pointer**, **get_list_space**	

Kontrollbefehl	get_list_serial
Funktion	Gibt die Nummer des zuletzt mit **select_serial_set_list** ausgewählten Seriennummernsatzes (oder nach **load_program_file** des Seriennummernsatzes 0) und die aktuelle Seriennummer dieses Seriennummernsatzes zurück.
Aufruf	`LastMarkedSerialNo = get_list_serial( &Set )`
Rückgabe	Seriennummer. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Parameter-Rückgaben	`Set` Nummer des ausgewählten Seriennummernsatzes. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_list_serial**
Hinweise	• In der Regel ist die mit **get_list_serial** ausgelesene Seriennummer auch die zuletzt mit **mark_serial** oder **mark_serial_abs** markierte Seriennummer. • Zur Verwendung von **get_list_serial** siehe Kapitel 7.5.2 "Seriennummern markieren", Seite 222.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_serial_set_list**, **get_serial**

Kontrollbefehl	get_list_space
Funktion	Gibt den freien Listen-Speicher, d.h. die Anzahl der noch ladbaren Listenbefehle von der aktuellen Position des Eingabe-Zeigers ab bis zur letzten Listenposition der jeweiligen Liste.
Aufruf	`ListSpace = get_list_space()`
Rückgabe	Zahl der freien Listenplätze. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_list_space**
Hinweise	• Wenn gerade ein indiziertes Unterprogramm oder ein indizierter Zeichensatz in den geschützten RTC6-Listenspeicher-Bereich "Liste 3" geladen wird, liefert **get_list_space** den noch verfügbaren, geschützten Speicherplatz zurück (anderenfalls befindet sich der Eingabe-Zeiger nicht im geschützten Bereich).
RTC4→RTC6	**get_list_space** wurde bei der RTC4 für den RTC4-Ringpuffer-Modus zur Verfügung gestellt und liefert dort den Abstand zwischen Eingabe-Zeiger und Ausgabe-Zeiger. Der RTC4-Ringpuffer-Modus wird nicht mehr von der RTC6 unterstützt. Die Position des Eingabe-Zeigers kann mit **get_input_pointer** oder **get_list_pointer** und die Position des Ausgabe-Zeigers mit **get_status** oder **get_out_pointer** ausgelesen werden.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_input_pointer**, **get_status**, **get_out_pointer**, **get_list_pointer**

Kontrollbefehl	get_marking_info
Funktion	Gibt Informationen über eine mögliche Bereichsüberschreitung während einer “Processing on the fly”-Korrektur und über fehlerhafte Encodersignale zurück. **get_marking_info** liefert außerdem die Fehler-Bits der automatischen Lasersteuersignal-Unterdrückung zurück.
Aufruf	`MarkingInfo = get_marking_info()`
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. Bit #0 = 1: “Processing on the fly”-Unterschreitung in x-Richtung (x < –524.288). (LSB) Bit #1 = 1: “Processing on the fly”-Überschreitung in x-Richtung (x > +524.287). Bit #2 = 1: “Processing on the fly”-Unterschreitung in y-Richtung (y < –524.288). Bit #3 = 1: “Processing on the fly”-Überschreitung in y-Richtung (y > +524.287). Bit #4 = 1: “Processing on the fly”-Unterschreitung in x-Richtung (x < Xmin). Bit #5 = 1: “Processing on the fly”-Überschreitung in x-Richtung (x > Xmax). Bit #6 = 1: “Processing on the fly”-Unterschreitung in y-Richtung (y < Ymin). Bit #7 = 1: “Processing on the fly”-Überschreitung in y-Richtung (y > Ymax). Bit #8 = 1: TriggerError: während der Abarbeitung einer Liste wurde ein externer oder simulierter Start-Trigger gesetzt. Bit #9 = 1: Beim Aktivieren einer “Processing on the fly”-Korrektur mit **activate_fly_2d**, **activate_fly_2d_encoder**, **activate_fly_xy** oder **activate_fly_xy_encoder** ist ein Fehler aufgetreten (“ActivateFlyError”). Siehe auch Kapitel 8.6.7 ““Processing on the fly”-Anwendungen synchronisieren”, Seite 264, Kapitel 8.6.8 “Encoderresets”, Seite 266 und Kapitel 8.6.9 ““Processing on the fly”-Korrektur überwachen”, Seite 267. Bit #10 PosAck-Fehler-Bit von Scan-System A, x-Achse. Bit #11 TempOK-Fehler-Bit von Scan-System A, x-Achse. Bit #12 PowerOK-Fehler-Bit von Scan-System A, x-Achse. Bit #13 PosAck-Fehler-Bit von Scan-System A, y-Achse. Bit #14 TempOK-Fehler-Bit von Scan-System A, y-Achse. Bit #15 PowerOK-Fehler-Bit von Scan-System A, y-Achse. Bit #16 = 1: Der Abstand der Flanken im Signal 1 des Encodereingangs ENCODER X ist zu kurz. Bit #17 = 1: Der Abstand der Flanken im Signal 1 des Encodereingangs ENCODER Y ist zu kurz. Bit #18 = 1: Der Abstand der Flanken im Signal 2 des Encodereingangs ENCODER X ist zu kurz. Bit #19 = 1: Der Abstand der Flanken im Signal 2 des Encodereingangs ENCODER Y ist zu kurz.

Kontrollbefehl	get_marking_info
Rückgabe (Forts.)	Bit #20 = 1: Falsche Signalsequenz am Encodereingang ENCODER X. Bit #21 = 1: Falsche Signalsequenz am Encodereingang ENCODER Y. Bit #22 = 1: "Processing on the fly"-Unterschreitung in z-Richtung (z < –524.288). Bit #23 = 1: "Processing on the fly"-Überschreitung in z-Richtung (z > +524.287). Bit #24 = 1: "Processing on the fly"-Unterschreitung in z-Richtung (z < Zmin). Bit #25 = 1: "Processing on the fly"-Überschreitung in z-Richtung (z > Zmax). Bit #26 PosAck-Fehler-Bit von Scan-System B, x-Achse. Bit #27 TempOK-Fehler-Bit von Scan-System B, x-Achse. Bit #28 PowerOK-Fehler-Bit von Scan-System B, x-Achse. Bit #29 PosAck-Fehler-Bit von Scan-System B, y-Achse. Bit #30 TempOK-Fehler-Bit von Scan-System B, y-Achse. Bit #31 PowerOK-Fehler-Bit von Scan-System B, y-Achse.
Multi-Board-Befehlsname	**n_get_marking_info**
Hinweise	• Zur Verwendung von **get_marking_info** und der Fehler-Bits Bit #0…Bit #7 siehe Kapitel 8.6.9 ""Processing on the fly"-Korrektur überwachen", Seite 267. • Die Grenzen für den kundenspezifischen Überwachungsbereich werden festgelegt für: – Xmin, Xmax, Ymin, Ymax (Bit #4…Bit #7) durch **set_fly_limits** – Zmin, Zmax (Bit #24…Bit #25) durch **set_fly_limits_z** • Die Fehler-Bits Bit #4…Bit #7 und Bit #24…Bit #25: – Werden *nicht* durch **get_marking_info** zurückgesetzt – Werden implizit durch die bedingten Befehle zurückgesetzt – Können auch explizit mit **clear_fly_overflow** und **clear_fly_overflow_ctrl** zurückgesetzt werden • Der Abstand der Encodersignale kann zu kurz sein, wenn ein Störsignal auftritt, ein schneller Richtungswechsel erfolgt oder die Frequenz insgesamt zu hoch ist. • Eine falsche Encodersignalsequenz liegt vor, wenn sich die beiden Signale 1 und 2 gleichzeitig ändern. Dann lässt sich die Zählrichtung nicht mehr detektieren. • Die Fehler-Bits Bit #10…Bit #15 und Bit #26…Bit #31 werden nur bei aktivierter automatischer Lasersteuersignal-Unterdrückung im Fehlerfall gesetzt, siehe Abschnitt "Automatische Unterdrückung der Lasersteuersignale", Seite 195. Bei auftretenden Fehlern werden immer diejenigen Fehler-Bits (kumulativ) gesetzt, für die das zugehörige Statussignal einen Fehler meldet. Gegebenenfalls werden auch Fehler-Bits gesetzt, deren Statussignale nicht mit **set_laser_control** für die automatische Unterdrückung der Lasersteuersignale ausgewählt wurden. Alle Fehler-Bits werden beim Auslesen mit **get_marking_info** zurückgesetzt. • Alle Fehler-Bits werden bei der Initialisierung (mit **load_program_file**) zurückgesetzt.

Kontrollbefehl	get_marking_info
Hinweise (Forts.)	• Alle Fehler-Bits (außer Bit #4…Bit #7 und Bit #24…Bit #25): – werden beim Auslesen mit **get_marking_info** zurückgesetzt – können jedoch jederzeit neu gesetzt werden, solange die Fehlerbedingung noch besteht
RTC4→RTC6	Unveränderte Funktionalität für bereits bei der RTC4 verwendete Info-Bits.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_x**, **set_fly_y**, **set_fly_z**, **set_fly_rot**, **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos**, **set_fly_limits**, **set_fly_limits_z**, **clear_fly_overflow**, **clear_fly_overflow_ctrl**, **if_not_activated**

Kontrollbefehl	**get_master_slave**
Funktion	Gibt den Master/Slave-Status der angesprochenen RTC6-Karte zurück.
Aufruf	`MasterSlaveStatus = get_master_slave()`
Rückgabe	Master/Slave-Status. Als 32-Bit-Wert ohne Vorzeichen. Information, ob am Master- oder Slave-Anschluss der angesprochenen RTC6-Karte eine weitere RTC6-Karte angeschlossen ist: Bit #0 = 1: Am Slave-Anschluss ist eine RTC6-Karte angeschlossen. (LSB) Bit #1 = 1: Am Master-Anschluss ist eine RTC6-Karte angeschlossen. Bit #2 = 0. … … Bit 31 = 0. Information, ob die angesprochene Karte als Master-Karte, Slave-Karte oder Einzelkarte betrieben wird: = 0 Einzelkarte. = 1 Slave ohne nachgeschaltete Slave-Karte. = 2 Master. = 3 Slave mit nachgeschalteter Slave-Karte.
Multi-Board-Befehlsname	**n_get_master_slave**
Hinweise	• Siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127. • **get_master_slave** liefert den Status nur dann zurück, wenn die angesprochene RTC6-Karte zuvor mit **load_program_file** initialisiert wurde. Anderenfalls wird 0 zurückgegeben.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**sync_slaves**

Kontrollbefehl	**get_mcbsp**
Funktion	Liest den Eingabewert, der zuletzt vollständig über die McBSP-Schnittstelle an die Speicherstelle für “Processing on the fly”-Anwendungen übertragen wurde.
Aufruf	`mcbsp_value = get_mcbsp()`
Rückgabe	Eingabewert. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_get_mcbsp**
Hinweise	• **get_mcbsp** ist gleichbedeutend mit **read_mcbsp**(0) (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**read_mcbsp**

Unverzögerter Kurzer Listenbefehl	**get_mcbsp_list**
Funktion	Ohne Funktion.
Aufruf	`get_mcbsp_list()`
Multi-Board-Befehlsname	**n_get_mcbsp_list**
Hinweise	• **get_mcbsp_list** hat keine Auswirkung auf das Anwenderprogramm.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_mcbsp**

Kontrollbefehl	get_out_pointer	
Funktion	Gibt die aktuelle (oder die letzte) Position des Ausgabe-Zeigers als Offset bezüglich des Starts der jeweiligen Liste sowie die Nummer der jeweiligen Liste zurück.	
Aufruf	`get_out_pointer( &ListNo, &Pos )`	
Parameter-Rückgaben	`ListNo`	Nummer der Liste (“Liste 1” oder “Liste 2”), in der sich der Ausgabe-Zeiger aktuell befindet (oder zuletzt befand). Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
	`Pos`	Aktuelle (oder letzte) Position des Ausgabe-Zeigers (relative Speicheradresse). Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_out_pointer**	
Hinweise	• **get_out_pointer** ruft **get_status** auf (siehe Hinweise dort, insbesondere bezüglich “Liste 3”), und ermittelt aus der von diesem Befehl zurückgegebenen absoluten Ausgabe-Zeiger-Position die Listennummer und die relative Position in der Liste. Mit der von **get_out_pointer** zurückgegebenen relativen Position und Listennummer kann beim alternierenden Wechsel die Ausgabe-Zeiger-Position leichter mit der aktuellen Eingabe-Zeiger-Position verglichen werden, insbesondere bezüglich Listengleichheit.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**get_status**, **get_input_pointer**, **get_list_pointer**	

Kontrollbefehl	get_overrun
Funktion	Gibt die seit dem letzten Aufruf aufgetretenen 10 µs-Takt-Überläufe zurück und löscht den Überlaufzähler.
Aufruf	`NumberOfOverruns = get_overrun()`
Rückgabe	Anzahl der seit dem letzten Aufruf von **get_overrun** aufgetretenen 10 µs-Takt-Überläufe. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_overrun**
Hinweise	• Siehe Abschnitt “Takt-Überläufe”, Seite 190. • Siehe auch Sky-Writing-Modus 4.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Kontrollbefehl	get_rcm_value_ctrl	
Funktion	Relevant für Rückkanal-Multiplexing: Ersetzt **get_value**, wenn bei aktivem Rückkanal-Multiplexing ein einzelner Wert vom i*DRIVE*-Scan-System ausgelesen werden soll.	
Aufruf	`ReturnValue = get_rcm_value_ctrl( Head, Axis, Value )`	
Parameter	`Head`	Nummer des Scan-Kopf-Anschlusses der RTC-Karte. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Scan-System am 1. Scan-Kopf-Anschluss. = 2: Scan-System am 2. Scan-Kopf-Anschluss.
	`Axis`	Nummer der Achse. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: x-Achse (STATUS-Kanal, Galvanometerscanner 2). = 2: y-Achse (STATUS1-Kanal, Galvanometerscanner 1).
	`Value`	Rückkanal-Multiplexing-Index, dessen Wert zurückgegeben werden soll. Zulässiger Wertebereich: 0…15. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Wert am Rückkanal-Multiplexing-Index. Bit #31…Bit #20 sind immer 0. Nutzdaten in Bit #19…Bit #0. Als 32-Bit-Wert ohne Vorzeichen.	
Multi-Board-Befehlsname	**n_get_rcm_value_ctrl**	
Hinweise	• **get_rcm_value_ctrl** nutzt zur Synchronisation den letzten SL2-100-Blockstart. Nicht berücksichtigt werden die tatsächlichen Rückkanal-Multiplexing-Einstellungen. Deshalb kann der Rückgabe-Wert bis zu 1,92 ms “veraltet” sein. • **get_rcm_value_ctrl** kann auch ohne aktives Rückkanal-Multiplexing verwendet werden, was in diesem Fall jedoch keinen Mehrwert gegenüber **get_value** bietet. • **get_rcm_value_ctrl** funktioniert auch ohne aktuell laufende Aufzeichnung. Daher ist ein vorhergehender **set_trigger[*]**-Aufruf nicht nötig.	
Beispiel (C/C++)	–	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 647, OUT 650.	
Verweise	**demux_rcm_ctrl**	

Kontrollbefehl	get_rtc_mode
Funktion	Gibt den aktuell eingestellten Betriebsmodus der RTC6-DLL zurück.
Aufruf	`DLLMode = get_rtc_mode()`
Rückgabe	RTC6-DLL-Betriebsmodus. Als 32-Bit-Wert ohne Vorzeichen. = 4: RTC4-Kompatibilitätsmodus. = 5: RTC5-Kompatibilitätsmodus = 6: RTC6-Standardmodus (Standardeinstellung).
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Der RTC6-DLL-Betriebsmodus kann mit **set_rtc4_mode**, **set_rtc5_mode** oder **set_rtc6_mode** eingestellt werden. Standardmäßig ist der RTC6-Standardmodus voreingestellt. • **get_rtc_mode** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133, werden von **get_rtc_mode** weder erzeugt noch verändert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_rtc4_mode**, **set_rtc5_mode**, **set_rtc6_mode**

Kontrollbefehl	**get_rtc_version**
Funktion	Gibt die Versionsnummer der FPGA-Firmware (`RTC6RBF.rbf`) und Informationen über freigeschaltete Optionen der RTC6-Karte zurück.
Aufruf	`RTCVersion = get_rtc_version()`
Rückgabe	Versionsnummer der FPGA-Firmware und weitere Informationen. Als 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) … Bit #7: Versionsnummer der FPGA-Firmware (`RTC6RBF.rbf`). Bit #8 = 1: Die Option "Processing on the fly" ist freigeschaltet. Siehe auch Kapitel 8.6 ""Processing on the fly"", Seite 254. Bit #9 = 1: Die Option "Zweite-Scan-Kopf-Steuerung" ist freigeschaltet. Siehe auch Abschnitt "2. SCANHEAD-Stiftleiste", Seite 70. Bit #10 = 1: Die Option "3D" ist freigeschaltet. Siehe auch Kapitel 8.5.2 "3D-Scan-Systeme", Seite 249. Bit #11 = 1: Die Option "LDSA" ist freigeschaltet (ab DLL 628). Bit #12 = 1: Die Option "SCANa" ist freigeschaltet (ab DLL 605). Bit #13 = 1: Die Option "UFPM" ist freigeschaltet (ab DLL 605). Bit #14 = 1: Die Option "syncA" ist freigeschaltet (ab DLL 607). Bit #15: Reserviert. Bit #16 … Bit #23: DSP-Versionsnummer. Bit #24 … Bit #31: Subversionsnummer der FPGA-Firmware (`RTC6RBF.rbf`).
Multi-Board-Befehlsname	**n_get_rtc_version**
Hinweise	• Die FPGA-Firmware-Versionsnummern liegen im Bereich 600…699. • **get_rtc_version**`( Bit #0…Bit #7 )` gibt Werte im Bereich 0…99 (Versionsnummer – 600) zurück. • Die aktuelle DSP-Versionsnummer ist 3. • Die aktuelle FPGA-Firmware-Subversion ist 0. • Die FPGA-Firmware-Version kann auch nach einem `RTC6_VERSION_MISMATCH` und `RTC6_ACCESS_DENIED`-Fehler noch zurückgegeben werden. Der Rückgabewert ist 0, wenn noch kein Programm geladen ist.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_hex_version**, **get_dll_version**

Kontrollbefehl	get_scanahead_params
Funktion	Nur für SCANahead-Systeme. Liest aus dem angegebenen SCANahead-System bestimmte Parameter aus.
Einschränkung	**get_scanahead_params** kann nur ausgeführt werden, wenn gerade keine Liste abgearbeitet wird. Andernfalls wird der **get_last_error** Returncode `RTC6_BUSY` und der Fehlercode 5 zurückgegeben.
Aufruf	`Error = get_scanahead_params( HeadNo, &PreviewTime, &Vmax, &Amax )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf. Option "Zweite-Scan-Kopf-Steuerung" erforderlich. = 256: RTC6-Karte (kein Scan-Kopf-Anschluss). `Vmax` und `Amax` werden, in Bildfeld-Koordinaten umgerechnet, zurückgegeben.
Parameter-Rückgaben	`PreviewTime` Vorausschauzeit der Galvanometer-Regelung. In 10 µs. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. `Vmax` Für `HeadNo` 1, 2: Geschwindigkeitsgrenze in Ansteuer-Bits (Galvanometerscanner-Drehwinkel). In Bit/10 µs. Für `HeadNo` 256: Geschwindigkeitsgrenze in Programmier-Bits (Bildfeld-Koordinaten). In Bit/10 µs. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. `Amax` Für `HeadNo` 1, 2: Beschleunigungsgrenze in Ansteuer-Bits (Galvanometerscanner-Drehwinkel). In Bit/(10 µs)². Für `HeadNo` 256: Beschleunigungsgrenze in Programmier-Bits (Bildfeld-Koordinaten). In Bit/(10 µs)². Als Zeiger auf einen Wert im 64-Bit-IEEE Gleitkommaformat.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 3 Es ist kein SCANahead-System angeschlossen oder das SCANahead-System-Tuning ist nicht aktiv (**get_last_error** Returncode `RTC6_PARAM_ERROR`). 5 Eine Liste ist gerade aktiv (**get_last_error** Returncode: `RTC6_BUSY`). 6 Es wurde als Parameter `HeadNo` = `0` oder `HeadNo` >2 angegeben (**get_last_error** Returncode: `RTC6_PARAM_ERROR`). 8 Die RTC6-Karte reagiert nicht. Möglicherweise wurde noch kein Programm geladen (**get_last_error** Returncode: `RTC6_TIMEOUT`). 11 Ein PCI-Fehler ist aufgetreten (**get_last_error** Returncode: `RTC6_SEND_ERROR`).
Multi-Board-Befehlsname	**n_get_scanahead_params**

Kontrollbefehl	get_scanahead_params
Hinweise	• Die Parameter-Rückgabe-Werte werden auch bei **set_scanahead_params** verwendet. • `Vmax` ist die maximale Winkelgeschwindigkeit der Galvanometerscanner. `Vmax` *ist nicht* die Markiergeschwindigkeit in der Bildebene. Daher ist die Einheit "In Bit/10 *µs*." und *nicht* "In Bits/ms.". Die Verzeichnung durch die Korrekturdatei ist bei dem zurückgelesenen Wert noch nicht berücksichtigt (siehe bei **set_scanahead_params**). • Umrechnung der Maximalgeschwindigkeit `Vmax` von Programmier-Bits/Takt in m/s: (`Vmax` [Bit/Takt] / K [Bit/mm]) × ((0,001 [m/mm] / 0,00001 [s/Takt]) • Umrechnung der Maximalbeschleunigung `Amax` (aus `get_scanahead_params( HeadNo = 256 )`) von Programmier-Bits/Takt² in m/s²: (`Amax` [Bit/Takt²] / K [Bit/mm]) × ((0,001 [m/mm] / (0,00001)² [s²/Takt²]) • `HeadNo = 256` gibt nur dann sinnvolle Werte zurück, wenn vorher **set_scanahead_params** aufgerufen wurde. • Die mit `HeadNo = 256` zurückgegebenen Werte sind skaliert. Sie dürfen nicht in `set_scanahead_params( Mode = 2 )` verwendet werden, weil sie dann ein weiteres Mal skaliert werden würden. • Für Dual-SCANahead-Systeme (z.B. 2 excelli*SCAN*-Scan-Köpfe; Option "Zweite-Scan-Kopf-Steuerung" erforderlich) wird empfohlen, dass die verwendeten Einzel-SCANahead-Systeme möglichst ähnliche Eigenschaften haben (Apertur, Objektiv, Korrekturdatei): Zur automatischen Berechnung der Scanner-Delays und Laser-Delays wird u.a. `Amax` des SCANahead-Systems `HeadNo` benötigt. Die Delays werden nur für dieses SCANahead-System berechnet, aber auch unverändert für das zweite verwendet. Unterscheiden sich die beiden SCANahead-Systeme in z.B. `Amax` zu stark, kann es zu unerwünschten Abweichungen zwischen den Markierungen kommen.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600. Letzte Änderung mit Version DLL 605: Datentyp von `Amax`.
Verweise	**set_scanahead_params**, **get_last_error**

Kontrollbefehl	get_serial
Funktion	Gibt die aktuelle Seriennummer des mit **select_serial_set** ausgewählten Seriennummernsatzes (oder nach **load_program_file** des Seriennummernsatzes 0) zurück.
Aufruf	`CurrentSerialNo = get_serial()`
Rückgabe	Seriennummer. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_get_serial**
Hinweise	• Zur Verwendung von **get_serial** siehe Kapitel 7.5.2 "Seriennummern markieren", Seite 222. • **get_serial** sollte nicht verwechselt werden mit **get_serial_number**, der die Seriennummer der RTC6-Karte zurückliefert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_serial_set**

Kontrollbefehl	**get_serial_number**
Funktion	Gibt die Seriennummer der aktiven RTC6-Karte zurück.
Aufruf	`RTCSerialNumber = get_serial_number()`
Rückgabe	RTC6-Seriennummer. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_serial_number**
Hinweise	• Die Seriennummern der eingebauten Karten werden durch **init_rtc6_dll** ermittelt und in der RTC6-DLL zwischengespeichert. Dort werden sie dann von **get_serial_number** ausgelesen. • **get_serial_number** ist hilfreich, wenn mehrere RTC6-Karten in *einem* Computer benutzt werden, siehe Kapitel 6.6 "Mehrere RTC6 PCIe-Karten in einem PC verwenden", Seite 126. Mit dem zugehörigen Multi-Board-Befehl **n_get_serial_number** kann die Zuordnung zwischen den bei der Initialisierung für die installierten RTC6-Karten vergebenen RTC6-DLL-internen Nummern und den installierten Karten ermittelt werden. Die RTC6-DLL-internen Nummern werden bei jeder Initialisierung eines Anwenderprogramms (siehe **init_rtc6_dll**) neu vergeben und müssen bei diversen Befehlen (insbesondere bei allen Multi-Board-Befehlen) angegeben werden. Die Anzahl der bei der Initialisierung gefunden Karten kann mit **rtc6_count_cards** ausgelesen werden. • **get_serial_number** und **n_get_serial_number** sind auch ohne (momentane) explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. Die Rückgabewerte sind jedoch 0, solange nicht zuvor zumindest einmal ein Zugriff auf die angesprochene Karte erfolgreich war. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **get_serial_number** weder erzeugt noch verändert.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**rtc6_count_cards**

Kontrollbefehl	**get_standby**
Funktion	Gibt die aktuell eingestellten Standby-Parameter zurück.
Aufruf	`get_standby( &HalfPeriod, &PulseLength )`
Parameter-Rückgaben	`HalfPeriod` Eingestellte *halbe* Standby-Ausgabeperiode der Standby-Pulse. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs.
	`PulseLength` Eingestellte Pulslänge der Standby-Pulse. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs.
Multi-Board-Befehlsname	**n_get_standby**
Hinweise	• Zur Verwendung siehe Abschnitt "Signale für den "Laser standby"-Betrieb", Seite 194.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus dividiert die RTC6 die Werte für `HalfPeriod` und `PulseLength` durch 8.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_standby**

Kontrollbefehl	**get_startstop_info**
Funktion	Gibt Informationen über interne und externe Starts und Stopps seit dem letzten Aufruf von **get_startstop_info**. Außerdem: die aktuellen externen Start- und Stopp-Pegel, der Status und der Signalpegel der Lasersteuersignale, sowie etwaige Übertragungsfehler von und zum angeschlossenen Scan-System.
Aufruf	`StartStopInfo = get_startstop_info()`
Rückgabe	Infosignal. Als 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) = 1: Ein interner Start wurde seit dem letzten Aufruf von **get_startstop_info** (mit **execute_list** o.ä.) ausgeführt. Bit #1 = 1: Ein Externer Start wurde seit dem letzten Aufruf von **get_startstop_info** (über /START, /START2, /Slave-START, mit **simulate_ext_start** oder **simulate_ext_start_ctrl**) ausgeführt. Bit #2 = 1: Ein interner Stopp wurde seit dem letzten Aufruf von **get_startstop_info** (mit **stop_execution**) ausgeführt. Bit #3 = 1: Ein Externer Stopp wurde seit dem letzten Aufruf von **get_startstop_info** (über /STOP, /STOP2, /Slave-STOP oder mit **simulate_ext_stop**) ausgeführt. Bit #4 Ext-Stop-Status (= logische UND-Verknüpfung der Signale /STOP, /STOP2, /Slave-STOP und **simulate_ext_stop**, siehe Abbildung 72): = 1: An den Eingängen liegt momentan *kein* Stoppsignal an oder die Eingänge sind nicht angeschlossen. = 0: An mindestens einem der Eingänge liegt ein Stoppsignal an. Bit #5 ≥ DLL 619: = 1: Timeout bei **wait_for_timestamp_counter_mode** und **wait_for_timestamp_counter_long**. Bit #6 Reserviert. Bit #7 Reserviert. Bit #8 Reserviert. Bit #9 = 1: Die Lasersteuersignale sind global freigegeben, siehe Kapitel 7.4.1 "Lasersteuersignale freigeben, aktivieren und schalten", Seite 192. Siehe auch **set_laser_control**. Bit #10 = 1: Die TTL-Lasersteuersignale an den LASER1- und LASER2-Ausgängen sind LOW-aktiv (der Signalpegel kann mit **set_laser_control** eingestellt werden). Bit #11 = 1: Seit dem letzten Aufruf von **get_startstop_info** ist mindestens ein Externer Start nicht erfolgt (es sind mehr externe Starts angetriggert worden als gleichzeitig in der Warteschleife von 8 Starts gehalten werden können). Bit #12 Ext-Start-Status (= logische UND-Verknüpfung der Signale /START, /START2 und /Slave-START, siehe Abbildung 73): = 1: An den Eingängen liegt momentan *kein* Startsignal an oder die Eingänge sind nicht angeschlossen. = 0: An mindestens einem der Eingänge liegt ein Startsignal an. Bit #13 = 1: Das TTL-Lasersteuersignal am LASERON-Ausgang ist LOW-aktiv (der Signalpegel kann mit **set_laser_control** eingestellt werden).

Kontrollbefehl	**get_startstop_info**
Rückgabe (Forts.)	Bit #14 = 1: Die Lasersteuersignale sind freigegeben (**enable_laser**) = 0: Die Lasersteuersignale sind unterdrückt (**disable_laser**) Bit #15 = 1: Wenn zuvor mit **eth_configure_link_loss** aktiviert: Ein Ethernet Link Loss wurde seit dem letzten Aufruf von **get_startstop_info** erkannt. Bit #16 … Bit #31 Fehler-Bits. Werden gesetzt, wenn bei der Datenübertragung vom Scan-System ein Fehler auftritt: Bit #16…Bit #23 für den Anschluss für ersten Scan-Kopf Bit #24…Bit #31 für den Anschluss für zweiten Scan-Kopf Bit #16, Bit #24: Anzahl Frames innerhalb eines Datenblockes fehlerhaft. Bit #17, Bit #25: Pulslänge des vom Scan-System empfangenen Signals unzulässig, möglicherweise kein Scan-System angeschlossen. Bit #18, Bit #26: Preamble-Folge unzulässig. Bit #19, Bit #27: Bitanzahl innerhalb eines Subframes fehlerhaft. Bit #20, Bit #28: Parityfehler beim Lesen der vom Scan-System empfangenen Daten. Bit #21, Bit #29: Die anliegenden Daten sind ungültig (veraltet). Bit #22, Bit #30: Reserviert. Bit #23, Bit #31: Reserviert.
Multi-Board-Befehlsname	**n_get_startstop_info**
Hinweise	• Nachdem **get_startstop_info** ausgeführt wurde, werden zurückgesetzt: – Die Info-Bits Bit #0…Bit #3, Bit #15 – Die Fehler-Bits Bit #5, Bit #16…Bit #31 • Siehe auch Abschnitt "Externer Stopp", Seite 314 und Abschnitt "Externer Start", Seite 315.
RTC4→RTC6	Unveränderte Funktionalität für bereits bei der RTC4 verwendete Info-Bits.
RTC5→RTC6	Unveränderte Funktionalität für bereits bei der RTC5 verwendete Info-Bits.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 622, OUT 622: Bit #15.
Verweise	**get_counts**, **get_status**, **set_control_mode**

Kontrollbefehl	get_status	
Funktion	Gibt die aktuellen Listenausführungsstatus-Werte und die aktuelle (oder die letzte) Position des Ausgabe-Zeigers zurück.	
Aufruf	`get_status( &Status, &Pos )`	
Parameter-Rückgaben	`Status`	Statuswert. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) = 1: BUSY-Listenausführungsstatus gesetzt. Bit #1 Reserviert. … … Bit #6 Reserviert. Bit #7 = 1: INTERNAL-BUSY-Listenausführungsstatus gesetzt. Bit #8 Reserviert. … … Bit #14 Reserviert. Bit #15 = 1: PAUSED-Listenausführungsstatus gesetzt. Bit #16 Reserviert. … … Bit #22 Reserviert. Bit #23 = 1: HEAD BUSY-Listenausführungsstatus gesetzt. Betriebsstatus nur für SCANahead-Systeme. Bit #24 Reserviert. … … Bit #31 Reserviert.
	`Pos`	Aktuelle (oder letzte) Position des Ausgabe-Zeigers (absolute Speicheradresse). Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_status**	
Hinweise	• Wann die BUSY-Listenausführungsstatus, INTERNAL-BUSY-Listenausführungsstatus- oder PAUSED-Listenausführungsstatus-Werte gesetzt oder nicht gesetzt sind, wird unter Kapitel 6.4.3 "Listenausführungsstatus", Seite 111 beschrieben. • Für (BUSY-Listenausführungsstatus und PAUSED-Listenausführungsstatus gesetzt) muss mit **restart_list** fortgefahren werden, für (BUSY-Listenausführungsstatus nicht gesetzt und PAUSED-Listenausführungsstatus gesetzt) mit **release_wait** und für (BUSY-Listenausführungsstatus und PAUSED-Listenausführungsstatus nicht gesetzt) mit **execute_list_pos**. Für (BUSY-Listenausführungsstatus gesetzt und PAUSED-Listenausführungsstatus nicht gesetzt) läuft aktuell eine Liste, ein "Fortfahren" ist nicht erlaubt. Ein falsches Fortfahren erzeugt den **get_last_error**-Returncode `RTC6_BUSY`. Für (INTERNAL-BUSY-Listenausführungsstatus gesetzt) werden **release_wait** und **execute_list_pos** erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Listenausführungsstatus wieder zurückgesetzt wurde).	

Kontrollbefehl	get_status
Hinweise (Forts.)	• Der Ausgabe-Zeiger zeigt auf den Listenbefehl (in “Liste 1” oder “Liste 2”), der momentan ausgeführt wird oder zuletzt ausgeführt wurde. Wenn die Ausgabe-Zeiger-Position `Pos` während der Abarbeitung eines Unterprogramms im geschützten RTC6-Listenspeicher-Bereich “Liste 3” abgefragt wird, so wird die Position *des* Listenbefehls aus dem Listen-Bereich (“Liste 1” oder “Liste 2”) zurückgegeben, in der sich der Ausgabe-Zeiger zuletzt befunden hat (in der Regel von wo aus das Unterprogramm z.B. mit **list_call** aufgerufen wurde). **pause_list** und **set_wait** lassen `Pos` unverändert. • **get_status** liefert die Ausgabe-Zeiger-Position als absolute Speicheradresse (Offset bezüglich Listenanfang von “Liste 1”) zurück. Die relative Position bezüglich des Listenanfangs des jeweiligen Listen-Bereichs kann mit **get_out_pointer** ausgelesen werden. • Die aktuelle Eingabe-Zeiger-Position kann mit **get_input_pointer** oder **get_list_pointer** ausgelesen werden. • Die Listenstatuswerte für die einzelnen Listen können mit **read_status** ausgelesen werden. • **get_status**, **get_input_pointer** und **read_status** kann man z.B. dazu verwenden, beim Laden einer Liste sicherzustellen, dass keine Listenbefehle überschrieben werden, die noch nicht abgearbeitet sind (siehe auch **load_list**). • Wenn noch kein Programm (mit **load_program_file**) geladen ist, liefert **get_status** undefinierte Werte zurück. • Zum Auslesen der Statussignale der Scan-Köpfe dient **get_head_status**.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Bei der RTC6 liefert der Parameter `Status` zusätzlich den INTERNAL-BUSY-Listenausführungsstatus und PAUSED-Listenausführungsstatus.
RTC5→RTC6	Veränderte Funktionalität. • Der HEAD BUSY-Listenausführungsstatus wird über `Bit #23` zurückgegeben.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**read_status**, **get_input_pointer**, **get_list_pointer**, **get_out_pointer**

Kontrollbefehl	**get_stepper_status**
Funktion	Gibt für beide Schrittmotor-Ausgänge den jeweils aktuellen Status der Schrittmotorsignale, die aktuell eingestellte CLOCK-Pulsperiode, den Status “Busy” und Status “Init” sowie den aktuellen Wert der internen Positionsvariablen zurück.
Aufruf	`get_stepper_status( &Status1, &Pos1, &Status2, &Pos2 )`
Parameter-Rückgaben	`Status1` Aktueller Status des Schrittmotor-Ausgangs 1. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) ENABLE-Signal. Bit #1 DIRECTION-Signal. Bit #2 CLOCK-Signal. Bit #3 SWITCH-Signal = Endschalter-Signal. Bit #4 Status “Busy”. Bit #5 Status “Init”. Bit #6 Reserviert. Bit #7 Reserviert. Bit #8 CLOCK-Pulsperiode (24-Bit-Wert). … Bit #31
	`Pos1` Aktueller Wert der internen Positionsvariablen von Schrittmotor-Ausgang 1. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen.
	`Status2` Aktueller Status des Schrittmotor-Ausgangs 2. Sonst wie `Status1`.
	`Pos2` Wie `Pos1`.
Multi-Board-Befehlsname	**n_get_stepper_status**
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 “Schrittmotoren ansteuern”, Seite 309. • Wenn das SWITCH-Signal (Endschalter-Signal) gesetzt ist (Pin ist LOW), werden keine CLOCK-Pulse mehr erzeugt. • Der Status “Busy” zeigt an, dass eine zuvor (mit **stepper_abs**, **stepper_rel**, etc.) initiierte Zielpositionsfahrt noch nicht abgeschlossen ist. • Der Status “Init” zeigt an, dass eine zuvor (mit **stepper_init**) initiierte Referenzfahrt noch nicht abgeschlossen ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Kontrollbefehl	**get_sub_pointer**
Funktion	Gibt die absolute Startadresse eines indizierten Unterprogramms zurück.
Aufruf	`SubPointer = get_sub_pointer( Index )`
Parameter	`Index` Index des indizierten Unterprogramms. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Rückgabe	Absolute Startadresse. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_sub_pointer**
Hinweise	• **get_sub_pointer** liest die Startadresse des indizierten Unterprogramms für den ausgewählten Index aus der internen Verwaltungstabelle. Je nachdem, ob das Unterprogramm in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” geladen wurde oder ein ungeschütztes Unterprogramm nur nachträglich referenziert wurde, liegt die zurückgelesene Speicheradresse im geschützten oder ungeschützten RTC6-Listenspeicher-Bereich. • Für `Index` > 1023 oder wenn kein Unterprogramm für den ausgewählten Index referenziert ist, gibt der Befehl einen Wert von “–1” (d.h. 2^{32}–1) zurück. • **get_sub_pointer** ist nützlich zur Überprüfung, ob ein Unterprogramm bereits definiert wurde oder auch, um das indizierte Unterprogramm auch wie ein nichtindiziertes Unterprogramm unter Angabe der absoluten Speicheradresse aufrufen zu können. Bei letzterem sollte man allerdings berücksichtigen, dass sich die absolute Speicheradresse bei einem anschließenden **save_disk**/**load_disk** u.U. ändert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_char_pointer**, **get_text_table_pointer**

Kontrollbefehl	**get_sync_status**		
Funktion	Gibt Master/Slave-Synchronisierungsinformationen zu der angesprochenen RTC6-Karte zurück.		
Aufruf	`MasterSlaveSyncStatus = get_sync_status()`		
Rückgabe	Master/Slave-Synchronisierungsinformationen. Als 32-Bit-Wert ohne Vorzeichen.		
	Bit #0 … Bit #9	Master/Slave-Synchronisierungszustand [0…640]. Als 10-Bit-Wert ohne Vorzeichen.	
		< 4:	Die angesprochene Karte ist synchron zur Master-Karte (oder ihrer Vorgängerkarte in der Master/Slave-Kette).
		640:	Die angesprochene Karte wird als Master-Karte betrieben.
	Bit #10	= 0:	Bit #0…Bit #9 enthalten gültige Werte.
		= 1:	Bit #0…Bit #9 enthalten keine gültigen Werte.
	Bit #11	Reserviert.	
	Bit #12	= 1:	Es wurde eine Master-Karte erkannt. Jedoch kann kein /Slave-START oder /Slave-STOP empfangen werden. Siehe **master_slave_config**.
	Bit #13	= 1:	Es wurde eine Slave-Karte erkannt. Jedoch kann kein /Slave-START oder /Slave-STOP empfangen werden. Siehe **master_slave_config**.
	Bit #14	= 1:	Seit dem letzten Aufruf von **get_sync_status** ist die Verbindung zum Master abgebrochen.
	Bit #15	= 1:	Seit dem letzten Aufruf von **get_sync_status** ist die Verbindung zum Slave abgebrochen.
	Bit #16 … Bit #20	Reserviert. … Reserviert.	
	Bit #21 … Bit #30	Exakte Laufzeit in 1/64 µs-Takten zwischen zwei RTC6-Karten (Hinweg und Rückweg). Als 10-Bit-Wert ohne Vorzeichen.	
	Bit #31	= 1:	Die Laufzeit (Bit #21…Bit #30) konnte nicht erfolgreich gemessen werden.
Multi-Board-Befehlsname	**n_get_sync_status**		

Kontrollbefehl	get_sync_status
Hinweise	• Zur Verwendung von **get_sync_status** siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127. • Falls die angesprochene RTC6-Karte nicht zuvor mit **load_program_file** initialisiert wurde, liefert **get_sync_status** 0 zurück. • Ab RBF 619 gilt: – Die Synchronisierung erfolgt automatisch. **sync_slaves** ist nicht mehr erforderlich. – Der Master/Slave-Synchronisierungszustand wird automatisch gemessen. Die exakte Laufzeit wird über Bit #21…Bit #30 zurückgegeben, wenn Bit #31 keinen Fehler anzeigt. • Bis RBF 618 gilt: – Die Karten der Master/Slave-Kette sollten vor **get_sync_status** mit **sync_slaves** synchronisiert worden sein. – Damit **get_sync_status** tatsächlich den aktuellen Master/Slave-Synchronisierungszustand zurückliefert, muss zuvor auf der Master-Karte (mit externem Start-Signal oder mit **simulate_ext_start_ctrl** ein Externer Start ausgelöst worden sein, siehe Abschnitt "Externer Start", Seite 315. Dieser Start löst bei allen Slave-Karten, bei denen externe Starts und Stopps nicht unterdrückt sind (siehe **master_slave_config**) eine Messung des Zeitunterschieds zwischen dem jeweiligen /Slave-START-Puls und dem jeweiligen 10 μs-Takt aus. Dieser Zeitunterschied wird als Master/Slave-Synchronisierungszustand (in Einheiten von 1/64 μs) auf der jeweiligen Karte gespeichert, bleibt dort gespeichert, bis an der Master-Karte ein neuer Externer Start ausgelöst wird und kann daher auch noch zu einem späteren Zeitpunkt mit **get_sync_status** ausgelesen werden. – Im synchronisierten Zustand ist der gemessene Zeitunterschied kleiner als der Laufzeitunterschied bei der Synchronisierung zwischen den Karten selbst, siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127: Master/Slave-Synchronisierungszustand < 3. – Im nicht explizit synchronisierten Zustand (vor **sync_slaves**) kann der Synchronisierungszustand dennoch zufällig < 3 sein. Die Karten verhalten sich dann wie synchronisierte Karten. – Ist Bit #12 oder Bit #13 gesetzt, können die betroffenen Karten nicht mit **sync_slaves** synchronisiert werden. – Ist Bit #14 oder Bit #15 gesetzt, sind /Slave-STARTs oder /Slave-STOPs gegebenenfalls nicht weitergeleitet worden. In diesem Fall wird empfohlen, **master_slave_config** anzupassen und **sync_slaves** erneut auszuführen. – Wenn eines oder mehrere der Bit #12…Bit #15 sporadisch 1 sind, kann dies auf elektromagnetische Störungen hinweisen, die die Kommunikation zwischen den Karten beeinträchtigt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Grundsätzlich unveränderte Funktionalität. Jedoch stehen mehr Details zur Verfügung.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 614, OUT 614, RBF 619: interne Verbesserungen und zusätzliche Rückgaben.
Verweise	**sync_slaves**, **get_master_slave**, **master_slave_config**

Kontrollbefehl	**get_table_para**
Funktion	Gibt den gewünschten Parameterwert aus einer aktuell geladenen Korrekturtabelle zurück.
Aufruf	`TablePara = get_table_para( TableNo, ParaNo )`
Parameter	`TableNo` Nummer der aktuell geladenen Korrekturtabelle. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [1…8]. Siehe auch **number_of_correction_tables**.
	`ParaNo` Nummer des Parameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: 0…15. Zuordnung siehe Abschnitt "ct5-Korrekturdateikopf", Seite 186.
Rückgabe	Parameterwert, siehe Abschnitt "ct5-Korrekturdateikopf", Seite 186. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_get_table_para**
Hinweise	• Die Parameterwerte können mit **get_table_para** aus einer aktuell geladenen und mit **get_head_para** aus einer zugewiesenen RTC6-Korrekturtabelle ausgelesen und auf diese Weise direkt in ein Anwenderprogramm eingebunden werden, siehe Abschnitt "ct5-Korrekturdateikopf", Seite 186. • Liegen die Parameter `TableNo` und `ParaNo` im unzulässigen Bereich, ist der Rückgabewert 0 (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Wenn noch keine Korrekturtabelle mit der angegebenen Nummer geladen wurde, sind die Parameterwerte undefiniert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Kontrollbefehl	**get_temperature**
Funktion	Gibt die aktuelle Temperatur der RTC6-Karte zurück.
Aufruf	`Temp = get_temperature()`
Parameter	Keine.
Rückgabe	Temperatur der RTC6-Karte. In Grad Celsius. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_get_temperature**
Hinweise	• Die Ausführung von **get_temperature** dauert etwa 1 ms. • Das kontinuierliche Auslesen der Analog-Spannung (nach einem **read_analog_in**-Aufruf) wird durch **get_temperature** um etwa 0,1 ms verzögert, siehe auch Abschnitt "Analog-Eingänge", Seite 89.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 631, OUT 632.
Verweise	–

Kontrollbefehl	**get_text_table_pointer**
Funktion	Gibt die absolute Startadresse einer indizierten Zeichenfolge zurück.
Aufruf	`TextTablePointer = get_text_table_pointer( Index )`
Parameter	`Index` Index der indizierten Zeichenfolge. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…41].
Rückgabe	Absolute Startadresse. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_text_table_pointer**
Hinweise	• **get_text_table_pointer** liest die Startadresse der indizierten Zeichenfolge für den ausgewählten Index aus der internen Verwaltungstabelle. Je nachdem, ob die Zeichenfolge in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” geladen wurde oder ein ungeschütztes Unterprogramm nur nachträglich referenziert wurde, liegt die zurückgelesene Speicheradresse im geschützten oder ungeschützten RTC6-Listenspeicher-Bereich. • Für `Index` > 41 oder wenn keine Zeichenfolge für den ausgewählten Index referenziert ist, gibt der Befehl einen Wert von “–1” (d.h. 2^{32}–1) zurück. • **get_text_table_pointer** ist nützlich zur Überprüfung, ob eine Zeichenfolge bereits definiert wurde oder auch, um die indizierte Zeichenfolge auch wie ein nichtindiziertes Unterprogramm unter Angabe der absoluten Speicheradresse aufrufen zu können, z.B. für eine bedingte Ausführung mit **list_call_cond**. Dabei sollte man allerdings berücksichtigen, dass sich die absolute Speicheradresse bei einem anschließenden **save_disk**/**load_disk** u.U. ändert. Außerdem sollte dann sichergestellt werden, dass **get_text_table_pointer** nicht “–1” zurückgibt, sonst wird **list_call_cond** ignoriert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_char_pointer**, **get_sub_pointer**

Kontrollbefehl	**get_time**
Funktion	Gibt den Wert des RTC6-Timers zurück (ohne ihn auf Null zurückzusetzen). Er wurde beim letzten Aufruf von **save_and_restart_timer** gespeichert.
Aufruf	`TimerValue = get_time()`
Rückgabe	Wert des RTC6-Timers. In Sekunden. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_get_time**
Hinweise	• Siehe **save_and_restart_timer**. • Wie viele Listenbefehlstakte seit dem Zurücksetzen auf Null vergangen sind, kann mit **get_lap_time** abgefragt werden.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_lap_time**, **save_and_restart_timer**

Kontrollbefehl	**get_timestamp_long**
Funktion	Gibt den aktuellen 64-Bit-"Zeitstempel-Zähler"-Wert zurück.
Aufruf	`get_timestamp_long( &TimeStampL, &TimeStampH )`
Parameter-Rückgaben	`TimeStampL` Unterer Teil des 64-Bit-"Zeitstempel-Zähler"-Werts. Als 32-Bit-Wert ohne Vorzeichen.
	`TimeStampH` Oberer Teil des 64-Bit-"Zeitstempel-Zähler"-Werts. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_timestamp_long**
Hinweise	• Der RTC6-Karten-interne 64-Bit-"Zeitstempel-Zähler" `TimeStampCounterLong` wird: – Bei **load_program_file** mit 0 initialisiert und läuft dann bis zum nächsten **load_program_file** weiter – Bei **wait_for_timestamp_counter_long** verwendet • Der untere Teil von `TimeStampCounterLong` (= Parameterrückgabe `TimeStampL`) und der 32-Bit-"Zeitstempel-Zähler" sind identisch. • Siehe Kapitel 8.12 "Zeitmessungen", Seite 293.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 624, OUT 624.
Verweise	**wait_for_timestamp_counter_long**

Kontrollbefehl	**get_transform**
Funktion	Wie **get_transform_offset**, aber ohne Parameter `Channel1`, `Channel2`, `Offset`.
Aufruf	`get_transform( Number, Ptr1, Ptr2, Ptr, Code )`
Parameter	`Number` Anzahl der rückwärts zu transformierenden Positionswerte. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…2^{24}]. Es werden die Messwerte mit Index 0 bis (`Number`–1) rückwärtstransformiert.
	`Ptr1` Wie **get_transform_offset**.
	`Ptr2` Wie **get_transform_offset**.
	`Ptr` Wie **get_transform_offset**.
	`Code` Wie **get_transform_offset**.
Multi-Board-Befehlsname	**n_get_transform**
Hinweise	• `get_transform( Number, Ptr1, Ptr2, Ptr, Code )` – Überträgt die mit **set_trigger[*]** aufgezeichneten Datenpaare auf den PC mit • **get_waveform**(1, Number, Ptr1) und • **get_waveform**(2, Number, Ptr2) – Überschreibt die Daten paarweise mit **transform**(`Sig1`, `Sig2`, `Ptr`, `Code`) • Die Hinweise, Seite 490 zu **get_transform_offset** gelten analog.
RTC4→RTC6	Neuer Befehl. Alle auf den PC übertragenen Koordinatenwerte sind auch im RTC4-Kompatibilitätsmodus im RTC6-20-Bit-Bereich. Die rückwärtstransformierten Koordinatenwerte müssen gegebenenfalls vom Benutzer selbst durch 16 dividiert werden.
RTC5→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Zulässiger Wertebereich von `Number` noch [1…2^{23}].
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 635, OUT 636: `Number`. Zulässiger Wertebereich erweitert auf [1…2^{24}] (vormals: [1…2^{23}]).
Verweise	**upload_transform**, **transform**, **set_trigger**, **set_trigger4**, **set_trigger8**, **get_waveform**, **get_transform_offset**

Kontrollbefehl	get_transform_offset	
Funktion	Überträgt Positionswerte, die mit **set_trigger[*]** aufgezeichnet und auf der RTC6 gespeichert wurden, auf den PC und führt eine Rückwärtstransformation durch.	
Aufruf	`get_transform_offset( Channel1, Channel2, Offset, Number, Ptr1, Ptr2, Ptr, Code )`	
Parameter	`Channel1`	Messkanal. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [1...8].
	`Channel2`	Wie `Channel1`.
	`Offset`	Startposition. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…2^{24}].
	`Number`	Anzahl der rückwärts zu transformierenden Positionswerte. Als 32-Bit-Wert ohne Vorzeichen. Rückwärtstransformiert werden die Messwerte von Position `Offset`…[(`Offset` + (`Number` – 1)].
	`Ptr1`	Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, an den der rückwärtstransformierte Wert übertragen werden soll (siehe auch `Code`).
	`Ptr2`	Wie `Ptr1`.
	`Ptr`	Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, in den die Korrektur- und Transformationseinstellungen, die zur Rückwärtstransformation verwendet werden sollen, zuvor mit **upload_transform** übertragen wurden.
	`Code`	Details der Rückwärtstransformation, insbesondere welche Teiltransformationen durchgeführt werden. Soll eine Teiltransformationen *nicht* durchgeführt werden, so muss das entsprechende Bit auf 1 (Bit #2…Bit #5, Bit #2…Bit #4) gesetzt werden. `Code` hat die gleiche Bedeutung wie `Code` von **transform** (`Ptr1` entspricht `Sig1`, `Ptr2` entspricht `Sig2`). Als 32-Bit-Wert ohne Vorzeichen.

Kontrollbefehl	get_transform_offset		
Parameter (Forts.)	Code (Forts.)	Für Bit #0 = 0 werden die mit **set_trigger[*]** aufgezeichneten Messwertpaare als xy-Koordinaten rückwärtstransformiert:	
		Bit #1	= 0: Die über Speicherkanal 1 (Signal1) aufgezeichneten Werte werden unter Ptr1 auf den PC übertragen und als x-Koordinaten rückwärtstransformiert. Die über Speicherkanal 2 (Signal2) aufgezeichneten Werte werden unter Ptr2 auf den PC übertragen und als y-Koordinaten rückwärtstransformiert.
			= 1: Die über Speicherkanal 1 (Signal1) aufgezeichneten Werte werden unter Ptr1 auf den PC übertragen und als y-Koordinaten rückwärtstransformiert. Die über Speicherkanal 2 (Signal2) aufgezeichneten Werte werden unter Ptr2 auf den PC übertragen und als x-Koordinaten rückwärtstransformiert.
		Bit #2	= 0: Die Gain/Offset-Korrektur der automatischen Selbstkalibrierung wird rückwärtstransformiert.
		Bit #3	= 0: Die Bildfeld-Korrektur wird rückwärtstransformiert.
		Bit #4	= 0: Der Offset der definierten Koordinatentransformation wird rückwärtstransformiert.
		Bit #5	= 0: Die Gesamtmatrix der definierten Koordinatentransformation wird rückwärtstransformiert.
		Bit #6	Reserviert.
		…	…
		Bit #31	Reserviert.

Kontrollbefehl	**get_transform_offset**
Parameter (Forts.)	`Code` (Forts.) Für Bit #0 = 1 werden (nur) die Werte, die über einen der beiden Speicherkanäle (wahlweise Kanal 1 oder 2) mit **set_trigger[*]** aufgezeichnet wurden, als z-Koordinaten rückwärtstransformiert: Bit #1 = 0: Die über Speicherkanal 1 (`Signal1`) aufgezeichneten Werte werden unter `Ptr1` auf den PC übertragen und als z-Koordinaten rückwärtstransformiert. Die über Speicherkanal 2 (`Signal2`) aufgezeichneten Werte werden untransformiert unter `Ptr2` auf den PC übertragen. = 1: Die über Speicherkanal 2 (`Signal2`) aufgezeichneten Werte werden unter `Ptr1` auf den PC übertragen und als z-Koordinaten rückwärtstransformiert. Die über Speicherkanal 1 (`Signal1`) aufgezeichneten Werte werden untransformiert unter `Ptr2` auf den PC übertragen. Bit #2 = 0: Der mit **set_defocus** oder **set_defocus_list** definierte Offset auf die Fokuslänge wird rückwärtstransformiert. Bit #3 = 0: Die ABC-Korrektur wird rückwärtstransformiert. Bit #4 = 0: Der mit **set_offset_xyz** oder **set_offset_xyz_list** definierte Offset auf die z-Koordinate wird rückwärtstransformiert. Bit #5 Reserviert. … … Bit #31 Reserviert.
Multi-Board-Befehlsname	**n_get_transform_offset**

Kontrollbefehl	get_transform_offset
Hinweise	• `get_transform_offset( Channel1, Channel2, Offset, Number, Ptr1, Ptr2, Ptr, Code )` – Überträgt die mit **set_trigger[*]** aufgezeichneten Datenpaare auf den PC mit • **get_waveform_offset**(Channel1, Offset, Number, Ptr1) und • **get_waveform_offset**(Channel2, Offset, Number, Ptr2) – Überschreibt die Daten paarweise mit **transform**(`Sig1`, `Sig2`, `Ptr`, `Code`) • Vor einem **get_transform_offset**-Aufruf: – Muss **upload_transform** ausgeführt worden sein – Sollten mit **set_trigger** Positionswerte aufgezeichnet worden sein • Vor einem **get_transform_offset**-Aufruf oder **get_waveform_offset**-Aufruf kann mit **measurement_status** überprüft werden, ob noch eine mit **set_trigger[*]** gestartete Aufzeichnung läuft. Zusätzlich kann mit **measurement_status** die Anzahl `Pos` der bei der letzten (oder aktuellen) Messung aufgezeichneten Datenpaare ausgelesen werden. Lesen Sie höchstens `Pos`+1 Datenelemente aus. Alle weiteren Datenelemente stammen aus früheren Aufzeichnungen oder der Initialisierung. • Sie müssen die Arbeitsspeicherbereiche unter `Ptr1` und `Ptr2` in ausreichender Größe (`Number` × 4 Byte je Kanal) zur Verfügung stellen. • Zur Rückwärtstransformation von Positionswerten siehe Kapitel 8.1.3 "Positionierung überwachen", Seite 226. • Wenn nur z-Positionswerte rückwärtstransformiert werden sollen (`Code` Bit #0 = 1), dann kann für den nicht benötigten Speicherkanal der Zeiger (`Ptr1` oder `Ptr2`) auf NULL gesetzt werden. Für diesen Speicherkanal werden dann keine Daten auf den PC übertragen und rückwärtstransformiert. Achten Sie in diesem Fall auf ein passend gesetztes `Code`-Bit #1. • Für beide Speicherkanäle werden keine Daten an den PC übertragen und auch keine Rückwärtstransformation durchgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`) bei: – **get_transform_offset**(`Ptr1` = NULL, `Ptr2` = NULL) – `Number` = 0 – `Number` > 2^{24} • Bei **get_transform_offset**(`Ptr` = NULL) wird keine Rückwärtstransformation durchgeführt. Die mit **set_trigger[*]** erfassten Daten werden dann aber wie mit **get_waveform**(1, `Number`, `Ptr1`) und **get_waveform**(2, `Number`, `Ptr2`) untransformiert auf den PC übertragen. Das Gleiche gilt auch für **get_transform_offset**(`Ptr` ≠ NULL), falls bei der Ausführung von **get_transform_offset** ein Fehler auftritt (z.B. unter `Ptr` stehen ungültige oder fehlerhafte Daten, eine z-Achsen-Invertierung ist nicht möglich). In diesem Fall wird zusätzlich ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt.

Kontrollbefehl	get_transform_offset
Hinweise (Forts.)	• Mit **set_trigger** erfasste und mit **get_transform_offset** rückwärtstransformiert auf dem PC gespeicherte Werte können bei Bedarf zusätzlich mit **get_waveform** untransformiert auf den PC übertragen werden. • Wird eine Rückwärtstransformation von z-Positionswerten veranlasst (`Code`-Bit-#0 = 1), war aber zum Zeitpunkt der zuvor erfolgten Ausführung von **upload_transform** nur eine 2D-Korrekturtabelle zugewiesen, so werden die Offsets auf die Fokuslänge oder die z-Koordinate mit 0 und die Werte A, B, C mit 0, 1, 0 initialisiert (1zu1-Rückwärtstransformation). • Bei der Rückwärtstransformation von xy-Positionswerten (`Code`-Bit-#0 = 0) wird nur die z = 0-Ebene transformiert. xy-Stretching und z-Defokus infolge von z-Abweichungen, insbesondere bei Nicht-F-Theta-Systemen, werden nicht berücksichtigt. • Im Falle einer fehlerhaften PCI-Übertragung folgt ein **get_last_error**-Returncode `RTC6_SEND_ERROR`.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 635, OUT 636.
Verweise	**get_transform**, **upload_transform**, **transform**, **set_trigger**, **set_trigger4**, **set_trigger8**, **get_waveform**, **get_transform**

Kontrollbefehl	**get_value**
Funktion	Gibt den aktuellen Wert des ausgewählten Signals zurück.
Aufruf	`Value = get_value( Signal )`
Parameter	`Signal` Gewünschter Signaltyp. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Aktueller Wert des ausgewählten Signals. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_get_value**
Hinweise	• Die auswählbaren Signaltypen sind identisch zu **set_trigger** (für den zulässigen Wertebereich, die Signaltypen und entsprechende Hinweise siehe dort). Bei unzulässigen Angaben für `Signal` führt `get_value` keine Signalabfrage aus und gibt den Wert 0 zurück (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Soll das ausgewählte `Signal` über einen längeren Zeitraum beobachtet werden, so empfiehlt es sich, mit **set_trigger** eine entsprechende Messung zu starten. • Bei iDRIVE-Scan-Systemen kann es nach einem Neustart oder Reset des Scan-Systems ca. 5 Sekunden dauern, bis es korrekte Daten überträgt. • Für `Signal` = 0 liefert **get_value** den aktuellen Laserstatus (LASERON-Signal) auch außerhalb einer Listenausführung zurück. • Wenn Sie Daten auslesen, die als Statussignale vom Scan-System an die RTC6 übertragen werden (`Status<AX…BY>`), dann achten Sie beim Auswerten auf den Wertebereich des übertragenen Datentyps (siehe **control_command**): – Bei Datentypen (wie dem XY2-100-Statuswort oder der Seriennummer), die im Scan-System ursprünglich als vorzeichenlose 16-Bit-Werte erzeugt werden und als vorzeichenlose 20-Bit-Werte an die RTC6 übertragen werden (Bit #0…Bit #3 = 0) werden die relevanten Informationen über die Bit #4…Bit #19 übertragen. Da **get_value** bei diesen Datentypen für Bit #20…Bit #31 nicht nur 0, sondern – abhängig vom Bit #19 des eigentlichen 16-Bit-Statuswerts – auch den Wert 1 an den PC zurückgibt, sollten hier ausschließlich die Bit #4…Bit #19 ausgewertet werden (siehe Beispiel-Code unten). – Datentypen (wie die Ist-Position oder die Ist-Geschwindigkeit), die als 20-Bit-Werte mit Vorzeichen an die RTC6 übertragen werden, können dagegen durch Übernahme des kompletten von **get_value** zurückgegebenen 32-Bit-Wertes mit Vorzeichen ausgewertet werden (siehe auch Beispiel-Code unten).

Kontrollbefehl	get_value

Beispiel (C/C++)	Auslesen von unterschiedlichen Datentypen (jeweils Anschluss für ersten Scan-Kopf, x-Achse):
	a) XY2-100-Statuswort, PowerOK-Status
	`UINT statusword, powerOK;`
	`control_command (1, 1, 0x0500); // nur bei iDRIVE-Scan-Systemen sinnvoll`
	`statusword = (get_value(1) & 0x000FFFF0) >> 4;`
	`powerOK = (statusword & 0x00000080);`
	b) nur bei iDRIVE-Scan-Systemen: Seriennummer
	`UINT SN_low, SN_high, SN;`
	`// die unteren 16 Bits der Seriennummer werden zur Rückmeldung ausgewählt`
	`// und mit get_value ausgelesen:`
	`control_command (1, 1, 0x051E);`
	`SN_low = (get_value(1) & 0x000FFFF0)>>4;`
	`// die oberen 16 Bits der Seriennummer werden zur Rückmeldung ausgewählt`
	`// und mit get_value ausgelesen:`
	`control_command (1, 1, 0x051F);`
	`SN_high = (get_value(1) & 0x000FFFF0)>>4;`
	`// Gesamte Seriennummer:`
	`SN = (SN_high << 16) + SN_low;`
	c) nur bei iDRIVE-Scan-Systemen: Ist-Position
	`long real_position;`
	`control_command (1, 1, 0x0501);`
	`real_position = get_value(1);`

RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Alle zurückgegebenen Werte sind auch im RTC4-Kompatibilitätsmodus im RTC6-20-Bit-Bereich, werden jedoch als 32-Bit-Werte an den PC übertragen (siehe oben).
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_values**, **set_trigger**, **get_waveform**, **get_head_status**

Kontrollbefehl	**get_values**
Funktion	Gibt die aktuellen Werte von bis zu 4 ausgewählten Signalen zurück.
Aufruf	`get_values( SignalPtr, ResultPtr )`
Parameter	`SignalPtr` Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von 4 32-Bit-Werten ohne Vorzeichen, in dem die auszugebenden Signaltypen angegeben sind. `ResultPtr` Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von 4 32-Bit-Werten mit Vorzeichen, in dem die aktuellen Werte der bis zu 4 ausgewählten Signale abgelegt werden sollen.
Multi-Board-Befehlsname	**n_get_values**
Hinweise	• Es können bis zu 4 beliebige Signale gleichzeitig ausgelesen werden. Die auswählbaren Signaltypen sind identisch zu **set_trigger** (für den zulässigen Wertebereich, die Signaltypen und entsprechende Hinweise siehe dort). Die auszugebenden Signaltypen müssen unter `SignalPtr` angegeben werden. Die zugehörigen Signalwerte werden dann unter `ResultPtr` abgelegt. Zur Speicherung eines ausgelesenen Datensatzes muss von der Anwendung jeweils (an der über `ResultPtr` angegebenen Adresse) ein Speicherbereich der Größe 4 × 4 Byte auf dem PC bereitgestellt werden. • **get_values** funktioniert ähnlich wie **get_value** (siehe Hinweise dort). Für Kanäle, für die unter `SignalPtr` ein unzulässiger Signaltypwert angegeben wird, führt **get_values** keine Signalabfrage aus und gibt den Wert 0 zurück. Ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` wird nur dann erzeugt, wenn alle 4 angegebenen Signaltypwerte ungültig sind. • Ist mindestens einer der Zeigerparameter gleich NULL, wird **get_values** nicht ausgeführt und ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt.
RTC4→RTC6	Neuer Befehl. Die 4 zurückgegebenen Werte sind auch im RTC4-Kompatibilitätsmodus immer im RTC6-20-Bit-Bereich, werden jedoch als 32-Bit-Werte an den PC übertragen (siehe Hinweise bei **get_value**).
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_value**, **set_trigger**, **transform**

Kontrollbefehl	**get_wait_status**
Funktion	Gibt den Unterbrechungs-Status der RTC6-Karte zurück.
Aufruf	`WaitStatus = get_wait_status()`
Rückgabe	Unterbrechungs-Status. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_get_wait_status**
Hinweise	• Wenn die Listenabarbeitung an einem Unterbrechungspunkt (“wait marker”) gestoppt wurde, dann liefert **get_wait_status** die entsprechende Nummer zurück. Siehe **set_wait**. • Falls kein Unterbrechungspunkt angetroffen wurde, dann gibt **get_wait_status** den Wert Null zurück. • Die Listenabarbeitung wird durch den Aufruf von **release_wait** fortgesetzt.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_wait**, **release_wait**

Kontrollbefehl	**get_waveform**
Funktion	Wie **get_waveform_offset**, aber ohne Parameter `Offset`.
Aufruf	`get_waveform( Channel, Number, Ptr )`
Parameter	`Channel` Wie **get_waveform_offset**.
	`Number` Anzahl der zu übertragenden Werte. Zulässiger Wertebereich: [0…max. Kanalgröße, siehe **set_trigger[*]**]. Übertragen werden die Messwerte von der Position 0 bis (`Number`–1). Als 32-Bit-Wert ohne Vorzeichen.
	`Ptr` Wie **get_waveform_offset**.
Multi-Board-Befehlsname	**n_get_waveform**
Hinweise	• Keine Daten werden übertragen (**get_last_error**-Returncode `RTC6_PARAM_ERROR`): – Für `Number` = 0 – Für `Number` > 2^{22}, falls die Aufzeichnung mit **set_trigger8** gestartet wurde – Für `Number` > 2^{23} × 4 und `Channel` = 1 – Für `Number` > 2^{23} × 2 und `Channel` = 2 – Für `Number` > 2^{23} × 3 und `Channel` = 3 – Für `Number` > 2^{23} × 1 und `Channel` = 4 – Für `Ptr` = NULL. • **get_waveform** ist gleichbedeutend mit: – **get_waveform_offset**`( Channel, 0, Number, Ptr )` • Die Hinweise, Seite 497 zu **get_waveform_offset** gelten analog.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Alle Werte sind auch im RTC4-Kompatibilitätsmodus immer im RTC6-20-Bit-Bereich, werden jedoch als 32-Bit-Werte an den PC übertragen (siehe Hinweise bei **get_value**).
RTC5→RTC6	Grundsätzlich unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 635, OUT 636: `Channel`. Zulässige Werte erweitert auf 8 (vormals: 4).
Verweise	**set_trigger**, **set_trigger4**, **set_trigger8**, **get_value**, **get_values**, **get_waveform_offset**

Kontrollbefehl	**get_waveform_offset**
Funktion	Überträgt die Messwerte, die mit **set_trigger[*]** aufgezeichnet und auf der RTC6 gespeichert wurden, an den PC.
Aufruf	`get_waveform_offset( Channel, Offset, Number, Ptr )`
Parameter	`Channel` Messkanal. Zulässige Werte: 1…2 bei mit **set_trigger** gestarteten Aufzeichnungen. 1…4 bei mit **set_trigger4** gestarteten Aufzeichnungen. 1…8 bei mit **set_trigger8** gestarteten Aufzeichnungen. Als 32-Bit-Wert ohne Vorzeichen.
	`Offset` Startposition. Als 32-Bit-Wert ohne Vorzeichen.. Zulässiger Wertebereich: [0…max. Kanalgröße, siehe **set_trigger[*]**].
	`Number` Anzahl der zu übertragenden Werte. Übertragen werden die Messwerte von der Position `Offset` bis (`Offset` + (`Number`–1)). Als 32-Bit-Wert ohne Vorzeichen.
	`Ptr` Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, an den die Messwerte übertragen werden sollen.
Multi-Board-Befehlsname	**n_get_waveform_offset**
Hinweise	• Keine Daten werden übertragen (**get_last_error**-Returncode `RTC6_PARAM_ERROR`): – Für `Number` = 0 – Für `Number` > 2^{22}, falls die Aufzeichnung mit **set_trigger8** gestartet wurde – Für `Offset` + `Number` > max. Kanalgröße – Für `Ptr` = NULL • PCI-Übertragungsfehler erzeugen einen **get_last_error**-Returncode `RTC6_SEND_ERROR`. • Siehe auch **set_trigger[*]** zur Konfiguration des Kanals. • Vor einem **get_transform_offset**-Aufruf oder **get_waveform_offset**-Aufruf kann mit **measurement_status** überprüft werden, ob noch eine mit **set_trigger[*]** gestartete Aufzeichnung läuft. Zusätzlich kann mit **measurement_status** die Anzahl `Pos` der bei der letzten (oder aktuellen) Messung aufgezeichneten Datenpaare ausgelesen werden. Lesen Sie höchstens `Pos`+1 Datenelemente aus. Alle weiteren Datenelemente stammen aus früheren Aufzeichnungen oder der Initialisierung.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 610, OUT 610, RBF 614. Letzte Änderung DLL 635, OUT 636: `Channel`. Zulässige Werte erweitert auf 8 (vormals: 4).
Verweise	**get_waveform**, **set_trigger**, **set_trigger4**, **set_trigger8**, **get_value**, **get_values**

Kontrollbefehl	**get_z_distance**
Funktion	Gibt den Fokuslängenwert l für den angegebenen Punkt im 3D-Bildfeld zurück.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), liefert **get_z_distance** den Wert 0 zurück und hat sonst keine Wirkung.
Aufruf	`ZDistance = get_z_distance( X, Y, Z )`
Parameter	`X` Absolute x-Koordinate des Punkts (x|y|z) im 3D-Bildfeld. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y` Wie `X` (analog).
	`Z` Wie `X` (analog).
Rückgabe	Fokuslängenwert. [–524.288…+524.287]. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_get_z_distance**
Hinweise	• **get_z_distance** wird nur zur Neu-Kalibrierung der z-Achse in einem 3-Achsen-Scan-System benötigt, siehe Abschnitt "z-Achsen-Kalibrierung überprüfen", Seite 178. • Der Fokuslängenwert l: – Ist einheitenlos – Entspricht dem Fokuslängenunterschied zwischen dem angegebenen Punkt (x|y|z) und dem Punkt (0|0|0) – Kann positiv oder negativ sein • `ZDistance` ist bei der RTC6 immer im 20-Bit-Bereich [–524.288…+524.287]: – Im RTC4-Kompatibilitätsmodus – Im RTC5-Kompatibilitätsmodus – Im RTC6-Standardmodus Wichtig: Wenn Sie nicht mit **load_z_table_20b** oder **load_z_table_no_20b** arbeiten, müssen Sie `ZDistance` durch 16 teilen und das Ergebnis in die Parabolische Funktion $z_{out} = A + Bl + Cl^2$ einsetzen. • **get_z_distance** führt zunächst einen (virtuellen) Sprung zum Punkt (x|y|z) aus und liefert dann den Fokuslängenwert zurück. • **get_z_distance** wird nicht ausgeführt und liefert den Wert 0 zurück (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **get_z_distance** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Zur 3D-Bildfeld-Kalibrierung siehe Kapitel "3D-Befehle", Seite 250.

Kontrollbefehl	get_z_distance
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X`, `Y` und `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**load_z_table**, **load_z_table_no**, **load_z_table_20b**, **load_z_table_no_20b**

Kontrollbefehl	goto_xy	
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.	
Aufruf	`goto_xy( X, Y )`	
Parameter	`X`	Absolute x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y`	Wie `X` (analog).
Multi-Board-Befehlsname	**n_goto_xy**	
Hinweise	• Wenn die Sprunggeschwindigkeit zuvor nicht explizit über **set_jump_speed** oder **set_jump_speed_ctrl** festgelegt wurde, wird der Sprung mit der voreingestellten Sprunggeschwindigkeit von 10.000 *Bits/ms* ausgeführt. • **goto_xy** hat im Gegensatz zu den Listenbefehlen **jump_abs** oder **jump_rel** keine Auswirkung auf die Lasersteuersignale und setzt auch kein Jump-Delay. • Zuvor angesammelte Koordinatentransformationen, siehe Listenpunkt "Mit `at_once` = 2…", Seite 238, werden durch **goto_xy** wirksam. • **goto_xy** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist – aktuell ein /STOP, /STOP2 oder /Slave-STOP anliegt Dieses kann auch durch eine automatische Überwachung generiert werden, siehe **set_laser_control** und **range_checking**. • **goto_xy** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • **goto_xy** setzt den INTERNAL-BUSY-Listenausführungsstatus für die Dauer seiner Ausführung. • **goto_xy** kehrt erst zum Anwenderprogramm zurück, wenn die Bewegung abgearbeitet ist.	
RTC4→RTC6	• Erhöhter Wertebereich. • **goto_xy** kehrt erst nach Abarbeitung der Bewegung zurück (siehe Hinweis oben). • Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**set_jump_speed**, **jump_abs**, **jump_rel**, **goto_xyz**, **get_status**	

Kontrollbefehl	goto_xyz	
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld.	
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **goto_xyz** wie **goto_xy**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.	
Aufruf	`goto_xyz( X, Y, Z )`	
Parameter	`X`	Absolute x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y`	Wie `X` (analog).
	`Z`	Wie `X` (analog).
Multi-Board-Befehlsname	**n_goto_xyz**	
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **goto_xyz** gleich wie **goto_xy** (siehe Hinweise dort). • Der Parameter `DirectMove3D` von **set_delay_mode** bestimmt die Art der z-Achsen-Bewegung (linear oder schrittweise korrigiert). • Siehe auch Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189.	
RTC4→RTC6	• Erhöhter Wertebereich. • **goto_xyz** kehrt erst nach Ende der Bewegung zurück (siehe Hinweis bei **goto_xy**). • RTC4-Kompatibilitätsmodus: siehe **goto_xy**.	
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**goto_xy**, **jump_abs_3d**, **jump_rel_3d**	

Kontrollbefehl	**home_position**
Funktion	Aktiviert den Home-Jump-Modus (für die x- und y-Achse) und definiert die Home-Position.
Aufruf	`home_position( XHome, YHome )`
Parameter	`XHome` Absolute x-Koordinate der Home-Position. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Zu große Werte werden geclippt.
	`YHome` Wie `XHome` (analog).
Multi-Board-Befehlsname	**n_home_position**
Hinweise	• **home_position** definiert die Koordinaten für einen Home-Jump, der z.B. beim Beenden einer Liste ausgeführt wird. Entsprechend wird dann beim Start wieder ein Home-Return zur zuletzt gültigen Position ausgeführt. Home-Jump und Home-Return werden mit Sprunggeschwindigkeit ausgeführt. • **home_position** ist für Lasersysteme vorgesehen, die kein schnelles Schalten des Lasers erlauben. Nach Aufruf von **home_position** bewegt sich der Laserfokus immer dann zur angegebenen Home-Position, wenn gerade keine Liste ausgeführt wird oder eine Liste mit **set_wait** angehalten wurde. Ein Home-Jump wird auch ausgeführt, wenn eine Listenausführung mit **stop_execution** oder einem externen Stoppsignal abgebrochen wird. Der Home-Jump selbst kann nicht abgebrochen werden, ein Home-Return dagegen schon. • Für die Dauer der Ausführung eines Home-Jumps oder Home-Returns wird der INTERNAL-BUSY-Listenausführungsstatus gesetzt. • An die Home-Position sollte eine *Strahlenfalle* (engl. “beam dump”) angebracht werden. • Der Home-Jump-Modus wird mit `home_position(0, 0)` deaktiviert, auch wenn er mit **home_position_xyz** aktiviert wurde. • **home_position** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 “Automatisches Booten – Ablauf im Detail”, Seite 1018.
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `XHome` und `YHome` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**home_position_xyz**

Kontrollbefehl	**home_position_xyz**
Funktion	Aktiviert den Home-Jump-Modus (für die x-, y- und z-Achse) und definiert die Home-Position.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **home_position_xyz** wie **home_position**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`home_position_xyz( XHome, YHome, ZHome )`
Parameter	`XHome` Absolute x-Koordinate der Home-Position. In Bits. Als 32-Bit-Wert mit Vorzeichen.. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`YHome` Wie `XHome` (analog).
	`ZHome` Wie `XHome` (analog).
Multi-Board-Befehlsname	**n_home_position_xyz**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **home_position_xyz** gleich wie **home_position** (siehe Hinweise dort). • Der Home-Jump-Modus wird deaktiviert mit: – `home_position( 0, 0 )` – `home_position_xyz( 0, 0, 0)` • Der Parameter `DirectMove3D` von **set_delay_mode** bestimmt die Art der z-Achsen-Bewegung (linear oder schrittweise korrigiert). • **home_position_xyz** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `XHome`, `YHome` und `ZHome` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den für `ZHome` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**home_position**

Unverzögerter Kurzer Listenbefehl	**if_cond**
Funktion	*Bedingte Befehlsausführung*: **if_cond** führt sofort den unmittelbar folgenden Listenbefehl aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_cond( Mask1, Mask0 )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
Multi-Board-Befehlsname	**n_if_cond**
Hinweise	• Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**if_not_cond**, **if_pin_cond**, **if_not_pin_cond**

Unverzögerter Kurzer Listenbefehl	**if_fly_x_overflow**
Funktion	*Bedingte Befehlsausführung für "Processing on the fly"-Anwendungen*: **if_fly_x_overflow** führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die x-Achse erfüllt ist. Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_fly_x_overflow( Mode )`
Parameter	`Mode` Zu überprüfende Bedingung. Als 32-Bit-Wert mit Vorzeichen. = 0: Es hat irgendeine Bereichsüberschreitung stattgefunden (Fehler-Bit Bit #4 = 1 oder Fehler-Bit Bit #5 = 1). > 0: Es hat ein Overflow stattgefunden (Fehler-Bit Bit #5 = 1). < 0: Es hat ein Underflow stattgefunden (Fehler-Bit Bit #4 = 1).
Multi-Board-Befehlsname	**n_if_fly_x_overflow**
Hinweise	• Zur Verwendung von **if_fly_x_overflow** siehe Abschnitt "Kundenspezifischer Überwachungsbereich", Seite 268. • Die gemäß `Mode` abgefragten Fehler-Bits (Fehler-Bit Bit #4 und/oder Fehler-Bit Bit #5) werden zurückgesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **clear_fly_overflow_ctrl**, **if_not_fly_x_overflow**

Unverzögerter Kurzer Listenbefehl	**if_fly_y_overflow**
Funktion	*Bedingte Befehlsausführung für "Processing on the fly"-Anwendungen*: **if_fly_y_overflow** führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die y-Achse erfüllt ist. Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_fly_y_overflow( Mode )`
Parameter	`Mode` Zu überprüfende Bedingung. Als 32-Bit-Wert mit Vorzeichen. = 0: Es hat irgendeine Bereichsüberschreitung stattgefunden (Fehler-Bit Bit #6 = 1 oder Fehler-Bit Bit #7 = 1). > 0: Es hat ein Overflow stattgefunden (Fehler-Bit Bit #7 = 1). < 0: Es hat ein Underflow stattgefunden (Fehler-Bit Bit #6 = 1).
Multi-Board-Befehlsname	**n_if_fly_y_overflow**
Hinweise	• Zur Verwendung von **if_fly_y_overflow** siehe Abschnitt "Kundenspezifischer Überwachungsbereich", Seite 268. • Die gemäß `Mode` abgefragten Fehler-Bits (Fehler-Bit Bit #6 und/oder Fehler-Bit Bit #7) werden zurückgesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **clear_fly_overflow_ctrl**, **if_not_fly_y_overflow**

Unverzögerter Kurzer Listenbefehl	**if_fly_z_overflow**
Funktion	*Bedingte Befehlsausführung für "Processing on the fly"-Anwendungen*: **if_fly_z_overflow** führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die z-Achse erfüllt ist. Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_fly_z_overflow( Mode )`
Parameter	`Mode` Zu überprüfende Bedingung. Als 32-Bit-Wert mit Vorzeichen. = 0: Es hat irgendeine Bereichsüberschreitung stattgefunden (Fehler-Bit Bit #24 = 1 oder Fehler-Bit Bit #25 = 1). > 0: Es hat ein Overflow stattgefunden (Fehler-Bit Bit #25 = 1). < 0: Es hat ein Underflow stattgefunden (Fehler-Bit Bit #24 = 1).
Multi-Board-Befehlsname	**n_if_fly_z_overflow**
Hinweise	• Zur Verwendung von **if_fly_z_overflow** siehe auch Kapitel 8.6.9 ""Processing on the fly"-Korrektur überwachen", Seite 267. • Die gemäß `Mode` abgefragten Fehler-Bits (Fehler-Bit Bit #24 und/oder Fehler-Bit Bit #25) werden zurückgesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**, **set_fly_limits_z**, **clear_fly_overflow**, **clear_fly_overflow_ctrl**, **if_not_fly_z_overflow**

Unverzögerter Kurzer Listenbefehl	**if_not_activated**
Funktion	Bedingte Befehlsausführung aufgrund eines Fehler-Bits aus activate_fly_xy/activate_fly_xy_encoder oder activate_fly_2d/activate_fly_2d_encoder: Wenn das Fehler-Bit gesetzt ist, wird der unmittelbar nächste Listenbefehl ausgeführt, ansonsten übersprungen.
Aufruf	`if_not_activated()`
Multi-Board-Befehlsname	**n_if_not_activated**
Hinweise	• Siehe auch Hinweise zu activate_fly_2d/activate_fly_2d_encoder und activate_fly_xy/activate_fly_xy_encoder. • Sinnvollerweise befindet sich in der Liste unmittelbar hinter **if_not_activated** ein Listensprung oder ein Unterprogramm-Aufruf, welches zu einer Fehlerbehandlungssequenz verzweigt, oder vielleicht auch gleich ein set_end_of_list. • **if_not_activated** löscht das Fehler-Bit aus activate_fly_xy/activate_fly_xy_encoder oder activate_fly_2d/activate_fly_2d_encoder. Wenn es zur späteren Abfrage mittels `get_marking_info( Bit #9 )` noch benötigt wird, kann in der Fehlerbehandlungssequenz activate_fly_2d/activate_fly_2d_encoder oder activate_fly_xy/activate_fly_xy_encoder erneut aufgerufen werden, um das Fehler-Bit wieder zu setzen (vorausgesetzt, die fehlerhafte Situation besteht noch). • Eine erfolgreiche Aktivierung mit activate_fly_2d/activate_fly_2d_encoder oder activate_fly_xy/activate_fly_xy_encoder löscht ein bereits zuvor gesetztes Fehler-Bit *nicht*. Es bleibt für get_marking_info stehen, bis get_marking_info aufgerufen wurde.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	activate_fly_2d, activate_fly_2d_encoder, activate_fly_xy, activate_fly_xy_encoder, get_marking_info

Unverzögerter Kurzer Listenbefehl	**if_not_cond**
Funktion	*Bedingte Befehlsausführung*: **if_not_cond** führt sofort den unmittelbar folgenden Listenbefehl aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung *nicht* erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` *nicht* gleich `1` *oder* die in `Mask0` gesetzten Bits von `IOvalue` *nicht* gleich `0` sind). Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_not_cond( Mask1, Mask0 )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
Multi-Board-Befehlsname	**n_if_not_cond**
Hinweise	• Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**if_cond**, **if_pin_cond**, **if_not_pin_cond**

Unverzögerter Kurzer Listenbefehl	**if_not_fly_x_overflow**
Funktion	*Bedingte Befehlsausführung für "Processing on the fly"-Anwendungen*: **if_not_fly_x_overflow** führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die x-Achse *nicht* erfüllt ist. Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_not_fly_x_overflow( Mode )`
Parameter	`Mode` Zu überprüfende Bedingung. Als 32-Bit-Wert mit Vorzeichen. = 0: Es hat irgendeine Bereichsüberschreitung stattgefunden (Fehler-Bit Bit #4 = 1 oder Fehler-Bit Bit #5 = 1). > 0: Es hat ein Overflow stattgefunden (Fehler-Bit Bit #5 = 1). < 0: Es hat ein Underflow stattgefunden (Fehler-Bit Bit #4 = 1).
Multi-Board-Befehlsname	**n_if_not_fly_x_overflow**
Hinweise	• Zur Verwendung von **if_not_fly_x_overflow** siehe Abschnitt "Kundenspezifischer Überwachungsbereich", Seite 268. • Die gemäß `Mode` abgefragten Fehler-Bits (Fehler-Bit Bit #4 und/oder Fehler-Bit Bit #5) werden zurückgesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **clear_fly_overflow_ctrl**, **if_fly_x_overflow**

Unverzögerter Kurzer Listenbefehl	**if_not_fly_y_overflow**
Funktion	*Bedingte Befehlsausführung für "Processing on the fly"-Anwendungen*: **if_not_fly_y_overflow** führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die y-Achse *nicht* erfüllt ist. Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_not_fly_y_overflow( Mode )`
Parameter	`Mode` Zu überprüfende Bedingung. Als 32-Bit-Wert mit Vorzeichen. = 0: Es hat irgendeine Bereichsüberschreitung stattgefunden (Fehler-Bit Bit #6 = 1 oder Fehler-Bit Bit #7 = 1). > 0: Es hat ein Overflow stattgefunden (Fehler-Bit Bit #7 = 1). < 0: Es hat ein Underflow stattgefunden (Fehler-Bit Bit #6 = 1).
Multi-Board-Befehlsname	**n_if_not_fly_y_overflow**
Hinweise	• Zur Verwendung von **if_not_fly_y_overflow** siehe Abschnitt "Kundenspezifischer Überwachungsbereich", Seite 268. • Die gemäß `Mode` abgefragten Fehler-Bits (Fehler-Bit Bit #6 und/oder Fehler-Bit Bit #7) werden zurückgesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **clear_fly_overflow_ctrl**, **if_fly_y_overflow**

Unverzögerter Kurzer Listenbefehl	**if_not_fly_z_overflow**
Funktion	*Bedingte Befehlsausführung für "Processing on the fly"-Anwendungen:* **if_not_fly_z_overflow** führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die z-Achse *nicht* erfüllt ist. Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_not_fly_z_overflow( Mode )`
Parameter	`Mode` Zu überprüfende Bedingung. Als 32-Bit-Wert mit Vorzeichen. = 0: Es hat irgendeine Bereichsüberschreitung stattgefunden (Fehler-Bit Bit #24 = 1 oder Fehler-Bit Bit #25 = 1). > 0: Es hat ein Overflow stattgefunden (Fehler-Bit Bit #25 = 1). < 0: Es hat ein Underflow stattgefunden (Fehler-Bit Bit #24 = 1).
Multi-Board-Befehlsname	**n_if_not_fly_z_overflow**
Hinweise	• Zur Verwendung von **if_not_fly_z_overflow** siehe auch Kapitel 8.6.9 ""Processing on the fly"-Korrektur überwachen", Seite 267. • Die gemäß `Mode` abgefragten Fehler-Bits (Fehler-Bit Bit #24 und/oder Fehler-Bit Bit #25) werden zurückgesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**, **set_fly_limits_z**, **clear_fly_overflow**, **clear_fly_overflow_ctrl**, **if_fly_z_overflow**

Unverzögerter Kurzer Listenbefehl	**if_not_pin_cond**
Funktion	*Bedingte Befehlsausführung*: **if_not_pin_cond** führt sofort den unmittelbar folgenden Listenbefehl aus, falls der aktuelle Wert (`IOvalue`) am 2 Bit-Digital-Eingang der LASER-Buchse die folgende Bedingung *nicht* erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` *nicht* gleich `1` *oder* die in `Mask0` gesetzten Bits von `IOvalue` *nicht* gleich `0` sind). Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_not_pin_cond( Mask1, Mask0 )`
Parameter	`Mask1` 2-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 2 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
Multi-Board-Befehlsname	**n_if_not_pin_cond**
Hinweise	• Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**if_pin_cond**, **if_cond**, **if_not_cond**

Unverzögerter Kurzer Listenbefehl	**if_pin_cond**
Funktion	*Bedingte Befehlsausführung*: **if_pin_cond** führt sofort den unmittelbar folgenden Listenbefehl aus, falls der aktuelle Wert (`IOvalue`) am 2 Bit-Digital-Eingang der LASER-Buchse die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird dieser Listenbefehl übersprungen.
Aufruf	`if_pin_cond( Mask1, Mask0 )`
Parameter	`Mask1` 2-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 2 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
Multi-Board-Befehlsname	**n_if_pin_cond**
Hinweise	• Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**if_not_pin_cond**, **if_cond**, **if_not_cond**

Kontrollbefehl	**init_fly_2d**
Funktion	Initialisiert die Encoderreferenzwerte für die 2D-Encoderkompensation einer nachfolgend mit **set_fly_2d** eingestellten "Processing on the fly"-Anwendung und setzt beide Encoderwerte zurück.
Aufruf	`init_fly_2d( OffsetX, OffsetY, No )`
Parameter	`OffsetX` Referenzwert des x-Achsen-Encoders. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: abhängig von der mit **load_fly_2d_table** geladenen Kompensationstabelle. Defaultwert nach **load_program_file** = 0.
	`OffsetY` Wie `OffsetX` (analog).
	`No` Nummer der zu verwendenden 2D-Kompensationstabelle. Siehe auch Seite 262.
Multi-Board-Befehlsname	**n_init_fly_2d**
Hinweise	• Der endgültig für die 2D-Korrektur verwendete Encoderwert + Referenzwert darf den zulässigen Wertebereich nicht überschreiten. Anderenfalls erfolgt ein Clipping (siehe auch **load_fly_2d_table**). • `No` = 1: verwende Tabelle 1. • `No` = 2: verwende Tabelle 2. • `No` = 0: verwende die bisherige Tabelle. • `No` > 2: verwende keine Tabelle. • Die zu verwendenden Tabellen müssen zuvor mit **load_fly_2d_table** gültig geladen worden sein. • Mit `No` > 2 bleibt die Tabelle gültig geladen. • Mit `load_fly_2d_table( Name = NULL )` wird die Tabelle ungültig. • **init_fly_2d** ist in Boot-Phase 2 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Veränderte Funktionalität. • Parameter `No`.
Versionsinfo	Verfügbar ab DLL 614, OUT 614, RBF 619.
Verweise	**init_fly_2d_list**, **set_fly_2d**, **load_fly_2d_table**, **get_fly_2d_offset**

Normaler Listenbefehl	**init_fly_2d_list**
Funktion	Wie **init_fly_2d**, aber ein Listenbefehl.
Aufruf	`init_fly_2d_list( OffsetX, OffsetY, No )`
Parameter	`OffsetX` Wie **init_fly_2d**.
	`OffsetY` Wie **init_fly_2d**.
	`No` Wie **init_fly_2d**.
Multi-Board-Befehlsname	**n_init_fly_2d_list**
Hinweise	• Siehe **init_fly_2d**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 643, OUT 645.
Verweise	**init_fly_2d**

Kontrollbefehl	init_rtc6_dll
Funktion	Initialisiert die Ansteuerung der installierten RTC6-Karten für ein Anwenderprogramm.
Aufruf	`InitErrorNo = init_rtc6_dll()`
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. Sind mehrere Fehler gleichzeitig aufgetreten, dann sind mehrere Bits gesetzt. Die Zuordnung zwischen Bitnummern, Fehlertypen und Fehlerkonstanten ist identisch zu **get_error**.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• **init_rtc6_dll** *muss von jedem Anwenderprogramm zu Beginn aufgerufen werden*, damit von dem Anwenderprogramm überhaupt eine RTC6-Karte ansprechbar ist. • **init_rtc6_dll** sucht nach sämtlichen vorhandenen RTC6 PCIe-Karten (nicht nach RTC6 Ethernet-Karten! – dazu siehe Kapitel 16.5.2 "Über die Suche nach RTC6 Ethernet-Karten", Seite 1007) und baut die RTC6-Kartenverwaltung auf. Wenn keine RTC6 PCIe-Karte gefunden wird (z.B. falls ausschließlich RTC6 Ethernet-Karten betrieben werden), dann wird der Fehlercode `RTC6_NO_PCIE_CARD_FOUND` zurückgegeben (siehe **get_error**). • **init_rtc6_dll** vergibt dem Anwenderprogramm automatisch (wie mit **acquire_rtc**) die Zugriffsberechtigung für die gefundenen Karten, wenn die Zugriffsberechtigung nicht bereits an ein anderes Anwenderprogramm vergeben wurde (es können zwar beliebig viele RTC6-Karten oder Anwenderprogramme gleichzeitig verwendet werden, ein und dieselbe Karte kann aber nicht gleichzeitig von mehreren Anwenderprogrammen verwendet werden). Die erste Initialisierung akquiriert alle gefundenen Karten für sich. Bei nachfolgenden, durch andere Anwenderprogramme gestarteten Initialisierungen liefert **init_rtc6_dll** dann den Fehlercode `RTC6_ACCESS_DENIED` zurück. Die Karten sind dann von diesen Anwenderprogrammen nur mit ausschließlich RTC6-DLL-intern wirkenden Funktionen ansprechbar, die keine Zugriffsberechtigung erfordern – z.B. mit **get_error**, **get_last_error** oder **select_rtc** (auch die meisten Multi-Board-Befehle sind nur auf zugriffsberechtigten Karten ausführbar). Hat ein Anwenderprogramm die Zugriffsberechtigung für eine Karte erhalten, dann kann diese Karte erst dann wieder – mit **init_rtc6_dll** oder **acquire_rtc** – von einem anderen Anwenderprogramm beansprucht werden, nachdem die Karte mit **release_rtc** oder **free_rtc6_dll** explizit vom zugriffsberechtigten Anwenderprogramm freigegeben wurde. Der Fehlercode `RTC6_ACCESS_DENIED` wird zurückgegeben, wenn mindestens eine der gefundenen Karten den Zugriff verweigert hat. Welche das ist oder sind, kann mit `n_get_error( CardNo )` (`CardNo` von 1 bis zur Zahl der gefundenen Karten) oder unmittelbar nach **init_rtc6_dll** (noch vor dem ersten anderen Befehl) mit `n_get_last_error( CardNo )` abgefragt werden. • Beim Übernehmen einer Karte durch **init_rtc6_dll** wird (wie mit **acquire_rtc**) eine Versionsprüfung durchgeführt. Liegt ein Versionsfehler vor, wird der Zugriff auf die Karte verweigert (Rückgabewert `RTC6_ACCESS_DENIED`|`RTC6_VERSION_MISMATCH`). • Es kann zur gleichen Zeit immer nur ein einziges Anwenderprogramm die Initialisierung durchführen. Durch andere Anwenderprogramme später gestartete Initialisierungen warten, bis die aktuelle Initialisierung abgelaufen ist.

Kontrollbefehl	init_rtc6_dll
Hinweise (Forts.)	• Wird **init_rtc6_dll** von demselben Anwenderprogramm mehrmalig aufgerufen, so wird die beim vormaligen Aufruf des Befehls aufgebaute RTC6-Kartenverwaltung für dieses Anwenderprogramm gelöscht und die dabei an das Anwenderprogramm vergebenen Zugriffsberechtigungen aufgehoben, bevor die neue RTC6-Kartenverwaltung aufgebaut wird und Zugriffsberechtigungen wieder neu vergeben werden. • Bei jeder Initialisierung eines Anwenderprogramms mit **init_rtc6_dll** werden RTC6-DLL-interne Nummern für alle gefundenen RTC6-Karten neu vergeben. Die Zuordnung zwischen den RTC6-DLL-internen Nummern und den installierten Karten kann über **get_serial_number** ermittelt werden. Die Karte mit der kleinsten RTC6-DLL-internen Nummer aller zugriffsberechtigten Karten wird bei der Initialisierung gleichzeitig zur aktiven Karte und damit zum Ziel für Nicht-Multi-Board-Befehle. Ein Wechsel der aktiven Karte ist jederzeit über **select_rtc** möglich. Besteht für keine Karte eine Zugriffsberechtigung, wird die Karte mit der höchsten RTC6-DLL-internen Nummer (siehe **rtc6_count_cards**) zur aktiven Karte. Im letzteren Fall können dann aber ausschließlich rein RTC6-DLL-interne Befehle ausgeführt werden, die keine Zugriffsberechtigung erfordern. • Beachten Sie auch Kapitel 6.7.1 "Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm", Seite 131. • **init_rtc6_dll** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Nach der Initialisierung der RTC6-DLL mit **init_rtc6_dll** ist standardmäßig der RTC6-Standardmodus voreingestellt. Danach kann ein anderer RTC6-DLL-Betriebsmodus eingestellt werden, siehe **set_rtc4_mode** und **set_rtc5_mode**. • **init_rtc6_dll** löst keine Initialisierung von RTC6-Karten aus. Nur **load_program_file** führt eine Initialisierung von RTC6-Karten durch.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl. Die Funktionalitäten von **init_rtc6_dll** und dem RTC5-Befehl **init_rtc5_dll** sind identisch.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_rtc4_mode**, **set_rtc5_mode**

Normaler Listenbefehl	**jump_abs**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`jump_abs( X, Y )`
Parameter	`X` Absolute x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`Y` Wie `X` (analog).
Multi-Board-Befehlsname	**n_jump_abs**
Hinweise	• Wenn die Sprunggeschwindigkeit zuvor nicht explizit mit **set_jump_speed** oder **set_jump_speed_ctrl** festgelegt wurde, wird der Sprung mit der voreingestellten Sprunggeschwindigkeit von 10.000 *Bits/ms* ausgeführt. • Die Signale für den "Laser active"-Betrieb werden vor dem Sprung ausgeschaltet und bleiben während des Sprungs ausgeschaltet. • Nach einem Sprung-Befehl wird ein (variables) Jump-Delay eingefügt. Ausnahme: Das einem Sprung-Vektor der Länge 0 folgende (variable) Jump-Delay wird nicht ausgeführt. **jump_abs** selber benötigt dafür aber dennoch einen 10 μs-Takt zur Ausführung. • Zuvor angesammelte Koordinatentransformationen, siehe Listenpunkt "Mit `at_once` = 2…", Seite 238, werden durch **jump_abs** wirksam.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_jump_speed**, **set_scanner_delays**, **jump_rel**, **timed_jump_abs**, **goto_xy**

Normaler Listenbefehl	**jump_abs_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **jump_abs_3d** wie **jump_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`jump_abs_3d( X, Y, Z )`
Parameter	`X` Absolute x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`Y` Wie `X` (analog).
	`Z` Wie `X` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287].
Multi-Board-Befehlsname	**n_jump_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **jump_abs_3d** gleich wie **jump_abs** (siehe Hinweise dort). • Der Parameter `DirectMove3D` von **set_delay_mode** bestimmt die Art der z-Achsen-Bewegung (linear oder schrittweise korrigiert).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X`, `Y` und `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_abs**, **jump_rel_3d**, **timed_jump_abs_3d**

Normaler Listenbefehl	**jump_abs_drill**
Aufruf	`jump_abs_drill( X, Y, DrillTime )`
Multi-Board-Befehlsname	**n_jump_abs_drill**
Hinweise	• Dieser Befehl ist z.B. im Handbuch für das intelli*DRILL* $_{de}$ II 20 Modul beschrieben.

Normaler Listenbefehl	**jump_abs_drill_2**
Aufruf	`jump_abs_drill_2( X, Y, DrillTime, XOff, YOff )`
Multi-Board-Befehlsname	**n_jump_abs_drill_2**
Hinweise	• Dieser Befehl ist z.B. im Handbuch für das intelli*DRILL* $_{de}$ II 20 Modul beschrieben.

Normaler Listenbefehl	**jump_rel**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`jump_rel( dX, dY )`
Parameter	`dX` Relative x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. `dY` Wie `dX` (analog).
Multi-Board-Befehlsname	**n_jump_rel**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **jump_rel** identisch mit **jump_abs** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_abs**, **jump_rel_3d**, **timed_jump_rel**

Normaler Listenbefehl	**jump_rel_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **jump_rel_3d** wie **jump_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`jump_rel_3d( dX, dY, dZ )`
Parameter	`dX` Relative x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`dY` Wie `dX` (analog).
	`dZ` Wie `dX` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287].
Multi-Board-Befehlsname	**n_jump_rel_3d**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **jump_rel_3d** identisch mit **jump_abs_3d** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX`, `dY` und `dZ` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_abs_3d**, **jump_abs**, **jump_rel**, **timed_jump_rel_3d**

Normaler Listenbefehl	**jump_rel_drill**
Aufruf	`jump_rel_drill( dX, dY, DrillTime )`
Multi-Board-Befehlsname	**n_jump_rel_drill**
Hinweise	• Dieser Befehl ist z.B. im Handbuch für das intelli*DRILL* $_{de}$ II 20 Modul beschrieben.

Normaler Listenbefehl	**jump_rel_drill_2**
Aufruf	`jump_rel_drill_2( dX, dY, DrillTime, XOff, YOff )`
Multi-Board-Befehlsname	**n_jump_rel_drill_2**
Hinweise	• Dieser Befehl ist z.B. im Handbuch für das intelli*DRILL* $_{de}$ II 20 Modul beschrieben.

Variabler Listenbefehl	**laser_on_list**
Funktion	Schaltet die Signale für den "Laser active"-Betrieb für die angegebene Zeit ein.
Aufruf	`laser_on_list( Period )`
Parameter	`Period` Zeitintervall. In Bits. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 μs. Zulässiger Wertebereich: 0 ≤ `Period` ≤ (2^{31}–1).
Multi-Board-Befehlsname	**n_laser_on_list**
Hinweise	• Bit #31 von `Period` wird ignoriert (siehe auch Hinweise zu **laser_on_pulses_list**). • Die Signale für den "Laser active"-Betrieb müssen erst mit **set_laser_mode** ausgewählt, deren Parameter mit weiteren Befehlen angegeben und mit **set_laser_control** oder **enable_laser** freigegeben werden, bevor sie mit **laser_on_list** eingeschaltet werden können, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192. • Während die Lasersteuersignale eingeschaltet sind, bleibt die Position des Scanners unverändert. Der nachfolgende Listenbefehl wird ausgeführt, sobald das angegebene Zeitintervall verstrichen ist. • Das aktuell eingestellte LaserOn-Delay wird am Anfang des programmierten Zeitintervalls ausgeführt: die Lasersteuersignale werden nach einem LaserOn-Delay eingeschaltet. • Die Lasersteuersignale werden wie bei anderen [*]mark[*]-Befehlen (z.B. **mark_abs**) am Ende des Zeitintervalls *nicht* explizit ausgeschaltet, sondern erst durch einen nachfolgenden Listenbefehl (z.B. **jump_abs** oder **list_nop**). Das aktuell eingestellte LaserOff-Delay wird dabei berücksichtigt. • **laser_on_list** ist nützlich für das Markieren einzelner Punkte, siehe Kapitel 7.1.3 "Einzelne Punkte markieren", Seite 144. • Der Wobbelmode (siehe Kapitel 8.4 "Wobbelmode", Seite 244) bleibt erhalten, wird aber nicht berücksichtigt. • Für `Period` = 0 gehen die Lasersteuersignale nicht an. **laser_on_list** ist dann ein Kurzer Listenbefehl. • **laser_on_list** löst kein Scanner-Delay aus. Wenn vor einem Sprung gewartet werden soll, bis der Laser (nach einem LaserOff-Delay) tatsächlich aus ist, dann sollte explizit ein **long_delay** eingefügt werden.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**laser_on_pulses_list**, **long_delay**, **para_laser_on_pulses_list**

Variabler Listenbefehl	**laser_on_pulses_list**
Funktion	Schaltet das LASERON-Lasersteuersignal für die angegebene Anzahl von externen Signalpulsen, jedoch maximal für das angegebene Zeitintervall ein. **laser_on_pulses_list** wirkt für `Pulses` > 65.535 wie **laser_on_list**.
Aufruf	`laser_on_pulses_list( Period, Pulses )`
Parameter	`Period` Zeitintervall. In Bits. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 μs. Zulässiger Wertebereich: 0 ≤ `Period` ≤ (2^{32}–1). `Pulses` Anzahl externer Signalpulse. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: 0 ≤ `Pulses` ≤ 65.535 oder größer (siehe Hinweis unten).
Multi-Board-Befehlsname	**n_laser_on_pulses_list**
Hinweise	• **laser_on_pulses_list** ist nützlich für das Markieren separater Punkte im Laser-Mode 6, ist aber auch in allen anderen Laser-Modi wirksam. • Die externen Signalpulse müssen als TTL-Pulse an DIGITAL IN1 der LASER-Buchse (siehe Abschnitt "2-Bit-Digital-Eingang", Seite 77) angelegt werden. Mit **set_laser_control**(Bit #5) kann eingestellt werden, ob die Signalpulse jeweils mit der steigenden oder der fallenden Flanke gezählt werden. • Für `Period` = 0 hat **laser_on_pulses_list** keine Wirkung. **laser_on_pulses_list** ist dann ein Kurzer Listenbefehl. • Für 0 < `Period` ≤ (2^{31}–1) dauert der Befehl immer genau `Period` Takte (`Period` × 10 μs), auch wenn die angegebene Anzahl von externen Signalpulsen in kürzerer Zeit abläuft. • Für 2^{31} ≤ `Period` ≤ (2^{32}–1) dauert der Befehl maximal (`Period` – 2^{31}) Takte ((`Period` – 2^{31}) × 10 μs). Hier wird der Befehl jedoch vorzeitig abgebrochen, sobald die angegebene Anzahl von externen Signalpulsen detektiert wurde. • Für `Pulses` > 65.535 wirkt **laser_on_pulses_list** identisch zu **laser_on_list** (hier werden keine externen Signalpulse berücksichtigt; siehe auch Hinweise dort). Ansonsten (0 ≤ `Pulses` ≤ 65.535) schaltet **laser_on_pulses_list** (im Gegensatz zu **laser_on_list**) *nicht* zwischen den Lasersteuersignalen von "Laser active"- und "Laser standby"-Betrieb hin und her, sondern schaltet nur das LASERON-Signal und berücksichtigt keine Laser-Delays. Auch in vorherigen Befehlen eventuell aktivierte, aber noch nicht abgelaufene Laser-Delays sind während dieses Befehls nicht wirksam (jedoch sofort dann wieder, wenn das LASERON-Signal nach dem letzten Puls oder nach `Period` wieder ausgeschaltet wird).

Variabler Listenbefehl	**laser_on_pulses_list**
Hinweise (Forts.)	• Für 1 ≤ `Pulses` ≤ 65.535 wird das LASERON-Signal mit der Flanke (entsprechend der mit **set_laser_control**(Bit #5) eingestellten Polarität) des ersten externen Pulses eingeschaltet (es sei denn, es ist bereits aufgrund eines noch nicht abgelaufenen LaserOff-Delays eingeschaltet) und bleibt dann für die angegebene Zahl von externen Pulsen eingeschaltet, längstens jedoch bis zum Ende der mit `Period` angegebenen Zahl von 10 µs-Perioden. Es liegt in der Verantwortung des Anwenders, `Period` so groß zu wählen, dass in dieser Zeit auch `Pulses` Signalpulse abgearbeitet werden können. Werden am DIGITAL IN1-Eingang keine Pulse eingegeben, so lässt **laser_on_pulses_list** das LASERON-Signal unverändert. Werden innerhalb des angegebenen Zeitintervalls `Period` mehr als `Pulses` Signalpulse eingegeben, so werden die überzähligen ignoriert. • Für `Pulses` = 0 bleibt **laser_on_pulses_list** ohne Auswirkung (das LASERON-Signal wird nicht eingeschaltet). **laser_on_pulses_list** ist dann ein Kurzer Listenbefehl. • Das LASERON-Signal muss erst mit **set_laser_control** oder **enable_laser** angegeben und freigegeben werden, bevor es mit **laser_on_pulses_list** eingeschaltet werden kann (siehe Kapitel 7.4 "Laser-Steuerung", Seite 192). • Während der **laser_on_pulses_list**-Ausführung bleibt die Position des Scanners unverändert. Der nachfolgende Listenbefehl wird erst ausgeführt, wenn das angegebene Zeitintervall `Period` verstrichen ist. • Der Wobbelmode (siehe Kapitel 8.4 "Wobbelmode", Seite 244) bleibt erhalten, wird aber nicht berücksichtigt. • Ein eventuell eingestellter Softstart wird nicht ausgeführt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**laser_on_list**, **para_laser_on_pulses_list**

Kontrollbefehl	laser_signal_off
Funktion	Schaltet die Signale für den “Laser active”-Betrieb ohne Verzug aus.
Aufruf	`laser_signal_off()`
Multi-Board-Befehlsname	**n_laser_signal_off**
Hinweise	• **laser_signal_off** ist für eine direkte Laser-Steuerung in Kombination mit **laser_signal_on** vorgesehen. • **laser_signal_off** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist • **laser_signal_off** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**laser_signal_on**

Normaler Listenbefehl	laser_signal_off_list
Funktion	Wie **laser_signal_off**, aber ein Listenbefehl.
Aufruf	`laser_signal_off_list()`
Hinweise	• –
Multi-Board-Befehlsname	**n_laser_signal_off_list**
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**laser_signal_off**

Kontrollbefehl	laser_signal_on
Funktion	Schaltet die Signale für den "Laser active"-Betrieb ohne Verzug ein.
Aufruf	`laser_signal_on()`
Multi-Board-Befehlsname	**n_laser_signal_on**
Hinweise	• *Vorsicht! Überprüfen Sie den Strahlengang, bevor Sie den Laser einschalten!* • Die Signale für den "Laser active"-Betrieb müssen erst mit **set_laser_mode** ausgewählt, deren Parameter mit weiteren Befehlen angegeben und mit **set_laser_control** oder **enable_laser** freigegeben werden, bevor sie mit **laser_signal_on** eingeschaltet werden können, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192. • **laser_signal_on** ist dafür vorgesehen, den Laser etwa zur Justage direkt einzuschalten. • Die Signale für den "Laser active"-Betrieb müssen mit **laser_signal_off** ausgeschaltet werden. • **laser_signal_on** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **laser_signal_on** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt)
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**laser_signal_off**

Normaler Listenbefehl	laser_signal_on_list
Funktion	Wie **laser_signal_on**, aber ein Listenbefehl und nie ignoriert.
Aufruf	`laser_signal_on_list()`
Multi-Board-Befehlsname	**n_laser_signal_on_list**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**laser_signal_on**

Unverzögerter Kurzer Listenbefehl	**list_call**
Funktion	Bewirkt einen unbedingten Sprung zu einem Unterprogramm, das an der angegebenen absoluten Adresse (an einem beliebigen Ort des RTC6-Listenspeichers) beginnt.
Aufruf	`list_call( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_list_call**
Hinweise	• Der erste Befehl des mit **list_call** aufgerufenen Unterprogramms wird (gegebenenfalls nach einem **list_continue**) sofort und ohne Verzögerung ausgeführt. Es sind auch geschachtelte oder rekursive Unterprogramm-Aufrufe zulässig mit einer Verschachtelungstiefe von bis zu 63 Aufrufen, siehe auch Kapitel 6.5.1 "Unterprogramme", Seite 115. • Jedes Unterprogramm muss mit **list_return** abgeschlossen sein, damit nach Abarbeiten eines Unterprogramms einschließlich des abschließenden **list_return** mit dem der aufrufenden Stelle nachfolgenden Befehl fortgefahren wird. Dieser wird dann auch (gegebenenfalls nach einem **list_continue**) sofort und ohne Verzögerung ausgeführt. Wird statt eines erwarteten **list_return** ein **set_end_of_list** angetroffen, wird die Listenausführung beendet oder – wenn zuvor aktiviert – ein automatischer Listenwechsel ausgeführt (für letzteres ist der momentane Listenstatus entscheidend, s.u.). Es erfolgt dann in keinem Fall mehr ein Rücksprung zur aufrufenden Stelle, auch nicht bei verschachtelten Unterprogramm-Aufrufen. Ein noch unfertiger **mark_text**-Befehl wird nicht mehr zu Ende ausgeführt. Wird das Ende eines Listen-Bereichs ("Liste 1" oder "Liste 2") erreicht, ohne dass zuvor ein **list_return** oder **set_end_of_list** angetroffen wurde, wird mit der Ausführung am Anfang der aktuellen Liste fortgefahren. Im geschützten RTC6-Listenspeicher-Bereich "Liste 3" wird in einem solchen Fall zwangsweise ein **list_return**-Befehl eingefügt und ausgeführt. • **list_call** wird durch **list_nop** ersetzt, wenn `Pos` >(2^{23}–1) oder wenn `Pos` gleich der aktuellen Adresse ist (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Erfolgt in einem aufgerufenen Unterprogramm ein weiterer **list_call**-Befehl auf die Adresse des aufrufenden **list_call** (rekursiver Aufruf), wird die dadurch entstehende Endlos-Schleife beendet, sobald die Obergrenze von 63 Aufrufen erreicht ist. Weitere **list_call**-Befehle werden dann ignoriert und stattdessen der unmittelbar folgende Befehl ausgeführt. • Beginnt das Unterprogramm unmittelbar an der auf **list_call** folgenden Adresse, wird das Unterprogramm nach dem Ausführen von **list_return** erneut ausgeführt (siehe auch Hinweise in der Befehlsbeschreibung von **list_return** zu fehlendem zugehörigem Funktionsaufruf). Es wird in diesem Fall mit dem auf **list_return** folgenden Befehl fortgefahren (siehe auch **list_repeat**…**list_until**), um diese wahrscheinlich nicht gewünschte Listenabarbeitung zu umgehen.

Unverzögerter Kurzer Listenbefehl	**list_call**
Hinweise (Forts.)	• Der mit **read_status** auslesbare BUSY-Listenstatus wird durch **list_call** geändert, wenn sich die aufgerufene Adresse im Listen-Bereich (“Liste 1” oder “Liste 2”) befindet. Liegt diese dagegen im geschützten RTC6-Listenspeicher-Bereich “Liste 3”, wird der Listenstatus der aufrufenden Stelle beibehalten, da der geschützte Bereich keinen eigenen Listenstatus besitzt. Beim Ausführen eines Unterprogramms im geschützten RTC6-Listenspeicher-Bereich “Liste 3” gibt auch **get_status** den Listenausführungsstatus der aufrufenden Stelle und **get_status** und **get_out_pointer** als Position des Ausgabe-Zeigers die Position des aufrufenden **list_call** zurück. • Absolute Vektor-Befehle und Bogen-Befehle werden nach Aufruf mit **list_call** auch absolut ausgeführt. Wenn das Unterprogramm an unterschiedlichen Stellen im Bildfeld ausgeführt werden soll, darf es nur relative [*]mark[*]-Befehle, Bogen-Befehle oder Sprung-Befehle enthalten oder es ist **list_call_abs** zu verwenden. • Steht **list_call** auf dem letztmöglichen Speicherplatz einer Liste (“Liste 1” oder “Liste 2”), so wird – auch bei einem zuvor eingestellten automatischen Listenwechsel – nach Abarbeitung des aufgerufenen Unterprogramms am Anfang derselben Liste mit der Ausführung fortgefahren. • `list_call( Pos )` ist gleichbedeutend mit `list_call_repeat( Pos, 1 )`.
RTC4→RTC6	Weitgehend unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_continue**, **list_repeat**, **list_return**, **list_until**, **list_call_abs**, **list_call_cond**, **list_call_repeat**, **sub_call**

Unverzögerter Kurzer Listenbefehl	**list_call_abs**
Funktion	Bewirkt einen unbedingten Sprung zu einem Unterprogramm, das an der angegebenen absoluten Adresse (an einem beliebigen Ort des RTC6-Listenspeichers) beginnt. Im aufgerufenen Unterprogramm werden absolute Vektor-Befehle und Bogen-Befehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt.
Aufruf	`list_call_abs( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_list_call_abs**
Hinweise	• **list_call_abs** ist weitgehend identisch mit **list_call** (siehe Hinweise dort). **list_call_abs** bewirkt allerdings eine andere Ausführung von absoluten Vektor-Befehlen und Bogen-Befehlen im aufgerufenen Unterprogramm. Bei der Ausführung von **list_call_abs** wird die momentane Position durch Setzen des Offsets an das aufgerufene Unterprogramm übergeben, der dann bei der nachfolgenden Ausführung von absoluten Vektor-Befehlen und Bogen-Befehlen im Unterprogramm verwendet wird. Eine Verschachtelung der Aufrufe wird bei der Ermittlung des Offsets berücksichtigt (bei **list_return** werden die vorherigen Offset-Werte wieder hergestellt). Dadurch können Unterprogramme absolute Vektor-Befehle und Bogen-Befehle enthalten, auch wenn sie an verschiedenen Ortskoordinaten ausgeführt werden sollen. • Enthält ein aufzurufendes Unterprogramm keine absoluten Befehle, besteht zwischen **list_call_abs** und **list_call** kein Unterschied. • `list_call_abs( Pos )` ist gleichbedeutend mit `list_call_abs_repeat( Pos, 1 )`.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_call**, **list_call_abs_cond**, **list_call_abs_repeat**

Unverzögerter Kurzer Listenbefehl	**list_call_abs_cond**
Funktion	*Bedingter Aufruf (AbsCall) eines Unterprogramms*: **list_call_abs_cond** führt den Befehl **list_call_abs**(`Pos`) aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_call_abs_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_list_call_abs_cond**
Hinweise	• Siehe **list_call_abs**. • Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_call_abs**

Unverzögerter Kurzer Listenbefehl	**list_call_abs_repeat**
Funktion	Bewirkt einen unbedingten Sprung zu einem Unterprogramm, das an der angegebenen absoluten Adresse (an einem beliebigen Ort des RTC6-Listenspeichers) beginnt und führt dessen Rumpf mehrmals aus.
Aufruf	`list_call_abs_repeat( Pos, Number )`
Parameter	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Wie bei **list_call_abs**: Als 32-Bit-Wert ohne Vorzeichen. `Number` Anzahl der Wiederholungen. Als 32-Bit-Wert ohne Vorzeichen. `Number` = 0 wird wie `Number` = 1 behandelt.
Multi-Board-Befehlsname	**n_list_call_abs_repeat**
Hinweise	• `list_call_abs( Pos )` ist gleichbedeutend mit `list_call_abs_repeat( Pos, 1 )`. • **list_call_abs_repeat** vermeidet bei der Wiederholung einen Leertakt, der sonst bei Konstruktionen wie `list_call_abs`…`list_call_abs` oder `list_repeat`…`list_call_abs`…`list_until` unvermeidlich ist. • Siehe **list_call_repeat** und **list_call_abs**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_call**, **list_call_abs**, **list_call_abs_cond**, **list_call_repeat**, **list_repeat**, **list_until**

Unverzögerter Kurzer Listenbefehl	**list_call_cond**
Funktion	*Bedingter Aufruf eines Unterprogramms*: **list_call_abs_cond** führt **list_call**(`Pos`) aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_call_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_list_call_cond**
Hinweise	• Siehe **list_call**. • Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Weitgehend unveränderte Funktionalität (siehe **list_call**).
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_call**

Unverzögerter Kurzer Listenbefehl	**list_call_repeat**
Funktion	Bewirkt einen unbedingten Sprung zu einem Unterprogramm, das an der angegebenen absoluten Adresse (an einem beliebigen Ort des RTC6-Listenspeichers) beginnt und führt dessen Rumpf mehrmals aus.
Aufruf	`list_call_repeat( Pos, Number )`
Parameter	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Wie bei **list_call**: Als 32-Bit-Wert ohne Vorzeichen. `Number` Anzahl der Wiederholungen. Als 32-Bit-Wert ohne Vorzeichen. `Number` = 0 wird wie `Number` = 1 behandelt.
Multi-Board-Befehlsname	**n_list_call_repeat**
Hinweise	• `list_call( Pos )` ist gleichbedeutend mit `list_call_repeat( Pos, 1 )`. • **list_call_repeat** vermeidet bei der Wiederholung einen Leertakt, der sonst bei Konstruktionen wie `list_call`…`list_call` oder `list_repeat`…`list_call`…`list_until` unvermeidlich ist. • Mit **list_call_repeat** können z.B. Trajektorien für Anlauf- und Auslaufkurven aus micro_vector[*]-Befehlen mit Figuren aus Unterprogrammen nahtlos aneinander gefügt werden. • Wenn bei **list_call_repeat** unmittelbar ein weiterer Unterprogramm-Aufruf folgt (verschachtelte Aufrufe), muss allerdings ein Leertakt eingefügt werden. Das kann vermieden werden, wenn zwischen zwei Unterprogramm-Aufrufen z.B. mindestens ein micro_vector[*]-Befehl steht. • Der Beginn des Unterprogramms `Pos` kann an einer beliebigen Stelle im RTC6-Listenspeicher-Bereich liegen. Das gilt insbesondere auch unmittelbar nach **list_call_repeat**. Dadurch kann der gesamte RTC6-Listenspeicher kontinuierlich für Listenbefehle verwendet werden, ohne einen speziellen Bereich für geschützte Unterprogramme reservieren zu müssen. • Nach einem **list_return** wird mit dem **list_call_repeat** unmittelbar nachfolgenden Befehl fortgefahren. Wenn jedoch der Beginn des Unterprogramms unmittelbar auf **list_call_repeat** folgt, wird mit dem Befehl unmittelbar nach **list_return** fortgefahren.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_call**, **list_call_abs**, **list_call_abs_repeat**, **list_repeat**, **list_return**, **list_until**, **micro_vector_abs**, **micro_vector_abs_3d**, **micro_vector_rel**, **micro_vector_rel_3d**

Normaler Listenbefehl	**list_continue**
Funktion	Fügt eine Null-Operation (no operation) in den RTC6-Listenspeicher ein.
Aufruf	`list_continue()`
Multi-Board-Befehlsname	**n_list_continue**
Hinweise	• **list_continue** benötigt wie **list_nop** 10 μs zur Ausführung. • Folgt **list_continue** unmittelbar auf einen kurzen Listenbefehl, so sorgt er wie **list_nop** dafür, dass der nachfolgende Listenbefehl garantiert erst im nächsten 10 μs-Takt ausgeführt wird (die "effektive" Ausführungsdauer des kurzen Listenbefehls beträgt dann 10 μs). Im Gegensatz zu **list_nop** lässt **list_continue** aber die Lasersteuersignale unangetastet und wartet kein Scanner-Delay ab. Im Gegensatz zu **list_nop** erlaubt also **list_continue** das Verschieben von kurzen Listenbefehlen auf den nachfolgenden 10 μs-Takt, ohne dass ein eventueller Polygonzug durch Ausschalten des Lasers unterbrochen wird. • Bei einer Überschreitung der maximal zulässigen Anzahl von unmittelbar aufeinander folgenden kurzen Listenbefehlen wird automatisch ein Leertakt eingefügt, der genau **list_continue** entspricht. • **list_continue** kann auch dazu verwendet werden, mehrfache Ausgaben auf ein und denselben Ausgabeport voneinander zu trennen. Ohne **list_continue** würden beispielsweise mehrere unmittelbar aufeinanderfolgende **write_da_x_list**-Aufrufe (kurze Listenbefehle) den analogen Ausgabeport in einem 10 μs-Takt mehrmals beschreiben, wodurch einige Werte nie zur Ausgabe gelangen könnten. • Die RTC6 setzt **list_continue** nie als Platzhalter für andere (zurückgewiesene) Listenbefehle ein, sondern immer nur **list_nop**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_nop**

Unverzögerter Kurzer Listenbefehl	**list_jump_cond**
Funktion	*Bedingter absoluter Listensprung*: **list_jump_cond** führt **list_jump_pos**(`Pos`) aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_jump_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_list_jump_cond**
Hinweise	• **list_jump_cond** ist gleichbedeutend mit **list_jump_pos_cond** (siehe Hinweise dort).
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: • Sprünge in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” und Sprünge aus dem geschützten RTC6-Listenspeicher-Bereich “Liste 3” heraus sind verboten (fehlerhafte Sprung-Befehle werden beim Abarbeiten ignoriert, siehe auch **list_jump_pos**).
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_jump_pos**, **list_jump_pos_cond**

Unverzögerter Kurzer Listenbefehl	**list_jump_pos**
Funktion	Bewirkt bei der Ausführung einen unbedingten Sprung auf die angegebene Adresse des RTC6-Listenspeichers. Der dortige nächste Befehl wird sofort und ohne Verzögerung ausgeführt.
Aufruf	`list_jump_pos( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_list_jump_pos**
Hinweise	• Für `Pos` kann eine absolute Speicheradresse innerhalb des konfigurierten Listen-Bereichs (“Liste 1” und “Liste 2”) angegeben werden, nicht jedoch innerhalb des geschützten Bereichs “Liste 3” oder ganz außerhalb des RTC6-Listenspeicher-Bereichs. • Verboten sind neben Sprüngen in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” auch solche aus dem geschützten RTC6-Listenspeicher-Bereich “Liste 3” heraus. • Fehlerhafte Sprung-Befehle werden zwar unverändert zur RTC6 übertragen, beim Abarbeiten jedoch ignoriert. Stattdessen wird der unmittelbar nachfolgende Befehl ausgeführt. Das Programm funktioniert vermutlich nicht mehr wie gewünscht. Bei der Programmierung mit Sprung-Befehlen muss daher mit einer entsprechenden Sorgfalt vorgegangen werden (siehe auch Kapitel 6.5.3 “Sprünge”, Seite 123). • Sprung-Befehle, die einen Sprung auf sich selbst veranlassen (`Pos` = Listenposition des Sprung-Befehls) werden zur Laufzeit ebenso ignoriert, sonst würden sie bei der Ausführung eine Endlosschleife ohne weitere Aktivitäten ergeben, die sich nur mit **stop_execution** oder mit externem Stopp beenden ließe. • Entscheidend sind die Bedingungen zur Laufzeit. Beim Umkonfigurieren des RTC6-Listenspeichers oder bei der Konvertierung eines Unterprogramms kann es passieren, dass eine ursprünglich gültige Sprungadresse aufgrund neuer Listengrenzen oder einer veränderten Speicherposition des Unterprogramms zu einer ungültigen Sprungadresse wird. • Der BUSY-Listenstatus der Listen wird durch **list_jump_pos** wechselnd gesetzt, der USED-Listenstatus der Listen bleibt unverändert (siehe **read_status**). • **list_jump_pos** ist gleichbedeutend mit **set_list_jump**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_list_jump**, **list_jump_rel**, **list_jump_pos_cond**

Unverzögerter Kurzer Listenbefehl	**list_jump_pos_cond**
Funktion	*Bedingter absoluter Listensprung*: **list_jump_pos_cond** führt **list_jump_pos**(`Pos`) aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_jump_pos_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Mask0` Siehe `Mask1`. `Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_list_jump_pos_cond**
Hinweise	• Siehe **list_jump_pos**. • Entgegen den Regeln zur Verhinderung von Endlosschleifen (siehe **list_jump_pos**) sind bei **list_jump_pos_cond** auch Sprünge auf die eigene Adresse (`Pos` = Listenposition des Sprung-Befehls) erlaubt, z.B. zum Warten auf die Bestätigung eines Signals. • **list_jump_pos_cond** ist gleichbedeutend mit **list_jump_cond**. • Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.

Beispiele (Pascal)	• Warte bis Bit #3 des Digital-Eingangs auf HIGH wechselt (= durchlaufe die Schleife, solange das Bit auf *LOW* steht):
	`list_jump_pos_cond(0, $0008, get_input_pointer);`
	• überspringe die zwei nächsten Listenbefehle, falls sich der Digital-Eingang im folgenden Zustand befindet: `xxxx xxxx xxxx 0110`:
	`list_jump_pos_cond(6, 9, get_input_pointer + 3);`
	• Siehe auch Abschnitt "Beispiel-Code (Pascal)", Seite 321.

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_jump_pos**

Unverzögerter Kurzer Listenbefehl	**list_jump_rel**
Funktion	Bewirkt bei der Ausführung einen unbedingten Sprung um die angegebenen Stellen innerhalb der aktuellen Liste. Der dortige nächste Befehl wird sofort und ohne Verzögerung ausgeführt.
Aufruf	`list_jump_rel( Pos )`
Parameter	`Pos` Sprungweite [$(-2^{23}+1)$…$(2^{23}-1)$]. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_list_jump_rel**
Hinweise	• Mit **list_jump_rel** können z.B. “if-then-else“-Verzweigungen unabhängig von der Listenposition des Befehls, insbesondere auch unabhängig von der Listennummer geschrieben werden, da relative Listenadressen angegeben werden. • Mit **get_list_pointer** kann der aktuelle Input-ListPointer abgefragt werden. • **list_jump_rel** kann in allen RTC6-Listenspeicher-Bereichen, also auch im geschützten RTC6-Listenspeicher-Bereich “Liste 3”, verwendet werden. • Im Bereich der “Liste 1” oder “Liste 2” oder bei nicht indizierten Unterprogrammen im Bereich der “Liste 3” muss bei der Angabe der Sprungweite darauf geachtet werden, dass der Sprung nicht über die Grenzen des jeweiligen RTC6-Listenspeicher-Bereichs hinaus erfolgt. • Falls der Befehl in einem indizierten Unterprogramm oder Zeichensatz eingesetzt wird, muss zusätzlich darauf geachtet werden, dass der Sprung nicht über die Grenzen des Unterprogramms oder Zeichensatzes hinaus geht. • Fehlerhafte Sprung-Befehle werden zwar unverändert zur RTC6 übertragen, beim Abarbeiten jedoch ignoriert. Stattdessen wird der unmittelbar nachfolgende Befehl ausgeführt. Das Programm funktioniert vermutlich nicht mehr wie gewünscht. Bei der Programmierung mit Sprung-Befehlen muss daher mit einer entsprechenden Sorgfalt vorgegangen werden (siehe auch Kapitel 6.5.3 ”Sprünge”, Seite 123). • Sprung-Befehle, die einen Sprung auf sich selbst veranlassen (`Pos` = 0) werden zur Laufzeit ebenso ignoriert, sonst würden sie bei der Ausführung eine Endlosschleife ohne weitere Aktivitäten ergeben, die sich nur mit **stop_execution** oder mit externem Stopp beenden ließe. • Entscheidend sind die Bedingungen zur Laufzeit. Beim Umkonfigurieren des RTC6-Listenspeichers oder bei der Konvertierung eines Unterprogramms kann es passieren, dass eine ursprünglich gültige Sprungadresse aufgrund neuer Listengrenzen oder einer veränderten Speicherposition des Unterprogramms zu einer ungültigen Sprungadresse wird.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_jump_pos**, **list_jump_rel_cond**

Unverzögerter Kurzer Listenbefehl	**list_jump_rel_cond**
Funktion	*Bedingter relativer Listensprung*: **list_jump_rel_cond** führt **list_jump_rel**(`Pos`) aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_jump_rel_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Mask0` Siehe `Mask1`. `Pos` Sprungweite [$(-2^{23}+1)$…$(2^{23}-1)$]. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_list_jump_rel_cond**
Hinweise	• Siehe **list_jump_rel**. • Entgegen den Regeln zur Verhinderung von Endlosschleifen (siehe **list_jump_rel**) sind bei **list_jump_rel_cond** auch Sprünge auf die eigene Adresse (**Pos** = 0) erlaubt, z.B. zum Warten auf die Bestätigung eines Signals. • Siehe auch Kapitel 6.5 "Strukturierte Programmierung", Seite 115.

Beispiele (Pascal)	• Warte bis Bit #3 des Digital-Eingangs auf HIGH wechselt (= durchlaufe die Schleife, solange das Bit auf *LOW* steht):
	`list_jump_rel_cond(0, $0008, 0);`
	• Überspringe die zwei nächsten Listenbefehle, falls sich der Digital-Eingang im folgenden Zustand befindet: `xxxx xxxx xxxx 0110`:
	`list_jump_rel_cond(6, 9, 3);`
	• Siehe auch Abschnitt "Beispiel-Code (Pascal)", Seite 321.

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_jump_rel**

Unverzögerter Kurzer Listenbefehl	**list_next**
Funktion	Führt den nächsten Listenbefehl aus.
Aufruf	`list_next()`
Multi-Board-Befehlsname	**n_list_next**
Hinweise	• **list_next** liest den nächsten Listenbefehl und führt ihn sofort aus, wenn nicht die maximale Anzahl von zulässigen Kurzbefehlen überschritten ist. Anderenfalls wird er erst im nächsten 10 μs-Takt ausgeführt. • **list_next** eignet sich daher als Platzhalter für einen anderen Befehl in einer Liste, weil er einen zusätzlichen 10 μs-Takt vermeidet, der sonst bei Platzhaltern wie **list_nop** oder **list_continue** unvermeidlich ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_nop**, **list_continue**

Normaler Listenbefehl	**list_nop**
Funktion	Fügt eine Null-Operation (no operation) in den RTC6-Listenspeicher ein.
Aufruf	`list_nop()`
Multi-Board-Befehlsname	**n_list_nop**
Hinweise	• **list_nop** wirkt wie **long_delay**(1), schaltet also die Signale für den "Laser active"-Betrieb nach einem LaserOff-Delay aus und wartet ein etwaiges Scanner-Delay ab. Auch wenn keine Delays abzuwarten sind, benötigt **list_nop** dennoch 10 μs zur Ausführung. • **list_nop** dient als Platzhalter für zurückgewiesene Listenbefehle (**get_last_error**-Returncode `RTC6_IGNORED`). • Folgt **list_nop** unmittelbar auf einen kurzen Listenbefehl, so sorgt er dafür, dass der nachfolgende Listenbefehl garantiert erst im nächsten 10 μs-Takt ausgeführt wird (die "effektive" Ausführungsdauer des kurzen Listenbefehls beträgt dann 10 μs).
RTC4→RTC6	Der RTC4-Befehl ist ein reiner Platzhalter und entspricht eher **list_continue**. Der RTC6-Befehl schaltet die Signale für den "Laser active"-Betrieb aus und wartet ein Scanner-Delay ab.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**long_delay**, **list_continue**

Unverzögerter Kurzer Listenbefehl	**list_repeat**
Funktion	Leitet eine Befehlswiederholung für eine Gruppe von Listenbefehlen ein.
Aufruf	`list_repeat()`
Multi-Board-Befehlsname	**n_list_repeat**
Hinweise	• Siehe Kapitel 6.5.5 "Schleifen", Seite 125. • Ein noch anliegender Verzögerter Kurzer Listenbefehl wird noch zuvor ausgeführt. • Alle Listenbefehle zwischen **list_repeat** und dem nächsten **list_until**-Aufruf werden gegebenenfalls mehrfach wiederholt. • Jede ausgeführte **list_repeat**…**list_until**-Schleife benötigt einen vollen 10 μs-Takt. Entsprechend wird der erste Listenbefehl nach einer Wiederholung mit einer Verzögerung von 10 μs ausgeführt. • Es können bis zu 8 verschachtelte **list_repeat**…**list_until**-Schleifen programmiert werden. Jeder weitere **list_repeat**-Aufruf darüber hinaus wird ignoriert, solange kein **list_until**-Aufruf eine Schleife abgeschlossen hat. • Wenn eine Liste beendet wird (durch **set_end_of_list** oder **stop_execution** oder /STOP), werden alle noch nicht fertig abgearbeiteten **list_repeat**…**list_until**-Schleifen gelöscht. Das erfolgt jedoch nicht, wenn ein automatischer Listenwechsel (durch **auto_change**, **auto_change_pos** oder **start_loop**) aktiv ist. • Vollständige **list_repeat**…**list_until**-Schleifen können sich innerhalb einer Liste oder innerhalb eines Unterprogramms befinden. Ein Wechsel zwischen Listen und Unterprogrammen oder zwischen verschiedenen Unterprogrammen ist nicht zulässig. Ein Wechsel zwischen Listen ist zulässig, solange der Listenwechsel ohne explizites Beenden einer Liste stattfindet (entweder durch einen automatischen Listenwechsel oder einen expliziten Listensprung mit **list_jump_pos** in die andere Liste). • Listensprünge in den Rumpf einer **list_repeat**…**list_until**-Schleife oder aus ihm heraus sollten vermieden werden. Bei unsachgemäßem Gebrauch kann die Integrität der Schleifenverwaltung so gestört werden, dass die Schleifen u. U. nicht mehr wie geplant ablaufen. Unterprogramm-Aufrufe aus einer Schleife heraus sind jedoch problemlos möglich.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_until**

Unverzögerter Kurzer Listenbefehl	**list_return**
Funktion	Beendet ein zuvor aufgerufenes Unterprogramm, springt zu der aufrufenden Stelle und führt den unmittelbar darauf folgenden Listenbefehl (gegebenenfalls nach einem **list_continue**) sofort und ohne Verzögerung aus.
Aufruf	`list_return()`
Multi-Board-Befehlsname	**n_list_return**
Hinweise	• Beim Laden eines indizierten Unterprogramms, eines indizierten Zeichens oder einer indizierten Zeichenfolge in den geschützten Bereich bewirkt der Befehl **list_return** außerdem, dass u.a. die Startadresse in der entsprechenden internen Verwaltungstabelle eingetragen wird. • Ist **list_return** dagegen der unmittelbar auf **load_sub**, **load_char** oder **load_text_table** folgende Befehl, wird das angesprochene Unterprogramm, das Zeichen oder die Zeichenfolge in der entsprechenden internen Verwaltungstabelle gelöscht. • Werden indizierte Unterprogramme, Zeichen oder Zeichenfolgen beim Laden in den geschützten Bereich *nicht* mit **list_return** abgeschlossen, bevor der Eingabe-Zeiger explizit auf eine andere Position gesetzt wird, so werden sie nicht gespeichert und sind im Folgenden nicht verfügbar. • **list_return** löst beim Laden einen Flush der gepufferten Listeneingabe aus, siehe Kapitel 6.4.1 "Listen laden", Seite 108. • Folgt **list_return** unmittelbar nach **load_sub**, **load_char** oder **load_text_table**, dann ist der Eingabe-Zeiger nach **list_return** ungültig. D.h. es können keine weiteren Befehle geladen werden, ohne zuvor eine neue Positionierung angefordert zu haben. • Wird beim Abarbeiten ein **list_return** angetroffen, ohne dass zuvor ein Unterprogramm, ein Zeichen oder eine Zeichenfolge explizit aufgerufen wurde, wird mit der absoluten Adresse 0 (Startadresse von "Liste 1") fortgefahren. Bei verschachtelten Aufrufen wird die Integrität der Unterprogramm-Struktur zerstört.
RTC4→RTC6	In der alten Funktionalität (Beenden eines Unterprogramms) unverändert. Neu: Bedeutung beim Laden in den geschützten RTC6-Listenspeicher-Bereich "Liste 3".
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_call**, **sub_call**, **load_char**, **load_text_table**, **load_sub**, **list_continue**

Unverzögerter Kurzer Listenbefehl	**list_until**
Funktion	Beendet gegebenenfalls eine Befehlswiederholung für eine Gruppe von Listenbefehlen.
Aufruf	`list_until( Number )`
Parameter	`Number` Anzahl von Wiederholungen. Als 32-Bit-Wert ohne Vorzeichen. `Number` = 0 wird wie `Number` = 1 behandelt.
Multi-Board-Befehlsname	**n_list_until**
Hinweise	• Siehe Kapitel 6.5.5 "Schleifen", Seite 125. • Ein noch anliegender Verzögerter Kurzer Listenbefehl wird noch zuvor ausgeführt. • Alle Listenbefehle zwischen **list_until** und dem zuletzt vorausgegangenen **list_repeat**-Aufruf werden `Number` mal wiederholt. • Es können bis zu 8 verschachtelte **list_repeat**…**list_until**-Schleifen programmiert werden. Jeder weitere **list_until**-Aufruf, für den kein **list_repeat**-Aufruf vorausgeht, wird ignoriert. • Eine leere Schleife, bestehend aus einem **list_repeat**-Aufruf, unmittelbar gefolgt von einem **list_until**-Aufruf, wird sofort beendet und nicht wiederholt. • Wenn eine Liste beendet wird (durch **set_end_of_list** oder **stop_execution** oder /STOP), werden alle noch nicht fertig abgearbeiteten **list_repeat**…**list_until**-Schleifen gelöscht. Das erfolgt jedoch nicht, wenn ein automatischer Listenwechsel (durch **auto_change**, **auto_change_pos** oder **start_loop**) aktiv ist. • Vollständige **list_repeat**…**list_until**-Schleifen können sich innerhalb einer Liste oder innerhalb eines Unterprogramms befinden. Ein Wechsel zwischen Listen und Unterprogrammen oder zwischen verschiedenen Unterprogrammen ist nicht zulässig. Ein Wechsel zwischen Listen ist zulässig, solange der Listenwechsel ohne explizites Beenden einer Liste stattfindet (entweder durch einen automatischen Listenwechsel oder einen expliziten Listensprung mit **list_jump_pos** in die andere Liste). • Listensprünge in den Rumpf einer **list_repeat**…**list_until**-Schleife oder aus ihm heraus sollten vermieden werden. Bei unsachgemäßem Gebrauch kann die Integrität der Schleifenverwaltung so gestört werden, dass die Schleifen u. U. nicht mehr wie geplant ablaufen. Unterprogramm-Aufrufe aus einer Schleife heraus sind jedoch problemlos möglich.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_repeat**

Kontrollbefehl	load_auto_laser_control	
Funktion	Lädt eine Tabelle mit Datenpunkten aus einer ASCII-Textdatei. Bestimmt durch lineare Interpolation die Nichtlinearitätskurve (siehe Abschnitt "Nichtlinearitätskurve laden und ermitteln", Seite 215) für die "Automatische Laser-Steuerung" (außer der Vektorkontrollierte Laser-Steuerung), siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208.	
Aufruf	`NoOfDataPoints = load_auto_laser_control( Name, No )`	
Parameter	`Name`	Name der Textdatei oder NULL. Die Textdatei kann eine oder mehrere Tabellen enthalten. Als Zeiger auf einen nullterminierten ANSI-String.
	`No`	Legt fest, welche Tabelle aus der Textdatei geladen werden soll. Der Parameter entspricht der Erweiterung `<No>` von `[AutoLaserCtrlTable<No>]` am Anfang der gewünschten Tabelle. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Im Fehlerfall ein positiver Fehlercode. Im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte. Als 32-Bit-Wert mit Vorzeichen.	
	Wert	Beschreibung
	–1…– 50	Erfolg. Der Absolutwert des Rückgabewerts ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert, siehe auch Abschnitt "Nichtlinearitätskurve laden und ermitteln", Seite 215.
	–1024	Für `Name` = NULL (siehe auch Hinweise).
	1	Keine gültigen Datenpunkte gefunden (wohl aber Tabelle `No`).
	3	Datei nicht gefunden.
	4	DSP-Speicher-Fehler.
	5	Busy-Fehler, Karte ist BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus, kein Download (**get_last_error**-Returncode `RTC6_BUSY`).
	8	Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC6_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC6_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer konnte in der Datei nicht gefunden werden.
Multi-Board-Befehlsname	**n_load_auto_laser_control**	
Hinweise	• Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für die Nichtlinearitätskurve in die Text-Datei sind im Abschnitt "Nichtlinearitätskurve laden und ermitteln", Seite 215 beschrieben. Beim Laden der Tabelle bestimmt die RTC6 durch lineare Interpolation geeignete Werte für den ganzen Prozentbereich. • **load_auto_laser_control** überschreibt jede zuvor geladene Nichtlinearitätskurve. • Für `Name` = NULL wird (wie bei Initialisierung mit **load_program_file**) für den vollen Prozentbereich die Funktion *Scale*(*Percent*)`=1.0` geladen (keine Nichtlinearität).	

Kontrollbefehl	load_auto_laser_control
Hinweise (Forts.)	• **load_auto_laser_control** wird nicht ausgeführt (**get_last_error**-Returncode RTC6_BUSY), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_auto_laser_control** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Laufzeit von **load_auto_laser_control** werden externe Starts unterdrückt. • Vor dem Laden einer Tabelle führt **load_auto_laser_control** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt. • Die Tabelle kann mit **create_dat_file** gespeichert werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_auto_laser_control**, **create_dat_file**

Kontrollbefehl	load_char
Funktion	Versieht ein durch nachfolgende Listenbefehle definiertes Zeichen mit dem gewünschten Index und lädt es in den geschützten RTC6-Listenspeicher-Bereich “Liste 3”.
Aufruf	`load_char( Char )`
Parameter	`Char` Index des indizierten Zeichens. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Multi-Board-Befehlsname	**n_load_char**
Hinweise	• Es können 1024 indizierte Zeichen d. h. 4 Zeichensätze zu jeweils 256 indizierten Zeichen gespeichert werden. Für die Definition von Zeichensätzen gilt: `Char` = Zeichensatznummer × 256 + ASCII-Nummer des Zeichens (die Zeichensatznummer zählt dabei von 0 bis 3). Für `Char` > 1023 wird **load_char** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Die Adressen im geschützten RTC6-Listenspeicher-Bereich “Liste 3”, wo die jeweiligen Zeichendefinitionen gespeichert werden, werden automatisch ermittelt und intern verwaltet. Die Verwaltung erfolgt separat von indizierten Unterprogrammen (siehe **load_sub**) und Zeichenfolgendefinitionen (siehe **load_text_table**). • Indizierte Zeichendefinitionen müssen jeweils mit einem **list_return**-Aufruf abgeschlossen werden. Erst damit werden die Befehle endgültig gespeichert, die Startadresse in die interne Verwaltungstabelle eingetragen und auch ein Flush der gepufferten Listeneingabe ausgelöst, siehe Kapitel 6.4.1 “Listen laden”, Seite 108. Anderenfalls (der Eingabe-Zeiger wird ohne vorhergehendes **list_return** verändert) ist die Zeichendefinition mit diesem Index nicht verfügbar. • Eine indizierte Zeichendefinition wird nicht gespeichert, falls nicht zuvor der geschützte RTC6-Listenspeicher-Bereich “Liste 3” in ausreichender Größe jenseits von “Liste 1” und “Liste 2” konfiguriert wurde. • Ist **list_return** der unmittelbar auf **load_char** folgende Befehl, wird die entsprechende Zeichendefinition aus der internen Verwaltungstabelle gelöscht. • Beachten Sie auch alle Hinweise in Kapitel 6.5.2 “Zeichensätze und Zeichenfolgen”, Seite 121.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_return**, **load_sub**, **load_text_table**

Kontrollbefehl	load_correction_file
Funktion	Überträgt die angegebene Korrekturdatei in den Speicher der RTC6. Ruft dann automatisch **select_cor_table** mit den zuletzt verwendeten Parameterwerten (oder den Default-Parameterwerten) auf.
Aufruf	`ErrorNo = load_correction_file( Name, No, Dim )`
Parameter	`Name` Name der Korrekturdatei. Als Zeiger auf einen nullterminierten ANSI-String.
	`No` Nummer der Korrekturtabelle. Zulässige Werte: [1…8]. Als 32-Bit-Wert ohne Vorzeichen. Siehe auch **number_of_correction_tables**.
	`Dim` Bestimmt, ob der Korrekturdatei-Inhalt als 2D-Korrekturtabelle oder (wenn möglich) als 3D-Korrekturtabelle gespeichert wird (siehe auch Hinweise). Als 32-Bit-Wert ohne Vorzeichen. = 2: Sowohl 2D- als auch 3D-Korrekturdateien werden als 2D-Korrekturtabelle gespeichert (gegebenenfalls Downgrade). = 3: Bei freigeschalteter Option "3D" werden sowohl 2D-Korrekturdateien als auch 3D-Korrekturdateien als 3D-Korrekturtabelle gespeichert (gegebenenfalls Upgrade). Bei nicht freigeschalteter Option "3D" wie `Dim` = 2. Sonst: `Dim` hat keine Auswirkung. Eine 2D-Korrekturdatei wird als 2D-Korrekturtabelle gespeichert. Eine 3D-Korrekturdatei wird bei freigeschalteter Option "3D" als 3D-Korrekturtabelle gespeichert, anderenfalls als 2D-Korrekturtabelle (Downgrade).
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Erfolg. 1 Datei-Fehler (Datei korrupt oder unvollständig). 2 Memory Fehler (RTC6-DLL-intern, Windows-Arbeitsspeicher). 3 Datei-Open-Fehler (leerer String für Parameter `Name` angegeben, Datei nicht gefunden etc.). 4 DSP-Speicher-Fehler. 5 PCI-Download-Fehler (RTC6-Kartentreiber-Fehler), Ethernet-Download-Fehler. 8 RTC6-Kartentreiber nicht gefunden (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`). 10 Parameter-Fehler (falsche `No`).

Kontrollbefehl	load_correction_file	
Rückgabe (Forts.)	11	Access-Fehler: Karte für anderes Anwenderprogramm reserviert (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`) oder die Versionsprüfung hat einen Fehler festgestellt (OUT- oder RTC-Version nicht kompatibel mit der aktuellen RTC6-DLL-Version, **get_last_error**-Returncode `RTC6_ACCESS_DENIED` | `RTC6_VERSION_MISMATCH`).
	12	Warnung: 3D-Korrekturtabelle oder Dim = 3 ausgewählt, aber Option "3D" nicht freigeschaltet. Das System funktioniert aber weiterhin als völlig normales 2D-System (die Warnung wird nur ausgegeben, wenn kein anderer Fehler vorliegt).
	13	Busy-Fehler: kein Download, Karte BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus (**get_last_error**-Returncode `RTC6_BUSY`).
	14	PCI-Upload-Fehler (RTC6-Kartentreiber-Fehler, nur bei Download-Überprüfung).
	15	Verify-Fehler (nur bei Download-Überprüfung).
Multi-Board-Befehlsname	**n_load_correction_file**	
Hinweise	Zum Laden von Korrekturtabellen: • Die RTC6 kann 8 verschiedene Korrekturtabellen zur selben Zeit speichern, z.B. zur Verwendung in einer Konfiguration mit mehreren Scan-Köpfen. • Vor dem Speichern einer Korrekturtabelle führt **load_correction_file** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt. • Liegt der Parameter `No` im unzulässigen Bereich, wird keine Korrekturtabelle geladen (Rückgabewert 10). Den zulässigen Bereich kann der Benutzer mit **number_of_correction_tables** weiter einschränken. • Der Name der zu ladenden Korrekturdatei muss bei **load_correction_file** als Zeiger auf einen nullterminierten ANSI-String übergeben werden. Wird für `Name` ein NULL-Zeiger angegeben, wird die entsprechende Tabelle durch eine 1to1-Tabelle ersetzt (für `Dim` = 3 bei freigeschalteter Option "3D" durch eine 3D-1to1-Tabelle, sonst durch eine 2D-1to1-Tabelle). Ein leerer String ("") für `Name` erzeugt einen Fehlercode 3. • Ist die Option "3D" *nicht* freigeschaltet (Standard), werden – unabhängig vom angegebenen Wert für `Dim` – sowohl 2D- als auch 3D-Korrekturdateien als 2D-Korrekturtabellen gespeichert. Der 3D-Anteil der 3D-Korrekturdateien wird hier ignoriert. • Ist die Option "3D" dagegen *freigeschaltet*, können sowohl 2D- als auch 3D-Korrekturdateien geladen werden. – Für `Dim` = 2 wird in jedem Fall eine 2D-Tabelle gespeichert (bei 3D-Korrekturdateien wird der 3D-Anteil der Daten ignoriert) und in der Folge auch immer nur eine 2D-Korrektur berechnet, die z-Achse bleibt dann unverändert, so als ob keine Option "3D" freigeschaltet wäre. – Für `Dim` = 3 werden bei freigeschalteter Option "3D" sowohl 2D- als auch 3D-Korrekturdateien als 3D-Korrekturtabellen gespeichert. 2D-Korrekturtabellen werden dabei automatisch um eine lineare z-Korrektur ergänzt. Die tatsächlich passende z-Korrektur kann anschließend mit **load_z_table** oder **load_z_table_no** geladen werden, siehe Kapitel 7.3.4 "3D-Bildfeld", Seite 178. – Alle anderen Werte von `Dim` ändern am Korrekturdatei-Typ nichts.	

Kontrollbefehl	load_correction_file
Hinweise (Forts.)	Zum Zuweisen von Korrekturtabellen mit **select_cor_table**: • Eine Zuweisung einer (oder zweier) auf der RTC6 gespeicherten Korrekturtabelle(n) an den Scan-Kopf (oder an die beiden Scan-Kopf-Anschlüsse) erfolgt mit **select_cor_table** oder **select_cor_table_list**. • **load_correction_file** ruft nach dem Speichern einer Korrekturtabelle automatisch **select_cor_table** auf. Wird **load_correction_file** *vor* dem Laden der Programmdatei (mit **load_program_file**) aufgerufen, dann hat der automatische Aufruf von **select_cor_table** allerdings keine Wirkung. Wird **load_correction_file** *nach* **load_program_file** aufgerufen, dann erfolgt der **select_cor_table**-Aufruf mit den Parameterwerten (`HeadA` = 1, `HeadB` = 0) oder mit den Parametern des zuletzt (nach **load_program_file**) erfolgten **select_cor_table**-Aufrufs. **load_correction_file** kehrt erst zum Anwenderprogramm zurück, wenn der durch **select_cor_table** ausgelöste Sprung auf die korrigierte Galvanometerscanner-Position abgeschlossen ist. Siehe auch Hinweise, Seite 184. Sonstige: • In RTC5/RTC6-Korrekturtabellen sind Parameter eingetragen, die bei der RTC4 noch der mitgelieferten `*_ReadMe.txt`-Datei zu entnehmen sind und händisch in das Anwenderprogramm übertragen werden müssen. Diese Parameter können bei der RTC6 mit **get_table_para** aus den aktuell geladenen oder mit **get_head_para** aus den zugewiesenen Korrekturtabellen ausgelesen und auf diese Weise direkt in ein Anwenderprogramm eingebunden werden. • Während der Laufzeit von **load_correction_file** werden: – Keine weiteren Kontrollbefehle ausgeführt – Externe Starts unterdrückt • **load_correction_file** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_correction_file** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Bei einem `RTC6_VERSION_MISMATCH`-Fehler (Rückgabewert 11) muss die RTC6-DLL-Version passend zu den Programmdateien gewählt und die Karte einmal stromlos gemacht werden (um die Programmsoftware zu entladen) oder die zur RTC6-DLL-Version passenden Programmdateien mit **load_program_file** neu geladen werden. Anschließend (und nachdem die Karte wieder durch **acquire_rtc** oder **select_rtc** in Besitz genommen) kann **load_correction_file** wieder normal ausgeführt werden.

Kontrollbefehl	load_correction_file
RTC4→RTC6	Bis auf die Grundfunktion des Befehls und seiner Parameter `Name` und `No` hat sich die Funktionalität des Befehls stark geändert: • **load_correction_file** enthält jetzt auch den neuen Parameter `Dim`. Unter anderem können damit 3D-Korrekturdateien auch als 2D-Korrekturtabellen und 2D-Korrekturdateien auch als 3D-Korrekturtabellen gespeichert werden. • Im Speicher der RTC6 (mit freigeschalteter Option "3D") können jetzt auch *bis zu 8* 3D-Korrekturtabellen gespeichert werden. Es können aber *nicht* zwei 3D-Korrekturtabellen gleichzeitig den Scan-Kopf-Steuerungs-Ports zugewiesen werden (siehe **select_cor_table**). • Die Parameter `k`, `Phi` und `Offset` gibt es nicht mehr. Damit sind beim Laden der Korrekturtabelle auch keine Transformationen (Skalierung, Rotation, Translation) der Korrekturtabelle mehr möglich. Solche Koordinaten-Transformationen können jetzt mit **set_scale**, **set_angle** und **set_offset** zusätzlich zur mit **set_matrix** definierten Koordinatentransformation festgelegt werden. • **select_cor_table** wird automatisch aufgerufen, siehe Hinweise oben und Abschnitt "Hinweise", Seite 184. • **load_correction_file** kehrt erst nach Ende der Korrekturbewegung zum Anwenderprogramm zurück (siehe Hinweise oben).
RTC5→RTC6	Veränderte Funktionalität. • Mit **load_correction_file** können 8 Korrekturtabellen gleichzeitig in den Speicher der RTC6 geladen werden. • Es gibt keine Überschneidung mit anderweitig benutzten Speicherbereichen mehr.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**number_of_correction_tables**

Kontrollbefehl	load_disk
Funktion	Lädt die zuvor mit **save_disk** in einer binären Datei gespeicherten indizierten Zeichen, Zeichenfolgen und Unterprogramme in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” und liefert die Anzahl der tatsächlich geladenen Listenbefehle zurück.
Aufruf	`NoOfLoadedCommands = load_disk( Name, Mode )`
Parameter	`Name` Dateiname oder NULL. Als Zeiger auf einen nullterminierten ANSI-String.
	`Mode` Legt fest, wie geladen werden soll. Als 32-Bit-Wert ohne Vorzeichen. = 0: Die internen Verwaltungstabellen für indizierte Zeichen, Zeichenfolgen und Unterprogramme werden initialisiert (alle alten Referenzierungen gehen dabei verloren), und der Eingabe-Zeiger wird auf den Anfang von “Liste 3” gesetzt (durch den folgenden Ladevorgang werden die dort gespeicherten Listenbefehle überschrieben). > 0: Es findet keine Initialisierung der internen Verwaltungstabellen statt (alle alten Referenzierungen werden zunächst beibehalten, aber während des Ladevorgangs je nach Inhalt der Datei durch andere ersetzt oder ergänzt), und der Eingabe-Zeiger wird hinter das zuletzt (mit **load_char**) gespeicherte Zeichen, die zuletzt (mit **load_text_table**) gespeicherte Zeichenfolge oder das zuletzt (mit **load_sub**) gespeicherte Unterprogramm gesetzt (durch den folgenden Ladevorgang werden die Listenbefehle von alten indizierten Zeichen, Zeichenfolgen und Unterprogrammen *nicht* überschrieben).
Rückgabe	Anzahl der mit **load_disk** geladenen Listenbefehle. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_load_disk**
Hinweise	• Für `Name` = NULL werden nur die internen Verwaltungstabellen (wie bei `Mode` = 0) initialisiert (es findet dann kein Ladevorgang statt, es muss keine “leere” Datei bereitgestellt werden). Ansonsten darf **load_disk** nur zum Einlesen von Dateien verwendet werden, die mit **save_disk** gespeichert wurden. • **load_disk** führt für alle in der angegebenen Datei abgelegten indizierten Zeichen, Zeichenfolgen und Unterprogramme einen entsprechenden **load_char**-, **load_text_table**- oder **load_sub**-Befehl aus. Dadurch werden die zugehörigen Listenbefehle in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” geschrieben und gemäß der in der Datei gespeicherten Indexzuordnung die Einträge in den internen Verwaltungstabellen vorgenommen. Falls in der angegebenen Datei für einen bestimmten Index weder ein Zeichen noch eine Zeichenfolge oder ein Unterprogramm abgelegt ist, werden für diesen Index auch keine Listenbefehle geladen und – für `Mode` > 0 – gegebenenfalls bereits vorhandene Einträge in den internen Verwaltungstabellen unverändert gelassen. Dadurch ist auch ein Hinzu-Laden von indizierten Zeichen, Zeichenfolgen und Unterprogrammen aus unterschiedlichen Dateien möglich.

Kontrollbefehl	load_disk
Hinweise (Forts.)	• Zusammen mit **save_disk** kann **load_disk** u.a. zur Defragmentierung des geschützten RTC6-Listenspeicher-Bereichs “Liste 3” und zum nachträglichen Schützen von Unterprogrammen verwendet werden, siehe Abschnitt “Nachträglich nichtindizierte Unterprogramme schützen und konvertieren”, Seite 119 und Abschnitt “Indexverwaltung und Defragmentierung”, Seite 118. Wenn die Zeichen, Zeichenfolgen und Unterprogramme auf verschiedene Dateien verteilt sind, erreicht man eine Defragmentierung, wenn die erste Datei mit `Mode` = 0 und die nachfolgenden mit `Mode` > 0 geladen werden, wenn keine Indices in verschiedenen Dateien gleichzeitig vorkommen. Speicherlücken im geschützten RTC6-Listenspeicher-Bereich “Liste 3”, die beim Entreferenzieren von indizierten Zeichen, Zeichenfolgen oder Unterprogrammen entstehen, werden dadurch geschlossen. • Wird beim Laden das Ende des geschützten RTC6-Listenspeicher-Bereichs “Liste 3” vor dem Ende der Datei (EOF) erreicht, werden alle weiteren Listenbefehle der Datei ignoriert. Auch unvollständige Zeichen, Zeichenfolgen und Unterprogramme werden (wie bei individuellen **load_char**-, **load_text_table**- oder **load_sub**-Aufrufen) nicht gespeichert. Vor einem **load_disk**-Aufruf sollte daher sichergestellt werden, dass der geschützte RTC6-Listenspeicher-Bereich “Liste 3” in ausreichender Größe jenseits des RTC6-Listenspeicher-Bereichs (“Liste 1” und “Liste 2”) konfiguriert ist. **save_disk** gibt die Anzahl der in der Datei gespeicherten Listenbefehle zurück. Beim Besitzwechsel einer Karte ist es Aufgabe des Benutzers sicherzustellen, dass die Informationen über die Speicherkonfiguration konsistent sind (`Mem1` und `Mem2` werden aus der RTC6-DLL abgefragt, nicht aus der Karte – siehe auch **get_config_list** und Kapitel 6.7.1 “Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm”, Seite 131). • Wenn die angegebene Datei korrupt ist und nicht zu Ende gelesen werden kann, werden nur alle bis dahin vollständig eingelesenen Zeichen, Zeichenfolgen und Unterprogramme gespeichert. • **load_disk** wird nicht ausgeführt, wenn: – “Liste 3” kein RTC6-Listenspeicher-Bereich zugewiesen ist, d.h. `Mem1` + `Mem2` = 2^{23} (**get_last_error**-Returncode `RTC6_PARAM_ERROR`) – der BUSY-Listenausführungsstatus gesetzt ist (**get_last_error**-Returncode `RTC6_BUSY`) – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist (**get_last_error**-Returncode `RTC6_BUSY`) • **load_disk** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt). Benutzer sind aber dafür verantwortlich, dass keine Befehle überschrieben werden, die noch gebraucht werden, z.B. wenn **set_wait** aus einem indizierten Unterprogramm heraus aufgerufen wurde. • Während der Laufzeit von **load_disk** werden: – Keine weiteren Befehle ausgeführt – Externe Starts unterdrückt • **load_disk** überprüft die von **save_disk** gespeicherte Versionsinformation und vergleicht diese mit der aktuellen Laufzeitversion. Beide müssen übereinstimmen. Anderenfalls wird **load_disk** nicht ausgeführt (**get_last_error**-Returncode `RTC6_VERSION_MISMATCH`).

Kontrollbefehl	load_disk
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 615: Versionsprüfung.
Verweise	**save_disk**

Kontrollbefehl	load_fly_2d_table	
Funktion	Lädt eine 2D-Tabelle aus einer ASCII-Textdatei für eine **set_fly_2d**-"Processing on the fly"-Anwendung mit 2D-Encoderkompensation für xy-Verfahrtische, siehe Abschnitt "2D-Encoderkompensation für xy-Verfahrtische", Seite 261.	
Aufruf	`NoOfDataPoints = load_fly_2d_table( Name, No )`	
Parameter	`Name`	Name der Textdatei oder NULL. Die Textdatei kann eine oder mehrere Tabellen enthalten. Als Zeiger auf einen nullterminierten ANSI-String.
	`No`	Legt fest, welche Tabelle aus der Textdatei geladen werden soll. `No` entspricht der Erweiterung `<No>` von `[Fly2DTable<No>]` am Anfang der gewünschten Tabelle. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Im Fehlerfall ein positiver Fehlercode. Im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte. Als 32-Bit-Wert mit Vorzeichen.	
	< 0	Erfolg. Der Absolutwert des Rückgabewerts ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert, siehe auch Abschnitt "Für die 2D-Kompensationstabellen gelten die folgenden Regeln:", Seite 262.
	0	Für `Name` = NULL (siehe auch Hinweise).
	1	Keine gültigen Datenpunkte gefunden (wohl aber Tabelle `No`).
	2	Out of Memory (nicht genügend Windows-Arbeitsspeicher).
	3	Datei nicht gefunden.
	4	DSP-Speicher-Fehler.
	5	Busy-Fehler, Karte ist BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus, kein Download (**get_last_error**-Returncode `RTC6_BUSY`).
	8	Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC6_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC6_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer konnte in der Datei nicht gefunden werden.
Multi-Board-Befehlsname	**n_load_fly_2d_table**	
Hinweise	• `No` = `<No>` lädt Tabelle Nummer 1; `No` = `<No>` + 65.536 lädt Tabelle Nummer 2. • Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für die 2D-Encoderkompensation in die Text-Datei sind im Abschnitt "Für die 2D-Kompensationstabellen gelten die folgenden Regeln:", Seite 262 beschrieben. Die größten vorkommenden Stützstellen sollten den Wertebereich –524.288…+524.287 nicht überschreiten (sonst könnte Genauigkeit verloren gehen). Zur Laufzeit dürfen die aktuellen Encoderwerte (inkl. Referenzwerte) die größten in der Tabelle angegebenen Werte nicht überschreiten. Anderenfalls erfolgt ein Clipping. • **load_fly_2d_table** überschreibt eine zuvor für die 2D-Encoderkompensation geladene Tabelle. • Für `Name` = NULL wird eine 0-Korrektur-Tabelle für die 2D-Encoderkompensation geladen und die Tabelle als ungültig markiert (siehe **init_fly_2d**).	

Kontrollbefehl	load_fly_2d_table
Hinweise (Forts.)	• **load_fly_2d_table** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_fly_2d_table** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Laufzeit von **load_fly_2d_table** werden externe Starts unterdrückt. • Vor dem Laden einer Tabelle führt **load_fly_2d_table** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Es können jetzt 2 Tabellen gleichzeitig geladen werden.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_2d**, **init_fly_2d**, **get_fly_2d_offset**

Kontrollbefehl	load_jump_table	
Funktion	Lädt eine Wertetabelle mit Jump-Delay-Datenpunkten aus einer ASCII-Textdatei (oder führt alternativ eine automatische Ermittlung am angegebenen Scan-System durch) und bestimmt durch lineare Interpolation die interne Jump-Delay-Tabelle für 2D-Sprünge, die im Sprungmodus ausgeführt werden.	
Aufruf	`NoOfDataPoints = load_jump_table( Name, No, PosAck, MinDelay, MaxDelay, ListPos )`	
Parameter	`Name`	Siehe **load_jump_table_offset**.
	`No`	Siehe **load_jump_table_offset**.
	`PosAck`	Siehe **load_jump_table_offset**.
	`MinDelay`	Siehe **load_jump_table_offset**.
	`MaxDelay`	Siehe **load_jump_table_offset**.
	`ListPos`	Siehe **load_jump_table_offset**.
Rückgabe	Siehe **load_jump_table_offset**.	
Multi-Board-Befehlsname	**n_load_jump_table**	
Hinweise	• **load_jump_table** ist identisch zu **load_jump_table_offset** mit `Offset` = 0.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**load_jump_table_offset**	

Kontrollbefehl	load_jump_table_offset	
Funktion	Lädt eine Wertetabelle mit Jump-Delay-Datenpunkten aus einer ASCII-Textdatei (oder führt alternativ eine automatische Ermittlung am angegebenen Scan-System durch) und bestimmt durch lineare Interpolation die interne Jump-Delay-Tabelle für 2D-Sprünge, die im Sprungmodus ausgeführt werden.	
Aufruf	`NoOfDataPoints = load_jump_table_offset( Name, No, PosAck, Offset, MinDelay, MaxDelay, ListPos )`	
Parameter	`Name`	Name der Textdatei oder NULL. Die Textdatei kann eine oder mehrere Tabellen enthalten. Als Zeiger auf einen nullterminierten ANSI-String.
	`No`	Gibt an: • Für `Name` = Dateiname, welche Tabelle aus der Textdatei geladen werden soll. Der Parameter entspricht der Erweiterung `<No>` von `[JumpTable<No>]` am Anfang der gewünschten Tabelle. • Für `Name` = NULL, welches Scan-System (oder welcher Scan-Kopf-Anschluss) für die automatische Ermittlung verwendet werden soll. Zulässige Werte: [1, 2]. Als 32-Bit-Wert ohne Vorzeichen.
	`PosAck`	Positionstoleranzwert in [Bits] (nur bei automatischer Ermittlung relevant). Als 32-Bit-Wert ohne Vorzeichen.
	`Offset`	Offset in [10 µs], der zu allen automatisch ermittelten Delays aufaddiert wird (nur bei automatischer Ermittlung relevant). Als 32-Bit-Wert mit Vorzeichen.
	`MinDelay`	Minimales Jump-Delay in [10 µs] (nur bei automatischer Ermittlung relevant). Als 32-Bit-Wert ohne Vorzeichen.
	`MaxDelay`	Maximales Jump-Delay in [10 µs] (nur bei automatischer Ermittlung relevant). Als 32-Bit-Wert ohne Vorzeichen.
	`ListPos`	Listenposition (im Bereich von Liste 1 oder 2) für sechs Listenbefehle, die überschrieben werden dürfen (nur bei automatischer Ermittlung relevant). Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Fehlercode.	Als 32-Bit-Wert mit Vorzeichen.
	–1…– 50	Erfolg für `Name` = Dateiname. Der Absolutwert des Rückgabewerts ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert, siehe auch Abschnitt "Hinweise zum Laden der ermittelten Jump-Delay-Werte", Seite 231.
	–1024	Erfolg für `Name` = NULL: Tabelle wurde automatisch ermittelt.
	1	Keine gültigen Datenpunkte gefunden (wohl aber Tabelle `No`).
	3	Datei nicht gefunden.
	4	Verify-Fehler: DSP-Speicher-Fehler.
	5	Busy-Fehler, Karte ist BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus, kein Download (**get_last_error**-Returncode `RTC6_BUSY`).
	8	Access-Fehler: Karte durch anderes Anwenderprogramm reserviert (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`).

Kontrollbefehl	load_jump_table_offset
Rückgabe (Forts.)	10 Nur für Name = NULL: Param-Error: HeadNo oder ListPos unzulässig. 11 PCI-Fehler beim Download (**get_last_error**-Returncode RTC6_SEND_ERROR). 13 Die angegebene Tabellennummer konnte in der Datei nicht gefunden werden.
Multi-Board-Befehlsname	**n_load_jump_table_offset**
Hinweise	• Zur Verwendung von **load_jump_table_offset** siehe Abschnitt "Sprunglängen-abhängige Jump-Delays", Seite 230. • Siehe auch Kapitel 8.1.5 "Sprungmodus", Seite 229. • Die Formatvorschriften zum Eintragen der Tabelle mit den Jump-Delay-Datenpunkten in die Text-Datei sind in Abschnitt "Hinweise zum Laden der ermittelten Jump-Delay-Werte", Seite 231 beschrieben. Beim Laden der Tabelle bestimmt die RTC6 durch lineare Interpolation geeignete Werte für den ganzen Sprunglängen-Bereich. • **load_jump_table_offset** überschreibt jede zuvor für den Sprungmodus geladene Jump-Delay-Tabelle. • Für Name = Dateiname wird die Tabelle mit der Nummer No geladen. Die restlichen Parameter spielen dann keine Rolle. • Für Name = NULL wird die Jump-Delay-Tabelle für Kopf No automatisch ermittelt und geladen (dafür muss zuvor mit **set_jump_mode** der Sprungmodus erfolgreich freigegeben und aktiviert worden sein (Flag = 1)). Dabei werden die Parameter PosAck, Offset, MinDelay, MaxDelay sowie ListPos verwendet (siehe Abschnitt "Jump-Delay-Tabelle automatisch ermitteln", Seite 233). • Wirksam wird der Sprungmodus bei nachfolgenden 2D-Sprüngen erst dann, wenn er mit **set_jump_mode** freigegeben und aktiviert oder mit **set_jump_mode_list** aktiviert wurde. • **load_jump_table_offset** wird nicht ausgeführt (**get_last_error**-Returncode RTC6_BUSY), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_jump_table_offset** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Laufzeit von **load_jump_table_offset** werden externe Starts unterdrückt. • Vor dem Laden einer Tabelle führt **load_jump_table_offset** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt. • Die Tabelle kann mit **create_dat_file** gespeichert werden.
RTC4→RTC6	Neuer Befehl. Für diesen Befehl gibt es keinen RTC4-Kompatibilitätsmodus.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**load_jump_table**, **set_jump_mode**, **set_jump_mode_list**, **get_jump_table**, **set_jump_table**, **create_dat_file**

Kontrollbefehl	load_list
Funktion	Öffnet den RTC6-Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Eingabe-Zeiger an die angegebene *relative* Position in der gewünschten Liste (“Liste 1” oder “Liste 2”), aber nur, wenn die Liste nicht gerade abgearbeitet wird (BUSY-Listenstatus nicht gesetzt) oder wenn sie bereits abgearbeitet ist (USED-Listenstatus ist gesetzt).
Aufruf	`NoOfOpenedList = load_list( ListNo, Pos )`
Parameter	`ListNo` Listennummer der Liste, in die der Eingabe-Zeiger gesetzt werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [0…3]. Nur die beiden niederwertigsten Bits werden berücksichtigt. = 0: Diejenige Liste wird geöffnet, für die momentan nicht der BUSY-Listenstatus gesetzt ist. “Liste 1” wird geöffnet, wenn für beide Listen nicht der BUSY-Listenstatus gesetzt ist. = 1: “Liste 1” wird geöffnet, wenn für sie nicht der BUSY-Listenstatus gesetzt ist (BUSY1 nicht gesetzt). = 2: “Liste 2” wird geöffnet, wenn für sie nicht der BUSY-Listenstatus gesetzt ist (BUSY2 nicht gesetzt). = 3: Diejenige Liste wird geöffnet, für die momentan nicht der BUSY-Listenstatus, aber der USED-Listenstatus gesetzt ist. Ist für beide Listen der BUSY-Listenstatus nicht gesetzt aber der USED-Listenstatus: diejenige Liste wird geöffnet, die beim nächsten automatischen Listenwechsel ausgeführt werden würde. Für `Mem2` = 0 (siehe **config_list**) wird “Liste 1” geöffnet, wenn: • für sie nicht der BUSY-Listenstatus gesetzt ist (`ListNo` = 0…2) • für sie nicht der BUSY-Listenstatus, aber der USED-Listenstatus gesetzt ist (`ListNo` = 3) `Pos` Position des Eingabe-Zeigers (Offset bezüglich des Listenanfangs der jeweiligen Liste). Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Rückgabe	Nummer der geöffneten Liste [1 oder 2] im Erfolgsfall, sonst 0. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_load_list**
Hinweise	• Falls **load_list** erfolgreich war, wird der nächste Listenbefehl an der angegebenen Adresse gespeichert und alle nachfolgenden Listenbefehle an den darauffolgenden Adressen in der gewählten Liste. • Falls **load_list** nicht erfolgreich war (Rückgabewert 0), wird keine Liste geöffnet und der Eingabe-Zeiger auf eine ungültige Position gesetzt. Es ist dann so lange keine Eingabe von weiteren Listenbefehlen möglich, bis der Eingabe-Zeiger (z.B. mit Hilfe eines erneuten **load_list** mit positivem Resultat, mit **set_start_list_pos** etc.) wieder gültig gesetzt wird. Es liegt in der Verantwortung des Benutzers, auf den Rückgabewert 0 geeignet zu reagieren. **load_list** führt *keine* Warteschleife aus und blockiert *nicht* die Ausführung nachfolgender Kontrollbefehle.

Kontrollbefehl	load_list
Hinweise (Forts.)	• **load_list** (`ListNo` = 3) hilft u.a. beim alternierenden Listenwechsel, um gezielt auf das Abarbeiten einer Liste zu warten (siehe Abschnitt "Alternierender Listenwechsel", Seite 114), ohne dass der Listenstatus separat abgefragt werden muss. Ein unbeabsichtigtes Überschreiben von Befehlen, die noch ausgeführt werden sollen, wird dabei automatisch vermieden. Soll dagegen ein *unbedingtes* Laden ausgeführt werden, kann z.B. **set_start_list_pos** verwendet werden. • Nach einem erfolgreichen Check der Listennummer (BUSY-Listenstatus nicht gesetzt und gegebenenfalls USED-Listenstatus gesetzt) wirkt **load_list** wie **set_start_list_pos** (siehe Hinweise dort). • Für `ListNo` = 0…2 sollte bei der Verwendung von **load_list** berücksichtigt werden, dass sich der BUSY-Listenstatus einer Liste u.U. bereits zwischen dem Öffnen der Liste mit **load_list** und dem tatsächlichen Laden von Listenbefehlen ändern kann, z.B. wenn in der Zwischenzeit ein zuvor mit **auto_change** oder **start_loop** eingestellter automatischer Listenwechsel erfolgt. Immer, wenn wie in einem solchen Fall das Laden einer Liste u.U. zeitgleich mit dem Ausführen derselben Liste erfolgen kann, muss der Benutzer dafür sorgen, dass der Eingabe-Zeiger nicht vom Ausgabe-Zeiger überholt wird. • Für `ListNo` = 3 kann dagegen mit **load_list** sichergestellt werden, dass eine Liste wirklich erst unmittelbar nach dem Abarbeiten (und nach einem dabei gegebenenfalls erfolgten automatischen Listenwechsel) zum Laden geöffnet wird. Insbesondere wenn für keine Liste der BUSY-Listenstatus gesetzt ist, aber für beide der USED-Listenstatus (z.B. nach der Initialisierung, einem **stop_execution** oder einem externen Stopp), ist die geöffnete Listennummer mit dem nächsten automatischen Listenwechsel synchronisiert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_start_list_pos**

Kontrollbefehl	load_position_control	
Funktion	Lädt eine Tabelle mit Datenpunkten aus einer ASCII-Textdatei und bestimmt durch lineare Interpolation die Skalierungsfunktion für die Positions-abhängige Laser-Steuerung (radiale Korrektur).	
Aufruf	`NoOfDataPoints = load_position_control( Name, No )`	
Parameter	`Name`	Name der Textdatei oder NULL. Die Textdatei kann eine oder mehrere Tabellen enthalten. Als Zeiger auf einen nullterminierten ANSI-String.
	`No`	Legt fest, welche Tabelle aus der Textdatei geladen werden soll. Der Parameter entspricht der Erweiterung `<No>` von `[PositionCtrlTable<No>]` am Anfang der gewünschten Tabelle. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Im Fehlerfall ein positiver Fehlercode. Im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte. Als 32-Bit-Wert mit Vorzeichen.	
	Wert	Beschreibung
	–1…– 50	Erfolg. Der Absolutwert des Rückgabewerts ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert, siehe auch Abschnitt "Hinweise zum Laden der Skalierungsfunktion", Seite 210.
	–256	Für `Name` = NULL (siehe auch Hinweise).
	1	Keine gültigen Datenpunkte gefunden (wohl aber Tabelle `No`).
	3	Datei nicht gefunden.
	4	DSP-Speicher-Fehler.
	5	Busy-Fehler, Karte ist BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus, kein Download (**get_last_error**-Returncode `RTC6_BUSY`).
	8	Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC6_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC6_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer konnte in der Datei nicht gefunden werden.
Multi-Board-Befehlsname	**n_load_position_control**	
Hinweise	• Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für die Positions-abhängige Laser-Steuerung in die Text-Datei sind im Abschnitt "Hinweise zum Laden der Skalierungsfunktion", Seite 210 beschrieben. Beim Laden der Tabelle bestimmt die RTC6 durch lineare Interpolation geeignete Werte für den ganzen Ansteuerbereich. • **load_position_control** überschreibt jede zuvor für die Positions-abhängige Laser-Steuerung geladene Skalierungsfunktion. • Für `Name` = NULL wird (wie bei Initialisierung mit **load_program_file**) für den vollen Positionsbereich die Skalierungsfunktion *Scale*(*Position*)=1.0 geladen, so dass effektiv keine positionsabhängige Korrektur durchgeführt wird.	

Kontrollbefehl	load_position_control
Hinweise (Forts.)	• Wirksam wird die Positions-abhängige Laser-Steuerung bei nachfolgenden [*]mark[*]-Befehlen oder Arc-Befehlen erst dann, wenn sie mit **set_auto_laser_control** initialisiert wurde. Deaktiviert wird die Positions-abhängige Laser-Steuerung mit **set_auto_laser_control** (`Ctrl` = 0) oder durch Laden von *Scale*(*Position*)=1.0. Siehe auch Abschnitt "Positions-abhängige Laser-Steuerung", Seite 210. • **load_position_control** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_position_control** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Laufzeit von **load_position_control** werden externe Starts unterdrückt. • Vor dem Laden einer Tabelle führt **load_position_control** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt. • Die Tabelle kann mit **create_dat_file** gespeichert werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_auto_laser_control**, **create_dat_file**

Kontrollbefehl	load_position_control_2d_ctrl
Funktion	Lädt eine Laser-Korrekturtabelle aus einer ASCII-Textdatei für die 2D-Positions-abhängige Laser-Steuerung.
Aufruf	`ErrorNo = load_position_control_2d_ctrl( Name )`
Parameter	`Name` Name der Textdatei oder NULL. Die Textdatei kann eine oder mehrere Tabellen enthalten. Als Zeiger auf einen nullterminierten ANSI-String.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Erfolg. 1 Fehler beim Parsen oder nicht genügend Datenpunkte gefunden. 2 Out of Memory (nicht genügend Windows-Arbeitsspeicher). 3 Datei nicht gefunden (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). 4 DSP-Speicher-Fehler. 5 Busy-Fehler, Karte hat BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus, kein Download (**get_last_error**-Returncode `RTC6_BUSY`). 6 Kalibrierfaktor ungültig (K stimmt nicht überein mit der Korrekturtabelle, die dem ersten Scan-Kopf zugewiesen ist, `get_head_para( HeadNo = 1, ParaNo = 1 )`. 8 Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`). 11 PCI-Fehler (**get_last_error**-Returncode `RTC6_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC6_VERIFY_ERROR`).
Multi-Board-Befehlsname	**n_load_position_control_2d_ctrl**
Hinweise	• Wirksam wird die 2D-Positions-abhängige Laser-Steuerung bei nachfolgenden [*]mark[*]-Befehlen und Arc-Befehlen erst dann, wenn sie mit **set_auto_laser_control** initialisiert wurde. • Deaktiviert wird die 2D-Positions-abhängige Laser-Steuerung durch: – `set_auto_laser_control( Ctrl = 0 )` – `Name` = NULL (weiterhin wird die Positions-abhängige Laser-Steuerung, Seite 210 aktiviert, falls diese zuvor aktiv war) – Aktivieren der Positions-abhängige Laser-Steuerung, Seite 210 • Siehe Abschnitt "2D-Positions-abhängige Laser-Steuerung", Seite 218. • Siehe auch **load_position_control**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 644, OUT 647.
Verweise	**load_position_control**

Kontrollbefehl	load_program_file
Funktion	• Setzt die RTC6 zurück (Software-Reset) • Initialisiert den RTC6-Listenspeicher • Führt einen DSP-Speicher-Check aus • Lädt RTC6RBF.rbf • Lädt RTC6OUT.out (bei RTC6 Ethernet-Karten stattdessen RTC6ETH.out) • Lädt RTC6DAT.dat • Startet den DSP
Aufruf	`ErrorNo = load_program_file( Path )`
Parameter	Path — Name des Verzeichnisses, in dem RTC6OUT.out, RTC6RBF.rbf und RTC6DAT.dat sind. Als Zeiger auf einen nullterminierten String.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 — Erfolg. 1 — Entfällt. 2 — Die Karte läuft nicht. Falls bei wiederholtem Aufruf nicht erfolgreich, ist ein Hardware-Reset erforderlich. 3 — RTC6DAT.dat oder RTC6RBF.rbf nicht gefunden. 4 — Entfällt. 5 — Nicht genügend Windows-Speicher. 6 — Ein anderes Anwenderprogramm hat die Karte bereits in Besitz genommen. 7 — Versions-Fehler: RTC6-DLL-Version, RTC6RBF.rbf-Version und RTC6OUT.out/RTC6ETH.out-Version passen nicht zusammen. Siehe auch **load_program_file**. 8 — RTC6-Kartentreiber nicht gefunden. 9 — Das Laden von RTC6OUT.out oder RTC6ETH.out schlug fehl oder hat falsches Format oder sonstiger Fehler. 10 — Entfällt. 11 — FPGA-Firmware-Fehler: das Laden von RTC6RBF.rbf schlug fehl. 12 — Fehler beim Öffnen/Lesen von RTC6DAT.dat. 13 — Entfällt. 14 — DSP-Speicher-Fehler (extern). 15 — Verify-Memory Fehler. 16 — DSP-Speicher-Fehler (intern). 17 — Ethernet-Fehler. Siehe **eth_get_last_error** und **eth_get_error**. 18 — Nur RTC6 Ethernet-Karte: NAND-Speicherfehler.
Multi-Board-Befehlsname	**n_load_program_file**

Kontrollbefehl	load_program_file
Hinweise	• Für `Path` = NULL wird der Pfad des aktuellen Arbeitsverzeichnisses des Anwenderprogramms verwendet. Achtung: Dieses stimmt nicht immer mit dem Verzeichnis überein, aus dem das Anwenderprogramm gestartet wird. Ein Wechsel des aktuellen Arbeitsverzeichnisses findet z.B. statt, wenn über den Windows-Explorer eine Datei aus einem anderen Verzeichnis ausgewählt wird (es sei denn, bei der Einbindung des Explorer-Fensters in das Anwenderprogramm wird das Flag "NoChangeDir" gesetzt). • **load_program_file** muss nach jedem Hardware-Reset (Einschalten) bei der Initialisierung der RTC6-Karte am Beginn des jeweils ersten Anwenderprogramms ausgeführt werden, siehe Abschnitt "Karte initialisieren", Seite 101. **load_program_file** muss auch – z.B. bei der Übernahme der Karte durch ein weiteres Anwenderprogramm (siehe **acquire_rtc**) – ausgeführt werden, wenn die Karte wieder in den Standard-Zustand zurückgesetzt werden soll. • Wenn mehrere RTC6-Karten als Master und Slave miteinander verbunden sind, dann muss auf allen Karten **load_program_file** aufgerufen worden sein, bevor einzelne Karten mit weiteren Befehlen initialisiert und betrieben werden können, siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127. • Nach der Ausführung von **load_program_file** ist: – Der Laserfokus im Zentrum des Bildfelds (0|0) – Die Laser-Steuerung *deaktiviert* – Zum Zustand der verschiedenen Ausgänge siehe Kapitel 7.4.1 "Lasersteuersignale freigeben, aktivieren und schalten", Seite 192 (LASERON, LASER1, LASER2), Kapitel 9.1.1 "16-Bit-Digital-Ausgang", Seite 306 (EXTENSION 1-Stiftleiste), Kapitel 9.1.2 "8-Bit-Digital-Ausgang", Seite 307 (EXTENSION 2-Stiftleiste), Kapitel 9.1.3 "2-Bit-Digital-Ausgang", Seite 307 (LASER-Buchse), Kapitel 9.1.4 "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 308 (LASER-Buchse). • *Vorsicht! Sporadische* **load_program_file***-Aufrufe in Anwenderprogrammen:* – *Widersprechen ganz allgemein der in Kapitel 5.6 "Sichere Einschalt- und Ausschaltreihenfolge", Seite 94 vorgeschriebenen, sicheren Einschaltreihenfolge* – *Bergen die Gefahr von Personenschäden und/oder Sachschäden (es sind Fälle bekannt geworden, in denen Laser mit schlechter Elektrik emittiert haben)* *Sollten Sie auf sporadische* **load_program_file***-Aufrufe in Ihrem Anwenderprogramm absolut nicht verzichten können, müssen Sie Meldungen implementieren, die die Benutzer entsprechend warnen und zu Handlungen auffordern, die diese Gefahren verhindern.* • **load_program_file** lädt *keine* Korrekturtabellen. Auch 1to1-Tabellen müssen daher explizit angefordert werden, siehe **load_correction_file**. Bereits vorher geladene Korrekturtabellen bleiben bei **load_program_file** erhalten. • **load_correction_file** weist zwar eine Korrekturtabelle mit `select_cor_table( 1, 0 )` zu, führt aber keine Galvanometerscanner-Bewegung zur korrigierten Ausgabeposition durch. • **load_program_file** kehrt erst dann wieder zum aufrufenden Anwenderprogramm zurück, wenn die Initialisierung des DSP komplett abgeschlossen ist.

Kontrollbefehl	load_program_file
Hinweise (Forts.)	• **load_program_file** führt automatisch **stop_execution** aus, wenn – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • Die Dateien `RTC6OUT.out`, `RTC6ETH.out`, `RTC6RBF.rbf` und `RTC6DAT.dat` sind im RTC6-Software-Paket enthalten. Zur Kennzeichnung und Verwaltung unterschiedlicher Softwareversionen werden die Dateien zusätzlich gezippt ausgeliefert (die `zip`-Dateinamen `RTC6<…>_<Version>.zip` enthalten die jeweiligen Versionsnummern). Kopieren oder entpacken Sie die drei Dateien auf die Festplatte Ihres PCs. • Die RTC6-DLL und die Dateien `RTC6OUT.out`, `RTC6ETH.out`, `RTC6RBF.rbf` und `RTC6DAT.dat` können nicht in beliebigen Programmversionen miteinander kombiniert werden (die `zip`-Dateien enthalten jeweils eine Text-Datei mit entsprechenden Versions-Informationen). **load_program_file** führt eine Versionsprüfung durch. Liegt ein Versionsfehler vor, bleiben die geladenen Dateien zwar im Speicher der RTC6, die Karte wird aber unmittelbar nach der Versionsprüfung mit **release_rtc** freigegeben und damit für weitere Befehle gesperrt, außer für solche, die keine Zugriffsberechtigung erfordern (**get_last_error**-Returncode `RTC6_ACCESS_DENIED` | `RTC6_VERSION_MISMATCH`). Lediglich **load_program_file** kann dann noch aufgerufen werden, um eine korrekte Programmversion zu laden. Dazu wird temporär die Zugriffsberechtigung angefordert, nach dem Download (wenn die Karte nicht von einem anderen Anwenderprogramm übernommen ist) aber wieder zurückgegeben (**load_program_file** führt kein **acquire_rtc** aus).
RTC4→RTC6	• Der Befehlsparameter gibt bei der RTC6 einen Verzeichnisnamen, bei der RTC4 einen Dateinamen an. • **load_program_file** lädt jetzt drei Dateien, die mit festen Formaten und Namensbezeichnungen (`RTC6OUT.out`, `RTC6RBF.rbf`, `RTC6DAT.dat`) bereitgestellt werden müssen (s.o.). • Nach Ausführung von **load_program_file** ist die Laser-Steuerung *deaktiviert*.
RTC5→RTC6	Veränderte Funktionalität. • Hinweise z.B. zur Migration von Quellcode von RTC5-Anwenderprogrammen: **load_program_file** ist abwärtskompatibel zur RTC5 und kann unverändert weiter benutzt werden. Ihr Anwenderprogramm muss nicht geändert werden. Allerdings ist zu beachten, dass: – Die Prüfung auf Status BUSY-Listenausführungsstatus entfällt (d.h. es kann niemals der Fehlercode 13 ausgegeben werden). – Die Datenübertragung an der McBSP-Schnittstelle ohne Warnung oder Fehlermeldung deaktiviert wird. – Eine laufende Liste wird ohne Warnung oder Fehlermeldung beendet (wie mit dem Kontrollbefehl **stop_execution**). Beachten Sie dort auch die Hinweise z.B. zur Spiegelstellung. – Andere Fehlercodes bereitgestellt werden, siehe Tabelle oben.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Kontrollbefehl	load_stretch_table	
Funktion	Lädt eine Tabelle aus einer ASCII-Textdatei für eine erweiterte 3D-Korrektur, siehe Abschnitt "Erweiterte 3D-Korrektur", Seite 252.	
Aufruf	`NoOfDataPairs = load_stretch_table( Name, No, TableNo )`	
Parameter	`Name`	Name der Textdatei oder NULL. Die Textdatei kann eine oder mehrere Tabellen enthalten. Als Zeiger auf einen nullterminierten ANSI-String.
	`No`	Als 32-Bit-Wert mit Vorzeichen. • Für `No` ≥ 0 wird mit diesem Parameter angegeben, welche Tabelle aus der Textdatei geladen werden soll. Der Parameter entspricht der Erweiterung `<No>` von `[StretchTable<No>]` am Anfang der gewünschten Tabelle. • `No` < 0: Reserviert.
	`TableNo`	Die bereits geladene 3D-Korrekturtabelle, der die erweiterte Korrektur zugeordnet wird. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: 1…8. Siehe auch number_of_correction_tables.
Rückgabe	Im Fehlerfall ein positiver Fehlercode. Im Erfolgsfall die negative Anzahl der gefundenen Datenpaare. Als 32-Bit-Wert mit Vorzeichen.	
	Wert	Beschreibung
	< 0	Erfolg. Der Absolutwert des Rückgabewerts ist gleich der Zahl der gültigen Datenpaare, die in der Tabelle gefunden wurden.
	0	Reserviert.
	2	Out of Memory (nicht genügend Windows-Arbeitsspeicher).
	3	Datei nicht gefunden.
	4	DSP-Speicher-Fehler.
	5	Busy-Fehler, Karte ist BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus, kein Download (**get_last_error**-Returncode `RTC6_BUSY`).
	6	Daten-Fehler: es fehlen Datenpaare.
	11	PCI-Fehler (**get_last_error**-Returncode `RTC6_SEND_ERROR`).
	13	Die angegebene Tabellennummer konnte in der Datei nicht gefunden werden.
	15	Verify-Fehler (**get_last_error**-Returncode `RTC6_VERIFY_ERROR`, nur mit aktivierter Download-Überprüfung, siehe **set_verify**, möglich).
Multi-Board-Befehlsname	**n_load_stretch_table**	
Hinweise	• Details zur erweiterten 3D-Korrektur siehe Abschnitt "Erweiterte 3D-Korrektur", Seite 252. • Die erweiterte 3D-Korrektur wird beim Aufruf von **select_cor_table** mit umgeschaltet. • Eine erfolgreich geladene Tabelle aktiviert die neue erweiterte 3D-Korrektur. Dabei überschreibt **load_stretch_table** eine zuvor für die erweiterte 3D-Korrektur geladene Tabelle der gleichen `TableNo`. • Falls `Name` *nicht* NULL ist, aber keine Tabelle erfolgreich gelesen wird, dann gibt **load_stretch_table** einen Fehlercode zurück, hat sonst aber keine Auswirkung (d.h. eine zuvor erfolgreich geladene Tabelle bleibt weiterhin gültig).	

Kontrollbefehl	load_stretch_table
Hinweise (Forts.)	• Für `Name` = NULL deaktiviert **load_stretch_table** eine zuvor mit **load_stretch_table** aktivierte erweiterte 3D-Korrektur. • **load_stretch_table** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_stretch_table** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Laufzeit von **load_stretch_table** werden externe Starts unterdrückt. • Vor dem Laden einer Tabelle führt **load_stretch_table** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Veränderte Funktionalität. • Zusätzlicher Parameter `TableNo`. Damit kann jeder 3D-Korrekturtabelle eine eigene erweiterte 3D-Korrektur zugeordnet werden.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_cor_table**

Kontrollbefehl	load_sub
Funktion	Versieht ein durch nachfolgende Listenbefehle definiertes Unterprogramm mit dem angegebenen Index und lädt es in den geschützten RTC6-Listenspeicher-Bereich “Liste 3”.
Aufruf	`load_sub( Index )`
Parameter	`Index` Index des indizierten Unterprogramms. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Multi-Board-Befehlsname	**n_load_sub**
Hinweise	• Es können 1024 indizierte Unterprogramm gespeichert werden. Für `Index` > 1023 wird **load_sub** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Die Adressen im geschützten RTC6-Listenspeicher-Bereich “Liste 3”, wo die jeweiligen Unterprogramme gespeichert werden, werden automatisch ermittelt und intern verwaltet. • Indizierte Unterprogramme müssen jeweils mit einem **list_return**-Aufruf abgeschlossen werden. Erst damit werden die Befehle endgültig gespeichert, die Startadresse in die interne Verwaltungstabelle eingetragen und auch ein Flush der gepufferten Listeneingabe ausgelöst, siehe Kapitel 6.4.1 “Listen laden”, Seite 108. Anderenfalls (der Eingabe-Zeiger wird ohne vorhergehendes **list_return** verändert) ist das Unterprogramm mit diesem Index nicht verfügbar. • Ein indiziertes Unterprogramm wird nicht gespeichert, falls nicht zuvor der geschützte RTC6-Listenspeicher-Bereich “Liste 3” in ausreichender Größe jenseits von “Liste 1” und “Liste 2” konfiguriert wurde. • Ist **list_return** der unmittelbar auf **load_sub** folgende Befehl, wird das entsprechende Unterprogramm aus der internen Verwaltungstabelle gelöscht. • Indizierte Unterprogramme werden mit dem **sub_call**-Befehl unter Angabe des Indexes aufgerufen (siehe Abschnitt “Allgemeines zum Aufruf von Unterprogrammen”, Seite 117). • Beachten Sie auch alle Hinweise in Abschnitt “Indizierte Unterprogramme”, Seite 116.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_return**, **sub_call**, **load_char**, **load_text_table**

Kontrollbefehl	load_text_table
Funktion	Versieht eine durch nachfolgende Listenbefehle definierte Zeichenfolge mit dem angegebenen Index und lädt sie in den geschützten RTC6-Listenspeicher-Bereich “Liste 3”.
Aufruf	`load_text_table( Index )`
Parameter	`Index` Index der indizierten Zeichenfolge. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…41].
Multi-Board-Befehlsname	**n_load_text_table**
Hinweise	• Es können 42 indizierte Zeichenfolgen (für die Beschriftung von Uhrzeit, Datum und Seriennummern mit weiteren Befehlen) gespeichert werden. Dabei gilt die folgende Zuordnung: – `Index` = 0…9: Ziffern für die Beschriftung von Uhrzeit und Datum [0…9] – `Index` = 10…21: Monate [Januar…Dezember] – `Index` = 22…28: Wochentage [Sonntag…Samstag] – `Index` = 29: Leerzeichen für die Beschriftung von Seriennummern – `Index` = 30…39: Ziffern für die Beschriftung von Seriennummern [0…9] – `Index` = 40: Text für “a.m.“ – `Index` = 41: Text für “p.m.“ Für `Index` > 41 wird **load_text_table** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Wenn auch die Ziffern statt über einzelne Zeichen mit **mark_char** über **mark_text** definiert werden, kann auch dafür der Zeichensatz nachträglich noch umgeschaltet werden (siehe **select_char_set**). • Die Adressen im geschützten RTC6-Listenspeicher-Bereich “Liste 3”, wo die jeweiligen Zeichenfolgedefinitionen gespeichert werden, werden automatisch ermittelt und intern verwaltet. Die Verwaltung erfolgt separat von indizierten Unterprogrammen (siehe **load_sub**) und Zeichendefinitionen (siehe **load_char**). • Indizierte Zeichenfolgedefinitionen müssen jeweils mit einem **list_return**-Aufruf abgeschlossen werden. Erst damit werden die Befehle endgültig gespeichert, die Startadresse in die interne Verwaltungstabelle eingetragen und auch ein Flush der gepufferten Listeneingabe ausgelöst, siehe Kapitel 6.4.1 “Listen laden”, Seite 108. Anderenfalls (der Eingabe-Zeiger wird ohne vorhergehendes **list_return** verändert) ist die Zeichenfolgedefinition mit diesem Index nicht verfügbar. • Eine indizierte Zeichenfolgedefinition wird nicht gespeichert, falls nicht zuvor der geschützte RTC6-Listenspeicher-Bereich “Liste 3” in ausreichender Größe jenseits von “Liste 1” und “Liste 2” konfiguriert wurde. • Ist **list_return** der unmittelbar auf **load_text_table** folgende Befehl, wird die entsprechende Zeichenfolgedefinition aus der internen Verwaltungstabelle gelöscht. • Beachten Sie auch alle Hinweise im Kapitel 6.5.2 “Zeichensätze und Zeichenfolgen”, Seite 121.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_return**, **load_sub**, **load_char**

Kontrollbefehl	load_varpolydelay	
Funktion	Lädt eine Tabelle mit Datenpunkten für die Skalierungsfunktion des benutzerdefinierten “Variablen Polygon-Delays“, Abschnitt “Benutzer-definierte “Variable Polygon-Delays““, Seite 156, aus einer ASCII-Textdatei.	
Aufruf	`NoOfDataPoints = load_varpolydelay( Name, No )`	
Parameter	`Name`	Name der Textdatei oder NULL. Die Textdatei kann eine oder mehrere Tabellen enthalten. Als Zeiger auf einen nullterminierten ANSI-String.
	`No`	Diejenige Tabelle aus der Textdatei, die geladen werden soll. Der Parameter entspricht der Erweiterung `<No>` von `[VarPolyTable<No>]` am Anfang der gewünschten Tabelle. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Im Fehlerfall ein positiver Fehlercode. Im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte. Als 32-Bit-Wert mit Vorzeichen.	
	Wert	Beschreibung
	–1…– 50	Erfolg. Der Absolutwert des Rückgabewerts ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert, siehe Abschnitt “Benutzer-definierte “Variable Polygon-Delays““, Seite 156.
	–1024	Für `Name` = NULL: es wurde die gemäß 1 – cos(φ) initialisierte Tabelle intern (wie bei Programmstart) geladen, siehe Abbildung 40.
	1	Keine gültigen Datenpunkte gefunden (wohl aber Tabelle `No`).
	3	Datei nicht gefunden.
	4	DSP-Speicher-Fehler.
	5	Busy-Fehler, Karte ist BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus, kein Download (**get_last_error**-Returncode `RTC6_BUSY`).
	8	Die Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC6_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC6_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer wurde in der Datei nicht gefunden.
Multi-Board-Befehlsname	**n_load_varpolydelay**	
Hinweise	• Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für das Benutzer-definierte “Variable Polygon-Delay” in die Text-Datei sind im Abschnitt “Benutzer-definierte “Variable Polygon-Delays““, Seite 156 beschrieben. Beim Laden der Tabelle bestimmt die RTC6 durch lineare Interpolation geeignete Werte für den gesamten Winkelbereich. • **load_varpolydelay** überschreibt jede zuvor für das “Variable Polygon-Delay” geladene Tabelle. • Für `Name` = NULL wird (wie bei Initialisierung mit **load_program_file**) die interne Standard-Tabelle für das “Variable Polygon-Delay” (1 – cos(φ), siehe Abbildung 40) geladen.	

Kontrollbefehl	load_varpolydelay
Hinweise (Forts.)	• **load_varpolydelay** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist (eine Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_varpolydelay** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Laufzeit von **load_varpolydelay** werden externe Starts unterdrückt. • Vor dem Laden einer Tabelle führt **load_varpolydelay** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt. • Die Tabelle kann mit **create_dat_file** gespeichert werden.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: • Um zur internen Standard-Polygon-Delay-Tabelle zurückzukehren ist *kein* Reset oder erneutes Programmladen mit **load_program_file** mehr erforderlich (s.o. `Name` = NULL). • Für die ASCII-Textdatei sind beliebige Dateinamenerweiterungen zulässig.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**load_program_file**, **set_delay_mode**, **create_dat_file**

Kontrollbefehl	load_z_table
Funktion	Wie **load_z_table_20b**. Jedoch für einen Fokuslängenwert *f* im RTC4-Kompatibilitätsbereich [–32.768…+32.767]).
Einschränkung	Wie **load_z_table_no_20b**.
Aufruf	`ErrorNo = load_z_table( A, B, C )`
Parameter	`A` Wie **load_z_table_no**.
	`B` Wie **load_z_table_no**.
	`C` Wie **load_z_table_no**.
Rückgabe	Wie **load_z_table_no_20b**.
Multi-Board-Befehlsname	**n_load_z_table**
Hinweise	• Wie **load_z_table_no_20b**. • `load_z_table( A, B, C )` ist gleichbedeutend mit `load_z_table_no( A, B, C, 0 )`.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: veränderte Wertebereiche und Fehlercodes. Wenn die korrekten Kalibrierfaktoren verwendet werden (siehe Kapitel "3D-Befehle", Seite 250), können bei gleichem 3-Achsen-Scan-System für RTC4- und RTC6-Karten dieselben ABC-Koeffizienten verwendet werden.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_z_distance**, **read_abc_from_file**, **write_abc_to_file**, **load_z_table_no**, **select_cor_table**, **load_z_table_20b**

Kontrollbefehl	load_z_table_20b
Funktion	Lädt die Koeffizienten *A*, *B* und *C* in die aktuell zugewiesene 3D-Korrekturtabelle. Für einen Fokuslängenwert *l* im RTC6-20-Bit-Bereich [–524.288…+524.287].
Einschränkung	Wie **load_z_table_no_20b**.
Aufruf	`ErrorNo = load_z_table_20b( A, B, C )`
Parameter	`A` Koeffizient *A* der parabolischen Funktion $z_{out} = A + Bl + Cl^2$, die zur Berechnung der z-Ausgabewerte verwendet wird (Fokuslängenwert *l* im RTC6-20-Bit-Bereich [–524.288…+524.287]). Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–1.073.741.824,0…+1.073.741.824,0]. Überlaufende Werte werden auf die Randwerte geclippt.. `B` Wie `A` (analog). Zulässiger Wertebereich: [–2.048,0…+2.048,0]. `C` Wie `A` (analog). Zulässiger Wertebereich: [–1,0…+1,0].
Rückgabe	Wie **load_z_table_no_20b**.
Multi-Board-Befehlsname	**n_load_z_table_20b**
Hinweise	• **load_z_table_20b** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **load_z_table_20b** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • **load_z_table_20b** sollte immer *nach* **load_correction_file** aufgerufen werden, da **load_correction_file** die drei Koeffizienten auf die in der geladenen Korrekturtabelle festgelegten Werte setzt. • Die `ABC`-Werte gehen durch **select_cor_table** verloren. • Siehe auch weitere Hinweise bei **load_z_table_no**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 631, OUT 632.
Verweise	**get_z_distance**, **read_abc_from_file_20b**, **write_abc_to_file_20b**, **load_z_table_no_20b**, **select_cor_table**, **load_z_table_no_20b**

Kontrollbefehl	load_z_table_no	
Funktion	Wie **load_z_table_no_20b**. Jedoch für einen Fokuslängenwert *f* im RTC4-Kompatibilitätsbereich [–32.768…+32.767].	
Einschränkung	Wie **load_z_table_no_20b**.	
Aufruf	`ErrorNo = load_z_table_no( A, B, C, No )`	
Parameter	`A`	Koeffizient *A* der parabolischen Funktion $z_{out} = A + Bf + Cf^2$, die zur Berechnung der z-Ausgabewerte verwendet wird (Fokuslängenwert *f* im RTC4-Kompatibilitätsbereich [–32.768…+32.767]). Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–67.108.864,0…67.108.864,0]. Überlaufende Werte werden auf die Randwerte geclippt.
	`B`	Wie `A` (analog). Zulässiger Wertebereich: [–2.048,0…2.048,0].
	`C`	Wie `A` (analog). Zulässiger Wertebereich: [–16,0…16,0].
	`No`	Nummer der 3D-Korrekturtabelle, der die drei Koeffizienten `A`, `B`, `C` zugeordnet werden sollen. Siehe auch **number_of_correction_tables**.
Rückgabe	Wie **load_z_table_no_20b**.	
Multi-Board-Befehlsname	**n_load_z_table_no**	
Hinweise	• Wie **load_z_table_no_20b**. • `load_z_table_no( A, B, C, 0 )` ist gleichbedeutend mit `load_z_table( A, B, C )`. • `load_z_table_no( A, B, C, No )` ist gleichbedeutend mit `load_z_table_no_20b( A × 16, B , C × 1/16, No )`. Die Werte sind nach einem **select_cor_table** oder **select_cor_table_list** verloren.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 611, OUT 611, RBF 616.	
Verweise	**load_z_table**, **get_z_distance**, **read_abc_from_file**, **write_abc_to_file**, **select_cor_table**	

Kontrollbefehl	load_z_table_no_20b	
Funktion	Lädt die Koeffizienten *A*, *B* und *C* und weist sie der 3D-Korrekturtabelle `No` zu. Für einen Fokuslängenwert *l* im RTC6-20-Bit-Bereich [–524.288…+524.287].	
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), liefert **load_z_table_no_20b** Fehlercode 12 oder 13 zurück und hat sonst keine Wirkung.	
Aufruf	`ErrorNo = load_z_table_no_20b( A, B, C, No )`	
Parameter	`A`	Koeffizient *A* der parabolischen Funktion $z_{out} = A + Bl + Cl^2$, die zur Berechnung der z-Ausgabewerte verwendet wird (Fokuslängenwert *l* im RTC6-20-Bit-Bereich [–524.288…+524.287]). Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–1.073.741.824,0…+1.073.741.824,0]. Überlaufende Werte werden auf die Randwerte geclippt.
	`B`	Wie `A` (analog). Zulässiger Wertebereich: [–2.048,0…+2.048,0].
	`C`	Wie `A` (analog). Zulässiger Wertebereich: [–1,0…+1,0].
	`No`	Nummer der 3D-Korrekturtabelle, der die drei Koeffizienten `A, B, C` zugeordnet werden sollen. Siehe auch **number_of_correction_tables**.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. Die Fehler-Bits mit den Werten 1…64 können auch kombiniert vorkommen, aber nicht zusammen mit Fehlercodes 11…15, letztere auch nur separat. Die Warnungen 12 und 13 werden nur dann zurückgegeben, wenn kein anderer Fehler vorliegt.	
	0	Kein Fehler.
	1	`A` hat den maximal zulässigen Wert überschritten.
	2	`A` hat den minimal zulässigen Wert unterschritten.
	4	`B` hat den maximal zulässigen Wert überschritten.
	8	`B` hat den minimal zulässigen Wert unterschritten.
	16	`C` hat den maximal zulässigen Wert überschritten.
	32	`C` hat den minimal zulässigen Wert unterschritten.
	64	Ausführung verweigert (wahrscheinlich BUSY-Listenausführungsstatus- oder INTERNAL-BUSY-Listenausführungsstatus-Fehler, für eine genaue Ursache siehe **get_last_error**).
	11	Zugriff verweigert.
	12	Option "3D" ist nicht freigeschaltet.
	13	Es ist aktuell keine 3D-Korrekturtabelle zugeordnet.
	14	RTC6-Kartentreiber nicht gefunden.
	15	Ungültige Tabellennummer (> **number_of_correction_tables**).
Multi-Board-Befehlsname	**n_load_z_table_no_20b**	

Kontrollbefehl	load_z_table_no_20b
Hinweise	• **load_z_table_no_20b** wird nur zur Neu-Kalibrierung der z-Achse in einem 3-Achsen-Scan-System benötigt. Zur Bestimmung geeigneter Koeffizienten siehe Abschnitt "z-Achsen-Kalibrierung überprüfen", Seite 178. Es können positive oder negative Koeffizienten angegeben werden. Die Koeffizienten sollten möglichst so gewählt werden, dass alle z-Ausgabewerte $z_{out} = A + B\mathrm{I} + C\mathrm{I}^2$ im RTC6-20-Bit-Bereich [–524.288…+524.287] liegen. • Vor dem **load_z_table_no_20b** unmittelbar nachfolgenden Listenbefehl wird ein fließender Übergang der letzten z-Position in die geänderte Position mit Sprunggeschwindigkeit ausgeführt. Durch **select_cor_table** kann dies auch sofort erzwungen werden. Dadurch werden Zeitverzögerungen bei einem externen Start vermieden. • Mit **get_table_para** können die Koeffizienten *A*, *B* und *C* aus der aktuell geladenen 3D-Korrekturtabelle (und mit **get_head_para** aus der aktuell zugewiesenen 3D-Korrekturtabelle) ausgelesen werden. • Mit **load_z_table_no_20b** können die drei Koeffizienten `A, B, C` der 3D-Korrekturtabelle `No` zugeordnet werden. Sie werden bei der Ausführung von **select_cor_table** mit umgeschaltet. • `No` = aktuell zugewiesene Tabellennummer überschreibt ebenfalls die aktuellen Werte und macht sie somit sofort verfügbar, wenn gerade keine Liste abgearbeitet wird. Ansonsten werden sie nur gespeichert und sind erst nach einem **select_cor_table** oder **select_cor_table_list** verfügbar. • Wenn gerade keine Liste abgearbeitet wird, überschreibt `No` = 0 nur die aktuellen Werte. • `load_z_table_no_20b( A, B, C, No )` ist gleichbedeutend mit `load_z_table_no( A × 1/16, B × 16 , C × 16, No )`. Die Werte sind nach einem **select_cor_table** oder **select_cor_table_list** verloren. Für `B` und `C` gilt: Entsprechend verringert sich der zulässige Wertebereich.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 631, OUT 632.
Verweise	**load_z_table_no_20b**, **load_z_table_20b**, **get_z_distance**, **read_abc_from_file_20b**, **write_abc_to_file_20b**, **select_cor_table**

Kontrollbefehl	load_zoom_correction_file
Multi-Board-Befehlsname	**n_load_zoom_correction_file**
Hinweise	• **load_zoom_correction_file** ist z.B. im Handbuch für den intelli*WELD* II FT beschrieben.

Normaler Listenbefehl	**long_delay**
Funktion	Wartet mit der weiteren Ausführung der Liste für die angegebene Zeit.
Aufruf	`long_delay( Delay )`
Parameter	`Delay` Delay-Zeit. In Bits. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 μs. Zulässiger Wertebereich: 0 ≤ delay ≤ (2^{32}–1).
Multi-Board-Befehlsname	**n_long_delay**
Hinweise	• **long_delay** schaltet die Signale für den "Laser active"-Betrieb nach einem LaserOff-Delay aus, wartet ein etwaiges Scanner-Delay ab und wartet mit der weiteren Ausführung der Liste für die angegebene Zeit. • **long_delay** sollte immer nach Änderung des Lampenstroms eines YAG-Lasers angewendet werden, damit sich eine konstante Laserleistung einstellen kann. • **list_nop** entspricht `long_delay( 1 )`.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_defocus_list**

Normaler Listenbefehl	**mark_abs**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`mark_abs( X, Y )`
Parameter	`X` Absolute x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`Y` Wie `X` (analog).
Multi-Board-Befehlsname	**n_mark_abs**
Hinweise	• Wenn die Markiergeschwindigkeit zuvor nicht explizit über **set_mark_speed** oder **set_mark_speed_ctrl** festgelegt wurde, wird die Markierung mit der voreingestellten Markiergeschwindigkeit von 1.000 *Bits/ms* ausgeführt. • Die Signale für den "Laser active"-Betrieb werden automatisch am Anfang der Markierung eingeschaltet (oder bleiben nach einem unmittelbar vorhergehenden [*]mark[*]-Befehl oder Bogen-Befehl eingeschaltet). Die eingestellten Scanner-Delays und Laser-Delays werden dabei berücksichtigt, siehe Kapitel 7.2 "Delayeinstellungen – Scan-Kopf-Steuerung und Laser-Steuerung abstimmen", Seite 148. Beachten Sie, dass im Sky-Writing-Modus andere Delays ausgeführt werden. Ausnahme: [*]mark[*]-Befehle der Länge 0, siehe Abschnitt "Hinweise", Seite 152.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mark_speed**, **set_scanner_delays**, **mark_rel**, **arc_abs**, **timed_mark_abs**

Normaler Listenbefehl	**mark_abs_3d**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **mark_abs_3d** wie **mark_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`mark_abs_3d( X, Y, Z )`
Parameter	`X` Absolute x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`Y` Wie `X` (analog).
	`Z` Wie `X` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287].
Multi-Board-Befehlsname	**n_mark_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **mark_abs_3d** gleich wie **mark_abs** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X`, `Y` und `Z` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den für `Z` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_abs**, **mark_rel_3d**, **timed_mark_abs_3d**

Unverzögerter Kurzer Listenbefehl	**mark_char**
Funktion	Markiert ein indiziertes Zeichen.
Aufruf	`mark_char( Char )`
Parameter	`Char` Index des zu markierenden indizierten Zeichens. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023]). Es gilt: `Char` = Zeichensatznummer × 256 + ASCII-Nummer des Zeichens (die Zeichensatznummer zählt dabei von 0…3).
Multi-Board-Befehlsname	**n_mark_char**
Hinweise	• **mark_char** liest die Startadresse des indizierten Zeichens entsprechend dem ausgewählten Index aus der internen Verwaltungstabelle und ruft dann **list_call** auf (siehe auch Hinweise dort), der dann die entsprechende Befehlsliste startet. • **mark_char** startet sowohl indizierte Zeichen im geschützten Bereich (die mit **load_char**, **load_disk** oder **copy_dst_src** geladen und/oder referenziert wurden) als auch indizierte Unterprogramme im ungeschützten RTC6-Listenspeicher-Bereich (die mit **set_char_pointer** oder **copy_dst_src** als Zeichen referenziert wurden). • Ist für den angegebenen Index kein Zeichen referenziert, dann unterbleibt der Sprung und die Ausführung wird mit dem der aufrufenden Stelle unmittelbar folgenden Befehl fortgefahren. Evtl. wird ein **list_continue** ausgeführt, siehe Abschnitt "Normaler Listenbefehl, Kurzer Listenbefehl, Variabler Listenbefehl und Multipler Listenbefehl", Seite 339. • Ob für einen bestimmten Index ein Zeichen referenziert ist, kann mit **get_char_pointer** (`Char`) abgefragt werden. Falls *kein* Zeichen referenziert ist, gibt **get_char_pointer** einen Wert von "–1" (d.h. 2^{32}–1) zurück. • Für `Char` > 1023 wird **mark_char** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Absolute Vektor-Befehle und Bogen-Befehle werden nach Aufruf mit **mark_char** auch absolut ausgeführt. Wenn das Zeichen an unterschiedlichen Stellen im Bildfeld ausgeführt werden soll, darf die Befehlsliste nur relative [*]mark[*]-Befehle, Bogen-Befehle oder Sprung-Befehle enthalten oder es ist **mark_char_abs** zu verwenden. • Das aufgerufene Zeichen sollte keinen **mark_text**-Befehl enthalten, der selbst dieses Zeichen enthält. Dessen Text wird *nicht* markiert. Das aufgerufene Zeichen selbst ist dann u.U. nicht vollständig. • Siehe auch Kapitel 6.5.2 "Zeichensätze und Zeichenfolgen", Seite 121.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_char_abs**, **mark_text**, **load_char**, **set_char_pointer**, **get_char_pointer**

Unverzögerter Kurzer Listenbefehl	**mark_char_abs**
Funktion	Markiert ein indiziertes Zeichen. In der dazu aufgerufenen Befehlsliste (s.u.) werden absolute Vektor-Befehle und Bogen-Befehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt.
Aufruf	`mark_char_abs( Char )`
Parameter	`Char` Index des zu markierenden indizierten Zeichens. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023]). Es gilt: `Char` = Zeichensatznummer × 256 + ASCII-Nummer des Zeichens (die Zeichensatznummer zählt dabei von 0…3).
Multi-Board-Befehlsname	**n_mark_char_abs**
Hinweise	• **mark_char_abs** liest die Startadresse des indizierten Zeichens entsprechend dem ausgewählten Index aus der internen Verwaltungstabelle und ruft dann den Befehl **list_call_abs** auf (siehe auch Hinweise dort), der dann die entsprechende Befehlsliste startet. • Enthält die Befehlsliste des indizierten Zeichens keine absoluten Befehle, besteht zwischen **mark_char_abs** und **mark_char** kein Unterschied.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_char**

Normaler Listenbefehl	**mark_date**
Funktion	Markiert einen Teil des zuvor mit **time_fix**, **time_fix_f** oder **time_fix_f_off** gespeicherten Datums im ausgewählten Format an der aktuellen Position.
Aufruf	`mark_date( Part, Mode )`
Parameter	`Part` Gibt an, welcher Teil des Datums markiert wird. = 0: Jahreszahl (nur die beiden letzten Ziffern). = 1: Monat (in anwenderspezifischer Darstellungsart). = 2: Tag im Monat (gemäß dem üblichen gregorianischen Datum). = 3: Wochentag (in anwenderspezifischer Darstellungsart). = 4: Tag im Jahr (gemäß julianischem Datum, siehe auch **time_fix_f_off**). = 5: Jahreszahl (vierstellig). = 6: Monat als Ziffern (Januar = 1,…). = 7: Wochentag als Ziffern (Sonntag = 1,…). Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…7].
	`Mode` Format. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…3]. a) Betrifft nur `Part` = 2, 4, 6 oder 7: Anzahl der führenden Nullstellen für die Tagesnummer. Bit #0 = 0: Führende Nullstellen werden unterdrückt. Bit #0 = 1: Monatstag (`Part` = 2) immer zweistellig. Jahrestag (`Part` = 4) immer dreistellig. Monat (`Part` = 6) immer zweistellig. Wochentag (`Part` = 7) immer zweistellig. b) Betrifft nur `Part` = 0, 2 oder 4…7: Auswahl des Schriftsatzes für die Ziffernmarkierung. Bit #1 = 0: Es werden die mit **load_text_table** oder **set_text_table_pointer** für die Ziffern [0…9] definierten indizierten Zeichenfolgen markiert. Bit #1 = 1: Es werden die mit **load_char** oder **set_char_pointer** für die Ziffern [0…9] definierten indizierten Zeichen markiert. Der gewünschte Zeichensatz kann mit **select_char_set** ausgewählt werden. Für `Part` = 1 oder 3 werden Monate oder Wochentage gemäß den mit **load_text_table** oder **set_text_table_pointer** definierten indizierten Zeichenfolgen (Index = 10…28) markiert (der Parameter `Mode` wird hier ignoriert). Für die Markierung als Ziffern kann aber auch `Part` = 6 oder 7 verwendet werden (`Mode` wird dann berücksichtigt).
Multi-Board-Befehlsname	**n_mark_date**

Normaler Listenbefehl	**mark_date**
Hinweise	• Vor einer Datumsmarkierung sollte (nach jedem Neustart) ein Abgleich zwischen RTC6 und PC-Zeit stattgefunden haben (siehe **time_update**) und der aktuelle (zu markierende) Datumswert mit **time_fix**, **time_fix_f** oder **time_fix_f_off** gespeichert worden sein (siehe Kapitel 7.5 "Uhrzeit, Datum und Seriennummern markieren", Seite 222). • Das vollständige Datum wird durch mehrfachen Aufruf von **mark_date** markiert. • **mark_date** liest entsprechend des gespeicherten Datums und des ausgewählten Datumteils die Startadressen der zugehörigen indizierten Zeichenfolgen oder Zeichen aus der internen Verwaltungstabelle und ruft dann den Befehl **list_call** entsprechend oft auf (siehe auch Hinweise dort), der dann die entsprechenden Befehlslisten startet. Die Befehlslisten müssen die Markierungsvorschriften für die Ziffern 0…9 und für die Monats- und Wochentagsbezeichnungen enthalten (siehe Abschnitt "Indizierte Zeichenfolgen für Uhrzeit, Datum und Seriennummer definieren", Seite 122). Nicht-definierte Zeichenfolgen oder Zeichen werden ignoriert (d.h. nicht markiert). Die aufgerufenen indizierten Zeichenfolgen können auch Aufrufe von indizierten Zeichen (**mark_char** oder **mark_char_abs**) und ganzen Texten (**mark_text** oder **mark_text_abs**) enthalten. Im letzteren Fall kann der Zeichensatz bei Bedarf (vor dem Markieren mit **mark_date**) mit **select_char_set** umgeschaltet werden (siehe auch Kapitel 6.5.2 "Zeichensätze und Zeichenfolgen", Seite 121). • Für `Part` > 7 und/oder `Mode` > 3 wird **mark_date** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Absolute Vektor-Befehle und Bogen-Befehle werden nach Aufruf mit **mark_date** auch absolut ausgeführt. Wenn Datumsmarkierungen an unterschiedlichen Stellen im Bildfeld ausgeführt werden sollen, dürfen die entsprechenden Zeichenfolgen (oder Zeichen) nur relative [*]mark[*]-Befehle, Bogen-Befehle oder Sprung-Befehle enthalten oder es ist **mark_date_abs** zu verwenden. • Bei der Beschriftung sowohl des gregorianischen als auch des julianischen Datums erfolgt die Umschaltung zwischen zwei Tagen um 00:00 Uhr. Bei beiden Datumsversionen werden die Schaltjahre berücksichtigt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**time_fix**, **time_fix_f**, **time_fix_f_off**, **load_text_table**, **set_text_table_pointer**, **mark_date_abs**

Normaler Listenbefehl	**mark_date_abs**
Funktion	Markiert einen Teil des zuvor mit **time_fix**, **time_fix_f** oder **time_fix_f_off** gespeicherten Datums im ausgewählten Format an der aktuellen Position. In den dabei ausgeführten indizierten Zeichenfolgen oder Zeichen (s.u.) werden absolute Vektor-Befehle und Bogen-Befehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt.
Aufruf	`mark_date_abs( Part, Mode )`
Parameter	`Part` Siehe **mark_date**.
	`Mode` Siehe **mark_date**.
Multi-Board-Befehlsname	**n_mark_date_abs**
Hinweise	• **mark_date_abs** wirkt wie **mark_date**. Allerdings erfolgt der interne Aufruf der indizierten Zeichenfolgen (oder Zeichen) mit **list_call_abs** statt mit **list_call**. • Enthalten die Befehlslisten der aufgerufenen indizierten Zeichenfolgen (oder Zeichen) keine absoluten Befehle, besteht zwischen **mark_date_abs** und **mark_date** kein Unterschied.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_date**

Normaler Listenbefehl	**mark_ellipse_abs**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines ellipsenförmigen Bogens um den angegebenen Mittelpunkt (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`mark_ellipse_abs( X, Y, Alpha )`
Parameter	`X` Absolute x-Koordinate des Ellipsen-Mittelpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y` Wie `X` (analog).
	`Alpha` Winkel zwischen der (mit **set_ellipse** definierten) Ellipsen-Halbachse a und der x-Achse in Grad (der Winkel bezieht sich auf die positive x-Richtung, ein positives Vorzeichen bedeutet im Gegenuhrzeigersinn). Als Wert im 64-Bit-IEEE Gleitkommaformat. `Alpha` wird auf den Wertebereich [0…<360°] normiert.
Multi-Board-Befehlsname	**n_mark_ellipse_abs**
Hinweise	• Die Parameter von **mark_ellipse_abs** legen nur die Lage und Orientierung des auszuführenden Bogens fest. Vor der Ausführung von **mark_ellipse_abs** muss mit **set_ellipse** seine Form definiert worden sein. Zur Bedeutung der einzelnen Parameter siehe auch Abschnitt "Ellipse-Befehle", Seite 141. • Stimmt der mit **mark_ellipse_abs** und **set_ellipse** festgelegte Startpunkt des Bogens nicht mit der aktuellen Position überein, dann wird vor der Markierung ein "Harter Sprung" an den Startpunkt ausgeführt, siehe auch Hinweise im Abschnitt "Ellipse-Befehle", Seite 141. • Siehe auch alle Hinweise unter **arc_abs**.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_ellipse**, **mark_ellipse_rel**, **set_mark_speed**, **set_scanner_delays**, **arc_abs**

Normaler Listenbefehl	**mark_ellipse_rel**	
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines ellipsenförmigen Bogens um den angegebenen Mittelpunkt (relative Koordinatenwerte) im 2D-Bildfeld.	
Aufruf	`mark_ellipse_rel( dX, dY, Alpha )`	
Parameter	`dX`	Relative x-Koordinate des Ellipsen-Mittelpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`dY`	Wie `dX` (analog).
	`Alpha`	Winkel zwischen der (mit **set_ellipse** definierten) Ellipsen-Halbachse a und der x-Achse (siehe **mark_ellipse_abs**). In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_mark_ellipse_rel**	
Hinweise	• Die Koordinaten des Ellipsen-Mittelpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **mark_ellipse_rel** identisch mit **mark_ellipse_abs** (siehe Hinweise dort).	
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.	
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**mark_ellipse_abs**, **set_ellipse**	

Normaler Listenbefehl	**mark_rel**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`mark_rel( dX, dY )`
Parameter	`dX` Relative x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`dY` Wie `dX` (analog).
Multi-Board-Befehlsname	**n_mark_rel**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **mark_rel** identisch mit **mark_abs** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_abs**, **mark_rel_3d**, **timed_mark_rel**

Normaler Listenbefehl	**mark_rel_3d**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **mark_rel_3d** wie **mark_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`mark_rel_3d( dX, dY, dZ )`
Parameter	`dX` Relative x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`dY` Wie `dX` (analog).
	`dZ` Wie `dX` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287].
Multi-Board-Befehlsname	**n_mark_rel_3d**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **mark_rel_3d** identisch mit **mark_abs_3d** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX`, `dY` und `dZ` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den für `dZ` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_abs_3d**, **mark_abs**, **mark_rel**, **timed_mark_rel_3d**

Normaler Listenbefehl	**mark_serial**
Funktion	Markiert die aktuelle Seriennummer des zuletzt mit **select_serial_set_list** ausgewählten Seriennummernsatzes (oder nach **load_program_file** des Seriennummernsatzes 0) im ausgewählten Format an der aktuellen Position. Die Seriennummer wird danach (optional) automatisch erhöht.
Aufruf	`mark_serial( Mode, Digits )`
Parameter	`Mode` Auswahl des Seriennummernformats und (De-) Aktivierung der automatischen Seriennummernerhöhung. Als 32-Bit-Wert ohne Vorzeichen. `Mode` = $20 \times M_1 + 10 \times M_2 + M_3$ Zulässiger Wertebereich: für M_1 [0, 1], für M_2 [0, 1], für M_3 [0…2]. a) Auswahl des Schriftsatzes für die Ziffernmarkierung. • M_1 = 0: Es werden die mit **load_text_table** oder **set_text_table_pointer** für die Ziffern [30…39] definierten indizierten Zeichenfolgen markiert. • M_1 = 1: Es werden die mit **load_char** oder **set_char_pointer** für die Ziffern [0…9] definierten indizierten Zeichen markiert. Der gewünschte Zeichensatz kann (zuvor) mit **select_char_set** ausgewählt werden. b) Erhöhung der Seriennummer nach der Markierung. • M_2 = 0: Seriennummer wird nach der Markierung erhöht. • M_2 = 1: Seriennummer wird nach der Markierung *nicht* erhöht. c) Markierung führender Nullen. • M_3 = 0: Führende Nullen werden als 0 markiert, abh. von M_1 gemäß indizierter Zeichenfolge (Index = 30) oder gemäß indiziertem Zeichen '0'. • M_3 = 1: Führende Nullen werden unterdrückt (linksbündige Markierung). • M_3 = 2: Führende Nullen werden als Leerzeichen markiert (rechtsbündige Markierung), abh. von M_1 gemäß indizierter Zeichenfolge (Index = 29) oder gemäß indiziertem Zeichen ' '. `Digits` Anzahl zu markierender Stellen. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0-12]. Zu große Werte werden geclippt.
Multi-Board-Befehlsname	**n_mark_serial**
Hinweise	• Die erste zu markierende Seriennummer sollte zuvor mit **set_serial**, **set_serial_step** oder **set_serial_step_list** festgelegt werden, sonst wird mit 0 gestartet. Die Startseriennummer ist maximal 10-stellig. • Die Seriennummer wird durch jeden Aufruf eines **mark_serial** entsprechend M_3 formatiert und für M_2 = 0 noch vor der Ausführung der eigentlichen Markierung automatisch erhöht. Die Erhöhungsschrittweite ist dabei 1, wenn nicht mit **set_serial_step** oder **set_serial_step_list** eine andere angegeben wird. • Die aktuelle Seriennummer kann mit **get_list_serial** ausgelesen werden, z.B. um nach einem Listenabbruch festzustellen, ob die aktuelle Nummer bereits hochgezählt wurde oder noch nicht.

Normaler Listenbefehl	**mark_serial**
Hinweise (Forts.)	• Nach einem Überlauf bei 10^{Digits} beginnt die Seriennummernzählung wieder bei 0. Mit dem Kontrollbefehl **set_max_counts** lässt sich die maximale Anzahl an externen Starts und damit die maximale Anzahl an Beschriftungen einstellen. Dabei werden alle Markierungen auch aus unterschiedlichen Seriennummernsätzen immer zusammen gezählt. • Mit `Digits` = 0 wird eine Leermarkierung ausgeführt. Für $M_2 = 0$ findet dabei eine Seriennummernerhöhung um 1 statt (nicht um die mit **set_serial_step** oder **set_serial_step_list** anders eingestellte Schrittweite!). Dies kann nützlich sein, wenn eine einzelne Seriennummer ausgelassen werden soll (gegebenenfalls n-mal wiederholen; n = Schrittweite), kann aber auch dazu verwendet werden, um indirekt die Seriennummer um eine *zusätzliche* Schrittweite zu erhöhen. • **mark_serial** liest für jede zu markierende Seriennummernziffer die Startadresse der zugehörigen indizierten Zeichenfolge (oder – für $M_1 = 1$ – des indizierten Zeichens) aus der internen Verwaltungstabelle und ruft dann **list_call** entsprechend oft auf (siehe auch Hinweise dort), der die entsprechenden Befehlslisten startet. Die Befehlslisten müssen die Markierungsvorschriften für die Ziffern 0…9 enthalten (siehe Abschnitt "Indizierte Zeichenfolgen für Uhrzeit, Datum und Seriennummer definieren", Seite 122). Nicht-definierte Zeichenfolgen oder Zeichen werden ignoriert (d. h. nicht markiert). Die aufgerufenen indizierten Zeichenfolgen können auch Aufrufe von indizierten Zeichen (**mark_char** oder **mark_char_abs**) und ganzen Texten (**mark_text** oder **mark_text_abs**) enthalten. Im letzteren Fall kann der Zeichensatz bei Bedarf (vor dem Markieren mit **mark_serial**) mit **select_char_set** umgeschaltet werden, siehe auch Kapitel 6.5.2 "Zeichensätze und Zeichenfolgen", Seite 121. • Für unzulässige Werte für `Mode` wird **mark_serial** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Absolute Vektor-Befehle und Bogen-Befehle werden nach Aufruf mit **mark_serial** auch absolut ausgeführt. Wenn Seriennummernmarkierungen an unterschiedlichen Stellen im Bildfeld ausgeführt werden sollen, dürfen die entsprechenden Zeichenfolgen (oder Zeichen) nur relative [*]mark[*]-Befehle, Bogen-Befehle oder Sprung-Befehle enthalten oder es ist **mark_serial_abs** zu verwenden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_serial**, **set_serial_step**, **set_serial_step_list**, **get_list_serial**, **set_max_counts**, **get_counts**, **load_text_table**, **set_text_table_pointer**, **mark_serial_abs**

Normaler Listenbefehl	**mark_serial_abs**
Funktion	Markiert die aktuelle Seriennummer des zuletzt mit **select_serial_set_list** ausgewählten Seriennummernsatzes (oder nach **load_program_file** des Seriennummernsatzes 0) im ausgewählten Format an der aktuellen Position. In den dabei ausgeführten indizierten Zeichenfolgen oder Zeichen (s.u.) werden absolute Vektor-Befehle und Bogen-Befehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt. Die Seriennummer wird danach (optional) automatisch erhöht.
Aufruf	`mark_serial_abs( Mode, Digits )`
Parameter	`Mode` Siehe **mark_serial**.
	`Digits` Siehe **mark_serial**.
Multi-Board-Befehlsname	**n_mark_serial_abs**
Hinweise	• **mark_serial_abs** wirkt wie **mark_serial**; der interne Aufruf der indizierten Zeichenfolgen (oder Zeichen) erfolgt allerdings mit **list_call_abs** statt mit **list_call**. • Enthalten die Befehlslisten der aufgerufenen indizierten Zeichenfolgen (oder Zeichen) keine absoluten Befehle, besteht zwischen **mark_serial_abs** und **mark_serial** kein Unterschied.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_serial**

Variabler Listenbefehl	**mark_text**
Funktion	Markiert einen nullterminierten String.
Aufruf	`mark_text( Text )`
Parameter	`Text` PC-Speicheradresse des ersten Zeichens (Byte) des zu markierenden Textes. Als Zeiger auf einen nullterminierten String.
Multi-Board-Befehlsname	**n_mark_text**
Hinweise	• Der zu markierende Text (Zeichenfolge, Byte-Array, nullterminierter String) muss mit einem \0-Zeichen (0-Byte, NUL) enden. Das 0\-Zeichen selbst wird nicht markiert. • Beim Laden eines **mark_text** wird der zu markierende Text, wenn er aus mehr als 12 Zeichen besteht (\0 nicht mitgezählt), in Blöcke zu je 12 Zeichen zerteilt und für jeden Block ein separater **mark_text**-Befehl in den RTC6-Listenspeicher geschrieben (achten Sie darauf, dass es dabei zu keinem unerwünschten Speicherüberlauf des jeweiligen RTC6-Listenspeicher-Bereichs kommt). Beim Abarbeiten der einzelnen **mark_text**-Befehle werden dann entsprechend des gewählten Zeichensatzes die zugehörigen **mark_char**-Befehle (indizierte Zeichen) ausgeführt. • Der gewünschte Zeichensatz kann vor **mark_text** mit **select_char_set** ausgewählt werden. In der Voreinstellung wird der Zeichensatz 0 verwendet. Wenn innerhalb eines aufgerufenen indizierten Zeichens **select_char_set** verwendet wird, werden alle nachfolgend aufgerufenen indizierten Zeichen mit diesem Zeichensatz markiert. • Wird beim Laden eines **mark_text** das Ende einer Liste (“Liste 1” oder “Liste 2”) erreicht, wird beim Anfang der jeweiligen Liste fortgefahren. Im geschützten Bereich (als Bestandteil eines indizierten Unterprogramms) dagegen wird das Laden abgebrochen (**get_last_error**-Returncode `RTC6_REJECTED`) und das indizierte Unterprogramm nicht gespeichert. • Absolute Vektor-Befehle und Bogen-Befehle werden nach Aufruf mit **mark_text** auch absolut ausgeführt. Wenn der Text an unterschiedlichen Stellen im Bildfeld markiert werden soll, dürfen die indizierten Zeichendefinitionen nur relative [*]mark[*]-Befehle, Bogen-Befehle oder Sprung-Befehle enthalten oder es ist **mark_text_abs** zu verwenden. • **mark_text** sollte nicht innerhalb einer indizierten Zeichendefinition verwendet werden. Der entsprechende Text würde *nicht* markiert und das indizierte Zeichen daher nicht vollständig ausgeführt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_text_abs**

Variabler Listenbefehl	**mark_text_abs**
Funktion	Markiert einen nullterminierten String. In den dabei ausgeführten indizierten Zeichen (s.u.) werden absolute Vektor-Befehle und Bogen-Befehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt.
Aufruf	`mark_text_abs( Text )`
Parameter	`Text` PC-Speicheradresse des ersten Zeichens (Byte) des zu markierenden Textes. Als Zeiger auf einen nullterminierten String.
Multi-Board-Befehlsname	**n_mark_text_abs**
Hinweise	• Beim Abarbeiten der einzelnen **mark_text_abs** werden entsprechend des gewählten Zeichensatzes die zugehörigen **mark_char_abs** ausgeführt. • Enthalten die Befehlslisten der verwendeten indizierten Zeichen keine absoluten Befehle, besteht zwischen **mark_text_abs** und **mark_text** kein Unterschied.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_text**

Normaler Listenbefehl	**mark_time**
Funktion	Markiert einen Teil der zuvor mit **time_fix**, **time_fix_f** oder **time_fix_f_off** gespeicherten Uhrzeit im ausgewählten Format an der aktuellen Position.
Aufruf	`mark_time( Part, Mode )`
Parameter	`Part` Gibt an, welcher Teil der Uhrzeit markiert wird. = 0: Stunden (24 h-Uhrzeit, kein a.m./p.m.) = 1: Minuten = 2: Sekunden = 3: Stunden (12 h-Uhrzeit, kein a.m./p.m.) = 4: a.m./p.m. (automatische Auswahl gemäß 24 h-Uhrzeit) Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…4]. `Mode` Auswahl des Formats. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…3], betrifft nur `Part` = 0…3. a) Anzahl der führenden Nullstellen: Bit #0 = 0: Führende Nullstellen werden unterdrückt. Bit #0 = 1: Immer zweistellig. b) Auswahl des Schriftsatzes für die Ziffernmarkierung: Bit #1 = 0: Es werden die mit **load_text_table** oder **set_text_table_pointer** für die Ziffern [0…9] definierten indizierten Zeichenfolgen markiert. Bit #1 = 1: Es werden die mit **load_char** oder **set_char_pointer** für die Ziffern [0…9] definierten indizierten Zeichen markiert. Der gewünschte Zeichensatz kann mit **select_char_set** angegeben werden. a.m./p.m. (Part = 4) können nur gemäß den mit **load_text_table** oder **set_text_table_pointer** definierten indizierten Zeichenfolgen (Index = 40, 41) markiert werden. Der Parameter Mode wird hier ignoriert.
Multi-Board-Befehlsname	**n_mark_time**
Hinweise	• Vor einer Uhrzeitmarkierung sollte (nach jedem Neustart) ein Abgleich zwischen RTC6 und PC-Zeit stattgefunden haben (siehe **time_update**) und die aktuelle (zu markierende) Uhrzeit mit **time_fix**, **time_fix_f** oder **time_fix_f_off** gespeichert worden sein, siehe Kapitel 7.5 "Uhrzeit, Datum und Seriennummern markieren", Seite 222. • Die vollständige Uhrzeit wird durch mehrfachen Aufruf von **mark_time** markiert.

Normaler Listenbefehl	**mark_time**
Hinweise (Forts.)	• **mark_time** liest entsprechend der gespeicherten Uhrzeit und des ausgewählten Uhrzeitteils die Startadressen der zugehörigen indizierten Zeichenfolgen (oder – für `Part` = 0…3 und `Mode` = 2 oder 3 – der indizierten Zeichen) aus der internen Verwaltungstabelle und ruft dann den Befehl **list_call** entsprechend oft auf (siehe auch Hinweise dort), der dann die entsprechenden Befehlslisten startet. Die Befehlslisten müssen die Markierungsvorschriften für die Ziffern 0…9 und für a.m./p.m. enthalten, siehe Abschnitt "Indizierte Zeichenfolgen für Uhrzeit, Datum und Seriennummer definieren", Seite 122. Nicht-definierte Zeichenfolgen oder Zeichen werden ignoriert (d.h. nicht markiert). Die aufgerufenen indizierten Zeichenfolgen können auch Aufrufe von indizierten Zeichen (**mark_char** oder **mark_char_abs**) und ganzen Texten (**mark_text** oder **mark_text_abs**) enthalten. Im letzteren Fall kann der Zeichensatz bei Bedarf (vor dem Markieren mit **mark_time**) mit **select_char_set** umgeschaltet werden, siehe auch Kapitel 6.5.2 "Zeichensätze und Zeichenfolgen", Seite 121. • Für `Part` > 4 und/oder `Mode` > 3 wird **mark_time** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Absolute Vektor-Befehle und Bogen-Befehle werden nach Aufruf mit **mark_time** auch absolut ausgeführt. Wenn Uhrzeitmarkierungen an unterschiedlichen Stellen im Bildfeld ausgeführt werden sollen, dürfen die entsprechenden Zeichenfolgen (oder Zeichen) nur relative [*]mark[*]-Befehle, Bogen-Befehle oder Sprung-Befehle enthalten oder es ist **mark_time_abs** zu verwenden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**time_fix**, **time_fix_f**, **time_fix_f_off**, **load_text_table**, **set_text_table_pointer**, **mark_time_abs**

Normaler Listenbefehl	**mark_time_abs**
Funktion	Markiert einen Teil der zuvor mit **time_fix**, **time_fix_f** oder **time_fix_f_off** gespeicherten Uhrzeit im ausgewählten Format an der aktuellen Position. In den dabei ausgeführten indizierten Zeichenfolgen oder Zeichen (s.u.) werden absolute Vektor-Befehle und Bogen-Befehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt.
Aufruf	`mark_time_abs( Part, Mode )`
Parameter	`Part` Siehe **mark_time**.
	`Mode` Siehe **mark_time**.
Multi-Board-Befehlsname	**n_mark_time_abs**
Hinweise	• **mark_time_abs** wirkt wie **mark_time**; der interne Aufruf der indizierten Zeichenfolgen (oder Zeichen) erfolgt allerdings mit **list_call_abs** statt mit **list_call**. • Enthalten die Befehlslisten der aufgerufenen indizierten Zeichenfolgen (oder Zeichen) keine absoluten Befehle, besteht zwischen **mark_time_abs** und **mark_time** kein Unterschied.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_time**

Kontrollbefehl	**master_slave_config**
Funktion	Setzt Einstellungen für die Master-Slave-Schnittstelle einer RTC6-Karte.
Aufruf	`master_slave_config( Config )`
Parameter	`Config` Als 32-Bit-Wert ohne Vorzeichen. Bit #0 Unterdrückung von /Slave-START-Signalen und /Slave-STOP-Signalen. = 0: /Slave-START- und /Slave-STOP-Signale über die Master-Slave-Schnittstelle werden empfangen und verarbeitet. = 1: /Slave-START- und /Slave-STOP-Signale, die diese Karte über die Master-Slave-Schnittstelle empfängt, werden auf dieser Karte ignoriert, wohl aber an weitere Karten der Master-Slave-Schnittstelle weitergeleitet. Bit #1 /STOP bei Master-Slave-Störungen. = 0: Die Karte reagiert nicht explizit auf eine Störung der Master-Slave-Verbindung. Etwaige Auswirkungen sind undefiniert. = 1: Löst ein /STOP aus, wenn die Verbindung zur Master-Karte unterbrochen wird. Dies tritt z.B. auf, wenn auf einer Karte der bereits synchronisierten Master-Slave-Kette **load_program_file** ausgeführt, das Master-Slave-Kabel entfernt oder das Signal der Master-Slave-Schnittstelle elektromagnetisch gestört wird. Bit #2 Weiterleitung von /STOP bei Master-Slave-Störungen. = 0: Ein /STOP wird nicht an andere Karten weitergeleitet. = 1: Wird ein Fehler der Master-Slave-Verbindung erkannt und daraufhin ein /STOP ausgelöst (siehe Bit #1) wird dieses /STOP an alle noch verbundenen Karten der Master-Slave-Kette weitergeleitet (auch aufwärts an eine Master-Karte).
Multi-Board-Befehlsname	**n_master_slave_config**
Hinweise	• Zur Verwendung von **master_slave_config**, siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127. • Die Master-Slave-Schnittstelle einer Karte sollte normalerweise erst nach erfolgreicher Synchronisierung konfiguriert werden. • Wenn Bit #0 schon vor **sync_slaves** gesetzt ist, kann diese Karte nicht mit **sync_slaves** auf eine Master-Karte synchronisiert werden. • In bestimmten weiteren Störungsszenarien kann es vorkommen, dass ein fehlerfreier Betrieb nicht mehr möglich ist. In diesem Fall wird immer ein /STOP auf der betroffenen Karte durchgeführt, auch wenn Bit #1 = 0 ist. • Wenn die Verbindung zu einer Master-Slave-synchronisierten Karte gestört ist, kann dies auch zur Folge haben, dass weitere Karten in der Kette ebenfalls gestört werden, und damit einen /STOP ausführen, auch wenn Bit #2 nicht gesetzt ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 613.
Verweise	**sync_slaves**

Kontrollbefehl	**master_slave_set_offset_ctrl**
Funktion	Stellt die Master/Slave Offset-Kompensation ein.
Aufruf	`master_slave_set_offset_ctrl( Mode, Ctrl )`
Parameter	`Mode` Modus für die Master/Slave Offset-Kompensation. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0 Deaktiviert. = 1 Aktiviert. > 1 Reserviert. **get_last_error**-Returncode `RTC6_PARAM_ERROR`. `Ctrl` Steuerparameter für die Master/Slave Offset-Kompensation. Als 32-Bit-Wert ohne Vorzeichen. `Mode` = 0 Wird ignoriert. `Mode` = 1 Wert für Offset-Kompensation. Einheit: [1/64 μs]. Zulässige Werte: [0…639]. Anderenfalls: **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
Rückgabe	Keine.
Multi-Board-Befehlsname	**n_master_slave_set_offset_ctrl**
Hinweise	• Siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	–

Kontrollbefehl	**mcbsp_init**
Funktion	Definiert die Daten-Delays für Senden und Empfangen von Daten über die McBSP-Schnittstelle, siehe auch Abschnitt "McBSP-Schnittstelle", Seite 87.
Aufruf	`mcbsp_init( XDelay, RDelay )`
Parameter	`XDelay` Sende-Delay. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…2].
	`RDelay` Empfangs-Delay. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…2].
Multi-Board-Befehlsname	**n_mcbsp_init**
Hinweise	• Bei unzulässigen Parameterangaben wird **mcbsp_init** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Die Initialisierungswerte (nach Programmstart) sind `XDelay` = `RDelay` = 1. • Die Signale und Betriebsbedingungen der McBSP-Schnittstelle sind in Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87 dargestellt. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Seite 89. • **mcbsp_init** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**read_mcbsp**, **set_mcbsp_out**, **set_mcbsp_out_ptr**, **set_mcbsp_freq**, **mcbsp_init_spi**

Kontrollbefehl	**mcbsp_init_spi**
Funktion	Ohne Funktion.
Aufruf	`mcbsp_init_spi( ClockLevel, ClockDelay )`
Parameter	`ClockLevel` = 0: inaktiv low. > 0: inaktiv high. Als 32-Bit-Wert ohne Vorzeichen. `ClockDelay` = 0: Taktsignal zeitgleich zu den Daten-Bits. > 0: Taktsignal um eine halbe Periode verzögert. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_mcbsp_init_spi**
Hinweise	• Siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Veränderte Funktionalität. • Im RTC6-Befehlssatz ist **mcbsp_init_spi** enthalten, jedoch nicht wirksam. Er wird mit **get_last_error**-Returncode `RTC6_REJECTED` verweigert und nicht ausgeführt. • Hinweise z.B. zur Migration von Quellcode von RTC5-Anwenderprogrammen: Bei RTC6-Karten kann Hardware-bedingt keine SPI-Funktionalität mehr eingestellt werden. Solche Quellcodeabschnitte müssen entsprechend angepasst werden.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mcbsp_init**, **set_mcbsp_freq**

Kontrollbefehl	**measurement_status**
Funktion	Gibt zurück: • Den Status einer mit **set_trigger[*]** gestarteten Messung • Die aktuelle Position des Messwertzählers
Aufruf	`measurement_status( &Busy, &Pos )`
Parameter-Rückgaben	`Busy` Status der Messung. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen. > 0: Momentan erfolgt eine Messung. = 0: Momentan erfolgt keine Messung.
	`Pos` Aktuelle Position des Messwertzählers (im Messwertspeicher der RTC6) [0…max. Kanalgröße, siehe **set_trigger[*]**]. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_measurement_status**
Hinweise	• Wenn die mit **set_trigger[*]** gestartete Messung nicht mehr aktiv ist, zeigt `Pos`+1 die Anzahl der bis zum Abbruch aufgezeichneten Datenpaare an ([0…max. Kanalgröße, siehe **set_trigger[*]**]). • `Pos` = 2^{32}–1 zeigt an, dass nach **load_program_file** noch keine Datenaufzeichnung stattgefunden hat. • Gespeicherte Daten können mit **get_waveform** ausgelesen werden. • Der Status der Messwertaufzeichnung wird zurückgesetzt mit (siehe Hinweis bei **set_trigger**): – **set_trigger**(`Period` = 0) – **set_trigger4**(`Period` = 0) – **set_trigger8**(`Period` = 0) – **stop_trigger** – **stop_execution**
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_trigger**, **set_trigger4**, **set_trigger8**

Normaler Listenbefehl	**micro_vector_abs**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus mit hartem Sprung (ohne Zerlegung in Mikroschritte; siehe "Harter Sprung") direkt von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`micro_vector_abs( X, Y, LasOn, LasOff )`
Parameter	`X` Absolute x-Koordinate des Mikrovektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`Y` Wie `X` (analog).
	`LasOn` LaserOn-Delay. 1 Bit entspricht 1/64 µs. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–1…+(2^{15}–1)]. ≥ 0: Delay wird neu gesetzt. Werte über (2^{15}–1) werden geclippt. < 0: Das zuvor eingestellte Delay läuft unverändert ab.
	`LasOff` LaserOff-Delay. 1 Bit entspricht 1/64 µs. Sonst wie `LasOn`.
Multi-Board-Befehlsname	**n_micro_vector_abs**
Hinweise	• Siehe auch Kapitel 8.8 "micro_vector[*]-Befehle", Seite 285. • Es wird kein Wobbel berücksichtigt, siehe 2 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189. • Der Mikrovektor wird in jedem Fall als "Harter Sprung" ausgeführt. • Mit `LasOn` ≥ 0 oder `LasOff` ≥ 0 kann für jeden einzelnen Mikrovektor ein neues LaserOn-Delay oder LaserOff-Delay gesetzt werden. Jedes Delay wird dabei jeweils am Ende desjenigen Taktes gesetzt, in dem die neue Position tatsächlich ausgegeben wird (dieser Ausgabe-Takt kann durch ein vorausgehendes Scanner-Delay verzögert werden). • Bei negativen Werten (`LasOn` < 0 oder `LasOff` < 0) bleiben die Laser-Delays unangetastet. Dadurch kann der Laser auch über mehrere Takte an- oder ausbleiben (Mark- und Jump-Simulation). • Die mit `LasOn` und `LasOff` gesetzten Delays gelten nur für die Ausführung von Mikrovektoren. Für die Ausführung von normalen Mark- oder Bogen-Befehlen (wie **mark_abs**) gelten ausschließlich die mit **set_laser_delays** festgelegten Laser-Delays. `LasOn` und `LasOff` überschreiben nicht die Laser-Delay-Parameter aus **set_laser_delays**.

Normaler Listenbefehl	**micro_vector_abs**
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16, die für `LasOn` und `LasOff` mit 64. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `LasOn` und `LasOff` mit 32. Entsprechend verringern sich die zulässigen Wertebereiche.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**micro_vector_rel**, **micro_vector_abs_3d**

Normaler Listenbefehl	**micro_vector_abs_3d**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus mit hartem Sprung (ohne Zerlegung in Mikroschritte; siehe "Harter Sprung") direkt von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **micro_vector_abs_3d** wie **micro_vector_abs**.
Aufruf	`micro_vector_abs_3d( X, Y, Z, LasOn, LasOff )`
Parameter	`X` Absolute x-Koordinate des Mikrovektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`Y` Wie `X` (analog).
	`Z` Wie `X` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287].
	`LasOn` LaserOn-Delay. 1 Bit entspricht 1/64 μs. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–1…+(2^{15}–1)]. ≥ 0: Delay wird neu gesetzt. Werte über (2^{15}–1) werden geclippt. < 0: Das zuvor eingestellte Delay läuft unverändert ab.
	`LasOff` LaserOff-Delay. Wie `LasOn`.
Multi-Board-Befehlsname	**n_micro_vector_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **micro_vector_abs_3d** gleich wie **micro_vector_abs** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X`, `Y` und `Z` mit 16, die für `LasOn` und `LasOff` mit 64. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Z` mit 16, die für `LasOn` und `LasOff` mit 32. Entsprechend verringern sich die zulässigen Wertebereiche.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**micro_vector_abs**, **micro_vector_rel_3d**

Normaler Listenbefehl	**micro_vector_rel**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus mit hartem Sprung (ohne Zerlegung in Mikroschritte; siehe "Harter Sprung") direkt von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`micro_vector_rel( dX, dY, LasOn, LasOff )`
Parameter	`dX` Relative x-Koordinate des Mikrovektor-Endpunkts. In Bits. Sonst wie `X` von **micro_vector_abs**.
	`dY` Relative y-Koordinate des Mikrovektor-Endpunkts. In Bits. Sonst wie `Y` von **micro_vector_abs**.
	`LasOn` Wie `LasOn` von **micro_vector_abs**.
	`LasOff` Wie `LasOn` von **micro_vector_abs**.
Multi-Board-Befehlsname	**n_micro_vector_rel**
Hinweise	• Die Koordinaten des Mikrovektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **micro_vector_rel** identisch mit **micro_vector_abs** (siehe Hinweise dort). • Der Mikrovektor wird in jedem Fall als "Harter Sprung" ausgeführt.
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **micro_vector_abs**.
RTC5→RTC6	Unveränderte Funktionalität. RTC5-Kompatibilitätsmodus: siehe **micro_vector_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**micro_vector_abs**, **micro_vector_rel_3d**

Normaler Listenbefehl	**micro_vector_rel_3d**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus mit hartem Sprung (ohne Zerlegung in Mikroschritte; siehe "Harter Sprung") direkt von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **micro_vector_rel_3d** wie **micro_vector_rel**.
Aufruf	`micro_vector_rel_3d( dX, dY, dZ, LasOn, LasOff )`
Parameter	`dX` Relative x-Koordinate des Mikrovektor-Endpunkts. In Bits. Sonst wie `X` von **micro_vector_abs_3d**.
	`dY` Relative y-Koordinate des Mikrovektor-Endpunkts. In Bits. Sonst wie `Y` von **micro_vector_abs_3d**.
	`dZ` Relative z-Koordinate des Mikrovektor-Endpunkts. In Bits. Sonst wie `Z` von **micro_vector_abs_3d**.
	`LasOn` Wie `LasOn` von **micro_vector_abs_3d**.
	`LasOff` Wie `LasOff` von **micro_vector_abs_3d**.
Multi-Board-Befehlsname	**n_micro_vector_rel_3d**
Hinweise	• Die Koordinaten des Mikrovektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **micro_vector_rel_3d** identisch mit **micro_vector_abs_3d** (siehe Hinweise dort). • Der Mikrovektor wird in jedem Fall als "Harter Sprung" ausgeführt.
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **micro_vector_abs_3d**.
RTC5→RTC6	Unveränderte Funktionalität. RTC5-Kompatibilitätsmodus: siehe **micro_vector_abs_3d**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**micro_vector_abs_3d**, **micro_vector_rel**

Kontrollbefehl	**move_to**
Funktion	Bewegt die Fokussieroptik eines vario*SCAN* 40 FLEX an die angegebene Position.
Einschränkung	• **move_to** wurde für die Erweiterungskarte “RTC STEP MOTOR EXTENSION” (#0112097) eingeführt. • **move_to** wird von der Erweiterungskarte “RTC5/6 varioSCAN 40 FLEX Extension” (#0128683) *nicht* unterstützt. Siehe auch Kapitel 2.8.7 “Erweiterungskarte “RTC5/6 varioSCAN FLEX Extension””, Seite 47.
Aufruf	`move_to( Pos )`
Parameter	`Pos` Positionswert. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_move_to**
Hinweise	• Die Verwendung von **move_to** ist nur sinnvoll, wenn als z-Achse ein vario*SCAN* 40 FLEX an eine Erweiterungskarte “RTC STEP MOTOR EXTENSION” (#0112097) angeschlossen ist, die an der EXTENSION 1-Stiftleiste installiert ist. Siehe Handbuch “Installation and Operation RTC Step Motor Extension for the RTC4 and RTC5 PC interface boards” (nur in Englisch verfügbar). Es ist nicht möglich, automatisch oder über RTC6-Befehle festzustellen, ob die Erweiterungskarte “RTC STEP MOTOR EXTENSION” (#0112097) und vario*SCAN* 40 FLEX an die RTC6-Karte angeschlossen sind. • Siehe vario*SCAN* 40 FLEX-Handbuch zu: – Grobe Positionswerte für verschiedene Arbeitsabstände – Zulässiger Wertebereich für den Positionswert • Beim ersten Aufruf von `move_to( Pos = 0 )` nach **load_program_file** führt der vario*SCAN* 40 FLEX-Schrittmotor eine Referenzfahrt aus, um seine Referenzposition zu finden. Dabei sucht der vario*SCAN* 40 FLEX den linken Endanschlag und positioniert den Schrittmotor direkt daneben. Diese Position dient dann als Nullpunkt. Tritt während der Referenzfahrt ein Fehler auf, so wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt und die Positionierung auf `Pos` > 0 wird nicht durchgeführt. • Für `Pos` > 0 fährt vario*SCAN* 40 FLEX die angegebene Position an. Tritt ein Fehler auf (z.B. weil der Endanschlag erreicht wurde), dann wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. In diesem Fall sollten Sie: **(1)** Den vario*SCAN* 40 FLEX (Anschlüsse, Endanschläge, Schrittmotor usw.) auf einwandfreie Funktion überprüfen **(2)** `move_to( 0 )` für eine weitere Referenzfahrt aufrufen **(3)** `move_to( Pos )` aufrufen

Kontrollbefehl	move_to
Hinweise (Forts.)	• Wenn die Erweiterungskarte “RTC STEP MOTOR EXTENSION” (#0112097) an die EXTENSION 1-Stiftleiste angeschlossen ist, ist nur eine Teilmenge der entsprechenden Signale nutzbar: – Bit #12…Bit #15 des 16-Bit-Digital-Ausgangs werden zum Steuern des vario*SCAN* 40 FLEX verwendet. Um sicherzustellen, dass diese 4 Bits nur durch **move_to** gesetzt werden (und nicht fälschlicherweise durch I/O-Befehle, siehe unten), sind sie ab dem ersten **move_to**-Aufruf ausschließlich für die vario*SCAN* 40 FLEX-Steuerung reserviert. Diese Exklusivität bleibt bis zum nächsten **load_program_file**-Aufruf erhalten. Vor der Ausführung des ersten **move_to**-Aufrufs sind die 4 Bits noch verfügbar, haben aber keinen Einfluss auf die vario*SCAN* 40 FLEX-Positionierung. Sie sollten nicht verwendet werden, solange die Erweiterungskarte “RTC STEP MOTOR EXTENSION” (#0112097) oder ein vario*SCAN* 40 FLEX angeschlossen sind. – Auch bei Verwendung von **move_to** können Sie weiterhin die unteren (aber nicht die oberen) 8 Bits des 16-Bit-Digital-Ausgangs (ohne LATCH-Synchronisierung) für andere Zwecke mit **write_io_port_list**, **write_io_port**, **write_io_port_mask_list**, **write_io_port_mask**, **set_io_cond_list**, **clear_io_cond_list** oder **stop_execution** ändern. – Ebenfalls verfügbar (über die Erweiterungskarte “RTC STEP MOTOR EXTENSION” (#0112097)) sind die unteren (aber nicht die oberen) 8 Bits des 16-Bit-Digital-Eingangs (ohne SYNC OUT) und das BUSY OUT-Signal.
RTC4→RTC6	Unveränderte Funktionalität. Im Gegensatz zum RTC4-Befehl ist kein separater **set_control_mode**-Aufruf mehr erforderlich, um Bits des 16-Bit-Digital-Ausgangs zum Steuern des vario*SCAN* 40 FLEX zu reservieren. Ein explizites Freigeben ist nicht möglich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_io_port_list**, **write_io_port**, **set_io_cond_list**, **clear_io_cond_list**, **load_program_file**, **get_last_error**, **write_io_port_mask_list**, **write_io_port_mask**, **stop_execution**

Kontrollbefehl	**number_of_correction_tables**
Funktion	Definiert die maximale Anzahl von zulässigen Korrekturtabellen.
Aufruf	`number_of_correction_tables( Number )`
Parameter	`Number` Anzahl der maximal zulässigen Korrekturtabellen. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…8]. Default nach **load_program_file**: 8. Siehe auch Kapitel 8.5.3 "Mehrere Korrekturtabellen verwenden", Seite 253.
Multi-Board-Befehlsname	**n_number_of_correction_tables**
Hinweise	• Für außerhalb des zulässigen Wertebereichs wird **number_of_correction_tables** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • **number_of_correction_tables** dient dazu, andere Befehle (wie z.B. **load_correction_file** und **select_cor_table**) vor unerwünschten Tabellennummern zu schützen. • Bestehende Anwenderprogramme müssen nicht geändert werden. Die Ausnahme ist, wenn künftig Benutzer-Eingaben (mittels expliziter RTC6-Fehlermeldungen) zurückgewiesen werden sollen. • **number_of_correction_tables** bezieht sich nur auf nachfolgende Befehlsausführungen. Bereits bestehende Zuweisungen von Korrekturtabellen können nicht automatisch korrigiert werden. Das ist insbesondere dann zu beachten, wenn RTC6-Karten nachträglich in Besitz genommen werden, die von anderen Anwenderprogrammen initialisiert wurden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Geänderte Auswirkungen (der RTC5-Befehl wurde aufgrund des begrenzten Benutzerkreises nicht dokumentiert).
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 613.
Verweise	**load_correction_file**, **select_cor_table**, **select_cor_table_list**

Normaler Listenbefehl	**para_jump_abs**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus mit Sprunggeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_jump_abs( X, Y, P )`
Parameter	`X` Absolute x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. `Y` Wie `X` (analog). `P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_jump_abs**
Hinweise	• Wenn nicht zuvor mit **set_vector_control** die Vektorkontrollierte Laser-Steuerung aktiviert wurde, wirkt **para_jump_abs** wie **jump_abs** (siehe Hinweise dort). Der Parameter `P` wird dann ignoriert. • Bei aktivierter Vektorkontrollierte Laser-Steuerung wird parallel zur Bewegung des Ausgabepunkts für den Laserfokus der mit **set_vector_control** ausgewählte Signalparameter linear vom zuletzt gültigen Wert zum Wert `P` variiert, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217. • Zu `P` gibt es keinen abs-Mechanismus. `P` wird auf den zulässigen Maximalwert geclippt. • Wird **para_jump_abs** gleichzeitig mit einer Positions-abhängige Laser-Steuerung oder Geschwindigkeits-abhängige Laser-Steuerung mit demselben Kontrollparameter verwendet, so wird der aktuelle Wert von `P` als Basis für den 100%-Wert der Laser-Steuerung verwendet, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche. Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC6-Einheiten verwendet werden. Ausnahme ist `Ctrl` = 7 (Defocus): Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 bei `Ctrl` = 7 den für `Value` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Ausnahme ist `Ctrl` = 7 (Defocus): Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 bei `Ctrl` = 7 den für `Value` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_abs**, **set_vector_control**, **para_jump_abs_3d**, **para_jump_rel**

Normaler Listenbefehl	**para_jump_abs_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **para_jump_abs_3d** wie **para_jump_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`para_jump_abs_3d( X, Y, Z, P )`
Parameter	`X` Absolute x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`Y` Wie `X` (analog).
	`Z` Wie `X` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287].
	`P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_jump_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **para_jump_abs_3d** gleich wie **para_jump_abs** (siehe Hinweise dort). • Weitere Hinweise siehe **jump_abs_3d**.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X`, `Y` und `Z` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche. Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC6-Einheiten verwendet werden. Ausnahme ist `Ctrl` = 7 (Defocus): Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 bei `Ctrl` = 7 den für `Value` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Ausnahme ist `Ctrl` = 7 (Defocus): Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 bei `Ctrl` = 7 den für `Value` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_abs**, **set_vector_control**, **para_jump_abs**, **para_jump_rel_3d**

Normaler Listenbefehl	**para_jump_rel**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_jump_rel( dX, dY, P )`
Parameter	`dX` Relative x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. `dY` Wie `dX` (analog). `P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_jump_rel**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **para_jump_rel** identisch mit **para_jump_abs** (siehe Hinweise dort). • `P` wird nicht relativ behandelt.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich. Für den Parameter `P` siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Für den Parameter `P` siehe **para_jump_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_rel**, **set_vector_control**, **para_jump_abs**, **para_jump_rel_3d**

Normaler Listenbefehl	**para_jump_rel_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) mit Sprunggeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **para_jump_rel_3d** wie **para_jump_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`para_jump_rel_3d( dX, dY, dZ, P )`
Parameter	`dX` Relative x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar.
	`dY` Wie `dX` (analog).
	`dZ` Wie `dX` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287].
	`P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_jump_rel_3d**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **para_jump_rel_3d** identisch mit **para_jump_abs_3d** (siehe Hinweise dort). • `P` wird nicht relativ behandelt.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich. Für den Parameter `P` siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Für den Parameter `P` siehe **para_jump_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_rel_3d**, **set_vector_control**, **para_jump_abs_3d**, **para_jump_rel**

Variabler Listenbefehl	**para_laser_on_pulses_list**
Funktion	Schaltet das LASERON-Signal für die angegebene Anzahl von externen Signalpulsen, jedoch maximal für das angegebene Zeitintervall ein. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_laser_on_pulses_list( Period, Pulses, P )`
Parameter	`Period` Zeitintervall. In Bits. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 μs. Zulässiger Wertebereich: $0 \le$ `Period` $\le (2^{32}-1)$. `Pulses` Anzahl externer Signalpulse. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: $0 \le$ `Pulses` ≤ 65.535 oder größer (siehe Hinweis unten). `P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_laser_on_pulses_list**
Hinweise	• **para_laser_on_pulses_list** ist nützlich für das Markieren separater Punkte, siehe Kapitel 7.1.3 "Einzelne Punkte markieren", Seite 144. • Wenn nicht zuvor mit **set_vector_control** die Vektorkontrollierte Laser-Steuerung aktiviert wurde, wirkt **para_laser_on_pulses_list** wie **laser_on_pulses_list**, d.h. für `Pulses` > 65.535 wie **laser_on_list** (siehe Hinweise dort). Der Parameter `P` wird dann ignoriert. • Bei aktivierter Vektorkontrollierte Laser-Steuerung wird der mit **set_vector_control** ausgewählte Signalparameter innerhalb der Ausführdauer von **para_laser_on_pulses_list** (`Period` × 10 μs) linear vom zuletzt gültigen Wert zum Wert `P` variiert, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217. • Zu `P` gibt es keinen abs-Mechanismus. `P` wird auf den zulässigen Maximalwert geclippt. Dieser Maximalwert ist $(2^{31}-1)$ oder abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`) auch ein niedrigerer Wert. • Wird **para_laser_on_pulses_list** gleichzeitig mit einer Positions-abhängige Laser-Steuerung oder Geschwindigkeits-abhängige Laser-Steuerung mit demselben Kontrollparameter verwendet, so wird der aktuelle Wert von `P` als Basis für den 100%-Wert der Laser-Steuerung verwendet, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217. • Für `Period` = 0 hat **para_laser_on_pulses_list** keine Wirkung. **para_laser_on_pulses_list** ist dann ein Kurzer Listenbefehl. • Für $0 <$ `Period` $\le (2^{31}-1)$ dauert **para_laser_on_pulses_list** immer genau `Period` Takte (`Period` × 10 μs), auch wenn die angegebene Anzahl von externen Signalpulsen in kürzerer Zeit abläuft. • Für $2^{31} \le$ `Period` $\le (2^{32}-1)$ dauert **para_laser_on_pulses_list** maximal (`Period` $- 2^{31}$) Takte (d.h. (`Period` $- 2^{31}$) × 10 μs). Hier wird **para_laser_on_pulses_list** jedoch vorzeitig abgebrochen, sobald die angegebene Anzahl von externen Signalpulsen detektiert wurde.

Variabler Listenbefehl	**para_laser_on_pulses_list**
RTC4→RTC6	Neuer Befehl. Für den Parameter `P` siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Für den Parameter `P` siehe **para_jump_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**laser_on_pulses_list**, **laser_on_list**

Normaler Listenbefehl	**para_mark_abs**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_mark_abs( X, Y, P )`
Parameter	`X` Absolute x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. `Y` Wie `X` (analog). `P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_mark_abs**
Hinweise	• Wenn nicht zuvor mit **set_vector_control** die Vektorkontrollierte Laser-Steuerung aktiviert wurde, wirkt **para_mark_abs** wie **mark_abs** (siehe Hinweise dort). Der Parameter `P` wird dann ignoriert. • Bei aktivierter Vektorkontrollierte Laser-Steuerung wird parallel zur Bewegung des Laserfokus der mit **set_vector_control** ausgewählte Signalparameter linear vom zuletzt gültigen Wert zum Wert `P` variiert, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217. • Zu `P` gibt es keinen abs-Mechanismus. `P` wird auf den zulässigen Maximalwert geclippt. • ≤ OUT 618: Für Markier-Vektorlängen = 0, dürfen keine Änderung des Signalparameters `P` programmiert sein: – Diese Änderung wird nicht ausgegeben – Der nachfolgende [*]para[*]-Befehl (nur dieser) erzeugt fehlerhafte Signalparameter-Ausgaben • ≥ OUT 619: Für Markier-Vektorlängen = 0, bei denen eine Änderung des Signalparameters `P` programmiert ist, gilt: – Diese werden als zeitbemessene Vektoren mit `T` = 10 µs behandelt – Der Endwert des Signalparameters `P` wird ausgegeben • Wird **para_mark_abs** gleichzeitig mit einer Positions-abhängige Laser-Steuerung oder Geschwindigkeits-abhängige Laser-Steuerung mit demselben Kontrollparameter verwendet, so wird der aktuelle Wert von `P` als Basis für den 100%-Wert der Laser-Steuerung verwendet, siehe Abschnitt "Vektorkontrollierte Laser-Steuerung", Seite 217. • **[*]para_mark[*]**-Befehle berücksichtigen grundsätzlich kein Sky-Writing.

Normaler Listenbefehl	**para_mark_abs**
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich. Für den Parameter `P` siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Für den Parameter `P` siehe **para_jump_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung OUT 619: Behandlung von Markier-Vektorlängen = 0.
Verweise	**mark_abs**, **set_vector_control**, **para_mark_abs_3d**, **para_mark_rel**

Normaler Listenbefehl	**para_mark_abs_3d**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **para_mark_abs_3d** wie **para_mark_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`para_mark_abs_3d( X, Y, Z, P )`
Parameter	`X` Absolute x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. `Y` Wie `X` (analog). `Z` Wie `X` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287]. `P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_mark_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **para_mark_abs_3d** gleich wie **para_mark_abs** (siehe Hinweise dort). • Weitere Hinweise siehe **mark_abs_3d**. • **[*]para_mark[*]**-Befehle berücksichtigen grundsätzlich kein Sky-Writing.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich. Für den Parameter `P` siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Für den Parameter `P` siehe **para_jump_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_abs_3d**, **set_vector_control**, **para_mark_abs**, **para_mark_rel_3d**

Normaler Listenbefehl	**para_mark_rel**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_mark_rel( dX, dY, P )`
Parameter	`dX` Relative x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. `dY` Wie `dX` (analog). `P` Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_mark_rel**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **para_mark_rel** identisch mit **para_mark_abs** (siehe Hinweise dort). • `P` wird nicht relativ behandelt. • **[*]para_mark[*]**-Befehle berücksichtigen grundsätzlich kein Sky-Writing.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.Für den Parameter `P` siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Für den Parameter `P` siehe **para_jump_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_rel**, **set_vector_control**, **para_mark_abs**, **para_mark_rel_3d**

Normaler Listenbefehl	**para_mark_rel_3d**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **para_mark_rel_3d** wie **para_mark_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`para_mark_rel_3d( dX, dY, dZ, P )`
Parameter	`dX` — Relative x-Koordinate des Markvektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt. Der volle Wertebereich ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. `dY` — Wie `X` (analog). `dZ` — Wie `X` (analog), außer Zulässiger Wertebereich: [–524.288…+524.287]. `P` — Endwert des Signalparameters. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch zu **set_vector_control** (Parameter `Value`).
Multi-Board-Befehlsname	**n_para_mark_rel_3d**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **para_mark_rel_3d** identisch mit **para_mark_abs_3d** (siehe Hinweise dort). • `P` wird nicht relativ behandelt. • **[*]para_mark[*]**-Befehle berücksichtigen grundsätzlich kein Sky-Writing.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringert sich der zulässige Wertebereich. Für den Parameter `P` siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Für den Parameter `P` siehe **para_jump_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_rel_3d**, **set_vector_control**, **para_mark_abs_3d**, **para_mark_rel**

Variabler Listenbefehl	**park_position**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) zum Zwischenparken mit Sprunggeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, wirkt **park_position** wie ein normaler **jump_abs**.
Aufruf	`park_position( Mode, X, Y )`
Parameter	`Mode` Als 32-Bit-Wert ohne Vorzeichen. = 0: `X` und `Y` werden als Koordinaten einer Parkposition im realen Bildfeld interpretiert. Zulässiger Wertebereich: [–524.288…+524.287]. Der aktuelle "Processing on the fly"-Modus wird abgeschaltet und der Laserfokus mit der aktuell eingestellten Sprunggeschwindigkeit in die angegebene Parkposition im realen Bildfeld bewegt. Während einer nachfolgenden Listenunterbrechung (mit **wait_for_encoder** etc.) bleiben die Galvanometerscanner stehen (auch bei einer mit **set_fly_2d** eingeleiteten "Processing on the fly"-Anwendung). > 0: `X` und `Y` werden als Koordinaten einer Parkposition im virtuellen Bildfeld interpretiert. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Der aktuelle "Processing on the fly"-Modus bleibt eingeschaltet und der Laserfokus wird mit der aktuell eingestellten Sprunggeschwindigkeit in die angegebene Parkposition im virtuellen Bildfeld bewegt (korrigiert um die "Processing on the fly"-Korrektur und geclippt auf den Randbereich des realen Bildfelds). Während einer nachfolgenden Listenunterbrechung (mit **wait_for_encoder** etc.) werden die Positionen der Galvanometerscanner kontinuierlich entsprechend der aktuellen Encoderwerte "Processing on the fly"-korrigiert (auch bei einer mit **set_fly_x**/**set_fly_y** eingeleiteten "Processing on the fly"-Anwendung), geclippt auf den Randbereich des realen Bildfelds. `X` Absolute x-Koordinate des Sprung-Vektor-Endpunkts. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: siehe oben. Überlaufende Werte werden auf die Randwerte geclippt. `Y` Wie `X` (analog).
Multi-Board-Befehlsname	**n_park_position**

Variabler Listenbefehl	park_position
Hinweise	• **park_position** ist dafür vorgesehen, den Laserfokus innerhalb einer encoderbasierten **set_fly_2d**- oder **set_fly_x**/**set_fly_y**-"Processing on the fly"-Anwendung während einer zwischengeschalteten Bewegungsfahrt (vor einer Listenunterbrechung mit **wait_for_encoder** etc.) in einem sicheren Bereich zu parken, siehe Kapitel 8.6.7 ""Processing on the fly"-Anwendungen synchronisieren", Seite 264. Für den Rücksprung (zum Verlassen der Parkposition) sollte **park_return** verwendet werden (siehe auch alle Hinweise dort). • Wenn keine oder wenn eine andere "Processing on the fly"-Anwendung aktiviert ist, dann wirkt **park_position** wie ein normaler Sprung-Befehl und der Rücksprung mit **park_return** ebenso (die "Processing on the fly"-Korrektur bleibt jedoch ausgeschaltet). • Wurde der Laserfokus zuletzt schon in eine sichere Parkposition bewegt, ist **park_position** ein Kurzer Listenbefehl und bewirkt nichts weiter. • **park_position** schaltet die Signale für den "Laser active"-Betrieb nach einem LaserOff-Delay ab, aktiviert danach aber kein Scanner-Delay.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**park_return**

Variabler Listenbefehl	**park_position_1_axis**
Funktion	"Fly-Erweiterung"-Befehl: Bewegt die Ausgabeposition für 1 Achse zum Zwischenparken mit Sprunggeschwindigkeit von der aktuellen zur angegebenen absoluten Position im 2D-Bildfeld.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, führt **park_position_1_axis** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`park_position_1_axis( Mode, Axis, ParkPos )`
Parameter	`Mode` = 0: Zwischenparkposition. Absolute Koordinate im realen Bildfeld. "Processing on the fly" wird bei der angegebenen Achse ausgeschaltet. > 0: Zwischenparkposition. Absolute Koordinate im virtuellen Bildfeld. "Processing on the fly" bleibt bei der angegebenen Achse eingeschaltet. Als 32-Bit-Wert ohne Vorzeichen. `Axis` Achse aus Tabelle 3, Seite 271. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: `1`...`2`. `ParkPos` Absolute Achsen-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_park_position_1_axis**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **park_position_1_axis** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • Siehe auch Hinweise zu **park_position**. • Zum Zurückspringen kann **park_return_1_axis** verwendet werden. • Bei einem unzulässigen Parameter-Wert wird **park_position_1_axis** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**park_position_2_axes**

Variabler Listenbefehl	**park_position_2_axes**
Funktion	"Fly-Erweiterung"-Befehl: Bewegt die Ausgabepositionen für 2 Achsen zum Zwischenparken mit Sprunggeschwindigkeit von der aktuellen zur angegebenen absoluten Position im 2D-Bildfeld.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, führt **park_position_2_axes** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`park_position_2_axes( Mode, ParkPosX, ParkPosY )`
Parameter	`Mode` = 0: Zwischenparkposition. Absolute Koordinate im realen Bildfeld. "Processing on the fly" wird bei den angegebenen Achsen ausgeschaltet. > 0: Zwischenparkposition. Absolute Koordinate im virtuellen Bildfeld. "Processing on the fly" bleibt bei den angegebenen Achsen eingeschaltet. Als 32-Bit-Wert ohne Vorzeichen. `ParkPosX` Absolute Achsen-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Überlaufende Werte werden auf die Randwerte geclippt. `ParkPosY` Wie `ParkPosX`.
Multi-Board-Befehlsname	**n_park_position_2_axes**

Hinweise	• Als "Fly-Erweiterung"-Befehl darf **park_position_2_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden.
	• Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271.
	• Siehe auch Hinweise zu **park_position**.
	• Zum Zurückspringen kann **park_return_2_axes** verwendet werden.
	• Bei einem unzulässigen Parameter-Wert wird **park_position_2_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	• Folgende Befehlsaufrufe werden gleichartig ausgeführt:
	– `park_position_2_axes( Mode, ParkPosX, ParkPosY ) =`
	`{`
	`park_position_1_axis( Mode, 1, ParkPosX );`
	`park_position_1_axis( Mode, 2, ParkPosY );`
	`}`

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**park_position_1_axis**

Variabler Listenbefehl	**park_return**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) aus einer Parkposition heraus mit Sprunggeschwindigkeit entlang eines 2D-Vektors zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, ist **park_return** ein normaler Sprung-Befehl. Falls sich der Laserfokus aktuell nicht in einer Parkposition befindet, ist **park_return** ein Kurzer Listenbefehl und bewirkt nichts weiter (s.u.).
Aufruf	`park_return( Mode, X, Y )`
Parameter	`Mode` Als 32-Bit-Wert ohne Vorzeichen. = 0: `X` und `Y` werden ignoriert (siehe Hinweise). > 0: `X` und `Y` werden als Koordinaten für den Rücksprung interpretiert (siehe Hinweise).
	`X` Absolute x-Koordinate des Sprung-Vektor-Endpunkts im virtuellen Bildfeld. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Y` Wie `X` (analog).
Multi-Board-Befehlsname	**n_park_return**
Hinweise	• **park_return** ist dafür vorgesehen, den Laserfokus innerhalb einer encoderbasierten **set_fly_2d**- oder **set_fly_x**/**set_fly_y**-"Processing on the fly"-Anwendung nach einer zwischengeschalteten Bewegungsfahrt (nach einer Listenunterbrechung mit **wait_for_encoder** etc.) wieder aus einem sicheren Bereich herauszubewegen, in den er zuvor mittels **park_position** bewegt worden ist (siehe Kapitel 8.6.7 ""Processing on the fly"-Anwendungen synchronisieren", Seite 264, siehe auch Hinweise zu **park_position**). • Wurde mit einem vorausgehenden **park_position**(`Mode` = 0) ein **set_fly_2d**- oder **set_fly_x**/**set_fly_y**-"Processing on the fly"-Mode abgeschaltet, schaltet **park_return** denselben "Processing on the fly"-Mode wieder ein. Andere "Processing on the fly"-Modi werden nicht wieder eingeschaltet. • Wurde die Parkposition mit **park_position** eingenommen, bewegt **park_return**(`Mode` = 0) die Galvanometerscanner an die zuletzt gültige Position vor Aufruf von **park_position** zurück, **park_return**(`Mode` = 1) dagegen an die mit angegebene Stelle. Die RTC6 berücksichtigt bei der Berechnung der neuen Position gegebenenfalls auch die dann aktuelle "Processing on the fly"-Korrektur. Kommt es dabei allerdings zu einem Clippen auf den Randbereich des virtuellen Bildfelds (z.B. wegen einer zu langen Bewegungsfahrt während einer Listenunterbrechung mit **wait_for_encoder**), so wird der "Processing on the fly"-Mode *nicht* wieder aktiviert (siehe auch **activate_fly_2d** und **activate_fly_xy**) und der Sprung wird ohne "Processing on the fly"-Korrektur ausgeführt.

Variabler Listenbefehl	park_return
Hinweise (Forts.)	• Wurde der Laserfokus *nicht* zuvor mit **park_position** in einen sicheren Bereich bewegt, ist **park_return** ein Kurzer Listenbefehl und bewirkt nichts weiter. • War zuvor kein "Processing on the fly"-Mode aktiv, oder es ist derzeit irgendein "Processing on the fly"-Mode aktiv, ist **park_return** ein normaler Sprung an die angegebene Stelle. • Falls eine Koordinatentransformation im virtuellen Bildfeld aktiv ist, so wird die angegebene oder die zuletzt gültige Position nachträglich entsprechend transformiert, siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 177. • Nach **park_return** wird ein Jump-Delay aktiviert. • **park_return** schaltet die Signale für den "Laser active"-Betrieb nach einem LaserOff-Delay ab.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Koordinatenwerte mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**park_position**

Variabler Listenbefehl	**park_return_1_axis**
Funktion	"Fly-Erweiterung"-Befehl: Bewegt die Ausgabeposition für 1 Achse mit Sprunggeschwindigkeit von der Zwischenparkposition zur angegebenen absoluten Position im 2D-Bildfeld.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, führt **park_return_1_axis** nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`park_return_1_axis( Mode, Axis, RetPos )`
Parameter	`Mode` = 0: Rückkehrposition wird ignoriert. > 0: Rückkehrposition. Absolute Koordinate im virtuellen Bildfeld. Als 32-Bit-Wert ohne Vorzeichen. `Axis` Achse aus Tabelle 3, Seite 271. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: `1`...`2`. `RetPos` Absolute Achsen-Koordinate der neuen Ausgabeposition. In Bits. Als 32-Bit-Wert mit Vorzeichen. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_park_return_1_axis**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **park_return_1_axis** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • Siehe auch Hinweise zu **park_return**. • **park_return_1_axis** kann zum Zurückspringen nach **park_position_1_axis** verwendet werden. • Wenn zuvor bei der angegebenen Achse eine "Processing on the fly"-Korrektur durch `park_position_1_axis( Mode = 0 )` oder `park_position_2_axes( Mode = 0 )` deaktiviert wurde, dann schaltet **park_return_1_axis** diese wieder ein. • Bei einem unzulässigen Parameter-Wert wird **park_return_1_axis** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**park_return_2_axes**

Variabler Listenbefehl	**park_return_2_axes**
Funktion	"Fly-Erweiterung"-Befehl: Bewegt die Ausgabeposition für 2 Achsen mit Sprunggeschwindigkeit von der Zwischenparkposition zur angegebenen absoluten Position im 2D-Bildfeld.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, führt **park_return_2_axes** nur den Sprung auf die angegebenen neuen Ausgabepositionen aus.
Aufruf	`park_return_2_axes( Mode, RetPosX, RetPosY )`
Parameter	`Mode` = 0: Rückkehrpositionen werden ignoriert. > 0: Rückkehrpositionen. Absolute Koordinaten im virtuellen Bildfeld. Als 32-Bit-Wert ohne Vorzeichen. `RetPosX` Absolute Achsen-Koordinaten der neuen Ausgabepositionen. In Bits. Als 32-Bit-Wert mit Vorzeichen. Überlaufende Werte werden auf die Randwerte geclippt. `RetPosY` Wie `RetPosX`.
Multi-Board-Befehlsname	**n_park_return_2_axes**

Hinweise	• Als "Fly-Erweiterung"-Befehl darf **park_return_2_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden.
	• Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271.
	• Siehe auch Hinweise zu **park_return**.
	• **park_return_2_axes** kann zum Zurückspringen nach **park_position_2_axes** verwendet werden.
	• Wenn zuvor eine "Processing on the fly"-Korrektur durch `park_position_2_axes( Mode = 0 )` deaktiviert wurde, dann schaltet **park_return_2_axes** diese wieder ein.
	• Bei einem unzulässigen Parameter-Wert wird **park_return_2_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	• Folgende Befehlsaufrufe werden gleichartig ausgeführt:
	– `park_return_2_axes( Mode, RetPosX, RetPosY ) =`
	`{`
	`park_return_1_axis( Mode, 1, RetPosX );`
	`park_return_1_axis( Mode, 2, RetPosY );`
	`}`

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**park_return_1_axis**

Kontrollbefehl	**pause_list**
Funktion	Unterbricht die Ausführung einer Liste und unterdrückt die Signale für den “Laser active”-Betrieb.
Aufruf	`pause_list()`
Parameter	–
Multi-Board-Befehlsname	**n_pause_list**
Hinweise	• **pause_list** ist gleichbedeutend mit **stop_list**. **stop_list** ist aber leicht zu verwechseln mit **stop_execution**. Bevorzugt sollte daher **pause_list** verwendet werden. • Wird **pause_list** während der Ausführung einer Liste aufgerufen, so werden die Signale für den “Laser active”-Betrieb unterdrückt (die Signale werden auf ihren jeweiligen “Off”-Pegel gesetzt) und das Scan-System bleibt im zuletzt vorgegebenen Zustand stehen – gegebenenfalls auch mitten in einer Zerlegung in Mikroschritte. Der (mit **get_status** abrufbare) PAUSED-Listenausführungsstatus wird gesetzt, der BUSY-Listenausführungsstatus dagegen unverändert gelassen. Ein Fortfahren mit **execute_list_pos**, **release_wait** oder mit einem externen Start ist in der Folge nicht möglich. Ein **stop_execution** oder Externer Stopp ist dagegen möglich. **restart_list**, **stop_execution** oder ein Externer Stopp heben die Unterdrückung des Starts wieder auf. • Wenn die Ausführung der Liste fortgesetzt werden soll, dann *muss* **restart_list** verwendet werden. Nach dem anschließenden **restart_list** setzt das Scan-System die vorgesehenen Bewegungen (des aktuellen Befehls) fort und die Lasersteuersignale werden wieder freigegeben (eine nahtlose Fortführung einer unterbrochenen Markierung ist im allgemeinen nicht möglich). Der PAUSED-Listenausführungsstatus wird dann zurückgesetzt (BUSY-Listenausführungsstatus bleibt auch hier unverändert). • Wenn beim Aufrufen des **pause_list** aktuell keine Liste ausgeführt wird (BUSY-Listenausführungsstatus nicht gesetzt) oder eine Liste bereits mit **pause_list** oder **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt), dann wird **pause_list** ignoriert (**get_last_error**-Returncode `RTC6_BUSY`; der Laser ist dann bereits aus).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**restart_list**, **set_wait**

Kontrollbefehl	**periodic_toggle**
Funktion	Wie **periodic_toggle_list**, aber ein Kontrollbefehl.
Aufruf	`periodic_toggle( Port, Mask, P1, P2, Count, Start )`
Parameter	`Port` Wie **periodic_toggle_list**. `Mask` Wie **periodic_toggle_list**. `P1` Wie **periodic_toggle_list**. `P2` Wie **periodic_toggle_list**. `Count` Wie **periodic_toggle_list**. `Start` Wie **periodic_toggle_list**.
Multi-Board-Befehlsname	**n_periodic_toggle**
Hinweise	• Siehe **periodic_toggle_list**. • Bei einem unzulässigen Wert für `Port` oder 0 für `P1` oder `P2` wird **periodic_toggle** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Siehe auch Kapitel 9.4 "Periodische I/O-Signale", Seite 326.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 605, OUT 605: endloses Toggeln bei `Count` = 2^{32}–1.
Verweise	**periodic_toggle_list**

Unverzögerter Kurzer Listenbefehl	**periodic_toggle_list**
Funktion	Erzeugt ein periodisch wechselndes Signal an einem einstellbaren Ausgang.
Aufruf	`periodic_toggle_list( Port, Mask, P1, P2, Count, Start )`
Parameter	`Port` Ausgang (0…4, wie bei **set_port_default**). Als 32-Bit-Wert ohne Vorzeichen. `Mask` Definiert die zu toggelnden Bits (der Höchstwert ist Port-abhängig). 1 = Bit wird getoggelt. 0 = Bit bleibt unverändert. Als 32-Bit-Wert ohne Vorzeichen. `P1` Dauer des getoggelten Signals in [10 µs] (> 0, höchstens 16 Bit). Als 32-Bit-Wert ohne Vorzeichen. `P2` Dauer des getoggelten Signals in [10 µs] (> 0, höchstens 16 Bit). Als 32-Bit-Wert ohne Vorzeichen. `Count` Anzahl der Wiederholungen einer P1–P2-Periode. Als 32-Bit-Wert ohne Vorzeichen. `Start` Dauer bis zum erstmaligen Toggeln in [10 µs]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_periodic_toggle_list**
Hinweise	• Bei einem unzulässigen Wert für `Port` oder 0 für `P1` oder `P2` wird **periodic_toggle_list** durch ein **list_nop** ersetzt und ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • `Mask` legt die Bits fest, die getoggelt werden: mit 1 belegte Bits werden getoggelt, die mit 0 belegten Bits bleiben unverändert. `Mask` wird auf den für den ausgewählten `Port` maximal zulässigen Wert begrenzt (siehe auch **set_port_default**). Überzählige Bits werden ignoriert. • Die ausgewählten Bits werden getoggelt. Dieser Zustand wird für `P1` × 10 µs-Takte beibehalten. Danach werden sie wieder getoggelt und für `P2` × 10 µs-Takte stabil gehalten. Die Anfangswerte (“Off”-Zustand) für die Toggle-Bits (HIGH-aktives oder LOW-aktives “On”-Signal) werden von Benutzern über andere Standard-Befehle festgelegt (**write_da_x**, **write_8bit_port**, **write_io_port**, etc.). Diese – und andere gegebenenfalls anstehende Verzögerter Kurzer Listenbefehle – werden noch vor **periodic_toggle_list** ausgeführt. • Das Toggeln findet erstmals nach `Start` × 10 µs-Takten statt und wird dann `Count` mal wiederholt. Für `Count` = 0 wird eine laufende Ausgabe sofort gestoppt. Die restlichen Parameter sind dann irrelevant. Wenn gerade die `P1`-Periode (“On”) aktiv war, wird noch einmal getoggelt, um den Anfangszustand (“Off”) wieder herzustellen. • `P1` und `P2` werden auf den Wertebereich [0…65.535] geclippt. `P1` und `P2` dürfen beide nicht 0 sein.

Unverzögerter Kurzer Listenbefehl	**periodic_toggle_list**
Hinweise (Forts.)	• Der ausgewählte `Port` sollte nicht gleichzeitig für andere Ausgaben, etwa eine "Automatische Laser-Steuerung", verwendet werden. • Bei einem **set_end_of_list**, **stop_execution** oder externem /STOP laufen die periodischen Signale weiter. • **periodic_toggle_list** toggelt bei `Count` = 2^{32}–1 endlos. • Siehe auch Kapitel 9.4 "Periodische I/O-Signale", Seite 326.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 605, OUT 605: endloses Toggeln bei `Count` = 2^{32}–1.
Verweise	**periodic_toggle**, **set_port_default**

Kontrollbefehl	**quit_loop**
Funktion	Beendet den mit **start_loop** aktivierten automatischen Listenwechsel.
Aufruf	`quit_loop()`
Multi-Board-Befehlsname	**n_quit_loop**
Hinweise	• Bevor die Ausführung beendet wird, wird die aktuelle Liste noch komplett bis zum Erreichen des nächsten **set_end_of_list** abgearbeitet. • **start_loop** muss vor **quit_loop** aufgerufen werden. Sonst hat **quit_loop** keine Auswirkung.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**start_loop**

Unverzögerter Kurzer Listenbefehl	**range_checking**	
Funktion	Definiert eine Notfallmaßnahme im Fall einer Galvanometerscanner-Positionsbereichsüberschreitung.	
Aufruf	`range_checking( HeadNo, Mode, Data )`	
Parameter	`HeadNo`	Zu überwachendes Scan-System. Als 32-Bit-Wert ohne Vorzeichen. 0: Die Überwachung ist ausgeschaltet. Default nach **load_program_file**. 1: Überwacht wird Anschluss für ersten Scan-Kopf. 2: Überwacht wird Anschluss für zweiten Scan-Kopf. 3: Beide Anschlüsse werden überwacht. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`Mode`	Zu treffende Maßnahme. Als 32-Bit-Wert ohne Vorzeichen. 0: Die Signale für den "Laser active"-Betrieb werden sofort unterdrückt. Die Listenausführung läuft weiter. 1: Die Signale für den "Laser active"-Betrieb werden sofort ausgeschaltet. Die Listenausführung wird sofort beendet (wie **stop_execution** oder /STOP). 2: Ein **simulate_ext_stop** wird an alle Slave-Karten weitergeleitet.
	`Data`	Zur Überwachung ausgewählter Datentyp. Als 32-Bit-Wert ohne Vorzeichen. 0: Sample-Werte. Wie **set_trigger**-Signaltypen 7…9. 1: Transformierte Ansteuerwerte. Wie **set_trigger**-Signaltypen 25…30. 2: Korrigierte Ansteuerwerte. Wie **set_trigger**-Signaltypen10…15. 3: Tatsächliche Ausgabewerte. Wie **set_trigger**-Signaltypen 20…23. 4: Galvanometerscanner-Ist-Position (nur mit iDRIVE-Scan-Systemen, vario*SCAN* ist intern an einer x-Achse angeschlossen). Wie **set_trigger**-Signaltypen [1, 2 und 4] oder [4, 5 und 1]. 5: Galvanometerscanner-Ist-Position (nur mit iDRIVE-Scan-Systemen, vario*SCAN* ist intern an einer y-Achse angeschlossen). Wie **set_trigger**-Signaltypen [1, 2 und 5] oder [4, 5 und 2].
Multi-Board-Befehlsname	**n_range_checking**	

Unverzögerter Kurzer Listenbefehl	**range_checking**
Hinweise	• Ist dem angegebenen Scan-Kopf keine Korrekturtabelle zugewiesen (siehe **select_cor_table**), wird auch keine Überwachung ausgeführt. • Als Positionsgrenzen, bei deren Überschreitung die Notfallmaßnahme ausgeführt wird, werden die Parameter aus der kundenspezifischen Überwachung von "Processing on the fly"-Anwendungen verwendet (siehe **set_fly_limits**, **set_fly_limits_z**). Diese Limits führen bei "Processing on the fly"-Anwendungen lediglich zu Fehlermeldungen. Die Fehlermeldungen können mit **get_marking_info** oder den entsprechenden **if_fly_x_overflow**/**if_fly_y_overflow**/**if_fly_z_overflow** und **if_not_fly_x_overflow**/**if_not_fly_y_overflow**/**if_not_fly_z_overflow** abgefragt werden. Sie lösen selbst keine Aktivitäten aus. • Die kundenspezifischen Bereichsgrenzen werden bei "Processing on the fly"-Anwendungen mit dem Datentyp `Data` = 0 verwendet (Sample-Werte). Für **range_checking**-Datentypen `Data` > 0 kann die gleichzeitige Verwendung zu inkonsistenten Fehlermeldungen und Abschaltungen führen. • Wurde der Laser mit `Mode` = 0 abgeschaltet und die Fehlerbedingung liegt nicht mehr vor, dann wird er erst mit dem nächsten Markier-Befehl wieder eingeschaltet. Er wird *nicht* mitten in einem aktuell ablaufenden Markier-Befehl eingeschaltet, auch nicht mitten in einem Polygonzug. • Die **range_checking**-Überwachung ist auch ohne "Processing on the fly"-Anwendung wirksam. • `Data` > 5 führt zu einem **get_last_error**-Returncode `RTC6_PARAM_ERROR`. **range_checking** wird in diesem Fall als **list_nop** an die RTC6 Karte gesendet. • `Data`=2 und `Data`=3 haben für die z-Achse identische Auswirkungen. • Für `Data` = 4 und `Data` = 5 muss die Ist-Position als vom Scan-System zu übertragender Datentyp vom Benutzer selbst eingestellt werden. Eine gleichzeitige Verwendung mit einer Geschwindigkeits-abhängige Laser-Steuerung mit `Mode` = 2 (siehe **set_auto_laser_control**) ist nicht möglich. • `Data` = 4 und `Data` = 5 unterscheiden sich nur um die Achse, an die ein vario*SCAN* angeschlossen ist. Für 2D-Systeme ohne vario*SCAN* haben beide Auswahlen identische Auswirkungen. • Ist für `Data` = 4 und `Data` = 5 kein intelli*SCAN* angeschlossen und eingeschaltet oder als Rückantwort ist keine Ist-Position eingestellt, ergeben sich unsinnige Abschaltungen.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 615: `Mode` = 2.
Verweise	**set_fly_limits**, **set_fly_limits_z**

Kontrollbefehl	read_abc_from_file	
Funktion	Wie **read_abc_from_file_20b**. Jedoch für einen Fokuslängenwert *f* im RTC4-Kompatibilitätsbereich [–32.768…+32.767].	
Aufruf	`ErrorNo = read_abc_from_file( Name, &A, &B, &C )`	
Rückgabe	`ErrorNo`	Wie **read_abc_from_file_20b**.
Parameter	`Name`	Wie **read_abc_from_file_20b**.

Parameter-Rückgaben	`A`
	`B`
	`C`
		Koeffizienten der parabolischen Funktion $z_{out} = A + Bf + Cf^2$, die zur Berechnung der z-Ausgabewerte verwendet wird (Fokuslängenwert *f* im RTC4-Kompatibilitätsbereich [–32.768…+32.767]). Als Werte im 64-Bit-IEEE Gleitkommaformat.

Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.	
Hinweise	• Wie **read_abc_from_file_20b**. • `read_abc_from_file( &A, &B, &C )` ist gleichbedeutend mit `read_abc_from_file_20b( &A, &B, &C × 1/16 )`.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**write_abc_to_file**	

Kontrollbefehl	read_abc_from_file_20b
Funktion	Liest die ABC-Werte direkt aus der angegebenen Korrekturdatei im PC. Für einen Fokuslängenwert *f* im RTC6-20-Bit-Bereich [–524.288…+524.287].
Aufruf	`ErrorNo = read_abc_from_file_20b( Name, &A, &B, &C )`
Rückgabe	`ErrorNo` Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. `0` Kein Fehler. `1` Dateifehler (korrupt oder unvollständig). `2` Memory Fehler (RTC6-DLL-intern, Windows-Arbeitsspeicher). `3` Datei-Open Fehler (leerer String, Datei nicht gefunden etc.).
Parameter	`Name` Name der Korrekturdatei. Als Zeiger auf einen nullterminierten ANSI-String.

Parameter-Rückgaben	`A`
	`B`
	`C`
	Koeffizienten der parabolischen Funktion $z_{out} = A + Bf + Cf^2$, die zur Berechnung der z-Ausgabewerte verwendet wird (Fokuslängenwert *f* im RTC6-20-Bit-Bereich [–524.288…+524.287]).
	Als Werte im 64-Bit-IEEE Gleitkommaformat.

Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• **read_abc_from_file_20b** ist auch ohne explizite Zugriffsberechtigung auf irgendeine RTC6-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133, werden von **read_abc_from_file_20b** weder erzeugt noch verändert. • `read_abc_from_file_20b( &A, &B, &C )` ist gleichbedeutend mit `read_abc_from_file( &A × 16, &B, &C × 1/16 )`.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 631, OUT 632.
Verweise	**write_abc_to_file_20b**

Kontrollbefehl	**read_analog_in**
Funktion	Liest die analogen Eingabewerte ein: • ANALOG IN0 und ANALOG IN1 an der McBSP/ANALOG-Stiftleiste der RTC6 PCIe-Karte • ANALOG IN0 und ANALOG IN1 an der SPI/ANA/UART-Stiftleiste der RTC6 Ethernet-Karte Gibt sie als 12-Bit-Digitalwerte zurück.
Aufruf	`AnalogValue = read_analog_in()`
Rückgabe	Als 32-Bit-Wert ohne Vorzeichen. Bit #0 ANALOG IN0-Eingabewert als 12-Bit-Digitalwert. … Bit #11 Bit #12 = 0. Kanalnummer. Bit #13 = 0. Bit #14 = 0. Bit #15 Old-Bit für ANALOG IN0. Bit #16 ANALOG IN1-Eingabewert als 12-Bit-Digitalwert. … Bit #27 Bit #28 = 1. Kanalnummer. Bit #29 = 0. Bit #30 = 0. Bit #31 Old-Bit für ANALOG IN1.
Multi-Board-Befehlsname	**n_read_analog_in**
Hinweise	• Siehe auch Abschnitt "Analog-Eingänge", Seite 89. • **read_analog_in** setzt die Old-Bits (Bit #15 und Bit #31) nach dem Einlesen auf 1. Sobald am jeweiligen Analog-Eingang neue Daten vorliegen, werden sie automatisch wieder auf 0 gesetzt. • Die analogen Eingabewerte (ANALOG IN0 und ANALOG IN1) können mit **set_trigger[*]** (Signal 54) aufgezeichnet werden. Allerdings werden die Old-Bits (Bit #15 und Bit #31) nicht aufgezeichnet. Das Format der Daten ist ((ANALOG IN1 << 16) + ANALOG IN0).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_trigger**, **set_trigger4**, **set_trigger8**

Kontrollbefehl	read_encoder
Funktion	Gibt die mit **store_encoder** zwischengespeicherten Zählerstände der beiden RTC6-Encoderzähler zurück.
Aufruf	`read_encoder( &Encoder0_0, &Encoder1_0, &Encoder0_1, &Encoder1_1 )`
Parameter-Rückgaben	`Encoder0_0` Zählerstand. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen. Für Parameter 'Encoder*n*_*m*' bedeutet: • *n*: Nummer der Encoderzählers ("Encoder0", "Encoder1"). • *m*: Nummer der Speicherposition (Parameter `Pos` von **store_encoder**).
	`Encoder1_0` Wie `Encoder0_0` (analog).
	`Encoder0_1` Wie `Encoder0_0` (analog).
	`Encoder1_1` Wie `Encoder0_0` (analog).
Multi-Board-Befehlsname	**n_read_encoder**
Hinweise	• Siehe auch Kapitel 8.6 ""Processing on the fly"", Seite 254 und Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323. • Wird die Werkstück-Bewegung mit einem Inkremental-Encoder registriert, so wird getriggert: – Encoderzähler "Encoder0" durch die Signale am Encodereingang ENCODER X – Encoderzähler "Encoder1" durch die Signale am Encodereingang ENCODER Y • Wurde mit **simulate_encoder** eine Encodersimulation gestartet, dann werden die Encoderzähler durch ein internes periodisches 1 MHz-Taktsignal getriggert. • Von Speicherpositionen, in die nicht zuvor mit **store_encoder** ein Zählerstand gespeichert wurde, wird jeweils der Wert 0 (Initialisierungswert) zurückgegeben.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**store_encoder**, **get_encoder**, **set_fly_x**, **set_fly_y**, **set_fly_rot**, **wait_for_encoder**

Kontrollbefehl	read_image_eth
Funktion	Standalone-Funktionalität: Liest die Daten für das automatische Booten aus dem NAND-Speicher und speichert sie als Binärdatei auf dem PC.
Voraussetzung	RTC6-Software-Paket ≥ V1.7.0 und BIOS-ETH ≥ 26.
Aufruf	`Result = read_image_eth( Name )`
Parameter	`Name` Name der Binärdatei. Als Zeiger auf einen nullterminierten ANSI-String.
Rückgabe	`Result` Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Kein Fehler. 1 Keine RTC6 Ethernet-Karte. 2 Leerer String als Dateiname angegeben. 3 Zugriff verweigert. 4 Die RTC6 Ethernet-Karte ist BUSY-Listenausführungsstatus. 5 Timeout-Fehler (RTC6 Ethernet-Karte antwortet nicht). 6 Datei kann nicht geöffnet werden. 7 Ethernet-Fehler, siehe **eth_get_last_error**. 8 NAND-Speicherfehler. 9 Vorzeitig abgebrochen, siehe **eth_get_last_error**.
Multi-Board-Befehlsname	**n_read_image_eth**
Hinweise	• **read_image_eth** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • Wenn `Name` nicht geöffnet werden kann, wird ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Der Inhalt einer bereits vorhandenen Binärdatei wird gelöscht. • Während der Ausführung von **read_image_eth** ist der 10 µs-Takt des DSP für bis zu 2 Minuten unterbrochen. • **read_image_eth** ist nur bei RTC6 Ethernet-Karten erlaubt. Anderenfalls wird ein **get_last_error**-Returncode `RTC6_TYPE_REJECTED` erzeugt. • Siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 618, OUT 618, RBF 623.
Verweise	**write_image_eth**, **store_program**

Kontrollbefehl	**read_io_port**
Funktion	Liest den aktuellen Zustand des 16-Bit-Digital-Eingangs der EXTENSION 1-Stiftleiste aus.
Aufruf	`IOPort = read_io_port()`
Rückgabe	16-Bit-Wert (DIGITAL IN 0…DIGITAL IN 15). Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_read_io_port**
Hinweise	• Siehe Kapitel 9.2.1 "16-Bit-Digital-Eingang", Seite 313.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_io_port**, **write_io_port_mask**, **set_io_cond_list**, **clear_io_cond_list**, **read_io_port_buffer**, **read_io_port_list**

Kontrollbefehl	read_io_port_buffer	
Funktion	Gibt die Daten zurück, die zuvor mit **read_io_port_list** im IOPort-Buffer abgespeichert wurden (ausgelesener Eingangswert am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste, zum Lesezeitpunkt aktuell gewesene Galvanometerscanner-Soll-Position und Zeitangabe).	
Aufruf	`CurrentIndex = read_io_port_buffer( Index, &Value, &XPos, &YPos, &Time )`	
Parameter	`Index`	Nummer des auszulesenden Eintrags im IOPort-Buffer. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die 12 niederwertigsten Bits ausgewertet (der IOPort-Buffer umfasst nur 4.096 Einträge). Dadurch kann im Anwenderprogramm ein durchlaufender Zähler für den Index verwendet werden.
Parameter-Rückgaben	`Value`	16-Bit-Wert. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
	`XPos`	Galvanometerscanner-Sollposition (x-Wert). Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen.
	`YPos`	Galvanometerscanner-Sollposition (y-Wert). Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen.
	`Time`	Zeit. In Sekunden. Als Zeiger auf einen 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Derjenige Index, an den beim nächsten **read_io_port_list** die Daten geschrieben werden. Als 32-Bit-Wert ohne Vorzeichen.	
Multi-Board-Befehlsname	**n_read_io_port_buffer**	
Hinweise	• Siehe auch Hinweise unter **read_io_port_list**. • Wenn vor **read_io_port_buffer** noch kein **read_io_port_list** aufgerufen wurde, werden die Initialisierungswerte (Voreinstellung: 0) zurückgegeben.	

Beispiel (C/C++)	Warten bis die Daten auf `Index` aktuell neu beschrieben worden sind:
	`while (Index == read_io_port_buffer( Index, &Value, &XPos, &YPos, &Time ));`
	Lesen aller Daten ab `Index` bis zur aktuellen (noch nicht neu beschriebenen) Leseposition:
	`while (Index != read_io_port_buffer( Index, &Value, &XPos, &YPos, &Time ))`
	`{ Index = (Index+1) & 0xf; …}`	

RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität. Größerer IOPort-Buffer.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**read_io_port_list**	

Unverzögerter Kurzer Listenbefehl	**read_io_port_list**
Funktion	Schreibt den aktuellen Zustand (DIGITAL IN 0…DIGITAL IN 15) des 16-Bit-Digital-Eingangs der EXTENSION 1-Stiftleiste in den internen IOPort-Buffer. Gleichzeitig werden die aktuellen xy-Soll-Positionen sowie die *relative* Zeit in Sekunden abgespeichert.
Aufruf	`read_io_port_list()`
Multi-Board-Befehlsname	**n_read_io_port_list**
Hinweise	• **read_io_port_list** gibt keine Werte zurück. Diese können aber anschließend mit **read_io_port_buffer** aus dem IOPort-Buffer ausgelesen werden. • Der IOPort-Buffer umfasst 4.096 Einträge und ist zyklisch organisiert. Er speichert immer die jeweils letzten 4.096 Einträge ab und überschreibt die vorherigen ohne Warnung. Der fortlaufende Index startet nach einem Reset (Programmstart) mit Index 0 und beginnt nach 4.095 wieder mit 0. • Die abgespeicherte Zeit ist der aktuelle Wert eines fortlaufenden internen Sekundenzählers, der mit dem Programmstart (**load_program_file**) oder durch **time_update** auf 0 gesetzt wird. • Siehe auch Kapitel 9.2.1 "16-Bit-Digital-Eingang", Seite 313.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Größerer IOPort-Buffer.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**read_io_port_buffer**, **read_io_port**

Kontrollbefehl	**read_mcbsp**
Funktion	Liest den Eingabewert, der zuletzt vollständig über die McBSP-Schnittstelle an eine interne Speicherstelle übertragen wurde.
Aufruf	`mcbsp_value = read_mcbsp( No )`
Parameter	`No` Nummer der internen Speicherstelle, von der der Eingabewert gelesen werden soll. Als 32-Bit-Wert ohne Vorzeichen. 0 Speicherstelle für "Processing on the fly"-Anwendungen und die **set_mcbsp_in**-Eingabe mit Codierungs-Bit #31 = 0. 1 Speicherstellen für Online-Positionierung und die **set_mcbsp_in**-Eingabe. 2 Wie 1. 3 Speicherstelle für **set_mcbsp_in**-Eingabe mit Codierungs-Bit #31 = 1. Für **set_multi_mcbsp_in** gilt: Die Speicherstellen 0 bis 3 werden als Ringpuffer fortlaufend beschrieben. Es werden nur die 2 niederwertigsten Bits ausgewertet.
Rückgabe	Eingabewert. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_read_mcbsp**
Hinweise	• Zur McBSP-Schnittstelle und den zugehörigen Speicherstellen siehe auch Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87. • Die Interpretation als – ein 32 Bit-Datenwort mit oder ohne Vorzeichen, – zwei 16 Bit-Datenworte mit Vorzeichen – zwei 15-Bit-Datenworte mit Vorzeichen liegt in der Verantwortung des Benutzers: – Zu "Processing on the fly"-Anwendungen (`No` = 0) siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Kapitel 9.3.4 "Synchronisation und Online-Positionierung mit McBSP-Signalen", Seite 325. – Zur "Lokale Online-Positionierung" (`No` = 1…2) siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240. – Zur "Globale Online-Positionierung" (`No` = 1…2) siehe Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243. – Zur **set_mcbsp_in**-Eingabe (`No` = 0…3) siehe Abschnitt "Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe", Seite 258 und Abschnitt "Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe", Seite 260. – Zur **set_multi_mcbsp_in**-Eingabe siehe Seite 781. • Nach **load_program_file** werden Eingabewerte an der McBSP-Schnittstelle an die Speicherstelle 0 übertragen (Default), auch dann, wenn noch keine "Processing on the fly"-Korrektur aktiviert wurde. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Seite 89.

Kontrollbefehl	read_mcbsp
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_out**, **mcbsp_init**, **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos**, **set_mcbsp_x**, **set_mcbsp_y**, **set_mcbsp_rot**, **set_mcbsp_matrix**, **apply_mcbsp**, **set_mcbsp_global_x**, **set_mcbsp_global_y**, **set_mcbsp_global_rot**, **set_mcbsp_global_matrix**

Kontrollbefehl	read_multi_mcbsp
Funktion	Liest den Eingabewert zurück, der zuletzt über die Mehrfach-McBSP-Übertragung typsortiert gespeichert wurde.
Aufruf	`mcbsp_value = read_multi_mcbsp( No )`
Parameter	`No` Nummer der typsortierten internen Speicherstelle. `No` entspricht den Codierungs-Bit #0…Bit #2 der Mehrfach-McBSP-Eingabe. Als 32-Bit-Wert ohne Vorzeichen. 0 x-Koordinate der "Processing on the fly"-Anwendung. 1 y-Koordinate der "Processing on the fly"-Anwendung. 2 z-Koordinate der "Processing on the fly"-Anwendung. 3 Laserleistung P. 4 Extraparameter E1. 5 Extraparameter E2. 6 Extraparameter E3. 7 Extraparameter E4. Es werden nur die 3 niederwertigsten Bits ausgewertet.
Rückgabe	Speicherwert. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_read_multi_mcbsp**
Hinweise	• Die Mehrfach-McBSP-Übertragung sollte zuvor mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktiviert und verwendet worden sein. Anderenfalls werden Default-Werte oder veraltete Werte ausgelesen.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_multi_mcbsp_in**, **set_multi_mcbsp_in_list**

Kontrollbefehl	read_status		
Funktion	Gibt den Listenstatus der RTC6 zurück, siehe Kapitel 6.4.2 "Listenstatus", Seite 110.		
Aufruf	`Status = read_status()`		
Rückgabe	Listenstatus. Als 32-Bit-Wert ohne Vorzeichen.		
	Bit #	Name	Beschreibung
	Bit #0 (LSB)	LOAD1	= 1: Zeigt an, dass sich der Eingabe-Zeiger aktuell in "Liste 1" befindet, d.h. dass alle nachfolgenden Listenbefehle in "Liste 1" gespeichert werden. LOAD1 wird gesetzt, wenn (z.B. mit **set_start_list_pos**) der Eingabe-Zeiger in die "Liste 1" gesetzt wird. LOAD1 wird zurückgesetzt, wenn ein **set_end_of_list** in die "Liste 1" geschrieben wird (READY1 gesetzt) oder wenn LOAD2 (z.B. mit **set_start_list_pos**) gesetzt wird.
	Bit #1	LOAD2	= 1: Zeigt an, dass sich der Eingabe-Zeiger aktuell in "Liste 2" befindet, d.h. dass alle nachfolgenden Listenbefehle in "Liste 2" gespeichert werden. LOAD2 wird gesetzt, wenn (z.B. mit **set_start_list_pos**) der Eingabe-Zeiger in die "Liste 2" gesetzt wird. LOAD2 wird zurückgesetzt, wenn ein **set_end_of_list** in die "Liste 2" geschrieben wird (READY2 gesetzt) oder wenn LOAD1 (z.B. mit **set_start_list_pos**) gesetzt wird.
	Bit #2	READY1	= 1: Zeigt an, dass während des Lade-Vorgangs ein **set_end_of_list** in die "Liste 1" geschrieben wurde. READY1 wird zurückgesetzt, wenn LOAD1 (z.B. mit **set_start_list_pos**) neu gesetzt wird.
	Bit #3	READY2	= 1: Zeigt an, dass während des Lade-Vorgangs ein **set_end_of_list** in die "Liste 2" geschrieben wurde. READY2 wird zurückgesetzt, wenn LOAD2 (z.B. mit **set_start_list_pos**) neu gesetzt wird.
	Bit #4	BUSY1	= 1: Zeigt an, dass gerade "Liste 1" ausgeführt wird (genauer: dass sich der Ausgabe-Zeiger aktuell in "Liste 1" befindet, nachdem eine Listenausführung von "Liste 1" oder "Liste 2" gestartet wurde). Gesetzt wird BUSY1 durch Starten der "Liste 1" (z.B. mit **execute_list_pos**) oder durch einen Listenwechsel nach "Liste 1" (automatischer Listenwechsel oder Sprung). Zurückgesetzt wird BUSY1 durch Ausführen von **set_end_of_list** in "Liste 1", durch einen Sprung in die "Liste 2 (z.B. mit **list_jump_pos**), durch Starten der "Liste 2" (z.B. mit **execute_list_pos**) oder durch **stop_execution**.

Kontrollbefehl	read_status			
Rückgabe (Forts.)	Bit #5	BUSY2	= 1:	Zeigt an, dass gerade “Liste 2“ ausgeführt wird (genauer: dass sich der Ausgabe-Zeiger aktuell in “Liste 2“ befindet, nachdem eine Listenausführung von “Liste 1“ oder “Liste 2“ gestartet wurde). Gesetzt wird BUSY2 durch Starten der “Liste 2” (z.B. mit **execute_list_pos**) oder durch einen Listenwechsel nach “Liste 2“ (automatischer Listenwechsel oder Sprung). Zurückgesetzt wird BUSY2 durch Ausführen des **set_end_of_list**s in “Liste 2”, durch einen Sprung in die “Liste 1 (z.B. mit **list_jump_pos**), durch Starten der “Liste 1” (z.B. mit **execute_list_pos**) oder durch **stop_execution**.
	Bit #6	USED1	= 1:	Zeigt an, dass beim Abarbeiten von “Liste 1“ ein **set_end_of_list** erreicht wurde. USED1 wird zurückgesetzt, wenn LOAD1 (z.B. mit **set_start_list_pos**) gesetzt wird.
	Bit #7	USED2	= 1:	Zeigt an, dass beim Abarbeiten von “Liste 2“ ein **set_end_of_list** erreicht wurde. USED2 wird zurückgesetzt, wenn LOAD2 (z.B. mit **set_start_list_pos**) gesetzt wird.
	Bit #8 … Bit #31		0	
Multi-Board-Befehlsname	**n_read_status**			
Hinweise	• Bei der Interpretation der mit **read_status** zurückgelesenen Statuswerte sollte immer der jeweils programmierte Lade- oder Ausführungsvorgang der Listen berücksichtigt werden. Unter gewissen Umständen können die Statuswerte nämlich auch irreführend sein, wie die folgenden Beispiele zeigen: – Auch während eines laufenden Ladevorgangs kann der LOAD-Listenstatus bereits zurückgesetzt und der READY-Listenstatus gesetzt sein, wenn nach dem Laden eines **set_end_of_list** weitere Listenbefehle in denselben Listenbereich geladen werden. – Die Statuswerte ändern sich nicht, wenn ein **set_end_of_list** durch einen anderen Befehl überschrieben wird (der READY-Listenstatus wird nicht zurückgesetzt). – Wenn während eines laufenden Ladevorgangs in derselben Liste ein **set_end_of_list** abgearbeitet wird, gilt die Liste als bereits abgearbeitet (USED-Listenstatus gesetzt), obwohl sie noch neu geladen wird (USED-Listenstatus war eigentlich zurückgesetzt). – Auch wenn in einem Listenbereich eine vollständig geladene Befehlsliste (inkl. **set_end_of_list**) gespeichert ist, kann der READY-Listenstatus einer Liste zurückgesetzt sein, wenn der Eingabe-Zeiger – z.B. mit **set_input_pointer** – neu in die Liste gesetzt wurde. Dann kann auch der USED-Listenstatus einer Liste zurückgesetzt sein, obwohl eine vollständig geladene und bereits abgearbeitete Befehlsliste gespeichert ist. – Bei Sprüngen während der Ausführung von einem Listenbereich in einen anderen (“Liste 1” <-> “Liste 2”, z.B. mit **list_jump_pos**) bleiben die USED-Listenstatuswerte beider Listen (im Gegensatz zu den BUSY-Listenausführungsstatus-Werten) unverändert und sind dann nicht mehr aussagekräftig.			

Kontrollbefehl	read_status
Hinweise (Forts.)	• Wenn der Listenstatus während der Abarbeitung eines Unterprogramms im geschützten RTC6-Listenspeicher-Bereich “Liste 3” abgefragt wird, so wird der Status derjenigen Liste (“Liste 1” oder “Liste 2”) zurückgegeben, in der sich der Ausgabe-Zeiger zuletzt befunden hat (in der Regel von wo aus das Unterprogramm ursprünglich aufgerufen wurde). • Bei einer Unterbrechung der Listenausführung (mit **pause_list**, **stop_list** oder **set_wait**) bleiben die Listenstatuswerte unverändert. • Die Listenausführungsstatuswerte BUSY-Listenausführungsstatus und PAUSED-Listenausführungsstatus können mit **get_status** ausgelesen werden. • Zum Auslesen der Statussignale der Scan-Köpfe dient **get_head_status**.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Zusätzlicher USED-Listenstatus.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_status**, **get_head_status**

Kontrollbefehl	read_user_data
Funktion	Ohne Funktion.
Multi-Board-Befehlsname	**n_read_user_data**
Hinweise	• **read_user_data** hat zur Zeit keinerlei Wirkung.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**send_user_data**

Normaler Listenbefehl	**regulation3**	
Funktion	Kategorie: Listenbefehle nur für einen begrenzten Benutzerkreis. Variiert Laserfrequenz und Pulslänge abhängig von der Encoderfrequenz.	
Aufruf	`regulation3( Fmax, Fmin )`	
Parameter	`Fmax`	Höchste Encoderfrequenz. In Hz. Zulässiger Wertebereich: 10.000…1.000.000. Als 32-Bit-Wert ohne Vorzeichen.
	`Fmin`	Abschaltfrequenz. In Hz. Zulässiger Wertebereich: 10…[`Fmax` / 10]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_regulation3**	
Hinweise	• **regulation3** wird solange endlos ausgeführt, bis mit **stop_execution** oder /STOP die Listenausführung beendet wird. • Für Encoderfrequenz < `Fmin` wird der Laser ausgeschaltet. • Eine Encoderfrequenz > `Fmax` wird auf `Fmax` geclippt. • Der Wert für die Laserfrequenz ergibt sich aus der Lookup-Tabelle von **set_laser_timing_table**. • Der Wert für die Pulslänge ergibt sich aus der Lookup-Tabelle von **set_duty_cycle_table**. • Für Listenbefehle nur für einen begrenzten Benutzerkreis gilt: Diese Benutzer wurden im Umgang mit diesen RTC6-Befehlen separat informiert. Deswegen wird nur die Befehlstabelle aufgenommen und auf weitere Informationen in diesem Handbuch verzichtet.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 626, OUT 627.	
Verweise	**set_duty_cycle_table**, **set_laser_timing_table**	

Kontrollbefehl	release_rtc
Funktion	Gibt die ausgewählte RTC6-Karte zur Verwendung durch andere Anwenderprogramme frei.
Aufruf	`NoOfReleasedCard = release_rtc( CardNo )`
Parameter	`CardNo` RTC6-DLL-interne Nummer der RTC6-Karte. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Der Rückgabewert ist `CardNo`, wenn das Anwenderprogramm zuvor im Besitz der Zugriffsberechtigung für diese Karte war. Sonst wird 0 zurückgegeben. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Nach der Freigabe einer Karte mit **release_rtc** hat das Anwenderprogramm keine Zugriffsberechtigung mehr für die Karte. Anschließend kann die Karte dann wieder von demselben oder von einem anderen Anwenderprogramm mit **acquire_rtc** oder **init_rtc6_dll** beansprucht werden. • Falls die mit **release_rtc** freigegebene Karte die aktive Karte war, sind anschließend – außer den Multi-Board-Befehlen – nur noch diejenigen Nicht-Multi-Board-Befehle verfügbar, die keine explizite Zugriffsberechtigung erfordern. Es wird nicht automatisch eine andere Karte als aktive Karte ausgewählt. Die Aktivierung einer anderen Karte (für die eine Zugriffsberechtigung an das Anwenderprogramm vergeben ist) kann mit **select_rtc** ausgeführt werden. • **release_rtc** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar, bleibt dann aber immer wirkungslos (mit Rückgabewert 0). • **release_rtc** bleibt auch wirkungslos (mit Rückgabewert 0), wenn: – `CardNo` die Zahl der bei der Initialisierung gefundenen RTC6-Karten (siehe **rtc6_count_cards**) übersteigt *und* – bei `CardNo` keine RTC6 Ethernet-Karte eingetragen ist – `CardNo` = 0 ist (reale Karten zählen ab 1)
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**acquire_rtc**, **init_rtc6_dll**

Kontrollbefehl	release_wait
Funktion	Setzt die Abarbeitung einer Liste fort, die mit **set_wait** angehalten wurde.
Aufruf	`release_wait()`
Multi-Board-Befehlsname	**n_release_wait**
Hinweise	• **release_wait** wird nur ausgeführt, wenn die RTC6 tatsächlich im Unterbrechungs-Status ist (d.h. wenn zuvor ein Unterbrechungspunkt erreicht und die Listenabarbeitung unterbrochen wurde; der PAUSED-Listenausführungsstatus ist dann gesetzt und der BUSY-Listenausführungsstatus *nicht*). Anderenfalls wird **release_wait** ignoriert (**get_last_error**-Returncode `RTC6_BUSY`). • Durch **release_wait** wird der (mit **get_status** abrufbare) PAUSED-Listenausführungsstatus zurückgesetzt und der BUSY-Listenausführungsstatus erneut gesetzt, siehe auch Kapitel 6.4.3 "Listenausführungsstatus", Seite 111. • **release_wait** setzt das `WaitWord` (das bei einer Unterbrechung den Wert der Unterbrechungspunkt-Nummer annimmt) auf Null zurück. • Wurde mit **set_wait** ein (mit **home_position** oder **home_position_xyz** definierter) Home-Jump ausgeführt, dann führt **release_wait** einen entsprechenden Home-Return aus (während des Home-Returns ist der INTERNAL-BUSY-Listenausführungsstatus gesetzt). • Der Unterbrechungs-Status kann mit **get_wait_status** abgefragt werden.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Zusätzlicher PAUSED-Listenausführungsstatus. Dieser wird durch **set_wait** gesetzt und durch **release_wait** zurückgesetzt.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_wait**, **get_wait_status**, **restart_list**

Kontrollbefehl	**reset_error**
Funktion	Setzt den akkumulierten Fehlercode zurück.
Aufruf	`reset_error( Code )`
Parameter	`Code` Oder-Verknüpfung der Fehlercodes = Summe über $2^{\text{Bitnummer}}$ für alle Bits, die zurückgesetzt werden sollen. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_reset_error**
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133. • Der akkumulierte Fehlercode `AccError` kann zurückgesetzt werden: – Bitweise (für jeden Fehlertyp einzeln, z.B. Bit #5 und Bit #6 mit `Code` = $2^5 \mid 2^6$ oder mit `Code` = `RTC6_BUSY` | `RTC6_REJECTED`) – Komplett (für `Code` = "–1", d.h. für `Code` = $2^{32}-1$) Die Zuordnung zwischen den Bitnummern, Fehlertypen und Fehlerkonstanten ist bei **get_error** beschrieben. • **reset_error** löscht nicht den Fehlercode eines anderen Anwenderprogramms, das gerade für die angegebene Karte zugriffsberechtigt ist. • **reset_error** und **n_reset_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Die kartenspezifische Fehlervariable `LastError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) wird von **reset_error** weder erzeugt noch verändert. `AccError` wird dagegen wie mit `Code` angegeben verändert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_error**, **get_last_error**

Kontrollbefehl	**restart_list**
Funktion	Gibt die Signale für den "Laser active"-Betrieb wieder frei und setzt eine mit **pause_list** oder **stop_list** unterbrochene Listenausgabe fort.
Aufruf	`restart_list()`
Multi-Board-Befehlsname	**n_restart_list**
Hinweise	• **restart_list** wird nur ausgeführt, wenn zuvor eine Liste mit **pause_list** oder **stop_list** angehalten wurde und sowohl der (mit **get_status** auslesbare) BUSY-Listenausführungsstatus als auch der PAUSED-Listenausführungsstatus gesetzt sind. Anderenfalls (z.B. wenn eine Liste mit **set_wait** angehalten wurde) wird **restart_list** ignoriert (**get_last_error**-Returncode `RTC6_BUSY`). • **restart_list** setzt den PAUSED-Listenausführungsstatus zurück. Der BUSY-Listenausführungsstatus wird unverändert gelassen. • Eine nahtlose Fortführung einer unterbrochenen Markierung ist im allgemeinen nicht möglich.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Zusätzlicher PAUSED-Listenausführungsstatus. Dieser wird durch **pause_list** und **stop_list** gesetzt und durch **restart_list** zurückgesetzt.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**pause_list**, **stop_list**, **release_wait**

Kontrollbefehl	**rs232_config**
Funktion	Konfiguriert die RS-232-Schnittstelle mit der angegebenen Baud-Rate.
Aufruf	`rs232_config( BaudRate )`
Parameter	`BaudRate` Baud-Rate. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [160 Bd…12,8 MBd].
Multi-Board-Befehlsname	**n_rs232_config**
Hinweise	• Siehe auch Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86. • **rs232_config** ist gleichbedeutend zu **uart_config**. Allerdings hat **rs232_config** keine Rückgabe. • < DLL 611: Wertebereich [300 Bd…+115.200 Bd].
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Ab Version DLL 611: erweiteter Wertebereich.
Verweise	**rs232_write_data**, **rs232_read_data**, **uart_config**

Kontrollbefehl	**rs232_read_data**
Funktion	Liest einen Wert aus dem Eingabepuffer der RS-232-Schnittstelle, siehe Kapitel 9.2.3 "RS-232-Schnittstelle", Seite 313.
Aufruf	`RS232Data = rs232_read_data()`
Rückgabe	Als 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) … Bit #7: Nächster (noch nicht vom Anwenderprogramm gelesener) Wert des Eingabepuffers. Bit #8: "Neu"-Bit: = 1: Der Wert ist neu (wurde noch nicht ausgelesen). = 0: Der Wert ist veraltet (wurde bereits einmal mit **rs232_read_data** ausgelesen). Bit #9 … Bit #15: 0. Bit #16 … Bit #23: Anzahl weiterer (noch nicht gelesener) Zeichen. Bit #24 … Bit #31: Anzahl der Pufferüberläufe.
Multi-Board-Befehlsname	**n_rs232_read_data**
Hinweise	• Die RS-232-Schnittstelle wird intern asynchron eingelesen (je ein Zeichen). Wenn dieses Zeichen neu ist, wird es in einem 256 Zeichen großen Ringpuffer abgelegt. Daraus kann es mit **rs232_read_data** (asynchron zum Einlesen) zum Anwenderprogramm übertragen werden. • Byte #0 (Bit #0…Bit #7) gibt jeweils nur ein Zeichen von der aktuellen Leseposition zurück. • Byte #1 (Bit #8) gibt an, ob das Zeichen bereits einmal vom Anwenderprogramm gelesen wurde oder nicht. • Byte #2 (Bit #16…Bit #23) gibt an, wie viele vom Anwenderprogramm noch ungelesene Zeichen im Eingabepuffer stehen. Sind keine ungelesenen Zeichen mehr vorhanden (Byte #2 = 0), wird das zuletzt empfangene Zeichen mit "Neu"-Bit = 0 übertragen. • Byte #3 (Bit #24…Bit #31) gibt an, wie oft der interne Puffer bereits übergelaufen ist (entsprechend viele Zeichen sind überschrieben worden und damit endgültig verloren). *Wichtig: Um den Überlaufzähler zurückzusetzen, müssen Sie* **rs232_read_data** *entsprechend oft aufrufen!* • Beispiel: Rückgabewert 459098 = 0x0007015A = (0, 7, 1, 90) bedeutet: Zeichen 90 ('Z') wurde gelesen, ist neu (1), 7 weitere Zeichen sind noch ungelesen, der Puffer war nicht übergelaufen (0).

Kontrollbefehl	rs232_read_data
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**rs232_write_data**, **uart_config**

Kontrollbefehl	rs232_write_data
Funktion	Sendet ein Datenwort (Byte) an die RS-232-Schnittstelle.
Aufruf	`rs232_write_data( Data )`
Parameter	`Data` Datenwort. Als 32-Bit-Wert ohne Vorzeichen. Nur das niederwertigste Byte wird an die RS-232-Schnittstelle übergeben.
Multi-Board-Befehlsname	**n_rs232_write_data**
Hinweise	• Das vollständige Senden eines etwaigen vorherigen Datenworts wird immer abgewartet. Ein Überrennen der Schnittstelle ist nicht möglich. • Siehe auch Kapitel 9.1.6 "RS-232-Schnittstelle", Seite 312.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**rs232_write_text**, **rs232_read_data**, **uart_config**, **rs232_write_text_list**

Kontrollbefehl	rs232_write_text
Funktion	Sendet einen Text (zeichenweise) an die RS-232-Schnittstelle.
Aufruf	`rs232_write_text( pData )`
Parameter	`pData` PC-Speicheradresse des ersten Zeichens (Byte) des auszugebenden Textes. Als Zeiger auf einen nullterminierten String.
Multi-Board-Befehlsname	**n_rs232_write_text**
Hinweise	• **rs232_write_text** wird in eine entsprechende Folge von **rs232_write_data** zerlegt. • Während der Ausführung von **rs232_write_text** können keine weiteren Kontrollbefehle ausgeführt werden, die Listenausführung auf der Karte wird davon jedoch nicht beeinflusst. Während der Ausführung einer Liste, die auch **rs232_write_text_list** enthält, sollte **rs232_write_text** möglichst nicht verwendet werden, um Konflikte zu vermeiden (race condition). • Siehe auch Kapitel 9.1.6 "RS-232-Schnittstelle", Seite 312.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**rs232_write_data**, **rs232_write_text_list**, **uart_config**

Variabler Listenbefehl	**rs232_write_text_list**
Funktion	Sendet einen Text (zeichenweise) an die RS-232-Schnittstelle.
Aufruf	`rs232_write_text_list( pData )`
Parameter	`pData` PC-Speicheradresse des ersten Zeichens (Byte) des auszugebenden Textes. Als Zeiger auf einen nullterminierten String.
Multi-Board-Befehlsname	**n_rs232_write_text_list**
Hinweise	• Beim Laden von **rs232_write_text_list** wird der auszugebende Text, wenn er aus mehr als 12 Zeichen besteht (\0 nicht mitgezählt), in Blöcke zu je 12 Zeichen zerteilt und für jeden Block ein separater **rs232_write_text_list** in den RTC6-Listenspeicher geschrieben (Achten Sie darauf, dass es dabei zu keinem unerwünschten Speicherüberlauf des jeweiligen RTC6-Listenspeicher-Bereichs kommt). Das Abarbeiten der einzelnen **rs232_write_text_list** erfolgt dann ähnlich wie das Abarbeiten von **rs232_write_text** (der 12er-Block wird in einzelne Zeichen zerlegt und die einzelnen Zeichen der Reihe nach an die RS-232-Schnittstelle gesendet). • **rs232_write_text_list** nimmt je nach eingestellter Baud-Rate und Länge des Textes eine gewisse Zeit in Anspruch. • Siehe auch Kapitel 9.1.6 "RS-232-Schnittstelle", Seite 312.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**rs232_write_text**, **rs232_read_data**, **uart_config**

Kontrollbefehl	rtc6_count_cards
Funktion	Gibt die Zahl der bei der Initialisierung gefundenen RTC6 PCIe-Karten zurück.
Aufruf	`NoOfCards = rtc6_count_cards()`
Rückgabe	Zahl der RTC6 PCIe-Karten. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Die Initialisierung der installierten RTC6 PCIe-Karten wird für jedes Anwenderprogramm separat mit **init_rtc6_dll** aufgerufen. • **rtc6_count_cards** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6 PCIe-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **rtc6_count_cards** weder erzeugt noch verändert.
RTC4→RTC6	Neuer Befehl. Funktionalität ist analog zu **rtc4_count_cards** aus dem RTC4-Befehlssatz.
RTC5→RTC6	Neuer Befehl. Funktionalität ist analog zu **RTC5-Kontrollbefehl** aus dem RTC5-Befehlssatz.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**init_rtc6_dll**

Verzögerter Kurzer Listenbefehl	**save_and_restart_timer**
Funktion	Speichert den aktuellen Wert des RTC6-Timers und setzt ihn auf Null zurück.
Aufruf	`save_and_restart_timer()`
Multi-Board-Befehlsname	**n_save_and_restart_timer**
Hinweise	• Der gespeicherte RTC6-Timer-Wert kann mit **get_time** ausgelesen werden. • **save_and_restart_timer** ist nützlich bei der Messung der Ausführungszeit eines Markierungsprozesses, siehe Kapitel 8.12 "Zeitmessungen", Seite 293. • Der RTC6-Timer zählt nur Listenbefehlstakte. Während Unterbrechungen durch **set_wait** oder **pause_list** bleibt die Zählung stehen. • Mit **get_lap_time** kann abgefragt werden, wie viele Listenbefehlstakte seit dem Zurücksetzen auf Null vergangen sind. • Falls die RTC6-interne **save_and_restart_timer**-Zeitmessung mit einer externen Zeitmessung über den BUSY-Pin verglichen werden soll, dann sollte zwischen **save_and_restart_timer** und **set_end_of_list** ein **list_nop** eingefügt werden, um ein Scanner-Delay noch vor **set_end_of_list** abzuarbeiten. Ohne **list_nop** misst **save_and_restart_timer** das Scanner-Delay mit, obwohl es erst nach **set_end_of_list** abgearbeitet wird (der BUSY-Pin also schon low ist). • Zusätzlich zum RTC6-Timer stehen auch noch ein 32-Bit-"Zeitstempel-Zähler" und 64-Bit-"Zeitstempel-Zähler" zur Verfügung, siehe auch Kapitel 8.12 "Zeitmessungen", Seite 293.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_lap_time**, **get_time**

Kontrollbefehl	**save_disk**
Funktion	Speichert indizierte Zeichen, Zeichenfolgen und/oder Unterprogramme in Indexreihenfolge in eine binäre Datei auf einen PC-Datenträger und liefert die Anzahl der dabei gespeicherten Listenbefehle zurück.
Aufruf	`NoOfSavedCommands = save_disk( Name, Mode )`
Parameter	`Name` Dateiname. Als Zeiger auf einen nullterminierten ANSI-String.
	`Mode` Gibt an, was gespeichert werden soll: Bit #0 = 1: Alle indizierten Zeichen und Zeichenfolgen werden gespeichert. Bit #1 = 1: Alle indizierten Unterprogramme werden gespeichert. Bit #2…31: Werden nicht berücksichtigt. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Anzahl der mit **save_disk** gespeicherten Listenbefehle. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_save_disk**
Hinweise	• **save_disk** kann zusammen mit **load_disk** u.a. zur Defragmentierung und zum nachträglichen Schützen von Unterprogrammen verwendet werden, siehe Abschnitt "Nachträglich nichtindizierte Unterprogramme schützen und konvertieren", Seite 119 und Abschnitt "Indexverwaltung und Defragmentierung", Seite 118. • Durch **save_disk** werden immer die kompletten Sätze (keine einzelnen Zeichen, Zeichenfolgen oder Unterprogramme!) in der angegebenen Datei gespeichert. Indizierte Zeichen, Zeichenfolgen oder Unterprogramme, die mit **copy_dst_src** mehrfach referenziert wurden, werden durch **save_disk** entsprechend auch mehrfach gespeichert. Nicht (nachträglich mit **set_char_pointer**, …) referenzierte nichtindizierte Zeichen, Zeichenfolgen und Unterprogramme werden von **save_disk** nicht berücksichtigt. • Die Anzahl der mit **save_disk** gespeicherten Listenbefehle kann von der Anzahl der im geschützten RTC6-Listenspeicher-Bereich "Liste 3" gespeicherten Listenbefehle (= 2^{23} – `Mem1` – `Mem2` – **get_list_space**) abweichen, weil: – Indizierte Zeichen/Zeichenfolgen/Unterprogramme sind mehrfach referenziert – Indizierte Zeichen/Zeichenfolgen/Unterprogramme befinden sich im nichtgeschützten RTC6-Listenspeicher-Bereich "Liste 1" oder "Liste 2" Vergleichen Sie vor einem anschließenden **load_disk** unbedingt die zurückgegebene Anzahl mit der Größe des geschützten RTC6-Listenspeicher-Bereichs "Liste 3" (= 2^{23} – `Mem1` – `Mem2`). • Nicht mehr benötigte Zeichen (oder Zeichenfolgen oder Unterprogramme) sollten vor **save_disk** mit **load_char** (oder **load_text_table** oder **load_sub**) und unmittelbar nachfolgendem **list_return** entreferenziert werden, siehe Abschnitt "Indexverwaltung und Defragmentierung", Seite 118.

Kontrollbefehl	save_disk
Hinweise (Forts.)	• **save_disk** speichert alle Zeichen, Zeichenfolgen und/oder Unterprogramme jeweils ab der referenzierten Adresse bis zum nächstmöglichen **list_return**. Sprung-Befehle (also auch Verzweigungen zu unterschiedlichen **list_return**-Befehlen) werden dabei weder ausgewertet noch ausgeführt. • Unzulässige Befehle (z.B. **set_end_of_list**) werden beim Speichern durch **save_disk** automatisch durch **list_nop**-Befehle ersetzt, fehlende Befehle (z.B. **list_return** beim Erreichen des letzten Speicherplatzes von "Liste 3") werden ergänzt. • **save_disk** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`), wenn: – `Mode` = 0 – `Name` = NULL • **save_disk** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **save_disk** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Laufzeit von **save_disk** werden: – Keine weiteren Befehle ausgeführt – Externe Starts unterdrückt • **save_disk** speichert auch eine Versionsinformation, die durch **load_disk** überprüft wird.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 615: Versionsinfo.
Verweise	**load_disk**

Unverzögerter Kurzer Listenbefehl	**select_char_set**
Funktion	Wählt einen der zur Verfügung stehenden Zeichensätze (ausschließlich) für nachfolgend ausgeführte **mark_text** und **mark_text_abs** aus.
Aufruf	`select_char_set( No )`
Parameter	`No` Nummer des gewünschten Zeichensatzes. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…3].
Multi-Board-Befehlsname	**n_select_char_set**
Hinweise	• Die Auswahl bleibt so lange gültig, bis eine andere getroffen wird. • Default-Wert (nach Initialisierung) ist `No` = 0. • Für `No` > 3 wird **select_char_set** durch **list_nop** ersetzt. Der bisherige Zeichensatz bleibt dann gültig (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Ein Wechsel des Zeichensatzes *innerhalb* eines **mark_text** oder **mark_text_abs** ist nur möglich, wenn **select_char_set** innerhalb eines (aufgerufenen) indizierten Zeichens steht. Die dort getroffene Auswahl gilt dann für alle nachfolgenden Zeichen. • Ist der gewählte Zeichensatz nicht (vollständig) definiert, werden fehlende Zeichen (durch **mark_text** oder **mark_text_abs**) nicht markiert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_text**, **mark_text_abs**

Kontrollbefehl	**select_cor_table**
Funktion	Weist die zuvor geladenen Korrekturtabellen den Scan-Kopf-Steuerungs-Ports zu und aktiviert die Bildfeld-Korrektur.
Aufruf	`select_cor_table( HeadA, HeadB )`
Parameter	`HeadA` = 0: Schaltet die Signale für Scan-Kopf A (Anschluss für ersten Scan-Kopf) ab. = 1…8: Weist Korrekturtabelle Nummer `HeadA` Scan-Kopf A zu. Als 32-Bit-Wert ohne Vorzeichen. Siehe auch **number_of_correction_tables**.
	`HeadB` = 0: Schaltet die Signale an Scan-Kopf B (Anschluss für zweiten Scan-Kopf) ab. = 1…8: Weist Korrekturtabelle Nummer `HeadB` Scan-Kopf B zu. Erfordert Option “Zweite-Scan-Kopf-Steuerung”. Als 32-Bit-Wert ohne Vorzeichen. Siehe auch **number_of_correction_tables**.
Multi-Board-Befehlsname	**n_select_cor_table**
Hinweise	• **select_cor_table** oder **select_cor_table_list** sollte unmittelbar nach dem Laden der gewünschten Korrekturtabelle(n) mit **load_correction_file** und/oder **load_z_table**/**load_z_table_no** (oder nach einem anschließenden **load_program_file** – siehe unten) aufgerufen werden, um die Korrekturtabelle(n) den jeweiligen Scan-Kopf-Anschluss zuzuweisen (**select_cor_table** wird automatisch von **load_correction_file** aufgerufen; siehe Abschnitt “Hinweise”, Seite 184). Damit sich die Bildfeld-Korrektur auch unmittelbar auf die aktuell eingestellte Galvanometerscanner-Positionen auswirkt, lösen **select_cor_table** und **select_cor_table_list** zusätzlich einen Sprung mit Sprunggeschwindigkeit (kein “Harter Sprung”) auf die korrigierte Galvanometerscanner-Position aus. **select_cor_table** tut dies automatisch und sofort, **select_cor_table_list** fügt den Sprung vor dem nächsten Listenbefehl ein. Je nach Tabelleninhalt und Galvanometerscanner-Position kann dies einige Taktzyklen (mindestens jedoch einen Taktzyklus) dauern. • Korrekturtabellen sollten geladen und zugewiesen werden, bevor erstmalig eine Liste gestartet wird oder mit einem Kontrollbefehl (wie **goto_xy**) die Galvanometerscanner des Scan-Systems bewegt werden, und **select_cor_table** oder **select_cor_table_list** sollten erst *nach* dem Laden der gewünschten Korrekturtabelle(n) gestartet werden. Sonst werden die Galvanometerscanner u.U. in eine unerwartete Position gefahren und der Laserstrahl in eine unbeabsichtigte Richtung gelenkt. • Korrekturtabellen, die vor **load_program_file** mit **load_correction_file** geladen werden, werden erst nach **select_cor_table** oder **select_cor_table_list** voll wirksam. • **select_cor_table** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist Der Listenbefehl **select_cor_table_list** kann uneingeschränkt verwendet werden. • **select_cor_table** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt)

Kontrollbefehl	select_cor_table
Hinweise (Forts.)	• Wenn der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist, wird **select_cor_table** erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Listenausführungsstatus wieder zurückgesetzt wurde). • Für die Dauer des Sprungs auf die korrigierte Galvanometerscanner-Position setzt **select_cor_table** den INTERNAL-BUSY-Listenausführungsstatus. • Bei ungültigen Werten für die Parameter `HeadA` oder `HeadB` (Werte > **number_of_correction_tables**) und beim Wert 0 schaltet **select_cor_table** die entsprechenden Scan-Kopf-Signale ab. Die Galvanometerscanner bleiben dann in ihrer letzten Position stehen. • Falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist, weist **select_cor_table** dem Anschluss für zweiten Scan-Kopf keine Korrekturtabelle zu. • Die Standardeinstellung, die mit **load_program_file** wirksam wird, ist `(1,0)`, d. h. Korrekturtabelle #1 wird für Scan-Kopf A verwendet und die Ausgabesignale für Scan-Kopf B sind abgeschaltet (dies entspricht den Parameterwerten `HeadA` = 1 und `HeadB` = 0). Die Galvanometerscanner bleiben allerdings nach **load_program_file** zunächst so lange in der Position (0|0) stehen (auch wenn der Inhalt der Korrekturtabelle anders lautet), bis **select_cor_table** aufgerufen oder die Galvanometerscanner explizit bewegt werden. • Wenn die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist, bleiben die Signale für den zweiten Scan-Kopf dauerhaft ausgeschaltet. Es können jedoch trotzdem mehrere Korrekturtabellen geladen und jederzeit auch für den ersten Scan-Kopf verwendet werden. Dadurch kann man auch bei einem Scan-Kopf schnell zwischen mehreren Korrekturtabellen wechseln, etwa zwischen einer Datei für einen Pilotlaser und einer zweiten für den Arbeitslaser mit unterschiedlicher Wellenlänge, siehe auch Kapitel 8.5 "2D-Scan-Systeme und 3D Scan-Systeme steuern", Seite 248. • In Systemen mit zwei Scan-Köpfen wird üblicherweise Tabelle #1 für Scan-Kopf A benutzt und Tabelle #2 für Scan-Kopf B: `select_cor_table(1,2)`. Es kann aber jederzeit auch eine andere Zuordnung getroffen werden. • Für 3D-Systeme muss die Option "3D" freigeschaltet sein. Dann können mehrere (2D- oder 3D-) Korrekturtabellen geladen werden. Ohne freigeschaltete Option "Zweite-Scan-Kopf-Steuerung" müssen x-Achse und y-Achse an den Anschluss für ersten Scan-Kopf und die z-Achse an den x-Kanal oder y-Kanal des Anschluss für zweiten Scan-Kopf angeschlossen werden. Wenn dem Anschluss für ersten Scan-Kopf eine 3D-Korrekturtabelle zugewiesen wird, werden am Anschluss für ersten Scan-Kopf die korrigierten Signale für einen xy-Scan-Kopf und über beide Kanäle des Anschluss für zweiten Scan-Kopf die korrigierten Signale für die z-Achse ausgegeben. Wenn mehrere RTC6-Karten mit freigeschalteter Option "3D" in einen PC eingebaut sind, dann können auch entsprechend viele 3-Achsen-Systeme gleichzeitig angesteuert werden.

Kontrollbefehl	select_cor_table
Hinweise (Forts.)	• Bei freigeschalteter Option "3D" und freigeschalteter Option "Zweite-Scan-Kopf-Steuerung" entscheidet der Benutzer bei der Zuweisung der Korrekturtabellen darüber, welche Signale (xy oder z) an welchem Anschluss ausgegeben werden (siehe auch Kapitel 4.5.1 "Scan-Kopf-Anschlüsse und Übertragungsprotokoll", Seite 69 und Abschnitt "2D- und 3D-Korrekturdateien", Seite 182). 2D-Korrekturtabellen sollten und 3D-Korrekturtabellen dürfen dabei allerdings ausschließlich dem Anschluss für die xy-Achsen und *nicht* dem Anschluss für die z-Achse zugewiesen werden (z.B. mit `select_cor_table(0,1)` oder `select_cor_table(0,2)` für xy an Anschluss Kopf B und z an Anschluss Kopf A). Da es ansonsten zu unerwarteten Reaktionen des Systems kommen könnte und dem System nicht bekannt ist, wo die xy-Achsen und wo die z-Achse angeschlossen sind, werden die Signale beider Anschlüsse ausgeschaltet, falls beiden Anschlüssen eine Korrekturtabelle zugeordnet wird und (mindestens) eine davon eine 3D-Korrekturtabelle ist, also z.B. für `select_cor_table(1,1)`. • Parameter aus aktuell geladenen Korrekturtabellen können mit **get_table_para**, aus aktuell zugewiesenen Korrekturtabellen mit **get_head_para** ausgelesen werden (siehe auch **load_correction_file**). • **select_cor_table** ist in Boot-Phase 2 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Unveränderte Funktionalität. Ausnahme: Im Speicher der RTC6 können jetzt auch bis zu 8 Korrekturtabellen insgesamt gespeichert werden, siehe Kapitel 8.5.3 "Mehrere Korrekturtabellen verwenden", Seite 253. Jedoch können *nicht* gleichzeitig zwei *3D*-Korrekturtabellen den Scan-Kopf-Anschlüssen zugewiesen werden.
RTC5→RTC6	Veränderte Funktionalität. • Im RTC6-Befehlssatz schaltet **select_cor_table** Zoom und/oder Stretch automatisch ein, wenn vorher eine Tabelle mit **load_stretch_table** oder **load_zoom_correction_file** auf diese Tabellennummer geladen worden ist.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_cor_table_list**, **load_correction_file**, **load_program_file**, **load_stretch_table**, **number_of_correction_tables**

Variabler Listenbefehl	**select_cor_table_list**
Funktion	Wie **select_cor_table**, aber ein Listenbefehl.
Aufruf	`select_cor_table_list( HeadA, HeadB )`
Parameter	`HeadA` Wie **select_cor_table**.
	`HeadB` Wie **select_cor_table**.
Multi-Board-Befehlsname	**n_select_cor_table_list**
Hinweise	• Siehe **select_cor_table**. • **select_cor_table_list** ist zwar ein Kurzer Listenbefehl, der unmittelbar nachfolgende Listenbefehl wird aber durch den dazwischengeschalteten Sprung auf die korrigierte Galvanometerscanner-Position um einige Takte verzögert ausgeführt, abhängig vom Inhalt der zugewiesenen Korrekturtabelle an der momentanen Galvanometerscanner-Position (z.B. (0|0) nach **load_program_file**), mindestens jedoch um 10 μs, auch wenn die Korrekturtabelle keine Korrektur vorsieht.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_cor_table**

Kontrollbefehl	select_rtc
Funktion	Definiert die für ein Anwenderprogramm aktive RTC6-Karte in einem Multi-Board-System. Siehe Kapitel 6.6 "Mehrere RTC6 PCIe-Karten in einem PC verwenden", Seite 126.
Aufruf	`NoOfSelectedCard = select_rtc( CardNo )`
Parameter	`CardNo` RTC6-DLL-interne Nummer der RTC6-Karte. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Der Rückgabewert ist: • `CardNo`, wenn für die angegebene Karte eine Zugriffsberechtigung vorliegt oder wenn die angegebene Karte nicht durch ein anderes Anwenderprogramm belegt ist (im letzteren Fall wird die Zugriffsberechtigung durch **select_rtc** – wie mit **acquire_rtc** – verschafft, s.u.). • Die Nummer der aktiven Karte, wenn `CardNo` die Zahl der bei der Initialisierung gefundenen RTC6-Karten übersteigt, dort keine RTC6 Ethernet-Karte eingetragen ist oder für `CardNo` = 0. • 0 sonst (also insbesondere dann, wenn die angegebene Karte durch ein anderes Anwenderprogramm reserviert ist und die Aktivierung der Karte deswegen oder wegen eines Versionsfehlers nicht erfolgreich durchgeführt werden kann) (**get_last_error**-Returncode `RTC6_ACCESS_DENIED` und gegebenenfalls `RTC6_VERSION_MISMATCH`). Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Die Aktivierung einer Karte für ein Anwenderprogramm wird auch bereits bei der Initialisierung der RTC6-DLL für dieses Anwenderprogramm ausgeführt (siehe **init_rtc6_dll**). **select_rtc** ermöglicht aber zusätzlich jederzeit einen Wechsel der aktiven Karte. Alle **select_rtc** nachfolgenden Befehle (außer Multi-Board-Befehle) des Anwenderprogramms werden an die jeweils aktive Karte weitergeleitet. • Falls die angegebene Karte durch ein anderes Anwenderprogramm reserviert ist, bleibt **select_rtc** wirkungslos (Rückgabewert 0, **get_last_error**-Returncode `RTC6_ACCESS_DENIED`). • Wenn die angegebene Karte keine Zugriffsberechtigung hat und nicht durch ein anderes Anwenderprogramm in Besitz genommen ist, dann wird versucht, sie (wie mit **acquire_rtc**) in Besitz zu nehmen. Beim Akquirieren der Karte wird (wie mit **acquire_rtc**) eine Versionsprüfung durchgeführt. Liegt ein Versionsfehler vor, wird der Zugriff auf die Karte verweigert und **select_rtc** bleibt wirkungslos (Rückgabewert 0, **get_last_error**-Returncode `RTC6_ACCESS_DENIED`|`RTC6_VERSION_MISMATCH`). • Falls die angegebene Karte bereits die aktive Karte für dieses Anwenderprogramm ist, bleibt **select_rtc** wirkungslos (Rückgabewert: `CardNo`). • **select_rtc** bleibt auch wirkungslos, wenn `CardNo` die Zahl der bei der Initialisierung gefundenen RTC6-Karten (siehe **rtc6_count_cards**) übersteigt, dort keine RTC6 Ethernet-Karte eingetragen ist oder wenn `CardNo` = 0 ist (reale Karten zählen ab 1). Rückgabewert ist dann die Nummer der aktiven Karte (s.o.). Mit `select_rtc( 0 )` lässt sich dadurch jederzeit die Nummer der aktiven Karte ermitteln. • **select_rtc** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar.

Kontrollbefehl	select_rtc
RTC4→RTC6	Bei der RTC4 führt **select_rtc** nur die Aktivierung der angegebenen Karte (unbedingt) aus. Bei der RTC6 liefert **select_rtc** zusätzlich einen Rückgabewert und versucht für die angegebene Karte (wie mit **acquire_rtc**) die Zugriffsberechtigung zu erlangen oder bleibt wirkungslos.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Kontrollbefehl	**select_serial_set**
Funktion	Wählt einen Seriennummernsatz für Seriennummern-Kontrollbefehle aus.
Aufruf	`select_serial_set( No )`
Parameter	`No` Nummer des gewünschten Seriennummernsatzes. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…3]. Es werden nur die beiden niederwertigsten Bits ausgewertet.
Multi-Board-Befehlsname	**n_select_serial_set**
Hinweise	• Nach Initialisierung mit **load_program_file** ist Seriennummernsatz 0 ausgewählt. • **select_serial_set** legt *nicht* den Seriennummernsatz für die Seriennummernmarkierung fest. Für diesen Zweck ist der Listenbefehl **select_serial_set_list** vorgesehen. • Zur Verwendung von **select_serial_set** siehe Kapitel 7.5.2 "Seriennummern markieren", Seite 222.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_serial_set_list**, **set_serial_step**, **get_serial**

Unverzögerter Kurzer Listenbefehl	**select_serial_set_list**
Funktion	Wählt einen Seriennummernsatz für Seriennummern-Listenbefehle (also auch für die Seriennummernmarkierung) aus.
Aufruf	`select_serial_set_list( No )`
Parameter	`No` Nummer des gewünschten Seriennummernsatzes. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…3]. Es werden nur die beiden niederwertigsten Bits ausgewertet.
Multi-Board-Befehlsname	**n_select_serial_set_list**
Hinweise	• Nach Initialisierung mit **load_program_file** ist Seriennummernsatz 0 ausgewählt. • Zur Verwendung von **select_serial_set_list** siehe Kapitel 7.5.2 "Seriennummern markieren", Seite 222.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**select_serial_set**, **set_serial_step_list**, **get_list_serial**, **mark_serial**, **mark_serial_abs**

Kontrollbefehl	**send_user_data**
Funktion	Ohne Funktion.
Multi-Board-Befehlsname	**n_send_user_data**
Hinweise	• Dieser Befehl hat zur Zeit keinerlei Wirkung.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**read_user_data**

Kontrollbefehl	**set_angle**
Funktion	Definiert durch Angabe eines Drehwinkels die Drehmatrix M_R für nachfolgende Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236.
Aufruf	`set_angle( HeadNo, Angle, at_once )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. = 1: Die Definition wirkt nur auf den Anschluss für ersten Scan-Kopf. = 2: Die Definition wirkt nur auf den Anschluss für zweiten Scan-Kopf. = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. = 4: Die Definition wirkt auf das virtuelle Bildfeld (siehe auch Hinweise). Es werden nur die 3 niederwertigsten Bits ausgewertet.
	`Angle` Drehwinkel. In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat. Positive Werte entsprechen einer Drehung um den Mittelpunkt des Bildfelds im Gegenuhrzeigersinn.
	`at_once` Legt fest, wann die definierte Transformation wirksam wird. Als 32-Bit-Wert ohne Vorzeichen. Wie **set_matrix**.
Multi-Board-Befehlsname	**n_set_angle**
Hinweise	• Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 236. • Ab DLL 617, OUT 617 gilt für `HeadNo` = 4: Die globalen Koordinatentransformationen sind allgemein verfügbar, auch außerhalb einer "Processing on the fly"-Session.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Letzte Änderung DLL 617, OUT 617: `HeadNo` = 4.
Verweise	**set_angle_list**, **set_offset**, **set_matrix**, **set_scale**

Variabler Listenbefehl	**set_angle_list**
Funktion	Wie **set_angle**, aber ein Listenbefehl.
Aufruf	`set_angle_list( HeadNo, Angle, at_once )`
Parameter	`HeadNo` Wie **set_angle**. Jedoch ist `HeadNo` = 4 nicht verfügbar.
	`Angle` Wie **set_angle**.
	`at_once` Legt fest, wann die definierte Transformation wirksam wird. Als 32-Bit-Wert ohne Vorzeichen. = 0: Die Transformationswerte werden nur gesammelt und zwischengespeichert, aber die Transformation wird nicht ausgeführt, solange dies nicht durch eine andere Koordinatentransformation (z.B. mit einem Listenbefehl mit `at_once` = 1 oder einem entsprechenden Kontrollbefehl) aktiviert wird. = 1: Die Transformation wird zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und noch vor dem nächsten Listenbefehl ausgeführt. Die Signale für den "Laser active"-Betrieb werden gegebenenfalls zuvor ausgeschaltet. = 2: Die Transformationswerte werden (wie bei `at_once` = 0) nur gesammelt und zwischengespeichert, die Transformation wird aber bei der Ausführung des nächsten **jump_abs** oder **jump_rel** (nur wenn keine Liste ausgeführt wird: auch **goto_xy** oder **goto_xyz**) zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewendet. = 3: Wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: Wie `at_once` = 2.
Multi-Board-Befehlsname	**n_set_angle_list**
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_angle**

Kontrollbefehl	set_auto_laser_control	
Funktion	Initialisiert oder deaktiviert die “Automatische Laser-Steuerung“, siehe Kapitel 7.4.9 ““Automatische Laser-Steuerung““, Seite 208.	
Aufruf	`ErrorCode = set_auto_laser_control( Ctrl, Value, Mode, MinValue, MaxValue )`	
Parameter	`Ctrl`	Steuerparameter. Als 32-Bit-Wert ohne Vorzeichen. = 1…7: Definiert, welcher Signalparameter durch die “Automatische Laser-Steuerung“ korrigiert wird. = 1: 12-Bit-Ausgabewert am ANALOG OUT1-Ausgang. = 2: 12-Bit-Ausgabewert am ANALOG OUT2-Ausgang. = 3: Ausgabewert am 8-Bit-Digital-Ausgang. = 4: Pulslänge (`PulseLength`) der Lasersteuersignale LASER1 und LASER2. = 5: Ausgabeperiode (`HalfPeriod`) der Lasersteuersignale LASER1 und LASER2. = 6: Ausgabewert am 16-Bit-Digital-Ausgang. = 7: Wie 5. Jedoch ist der Pulsabstand geometrisch konstant (“Spot Distance Control“). = 0 oder > 7: Deaktiviert die “Automatische Laser-Steuerung“ (für `Ctrl` > 7: **get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	`Value`	Legt den 100%-Wert für den über `Ctrl` ausgewählten Parameter fest. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: Für `Ctrl` = 1/2: 12-Bit-Werte [0…4.095]. Höherwertige Bits werden ignoriert. Für `Ctrl` = 3: 8-Bit-Werte [0…255]. Höherwertige Bits werden ignoriert. Für `Ctrl` = 4: [0…(2^{32}–1)]. 1 Bit entspricht 1/64 μs. Für `Ctrl` = 5: [0…(2^{32}–1)]. 1 Bit entspricht 1/64 μs. Für `Ctrl` = 6: 16-Bit-Werte [0…65.535]. Höherwertige Bits werden ignoriert.
	`Mode`	Für die Geschwindigkeits-abhängige Laser-Steuerung (`Mode` =1…5) kann ausgewählt werden, welche Größe bei der Berechnung der geschwindigkeitsabhängigen Korrektur zu Grunde gelegt wird. Als 32-Bit-Wert ohne Vorzeichen. =1: Sollgeschwindigkeit. =2: Aktuelle Geschwindigkeit (sowie Erweiterungen, siehe unten). =3: Reserviert. =4: Reserviert. =5: Encodergeschwindigkeit (Zählerpulsrate).

Kontrollbefehl	set_auto_laser_control	
Parameter (Forts.)	Mode (Forts.)	Für obigen Mode = 2 sind folgende Erweiterungen in beliebiger Kombination verfügbar: • M = +4 Encodergeschwindigkeits-abhängige Korrektur, siehe Hinweis, Seite 676 • M = +16 Für SCANahead-Systeme • M = +32 Die Winkel-synchrone Galvanometerscanner-Geschwindigkeit wird mit Hilfe der zugeordneten Korrekturtabelle in eine Bildfeld-synchrone Geschwindigkeit umgerechnet, siehe Hinweis, Seite 677 • M = +64 Koordinatentransformationen für den ersten Scan-Kopf werden rückwärtstransformiert, siehe Hinweis, Seite 677 • M = +128 "Spot Distance Control" in Kombination mit Sky-Writing. Der Spotabstand wird auch in einem Polygonzug konstant gehalten, der durch eine Sky-Writing-Bewegung unterbrochen wird. Dann gilt: Mode = 2 + Summe der Erweiterungen M. Für Mode = 0 oder keinen der oben genannten: Deaktiviert die Geschwindigkeits-abhängige Laser-Steuerung oder Encodergeschwindigkeits-abhängige Laser-Steuerung (**get_last_error**-Returncode RTC6_PARAM_ERROR). Die Positions-abhängige Laser-Steuerung bleibt mit gültigem Wert für Ctrl immer wirksam.
	MinValue	Legt die *untere* Bereichsgrenze für den über Ctrl ausgewählten Parameter fest, der durch die automatische Korrektur nicht unterschritten werden kann. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: siehe Value.
	MaxValue	Legt die *obere* Bereichsgrenze für den über Ctrl ausgewählten Parameter fest, der durch die automatische Korrektur nicht überschritten werden kann. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: siehe Value.
Rückgabe	ErrorCode. Als 32-Bit-Wert ohne Vorzeichen. 0 Kein Fehler. 1 Kein erster Scan-Kopf aktiv. 2 Kein *iDRIVE*-Scan-System (siehe Glossareintrag auf Seite 28) aktiv. 3 Ctrl unzulässig. 4 Mode unzulässig. 5 Zugriff verweigert (Karte gesperrt, **get_last_error**-Returncode RTC6_ACCESS_DENIED). 6 Mode = 2-Erweiterung M = +16: Kein SCANahead-System angeschlossen oder aktiv.	
Multi-Board-Befehlsname	**n_set_auto_laser_control**	

Kontrollbefehl	set_auto_laser_control
Hinweise	• Zur Verwendung von **set_auto_laser_control** siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208. • `MinValue` und `MaxValue` werden automatisch vertauscht, wenn `MinValue` > `MaxValue` ist und `MinValue` auf ≤ `Value` sowie `MaxValue` auf ≥ `Value` geclippt. • Die Steuerungsdaten der Geschwindigkeits-abhängige Laser-Steuerung werden ausschließlich aus den Daten des Scan-Systems am Anschluss für ersten Scan-Kopf abgeleitet und dann auch für den Anschluss für zweiten Scan-Kopf verwendet (wenn dieser freigeschaltet ist und dort ein Scan-System angeschlossen ist). Dem Anschluss für ersten Scan-Kopf muss bereits zum Zeitpunkt des **set_auto_laser_control**-Aufrufs eine Korrekturtabelle zugewiesen sein, sonst wird Fehlercode 1 zurückgegeben und `Ctrl` = 0 gesetzt. Wenn nur der Anschluss für zweiten Scan-Kopf verwendet wird und/oder das Scan-System am Anschluss für ersten Scan-Kopf abgeschaltet ist, funktioniert die Geschwindigkeits-abhängige Laser-Steuerung nicht. • Für `Mode` = 2 muss am Anschluss für ersten Scan-Kopf ein funktionsfähiges i*DRIVE*-Scan-System (siehe Glossareintrag auf Seite 28) angeschlossen sein. In der Folge ist **control_command** für den Anschluss für ersten Scan-Kopf gesperrt, solange diese `Mode`-Auswahl besteht (weil die aktuelle vom Scan-System übertragene Geschwindigkeit genutzt wird). Für `Mode` = 0 oder 1 ist **control_command** uneingeschränkt verfügbar. Ist kein funktionsfähiges i*DRIVE*-Scan-System angeschlossen, wird Fehlercode 2 zurückgegeben und `Ctrl` = 0 gesetzt. • Die Encodergeschwindigkeits-abhängige Laser-Steuerung (`Mode` = 5) funktioniert auch dann, wenn zur Laufzeit kein Scan-Kopf an den Scan-Kopf-Anschlüssen angeschlossen ist. Auch die Option "Processing on the fly" muss dafür nicht freigeschaltet sein. Die Soll-Encodergeschwindigkeit wird mit **set_encoder_speed** eingestellt. • **set_auto_laser_control** mit `Mode` = 2-Erweiterung M = +4 rechnet die Encodergeschwindigkeiten mit den Skalierungsfaktoren aus **set_fly_x** und **set_fly_y** oder **set_fly_2d** in Galvanometerscanner-Einheiten (Bits/ms) um und addiert sie zu den aktuellen Galvanometerscanner-Geschwindigkeiten. Die Option "Processing on the fly" muss dazu freigeschaltet sein (im Gegensatz zu `Mode` = 5). Als Referenzgeschwindigkeit wird die aktuelle Markiergeschwindigkeit verwendet. Der für `Mode` = 5 verwendete Befehl **set_encoder_speed** bleibt mit der Erweiterung +4 unberücksichtigt. Ist zur Laufzeit eine Achse nicht für "Processing on the fly" aktiviert, so bleibt deren Encodergeschwindigkeit unberücksichtigt. Sind beide Achsen nicht für "Processing on the fly" aktiviert, wirkt Erweiterung +4 nicht. **set_fly_rot**, **set_fly_rot_pos**, **set_fly_x_pos** und **set_fly_y_pos** können nicht mit der Galvanometerscanner-Geschwindigkeit kombiniert werden. • **set_auto_laser_control** mit `Mode` = 2-Erweiterung M = +4 funktioniert auch, wenn keine "Processing on the fly"-Korrektur aktiv ist. Allerdings müssen zuvor die entsprechenden Skalierungsfaktoren mit einem der **[*]fly[*]**-Befehle (z.B. **set_fly_x**, **set_fly_2d**, …) gesetzt worden sein. Die dadurch aktivierte "Processing on the fly"-Korrektur muss mit **fly_disable_list** deaktiviert werden, damit M = +4 auch nach der "Processing on the fly"-Korrektur weiterhin funktioniert. Siehe Hinweise, Seite 426 zu **fly_disable_list**.

Kontrollbefehl	set_auto_laser_control
Hinweise (Forts.)	• **set_auto_laser_control** mit `Mode` = 2-Erweiterung M = +64 muss aufgerufen werden bevor entsprechende Koordinatentransformationen (z.B. mit **set_scale**, **set_angle**, …) eingestellt werden. Anderenfalls wird die Rückwärtstransformation nicht korrekt ausgeführt. • **set_auto_laser_control** mit `Mode` = 2-Erweiterung M = +128 kann mit **set_sky_writing_min_speed_ctrl** kombiniert werden. Der Energieeintrag wird trotzdem konstant gehalten. • Für `Ctrl` = 1…6 wird der ausgewählte Signalparameter alle 10 μs aktualisiert. Für `Ctrl` = 5 (Ausgabeperiode) ist er jedoch immer erst nach Ablaufen einer bereits begonnenen Lasersteuersignal-Periode wirksam. Für `Ctrl` = 7 wird die Ausgabeperiode dagegen kontinuierlich (von Puls zu Puls) angepasst. • `Ctrl` = 7 ("Spot Distance Control") ist nur für SCANahead-Systeme verfügbar. Damit `Ctrl` = 7 tatsächlich wirken kann, muss mit **spot_distance** oder **spot_distance_ctrl** der geforderte geometrische Pulsabstand angegeben werden. Solange der geometrische Pulsabstand 0 ist (Defaultwert) ist diese Regelung unwirksam, aber nicht deaktiviert. Die tatsächliche Regelgröße ("zeitlicher Pulsabstand") kann nicht mit **set_trigger[*]** und Signal 24 aufgezeichnet werden. Die Befehlsparameter `MinValue`, `MaxValue` und `Value` werden ignoriert. Deshalb kann die tatsächliche Regelgröße ("zeitlicher Pulsabstand") nicht mit `MinValue` und `MaxValue` begrenzt werden. Es wird automatisch auf die aktuelle Markiergeschwindigkeit geregelt. Siehe auch Abschnitt ""Spot Distance Control"", Seite 213. • Ein Aktivieren der Winkel-zu-Bildfeld-Transformation der Galvanometerscanner-Geschwindigkeit (Erweiterung +32) ist insbesondere dann sinnvoll, wenn die Korrekturdatei starke Verzeichnungen ausgleichen muss. • Für einen unzulässigen Wert von `Ctrl` oder `Mode` wird **set_auto_laser_control** nicht ausgeführt (Rückgabewert 3 oder 4, **get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Die "Automatische Laser-Steuerung" sollte nicht zusammen mit der variablen Laserleistung aus **set_multi_mcbsp_in** verwendet werden. • Nur für SCANahead-Systeme: Wenn ein SCANahead-System angeschlossen und aktiv ist, stellen Sie vor dem **set_auto_laser_control**-Aufruf sicher, dass es nicht mehr beschäftigt ist (siehe **get_status** und HEAD BUSY-Listenausführungsstatus). Anderenfalls kann der noch nicht verarbeitete Rest der letzten Markierung gestört werden. • **set_auto_laser_control** ist in Boot-Phase 2 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 für `Ctrl` = 1/2 die Werte für `Value`, `MinValue` und `MaxValue` mit 4. Für `Ctrl` = 4/5 werden die Werte für `Value`, `MinValue` und `MaxValue` mit 8 multipliziert. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Veränderte Funktionalität. Weitere Laser-Steuerungsmodi können eingestellt werden.

Kontrollbefehl	set_auto_laser_control
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Änderung DLL 609, OUT 609, RBF 613: `Ctrl` = 7. Änderung DLL 641, OUT 642: `Mode` = 2-Erweiterung M = +64. Letzte Änderung DLL 646, OUT 649, RBF 641: `Mode` = 2-Erweiterung M = +128.
Verweise	**load_position_control**, **load_auto_laser_control**, **set_auto_laser_params**, **set_auto_laser_params_list**, **set_fly_x**, **set_fly_y**, **set_fly_2d**, **set_encoder_speed**, **spot_distance**, **spot_distance_ctrl**

Kontrollbefehl	**set_auto_laser_params**	
Funktion	Legt den zu steuernden Signalparameter der “Automatischen Laser-Steuerung”, den 100%-Wert und seine ihm zugeordneten Grenzwerte fest.	
Aufruf	`ErrorCode = set_auto_laser_params( Ctrl, Value, MinValue, MaxValue )`	
Parameter	`Ctrl`	Der zu steuernde Signalparameter. Siehe **set_auto_laser_control**. Zulässiger Wertebereich: [1…7].
	`Value`	100%-Wert. Siehe **set_auto_laser_control**.
	`MinValue`	*Untere* Bereichsgrenze. Siehe **set_auto_laser_control**.
	`MaxValue`	*Obere* Bereichsgrenze. Siehe **set_auto_laser_control**.
Rückgabe	`ErrorCode`. Als 32-Bit-Wert ohne Vorzeichen. 0 Kein Fehler. 3 `Ctrl` unzulässig.	
Multi-Board-Befehlsname	**n_set_auto_laser_params**	
Hinweise	• Zur Verwendung von **set_auto_laser_params** siehe Kapitel 7.4.9 ““Automatische Laser-Steuerung””, Seite 208. • Die Bedeutung der Parameter ist identisch zu **set_auto_laser_control** (siehe auch Hinweise dort). Jedoch kann die “Automatische Laser-Steuerung“ mit **set_auto_laser_params** weder aus- noch eingeschaltet werden. • Wenn der Wert von `Ctrl` unzulässig ist (0 oder > 7), dann wird **set_auto_laser_params** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).	
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_auto_laser_control**.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 609, OUT 609, RBF 613: `Ctrl` = 7.	
Verweise	**set_auto_laser_control**, **set_auto_laser_params_list**	

Verzögerter Kurzer Listenbefehl	**set_auto_laser_params_list**
Funktion	Wie **set_auto_laser_params**, aber ein Verzögerter Kurzer Listenbefehl.
Aufruf	`ErrorCode = set_auto_laser_params_list( Ctrl, Value, MinValue, MaxValue )`
Parameter	`Ctrl` Wie **set_auto_laser_params**.
	`Value` Wie **set_auto_laser_params**.
	`MinValue` Wie **set_auto_laser_params**.
	`MaxValue` Wie **set_auto_laser_params**.
Multi-Board-Befehlsname	**n_set_auto_laser_params_list**
Hinweise	• Wenn der Wert von `Ctrl` unzulässig ist (0 oder > 7), dann wird **set_auto_laser_params_list** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_auto_laser_control**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 609, OUT 609, RBF 613: `Ctrl` = 7.
Verweise	**set_auto_laser_params**, **set_auto_laser_control**

Kontrollbefehl	**set_char_pointer**	
Funktion	Speichert die absolute Startadresse einer Befehlsliste in der internen Verwaltungstabelle für indizierte Zeichen.	
Aufruf	`set_char_pointer( Char, Pos )`	
Parameter	`Char`	Index des indizierten Zeichens, für das die Startadresse `Pos` in die Verwaltungstabelle eingetragen werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023]. Wie bei **load_char** gilt: `Char` = Zeichensatznummer × 256 + ASCII-Nummer des Zeichens (die Zeichensatznummer zählt dabei von 0 bis 3).
	`Pos`	Absolute Startadresse. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_set_char_pointer**	
Hinweise	• Für `Char` > 1023 und/oder `Pos` > (2^{23}–1) wird **set_char_pointer** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Mit **set_char_pointer** kann ein nichtindiziertes Unterprogramm referenziert werden. Damit wird es zu einem indizierten Zeichen und kann mit **save_disk**/**load_disk** geschützt und/oder über den Index aufgerufen werden. • Mit **set_char_pointer** kann aber auch ein indiziertes Unterprogramm oder Zeichen (oder Zeichenfolge) erneut referenziert werden, so dass es auch über einen zweiten Index aufgerufen werden kann. Zur Indexverwaltung sollte aber bevorzugt **copy_dst_src** verwendet werden. • Die Anfangsadressen von Befehlslisten, die mit **set_char_pointer** referenziert werden sollen, können – *vor* dem Abspeichern der Befehlslisten – mit **get_input_pointer** ausgelesen werden. • **set_char_pointer** speichert nur *Startadressen* in der internen Verwaltungstabelle. Erst bei einem anschließenden **save_disk**/**load_disk** wird das indizierte Zeichen auch zu einer geschützten Funktion. • `Pos` sollte keine beliebige Adresse innerhalb einer Liste, sondern immer die Anfangsadresse eines tatsächlichen Unterprogramms sein, welches mit **list_return** abgeschlossen worden ist und kein **set_end_of_list** enthält.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**load_char**	

Kontrollbefehl	**set_char_table**
Funktion	Speichert die absolute Startadresse einer Befehlsliste in der internen Verwaltungstabelle für indizierte Zeichenfolgen.
Aufruf	`set_char_table( Index, Pos )`
Parameter	`Index` Index der indizierten Zeichenfolge, für die die Startadresse `Pos` in die Verwaltungstabelle eingetragen werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…41]. Es gilt die gleiche Zuordnung wie bei **load_text_table**: = 0…9: Ziffern für die Beschriftung von Uhrzeit und Datum [0…9]. = 10…21: Monate [Januar…Dezember]. = 22…28: Wochentage [Sonntag…Samstag]. = 29: Leerzeichen für die Beschriftung von Seriennummern. = 30…39: Ziffern für die Beschriftung von Seriennummern [0…9]. = 40: Text für “a.m.“. = 41: Text für “p.m.“.
	`Pos` Absolute Startadresse. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_set_char_table**
Hinweise	• **set_char_table** ist gleichbedeutend mit **set_text_table_pointer**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_text_table_pointer**

Kontrollbefehl	set_control_mode		
Funktion	Aktiviert oder deaktiviert die externe Start-Eingabe und setzt den Zähler für externe Starts auf Null.		
Aufruf	`set_control_mode( Mode )`		
Parameter	`Mode`	Als 32-Bit-Wert ohne Vorzeichen.	
	Bit #	Wert	Beschreibung
	Bit #0 (LSB)	= 1:	Die externe Start-Eingabe (über /START, /START2 oder /Slave-START) wird aktiviert.
		= 0:	Die externe Start-Eingabe wird deaktiviert.
	Bit #1	= 1:	Bei einem /STOP, /STOP2, /Slave-STOP oder **simulate_ext_stop** wird die Warteschlange der externen Start-Eingabe (/START, /START2, /Slave-START oder **simulate_ext_start**) explizit gelöscht.
		= 0:	Keine Auswirkung.
	Bit #2	= 1:	Die mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** konfigurierte Streckenverzögerung (track delay) für eine verzögerte Ausführung des Starts relativ zum auslösenden Eingangssignal oder **simulate_ext_start** oder **simulate_ext_start_ctrl** (siehe Abschnitt "Externer Start", Seite 315) wird deaktiviert.
		= 0:	Keine Auswirkung. Um eine Streckenverzögerung (z.B. für "Processing on the fly"-Anwendungen) zu konfigurieren und aktivieren, können **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** verwendet werden.
	Bit #3	= 1:	Die externe Start-Eingabe wird *nicht* durch ein externes Stoppsignal deaktiviert.
		= 0:	Die externe Start-Eingabe *wird* durch ein externes Stoppsignal deaktiviert.
	Bit #4		Deaktiviert **simulate_ext_start_ctrl**.
	Bit #5		Reserviert.
	Bit #6		Reserviert.
	Bit #7		Reserviert.
	Bit #8		Reserviert.
	Bit #9	= 1:	Encoderresets der 2 internen Encoderzähler erfolgen mit: • Einem externen Startsignal • **simulate_ext_start** • **simulate_ext_start_ctrl**, verzögert um eine mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** eingestellte Streckenverzögerung, siehe auch Bit #2
		= 0:	Encoderresets erfolgen unmittelbar mit dem jeweiligen "Processing on the fly"-Befehl.

Kontrollbefehl	set_control_mode		
Parameter (Forts.)	Bit #10	= 1:	Die mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** konfigurierte Streckenverzögerung wird ab dem zuletzt extern oder simuliert (nicht aber mit **execute_list_pos** etc.) ausgelösten externen Start gezählt. Der Abstand aufeinander folgender Externer Starts (in Encoderpulsen) wird dadurch konstant, siehe auch Abschnitt "Regelmäßige (periodische) externe Starts", Seite 318. Bei **stop_execution** oder einem externen Stoppsignal wird Bit #10 auf "0" zurückgesetzt.
		= 0:	Die mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** konfigurierte Streckenverzögerung wird bei einem extern oder simuliert angeforderten externen Start ab dem Anforderungs-Zeitpunkt (**simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl oder externes Startsignal) gezählt. Der Abstand aufeinander folgender Externer Starts (in Encoderpulsen) kann dadurch variieren.
	Bit #11		Reserviert.
	…		…
	Bit #31		Reserviert.
Multi-Board-Befehlsname	**n_set_control_mode**		
Hinweise	• Wenn die Ausführung mit **stop_execution** unterbrochen wird, so werden Bit #0 und Bit #10 auf Null zurückgesetzt, d. h. die externe Start-Eingabe wird deaktiviert und die Zählung der Streckenverzögerung auf das auslösende Ereignis bezogen. • Für Bit #9 = 0 gibt es in der Regel einen kleinen (zufälligen) Zeitversatz (10 µs-Jitter) zwischen dem Startsignal an /START, /START2 und dem tatsächlichen Ausführungsstart. Mit Bit #9 = 1 wird dieser 10 µs-Jitter ausgeschaltet, weil dann der Encoderreset exakt mit dem Startsignal erfolgt. • Bit #9 = 0 ist sinnvoll, wenn innerhalb einer Liste mehrere separate "Processing on the fly"-Sessions gestartet werden sollen. • Siehe auch Abschnitt "Externer Stopp", Seite 314 und Abschnitt "Externer Start", Seite 315. • **set_control_mode** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.		

Kontrollbefehl	set_control_mode
RTC4→RTC6	Bit #8 kann nicht dafür verwendet werden, eine Sperrung oder Entsperrung für die oberen 8 Bits des 16-Bit-Digital-Ausgangs (für I/O-Befehle) vorzunehmen. Die RTC6 reserviert diese Bits für die vario*SCAN FLEX*-Ansteuerung automatisch durch **move_to**. Eine explizite Entsperrung kann nicht vorgenommen werden (dies erfolgt nur automatisch durch **load_program_file**). Ansonsten unveränderte Funktionalität für bereits bei der RTC4 verwendete Bits.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 605, OUT 605: Bit #4.
Verweise	**set_control_mode_list**, **set_extstartpos**, **get_counts**, **set_max_counts**, **get_startstop_info**, **move_to**

Normaler Listenbefehl	**set_control_mode_list**
Funktion	Wie **set_control_mode**, aber ein Listenbefehl.
Aufruf	`set_control_mode_list( Mode )`
Parameter	`Mode` Wie **set_control_mode**.
Multi-Board-Befehlsname	**n_set_control_mode_list**
Hinweise	• Der Zähler für externe Starts wird durch **set_control_mode_list** *nicht* zurückgesetzt.
RTC4→RTC6	Unveränderte Funktionalität für bereits bei der RTC4 verwendete Bits.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 605, OUT 605: Bit #4.
Verweise	**set_control_mode**

Kontrollbefehl	**set_controlpreview_compensation_ctrl**	
Funktion	Nur für Vorverarbeitendes System. Gleicht die `ControlPreview`-Zeit durch verzögertes Ausgeben der Lasersteuersignale aus.	
Aufruf	`set_controlpreview_compensation_ctrl( ControlPreview, Mode )`	
Parameter	`ControlPreview`	Verzögerung der Lasersteuersignal-Ausgabe. In 1/64 µs. Als 32-Bit-Wert ohne Vorzeichen.
	`Mode`	Reserviert.
Multi-Board-Befehlsname	**n_set_controlpreview_compensation_ctrl**	
Hinweise	• **set_controlpreview_compensation_ctrl** wird ausschließlich zum Ansteuern von Scan-Systemen bereitgestellt, für die eine `ControlPreview`-Zeit spezifiziert ist (z.B. *SCAN*cube IV), siehe Vorverarbeitendes System und entsprechendes Scan-System-Handbuch. • Bei aktiver SCANahead-Funktionalität hat `ControlPreview` keine Wirkung. • **set_controlpreview_compensation_ctrl** aktualisiert auch die "Processing on the fly"Gesamt-Prädiktionswerte ein, siehe **set_fly_tracking_error**. • **set_controlpreview_compensation_ctrl** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 634, OUT 635. Letzte Änderung DLL 646, OUT 649: Aktualisiert "Processing on the fly"Gesamt-Prädiktionswerte.	
Verweise	**set_fly_tracking_error**, **fly_prediction**	

Kontrollbefehl	**set_default_pixel**
Funktion	Legt den Pulslängen-Default-Wert für das Default-Pixel fest, das den Pixelausgabemodus abschließt.
Aufruf	`set_default_pixel( PulseLength )`
Parameter	`PulseLength` Länge des Default-Pixels. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_set_default_pixel**
Hinweise	• Im Pixelausgabemodus wird am Ende der Bildzeile (zu Beginn des Default-Pixels) die Pixelpulslänge auf den angegebenen Default-Wert gesetzt, siehe Kapitel 8.7.4 "Synchronisation", Seite 280. • Bei der Initialisierung (mit **load_program_file**) wird der `PulseLength`-Default-Wert auf 0 gesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_port_default**

Unverzögerter Kurzer Listenbefehl	**set_default_pixel_list**
Funktion	Wie **set_default_pixel**, aber ein Listenbefehl.
Aufruf	`set_default_pixel_list( PulseLength )`
Parameter	`PulseLength` Wie **set_default_pixel**.
Multi-Board-Befehlsname	**n_set_default_pixel_list**
Hinweise	• Siehe **set_default_pixel**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_default_pixel**, **set_port_default**

Kontrollbefehl	**set_defocus**
Funktion	Legt eine Fokusverschiebung für 3D-Ausgaben fest.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), hat **set_defocus** keine Auswirkung. Der angegebene Wert für die Fokusverschiebung bleibt aber intern gespeichert und wird wirksam, sobald eine 3D-Korrekturtabelle zugewiesen wird.
Aufruf	`set_defocus( Shift )`
Parameter	`Shift` Fokusverschiebung. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf den Randwert geclippt.
Multi-Board-Befehlsname	**n_set_defocus**
Hinweise	• Eine Fokusverschiebung bewirkt eine Defokussierung des Laserfokus relativ zur Arbeitsebene. Ein positiver Wert vergrößert die Fokuslänge der dynamischen Fokussiereinheit (siehe Glossareintrag, Seite 28) und verschiebt die Fokuslage für z.B. den Ansteuerwert (0|0|0) um etwa d = `Shift` / *K* in die Ebene z = –d [mm]. Zum Kalibrierfaktor *K* siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175; außerdem auch #8 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189. • Durch die Fokusverschiebung kann sich u.U. ein so großer Ausgabewert für die z-Achse ergeben, dass die z-Achse bis an den Anschlag fährt. Stellen Sie sicher, dass dies möglichst vermieden wird, siehe **get_galvo_controls**. • Wird **set_defocus** während einer Vektorausgabe aufgerufen, wird er erst vor dem nächsten Listenbefehl ausgeführt. Um harte Sprünge (siehe "Harter Sprung") zu vermeiden, wird ein Sprung mit Sprunggeschwindigkeit auf die geänderte z-Ausgabe ausgeführt. Länge und Dauer des Sprungs sind abhängig vom geänderten z-Ansteuerwert. Die Dauer kann mit **get_galvo_controls** berechnet werden. Wenn gerade *keine* Liste BUSY-Listenausführungsstatus ist, wird der Sprung sofort ausgeführt, wodurch beim nächsten Start keine Verzögerung entsteht. • Wenn der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist, wird **set_defocus** erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Listenausführungsstatus wieder zurückgesetzt wurde). • Für die Dauer des Sprungs auf die geänderte z-Ausgabe setzt **set_defocus** den INTERNAL-BUSY-Listenausführungsstatus. • Nach Aktivierung der Vektorkontrollierte Laser-Steuerung mit **set_vector_control**(`Ctrl` = 7) wird die Fokusverschiebung auch bei parametrisierten [*]mark[*]-Befehlen oder Sprung-Befehlen verändert.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Der RTC6-Befehl vermeidet aber harte Sprünge (siehe "Harter Sprung").
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_defocus_list**, **set_defocus_2_list**, **set_offset_xyz**, **set_defocus_offset**, **get_galvo_controls**

Variabler Listenbefehl	**set_defocus_list**
Funktion	Wie **set_defocus**, aber ein Listenbefehl.
Einschränkung	Wie **set_defocus**.
Aufruf	`set_defocus_list( Shift )`
Parameter	`Shift` Wie **set_defocus**.
Multi-Board-Befehlsname	**n_set_defocus_list**
Hinweise	• Siehe **set_defocus**. • **set_defocus_list** ist zwar ein Kurzer Listenbefehl, der unmittelbar nachfolgende Listenbefehl wird aber durch den dazwischengeschalteten Sprung auf die geänderte z-Ausgabe um einige 10 μs-Takte verzögert ausgeführt, abhängig von der Größe der angegebenen Fokusverschiebung, mindestens jedoch um 10 μs, auch für `Shift` = 0. • Nach dem Sprung auf die geänderte z-Ausgabe wird ein Jump-Delay, das zuvor mit **set_scanner_delays** konfiguriert wurde, eingefügt. Je nach Dynamik der z-Achse kann es sinnvoll sein, das Jump-Delay vor dem Aufruf von **set_defocus_list** zu erhöhen. Im Falle eines SCANahead-Systems mit automatischer Delay-Berechnung kann alternativ dazu **long_delay** verwendet werden, um den Schleppverzug der z-Achse auszugleichen.
RTC4→RTC6	Siehe **set_defocus**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_defocus**, **set_defocus_2_list**, **long_delay**

Unverzögerter Kurzer Listenbefehl	**set_defocus_2_list**
Funktion	Wie **set_defocus_list**, aber mit zusätzlichem Parameter `Mode`.
Einschränkung	Wie **set_defocus_list**.
Aufruf	`set_defocus_2_list( Shift, Mode )`
Parameter	`Shift` Wie **set_defocus_list**.
	`Mode` Modus. Als 32-Bit-Wert mit Vorzeichen.
	= 0 `Shift` wird sofort angewendet ("Harter Sprung").
	> 0 Reserviert.
Multi-Board-Befehlsname	**n_set_defocus_2_list**
Hinweise	• **set_defocus_2_list**(`Mode` = 0) führt – anders als **set_defocus_list** – noch im selben 10 μs-Takt einen harten Sprung (siehe "Harter Sprung") auf die geänderte z-Ausgabe aus. • Siehe **set_defocus_list**. • Für `Mode` > 0 wird **set_defocus_2_list** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 646, OUT 649.
Verweise	**set_defocus_list**

Kontrollbefehl	**set_defocus_offset**
Funktion	Legt einen Offset zusätzlich zur Fokusverschiebung (siehe **set_defocus**) für alle 3D-Ausgaben fest.
Einschränkung	Wie **set_defocus**.
Aufruf	`set_defocus_offset( Shift )`
Parameter	`Shift` Wie **set_defocus**.
Multi-Board-Befehlsname	**n_set_defocus_offset**
Hinweise	• **set_defocus_offset** wirkt wie **set_defocus**. • **set_defocus_offset** dient dazu, einen globalen Defokus-Shift einzustellen, der nicht durch parametrisierte Befehle wie **para_mark_abs** überschrieben wird, siehe **set_vector_control**(`Ctrl` = 7). • Der **set_defocus_offset**-`Shift`: – Wirkt additiv zu **set_defocus** – Kann nicht als Steuerparameter für die Vektorkontrollierte Laser-Steuerung verwendet werden – Kann nicht über Signal 32 mit **set_trigger[*]** aufgezeichnet werden – Wird bei **get_z_distance**, **get_galvo_controls** und bei der Rückwärtstransformation (**transform**, **get_transform**/**get_transform_offset**) berücksichtigt
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Shift` mit 16.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Shift` mit 16.
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 614.
Verweise	**set_defocus**, **set_defocus_offset_list**

Variabler Listenbefehl	**set_defocus_offset_list**
Funktion	Wie **set_defocus_offset**, aber ein Listenbefehl.
Einschränkung	Wie **set_defocus**.
Aufruf	`set_defocus_offset_list( Shift )`
Parameter	`Shift` Wie **set_defocus**.
Multi-Board-Befehlsname	**n_set_defocus_offset_list**
Hinweise	• Siehe **set_defocus_offset**. • Siehe **set_defocus_list**. • **set_defocus_offset_list** ist zwar ein Kurzer Listenbefehl, der unmittelbar nachfolgende Listenbefehl wird aber durch den dazwischengeschalteten Sprung auf die geänderte z-Position um einige Takte verzögert ausgeführt, abhängig von der Größe der angegebenen Fokusverschiebung, mindestens jedoch um 10 μs, auch für `Shift`= 0.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Shift` mit 16.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Shift` mit 16.
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 614.
Verweise	**set_defocus_offset**, **set_defocus_list**

Kontrollbefehl	set_delay_mode		
Funktion	Schaltet den “Variable Polygon-Delays”-Modus und den “Variable Jump-Delays”-Modus ein oder aus und setzt einige spezielle Scanner-Delay-bezogene Parameter sowie den 3D-Z-Move-Modus.		
Aufruf	`set_delay_mode( VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit )`		
Parameter	Name	Zulässige Werte	Beschreibung
	`VarPoly`	> 0	Aktiviert den “Variable Polygon-Delays”-Modus.
		= 0	Deaktiviert den “Variable Polygon-Delays”-Modus. Standardeinstellung.
			Als 32-Bit-Wert ohne Vorzeichen.
	`DirectMove3D`		Dieser Parameter beeinflusst nur 3D-Anwendungen.
		> 0	Die z-Ausgabe wird während eines Sprungs direkt (linear) zu ihrem Endwert geführt.
		= 0	Die z-Ausgabe wird schrittweise so variiert, dass der Fokus während des ganzen Sprungs in einer Ebene bleibt (wie bei einem Mark-Befehl).
			Als 32-Bit-Wert ohne Vorzeichen.
	`EdgeLevel`	0…(2^{32}–1). 1 Bit entspricht 10 μs.	Dieser Parameter definiert für die Ecken eines Polygonzugs eine maximale “Laser On”-Zeit. Wenn das Polygon-Delay länger als oder gleich diesem Wert ist (z.B. weil der Winkel φ ungefähr 180° ist), wird der Laser (nach einem LaserOff-Delay) ausgeschaltet und ein neuer Polygonzug gestartet. Dadurch können Einbrenn-Effekte verhindert werden. Der `EdgeLevel`-Wert muss kleiner sein als das Zweifache des eingestellten Wertes für das Polygon-Delay, sonst hat er keine Wirkung, siehe auch Abbildung 40.
			Hinweis: Um dieses Feature zu deaktivieren, ist der `EdgeLevel`-Wert auf (2^{32}–1) (Standardwert) zu setzen.
			Als 32-Bit-Wert ohne Vorzeichen.
	`MinJumpDelay`	0…(2^{32}–1). 1 Bit entspricht 10 μs.	Minimales Jump-Delay, das für Sprünge kleiner als `JumpLengthLimit` auch für Sprung-Vektoren der Länge Null nicht unterschritten werden kann, siehe Abbildung 36. Bei Sprüngen größer als `JumpLengthLimit` bleibt `MinJumpDelay` ohne Bedeutung. Um Anomalien im Bereich von `MinJumpDelay` zu vermeiden, sollte `MinJumpDelay` nicht größer als das Jump-Delay gewählt werden.
			Als 32-Bit-Wert ohne Vorzeichen.

Kontrollbefehl	**set_delay_mode**	
	`JumpLengthLimit` 0…(2^{32}–1)	Sprunglängengrenze. In Bits. `JumpLengthLimit` > 0 schaltet den variablen Jump-Delay-Modus ein. Wenn der Sprung-Vektor *länger* ist als dieser Wert, dann wird das konstante Jump-Delay (siehe **set_scanner_delays**) eingefügt. Für alle kleineren Sprunglängen wird ein zwischen `MinJumpDelay` und dem Jump-Delay linear interpoliertes "Variables Jump-Delay" (Seite 151) berechnet und eingefügt, siehe Abbildung 36. `JumpLengthLimit` = 0 schaltet den "Variable Jump-Delays"-Modus aus. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_delay_mode**	
Hinweise	• –	
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: erhöhte Wertebereiche. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `JumpLengthLimit` mit 16. Entsprechend verringert sich der zulässige Wertebereich.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**set_scanner_delays**, **load_varpolydelay**, **set_delay_mode_list**	

Multipler Listenbefehl	**set_delay_mode_list**
Funktion	Wie **set_delay_mode**, aber ein Listenbefehl.
Aufruf	`set_delay_mode_list( VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit )`
Parameter	`VarPoly` Wie **set_delay_mode**.
	`DirectMove3D` Wie **set_delay_mode**.
	`EdgeLevel` Wie **set_delay_mode**.
	`MinJumpDelay` Wie **set_delay_mode**.
	`JumpLengthLimit` Wie **set_delay_mode**.
Multi-Board-Befehlsname	**n_set_delay_mode_list**
Hinweise	• Siehe **set_delay_mode**. • **set_delay_mode_list** belegt 2 RTC6-Listenspeicherplätze. • **set_delay_mode_list** wird als 2 Unverzögerte Kurze Listenbefehle ausgeführt. • Ein noch anliegender Verzögerter Kurzer Listenbefehl wird noch zuvor ausgeführt.
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_delay_mode**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_delay_mode**

Kontrollbefehl	**set_dsp_mode**
Funktion	Nur für einige abgekündigte Systeme. Setzt einen DSP-Modus für die Kurzbefehlszählung.
Aufruf	`dsp_mode = set_dsp_mode( Mode )`
Parameter	`Mode` Gewünschter DSP-Modus. Als 32-Bit-Wert ohne Vorzeichen. = 0: Entspricht 500 MHz RTC5-Karten (ältere Baureihen). = 1: Reserviert. = 2: Entspricht 720 MHz RTC5-Karten (neuere Baureihen). = 3: Entspricht RTC6-Karten. Standardeinstellung.
Rückgabe	Eingestellter DSP-Modus vor der Ausführung von **set_dsp_mode**. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_dsp_mode**
Hinweise	• **set_dsp_mode** dient dazu, die im Standardmodus (DSP-Modus 3) schnellere Abarbeitung kurzer Vektoren auf eine zur RTC5 kompatible, langsamere Abarbeitung umzustellen, siehe Abschnitt "Automatische Delay-Anpassung", Seite 158. • **set_dsp_mode** wird benötigt, wenn: – mehrere RTC6-Karten unterschiedlicher DSP-Modi in einer Master/Slave-Kette synchron betrieben werden sollen – *UND* die Anpassung der Kurzbefehlszählung nicht mit **sync_slaves** ausgeführt wird • Ein DSP-Modus > 3 kann nicht gesetzt werden. In diesem Fall wird `Mode` auf 3 geclippt. • **set_dsp_mode** ist in Boot-Phase 2 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Der Standardmodus (DSP-Modus 3) ist verfügbar.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**sync_slaves**, **get_rtc_version**

Normaler Listenbefehl	**set_duty_cycle_table**
Funktion	Kategorie: Listenbefehle nur für einen begrenzten Benutzerkreis. Setzt einen Wert in der Lookup-Tabelle für die Pulslänge.
Aufruf	`set_duty_cycle_table( Index, DutyCycle )`
Parameter	`Index` — Index in der Lookup-Tabelle für die Pulslänge. Zulässiger Wertebereich: 0…10. Als 32-Bit-Wert ohne Vorzeichen.
	`DutyCycle` — Duty Cycle. In Promille. Zulässiger Wertebereich: 0…1.000. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_duty_cycle_table**
Hinweise	• Der Zusammenhang zwischen Encoderfrequenz und Laserfrequenz ist wie folgt definiert: – Encoderfrequenz = `Index` × [`Fmax` / 10] • Innerhalb der Lookup-Tabelle werden die Werte linear interpoliert. • Der Wert für Pulslänge wird abhängig von der Laserfrequenz gesetzt: – Pulslänge = Laserfrequenz × `DutyCycle` / 1.000 • Für Listenbefehle nur für einen begrenzten Benutzerkreis gilt: Diese Benutzer wurden im Umgang mit diesen RTC6-Befehlen separat informiert. Deswegen wird nur die Befehlstabelle aufgenommen und auf weitere Informationen in diesem Handbuch verzichtet.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 626, OUT 627.
Verweise	**set_laser_timing_table**, **regulation3**

Unverzögerter Kurzer Listenbefehl	**set_ellipse**
Funktion	Definiert die Form eines ellipsenförmigen Bogens, der anschließend mit **mark_ellipse_abs** oder **mark_ellipse_rel** markiert werden kann.
Aufruf	`set_ellipse( a, b, Phi0, Phi )`
Parameter	`a` Länge der Ellipsenhalbachse. In Bits. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…8.388.607]. Überlaufende positive Werte werden auf die Randwerte geclippt.
	`b` Siehe `a`.
	`Phi0` Anfangsphasenwinkel (Lage des Bogenstartpunkts relativ zum Endpunkt der Halbachse a). In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat. Ein positives Vorzeichen bedeutet “im Uhrzeigersinn”. `Phi0` wird auf den Wertebereich [0…<360°] normiert.
	`Phi` Bogenwinkel (zu markierender Ellipsenausschnitt). In Grad. Als Wert im 64-Bit-IEEE Gleitkommaformat. Ein positives Vorzeichen bedeutet “im Uhrzeigersinn”. Zulässiger Wertebereich: [–2.880,0°…+2.880,0°] (±8 volle Ellipsen). Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_set_ellipse**
Hinweise	• Die Position und Orientierung des zu markierenden ellipsenförmigen Bogens im 2D-Bildfeld werden mit **mark_ellipse_abs** oder **mark_ellipse_rel** angegeben. Zur Bedeutung der einzelnen Parameter, siehe Abschnitt “Ellipse-Befehle”, Seite 141. • Für `a` < 1 und/oder `b` < 1 wird **set_ellipse** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `a` und `b` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 639, OUT 645: Zulässiger Wertebereich von `Phi` wurde verringert (vorher: [–3.600,0°…+3.600,0°] = ±10 volle Ellipsen).
Verweise	**mark_ellipse_abs**, **mark_ellipse_rel**

Kontrollbefehl	**set_encoder_filter_ctrl**
Funktion	Schaltet entweder den Encodergeschwindigkeitsfilter oder den Encodereingangsfilter ein (aus).
Aufruf	`set_encoder_filter_ctrl( Length, Mode )`
Parameter	`Length` Kombiniert mit `Mode` = 0: • Encodergeschwindigkeitsfilter-Länge. Als Exponent zur Basis 2. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [0, 1, 2, 3, 4, 5, 6, 7]. Kombiniert mit `Mode` = 1: • Encodereingangsfilter. Als 32-Bit-Wert ohne Vorzeichen. = 0 Aus. = 1 Ein. `Mode` Art des Encoderfilters. Als 32-Bit-Wert ohne Vorzeichen. = 0 Encodergeschwindigkeitsfilter. = 1 Encodereingangsfilter. > 1 Reserviert.
Multi-Board-Befehlsname	**n_set_encoder_filter_ctrl**
Hinweise	• **set_encoder_filter_ctrl** wird nicht ausgeführt für (**get_last_error**-Returncode `RTC6_PARAM_ERROR`): – `Length` > 7, `Mode` = 0 – `Length` > 1, `Mode` = 1 – `Mode` > 1 • Encodergeschwindigkeitsfilter – Glättet Encodergeschwindigkeiten auf Basis des gleitenden Mittelwerts. – `Length` = 0 (2^0 = 1) schaltet den Encodereingangsfilter aus. – Ein höherer `Length`-Wert verbessert die Filterwirkung auf Kosten einer gewissen Verzögerung der verwendeten Encodergeschwindigkeiten. • Encodereingangsfilter – Begrenzt die Encodersignalfrequenz auf 4 MHz (16 Encodersignalflanken / µs). Siehe auch Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323. – `Length` = 1 schaltet den Encodereingangsfilter ein. Default nach **load_program_file**. – `Length` = 0 schaltet den Encodereingangsfilter aus. Damit sind Encodersignalfrequenzen von > 4 MHz möglich. Allerdings erkennt die RTC6-Karte dann keine Encoderfehler mehr (**get_marking_info**(`Bit #16`…`Bit #21`). Sie müssen daher selber sicher stellen: • Die empfangenen Signale sind gültig • Die "Processing on the fly"-Anwendung funktioniert wie erwartet • Siehe auch Abschnitt "Eingänge für externe Encodersignale", Seite 324.

Kontrollbefehl	set_encoder_filter_ctrl
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 641, OUT 642, RBF 638. Letzte Änderung DLL 646, OUT 649, RBF 641: `Mode` = 1.
Verweise	–

Verzögerter Kurzer Listenbefehl	**set_encoder_speed**
Funktion	Definiert die Soll-Encodergeschwindigkeit und weitere Parameter für die Encodergeschwindigkeits-abhängige Laser-Steuerung.
Aufruf	`set_encoder_speed( EncoderNo, Speed, Smooth )`
Parameter	`EncoderNo` Nummer des Encoders, der für die Geschwindigkeitsmessung verwendet werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoderzähler "Encoder0". = 1: Encoderzähler "Encoder1". = 2 und 3: Vektorielle Encodergeschwindigkeit: aus beiden Encodergeschwindigkeiten (aus den Zählerpulsraten beider Encoderzähler wird eine vektorielle Geschwindigkeit berechnet und dann bei der Encodergeschwindigkeits-abhängige Laser-Steuerung mit `Mode`= 5 verwendet. Höherwertige Bits werden ignoriert.
	`Speed` Soll-Encodergeschwindigkeit (Zählerpulsrate) in [Zählerpulse/ms]. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [100.0…16000.0]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Smooth` Glättungsfaktor eines 2-stufigen Tiefpasses. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0.0…1.0]. Zu große Werte werden geclippt.
Multi-Board-Befehlsname	**n_set_encoder_speed**
Hinweise	• Für `Smooth` = 0.0 wird ausschließlich die aktuelle Encodergeschwindigkeit `CurrentSpeed` (entsprechend der innerhalb der letzten 10 µs empfangenen Zählerpulse) verwendet, für `Smooth` = 1.0 jeweils die Geschwindigkeit des vorhergehenden Taktes `PreviousSpeed`. Allgemein gilt für die verwendete Geschwindigkeit: Speed = `PreviousSpeed` × `Smooth` + `CurrentSpeed` × (1.0 – `Smooth`). • Für `Speed` ≤ 0.0 oder `Smooth` < 0.0 wird **set_encoder_speed** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Ein Encoderinkrement liefert 4 Zählerpulse, siehe Abschnitt "Eingänge für externe Encodersignale", Seite 324. • Der maximale Wert für `Speed` (16000.0) entspricht einer Zählrate von 16 MHz. Der minimale Wert für `Speed` (100.0) entspricht einer Zählrate von 1/(10 µs), d.h. einem Zählerpuls pro Ausgabeperiode. Man beachte die geringe Auflösung der Encodergeschwindigkeits-abhängige Laser-Steuerung bei kleinen Geschwindigkeitswerten! Die Encodergeschwindigkeits-abhängige Korrektur ist nur dann empfehlenswert, wenn deutlich mehr als ein Zählerpuls pro Ausgabeperiode (10 µs) ankommt. • Siehe auch Abschnitt "Encodergeschwindigkeits-abhängige Laser-Steuerung", Seite 215.

Verzögerter Kurzer Listenbefehl	**set_encoder_speed**
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_encoder_speed_ctrl**, **set_auto_laser_control**

Kontrollbefehl	**set_encoder_speed_ctrl**	
Funktion	Wie **set_encoder_speed**, aber ein Kontrollbefehl.	
Aufruf	`set_encoder_speed_ctrl( EncoderNo, Speed, Smooth )`	
Parameter	`EncoderNo`	Wie **set_encoder_speed**.
	`Speed`	Wie **set_encoder_speed**.
	`Smooth`	Wie **set_encoder_speed**.
Multi-Board-Befehlsname	**n_set_encoder_speed_ctrl**	
Hinweise	• **set_encoder_speed_ctrl** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`), wenn: – `Speed` ≤ 0.0 – `Smooth` < 0.0 • **set_encoder_speed_ctrl** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist • **set_encoder_speed_ctrl** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**set_encoder_speed**, **set_auto_laser_control**	

Normaler Listenbefehl	**set_end_of_list**
Funktion	Beendet die Ausführung einer Liste.
Aufruf	`set_end_of_list()`
Parameter	Keine.
Rückgabe	Keine.
Multi-Board-Befehlsname	**n_set_end_of_list**
Hinweise	• Falls beim Abarbeiten einer Liste ein **set_end_of_list** erreicht wird und zuvor *kein* automatischer Listenwechsel aktiviert wurde, siehe Kapitel 6.4.6 “Listen automatisch wechseln”, Seite 113, so endet die Listenausführung. Die Signale für den “Laser active”-Betrieb werden dann ausgeschaltet und ein gegebenenfalls mit **home_position** oder **home_position_xyz** definierter Home-Jump ausgeführt (für die Dauer des Home-Jumps ist der INTERNAL-BUSY-Listenausführungsstatus gesetzt). • Die Ausführung fährt dagegen beim Erreichen eines **set_end_of_list** mit der jeweils anderen Liste fort, wenn vorher ein automatischer Listenwechsel *aktiviert* wurde. Die jeweils andere Liste kann auch immer “Liste 1” sein, falls “Liste 2” nicht konfiguriert wurde (`Mem2` = 0, siehe **config_list**). • Beim Abarbeiten von **set_end_of_list** wird in jedem Fall der USED-Listenstatus der jeweiligen Liste (USED1 oder USED2) gesetzt und der BUSY-Listenstatus (BUSY1 oder BUSY2) der Liste zurückgesetzt, siehe auch Kapitel 6.4.3 “Listenausführungsstatus”, Seite 111. Der BUSY-Listenausführungsstatus wird dagegen nur zurückgesetzt, wenn zuvor *kein* automatischer Listenwechsel aktiviert wurde. • Beim Laden des **set_end_of_list** erfolgt niemals ein automatischer Listenwechsel des Eingabe-Zeigers (im Gegensatz zum automatischen Listenwechsel des Ausgabe-Zeigers bei der Ausführung von **set_end_of_list**, wenn zuvor mit **auto_change_pos** oder **start_loop** aktiviert). • Beim Laden von **set_end_of_list** wird der READY-Listenstatus (READY1 oder READY2) gesetzt und der LOAD-Listenstatus (LOAD1 oder LOAD2) zurückgesetzt. Zusätzlich wird ein Flush der gepufferten Listeneingabe ausgelöst, siehe Kapitel 6.4.1 “Listen laden”, Seite 108. • **set_end_of_list** wird sowohl beim Laden als auch bei der Ausführung ignoriert, d. h. durch ein **list_nop** ersetzt, wenn gerade ein indiziertes Unterprogramm geladen oder ausgeführt wird (**get_last_error**-Returncode `RTC6_IGNORED`).
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Zusätzlicher USED-Listenstatus.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_start_list**

Kontrollbefehl	**set_eth_boot_control**
Funktion	Standalone-Funktionalität: Aktiviert oder deaktiviert das automatische Booten.
Voraussetzung	RTC6-Software-Paket ≥ V1.7.0 und BIOS-ETH ≥ 26.
Aufruf	`set_eth_boot_control( Ctrl )`
Parameter	`Ctrl` Steuerparameter. = 0: Deaktiviert das automatische Booten. > 0: Aktiviert das automatische Booten. Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Keine.
Multi-Board-Befehlsname	**n_set_eth_boot_control**
Hinweise	• **set_eth_boot_control** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • Während der Ausführung von **set_eth_boot_control** ist der 10 µs-Takt des DSP für mehrere ms unterbrochen. • **set_eth_boot_control** ist nur bei RTC6 Ethernet-Karten erlaubt. Anderenfalls wird ein **get_last_error**-Returncode `RTC6_TYPE_REJECTED` erzeugt. • Falls **set_eth_boot_control** nicht erfolgreich war, wird Bit 27 gesetzt, siehe **eth_get_error** und **eth_get_last_error**. • Siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 618, OUT 618, RBF 623.
Verweise	**eth_boot_dcmd**

Kontrollbefehl	**set_extstartpos**
Funktion	Definiert die Adresse (im RTC6-Listenspeicher-Bereich “Liste 1” oder “Liste 2”), bei der die Ausführung bei externen Starts fortfahren soll.
Aufruf	`set_extstartpos( Pos )`
Parameter	`Pos` Absolute Speicheradresse des ersten auszugebenden Listenbefehls. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_set_extstartpos**
Hinweise	• In der Standardeinstellung wird die Ausführung durch einen externen Start am Anfang von “Liste 1” (`Pos` = 0) gestartet oder fortgesetzt. • `Pos` muss sich im Bereich von “Liste 1” oder “Liste 2” befinden. Anderenfalls wird `Pos` = 0 gesetzt. • Die angegebene Startadresse wird so lange für alle externen Starts verwendet, bis eine neue Adresse mit **set_extstartpos** oder **set_extstartpos_list** angegeben wird. Die Bereichsprüfung für `Pos` findet vor jedem externen Start statt; `Pos` kann daher auch zu einem späteren Zeitpunkt auf 0 gesetzt werden (und bleibt dann auf diesem Wert), falls die Konfiguration in der Zwischenzeit entsprechend geändert wurde (siehe **config_list**). • Siehe auch Abschnitt “Externer Start”, Seite 315. • **set_extstartpos** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 “Automatisches Booten – Ablauf im Detail”, Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_extstartpos_list**, **set_control_mode**

Unverzögerter Kurzer Listenbefehl	**set_extstartpos_list**
Funktion	Wie **set_extstartpos**, aber ein Listenbefehl.
Aufruf	`set_extstartpos_list( Pos )`
Parameter	`Pos` Wie **set_extstartpos**.
Multi-Board-Befehlsname	**n_set_extstartpos_list**
Hinweise	• **set_extstartpos_list** kann innerhalb einer Liste verwendet werden, um sie mit der nachfolgenden Liste zu verknüpfen. • Siehe **set_extstartpos**.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_extstartpos**

Kontrollbefehl	**set_ext_start_delay**
Funktion	Stellt eine Streckenverzögerung für externe Starts ein, so dass die Ausführung der Starts relativ zum jeweils auslösenden Eingangssignal oder **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl entsprechend verzögert wird.
Aufruf	`set_ext_start_delay( Delay, EncoderNo )`
Parameter	`Delay` Streckenverzögerung (Zählerschritte des mit `EncoderNo` ausgewählten Encoderzählers). Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}\ldots+(2^{31}-1)]$ `EncoderNo` Nummer des zu verwendenden Encoderzählers. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoderzähler "Encoder0". = 1: Encoderzähler "Encoder1".
Multi-Board-Befehlsname	**n_set_ext_start_delay**
Hinweise	• Zu externen Starts, siehe Abschnitt "Externer Start", Seite 315. • Die Streckenverzögerung wird in (relativen) Zähleinheiten des ausgewählten Encoderzählers angegeben (die RTC6-Encoderzähler werden durch ein externes oder simuliertes Encodersignal getriggert, siehe Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323). • Das Vorzeichen der Streckenverzögerung (Parameter `Delay`) sollte passend zur jeweiligen Encoderzählrichtung ausgewählt werden (diese richtet sich bei externer Triggerung nach der Bewegungsrichtung des Werkstücks und ist bei simulierten Encodern immer positiv). • Mit `Delay` = 0 wird die Streckenverzögerung deaktiviert. • Wird die Streckenverzögerung so angegeben, dass ein Start-Trigger (initiiert durch **simulate_ext_start** oder ein externes Startsignal) erfolgt, während der BUSY-Listenausführungsstatus (wie bei Listenausgabe) oder der INTERNAL-BUSY-Listenausführungsstatus (wie z.B. bei **goto_xy**) gesetzt ist, so wird durch diesen Start-Trigger kein Start ausgelöst (in diesem Fall wird Bit #11 des **get_startstop_info**-Rückgabewerts gesetzt). • Streckenverzögerungen werden auch mit **simulate_ext_start** eingestellt. Bei der Initialisierung (mit **load_program_file**), durch einen externen Stopp und durch **stop_execution** wird die Steckenverzögerung deaktiviert. Sie kann auch mit **set_control_mode**/**set_control_mode_list** (Bit #2) deaktiviert werden. • Zuvor angetriggerte externe Starts, die aber noch nicht ausgeführt wurden und stattdessen noch in der Warteschleife von bis zu 8 Starts gehalten werden, werden durch **set_ext_start_delay** vernichtet. • Für `EncoderNo` > 1 wird **set_ext_start_delay** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).

Kontrollbefehl	set_ext_start_delay
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. **set_ext_start_delay** kann bei der RTC6 nicht nur zusammen mit einem externen Encoder, sondern auch mit einer mit **simulate_encoder** gestarteten Encodersimulation (siehe Abschnitt "Encodersimulation", Seite 324) verwendet werden. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Delay` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**simulate_ext_start**, **set_ext_start_delay_list**, **set_extstartpos**, **set_extstartpos_list**, **set_control_mode**, **set_control_mode_list**

Normaler Listenbefehl	set_ext_start_delay_list
Funktion	Wie **set_ext_start_delay**, aber ein Listenbefehl.
Aufruf	`set_ext_start_delay_list( Delay, EncoderNo )`
Parameter	`Delay` Wie **set_ext_start_delay**. `EncoderNo` Wie **set_ext_start_delay**.
Multi-Board-Befehlsname	**n_set_ext_start_delay_list**
Hinweise	• Für `EncoderNo` > 1 wird **set_ext_start_delay_list** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Siehe **set_ext_start_delay**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_ext_start_delay**

Kontrollbefehl	**set_firstpulse_killer**
Funktion	Stellt die Länge des FirstPulseKiller-Signals ein.
Aufruf	`set_firstpulse_killer( Length )`
Parameter	`Length` Länge des FirstPulseKiller-Signals. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs. Zulässiger Wertebereich: [0…+(2^{26}–1)]. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_set_firstpulse_killer**
Hinweise	• Die mit dem FirstPulseKiller-Signal verbundene Taktfrequenz ist 64 MHz. 1 Bit entspricht 1/64 μs. • Der Signalpegel wird mit **set_laser_control** eingestellt. • Im YAG-Modus 2 wird auch das Q-Switch-Delay entsprechend verändert, siehe Abschnitt "YAG-Modus-Unterschiede", Seite 200. • Im CO_2-Modus hat **set_firstpulse_killer** keine Auswirkung. • Der Laser-Mode wird mit **set_laser_mode** eingestellt, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192. • Q-Switch-Pulslänge und Q-Switch-Periodendauer können mit **set_laser_pulses_ctrl**, **set_laser_pulses** oder **set_laser_timing** eingestellt werden.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert mit 8. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_mode**, **set_laser_timing**, **config_laser_signals**

Unverzögerter Kurzer Listenbefehl	**set_firstpulse_killer_list**
Funktion	Wie **set_firstpulse_killer**, aber ein Listenbefehl.
Aufruf	`set_firstpulse_killer_list( Length )`
Parameter	`Length` Wie **set_firstpulse_killer**.
Multi-Board-Befehlsname	**n_set_firstpulse_killer_list**
Hinweise	• –
RTC4→RTC6	Wie **set_firstpulse_killer**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_firstpulse_killer**

Normaler Listenbefehl	**set_fly_1_axis**
Funktion	"Fly-Erweiterung"-Befehl: Aktiviert eine 1-Achsen-"Processing on the fly"-Anwendung.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **set_fly_1_axis** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_1_axis( Axis, Mode, Scale )`
Parameter	`Axis` Achse aus Tabelle 3, Seite 271. Als 32-Bit-Wert ohne Vorzeichen. `Mode` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. `Scale` Skalierungsfaktor. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: • 1/256 ≤ |`Scale`| ≤ 16.000,0 bei Linear-Achse (`1`, `2` oder `3`) • |`Scale`| > 100,0 bei Dreh-Achse (`4`) `Scale` kann + oder – annehmen. Nur der Absolutbetrag ist eingeschränkt. `Scale` wird ignoriert für `Mode` = (8/12/16).
Multi-Board-Befehlsname	**n_set_fly_1_axis**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **set_fly_1_axis** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • Bei `Mode`(1…4) benötigt **set_fly_1_axis** zwei 10 µs-Takte zur Ausführung. • **set_fly_1_axis** überschreibt eine vorherige Festlegung für dieselbe Achse. **set_fly_1_axis** kann als Linear-Achse mit anderen Linear-Achsen kombiniert werden, aber nicht mit einer Dreh-Achse. Eine noch aktive Rotations-Fly-Anwendung (`set_fly_1_axis( Axis =4 )`) wird automatisch deaktiviert und umgekehrt. Es wird empfohlen, inkompatible "Processing on the fly"-Korrekturen vorher explizit auszuschalten, siehe z.B. **fly_return_1_axis** und **fly_return_2_axes**. • Bei einem unzulässigen Parameter-Wert wird **set_fly_1_axis** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Folgende Befehlsaufrufe werden gleichartig ausgeführt: – `set_fly_1_axis( 1, 1, ScaleX )` = `set_fly_x( ScaleX )` – `set_fly_1_axis( 2, 2, ScaleY )` = `set_fly_y( ScaleY )` – `set_fly_1_axis( 3, EncoderNo+3, ScaleZ )` = `set_fly_z( ScaleZ, EncoderNo )` – `set_fly_1_axis( 4, 1, Resolution )` = `set_fly_rot( Resolution )` – `set_fly_1_axis( 1, 6, ScaleX )` = `set_fly_x_pos( ScaleX )` – `set_fly_1_axis( 2, 6, ScaleY )` = `set_fly_y_pos( ScaleY )` – `set_fly_1_axis( 4, 6, Resolution )` = `set_fly_rot_pos( Resolution )` – `set_fly_1_axis( 1, 7, Scale )` entspricht `set_mcbsp_in( 1, Scale )`
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_fly_2_axes**, **set_fly_3_axes**

Normaler Listenbefehl	**set_fly_2_axes**
Funktion	"Fly-Erweiterung"-Befehl: Aktiviert eine 2-Achsen-"Processing on the fly"-Anwendung.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **set_fly_2_axes** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_2_axes( Axis1, Mode1, Scale1, Axis2, Mode2, Scale2 )`
Parameter	`Axis1` Achse aus Tabelle 3, Seite 271. Zulässige Werte: `1`…`3` (nur Linear-Achse). Als 32-Bit-Wert ohne Vorzeichen. `Mode1` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. `Scale1` Skalierungsfaktor. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: • 1/256 ≤ |`Scale1`| ≤ 16.000,0 bei Linear-Achse (`1`, `2` oder `3`) `Scale1` kann + oder – annehmen. Nur der Absolutbetrag ist eingeschränkt. `Scale1` wird ignoriert für `Mode1` = (8/12/16). `Axis2` Wie `Axis1`. `Mode2` Wie `Mode1`. `Scale2` Wie `Scale1`.
Multi-Board-Befehlsname	**n_set_fly_2_axes**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **set_fly_2_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • Bei `Mode`(1…4) benötigt **set_fly_2_axes** zwei 10 µs-Takte zur Ausführung. • **set_fly_2_axes** überschreibt eine vorherige Festlegung für dieselben Achsen (auch einzeln). **set_fly_2_axes** kann als 2 Linear-Achsen mit einer dritten Linear-Achse kombiniert werden, aber nicht mit einer Dreh-Achse. Eine noch aktive Rotations-Fly-Anwendung (`set_fly_1_axis( Axis =4 )`) wird automatisch deaktiviert und umgekehrt. Es wird empfohlen, inkompatible "Processing on the fly"-Korrekturen vorher explizit auszuschalten, siehe z.B. **fly_return_2_axes**. • `Axis1` und `Axis2` müssen verschiedene Linear-Achsen sein und dürfen keine Dreh-Achsen sein. • `Mode1` und `Mode2` können gleich sein. Wenn sie denselben Encoder verwenden, kann mit `Mode1`/`Mode2` = 1 / 3 oder 2 / 4 jeweils ein unterschiedlicher Skalierungsfaktor angegeben werden. • Bei einem unzulässigen Parameter-Wert wird **set_fly_2_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).

Normaler Listenbefehl	**set_fly_2_axes**

Hinweise (Forts.)	• Folgende Befehlsaufrufe werden gleichartig ausgeführt:
	– `set_fly_2_axes( 1, 1, ScaleX, 2, 2, ScaleY ) = set_fly_2d( ScaleX, ScaleY )`
	– `set_fly_2_axes( 1, 1, ScaleX, 2, 2, ScaleY ) =`
	`{`
	`set_fly_1_axis( 1, 1, ScaleX );`
	`set_fly_1_axis( 2, 2, ScaleY ));`
	`}`
	Die Encoderresets erfolgen:
	• Bei einem **set_fly_2_axes**-Aufruf im gleichen 10 μs-Takt
	• Bei zwei **set_fly_1_axis**-Aufrufen in zwei verschiedenen 10 μs-Takten
	– `set_fly_2_axes( 1, 11, Scale, 2, 15, Scale)` entspricht
	`set_mcbsp_in_list( 3, Scale )`

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_fly_1_axis**, **set_fly_3_axes**

Normaler Listenbefehl	**set_fly_2d**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer linearen Werkstück-Bewegung in 2 Dimensionen (gemäß den Encoderwerten, die über die Encoderzähler “Encoder0” und “Encoder1” an die RTC6 übergeben werden). Setzt die zugehörigen Skalierungsfaktoren.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_2d** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_2d( ScaleX, ScaleY )`
Parameter	`ScaleX` Skalierungsfaktor für die x-Richtung (Encoderzähler “Encoder0”). In Bits/Count. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleX`| ≤ 16.000,0.
	`ScaleY` Skalierungsfaktor für die y-Richtung (Encoderzähler “Encoder1”). In Bits/Count. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleY`| ≤ 16.000,0.
Multi-Board-Befehlsname	**n_set_fly_2d**
Hinweise	• `ScaleX` und `ScaleY` können je nach Bewegungsrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur (u.a. zur Ermittlung der Skalierungsfaktoren und zur Deaktivierung der “Processing on the fly”-Korrektur) siehe Kapitel 8.6 ““Processing on the fly””, Seite 254. Zur Verwendung von **set_fly_2d** siehe Kapitel 8.6.4 “2D-Bewegungen ausgleichen”, Seite 261. • Für unzulässige Parameterwerte (z.B. `ScaleX` = 0) aktiviert **set_fly_2d** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_2d** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne **set_fly_2d**-“Processing on the fly”-Korrektur) ausgeführt. Eine mit **set_fly_2d** erfolgreich aktivierte “Processing on the fly”-Korrektur schaltet jedoch jede andere “Processing on the fly”-Korrektur ab und wird seinerseits durch jeden anderen “Processing on the fly”-Befehl abgeschaltet, auch wenn dieser unzulässige Parameter enthalten sollte, siehe Abschnitt “Übersicht”, Seite 254. • Wenn mit **load_fly_2d_table** und **init_fly_2d** eine Encoderkompensation eingestellt ist, werden die aktuellen Encoderwerte zu den bisherigen Referenzwerten der 2D-Encoderkompensation addiert und dann die Summen als neue Referenzwerte gespeichert. Die Encoderzähler werden danach auf 0 zurückgesetzt. • Mit **set_control_mode**(Bit #9) sollte *nicht* eingestellt werden, dass die Encoderzähler erst mit dem nachfolgenden externen Start-Trigger zurückgesetzt werden. Anderenfalls gehen die Referenzwerte der 2D-Encoderkompensation verloren. **set_fly_2d** setzt die Encoderzähler immer zurück, auch wenn diese bereits durch einen externen Start-Trigger zurückgesetzt wurden. Siehe auch Abschnitt “2D-Encoderkompensation für xy-Verfahrtische”, Seite 261.

Normaler Listenbefehl	**set_fly_2d**
Hinweise (Forts.)	• Verwenden Sie nicht zwischendurch **set_fly_x** oder **set_fly_y** zum Einschalten der “Processing on the fly”-Anwendung, wenn Sie **set_fly_2d** zusammen mit einer 2D-Encoderkompensation für einen xy-Verfahrtisch verwenden wollen, da hierbei ebenfalls die Referenzwerte verloren gehen. • Wenn noch keine Korrekturtabelle für die 2D-Encoderkompensation auf die Karte geladen ist (siehe **load_fly_2d_table**), werden die Encoderwerte unkorrigiert verwendet. • Zum Einschalten einer **set_fly_2d**-“Processing on the fly”-Korrektur kann auch **activate_fly_2d**/**activate_fly_2d_encoder** verwendet werden.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `ScaleX` und `ScaleY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**init_fly_2d**, **load_fly_2d_table**, **get_fly_2d_offset**, **activate_fly_2d**, **activate_fly_xy**

Multipler Listenbefehl	**set_fly_3_axes**
Funktion	"Fly-Erweiterung"-Befehl: Aktiviert eine 3-Achsen-"Processing on the fly"-Anwendung.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **set_fly_3_axes** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_3_axes( ModeX, ScaleX, ModeY, ScaleY, ModeZ, ScaleZ )`
Parameter	`ModeX` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. `ScaleX` Skalierungsfaktor. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: • 1/256 ≤ |`ScaleX`| ≤ 16.000,0 bei Linear-Achse (`1`, `2` oder `3`) `ScaleX` kann + oder – annehmen. Nur der Absolutbetrag ist eingeschränkt. `ScaleX` wird ignoriert für `ModeX` = (8/12/16). `ModeY` Wie `ModeX`. `ScaleY` Wie `ScaleX`. `ModeZ` Wie `ModeX`. `ScaleZ` Wie `ScaleX`.
Multi-Board-Befehlsname	**n_set_fly_3_axes**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **set_fly_3_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • Bei `Mode`(1…4) benötigt **set_fly_3_axes** zwei 10 µs-Takte zur Ausführung. • **set_fly_3_axes** belegt 2 RTC6-Listenspeicherplätze. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem zweiten Teil als Normaler Listenbefehl ausgeführt. Eventuell anliegende Verzögerte Kurze Listenbefehle werden davor ausgeführt. • **set_fly_3_axes** überschreibt eine vorherige Festlegung für dieselben Achsen (auch einzeln). Eine noch aktive Rotations-Fly-Anwendung (`set_fly_1_axis( Axis =4 )`) wird automatisch deaktiviert und umgekehrt. Es wird empfohlen, inkompatible "Processing on the fly"-Korrekturen vorher explizit auszuschalten, siehe z.B. **fly_return_3_axes**. • Die Achsen sind automatisch auf `1`, `2`, `3` festgelegt und müssen daher nicht explizit angegeben werden. • `ModeX`, `ModeY`, `ModeZ` können gleich sein. Wenn sie denselben Encoder verwenden, kann mit `ModeX`/`ModeY`/`ModeZ` = 1 / 3 oder 2 / 4 jeweils ein unterschiedlicher Skalierungsfaktor angegeben werden. • Bei einem unzulässigen Parameter-Wert wird **set_fly_3_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).

Multipler Listenbefehl	**set_fly_3_axes**

Hinweise (Forts.)	• Folgende Befehlsaufrufe werden gleichartig ausgeführt:
	– `set_fly_3_axes( 1, ScaleX, 2, ScaleY, 3 oder 4, ScaleZ ) =`
	`{`
	`set_fly_1_axis( 1, 1, Scale X );`
	`set_fly_1_axis( 2, 2, Scale Y );`
	`set_fly_1_axis( 3, 3 oder 4, Scale Z );`
	`}`
	– `set_fly_3_axes( 1, ScaleX, 2, ScaleY, 3 oder 4, ScaleZ ) =`
	`{`
	`set_fly_2_axes( 1, 1, Scale1 X, 2, 2, Scale2 X );`
	`set_fly_1_axis( 3, 3 oder 4, Scale Z );`
	`}`
	Die Encoderresets erfolgen:
	• Bei einem **set_fly_3_axes**-Aufruf im gleichen 10 µs-Takt
	• Bei drei **set_fly_1_axis**-Aufrufen in drei verschiedenen 10 µs-Takten
	• Bei einem **set_fly_2_axes**-Aufruf und einem **set_fly_1_axis**-Aufruf in zwei verschiedenen 10 µs-Takten
	– `set_fly_3_axes( 8, 1, 0, 12, 1, 0, 16, 1, 0 )` entspricht
	`set_multi_mcbsp_in_list( Ctrl, P, Mode )`

RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_fly_1_axis**, **set_fly_2_axes**

Unverzögerter Kurzer Listenbefehl	**set_fly_limits**
Funktion	Definiert die Bereichsgrenzen für den kundenspezifischen Überwachungsbereich in "Processing on the fly"-Anwendungen.
Aufruf	`set_fly_limits( Xmin, Xmax, Ymin, Ymax )`
Parameter	`Xmin` Bereichsgrenze. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Xmax` Wie `Xmin` (analog).
	`Ymin` Wie `Xmin` (analog).
	`Ymax` Wie `Xmin` (analog).
Multi-Board-Befehlsname	**n_set_fly_limits**
Hinweise	• Zur Verwendung von **set_fly_limits** siehe Abschnitt "Kundenspezifischer Überwachungsbereich", Seite 268. • Bei der Initialisierung (mit **load_program_file**) werden die Bereichsgrenzen auf die folgenden Standardwerte gesetzt: – `Xmin` = `Ymin` = –524.288 – `Xmax` = `Ymax` = +524.287 • Bereichsgrenzen, für die der Parameterwert 0 übergeben wird, werden auf die oben genannten Standardwerte gesetzt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_marking_info**

Unverzögerter Kurzer Listenbefehl	**set_fly_limits_z**
Funktion	Definiert die Bereichsgrenzen für den kundenspezifischen Überwachungsbereich in "Processing on the fly"-Anwendungen für die z-Achse.
Aufruf	`set_fly_limits_z( Zmin, Zmax )`
Parameter	`Zmin` Bereichsgrenze. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`Zmax` Wie `Zmin` (analog).
Multi-Board-Befehlsname	**n_set_fly_limits_z**
Hinweise	• Zur Verwendung von **set_fly_limits_z** siehe auch Kapitel 8.6.9 "'Processing on the fly"-Korrektur überwachen", Seite 267. • Bei der Initialisierung (mit **load_program_file**) werden die Bereichsgrenzen auf die folgenden Standardwerte gesetzt: – `Zmin` = –524.288 – `Zmax` = +524.287 • Auch Bereichsgrenzen, für die der Parameterwert 0 übergeben wird, werden auf die oben genannten Standardwerte gesetzt.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `Zmin` und `Zmax` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `Zmin` und `Zmax` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_limits**, **get_marking_info**

Normaler Listenbefehl	**set_fly_rot**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer Rotationsbewegung des Werkstücks (gemäß den Winkelpositionswerten, die über den Encoderzähler “Encoder0” an die RTC6 übergeben werden) und setzt den zugehörigen Parameter `Resolution`.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_rot** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_rot( Resolution )`
Parameter	`Resolution` Zählschritte (Counts) pro Umdrehung. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: |`Resolution`| > 100,0.
Multi-Board-Befehlsname	**n_set_fly_rot**
Hinweise	• `Resolution` kann je nach Drehrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Parameters `Resolution` siehe Kapitel 8.6 ““Processing on the fly””, Seite 254. • Vor dem Aufruf von **set_fly_rot** sollte mit **set_rot_center** oder **set_rot_center_list** das Rotationszentrum für die “Processing on the fly”-Korrektur angegeben werden. Anderenfalls wird der Default-Wert (0|0) verwendet. • Für unzulässige Parameterwerte (z.B. `Resolution` = 0) aktiviert **set_fly_rot** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_rot** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne “**set_fly_rot**”-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar, siehe Abschnitt “Übersicht”, Seite 254. • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Kapitel 8.6.5 ““Processing on the fly”-Korrektur deaktivieren”, Seite 263. • Mit **set_control_mode**(Bit #9) kann zuvor eingestellt werden, wann der Encoderzähler “Encoder0” durch **set_fly_rot** zurückgesetzt wird.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_rot_center**, **set_rot_center_list**, **fly_return**, **get_encoder**, **set_fly_rot_pos**

Normaler Listenbefehl	**set_fly_rot_pos**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer Rotationsbewegung des Werkstücks oder Scan-Systems (gemäß den Winkelpositionswerten, die über die McBSP-Schnittstelle an die RTC6 übergeben werden) und setzt dabei den zugehörigen Parameter `Resolution`.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_rot_pos** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_rot_pos( Resolution )`
Parameter	`Resolution` Zählschritte (Counts) pro Umdrehung. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: |`Resolution`| > 100,0.
Multi-Board-Befehlsname	**n_set_fly_rot_pos**
Hinweise	• `Resolution` kann je nach Drehrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Parameters `Resolution` siehe Kapitel 8.6 ““Processing on the fly””, Seite 254. • Vor dem Aufruf von **set_fly_rot_pos** sollte mit **set_rot_center** oder **set_rot_center_list** das Rotationszentrum für die “Processing on the fly”-Korrektur angegeben werden. Anderenfalls wird der Default-Wert (0|0) verwendet. • Für unzulässige Parameterwerte (z.B. `Resolution` = 0) aktiviert **set_fly_rot_pos** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_rot_pos** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne “**set_fly_rot_pos**“-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar, siehe Abschnitt “Übersicht”, Seite 254. • Zur Deaktivierung der “Processing on the fly”-Korrektur, siehe Kapitel 8.6.5 ““Processing on the fly”-Korrektur deaktivieren“, Seite 263. • Die McBSP-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden Siehe auch Abschnitt “Hinweise”, Seite 242. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Seite 89.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_rot_center**, **set_rot_center_list**, **fly_return**, **read_mcbsp**, **set_fly_rot**

Kontrollbefehl	**set_fly_tracking_error**	
Funktion	Kategorie: Kontrollbefehle nur für einen begrenzten Benutzerkreis. Schaltet die Schleppverzugskompensation von Encoderwerten für "Processing on the fly"-Anwendungen ein (aus).	
Aufruf	`set_fly_tracking_error( TrackingErrorX, TrackingErrorY )`	
Parameter	`TrackingErrorX`	Schleppverzug für die x-Achse. In Einheiten von 10 µs. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [0…4.095]. Höherwertige Bits werden ignoriert.
	`TrackingErrorY`	Schleppverzug für die y-Achse. Sonst wie `TrackingErrorX`.
Multi-Board-Befehlsname	**n_set_fly_tracking_error**	

Hinweise	• Bei aktiver SCANahead-Funktionalität wird **set_fly_tracking_error** nicht ausgeführt (**get_last_error**-Returncode `RTC6_REJECTED`).
	• `TrackingErrorX` und `TrackingErrorY` wirken additiv zu **set_controlpreview_compensation_ctrl**(`ControlPreview`).
	Beachten Sie deren unterschiedlichen Einheiten:
	– `TrackingErrorX` [10 µs]
	– `TrackingErrorY` [10 µs]
	– `ControlPreview` [1/64 µs]
	• Die Gesamt-Prädiktionswerte werden in [10/16 µs] umgerechnet:
	– Bei **set_fly_tracking_error**
	– Bei **set_controlpreview_compensation_ctrl**
	Gemäß:
	– `PredictionX` = (`TrackingErrorX` × 16) + (`ControlPreview` / 40)
	– `PredictionY` = (`TrackingErrorY` × 16) + (`ControlPreview` / 40)
	Das Ergebnis muss *jeweils* sein: ≤ 65.535.
	• Alternativ können mit **fly_prediction** die Gesamt-Prädiktionswerte auch explizit gesetzt werden.
	• Es wirken immer nur die Gesamt-Prädiktionswerte des zuletzt ausgeführten Befehls:
	– Für **set_fly_tracking_error**
	`TrackingErrorX`/`TrackingErrorY` + `ControlPreview`
	– Für **set_controlpreview_compensation_ctrl**
	`TrackingErrorX`/`TrackingErrorY` + `ControlPreview`
	– Für **fly_prediction**
	`PredictionX`/`PredictionY`
	• Siehe Kapitel 8.6.10 "Schleppverzugskompensation von Encoderwerten für "Processing on the fly"-Anwendungen", Seite 269.
	• **set_fly_tracking_error** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
	• Für Kontrollbefehle nur für einen begrenzten Benutzerkreis gilt: Diese Benutzer wurden im Umgang mit diesen RTC6-Befehlen separat informiert. Deswegen wird nur die Befehlstabelle aufgenommen und auf weitere Informationen in diesem Handbuch verzichtet.	

Kontrollbefehl	set_fly_tracking_error
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuimplementierung: vormalige Funktionalität wird reproduziert, aber das Verhalten ist im Detail anders. Zusätzlich: Wertebereich verringert.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 646, OUT 649, RBF 641: Funktionalität implementiert (vorher: keinerlei Wirkung).
Verweise	**set_controlpreview_compensation_ctrl**, **fly_prediction**

Normaler Listenbefehl	**set_fly_x**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer linearen Werkstück-Bewegung in x-Richtung (gemäß den Positionswerten, die über den Encoderzähler “Encoder0” an die RTC6 übergeben werden) und setzt den zugehörigen Skalierungsfaktor.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_x** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_x( ScaleX )`
Parameter	`ScaleX` Skalierungsfaktor für die x-Richtung (Encoderzähler “Encoder0”). In Bits/Count. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleX`| ≤ 16.000,0.
Multi-Board-Befehlsname	**n_set_fly_x**
Hinweise	• `ScaleX` kann je nach Bewegungsrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.6 “”Processing on the fly””, Seite 254. • Für unzulässige Parameterwerte (z.B. `ScaleX` = 0) aktiviert **set_fly_x** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_x** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne “**set_fly_x**”-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar, siehe Abschnitt “Übersicht”, Seite 254. • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Kapitel 8.6.5 “”Processing on the fly”-Korrektur deaktivieren”, Seite 263. • Mit **set_control_mode**(Bit #9) kann zuvor eingestellt werden, wann der Encoderzähler “Encoder0” durch **set_fly_x** zurückgesetzt wird. • Zum Einschalten einer **set_fly_x**/**set_fly_y**-“Processing on the fly”-Korrektur kann auch **activate_fly_xy**/**activate_fly_xy_encoder** verwendet werden.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: veränderter Wertebereich. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `ScaleX` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**fly_return**, **set_fly_y**, **get_encoder**, **set_fly_x_pos**, **set_fly_y_pos**, **activate_fly_xy**, **set_fly_2d**

Normaler Listenbefehl	**set_fly_x_pos**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer linearen Werkstück- oder Scan-System-Bewegung in x-Richtung (gemäß den Positionswerten, die über die McBSP-Schnittstelle an die RTC6 übergeben werden) und setzt dabei den zugehörigen Skalierungsfaktor.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_x_pos** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_x_pos( ScaleX )`
Parameter	`ScaleX` Skalierungsfaktor für die x-Richtung. In *(RTC6-)Bits/(McBSP-)Bit*. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleX`| ≤ 16.000,0.
Multi-Board-Befehlsname	**n_set_fly_x_pos**
Hinweise	• `ScaleX` kann je nach Bewegungsrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.6 ““Processing on the fly””, Seite 254. • Für unzulässige Parameterwerte (z.B. `ScaleX` = 0) aktiviert **set_fly_x_pos** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_x_pos** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne “**set_fly_x_pos**”-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar, siehe Abschnitt “Übersicht”, Seite 254. • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Kapitel 8.6.5 ““Processing on the fly”-Korrektur deaktivieren”, Seite 263. • Für 1D-Korrekturen (wenn nur **set_fly_x_pos** verwendet wird) stehen an der McBSP-Schnittstelle 32 Bit-Werte (mit Vorzeichen) zur Verfügung, für 2D-Korrekturen (**set_fly_x_pos** und **set_fly_y_pos**) dagegen nur jeweils 16 Bit (mit Vorzeichen) pro Achse. Dabei steht der von **set_fly_x_pos** verwendete x-Wert in den unteren 16 Bit, der von **set_fly_y_pos** verwendete y-Wert in den oberen 16 Bit des Wertes an der McBSP-Schnittstelle. • Die McBSP-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden. Siehe auch Abschnitt “Hinweise”, Seite 242. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Seite 89.

Normaler Listenbefehl	**set_fly_x_pos**
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `ScaleX` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**fly_return**, **set_fly_y_pos**, **read_mcbsp**, **set_fly_x**, **set_fly_y**

Normaler Listenbefehl	**set_fly_y**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer linearen Werkstück-Bewegung in y-Richtung (gemäß den Positionswerten, die über den Encoderzähler “Encoder1” an die RTC6 übergeben werden) und setzt den zugehörigen Skalierungsfaktor.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_y** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_y( ScaleY )`
Parameter	`ScaleY` Skalierungsfaktor für die y-Richtung (Encoderzähler “Encoder1”). In Bits/Count. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleY`| ≤ 16.000,0.
Multi-Board-Befehlsname	**n_set_fly_y**
Hinweise	• `ScaleY` kann je nach Bewegungsrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.6 ““Processing on the fly””, Seite 254. • Für unzulässige Parameterwerte (z.B. `ScaleY` = 0) aktiviert **set_fly_y** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_y** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne “**set_fly_y**”-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar, siehe Abschnitt “Übersicht”, Seite 254. • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Kapitel 8.6.5 ““Processing on the fly”-Korrektur deaktivieren”, Seite 263. • Mit **set_control_mode**(Bit #9) kann zuvor eingestellt werden, wann der Encoderzähler “Encoder0” durch **set_fly_y** zurückgesetzt wird. • Zum Einschalten einer **set_fly_x**/**set_fly_y**-“Processing on the fly”-Korrektur kann auch **activate_fly_xy**/**activate_fly_xy_encoder** verwendet werden.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: veränderter Wertebereich. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `ScaleY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**fly_return**, **set_fly_x**, **get_encoder**, **set_fly_x_pos**, **set_fly_y_pos**, **activate_fly_xy**, **set_fly_2d**

Normaler Listenbefehl	**set_fly_y_pos**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer linearen Werkstück- oder Scan-System-Bewegung in y-Richtung (gemäß den Positionswerten, die über die McBSP-Schnittstelle an die RTC6 übergeben werden) und setzt dabei den zugehörigen Skalierungsfaktor.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_y_pos** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_y_pos( ScaleY )`
Parameter	`ScaleY` Skalierungsfaktor für die y-Richtung in *(*RTC6*-)Bits/(McBSP-)Bit*. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleY`| ≤ 16.000,0.
Multi-Board-Befehlsname	**n_set_fly_y_pos**
Hinweise	• `ScaleY` kann je nach Bewegungsrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.6 ““Processing on the fly””, Seite 254. • Für unzulässige Parameterwerte (z.B. `ScaleY` = 0) aktiviert **set_fly_y_pos** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_y_pos** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne “**set_fly_y_pos**”-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar, siehe Abschnitt “Übersicht”, Seite 254. • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Kapitel 8.6.5 ““Processing on the fly”-Korrektur deaktivieren”, Seite 263. • Für 1D-Korrekturen (wenn nur **set_fly_y_pos** verwendet wird) stehen an der McBSP-Schnittstelle 32 Bit-Werte (mit Vorzeichen) zur Verfügung, für 2D-Korrekturen (**set_fly_x_pos** und **set_fly_y_pos**) dagegen nur jeweils 16 Bit (mit Vorzeichen) pro Achse. Dabei steht der von **set_fly_x_pos** verwendete x-Wert in den unteren 16 Bit, der von **set_fly_y_pos** verwendete y-Wert in den oberen 16 Bit des Wertes an der McBSP-Schnittstelle. • Die McBSP-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden Siehe auch Abschnitt “Hinweise”, Seite 242. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Seite 89.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `ScaleY` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**fly_return**, **set_fly_x_pos**, **read_mcbsp**, **set_fly_x**, **set_fly_y**

Normaler Listenbefehl	**set_fly_z**
Funktion	Aktiviert die “Processing on the fly”-Korrektur zur Kompensation einer linearen Werkstück-Bewegung in z-Richtung (gemäß den Positionswerten, die über den angegebenen Encoderzähler an die RTC6 übergeben werden) und setzt den zugehörigen Skalierungsfaktor.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_fly_z** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_z( ScaleZ, EncoderNo )`
Parameter	`ScaleZ` Skalierungsfaktor für die z-Richtung. In Bits/Count. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleZ`| ≤ 16.000,0.
	`EncoderNo` Nummer des zu verwendenden Encoderzählers. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoderzähler “Encoder0”. = 1: Encoderzähler “Encoder1”.
Multi-Board-Befehlsname	**n_set_fly_z**
Hinweise	• `ScaleZ` kann je nach Bewegungsrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.6 ““Processing on the fly””, Seite 254. • Für unzulässige Parameterwerte von `ScaleZ` (z.B. `ScaleZ` = 0) aktiviert **set_fly_z** keine “Processing on the fly”-Korrektur oder deaktiviert eine zuvor mit **set_fly_z** erfolgreich aktivierte “Processing on the fly”-Korrektur (deaktiviert aber keine anderen “Processing on the fly”-Korrekturen). Im letzteren Fall wird ein Sprung (mit Sprunggeschwindigkeit) auf den Endpunkt des zuletzt ausgeführten Vektor-Befehls oder Bogen-Befehls (ohne “**set_fly_z**”-“Processing on the fly”-Korrektur) ausgeführt. • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Abschnitt “Hinweise zur Verwendung”, Seite 270. • Mit **set_control_mode**(Bit #9) kann zuvor eingestellt werden, wann der Encoderzähler “Encoder0” oder “Encoder1” durch **set_fly_z** zurückgesetzt wird. • Für `EncoderNo` > 1 wird **set_fly_z** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `ScaleZ` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**fly_return_z**

Kontrollbefehl	**set_free_variable**
Funktion	Belegt eine freie Variable mit dem gewünschten Wert.
Aufruf	`set_free_variable( No, Value )`
Parameter	`No` Nummer derjenigen freien Variablen, deren Wert gesetzt werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…7]. Es werden nur die 3 niederwertigsten Bits ausgewertet.
	`Value` Gewünschter Variablenwert. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_set_free_variable**
Hinweise	• Siehe Kapitel 6.9.1 "Freie Variablen", Seite 138. • Standalone-Funktionalität: **set_free_variable** ist ein zum automatischen Booten zulässiger Kontrollbefehl (obwohl es den entsprechenden Listenbefehl **set_free_variable_list** dazu gibt), siehe Tabelle 8, Seite 1017 in Kapitel 16.8 "Standalone-Funktionalität", Seite 1012. Mit **set_free_variable** kann dem Boot-Image ein beliebiger Versionscode zugewiesen werden, der im *PC-Betrieb* vor einem /START ausgelesen und überprüft werden kann. • **set_free_variable** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 618: zum automatischen Booten zulässiger Kontrollbefehl, siehe Seite 1016.
Verweise	**set_free_variable_list**, **get_free_variable**, **set_trigger**, **eth_boot_dcmd**

Unverzögerter Kurzer Listenbefehl	**set_free_variable_list**
Funktion	Wie **set_free_variable**, aber ein Listenbefehl.
Aufruf	`set_free_variable_list( No, Value )`
Parameter	`No` Wie **set_free_variable**.
	`Value` Wie **set_free_variable**.
Multi-Board-Befehlsname	**n_set_free_variable_list**
Hinweise	• Siehe **set_free_variable**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_free_variable**

Kontrollbefehl	**set_hi**	
Funktion	Definiert Gain-Werte und Offset-Werte für die Galvanometerscanner des Scan-Systems am angegebenen Scan-Kopf-Anschluss.	
Aufruf	`set_hi( HeadNo, GalvoGainX, GalvoGainY, GalvoOffsetX, GalvoOffsetY )`	
Parameter	`HeadNo`	Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf.
	`GalvoGainX`	x-Gain-Wert. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0,01…100].
	`GalvoGainY`	Wie `GalvoGainY` (analog).
	`GalvoOffsetX`	x-Offset-Wert. In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287].
	`GalvoOffsetY`	Wie `GalvoOffsetX` (analog).
Multi-Board-Befehlsname	**n_set_hi**	
Hinweise	• Zur Verwendung von **set_hi** siehe Abschnitt "Kundeneigene Kalibrierung", Seite 290. • Die angegebenen Gain-Werte und Offset-Werte überschreiben die mit **auto_cal** eingestellten Werte oder können bei einem nachfolgenden **auto_cal**-Aufruf selbst wieder überschrieben werden. • Bei Änderungen der Gain-Werte und Offset-Werte wird der Übergang automatisch mit der voreingestellten Sprunggeschwindigkeit ausgeführt (siehe **set_jump_speed**). • **set_hi** wird nicht ausgeführt, wenn: – Ein Parameterwert unzulässig ist (**get_last_error**-Returncode `RTC6_PARAM_ERROR`) – Der BUSY-Listenausführungsstatus gesetzt ist (**get_last_error**-Returncode `RTC6_BUSY`) – Der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist (**get_last_error**-Returncode `RTC6_BUSY`) • **set_hi** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Bei einem `RTC6_PARAM_ERROR` wird der BUSY-Listenausführungsstatus nicht überprüft, d.h. Returncode `RTC6_BUSY` und `RTC6_PARAM_ERROR` kommen nicht gleichzeitig vor. • Falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist, bleiben Werte, die für den Anschluss für zweiten Scan-Kopf angegeben sind, wirkungslos. • **set_hi** ist in Boot-Phase 2 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**auto_cal**, **get_hi_pos**, **write_hi_pos**	

Kontrollbefehl	**set_input_pointer**
Funktion	Öffnet den RTC6-Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Eingabe-Zeiger an die angegebene *absolute* Adresse im RTC6-Listenspeicher-Bereich (“Liste 1” oder “Liste 2”).
Aufruf	`set_input_pointer( Pos )`
Parameter	`Pos` Position (absolute Speicheradresse) des Eingabe-Zeigers. [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_input_pointer**
Hinweise	• Der nächste Listenbefehl wird an der angegebenen Adresse gespeichert und alle nachfolgenden Listenbefehle an den darauffolgenden Adressen in der gewählten Liste. • **set_input_pointer** wirkt grundsätzlich wie **set_start_list_pos** (siehe Hinweise dort). Der Eingabe-Zeiger wird allerdings bei **set_input_pointer** unter Angabe einer *absoluten* Speicheradresse gesetzt, bei **set_start_list_pos** dagegen unter Angabe der Listennummer und der *relativen* Speicheradresse. • Für `Pos` ≥ `Mem1` + `Mem2` (siehe **config_list**) wird `Pos` auf 0 gesetzt.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_start_list_pos**, **get_input_pointer**

Unverzögerter Kurzer Listenbefehl	**set_io_cond_list**
Funktion	Setzt die Bits des 16-Bit-Digital-Ausgangs der EXTENSION 1-Stiftleiste, die im Parameter `MaskSet` gesetzt sind, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind).
Aufruf	`set_io_cond_list( Mask1, Mask0, MaskSet )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Wie `Mask1`.
	`MaskSet` Wie `Mask1`.
Multi-Board-Befehlsname	**n_set_io_cond_list**
Hinweise	• **set_io_cond_list** setzt nur diejenigen Bits des Digital-Ausgangs, die im Parameter `MaskSet` gesetzt sind und lässt die anderen Bits unverändert. • Siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81 und Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
Beispiele (Pascal)	• Setze Bit #4 des Digital-Ausgangs (DIGITAL OUT4), falls Bit #0 des Digital-Eingangs (DIGITAL IN0) gesetzt und Bit #1…Bit #3 (DIGITAL IN1…3) des Digital-Eingangs nicht gesetzt sind: `set_io_cond_list`($0001, $000E, $0010) • Setze immer Bit #15 des Digital-Ausgangs (und lasse die anderen Bits unverändert): `set_io_cond_list`(0, 0, $8000)
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**clear_io_cond_list**, **write_io_port**, **write_io_port_mask**, **get_io_status**, **read_io_port**

Kontrollbefehl	set_jump_mode	
Funktion	Gibt frei und aktiviert oder unterdrückt und deaktiviert den Sprungmodus für 2D-Sprünge und legt die zugehörigen Parameter fest.	
Voraussetzung	Eine Freigabe ist nur möglich, wenn mindestens an einem der beiden Scan-Kopf-Anschlüsse ein intelliSCAN (mit Scan-System-Firmware Version ≥ 2078) angeschlossen ist, der mit mindestens einem Sprungtuning ausgestattet ist. Ansonsten bleibt **set_jump_mode** wirkungslos.	
Aufruf	`ErrorCode = set_jump_mode( Flag, Length, VA1, VA2, VB1, VB2, JA1, JA2, JB1, JB2 )`	
Parameter	`Flag`	Umschaltflag. Als 32-Bit-Wert mit Vorzeichen. Zulässige Werte: = –1: Der Sprungmodus wird unterdrückt. Mit **set_jump_mode_list** ist anschließend *keine* Umschaltung des Flags mehr möglich. = 0: Der Sprungmodus wird freigegeben, aber deaktiviert. Anschließend ist auch mit **set_jump_mode_list** eine Umschaltung des Flags möglich. = 1: Der Sprungmodus wird freigegeben und aktiviert. Anschließend ist auch mit **set_jump_mode_list** eine Umschaltung des Flags möglich.
	`Length`	Sprunglängengrenze (je Achse), unterhalb derer 2D-Sprünge auch bei aktiviertem Sprungmodus im Vektormodus ausgeführt werden. Als 32-Bit-Wert ohne Vorzeichen.

	`VA1`
	`VA2`
	`VB1`
	`VB2`
	`JA1`
	`JA2`
	`JB1`
	`JB2`
		Nummern der Tunings, die bei der Sprungausführung im Sprungmodus verwendet werden sollen:
		V: Vektortuning, das am Ende eines 2D-Sprungs eingestellt wird.
		J: Sprungtuning, das am Anfang eines 2D-Sprungs eingestellt wird.
		A Anschluss für ersten Scan-Kopf.
		B Anschluss für zweiten Scan-Kopf.
		1: x-Achse (STATUS-Kanal, Galvanometerscanner 2).
		2: y-Achse (STATUS1-Kanal, Galvanometerscanner 1).
		Zulässige Werte:
		= – 1: Tuning wird weder überprüft noch verwendet.
		= 0…3: Tuning wird bei erfolgreich bestandener Überprüfung im Sprungmodus verwendet. Der zulässige Wertebereich hängt auch davon ab, mit wie vielen Tunings das angeschlossene Scan-System ausgestattet ist.
		Als 32-Bit-Wert mit Vorzeichen.

Kontrollbefehl	set_jump_mode
Rückgabe	Fehlercode. Als 32-Bit-Wert mit Vorzeichen. 0 Kein Fehler: Flag erfolgreich auf 0 umgeschaltet (Sprungmodus deaktiviert, Vektormodus aktiviert). 1 Kein Fehler: Flag erfolgreich auf 1 umgeschaltet (Sprungmodus aktiviert, Vektormodus deaktiviert). –1 Flag erfolgreich auf –1 umgeschaltet (Sprungmodus deaktiviert und unterdrückt). –2 Busy-Fehler: Karte BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus (**get_last_error**-Returncode `RTC6_BUSY`). –3 Karte reagiert nicht: kein Programm geladen oder PCI-Fehler (**get_last_error**-Returncode `RTC6_TIMEOUT`). –4 Access-Fehler: Karte für anderes Anwenderprogramm reserviert (**get_last_error**-Returncode `RTC6_ACCESS_DENIED`). > 1 Überprüfung nicht bestanden (siehe auch Hinweise). `Flag` wird auf –1 gesetzt (Sprungmodus deaktiviert und unterdrückt). Zurückgegeben wird: Byte #0 = 255 Byte #1 = Fehlercode für Anschluss für ersten Scan-Kopf Byte #2 = Fehlercode für Anschluss für zweiten Scan-Kopf Byte #3 = 0 Mit Fehlercode: =1: x-Achse (Galvanometerscanner 2) reagiert nicht oder kein intelliSCAN (mit Scan-System-Firmware Version ≥ 2078) angeschlossen. =2: y-Achse (Galvanometerscanner 1) reagiert nicht oder kein intelliSCAN (mit Scan-System-Firmware Version ≥ 2078) angeschlossen. =4: Keine Korrekturtabelle zugewiesen. =8: Falsche Tuningnummer(n): falscher Typ oder zum schnellen Umschalten ungeeignet.
Multi-Board-Befehlsname	**n_set_jump_mode**

Kontrollbefehl	set_jump_mode
Hinweise	• Zur Verwendung von **set_jump_mode** siehe Kapitel 8.1.5 “Sprungmodus”, Seite 229. • Eine Überprüfung (siehe auch Abschnitt “Voraussetzung und Aktivierung”, Seite 230) wird nur durchgeführt, wenn `Flag` vor Aufruf von **set_jump_mode** auf –1 stand (Initialisierungszustand) und/oder wenn die angegebenen Tuningnummern nicht mit den auf der Karte gespeicherten übereinstimmen. Anderenfalls wird nur das `Flag` umgeschaltet. Für die Prüfung darf die Karte weder BUSY-Listenausführungsstatus noch INTERNAL-BUSY-Listenausführungsstatus sein, da zwischenzeitlich der vom Scan-System zu übertragende Datentyp geändert und dazu die “Automatische Laser-Steuerung“ deaktiviert werden muss (beides wird am Befehlsende wieder hergestellt). Je nach Ausgang der Prüfung werden verschiedene Fehlercodes zurückgeliefert (s.o.). Im Fehlerfalle wird `Flag` immer auf –1 gesetzt (Sprungmodus deaktiviert und unterdrückt). Bei erfolgreich bestandener Überprüfung, kann dann zu einem späteren Zeitpunkt (auch während der Ausführung einer Liste mit **set_jump_mode_list**) zwischen den Zuständen `Flag` = 1 (Sprungmodus aktiviert, Vektormode deaktiviert) und `Flag` = 0 (Sprungmodus deaktiviert, Vektormode aktiviert) beliebig hin und her geschaltet werden, ohne dass eine Überprüfung erneut durchgeführt werden muss. • Als Tuningnummer ist –1 anzugeben, wenn bestimmte Tunings nicht überprüft werden sollen (z.B. weil kein intelli*SCAN* Scan-System angeschlossen ist oder weil gewisse Tunings nicht vorhanden sind, etwa kein Vektortuning für reine Bohr-Anwendungen) oder wenn nach der Umschaltung in das Sprungtuning keine Rückumschaltung mehr in ein Vektortuning gewünscht ist. In der Folge wird ein solches Tuning weder überprüft noch eingeschaltet. Ist die Option “Zweite-Scan-Kopf-Steuerung” nicht freigeschaltet, werden die Tuningnummern für den Anschluss für zweiten Scan-Kopf (B) automatisch auf –1 gesetzt (auch wenn andere angegeben wurden). Stehen alle Tuningnummern auf –1, wird `Flag` auf –1 gesetzt (Rückgabewert –1). • Auch nach einer erfolgreichen Freigabe und Aktivierung des Sprungmodus (`Flag` = 1), erfolgt die erste Reglerumschaltung erst mit dem ersten nachfolgenden 2D-Sprung, siehe Abschnitt “Funktionsweise”, Seite 229. Falls das dann aktuell eingestellte Tuning nicht mit dem unter **set_jump_mode** angegebenen Sprungtuning oder Vektortuning übereinstimmt, kann das erstmalige Umschalten je nach aktuell eingestelltem Tuning etwas länger (ca. 250 ms) dauern. Ob dies der Fall ist, kann aber zuvor mit Hilfe von **set_jump_mode** und dem aktuell eingestelltem Tuning als Parameter überprüft werden. Wenn ja (Rückgabewert > 1, Fehlercode = 2), dann sollte vor dem ersten 2D-Sprung das Tuning mit **control_command** auf eines der mit **set_jump_mode** angegebenen Tunings eingestellt werden, um den gewünschten Ablauf sicherzustellen. • **set_jump_mode** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 “Automatisches Booten – Ablauf im Detail”, Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_jump_mode_list**, **load_jump_table_offset**, **set_jump_table**

Normaler Listenbefehl	**set_jump_mode_list**
Funktion	Aktiviert und deaktiviert oder unterdrückt den Sprungmodus für 2D-Sprünge.
Voraussetzung	Siehe **set_jump_mode**.
Aufruf	`set_jump_mode_list( Flag )`
Parameter	`Flag` Siehe **set_jump_mode**.
Multi-Board-Befehlsname	**n_set_jump_mode_list**
Hinweise	• Zur Verwendung von **set_jump_mode_list** siehe Kapitel 8.1.5 "Sprungmodus", Seite 229. • **set_jump_mode_list** funktioniert wie der Kontrollbefehl **set_jump_mode** (siehe Hinweise dort), aber als Listenbefehl und mit den folgenden Unterschieden: – Bei **set_jump_mode_list** können keine Tunings und Sprunglängengrenze angegeben werden. – **set_jump_mode_list** führt keine Überprüfung durch. `Flag` muss zuvor durch **set_jump_mode** erfolgreich auf 0 oder 1 gesetzt worden sein. – Eine Deaktivierung oder Unterdrückung des Sprungmodus durch Setzen von `Flag` auf –1 ist sowohl mit **set_jump_mode** als auch **set_jump_mode_list** möglich, kann aber nur durch **set_jump_mode** (bei erfolgreich bestandener Prüfung) wieder aufgehoben werden. Stand `Flag` vor Aufruf von **set_jump_mode_list** auf –1, bleibt **set_jump_mode_list** wirkungslos.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_jump_mode**

Unverzögerter Kurzer Listenbefehl	**set_jump_speed**
Funktion	Definiert die Sprunggeschwindigkeit für Vektor-Befehle.
Aufruf	`set_jump_speed( Speed )`
Parameter	`Speed` Sprunggeschwindigkeit. In Bits/ms. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [1,6…800.000,0]. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_set_jump_speed**
Hinweise	• Standardmäßig ist eine Sprunggeschwindigkeit von 10.000 Bits/ms voreingestellt. • Die angegebene Sprunggeschwindigkeit wird für alle Sprung-Befehle verwendet, bis ein neuer Wert angegeben wird. • Die tatsächliche Sprunggeschwindigkeit v_{jump} in der Bildebene in m/s ergibt sich aus dem angegebenen `Speed`-Wert [Bits/ms] und dem Kalibrierfaktor K [Bits/mm] aus: $v_{jump} = \texttt{Speed} / K$ Der Kalibrierfaktor K kann mit **get_table_para** oder **get_head_para** aus der Korrekturtabelle ausgelesen werden. • **set_jump_speed** existiert auch als Kontrollbefehl **set_jump_speed_ctrl**.
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen `Speed` Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_abs**, **jump_rel**, **set_mark_speed**, **set_jump_speed_ctrl**

Kontrollbefehl	**set_jump_speed_ctrl**
Funktion	Wie **set_jump_speed**, aber ein Kontrollbefehl.
Aufruf	`set_jump_speed_ctrl( Speed )`
Parameter	`Speed` Wie **set_jump_speed**.
Multi-Board-Befehlsname	**n_set_jump_speed_ctrl**
Hinweise	• **set_jump_speed_ctrl** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist • **set_jump_speed_ctrl** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus: wie **set_jump_speed**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_jump_speed**, **set_mark_speed**, **set_mark_speed_ctrl**

Kontrollbefehl	**set_jump_table**
Funktion	Liest die an der angegebenen PC-Adresse gespeicherte Jump-Delay-Tabelle mit 1024 16-Bit-Werten ohne Vorzeichen und lädt sie als Jump-Delay-Tabelle auf die Karte (siehe Hinweise zu **get_jump_table**).
Aufruf	`ErrorCode = set_jump_table( Addr )`
Parameter	`Addr` PC-Adresse für den 2048 Byte großen Speicherbereich.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Kein Fehler. 1 Busy-Fehler: Karte BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus (**get_last_error**-Returncode `RTC6_BUSY`). 4 Verify-Fehler: DSP-Speicher-Fehler. 11 RTC6-Kartentreiber-Fehler.
Multi-Board-Befehlsname	**n_set_jump_table**
Hinweise	• **set_jump_table** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **set_jump_table** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt)
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_jump_table**, **load_jump_table_offset**

Kontrollbefehl	set_laser_control
Funktion	Lasersteuersignale definieren, freigegeben oder unterdrücken.
Aufruf	`set_laser_control( Ctrl )`
Parameter	`Ctrl` Steuerparameter. Als 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) Pulse Switch Setting (betrifft weder Laser-Mode 4 noch Laser-Mode 6). Die Einstellung betrifft nur diejenigen LASER1- oder LASER2-"Laser active"-Modulationspulse im CO_2-Modus oder LASER1-Q-Switch-Pulse in einem YAG-Modus), die am Ende des LASERON-Signals noch nicht vollständig abgearbeitet sind, siehe Abbildung 54 und Abbildung 55. = 0: Die Signale werden am Ende des LASERON-Signals abgeschnitten. = 1: Der letzte Puls wird trotz beendetem LASERON-Signals noch vollständig ausgeführt. Siehe Pulsvervollständigung, Seite 194. Bit #1 Phasenverschiebung der Lasersteuersignale (betrifft weder Laser-Mode 4 noch Laser-Mode 6). = 0: Keine Phasenverschiebung. = 1: Beim CO_2-Modus wird das LASER1-Signal mit dem LASER2-Signal vertauscht. Beim YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, YAG-Modus 5 wird das LASER1-Signal wird um eine halbe Signal-Periodendauer nach hinten phasenverschoben. Bit #2 Freigabe oder Unterdrückung der Signale für den "Laser active"-Betrieb. = 0: Die Signale für den "Laser active"-Betrieb werden freigegeben. = 1: Die Signale für den "Laser active"-Betrieb werden unterdrückt (die Signale werden auf ihren jeweiligen "Off"-Pegel gesetzt). Bit #3 Pegel von LASERON-Signal. = 0: Wird HIGH-aktiv gesetzt. = 1: Wird LOW-aktiv gesetzt. Erfolgt keine Änderung der Pin-Zuordnung mit **config_laser_signals**, entspricht das LASERON-Signal dem Signal am LASERON-Pin, siehe Abbildung 17. Bit #4 Pegel von LASER1-Signal und LASER2-Signal. = 0: Werden HIGH-aktiv gesetzt. = 1: Werden LOW-aktiv gesetzt. Erfolgt keine Änderung der Pin-Zuordnung mit **config_laser_signals**, entspricht das LASER1-Signal (LASER2-Signal) dem Signal am LASER1-Pin (LASER2-Pin), siehe Abbildung 17. Bit #5 Festlegung für **laser_on_pulses_list**, ob externe Signalpulse (am Digital-Eingang DIGITAL IN1 der LASER-Buchse) jeweils mit der steigenden oder mit der fallenden Flanke gezählt werden: = 0: Mit der fallenden Flanke. = 1: Mit der steigenden Flanke.

Kontrollbefehl	set_laser_control		
Parameter (Forts.)	Bit #6	= 0:	Die Synchronisation wird ausgeschaltet (Standardeinstellung). Siehe Kapitel 7.4.10 "Synchronisierung von RTC6-Takt und externem Taktsignal", Seite 219.
		= 1:	Die Synchronisation wird eingeschaltet.
	Bit #7	= 0:	Der "konstante Pulslänge"-Modus wird ausgeschaltet (Standardeinstellung).
		= 1:	Der "konstante Pulslänge"-Modus wird eingeschaltet. Siehe Kapitel 7.4.8 "Pulse-Picking-Laser-Modus", Seite 206 und **set_pulse_picking_length**.
	Bit #8	Reserviert.	
	Bit #9	Reserviert.	
	Bit #10	Reserviert.	
	Bit #11	Reserviert.	
	Bit #12	Reserviert.	
	Bit #13	Reserviert.	
	Bit #14	Reserviert.	
	Bit #15	Reserviert.	
	Bit #16	Für die automatische Unterdrückung der Lasersteuersignale wird verwendet: PowerOK von Kopf A, x-Achse.	
	Bit #17	Wie Bit #16, aber: TempOK von Kopf A, x-Achse.	
	Bit #18	Wie Bit #16, aber: PosAck von Kopf A, x-Achse.	
	Bit #19	Wie Bit #16, aber: PowerOK von Kopf A, y-Achse.	
	Bit #20	Wie Bit #16, aber: TempOK von Kopf A, y-Achse.	
	Bit #21	Wie Bit #16, aber: PosAck von Kopf A, y-Achse.	
	Bit #22	Wie Bit #16, aber: PowerOK von Kopf B, x-Achse.	
	Bit #23	Wie Bit #16, aber: TempOK von Kopf B, x-Achse.	
	Bit #24	Wie Bit #16, aber: PosAck von Kopf B, x-Achse.	
	Bit #25	Wie Bit #16, aber: PowerOK von Kopf B, y-Achse.	
	Bit #26	Wie Bit #16, aber: TempOK von Kopf B, y-Achse.	
	Bit #27	Wie Bit #16, aber: PosAck von Kopf B, y-Achse.	
	Bit #28	= 1:	Im Fehlerfall wird bei einer automatischen Überwachung (bei der automatischen Lasersteuersignal-Unterdrückung) auch automatisch ein /STOP-Signal erzeugt (Liste hält an, Lasersteuersignale gehen dauerhaft aus).
	Bit #29	= 1:	Im Fehlerfall gemäß Bit #28 wird das **stop_execution** als /Master-STOP (siehe Abbildung 72) an alle Master/Slave-verbundenen RTC6-Karten weitergeleitet. Siehe **master_slave_config**.
	Bit #30	Reserviert.	
	Bit #31	Reserviert.	
Multi-Board-Befehlsname	**n_set_laser_control**		

Kontrollbefehl	set_laser_control
Hinweise	• In der Standardeinstellung (nach **load_program_file**) sind alle Bits auf 0 gesetzt. Nach einem Hardware-Reset werden die Einstellungen erst durch einen mindestens einmaligen Aufruf von **set_laser_control** wirksam. Vorher befinden sich alle Lasersteuersignal-Ausgänge (LASERON, LASER1 und LASER2) im hochohmigen Zustand. TTL-Zustände (LOW oder HIGH) stellen sich erst dann ein, wenn **set_laser_control** aufgerufen und dabei der gewünschte TTL-Pegel festgelegt wird, siehe auch Kapitel 7.4 "Laser-Steuerung", Seite 192. Auch nach **load_program_file**, das die Lasersteuersignale deaktiviert, muss **set_laser_control** zur erstmaligen Aktivierung aufgerufen werden. • Bei den Vorgängerversionen der RTC5/RTC6 werden die Lasersteuersignal-Pegel über Lötjumper definiert. Bei der RTC5/RTC6 werden sie durch den Benutzer selbst komplett softwaregesteuert. Mit **get_startstop_info** kann der aktuelle Status der Lasersteuersignale (Bit #9) ausgelesen werden und ob die Signale HIGH-aktiv oder LOW-aktiv eingestellt sind (Bit #13). • Das Freigeben und Unterdrücken der Lasersteuersignale kann auch mit **enable_laser** oder **disable_laser** durchgeführt werden. Mit **get_startstop_info** (Bit #14) kann der aktuelle Zustand abgefragt werden. • Auch wenn die Lasersteuersignale mit **set_laser_control** oder **enable_laser** freigegeben wurden, werden sie ohne weitere Befehle noch nicht an den Ausgängen ausgegeben, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192. • Die Phasenverschiebung der Lasersteuersignale (Bit #1 = 1) kann zur besseren Synchronisation einer Analog-Ausgabe eingesetzt werden z.B. im Softstart-Modus (noch nicht implementiert) oder im Pixelausgabemodus, siehe Kapitel 8.7 "Pixelausgabemodus", Seite 275. • **set_laser_control** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 617, OUT 617, RBF 623: Unterstützung von **get_startstop_info** (Bit #14).
Verweise	**set_laser_mode**, **config_laser_signals**

Unverzögerter Kurzer Listenbefehl	**set_laser_delays**
Funktion	Definiert das LaserOn-Delay und das LaserOff-Delay.
Aufruf	`set_laser_delays( LaserOnDelay, LaserOffDelay )`
Parameter	`LaserOnDelay` LaserOn-Delay. Als 32-Bit-Wert mit Vorzeichen. 1 Bit entspricht 1/64 μs. Zulässiger Wertebereich: [-2^{31}…+(2^{21}–1)]. Werte über (2^{21}–1) werden geclippt.
	`LaserOffDelay` LaserOff-Delay. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs. Zulässiger Wertebereich: [0…+(2^{21}-1)]. Werte über (2^{21}–1) werden geclippt.
Multi-Board-Befehlsname	**n_set_laser_delays**
Hinweise	• Die Delays können innerhalb der zulässigen Bereiche beliebig gewählt werden. Für `LaserOffDelay` < `LaserOnDelay` wird ein "Überkreuzen" von Laser-On und Laser-Off bei kurzen Vektoren intern verhindert, siehe Abschnitt "Automatische Delay-Anpassung", Seite 158. • Beachten Sie die Hinweise in Kapitel 7.2.1 "Laser-Delays", Seite 148. • Ein negativer `LaserOnDelay`-Wert erhöht die Gesamtmarkierungszeit. • Der XY2-100-Konverter (Zubehör) verursacht bei der Scan-System-Ansteuerung eine Laufzeitverzögerung von 10 μs. Durch Vergrößern des LaserOn-Delays und des LaserOff-Delays um jeweils 10 μs wird diese Laufzeitverzögerung ausgeglichen. • Die Default-Einstellung nach **load_program_file** entspricht `set_laser_delays( 640, 640 )`.
RTC4→RTC6	Grundsätzlich gleiche Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Delay-Werte mit 64. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `LaserOnDelay` und `LaserOffDelay` mit 32. Entsprechend verringern sich die zulässigen Wertebereiche.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_delays_ctrl**, **set_scanner_delays**

Kontrollbefehl	**set_laser_delays_ctrl**
Funktion	Wie **set_laser_delays**, aber ein Kontrollbefehl.
Aufruf	`set_laser_delays_ctrl( LaserOnDelay, LaserOffDelay )`
Parameter	`LaserOnDelay` Wie **set_laser_delays**.
	`LaserOffDelay` Wie **set_laser_delays**.
Rückgabe	–
Multi-Board-Befehlsname	**n_set_laser_delays_ctrl**
Hinweise	• Siehe **set_laser_delays**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	**set_laser_delays**

Kontrollbefehl	**set_laser_mode**
Funktion	Stellt den Laser-Mode der RTC6 ein.
Aufruf	`set_laser_mode( Mode )`
Parameter	`Mode` = 0: CO_2-Modus. = 1: YAG-Modus 1. = 2: YAG-Modus 2. = 3: YAG-Modus 3. = 4: Laser-Mode 4. = 5: YAG-Modus 5. = 6: Laser-Mode 6. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_laser_mode**
Hinweise	• Welche Lasersteuersignale verfügbar sind, hängt vom eingestellten Laser-Mode ab, siehe auch Kapitel 7.4 "Laser-Steuerung", Seite 192. • **set_laser_mode** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Zusätzliche Laser-Modi z.B. YAG-Modus 5, Laser-Mode 6, Pulse-Picking. Sonst unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_control**, **set_laser_pulses_ctrl**, **set_laser_pulses**, **set_laser_timing**, **set_firstpulse_killer**, **set_firstpulse_killer_list**, **set_standby**, **set_standby_list**, **set_qswitch_delay**, **set_qswitch_delay_list**

Kontrollbefehl	set_laser_off_default	
Funktion	Stellt die Default-Ausgabewerte für die Ausgänge ANALOG OUT1 und ANALOG OUT2, sowie der 8-Bit-Digital-Ausgang ein.	
Aufruf	`set_laser_off_default( AnalogOut1, AnalogOut2, DigitalOut )`	
Parameter	`AnalogOut1`	12-Bit-Ausgabewert für den analogen Ausgang ANALOG OUT1, siehe auch Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77. Als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert. Ausnahme: `AnalogOut1/AnalogOut2 =` "–1", siehe Hinweise bei **set_port_default**.
	`AnalogOut2`	12-Bit-Ausgabewert für den analogen Ausgang ANALOG OUT2. Siehe `AnalogOut1`.
	`DigitalOut`	8-Bit-Ausgabewert für den 8-Bit-Digital-Ausgang, siehe auch Abschnitt "8-Bit-Digital-Ausgang", Seite 84. Als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert. Ausnahme: `DigitalOut=` "–1", siehe Hinweise bei **set_port_default**.
Multi-Board-Befehlsname	**n_set_laser_off_default**	
Hinweise	• Die Defaultwerte können auch mit **set_port_default** festgelegt werden. Siehe auch alle Hinweise dort.	
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 `AnalogOut1` und `AnalogOut2` jeweils mit 4.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**set_port_default**	

Kontrollbefehl	**set_laser_pin_out**
Funktion	Schreibt einen Wert auf die zwei digitalen Ausgänge der LASER-Buchse.
Aufruf	`set_laser_pin_out( Pins )`
Parameter	`Pins` Ausgabewert (DIGITAL OUT1 und DIGITAL OUT2). Als 32-Bit-Wert ohne Vorzeichen. Bit # 0: DIGITAL OUT1. Bit # 1: DIGITAL OUT2. Bit # 2: Reserviert. … Bit # 31:Reserviert.
Multi-Board-Befehlsname	**n_set_laser_pin_out**
Hinweise	• Siehe auch Kapitel 9.1.3 "2-Bit-Digital-Ausgang", Seite 307.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_pin_out_list**, **get_laser_pin_in**

Unverzögerter Kurzer Listenbefehl	**set_laser_pin_out_list**
Funktion	Wie **set_laser_pin_out**, aber ein Listenbefehl.
Aufruf	`set_laser_pin_out_list( Pins )`
Parameter	`Pins` Wie **set_laser_pin_out**.
Multi-Board-Befehlsname	**n_set_laser_pin_out_list**
Hinweise	• Siehe **set_laser_pin_out**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_pin_out**, **write_port_list**

Unverzögerter Kurzer Listenbefehl	**set_laser_power**
Funktion	Synchronisiert die Ausgaben für die Laserleistung an den Ausgängen (ANALOG OUT1, ANALOG OUT2, 8-Bit-Digital-Ausgang, 16-Bit-Digital-Ausgang) und die Laser-Delays.
Aufruf	`set_laser_power( Port, Power )`
Parameter	`Port` — Ausgang, für den ein Laserleistungswert `Power` definiert werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: ANALOG OUT1-Ausgang. = 1: ANALOG OUT2-Ausgang. = 2: 8-Bit-Digital-Ausgang (EXTENSION 2-Stiftleiste). = 3: 16-Bit-Digital-Ausgang (EXTENSION 1-Stiftleiste). > 3: **set_laser_power** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	`Power` — Gewünschter Laserleistungswert. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: Für `Port` = 0: 12-Bit-Werte [0…4.095]. Für `Port` = 1: 12-Bit-Werte [0…4.095]. Für `Port` = 2: 8-Bit-Werte [0…255]. Für `Port` = 3: 16-Bit-Werte [0…65.535]. Überlaufende Werte werden auf die Randwerte geclippt. **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
Multi-Board-Befehlsname	**n_set_laser_power**
Hinweise	• **set_laser_power** ist insbesondere dann zu verwenden, wenn mit einem SCANahead-System innerhalb eines Polygonzuges die Laserleistung geändert werden soll. • **set_laser_power** ist auch notwendig, wenn z.B. *kurze Vektoren* unterschiedliche Laserleistungen benötigen und die Laser-Delays über den nächsten Vektor hinweg reichen (insbesondere im DSP-Modus 3).
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen `Power`-Wert für `Port` = 0 und `Port` = 1 mit 4. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 602, OUT 602, RBF 602.
Verweise	**get_last_error**, **set_dsp_mode**, **set_rtc4_mode**, **write_da_x**

Kontrollbefehl	**set_laser_pulse_sync**
Funktion	Schaltet den Puls-Synchronisations-Modus ein (aus).
Aufruf	`set_laser_pulse_sync( Mode, Delay )`
Parameter	`Mode` = 0: Schaltet den Puls-Synchronisations-Modus aus. > 0: Schaltet den Puls-Synchronisations-Modus ein. Als 32-Bit-Wert ohne Vorzeichen.
	`Delay` Verzögerung. In 1/64 µs. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_laser_pulse_sync**
Hinweise	• Siehe auch Kapitel 7.4.11 "Puls-Synchronisations-Modus", Seite 220. • Siehe auch Abschnitt ""Spot Distance Control"", Seite 213. • **set_laser_pulse_sync** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 625, OUT 625, RBF 628.
Verweise	–

Verzögerter Kurzer Listenbefehl	**set_laser_pulses**
Funktion	Definiert die Ausgabeperiode und die Pulslänge der Lasersteuersignale LASER1 und LASER2 für den “Laser active”-Betrieb.
Aufruf	`set_laser_pulses( HalfPeriod, PulseLength )`
Parameter	`HalfPeriod` *Halbe* Ausgabeperiode. In Bits. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs. Zulässiger Wertebereich: [0…(2^{32}–1)].
	`PulseLength` Pulslänge der Lasersteuersignale LASER1 und LASER2. In Bits. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_set_laser_pulses**
Hinweise	• Mit dem Parameter `HalfPeriod` wird die halbe Periodendauer angegeben, siehe Abbildung 54 und Abbildung 55. • Wenn `HalfPeriod` und/oder `PulseLength` den Wert 0 haben, werden keine Lasersteuersignale ausgegeben. • Bei `PulseLength` ≥ (2 × `HalfPeriod`) bleibt der Laser dauernd an. • Der Signalpegel wird mit **set_laser_control** eingestellt. • **set_laser_pulses** existiert auch als Kontrollbefehl **set_laser_pulses_ctrl**. • **set_laser_pulses** ist weitgehend identisch mit dem RTC4-Befehl **set_laser_timing**, hat aber weniger Parameter.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `HalfPeriod` und `PulseLength` mit 8. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_pulses_ctrl**, **set_laser_timing**

Kontrollbefehl	**set_laser_pulses_ctrl**
Funktion	Wie **set_laser_pulses**, aber ein Kontrollbefehl.
Aufruf	`set_laser_pulses_ctrl( HalfPeriod, PulseLength )`
Parameter	`HalfPeriod` Wie **set_laser_pulses**.
	`PulseLength` Wie **set_laser_pulses**.
Multi-Board-Befehlsname	**n_set_laser_pulses_ctrl**
Hinweise	• Siehe **set_laser_pulses**.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus: wie **set_laser_pulses**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_pulses**, **set_laser_timing**

Verzögerter Kurzer Listenbefehl	**set_laser_timing**
Funktion	Definiert die Ausgabeperiode und die Pulslänge der Lasersteuersignale LASER1 und LASER2 für den “Laser active”-Betrieb.
Aufruf	`set_laser_timing( HalfPeriod, PulseLength1, (PulseLength2), TimeBase )`
Parameter	`HalfPeriod` — *Halbe* (Q-Switch-) Ausgabeperiode. In Bits. Als 32-Bit-Wert ohne Vorzeichen. • Im RTC6-Standardmodus: 1 Bit entspricht 1/64 µs. Zulässiger Wertebereich: [0…(2^{32}–1)]. • Im RTC4-Kompatibilitätsmodus: *1 Bit entspricht 1/8 µs oder 1 µs*, abhängig von der gewählten Taktfrequenz. Der Wert wird entsprechend der angegebenen `TimeBase` in ganzzahlige Vielfache von 1/64 µs umgerechnet. Der zulässige Bereich ist entsprechend verkleinert.
	`PulseLength1` — Pulslänge der Lasersteuersignale LASER1 und LASER2. In Bits. Als 32-Bit-Wert ohne Vorzeichen. • Im RTC6-Standardmodus: 1 Bit entspricht 1/64 µs. Zulässiger Wertebereich: [0…(2^{32}–1)]. • Im RTC4-Kompatibilitätsmodus: *1 Bit entspricht 1/8 µs oder 1 µs*, abhängig von der gewählten Taktfrequenz. Der Wert wird entsprechend der angegebenen `TimeBase` in ganzzahlige Vielfache von 1/64 µs umgerechnet. Der zulässige Bereich ist entsprechend verkleinert.
	`(PulseLength2)` — Wert wird nicht verwendet. (Als 32-Bit-Wert ohne Vorzeichen.) Die Pulslänge der Lasersteuersignale LASER1 und LASER2 ist bei der RTC6 immer identisch und wird durch `PulseLength1` festgelegt.
	`TimeBase` — Als 32-Bit-Wert ohne Vorzeichen. • Im RTC6-Standardmodus wird der Wert ignoriert und die Taktfrequenz fest auf 64 MHz gesetzt. 1 Bit entspricht 1/64 µs. • Im RTC4-Kompatibilitätsmodus wird der Wert von `TimeBase` wie folgt berücksichtigt: = 0: setzt die Taktfrequenz auf 1 MHz. *1 Bit entspricht 1 µs*. ≠ 0: setzt die Taktfrequenz auf 8 MHz. *1 Bit entspricht 1/8 µs*.
Multi-Board-Befehlsname	**n_set_laser_timing**

Verzögerter Kurzer Listenbefehl	**set_laser_timing**
Hinweise	• Im RTC6-Standardmodus ist **set_laser_timing** gleichbedeutend mit **set_laser_pulses**. Siehe Hinweise dort (zu `HalfPeriod`, `PulseLength`). • Die Taktfrequenz-Einstellungen gelten *nur* für die Parameter von **set_laser_timing**. • Im RTC4-Kompatibilitätsmodus ist es im Allgemeinen empfehlenswert, eine Taktfrequenz von 8 MHz zu einzustellen. Eine Taktfrequenz von 1 MHz sollte nur eingestellt werden, wenn unbedingt nötig. • Beachten Sie auch die Hinweise in Kapitel 7.4 "Laser-Steuerung", Seite 192. • Im RTC4-Kompatibilitätsmodus ist für Laser-Mode 4 und Laser-Mode 6 die Zeitbasis für die Signale LASER1 und LASER2 unabhängig von `TimeBase` immer 1/8 µs.
RTC4→RTC6	• Die Pulslängen der Lasersteuersignale LASER1 und LASER2 sind bei der RTC6 immer identisch. • Taktfrequenzen: – Im RTC6-Standardmodus: 64 MHz, fest – Im RTC4-Kompatibilitätsmodus: 1 MHz oder 8 MHz
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_pulses_ctrl**, **set_laser_pulses**, **set_laser_mode**, **set_firstpulse_killer**, **set_firstpulse_killer_list**, **set_standby**, **set_standby_list**

Normaler Listenbefehl	**set_laser_timing_table**	
Funktion	Kategorie: Listenbefehle nur für einen begrenzten Benutzerkreis. Setzt einen Wert in der Lookup-Tabelle für die Laserfrequenz.	
Aufruf	`set_laser_timing_table( Index, F )`	
Parameter	`Index`	Index in der Lookup-Tabelle für die Laserfrequenz. Zulässiger Wertebereich: 0…10. Als 32-Bit-Wert ohne Vorzeichen.
	`F`	Laserfrequenz. In Hz. Zulässiger Wertebereich: 65…100.000. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_laser_timing_table**	
Hinweise	• Es gelten alle Hinweise von **set_duty_cycle_table**. • Für Listenbefehle nur für einen begrenzten Benutzerkreis gilt: Diese Benutzer wurden im Umgang mit diesen RTC6-Befehlen separat informiert. Deswegen wird nur die Befehlstabelle aufgenommen und auf weitere Informationen in diesem Handbuch verzichtet.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 626, OUT 627.	
Verweise	**set_duty_cycle_table**, **regulation3**	

Unverzögerter Kurzer Listenbefehl	**set_list_jump**
Funktion	Bewirkt bei der Ausführung einen unbedingten Sprung auf die angegebene Adresse des RTC6-Listenspeichers. Der dortige nächste Befehl wird sofort und ohne Verzögerung ausgeführt.
Aufruf	`set_list_jump( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0…(2^{23}–1)]. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_list_jump**
Hinweise	• **set_list_jump** ist gleichbedeutend mit **list_jump_pos**. Siehe Hinweise dort.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_jump_pos**

Verzögerter Kurzer Listenbefehl	**set_mark_speed**
Funktion	Stellt die Markiergeschwindigkeit für die Vektor-Befehle und Bogen-Befehle ein.
Aufruf	`set_mark_speed( Speed )`
Parameter	`Speed` Markiergeschwindigkeit. In Bits/ms. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [1,6…800.000,0]. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_set_mark_speed**
Hinweise	• Standardmäßig ist eine Markiergeschwindigkeit von 1000 *Bits/ms* voreingestellt. • Die angegebene Markiergeschwindigkeit wird für alle [*]mark[*]-Befehle und Bogen-Befehle verwendet, bis ein neuer Wert angegeben wird. • Die tatsächliche Markiergeschwindigkeit v_{mark} in der Bildebene in m/s ergibt sich aus dem angegebenen `Speed`-Wert [Bits/ms] und dem Kalibrierfaktor K [Bits/mm] aus: $v_{mark} = \texttt{Speed} / K$ Der Kalibrierfaktor K kann mit **get_table_para** oder **get_head_para** aus der Korrekturtabelle ausgelesen werden. • **set_mark_speed** existiert auch als Kontrollbefehl **set_mark_speed_ctrl**.
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Speed` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_abs**, **mark_rel**, **set_jump_speed**, **set_mark_speed_ctrl**

Kontrollbefehl	**set_mark_speed_ctrl**
Funktion	Wie **set_mark_speed**, aber ein Kontrollbefehl.
Aufruf	`set_mark_speed_ctrl( Speed )`
Parameter	`Speed` Wie **set_mark_speed**.
Multi-Board-Befehlsname	**n_set_mark_speed_ctrl**
Hinweise	• **set_mark_speed_ctrl** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist • **set_mark_speed_ctrl** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • Siehe auch **set_sky_writing_min_speed_ctrl**.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus: wie **set_mark_speed**
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mark_speed**, **set_jump_speed**, **set_jump_speed_ctrl**, **set_sky_writing_min_speed_ctrl**

Kontrollbefehl	set_matrix	
Funktion	Definiert die Koeffizienten der • Allgemeinen Transformationsmatrix M_T für Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236 • Globalen Transformationsmatrix im virtuellen Bildfeld, Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 177	
Aufruf	`set_matrix( HeadNo, M11, M12, M21, M22, at_once )`	
Parameter	`HeadNo`	Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. = 1: Die Definition wirkt nur auf den Anschluss für ersten Scan-Kopf. = 2: Die Definition wirkt nur auf den Anschluss für zweiten Scan-Kopf. = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. = 4: Die Definition wirkt auf das virtuelle Bildfeld (siehe auch Hinweis). Es werden nur die drei niederwertigsten Bits ausgewertet.
	`M11`	Matrixkoeffizient der allgemeinen Transformationsmatrix M_T. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–50…+50] im realen Bildfeld und [–2,0…+2,0] im virtuellen Bildfeld. Bei unzulässigem Wert wird **set_matrix** ignoriert.
	`M12`	Wie `M11` (analog).
	`M21`	Wie `M11` (analog).
	`M22`	Wie `M11` (analog).
	`at_once`	Legt fest, wann die definierte Transformation wirksam wird. Als 32-Bit-Wert ohne Vorzeichen. Für `HeadNo` = 0…3 gilt: = 0: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten Listenbefehls berechnet und auf die dann aktuelle Position angewendet. = 1: Die neue Gesamttransformation wird sofort (oder vor dem nächsten Listenbefehl wenn gerade der BUSY-Listenausführungsstatus oder der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist) berechnet und auf die aktuelle Position angewendet. Zuvor werden die Signale für den "Laser active"-Betrieb ausgeschaltet. = 2: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten **jump_abs**, **jump_rel**, **goto_xy** oder **goto_xyz** berechnet und auf die dann aktuelle Position angewendet. = 3: Wie `at_once` = 1, aber die Lasersteuersignale bleiben unverändert. > 3: Wie `at_once` = 2. Für `HeadNo` = 4 gilt: = 0: Nur gespeichert. Wird beim nächsten Start einer "Processing on the fly"-Session auf die aktuelle Position angewendet. = 1: Wird sofort auf die aktuelle Position angewendet, wenn gerade keine Liste BUSY-Listenausführungsstatus oder keine Karte INTERNAL-BUSY-Listenausführungsstatus ist. Sonst wie 0. = 2, 3: Wie 0.
Multi-Board-Befehlsname	**n_set_matrix**	

Kontrollbefehl	set_matrix
Hinweise	• Bis DLL 613, OUT 613 gilt für `HeadNo` = 4: – Die globalen Koordinatentransformationen sind nur während einer "Processing on the fly"-Session verfügbar, die mit **set_fly_2d** gestartet wurde – `at_once` wird ignoriert • Ab DLL 614, OUT 614 gilt für `HeadNo` = 4: – Die globalen Koordinatentransformationen sind allgemein verfügbar, auch außerhalb einer "Processing on the fly"-Session – Wenn Koordinatentransformationen lediglich gespeichert wurden, dann werden sie beim nächsten Start einer "Processing on the fly"-Session angewendet • **set_matrix**(`HeadNo` = 4) ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Unveränderte Grundfunktion (Definition der Transformationsmatrix), allerdings: • Die Parameter `HeadNo` und `at_once` sind neu. • Verringerter Wertebereich für die Koeffizienten. • Siehe auch Abschnitt "Hinweise für RTC4-Benutzer", Seite 239.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 614, OUT 614: `HeadNo` = 4 allgemein verfügbar.
Verweise	**set_matrix_list**, **set_angle**, **set_offset**, **set_scale**

Variabler Listenbefehl	**set_matrix_list**
Funktion	Ändert *einen* der 4 Koeffizienten der allgemeinen Transformationsmatrix M_T während der Ausführung einer Liste, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236.
Aufruf	`set_matrix_list( HeadNo, Ind1, Ind2, Mij, at_once )`
Parameter	`HeadNo` Wie **set_matrix**. Jedoch ist `HeadNo` = 4 nicht verfügbar. `Ind1` `Ind2` Zeilenindex und Spaltenindex des zu ändernden Matrixkoeffizienten. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [Index ungerade: 1, Index gerade: 2]. `Mij` Matrixkoeffizient. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–50…+50]. Bei einem unzulässigen Wert wird **set_matrix_list** durch **list_nop** ersetzt. `at_once` Legt fest, wann die definierte Transformation wirksam wird. Als 32-Bit-Wert ohne Vorzeichen. = 0: Die Transformationswerte werden nur gesammelt und zwischengespeichert, aber die Transformation wird nicht ausgeführt, solange dies nicht durch eine andere Koordinatentransformation (z.B. mit einem Listenbefehl mit `at_once` = 1 oder einem entsprechenden Kontrollbefehl) aktiviert wird. = 1: Die Transformation wird zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und noch vor dem nächsten Listenbefehl ausgeführt. Die Signale für den "Laser active"-Betrieb werden zuvor ausgeschaltet. = 2: Die Transformationswerte werden (wie bei `at_once` = 0) nur gesammelt und zwischengespeichert, die Transformation wird aber bei der Ausführung des nächsten **jump_abs** oder **jump_rel** (nur wenn keine Liste ausgeführt wird: auch **goto_xy** oder **goto_xyz**) zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewendet. = 3: Wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: Wie `at_once` = 2.
Multi-Board-Befehlsname	**n_set_matrix_list**
Hinweise	• **set_matrix_list** erlaubt zu einem bestimmten Zeitpunkt nur die Änderung *eines* der 4 Koeffizienten. Um während der Ausführung einer Liste mehrere Koeffizienten zu ändern, muss **set_matrix_list** wiederholt aufgerufen werden. Es empfiehlt sich dabei, die ersten Aufrufe mit `at_once` = 0 und nur den letzten Aufruf mit `at_once` = 1 auszuführen. • Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 236.
RTC4→RTC6	Siehe **set_matrix**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_matrix**

Kontrollbefehl	**set_max_counts**
Funktion	Definiert die maximale Zahl an externen Starts.
Aufruf	`set_max_counts( Counts )`
Parameter	`Counts` Maximale Zahl an externen Starts. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_set_max_counts**
Hinweise	• Sobald die angegebene maximale Zahl an externen Starts erreicht worden ist, wird die externe Start-Eingabe deaktiviert (siehe **set_control_mode**, Bit #0 = 0). • Für `Counts` = 0 bleibt die Zahl der externen Starts unbegrenzt. • Bei der Initialisierung der RTC6 (**load_program_file**) wird `Counts` auf 0 gesetzt. • Die aktuelle Anzahl Externer Starts kann mit **get_counts** aus dem entsprechenden internen Zähler ausgelesen werden. Der Zähler kann mit **set_control_mode** zurückgesetzt werden. • **set_max_counts** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_counts**, **set_control_mode**

Kontrollbefehl	**set_mcbsp_freq**
Funktion	Legt die Sendefrequenz der McBSP-Schnittstelle fest.
Aufruf	`mcbsp_freq = set_mcbsp_freq( Freq )`
Parameter	`Freq` Gewünschte Sendefrequenz der McBSP-Schnittstelle in Hz. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [4.000.000…16.000.000] (4…16 MHz). Überlaufende Werte werden auf die Randwerte geclippt. Bei überlaufenden Werten wird ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt.
Rückgabe	Die tatsächlich eingestellte Frequenz in Hz. Als 32-Bit-Wert ohne Vorzeichen. Bei überlaufenden Werten wird 0 zurückgegeben.
Multi-Board-Befehlsname	**n_set_mcbsp_freq**
Hinweise	• Default-Wert der Sendefrequenz (nach Initialisierung) ist 8 MHz (`Freq` = 8.000.000). • Da nicht alle Frequenzen auch realisierbar sind, gibt **set_mcbsp_freq** die tatsächlich eingestellte Frequenz zurück. Beispiel: `mcbsp_freq = set_mcbsp_freq(7.000.000);` liefert `mcbsp_freq = 7.200.000`. • Die neue Sendefrequenz wird erst dann wirksam, wenn die McBSP-Schnittstelle mit **mcbsp_init** neu initialisiert wird. • Die Signale und Betriebsbedingungen der McBSP-Schnittstelle sind in Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87 dargestellt. • Hinweis: Die Empfangsfrequenz wird ausschließlich aus den ankommenden Clock-Pulsen bestimmt und darf maximal 16 MHz betragen. • **set_mcbsp_freq** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mcbsp_init**, **set_mcbsp_out**, **set_mcbsp_out_ptr**

Kontrollbefehl	**set_mcbsp_global_matrix**
Funktion	Aktiviert eine Matrix-Korrektur für die "Globale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_global_matrix()`
Multi-Board-Befehlsname	**n_set_mcbsp_global_matrix**
Hinweise	• Siehe auch Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243. • Eine Matrix-Korrektur kann nicht zusammen mit Offset- und/oder Rotationskorrekturen verwendet werden. Entsprechende zuvor aktivierte Optionen werden durch **set_mcbsp_global_matrix** deaktiviert. Das nachträgliche Aktivieren anderer Optionen (mit **set_mcbsp_global_x**, **set_mcbsp_global_y** oder **set_mcbsp_global_rot**) deaktiviert die Matrix-Option. • Für die über die McBSP-Schnittstelle zu übertragenden Matrixkoeffizienten gelten die gleichen Einschränkungen wie bei **set_matrix**`(HeadNo = 4, …)`: – Der erlaubte Wertebereich für die Matrixkoeffizienten ist [–2,0…+2,0]. – Übertragene Koeffizienten, die diesen Bereich überschreiten, werden ignoriert. • Der Benutzer muss jeden Matrixkoeffizient Mij der Transformationsmatrix M_T einzeln, ganzzahlig normiert und unter Angabe der zugehörigen Indices i, j als McBSP-Eingabewert M_{in} an die McBSP-Schnittstelle wie folgt übergeben: M_{in} = (integer(Mij * 2^{28}) << 2) + (i << 1) + j mit M_T = { M00, M01, M10, M11 } = { m_{11}, m_{12}, m_{21}, m_{22} }. Umgekehrt ermittelt die RTC6 einen Koeffizienten wie folgt aus dem Eingabewert: M_T[M_{in} & 0x3] = (M_{in} >> 2) / 2^{28}. • Die Koeffizienten werden an die interne Speicherstelle 1 übertragen und können von dort zur Kontrolle mit `read_mcbsp(1)` ausgelesen werden. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_matrix_list**, **set_mcbsp_matrix**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_global_matrix_list**
Funktion	Wie **set_mcbsp_global_matrix**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_global_matrix_list()`
Multi-Board-Befehlsname	**n_set_mcbsp_global_matrix_list**
Hinweise	• Siehe **set_mcbsp_global_matrix**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_matrix**

Kontrollbefehl	**set_mcbsp_global_rot**
Funktion	Aktiviert oder deaktiviert eine Rotations-Korrektur für die "Globale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_global_rot( Resolution )`
Parameter	`Resolution` Als Wert im 64-Bit-IEEE Gleitkommaformat. 2^{-26} < `Resolution` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert), anderenfalls: Korrektur wird deaktiviert. `Resolution` = McBSP-Bits pro Vollkreis.
Multi-Board-Befehlsname	**n_set_mcbsp_global_rot**
Hinweise	• Siehe auch Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243. • Bei einem McBSP-Eingabewert Rot_{in} für die Rotations-Korrektur wirkt **set_mcbsp_global_rot** wie `set_angle(4, Angle × 360°,…)` mit `Angle` (in Vollkreisen) = Rot_{in} / `Resolution`. Es sind nur `Angle`-Werte im Bereich [0,0…+20,0 Vollkreise] zulässig. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_rot_list**, **set_mcbsp_rot**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_global_rot_list**
Funktion	Wie **set_mcbsp_global_rot**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_global_rot_list( Resolution )`
Parameter	`Resolution` Wie **set_mcbsp_global_rot**.
Multi-Board-Befehlsname	**n_set_mcbsp_global_rot_list**
Hinweise	• Siehe **set_mcbsp_global_rot**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_rot**

Kontrollbefehl	**set_mcbsp_global_x**
Funktion	Aktiviert oder deaktiviert eine x-Offset-Korrektur für die "Globale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_global_x( Scale )`
Parameter	`Scale` Als Wert im 64-Bit-IEEE Gleitkommaformat. 2^{-26} < `Scale` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert), anderenfalls: Korrektur wird deaktiviert.
Multi-Board-Befehlsname	**n_set_mcbsp_global_x**
Hinweise	• Siehe auch Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243. • Bei einem McBSP-Eingabewert X_{in} für die x-Offset-Korrektur wirkt **set_mcbsp_global_x** wie `set_offset_xyz(4, XOffset,…)` mit `XOffset` (in Bits) = `Scale` × X_{in}. `XOffset`-Werte außerhalb von [–524.288…+524.287] werden auf den Randwert geclippt, wenn `Scale` × X_{in} den Wertebereich $\pm 2^{31}$ nicht überschreitet. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_x_list**, **set_mcbsp_x**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_global_x_list**
Funktion	Wie **set_mcbsp_global_x**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_global_x_list( Scale )`
Parameter	`Scale` Wie **set_mcbsp_global_x**.
Multi-Board-Befehlsname	**n_set_mcbsp_global_x_list**
Hinweise	• Siehe **set_mcbsp_global_x**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_x**

Kontrollbefehl	**set_mcbsp_global_y**
Funktion	Aktiviert oder deaktiviert eine y-Offset-Korrektur für die "Globale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_global_y( Scale )`
Parameter	`Scale` Als Wert im 64-Bit-IEEE Gleitkommaformat. 2^{-26} < `Scale` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert), anderenfalls: Korrektur wird deaktiviert.
Multi-Board-Befehlsname	**n_set_mcbsp_global_y**
Hinweise	• Siehe auch Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243. • Bei einem McBSP-Eingabewert Y_{in} für die y-Offset-Korrektur wirkt **set_mcbsp_global_y** wie `set_offset_xyz(4, …, YOffset,…)` mit `YOffset` (in Bits) = `Scale` × Y_{in}. `YOffset`-Werte außerhalb von [–524.288…+524.287] werden auf den Randwert geclippt, wenn `Scale` × Y_{in} den Wertebereich $\pm 2^{31}$ nicht überschreitet. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_y_list**, **set_mcbsp_y**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_global_y_list**
Funktion	Wie **set_mcbsp_global_y**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_global_y_list( Scale )`
Parameter	`Scale` Wie **set_mcbsp_global_y**.
Multi-Board-Befehlsname	**n_set_mcbsp_global_y_list**
Hinweise	• Siehe **set_mcbsp_global_y**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**set_mcbsp_global_y**

Kontrollbefehl	**set_mcbsp_in**
Funktion	Aktiviert die "Processing on the fly"-Korrektur zur Kompensation von Bewegungen des Werkstücks oder Scan-Systems gemäß den Positionswerten, die über die McBSP-Schnittstelle an die RTC6 übergeben werden. Die McBSP-Schnittstelle kann zusätzlich zur Eingabe von beliebigen anderen Signalen verwendet werden.
Einschränkung	Falls die Option "Processing on the fly" nicht freigeschaltet ist, schaltet **set_mcbsp_in** den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_mcbsp_in( Mode, Scale )`
Parameter	`Mode` Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Die "Processing on the fly"-Korrektur wird abgeschaltet. = 1: Kompensation linearer Bewegungen in x-Richtung. = 2: Kompensation linearer Bewegungen in y-Richtung. = 3: Kompensation linearer Bewegungen in x-Richtung und y-Richtung. = 4: Kompensation von Rotationsbewegungen. = 5: Die "Processing on the fly"-Korrektur wird abgeschaltet. • `Mode` = 0…5: Alle McBSP-Eingabewerte werden alternierend an die internen Speicherstellen 1 und 2 übertragen. • `Mode` = 1…5 (dagegen nicht für `Mode` = 0): Mit "Bit #31 = 0" codierte McBSP-Eingabewerte werden zusätzlich an die interne Speicherstelle 0 übertragen und mit "Bit #31 = 1" codierte McBSP-Eingabewerte an die interne Speicherstelle 3. • `Mode` = 1…4: Die an die interne Speicherstelle 0 übertragenen Werte werden für die "Processing on the fly"-Korrektur verwendet.
	`Scale` Skalierungsfaktor oder Rotationsauflösung. Als Wert im 64-Bit-IEEE Gleitkommaformat. • `Mode` = 1…3: Skalierungsfaktor in *(RTC6-)Bits/(McBSP-)Bit*. Zulässiger Wertebereich: 1/256 ≤ |`Scale`| ≤ 16.000,0 (für `Mode` = 3 gilt `Scale` für beide Achsen). • `Mode` = 4: Zählschritte (Counts) pro Umdrehung. Zulässiger Wertebereich: |`Scale`| > 100,0.
Multi-Board-Befehlsname	**n_set_mcbsp_in**
Hinweise	• Die internen Speicherstellen können jederzeit mit **read_mcbsp** ausgelesen werden. • Zur "Processing on the fly"-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.6 ""Processing on the fly"", Seite 254. • Unterschiedliche "Processing on the fly"-Korrekturen sind nicht beliebig kombinierbar, siehe Abschnitt "Übersicht", Seite 254.

Kontrollbefehl	set_mcbsp_in
Hinweise (Forts.)	• Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Kapitel 8.6.5 ““Processing on the fly”-Korrektur deaktivieren”, Seite 263. • Für 2D-Korrekturen (`Mode` = 3) stehen effektiv jeweils 15 Bit (mit Vorzeichen) pro Achse zur Verfügung. Dabei steht der x-Wert in den unteren 16 Bit, der y-Wert in den oberen 16 Bit des mit “Bit #31 = 0” codierten McBSP-Eingabewertes. • Die McBSP-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden Siehe auch Abschnitt ”Hinweise”, Seite 242. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Seite 89. • Für `Mode` > 5 wird **set_mcbsp_in** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Für unzulässige `Scale`-Werte (z.B. `Scale` = 0) wirkt **set_mcbsp_in** wie für `Mode` = 0. • Die “Processing on the fly”-Korrekturen sind ab RTC6-Software-Paket V1.6.1 kompatibel mit den Einstellungen aus Kapitel 8.6.12 ““Fly-Erweiterung”-Befehle”, Seite 271. Sie können nachträglich überschrieben werden, z.B. auch mit Encoder-basierten “Processing on the fly”-Korrekturen. Die McBSP-Werte werden dann nicht für “Processing on the fly”-Korrekturen verwendet. Die oben beschriebene McBSP-Speicherübertragung kann jedoch nicht verändert werden. Falls andere als die gemäß `Mode` eingestellten McBSP-basierten “Processing on the fly”-Korrekturen eingestellt werden, ist der Benutzer dafür verantwortlich, die McBSP-Daten im passenden Format zu übertragen. Eine für die z-Achse aktivierte Encoder-basierte “Processing on the fly”-Korrektur bleibt erhalten.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Letzte Änderung DLL 617, OUT 617: Kompatibilität mit “Fly-Erweiterung”-Befehlen.
Verweise	**set_mcbsp_in_list**

Normaler Listenbefehl	**set_mcbsp_in_list**
Funktion	Wie **set_mcbsp_in**, aber ein Listenbefehl.
Einschränkung	Falls die Option “Processing on the fly“ nicht freigeschaltet ist, schaltet **set_mcbsp_in_list** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_mcbsp_in_list( Mode, Scale )`
Parameter	`Mode` Wie **set_mcbsp_in**.
	`Scale` Wie **set_mcbsp_in**.
Multi-Board-Befehlsname	**n_set_mcbsp_in_list**
Hinweise	• Für `Mode` > 5 wird **set_mcbsp_in_list** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Siehe **set_mcbsp_in**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Letzte Änderung DLL 617, OUT 617: Kompatibilität mit “Fly-Erweiterung”-Befehlen.
Verweise	**set_mcbsp_in**

Kontrollbefehl	**set_mcbsp_matrix**
Funktion	Aktiviert eine Matrix-Korrektur für die "Lokale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_matrix()`
Multi-Board-Befehlsname	**n_set_mcbsp_matrix**
Hinweise	• Zur "Lokale Online-Positionierung" siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240. • Eine Matrix-Korrektur kann nicht zusammen mit Offset- und/oder Rotationskorrekturen verwendet werden. Entsprechende zuvor aktivierte Optionen werden durch **set_mcbsp_matrix** deaktiviert. Das nachträgliche Aktivieren anderer Optionen (mit **set_mcbsp_x**, **set_mcbsp_y** oder **set_mcbsp_rot**) deaktiviert die Matrix-Korrektur. • Für die über die McBSP-Schnittstelle zu übertragenden Matrixkoeffizienten gelten die gleichen Einschränkungen wie bei **set_matrix**: Der Wertebereich für die Matrixkoeffizienten ist [–50…+50]; bei einer Bereichsüberschreitung wird der jeweils übertragene Koeffizient ignoriert. • Der Benutzer muss jeden Matrixkoeffizient Mij der Transformationsmatrix M_T einzeln, ganzzahlig normiert und unter Angabe der zugehörigen Indices i, j als McBSP-Eingabewert M_{in} an die McBSP-Schnittstelle wie folgt übergeben: M_{in} = (Ganzzahl(Mij * 2^{24}) << 2) + (i << 1) + j mit M_T = { M00, M01, M10, M11 } = { m_{11}, m_{12}, m_{21}, m_{22} }. Umgekehrt ermittelt die RTC6 einen Koeffizienten wie folgt aus dem Eingabewert: M_T[M_{in} & 0x3] = (M_{in} >> 2) / 2^{24}. • Jeder übertragene Matrixkoeffizient muss separat mit **apply_mcbsp** oder **apply_mcbsp_list** abgeholt werden. Es empfiehlt sich, die ersten Koeffizienten mit `at_once` = 0 abzuholen und nur den letzten mit `at_once` > 0. • Die Koeffizienten werden an die interne Speicherstelle 1 übertragen und können von dort zur Kontrolle mit `read_mcbsp(1)` ausgelesen werden. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_matrix_list**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_matrix_list**
Funktion	Wie **set_mcbsp_matrix**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_matrix_list()`
Multi-Board-Befehlsname	**n_set_mcbsp_matrix_list**
Hinweise	• See **set_mcbsp_matrix**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_matrix**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_out**
Funktion	Definiert zwei Signaltypen für die Ausgabe an der McBSP-Schnittstelle, siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87.
Aufruf	`set_mcbsp_out( Signal1, Signal2 )`
Parameter	`Signal1` Auszugebender Signaltyp. Als 32-Bit-Wert ohne Vorzeichen.
	`Signal2` Auszugebender Signaltyp. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_mcbsp_out**
Hinweise	• Die auswählbaren Signaltypen sind identisch zu **set_trigger** (für den zulässigen Wertebereich, die Signaltypen und entsprechende Hinweise siehe dort). Bei unzulässigen Angaben für `Signal1`/`Signal2` wird **set_mcbsp_out** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Die beiden ausgewählten Datensignale werden ständig (einmal in jedem 10 µs-Takt) ausgegeben. • Von den ausgewählten Datensignalen wird nur jeweils je ein 16-Bit-Anteil in einem gemeinsamen 32-Bit-Datenwort ausgegeben. Dabei belegt `Signal1` die untere Hälfte und `Signal2` die obere Hälfte des 32-Bit-Datenworts. – Bei Sample-Werten und bei vom Scan-System übertragenen Statuswerten (`Signal1`, `Signal2` = 1…23, 25…30) werden nur die RTC4-kompatiblen Bit #4…Bit #19 ausgegeben. Die unteren 4 Bits sowie die aus dem virtuellen Bildfeld überlaufenden Bits werden ignoriert. – Bei den anderen Datentypen (`Signal1`, `Signal2` = 0, 24, 31…57) werden jeweils die unteren 16 Bit ausgegeben, die oberen Bits werden ignoriert – McBSP_Output = ((Out2 & 0x0000FFFF) << 16) | (Out1 & 0x0000FFFF)); • Die übertragenen Werte sind jeweils vom vorhergehenden Takt: Die Daten werden am Ende eines Zyklus aufbereitet und am Anfang des darauffolgenden Zyklus mit der eingestellten Sendefrequenz ausgegeben (siehe **set_mcbsp_freq**). • Die Signale und Betriebsbedingungen der McBSP-Schnittstelle sind in Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87 dargestellt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**read_mcbsp**, **mcbsp_init**, **set_mcbsp_freq**, **set_mcbsp_out_ptr**, **set_trigger**

Kontrollbefehl	**set_mcbsp_out_oie_ctrl**
Funktion	Schaltet den OIE-Ausgabemodus an der McBSP-Schnittstelle ein.
Aufruf	`set_mcbsp_out_oie_ctrl( Signal1, Signal2 )`
Parameter	`Signal1` Auszugebender Signaltyp. Als 32-Bit-Wert ohne Vorzeichen.
	`Signal2` Auszugebender Signaltyp. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_mcbsp_out_oie_ctrl**
Hinweise	• OIE-Ausgabemodus An der McBSP-Schnittstelle wird ständig (= einmal in jedem 10 µs-Takt) ein 32-Bit-Datenwort der folgenden Struktur ausgegeben: – Bit #00…Bit #11 Freie Variable 0, geclippt auf 12 Bit. – Bit #12…Bit #13 0. – Bit #14…Bit #21 `Signal1`, skaliert auf [–128…+127]. – Bit #22…Bit #29 `Signal2`, skaliert auf [–128…+127]. – Bit #30 BUSY-Listenausführungsstatus, siehe **get_status**. – Bit #31 LASERON-Signal. • Die ausgewählten Signaltypen für `Signal1` und `Signal2` sind identisch zu **set_trigger**. Für beide Signaltypen sollte ein gültiges Positionssignal (mit Wertebereich [–524.288…+524.287]) ausgewählt sein. • **set_mcbsp_out_oie_ctrl** wird bei einem unzulässigen Parameter-Wert nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 633, OUT 634.
Verweise	**set_mcbsp_out_oie_list**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_out_oie_list**
Funktion	Wie **set_mcbsp_out_oie_ctrl**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_out_oie_list( Signal1, Signal2 )`
Parameter	`Signal1` Wie **set_mcbsp_out_oie_ctrl**.
	`Signal2` Wie **set_mcbsp_out_oie_ctrl**.
Multi-Board-Befehlsname	**n_set_mcbsp_out_oie_list**
Hinweise	• Siehe **set_mcbsp_out_oie_ctrl**. • Bei einem unzulässigen Parameter-Wert wird **set_mcbsp_out_oie_list** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 633, OUT 634.
Verweise	**set_mcbsp_out_oie_ctrl**

Kontrollbefehl	**set_mcbsp_out_ptr**
Funktion	Definiert eine Liste von bis zu 8 Signaltypen für die Ausgabe an der McBSP-Schnittstelle, siehe Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87.
Aufruf	`set_mcbsp_out_ptr( Number, SignalPtr )`
Parameter	`Number` Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…8]. = 1…8: Anzahl der auszugebenden Signaltypen. = 0 Die Ausgabe an der McBSP-Schnittstelle erfolgt gemäß Voreinstellung oder gemäß Auswahl mit einem vorherigen **set_mcbsp_out**. Bit 31 = 0: Signale werden als 24-Bit-Werte ausgegeben. Bit 31 = 1: Signale werden als 23-Bit-Werte ausgegeben. Das höchstwertige Bit enthält immer den LASERON-Status.
	`SignalPtr` Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf ein Array von `Number` 32-Bit-Werten ohne Vorzeichen, in dem die auszugebenden `Number` Signaltypnummern gespeichert sind.
Multi-Board-Befehlsname	**n_set_mcbsp_out_ptr**
Hinweise	• Der Speicherbereich für das `SignalPtr`-Array muss vom Anwenderprogramm zur Verfügung gestellt werden. • Für `Number` > 8 und/oder `SignalPtr` = NULL wird **set_mcbsp_out_ptr** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Die auswählbaren Signaltypen sind identisch zu **set_trigger** (für den zulässigen Wertebereich, die Signaltypen und entsprechende Hinweise siehe dort). • Die bis zu 8 ausgewählten Datentypen werden sequenziell (je ein Datentyp in jedem 10 µs-Takt) ausgegeben. Jedes einzelne Signal gehört also zu einem anderen Takt. Die übertragenen Werte sind jeweils vom vorhergehenden Takt. Die Liste wird so lange wiederholt, bis die Ausgabe ausgeschaltet wird oder durch eine andere Liste ersetzt wird. • Jedem Signalwert wird die Signaltypnummer als unterstes Byte im 32-Bit-Datenwort hinzugefügt. Der eigentliche Signalwert wird daher um 8 Bit nach links geshiftet. Dabei werden alle Bits größer als 24 abgeschnitten (Überlauf, kein Clipping, relevant nur für Signaltypen 24, 31 und 37…57): 32-Bit-Ausgabewert = (Signalwert << 8) | (Signaltypnummer & 0xFF). • Die Signale und Betriebsbedingungen der McBSP-Schnittstelle sind in Kapitel 4.6.6 "McBSP/ANALOG-Stiftleiste", Seite 87 dargestellt. • `Number`-Bit 31 wird z.B. für die OIE PID-Steuerung benötigt. • Für `Number`-Bit 31 = 1 gilt: An der McBSP-Schnittstelle wird der LASERON-Status 10 µs später ausgegeben, als wie mit **set_trigger**(`Signal1` = 0) aufgezeichnet. • **set_mcbsp_out_ptr** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.

Kontrollbefehl	set_mcbsp_out_ptr
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 632, OUT 633. `Number`-Bit 31.
Verweise	**set_mcbsp_out_ptr_list**, **set_mcbsp_out**, **set_trigger**

Multipler Listenbefehl	**set_mcbsp_out_ptr_list**
Funktion	Wie **set_mcbsp_out_ptr**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_out_ptr_list( Number, SignalPtr )`
Parameter	`Number` Wie **set_mcbsp_out_ptr**.
	`SignalPtr` Wie **set_mcbsp_out_ptr**.
Multi-Board-Befehlsname	**n_set_mcbsp_out_ptr_list**
Hinweise	• **set_mcbsp_out_ptr_list** benötigt für `Number` ≥ 1 zwei RTC6-Listenspeicherplätze. Ausführungsreihenfolge: 1. Anliegende Verzögerte Kurze Listenbefehle 2. Erster Befehlsteil als Unverzögerter Kurzer Listenbefehl 3. Zweiter Befehlsteil als Normaler Listenbefehl • Siehe **set_mcbsp_out_ptr**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 632, OUT 633.
Verweise	**set_mcbsp_out_ptr**

Kontrollbefehl	**set_mcbsp_rot**
Funktion	Aktiviert oder deaktiviert eine Rotations-Korrektur für die "Lokale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_rot( Resolution )`
Parameter	`Resolution` Als Wert im 64-Bit-IEEE Gleitkommaformat. 2^{-26} < `Resolution` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert), sonst: Korrektur wird deaktiviert. `Resolution` = McBSP-Bits pro Vollkreis.
Multi-Board-Befehlsname	**n_set_mcbsp_rot**
Hinweise	• Zur "Lokale Online-Positionierung" siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240. • Bei einem McBSP-Eingabewert Rot_{in} für die Rotations-Korrektur wirkt **apply_mcbsp** wie `set_angle( …, Angle × 360°,…)` mit Angle in Vollkreisen = Rot_{in} / `Resolution` Es sind nur `Angle`-Werte im Bereich [0,0…+20,0 Vollkreisen] zulässig. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_angle**, **set_mcbsp_rot_list**, **set_mcbsp_x**, **set_mcbsp_y**, **apply_mcbsp**

Unverzögerter Kurzer Listenbefehl	**set_mcbsp_rot_list**
Funktion	Wie **set_mcbsp_rot**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_rot_list( Resolution )`
Parameter	`Resolution` Wie **set_mcbsp_rot**.
Multi-Board-Befehlsname	**n_set_mcbsp_rot_list**
Hinweise	• Siehe **set_mcbsp_rot**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_rot**

Kontrollbefehl	set_mcbsp_x
Funktion	Aktiviert oder deaktiviert eine x-Offset-Korrektur für die "Lokale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_x( Scale )`
Parameter	`Scale` Als Wert im 64-Bit-IEEE Gleitkommaformat. 2^{-26} < `Scale` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert). Sonst: Korrektur wird deaktiviert.
Multi-Board-Befehlsname	**n_set_mcbsp_x**
Hinweise	• Zur "Lokale Online-Positionierung" siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240. • Bei einem McBSP-Eingabewert X_{in} für die x-Offset-Korrektur wirkt **apply_mcbsp** wie `set_offset( …, XOffset,…)` mit `XOffset` (in Bits) = `Scale` × X_{in} `XOffset`-Werte außerhalb von [–524.288…+524.287] werden auf den Randwert geclippt, wenn `Scale` × X_{in} den Wertebereich $\pm 2^{31}$ nicht überschreitet. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_x_list**, **set_mcbsp_y**, **set_mcbsp_rot**, **apply_mcbsp**

Unverzögerter Kurzer Listenbefehl	set_mcbsp_x_list
Funktion	Wie **set_mcbsp_x**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_x_list( Scale )`
Parameter	`Scale` Wie **set_mcbsp_x**.
Multi-Board-Befehlsname	**n_set_mcbsp_x_list**
Hinweise	• Siehe **set_mcbsp_x**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_x**

Kontrollbefehl	set_mcbsp_y
Funktion	Aktiviert oder deaktiviert eine y-Offset-Korrektur für die "Lokale Online-Positionierung" über die McBSP-Schnittstelle.
Aufruf	`set_mcbsp_y( Scale )`
Parameter	`Scale` Als Wert im 64-Bit-IEEE Gleitkommaformat. 2^{-26} < `Scale` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert). Sonst: Korrektur wird deaktiviert.
Multi-Board-Befehlsname	**n_set_mcbsp_y**
Hinweise	• Zur "Lokale Online-Positionierung" siehe Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240. • Bei einem McBSP-Eingabewert Y_{in} für die y-Offset-Korrektur wirkt **apply_mcbsp** wie `set_offset( …, YOffset,…)` mit `YOffset` (in Bits) = `Scale` × Y_{in} `YOffset`-Werte außerhalb von [–524.288…+524.287] werden auf den Randwert geclippt, wenn `Scale` × Y_{in} den Wertebereich $\pm 2^{31}$ nicht überschreitet. • Die McBSP-Schnittstelle kann nicht gleichzeitig für eine Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden. Siehe auch Abschnitt "Hinweise", Seite 242.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_y_list**, **set_mcbsp_x**, **set_mcbsp_rot**, **apply_mcbsp**

Unverzögerter Kurzer Listenbefehl	set_mcbsp_y_list
Funktion	Wie **set_mcbsp_y**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_y_list( Scale )`
Parameter	`Scale` Wie **set_mcbsp_y**.
Multi-Board-Befehlsname	**n_set_mcbsp_y_list**
Hinweise	• Siehe **set_mcbsp_y**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mcbsp_y**

Kontrollbefehl	**set_multi_mcbsp_in**
Funktion	Aktiviert eine “Processing on the fly”-Anwendung und eine Mehrfach-Übertragung über die McBSP-Schnittstelle mit bis zu 8 verschiedenen Datentypen.
Einschränkung	Falls die Option “Processing on the fly” nicht freigeschaltet ist, schaltet **set_multi_mcbsp_in** den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_multi_mcbsp_in( Ctrl, P, Mode )`
Parameter	`Ctrl` Steuerparameter zur Initialisierung oder Deaktivierung der Laserleistungsvariation. Als 32-Bit-Wert ohne Vorzeichen. = 1…6: definiert welcher Signalparameter variiert wird. Zur Bedeutung siehe **set_auto_laser_control**. = 0 oder > 6: deaktiviert die Laserleistungsvariation (für `Ctrl` > 6: **get_last_error**-Returncode `RTC6_PARAM_ERROR`). `P` Initialisierungswert für die Laserleistung. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: siehe **set_auto_laser_control**. `Mode` Als 32-Bit-Wert ohne Vorzeichen. = 0: Der übertragene Wert wird direkt verwendet. = 1: Der übertragene Wert wird mit `P`/16384 multipliziert und dann ausgegeben.
	`Ctrl`, `P` und `Mode` sind nur für den Typ 3 (= Laserleistung) des übertragenen Datenworts relevant, siehe **read_multi_mcbsp**.
Multi-Board-Befehlsname	**n_set_multi_mcbsp_in**
Hinweise	• Bis V1.6.0 gilt: Jede andere bereits aktivierte McBSP-Übertragung für eine Online-Positionierung und jede andere aktivierte “Processing on the fly”-Anwendung mit Encodersignalen oder mit Positionsangaben wird zuvor beendet. Ab V1.6.1 gilt: Die “Processing on the fly”-Korrekturen sind kompatibel mit den “Fly-Erweiterung”-Befehlen. Die “Processing on the fly”-Korrekturen der 3 Achsen werden entsprechend überschrieben. Sie können nachträglich auch wieder durch andere Korrekturen überschrieben werden, z.B. Encoder-basierte “Processing on the fly”-Korrekturen. Die Datentypen 0…2 werden dann nicht für “Processing on the fly”-Korrekturen verwendet. Dadurch kann die Übertragung von Extra-Parametern auch mit Encoder-basierten “Processing on the fly”-Korrekturen kombiniert werden. Die McBSP-Speicherübertragung kann jedoch nicht verändert werden. • **set_multi_mcbsp_in** aktiviert eine Eingabemöglichkeit über die McBSP-Schnittstelle, bei der bis zu 8 verschiedene Typen asynchron zum 10 µs-Takt übertragen werden können. Die Daten werden alle 10 µs gemäß ihrer Typ-Codierung in einen separaten Speicher kopiert, von wo aus sie auch über **read_multi_mcbsp** ausgelesen werden können. • Um fehlerhafte Zuordnungen zu vermeiden, sollte während der Erteilung von **set_multi_mcbsp_in** keine McBSP-Übertragung mehr aktiv sein. • Die McBSP-Schnittstelle ignoriert das erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init**. Die bereitgestellten Daten werden also nicht übertragen, siehe Seite 89.

Kontrollbefehl	set_multi_mcbsp_in
Hinweise (Forts.)	• Für die Übertragung muss die Typ-Zuordnung in die 3 niederwertigsten Bits des Datenworts codiert werden, gemäß: – McBSPValue = (Value << 3) | Type Die RTC6-Karte speichert das Datenwort gemäß: – Memory[McBSPValue & 0x7] = (long) McBSPValue >> 3 • Zur Typ-Zuordnung siehe **read_multi_mcbsp**. • Die übrigen (oberen) 29 Bits stehen für das eigentliche Datenwort zur Verfügung. Es gibt keine weiteren Einschränkungen als die typgemäßen Wertebereiche selbst. Die übertragenen Werte werden nicht auf ihre Bereichsgrenzen überprüft. Gegebenenfalls erfolgt ein Clipping oder ein Datenüberlauf. • Die 4 Extraparameter sind nicht mit den freien Variablen identisch, siehe Kapitel 6.9.1 "Freie Variablen", Seite 138, obgleich sie für ähnliche Zwecke verwendbar sind. Sie werden beim Programmstart mit 0 initialisiert und bleiben ansonsten durch diesen Befehl unbeeinflusst. Die übertragenen Werte werden lediglich in den typsortierten Speicher kopiert. • Bei mehr als 4 abgeschlossenen McBSP-Übertragungen innerhalb eines 10 µs-Taktes können vorherige Werte überschrieben werden. • Wenn die Option "Processing on the fly" freigeschaltet ist, aktiviert **set_multi_mcbsp_in** eine "Processing on the fly"-Anwendung mit Positionsangaben für die drei Koordinatenrichtungen x, y und z. Die deren Typ zugeordneten Speicherwerte werden mit 0 initialisiert. • Zusätzlich kann die Laserleistung variiert werden, indem der übertragene Wert vom Typ 3 an den mit dem `Ctrl`-Parameter verbundenen Port ausgegeben wird. Der Initialisierungswert `P` wird sofort ausgegeben. • Für `Mode` = 0 werden nachfolgende Typ 3-Übertragungen direkt am über `Ctrl` zugeordneten Port ausgegeben. Für `Mode` = 1 wird der übertragene Wert als Multiplikationsfaktor mit der Normierung 1.0 = 16384 (14 Bit) interpretiert. Ausgegeben wird also: (`P` × der übertragene Wert / 16384). Dieser `Mode` ist eine Alternative zur Laserleistungsvariation bei "Frei definierbaren Wobbelfiguren". • Diese Methode der Laserleistungsvariation kann mit der Vektorkontrollierte Laser-Steuerung (mit parametrisierten Befehlen) kombiniert werden, jedoch nicht mit anderen Methoden der "Automatischen Laser-Steuerung". Sie überschreibt die anderen Variationen, wenn der jeweilige `Ctrl`-Parameter derselbe ist. Unterschiedliche `Ctrl`-Parameter sind zwar zulässig, dürften aber kaum einen praktischen Sinn ergeben. • Für `Ctrl` = 0 wird die Laserleistungsvariation abgeschaltet. Die Werte aus der McBSP-Übertragung werden zwar weiterhin in den typsortierten Speicher kopiert, aber nicht ausgegeben. • Sonderfall: Wenn eine "Frei definierbare Wobbelfigur" aktiv ist, wird `P` immer (unabhängig vom Parameter `Mode`) als relative Laserleistung interpretiert und multiplikativ mit der Laserleistung aus der Wobbelfigur verknüpft (siehe **set_wobbel_vector**). Da die Wobbelfigur ihren eigenen Signalparameter definiert (siehe **set_wobbel_control**), ist der `Ctrl`-Parameter aus diesem Befehl dafür irrelevant. Er sollte z.B. durch `Ctrl` = 0 ungültig gesetzt werden, damit keine doppelten und unsinnigen Ausgaben erfolgen.

Kontrollbefehl	set_multi_mcbsp_in
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 617, OUT 617: Kompatibilität mit "Fly-Erweiterung"-Befehlen.
Verweise	**set_multi_mcbsp_in_list**, **read_multi_mcbsp**, **set_wobbel_control**, **set_wobbel_vector**

Normaler Listenbefehl	set_multi_mcbsp_in_list
Funktion	Wie **set_multi_mcbsp_in**, aber ein Listenbefehl.
Einschränkung	Wie **set_multi_mcbsp_in**.
Aufruf	`set_multi_mcbsp_in_list( Ctrl, P, Mode )`
Parameter	`Ctrl` Wie **set_multi_mcbsp_in**. `P` Wie **set_multi_mcbsp_in**. `Mode` Wie **set_multi_mcbsp_in**.
Multi-Board-Befehlsname	**n_set_multi_mcbsp_in_list**
Hinweise	• Siehe **set_multi_mcbsp_in**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 617, OUT 617: Kompatibilität mit "Fly-Erweiterung"-Befehlen.
Verweise	**set_multi_mcbsp_in**, **read_multi_mcbsp**, **set_wobbel_control**, **set_wobbel_vector**

Variabler Listenbefehl	**set_n_pixel**
Funktion	Führt im Pixelausgabemodus den `PortOutValue1` und `PortOutValue2` zugeordneten **set_pixel**-Befehl `Number`-mal unmittelbar aufeinander folgend aus.
Aufruf	`set_n_pixel( PortOutValue1, PortOutValue2, Number )`
Parameter	`PortOutValue1` Wie **set_pixel**.
	`PortOutValue2` Wie **set_pixel**.
	`Number` Anzahl der Pixel. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…(2^{32}–1)]. 0 wird automatisch auf 1 gesetzt.
Multi-Board-Befehlsname	**n_set_n_pixel**
Hinweise	• Zur Verwendung von **set_n_pixel** siehe auch Kapitel 8.7 "Pixelausgabemodus", Seite 275. • Vor dem ersten **set_n_pixel** einer Zeile muss **set_pixel_line** aufgerufen worden sein. • **set_n_pixel** definiert die Parameter für die nachfolgenden `Number` (identische) **set_pixel**-Befehle einer Bildzeile. Zur Verwendung siehe Hinweise zu **set_pixel**. • Soll nur ein einzelner Pixel definiert werden, kann **set_pixel** alternativ zu **set_n_pixel** verwendet werden. **set_pixel** ist gleichbedeutend mit `set_n_pixel ( Number = 1 )`. • Außerhalb des Pixelausgabemodus (wenn **set_n_pixel** kein **set_pixel_line**, **set_pixel** oder **set_n_pixel** unmittelbar vorausgegangen ist) ist **set_n_pixel** ein Kurzer Listenbefehl und wird ansonsten ignoriert. Unter Umständen kann ein **list_continue** eingefügt werden, siehe Abschnitt "Normaler Listenbefehl, Kurzer Listenbefehl, Variabler Listenbefehl und Multipler Listenbefehl", Seite 339.
RTC4→RTC6	Neuer Befehl. Zum RTC4-Kompatibilitätsmodus: siehe **set_pixel**.
RTC5→RTC6	Unveränderte Funktionalität. Siehe **set_pixel**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_pixel**, **set_pixel_line**, **set_pixel_line_3d**

Kontrollbefehl	**set_offset**
Funktion	Definiert: • Den Offset für Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236 • Den globalen Offset im virtuellen Bildfeld, siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 177
Aufruf	`set_offset( HeadNo, XOffset, YOffset, at_once )`
Parameter	`HeadNo` Siehe **set_offset_xyz**.
	`XOffset` Siehe **set_offset_xyz**.
	`YOffset` Siehe **set_offset_xyz**.
	`at_once` Siehe **set_offset_xyz**.
Multi-Board-Befehlsname	**n_set_offset**
Hinweise	• **set_offset** wirkt wie **set_offset_xyz**, lässt jedoch `ZOffset` unverändert.
RTC4→RTC6	Unveränderte Grundfunktion (Offset-Definition). Siehe **set_offset_xyz**.
RTC5→RTC6	Unveränderte Funktionalität. Siehe **set_offset_xyz**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_offset_list**, **set_offset_xyz**, **set_angle**, **set_matrix**, **set_scale**

Variabler Listenbefehl	**set_offset_list**
Funktion	Wie **set_offset**, aber ein Listenbefehl.
Aufruf	`set_offset_list( HeadNo, XOffset, YOffset, at_once )`
Parameter	`HeadNo` Wie **set_offset_xyz**. Jedoch ist `HeadNo` = 4 nicht verfügbar.
	`XOffset` Wie **set_offset_xyz**.
	`YOffset` Wie **set_offset_xyz**.
	`at_once` Wie **set_offset_xyz_list**.
Multi-Board-Befehlsname	**n_set_offset_list**
Hinweise	• –
RTC4→RTC6	Siehe **set_offset_xyz**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_offset**, **set_offset_xyz_list**

Kontrollbefehl	**set_offset_xyz**
Funktion	Definiert: • Den Offset für Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236 • Den globalen Offset im virtuellen Bildfeld, siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 177
Aufruf	`set_offset_xyz( HeadNo, XOffset, YOffset, ZOffset, at_once )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. = 1: Die Definition wirkt nur auf den Anschluss für ersten Scan-Kopf. = 2: Die Definition wirkt nur auf den Anschluss für zweiten Scan-Kopf. = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. = 4: Die Definition wirkt auf das virtuelle Bildfeld. Es werden nur die drei niederwertigsten Bits ausgewertet.
	`XOffset` Offset für die x-Richtung (Verschiebung der Koordinaten gegenüber dem Ursprung des kartesischen Koordinatensystems). In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–268.435.456…+268.435.455]. Überlaufende Werte werden auf die Randwerte geclippt.
	`YOffset` Wie `XOffset` (analog).
	`ZOffset` Offset für die z-Richtung (Verschiebung der Koordinaten gegenüber dem Ursprung des kartesischen Koordinatensystems). In Bits. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–524.288…+524.287]. Überlaufende Werte werden auf die Randwerte geclippt.
	`at_once` Wie **set_matrix**.
Multi-Board-Befehlsname	**n_set_offset_xyz**
Hinweise	• `ZOffset` wird nur einmal für beide Scan-Kopf-Anschlüsse gemeinsam gespeichert (`HeadNo` wird dafür ignoriert). • `ZOffset` ≠ 0 bewirkt eine Verschiebung der Arbeitsebene (entgegengesetzt der Richtung des Laserstrahls). Ein positiver Wert vergrößert die z-Koordinate, beispielsweise für den Ansteuerwert (0|0|0) um d = `ZOffset` / *K* in die Ebene z = +d. Zum Kalibrierfaktor *K* siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175; außerdem auch #3 und #5 in Kapitel 7.3.6 "Ausgabewerte an das Scan-System", Seite 189. • Für `HeadNo` = 4 wird `ZOffset` ignoriert. • Für `at_once`, siehe **set_matrix**. • **set_offset_xyz**(`HeadNo` = 4) ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.

Kontrollbefehl	set_offset_xyz
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `XOffset`, `YOffset`, `ZOffset` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den für `ZOffset` angegebenen Wert mit 16. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_offset_xyz_list**, **set_offset**, **set_angle**, **set_matrix**, **set_scale**, **set_defocus**

Variabler Listenbefehl	**set_offset_xyz_list**
Funktion	Wie **set_offset_xyz**, aber ein Listenbefehl.
Aufruf	`set_offset_xyz_list( HeadNo, XOffset, YOffset, ZOffset, at_once )`
Parameter	`HeadNo` Wie **set_offset_xyz**. Jedoch ist `HeadNo` = 4 nicht verfügbar.
	`XOffset` Wie **set_offset_xyz**.
	`YOffset` Wie **set_offset_xyz**.
	`ZOffset` Wie **set_offset_xyz**.
	`at_once` Wie **set_matrix_list**.
Multi-Board-Befehlsname	**n_set_offset_xyz_list**
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_offset_xyz**, **set_offset_list**, **set_defocus_list**

Kontrollbefehl	**set_pause_list_cond**
Funktion	Definiert die Bedingung am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste, bei der automatisch ein **pause_list** ausgeführt wird.
Aufruf	`set_pause_list_cond( Mask1, Mask0 )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bit ausgewertet. `Mask0` Wie `Mask1`.
Multi-Board-Befehlsname	**n_set_pause_list_cond**
Hinweise	• Wenn gerade eine Liste ausgeführt wird und die folgende Bedingung, siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320, erfüllt ist: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von IOvalue gleich 1 und die in `Mask0` gesetzten Bits von IOvalue gleich 0 sind), wird automatisch ein **pause_list** ausgeführt. Die Bedingung wird einmal pro 10 μs-Takt überprüft. • Die angehaltene Liste kann nur mit **restart_list** wieder fortgesetzt werden. • `Mask1` = `Mask0` = 0 schaltet die Bedingung aus. • Ist die Bedingung ausgeschaltet oder es wird gerade keine Liste ausgeführt, geschieht weiter nichts. • Mit **set_pause_list_cond** kann im Fehlerfall über eine Hardware-Schaltung eine laufende Liste automatisch sofort angehalten werden. So muss nicht auf einen zeitkritischen Befehlsaufruf über das Betriebssystem zurückgegriffen werden. • Ein bedingtes **pause_list** hat Vorrang vor einem gleichzeitig anliegenden /STOP-Signal.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 609, OUT 609 RBF 614.
Verweise	**pause_list**, **restart_list**, **set_pause_list_not_cond**

Kontrollbefehl	**set_pause_list_not_cond**
Funktion	Definiert die “NOT”-Bedingung am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste, bei der automatisch ein **pause_list** ausgeführt wird.
Aufruf	`set_pause_list_not_cond( Mask1, Mask0 )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bit ausgewertet. `Mask0` Wie `Mask1`.
Multi-Board-Befehlsname	**n_set_pause_list_not_cond**
Hinweise	• Wenn gerade eine Liste ausgeführt wird und die folgende Bedingung, siehe auch Kapitel 9.3.2 “Ausführung Bedingter Befehle”, Seite 320 *nicht* erfüllt ist: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von IOvalue gleich 1 und die in `Mask0` gesetzten Bits von IOvalue gleich 0 sind), wird automatisch ein **pause_list** ausgeführt. Die Bedingung wird einmal pro 10 μs-Takt überprüft. • Die angehaltene Liste kann nur mit **restart_list** wieder fortgesetzt werden. • `Mask1` = `Mask0` = 0 schaltet die Bedingung aus. • Ist die Bedingung ausgeschaltet oder es wird gerade keine Liste ausgeführt, geschieht weiter nichts. • Mit **set_pause_list_not_cond** kann im Fehlerfall über eine Hardware-Schaltung eine laufende Liste automatisch sofort angehalten werden. So muss nicht auf einen zeitkritischen Befehlsaufruf über das Betriebssystem zurückgegriffen werden. • Ein bedingtes **pause_list** hat Vorrang vor einem gleichzeitig anliegenden /STOP-Signal.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 610, OUT 610 RBF 615.
Verweise	**pause_list**, **restart_list**, **set_pause_list_cond**

Variabler Listenbefehl	**set_pixel**
Funktion	Definiert im Pixelausgabemodus die Laser-Steuerungsparameter (im "Classic Mode" Pulslänge und Analogspannungswert) eines einzelnen Pixels einer Bildzeile.
Aufruf	`set_pixel( PortOutValue1, PortOutValue2 )`
Parameter	`PortOutValue1` Zur Bedeutung siehe Kapitel 8.7.3 "Laser-Steuerung", Seite 277. Als 32-Bit-Wert ohne Vorzeichen.
	`PortOutValue2` Wie `PortOutValue1`.
Multi-Board-Befehlsname	**n_set_pixel**
Hinweise	• **set_pixel** ist gleichbedeutend mit **set_n_pixel** mit `Number` = 1 (siehe Hinweise dort).
RTC4→RTC6	• Das Einlesen analoger Spannungen (siehe RTC4-Parameter `ADChannel` und RTC4-Befehl **read_pixel_ad**) wird von der RTC6 *nicht* unterstützt. • RTC4-Pixelmode 0 wird nicht mehr unterstützt. • "Classic Mode" (kompatibel zu RTC4-Pixelausgabemodus 1): Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den für `PulseLength` (`PortOutValue1`) angegebenen Wert mit 8 und den Wert für `AnalogOut` (`PortOutValue2`) mit 4. Entsprechend verringern sich die zulässigen Wertebereiche. Die anderen Pixelausgabemodi sind nicht RTC4-kompatibel.
RTC5→RTC6	Unveränderte Funktionalität. Betrifft nur den "Classic Mode", da die anderen Pixelausgabemodi von der RTC5 nicht unterstützt werden.
Versionsinfo	Letzte Änderung DLL 604, OUT 604, RBF 609.
Verweise	**set_n_pixel**, **set_pixel_line**, **set_pixel_line_3d**

Normaler Listenbefehl	**set_pixel_line**
Funktion	Aktiviert einen Pixelausgabemodus und definiert verschiedene Pixelausgabeparameter.
Aufruf	`set_pixel_line( Channel, HalfPeriod, dX, dY )`
Parameter	`Channel` Definiert den Pixelausgabemodus sowie die Ausgabeports, an denen die Pixelinhalte ausgegeben werden sollen. Die Pixelinhalte an sich werden in **set_pixel** oder **set_n_pixel**-Befehlen festgelegt. Diese müssen **set_pixel_line** unmittelbar folgen. Es gilt: `Channel` = Modus + Port. Als 32-Bit-Wert ohne Vorzeichen. Zulässig sind: Modus = 0: Ausgabe von `PortOutValue1` an die Pixel-Dauer (früher Parameter `PulseLength`) und Ausgabe von `PortOutValue2` an den Analog-Ausgang Port. Die Galvanometerscanner-Bewegung ist nicht kontinuierlich, siehe Abschnitt "RTC5→RTC6", Seite 794. "Classic Mode". Modus = 16: Zwei Ausgaben an Port. Je ein 32-Bit-Pixel pro Ausgabe. "Extended Mode". Modus = 32: Zwei Ausgaben an Port. Je zwei 16-Bit-Pixel pro Ausgabe. "Fast Mode" Modus = 64: Zwei Ausgaben an Port. Je 4 8-Bit-Pixel pro Ausgabe. "Ultra Fast Mode". Modus = 256: Wie Modus = 0, jedoch ist die Galvanometerscanner-Bewegung kontinuierlich, siehe Abschnitt "RTC5→RTC6", Seite 794. Modus + 512: Galvanometerscanner-Bewegung mit Sky-Writing (positionsgenaue Pixelzeile). Kann nicht mit Modus = 256 kombiniert werden. Siehe auch Kapitel 8.7.4 "Synchronisation", Seite 280 und Abbildung 67. Port = 1: 12-Bit-Analog-Ausgang 1 (LASER-Buchse). Siehe Hinweis, Seite 794. Port = 2: 12-Bit-Analog-Ausgang 2 (LASER-Buchse. Bei RTC6 PCIe-Karte auch an der MARKING ON THE FLY-Stiftleiste). Siehe Hinweis, Seite 794. Port = 3: 8-Bit-Digital-Ausgang (EXTENSION 2-Stiftleiste). Port = 4: 16-Bit-Digital-Ausgang (EXTENSION 1-Stiftleiste). Port = 5: Ausgabe von `PortOutValue1` und `PortOutValue2` von **set_pixel** und **set_n_pixel** an die Pixel-Dauer (früher Parameter `PulseLength`). Nicht erlaubt mit Modus = 0 und Modus = 256.

Normaler Listenbefehl	**set_pixel_line**	
Parameter (Forts.)	`HalfPeriod`	*Halbe* Pixelausgabeperiode. In Bits. 1 Bit entspricht 1/64 μs. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [Min…(2^{32}–1)] mit: Min = 80 für Modus = 0 und 256. Maximalfrequenz: 0,4 MHz. Min = 40 für Modus = 16. Maximalfrequenz: 0,8 MHz. Min = 20 für Modus = 32. Maximalfrequenz: 1,2 MHz. Für RTC6-Karten *ohne* Option "UFPM" gilt: Min = 40, Maximalfrequenz: 0,8 MHz. Min = 10 für Modus = 64. Maximalfrequenz: 3,2 MHz. Für RTC6-Karten *ohne* Option "UFPM" gilt: Min = 40, Maximalfrequenz: 0,8 MHz.
	`dX`	Abstand in x-Richtung zwischen zwei aufeinanderfolgenden Pixeln. In Bits. Als Wert im 64-Bit-IEEE Gleitkommaformat.
	`dY`	Abstand in y-Richtung zwischen zwei aufeinanderfolgenden Pixeln. In Bits. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_set_pixel_line**	
Hinweise	• Jede Pixelbildzeile muss mit **set_pixel_line** beginnen. **set_pixel_line** sollte ein Sprung-Befehl oder [*]mark[*]-Befehl auf den jeweiligen Startpunkt der Pixelbildzeile vorausgehen. • Unmittelbar nach **set_pixel_line** muss die erforderliche Anzahl von **set_pixel** und **set_n_pixel**-Befehlen folgen. Diese übertragen die Parameter `PortOutValue1` und `PortOutValue2`. Deren Bedeutung hängt von Modus und Port ab (siehe oben, Parameter `Channel`). Siehe auch Kapitel 8.7.3 "Laser-Steuerung", Seite 277. • Der erste Befehl einer Liste, der auf **set_pixel_line** folgt und *kein* **set_pixel** oder **set_n_pixel** ist, beendet den Pixelausgabemodus. Auch **set_pixel_line** beendet den Pixelausgabemodus, bevor er ihn wieder startet. Dabei wird ein Default-Pixel eingefügt. • Das Default-Pixel für Port 1…4 wird durch **set_port_default** festgelegt. Aber Achtung: die Port-Nummern bei **set_port_default** und **set_pixel_line** stimmen nicht überein! Das Default-Pixel für Port 5 (`PulseLength`) wird durch **set_default_pixel** festgelegt. • Bei unzulässigen `Channel`-Werten (Beispiel: 5, für Mode = 0 und Port = 5) wird **set_pixel_line** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). Der Pixelausgabemodus wird dann *nicht* aktiviert. • Beachten Sie, dass unter `HalfPeriod` die *halbe* Periodendauer angegeben werden muss. 2 × `HalfPeriod` ist also der zeitliche Abstand zwischen den einzelnen Pixeln, siehe Kapitel 8.7 "Pixelausgabemodus", Seite 275. `HalfPeriod` darf nicht kleiner als Min sein (wird automatisch geclippt).	

Normaler Listenbefehl	set_pixel_line
Hinweise (Forts.)	• Bei Ausgaben an 12-Bit-Analog-Ausgang 1 (Port = 1) oder 12-Bit-Analog-Ausgang 2 (Port = 2) ist zu beachten: die Digital-Analog-Wandlung wird für Pixelausgabefrequenzen nur bis etwa 100 kHz (d.h. für `HalfPeriod` < ca. 320) in jedem Fall vollständig abgeschlossen. Bei Pixelausgabefrequenzen in diesem Bereich müssen Benutzer die ausreichende Funktionsfähigkeit sorgfältig überprüfen. Bei Pixelausgabefrequenzen > 100 kHz wird empfohlen, die UFPM-Erweiterungskarte, Kapitel 17 "Anhang B: UFPM-Erweiterungskarte", Seite 1070, zu verwenden. • Der Pixelausgabemodus kann mit "Processing on the fly" kombiniert werden. Siehe auch Kapitel 8.6 ""Processing on the fly"", Seite 254. • Der Pixelausgabemodus *sollte nicht* zusammen mit der "Automatischen Laser-Steuerung" (siehe Kapitel 7.4.9 ""Automatische Laser-Steuerung"", Seite 208) verwendet werden, wenn dort der Port-Ausgang, die Pulslänge (`PulseLength`) oder Ausgabeperiode (`HalfPeriod`) der Lasersteuersignale LASER1 und LASER2 angepasst werden. • Der Pixelausgabemodus ist inkompatibel mit dem Softstart-Modus (noch nicht implementiert). • Der Pixelausgabemodus kann *nicht* mit Sky-Writing kombiniert werden. Jedoch kann mit Modus + 512 eine Sky-Writing-Modus 1-ähnliche Bewegung eingestellt werden. Dazu muss Sky-Writing eingeschaltet sein. Im Falle eines SCANahead-Systems muss zusätzlich die automatische Delay-Berechnung eingeschaltet sein. Damit sind positionsgenaue Pixelzeilen möglich. Diese Sky-Writing-Bewegung kann nicht mit Sky-Writing-Modus 2-Bewegungen kombiniert werden. Siehe auch: – Kapitel 8.7.4 "Synchronisation", Seite 280 und Abbildung 67 – Kapitel 7.2.4 "Sky-Writing", Seite 163 • Der Pixelausgabemodus kann *nicht* mit Wobbel kombiniert werden. Siehe auch Kapitel 8.4 "Wobbelmode", Seite 244. • Siehe auch Kapitel 8.7 "Pixelausgabemodus", Seite 275.
RTC4→RTC6	• Zu Unterschieden zum Pixelausgabemodus der RTC4 siehe Abschnitt "Spezielle Funktionen", Seite 58. • Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den für `HalfPeriod` angegebenen Wert mit 8 und die für `dX` und `dY` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Im "Classic Mode" (RTC5-kompatibel) sind Frequenzen bis 400 kHz möglich. Außerdem gibt es weitere Pixelausgabemodi.
Versionsinfo	Letzte Änderung DLL 604, OUT 604, RBF 609. Weitere Pixelausgabemodi: 16, 32, 64 und +512.
Verweise	**set_default_pixel**, **set_default_pixel_list**, **set_pixel**, **set_n_pixel**, **set_pixel_line_3d**, **set_port_default**

Multipler Listenbefehl	**set_pixel_line_3d**
Funktion	Aktiviert den Pixelausgabemodus und definiert verschiedene Pixelausgabeparameter.
Aufruf	`set_pixel_line_3d( Channel, HalfPeriod, dX, dY, dZ )`
Parameter	`Channel` Wie **set_pixel_line**.
	`HalfPeriod` Wie **set_pixel_line**.
	`dX` Wie **set_pixel_line**.
	`dY` Wie **set_pixel_line**.
	`dZ` Abstand in z-Richtung zwischen zwei aufeinanderfolgenden Pixeln. In Bits. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_set_pixel_line_3d**
Hinweise	• Mit **set_pixel_line_3d** kann der Pixelabstand zwischen zwei benachbarten Bildpunkten in einer Zeile über einen *3D*-Vektor definiert werden. • **set_pixel_line_3d** belegt 2 RTC6-Listenspeicherplätze für `dZ` ≠ 0. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem Hauptteil (Normaler Listenbefehl) ausgeführt. Ein noch anliegender Verzögerter Kurzer Listenbefehl wird noch zuvor ausgeführt. • Ansonsten wirkt **set_pixel_line_3d** wie **set_pixel_line**, siehe Hinweise, Seite 793.
RTC4→RTC6	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_pixel_line**. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den für `dZ` angegebenen Wert *nicht* mit 16.
RTC5→RTC6	Grundsätzlich unveränderte Funktionalität. RTC5-Kompatibilitätsmodus: siehe **set_pixel_line**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung: Siehe **set_pixel_line**.
Verweise	**set_pixel_line**, **set_pixel**, **set_n_pixel**

Kontrollbefehl	**set_port_default**
Funktion	Definiert den Default-Ausgabewert für den ausgewählten Ausgang.
Aufruf	`set_port_default( Port, Value )`
Parameter	`Port` Ausgang, für den ein Defaultwert `Value` definiert werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: ANALOG OUT1-Ausgang. Siehe auch Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77. = 1: ANALOG OUT2-Ausgang. Siehe auch Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77. = 2: 8-Bit-Digital-Ausgang. Siehe auch Abschnitt "8-Bit-Digital-Ausgang", Seite 84. = 3: 16-Bit-Digital-Ausgang. Siehe auch Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81. = 4: 2-Bit-Digital-Ausgang. Siehe auch Abschnitt "2-Bit-Digital-Ausgang", Seite 77.
	`Value` Defaultwert. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: Für `Port` = 0/1: 12-Bit-Werte [0…4.095]. Höherwertige Bits werden ignoriert.. Für `Port` = 2: 8-Bit-Werte [0…255]. Höherwertige Bits werden ignoriert. Für `Port` = 3: 16-Bit-Werte [0…65.535]. Höherwertige Bits werden ignoriert. Für `Port` = 4: 2-Bit-Werte [0…3]. Höherwertige Bits werden ignoriert. Für `Value` = "–1" (= 2^{32}–1 = 0xFFFFFFFF) wird die entsprechende Default-Ausgabefunktionalität abgeschaltet (siehe Hinweis unten).
Multi-Board-Befehlsname	**n_set_port_default**
Hinweise	• Bei der Initialisierung der RTC6 (mit **load_program_file**) werden die Defaultwerte für alle Ausgänge auf "–1" (= 2^{32}–1) gesetzt. • Wenn für einen Ausgang ein Defaultwert ≠ "–1" festgelegt wurde, dann wird er auf diesen Defaultwert gesetzt, sobald das Abarbeiten einer Liste mit **stop_execution** oder durch einen externen Stopp beendet wird. Bei einem Defaultwert von "–1" wird der entsprechende Ausgang *nicht* beschrieben (der gegebenenfalls noch hochohmige Zustand bei den Digital-Ausgängen bleibt erhalten). • Defaultwerte (≠ "–1") für die Ausgänge 0…3) werden auch am Ende des Pixelausgabemodus gesetzt, siehe Kapitel 8.7 "Pixelausgabemodus", Seite 275.

Kontrollbefehl	set_port_default
Hinweise (Forts.)	• Nach Initialisierung der Positions-abhängige Laser-Steuerung und/oder Geschwindigkeits-abhängige Laser-Steuerung mit **set_auto_laser_control** wird der entsprechende Defaultwert (für den Ausgang 0, 1, 2 oder 3, je nach gewähltem Lasersteuersignal-Parameter `Ctrl`) auch dann ausgegeben, wenn der Laser am Ende einer Beschriftung ausgeschaltet wird oder wenn die Positions-abhängige Laser-Steuerung und/oder Geschwindigkeits-abhängige Laser-Steuerung mit **set_auto_laser_control** auf einen anderen `Ctrl`-Parameter gesetzt oder mit **set_auto_laser_control** (`Ctrl` = 0) deaktiviert wird, siehe Kapitel 7.4.1 "Lasersteuersignale freigeben, aktivieren und schalten", Seite 192 und Abschnitt "Allgemeine Hinweise", Seite 210. Steht hierbei der Defaultwert auf "–1", so wird der maximal zulässige Wert (4.095, 4.095, 255 oder 65.535) ausgegeben. • Für einen unzulässigen Wert von `Port` wird **set_port_default** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Die Defaultwerte für die Ausgänge 0…2 können auch mit **set_laser_off_default** festgelegt werden.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 `Value` für `Port` = 0/1 mit 4.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_off_default**, **set_default_pixel**

Unverzögerter Kurzer Listenbefehl	set_port_default_list
Funktion	Wie **set_port_default**, aber ein Listenbefehl.
Aufruf	`set_port_default_list( Port, Value )`
Parameter	`Port` Wie **set_port_default**.
	`Value` Wie **set_port_default**.
Multi-Board-Befehlsname	**n_set_port_default_list**
Hinweise	• Siehe **set_port_default**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 614.
Verweise	**set_port_default**

Kontrollbefehl	**set_pulse_picking**
Funktion	Schaltet den Pulse-Picking-Laser-Modus ein.
Aufruf	`set_pulse_picking( No )`
Parameter	`No` Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…63]. = 0: Auf LASER2 wird das LASERON-Signal ausgegeben. = 1…63: Auf LASER2 wird jeder `No`-te LASER1-Puls ausgegeben. `No` > 63: Wird auf 63 geclippt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
Multi-Board-Befehlsname	**n_set_pulse_picking**
Hinweise	• Zum Pulse-Picking-Laser-Modus siehe Kapitel 7.4.8 "Pulse-Picking-Laser-Modus", Seite 206. • Die Pulse-Picking-Signale werden ausgegeben, bis mit **set_laser_mode** ein anderer Laser-Modus eingestellt wird (der Pulse-Picking-Laser-Modus wird durch Einschalten eines jeden anderen Laser-Modus mit **set_laser_mode** ausgeschaltet). • Der Pulse-Picking-Laser-Modus kann *nicht* mit **set_laser_mode** eingeschaltet werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_laser_mode**, **set_pulse_picking_list**

Kontrollbefehl	**set_pulse_picking_length**
Funktion	Definiert eine konstante Pulslänge für das LASER2-Signal im Pulse-Picking-Laser-Modus.
Aufruf	`set_pulse_picking_length( Length )`
Parameter	`Length` Pulslänge. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…65.535]. Höherwertige Bits werden ignoriert. 1 Bit entspricht 1/64 µs. Die Standard-Einstellung nach **load_program_file** ist 0.
Multi-Board-Befehlsname	**n_set_pulse_picking_length**
Hinweise	• Damit der Wert wirksam ist, müssen aktiviert sein: – Der Pulse-Picking-Laser-Modus mit **set_pulse_picking** – Der "konstante Pulslänge"-Modus mit **set_laser_control**(Bit #7 = 1) Der Wert ist dann sofort wirksam, auch während eine Markierung ausgeführt wird. • Ist zwar der Pulse-Picking-Laser-Modus aktiviert, jedoch nicht der "konstante Pulslänge"-Modus (**set_laser_control**(Bit #7 = 0)), übernimmt das Pulse-Picking-Signal die Pulslänge des LASER1-Signals, siehe Kapitel 7.4.8 "Pulse-Picking-Laser-Modus", Seite 206. • Solange weder der Pulse-Picking-Laser-Modus noch der "konstante Pulslänge"-Modus aktiviert sind, ist der Wert `Length` ohne Belang (die Reihenfolge der Aktivierungen **set_pulse_picking**, **set_laser_control** und **set_pulse_picking_length** ist beliebig). • Nach **set_pulse_picking**(0) wird auf LASER2 immer das LASERON-Signal ausgegeben, kein "konstante Pulslänge"-Signal. • **set_pulse_picking_length** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Unverzögerter Kurzer Listenbefehl	**set_pulse_picking_list**
Funktion	Wie **set_pulse_picking**, aber ein Listenbefehl.
Aufruf	`set_pulse_picking_list( No )`
Parameter	`No` Wie **set_pulse_picking**.
Multi-Board-Befehlsname	**n_set_pulse_picking_list**
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_pulse_picking**

Kontrollbefehl	**set_qswitch_delay**
Funktion	Legt in einem YAG-Modus die Länge des Delays des ersten Q-Switch-Pulses bezüglich des FirstPulseKiller-Signals fest, siehe auch Abbildung 55.
Aufruf	`set_qswitch_delay( Delay )`
Parameter	`Delay` Q-Switch-Delay. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 μs. Zulässiger Wertebereich: [0…(2^{26}–1)].
Multi-Board-Befehlsname	**n_set_qswitch_delay**
Hinweise	• Werte über (2^{26}–1) werden geclippt. • Ein YAG-Modus wird mit **set_laser_mode** ([1, 2, 3 oder 5]) eingestellt. • Bei YAG-Modus 1, YAG-Modus 2, YAG-Modus 3 kann die Länge des Q-Switch-Delays mit **set_qswitch_delay** noch nachträglich geändert werden, bei YAG-Modus 2 auch mit **set_firstpulse_killer** und **set_firstpulse_killer_list**.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den für `Delay` angegebenen Wert mit 8. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_qswitch_delay_list**, **set_laser_control**, **set_laser_pulses_ctrl**, **set_laser_pulses**, **set_laser_timing**, **set_firstpulse_killer**, **set_firstpulse_killer_list**

Unverzögerter Kurzer Listenbefehl	**set_qswitch_delay_list**
Funktion	Wie **set_qswitch_delay**, aber ein Listenbefehl.
Aufruf	`set_qswitch_delay_list( Delay )`
Parameter	`Delay` Wie **set_qswitch_delay**.
Multi-Board-Befehlsname	**n_set_qswitch_delay_list**
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_qswitch_delay**

Kontrollbefehl	**set_rot_center**
Funktion	Definiert die Lage des Rotationszentrums für "Processing on the fly"-Rotationskorrekturen (siehe **set_fly_rot** und **set_fly_rot_pos**).
Aufruf	`set_rot_center( X, Y )`
Parameter	`X` Lage des Rotationszentrums bezogen auf den Mittelpunkt (0,0) des Bildfelds. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [-2^{24}…(2^{24}–1)].
	`Y` Wie `X` (analog).
Multi-Board-Befehlsname	**n_set_rot_center**
Hinweise	• Zur "Processing on the fly"-Korrektur siehe Kapitel 8.6.3 "Rotationsbewegungen ausgleichen", Seite 259. • Die Lage des Rotationszentrums sollte mit **set_rot_center** oder **set_rot_center_list** vor der Aktivierung der "Processing on the fly"-Korrektur mit **set_fly_rot** oder **set_fly_rot_pos** definiert werden. • Das Rotationszentrum darf auch außerhalb des Bildfelds liegen. Der zulässige Bereich entspricht dem 32-fachen Bildfeld. • Ein zweiter Kopf kann sinnvoll nur verwendet werden, wenn er exakt auf dasselbe Rotationszentrum justiert ist. • **set_rot_center** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_fly_rot**, **set_fly_rot_pos**

Verzögerter Kurzer Listenbefehl	**set_rot_center_list**
Funktion	Wie **set_rot_center**, aber ein Listenbefehl.
Aufruf	`set_rot_center_list( X, Y )`
Parameter	`X` Wie **set_rot_center**.
	`Y` Wie **set_rot_center**.
Multi-Board-Befehlsname	**n_set_rot_center_list**
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_rot_center**

Kontrollbefehl	set_rtc4_mode
Funktion	Setzt den RTC4-Kompatibilitätsmodus als aktuellen RTC6-DLL-Betriebsmodus.
Aufruf	`set_rtc4_mode()`
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Der RTC4-Kompatibilitätsmodus ist ein optionaler Betriebsmodus der RTC6-DLL. Er wird zur Verfügung gestellt, damit Anwenderprogramme, die für die RTC4 geschrieben sind, auch von der RTC6 (weitgehend) ohne Programmcode-Änderung abgearbeitet werden können. Voraussetzung hierbei ist allerdings, dass der Programmcode nur RTC4-Befehle enthält, die mit unveränderter Funktionalität auch als RTC6-Befehle vorhanden sind. In den Befehlsbeschreibungen, Seite 346 dieses Handbuchs wird jeweils in der Zeile "RTC4→RTC6" auf entsprechende Änderungen hingewiesen. • Standardmäßig ist der RTC6-Standardmodus als RTC6-DLL-Betriebsmodus voreingestellt und kann auch nachträglich wieder mit **set_rtc6_mode** eingestellt werden. • Der aktuelle RTC6-DLL-Betriebsmodus kann mit **get_rtc_mode** ausgelesen werden. • **set_rtc4_mode** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • **set_rtc4_mode** wirkt nicht kartenspezifisch, sondern global für die RTC6-DLL und damit auf alle RTC6-Karten, für die das Anwenderprogramm zugriffsberechtigt ist. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **set_rtc4_mode** weder erzeugt noch verändert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_rtc_mode**, **set_rtc5_mode**, **set_rtc6_mode**

Kontrollbefehl	set_rtc5_mode
Funktion	Setzt den RTC5-Kompatibilitätsmodus als aktuellen RTC6-DLL-Betriebsmodus.
Aufruf	`set_rtc5_mode()`
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Der RTC5-Kompatibilitätsmodus ist ein optionaler Betriebsmodus der RTC6-DLL. Er wird zur Verfügung gestellt, damit Anwenderprogramme, die für die RTC5 geschrieben sind, auch von der RTC6 (weitgehend) ohne Programmcode-Änderung abgearbeitet werden können. Voraussetzung hierbei ist allerdings, dass der Programmcode nur RTC5-Befehle enthält, die mit unveränderter Funktionalität auch als RTC6-Befehle existieren. In den Befehlsbeschreibungen, Seite 346 dieses Handbuchs wird jeweils in der Zeile "RTC5→RTC6" auf entsprechende Änderungen hingewiesen. Siehe auch Kapitel 2.12.2 "RTC5-Quellcode für die RTC6 PCIe-Karte anpassen", Schritt 3, Seite 63. • Im RTC5-Kompatibilitätsmodus müssen z-Koordinatenwerte und Defocus-Werte (z.B. Parameter `Shift` von **set_defocus**) mit einer Parameterauflösung von 16 Bit angegeben werden. Sie werden automatisch mit 16 multipliziert. Entsprechend verringern sich die zulässigen Wertebereiche. • Im RTC5-Kompatibilitätsmodus müssen die Laser-Delay-Werte (Parameter `LaserOnDelay` und `LaserOffDelay` von **set_laser_delays**) mit einer Parameterauflösung von 1/2 μs angegeben werden. Sie werden automatisch mit 32 multipliziert. Entsprechend verringern sich die zulässigen Wertebereiche. • Standardmäßig ist der RTC6-Standardmodus als RTC6-DLL-Betriebsmodus voreingestellt und kann auch nachträglich wieder mit **set_rtc6_mode** eingestellt werden. • Der aktuelle RTC6-DLL-Betriebsmodus kann mit **get_rtc_mode** ausgelesen werden. • **set_rtc5_mode** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • **set_rtc5_mode** wirkt nicht kartenspezifisch, sondern global für die RTC6-DLL und damit auf alle RTC6-Karten, für die das Anwenderprogramm zugriffsberechtigt ist. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **set_rtc5_mode** weder erzeugt noch verändert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_rtc_mode**, **set_rtc4_mode**, **set_rtc6_mode**

Kontrollbefehl	**set_rtc6_mode**
Funktion	Stellt den RTC6-Standardmodus als aktuellen RTC6-DLL-Betriebsmodus ein (Standardeinstellung).
Aufruf	`set_rtc6_mode()`
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Im RTC6-Standardmodus als RTC6-DLL-Betriebsmodus müssen die z-Koordinatenwerte und die Defocus-Werte mit einer Parameterauflösung von 20 Bit angegeben werden. • Im RTC6-Standardmodus als RTC6-DLL-Betriebsmodus müssen die Laser-Delay-Werte mit einer Parameterauflösung von 1/64 µs angegeben werden. • Der aktuelle RTC6-DLL-Betriebsmodus kann mit **get_rtc_mode** abgefragt werden. • **set_rtc6_mode** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • **set_rtc6_mode** wirkt nicht kartenspezifisch, sondern global für die RTC6-DLL und damit auf alle RTC6-Karten, für die das Anwenderprogramm zugriffsberechtigt ist. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **set_rtc6_mode** weder erzeugt noch verändert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_rtc_mode**, **set_rtc4_mode**, **set_rtc5_mode**

Kontrollbefehl	**set_scale**
Funktion	Definiert durch Angabe eines gemeinsamen Skalierungsfaktors für die x- und y-Achse die Skalierungsmatrix M_S für nachfolgende Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 236.
Aufruf	`set_scale( HeadNo, Scale, at_once )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. = 1: Die Definition wirkt nur auf den Anschluss für ersten Scan-Kopf. = 2: Die Definition wirkt nur auf den Anschluss für zweiten Scan-Kopf. = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`Scale` Skalierungsfaktor. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–16…+16]. Bei unzulässigen Werten wird `Scale` auf 1 gesetzt. Negative Werte bewirken zusätzlich eine Spiegelung an beiden Achsen (dies entspricht einer 180°-Drehung).
	`at_once` Legt fest, wann die definierte Transformation wirksam wird. Wie `set_matrix( HeadNo` = 0…3). Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_scale**
Hinweise	• Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 236.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_scale_list**, **set_angle**, **set_offset**, **set_matrix**

Variabler Listenbefehl	**set_scale_list**
Funktion	Wie **set_scale**, aber ein Listenbefehl.
Aufruf	`set_scale_list( HeadNo, Scale, at_once )`
Parameter	`HeadNo` Wie **set_scale**.
	`Scale` Wie **set_scale**.
	`at_once` Legt fest, wann die definierte Transformation wirksam wird. Wie **set_matrix_list**. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_scale_list**
Hinweise	• –
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_scale**

Kontrollbefehl	**set_scanahead_laser_shifts**
Funktion	Nur für SCANahead-Systeme. Verschiebt die LASERON-Signale und LaserOff zeitlich nach vorne oder hinten.
Aufruf	`set_scanahead_laser_shifts( dLasOn, dLasOff )`
Parameter	`dLasOn` Zeitverschiebung des LASERON-Signals in [1/64 µs]. Als 32-Bit-Wert mit Vorzeichen. `dLasOff` Zeitverschiebung des LaserOff in [1/64 µs]. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_set_scanahead_laser_shifts**
Hinweise	• Mit **set_scanahead_laser_shifts** kann eine *Feinabstimmung* der Laser-Delays vorgenommen werden, z.B. um Signallaufzeiten zum Laser auszugleichen oder das Schaltverhalten des Lasersystems zu berücksichtigen. • Wenn **set_scanahead_params** nicht zuvor aufgerufen wurde und die automatische Berechnung nicht aktiviert ist (**activate_scanahead_autodelays** `Mode = 0`), dann werden die Werte nur gespeichert, aber nicht verwendet. Jedoch werden sie angewendet, sobald die Voraussetzungen dazu erfüllt sind. • Die Übertragungsverzögerung von RTC6-Karte zum Scan-Kopf beträgt 20 µs. Ab DLL 605 wird diese automatisch berücksichtigt. Parameterwerte für `dLasOn` und `dLasOff`, die für < DLL 605 optimiert sind, müssen für ≥ DLL 605 um 20 µs verringert werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600. Änderung ab Version DLL 603, OUT 603, RBF 607: Höhere Parameterauflösung von jetzt 1/64 µs statt 1/2 µs. Änderung ab Version DLL 605.
Verweise	**set_scanahead_laser_shifts_list**, **set_scanahead_params**, **activate_scanahead_autodelays**

Unverzögerter Kurzer Listenbefehl	**set_scanahead_laser_shifts_list**
Funktion	Nur für SCANahead-Systeme. Wie **set_scanahead_laser_shifts**, aber ein Listenbefehl.
Aufruf	**set_scanahead_laser_shifts_list**(dLasOn, dLasOff)
Parameter	dLasOn Siehe **set_scanahead_laser_shifts**. dLasOff Siehe **set_scanahead_laser_shifts**.
Multi-Board-Befehlsname	**n_set_scanahead_laser_shifts_list**
Hinweise	• **set_scanahead_laser_shifts_list** wird mit dem nächsten zu berechnenden Delay wirksam.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600.
Verweise	**set_scanahead_laser_shifts**

Kontrollbefehl	**set_scanahead_line_params**
Funktion	Nur für SCANahead-Systeme. Beeinflusst die Qualität von Markier-Ergebnissen zur Laufzeit. Geringere Prozentwerte steigern den Durchsatz auf Kosten der Qualität.
Aufruf	`set_scanahead_line_params( CornerScale, EndScale, AccScale )`
Parameter	`CornerScale` Eckenschärfe bei Abfolgen von Mark/Mark in Prozent. 100% = Scharfe Ecken. Als 32-Bit-Wert ohne Vorzeichen. `EndScale` Markiergenauigkeit bei Wechseln von Markierung/Sprung und Sprung/Markierung. 100% = gerade Linienenden. Als 32-Bit-Wert ohne Vorzeichen. `AccScale` Bestimmt den Anteil der Beschleunigungs*zeit* (nicht: Wegstrecke) mit aktivem Laser in Prozent. 100% = gesamte Beschleunigungszeit. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_scanahead_line_params**
Hinweise	• Nur Werte zwischen 0% und 100% sind sinnvoll. Höhere Werte verbessern nicht mehr die Qualität (exaktere als exakte Ecken gibt es nicht), sondern verlängern nur die Markierungszeit. • Wenn **set_scanahead_params** nicht zuvor aufgerufen wurde, dann werden die Werte nur gespeichert, aber nicht verwendet. • Die Parameter `CornerScale` und `EndScale` beeinflussen nur die Positionierung der Galvanometerscanner. Sie wirken nicht auf die Laser-Schaltzeitpunkte! • Der Parameter `AccScale` beeinflusst nur die Laser-Schaltzeitpunkte. Er beeinflusst *nicht* die Positionierung der Galvanometerscanner! • Wenn Sky-Writing aktiviert ist, wirkt nur der Parameter `EndScale`. • Die anderen Parameter `CornerScale` und `AccScale` werden nicht angewendet, außer es findet *keine* echte Sky-Writing-Bewegung statt: – Wenn Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 aktiviert ist und der Grenzwinkel unterschritten ist, wirkt `CornerScale`. • Der Parameter `EndScale` wirkt, wenn Sky-Writing aktiviert ist (auch hier gilt: geringere Prozentwerte steigern den Durchsatz auf Kosten der Qualität).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600.
Verweise	**set_scanahead_line_params_list**

Unverzögerter Kurzer Listenbefehl	**set_scanahead_line_params_list**
Funktion	Nur für SCANahead-Systeme. Wie **set_scanahead_line_params**, aber ein Listenbefehl.
Aufruf	**set_scanahead_line_params_list**(CornerScale, EndScale, AccScale)
Parameter	CornerScale Wie **set_scanahead_line_params**. EndScale Wie **set_scanahead_line_params**. AccScale Wie **set_scanahead_line_params**.
Multi-Board-Befehlsname	**n_set_scanahead_line_params_list**
Hinweise	• **set_scanahead_line_params_list** wird mit dem nächsten zu berechnenden Delay wirksam.
RTC4→RTC6	Neuer Befehl.
RTC5→ RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600.
Verweise	**set_scanahead_line_params**

Kontrollbefehl	set_scanahead_params
Funktion	Nur für SCANahead-Systeme. Aktiviert auf der RTC6-Karte die SCANahead-Funktionalität zur passenden Ansteuerung eines SCANahead-Systems. Die Laser-Steuerung und die digitalen Ein-/Ausgaben für die Peripherie werden auf ein SCANahead-System vorbereitet, d.h. um die `PreviewTime` verzögert. Um auch die Scanner-Delays und Laser-Delays automatisch berechnen zu lassen, muss **activate_scanahead_autodelays** aufgerufen werden.
Einschränkung	**set_scanahead_params** kann nur ausgeführt werden, wenn gerade keine Liste aktiv ist. Sonst wird der **get_last_error** Returncode `RTC6_BUSY` und Fehlercode 5 zurückgegeben. **set_scanahead_params** kann nur ausgeführt werden, wenn die Option "SCANa" freigeschaltet ist. Sonst wird der **get_last_error** Returncode `RTC6_REJECTED` und Fehlercode 1 zurückgegeben.
Aufruf	`Error = set_scanahead_params( Mode, HeadNo, TableNo, PreviewTime, Vmax, Amax )`
Parameter	`Mode` = 0: Deaktiviert die SCANahead-Funktionalität. Die weiteren Parameter `TableNo, PreviewTime, Vmax, Amax` werden nicht berücksichtigt. Default nach **load_program_file**. Modus z.B. zum Betrieb von intelli*SCAN*-Scan-Köpfen. = 1: Liest die Parameter aus dem mit `HeadNo` angegebenen Kopf aus und verwendet diese. Die Parameter `PreviewTime, Vmax, Amax` werden nicht berücksichtigt. Siehe auch nachfolgende Hinweise. Modus zum Betrieb von angeschlossenen SCANahead-Systemen. = 2: Verwendet die Parameter von **set_scanahead_params**. Modus z.B. zur Softwareentwicklung, für den kein SCANahead-System angeschlossen sein muss. Als 32-Bit-Wert ohne Vorzeichen. `HeadNo` Nummer des Scan-Kopf-Anschlusses. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf. Option "Zweite-Scan-Kopf-Steuerung" erforderlich. Als 32-Bit-Wert ohne Vorzeichen. `TableNo` Zulässiger Wertebereich: 1…8. Eine Korrekturdatei sollte mit **load_correction_file**(`No`) geladen worden sein. Diese wird verwendet, um die Parameter `Amax` und `Vmax` von Ansteuer-Bits (Galvanometerscanner-Drehwinkel) in Programmier-Bits (Bildfeld-Koordinaten) umzurechnen. Siehe **get_scanahead_params**. Als 32-Bit-Wert ohne Vorzeichen.

Kontrollbefehl	set_scanahead_params	
Parameter (Forts.)	PreviewTime	Vorausschauzeit der Regelung. In 10 μs. Im Mode = 0 und Mode = 1 wird PreviewTime ignoriert. Als 32-Bit-Wert ohne Vorzeichen.
	Vmax	Geschwindigkeitsgrenze in Ansteuer-Bits (Galvanometerscanner-Drehwinkel). In Bit/10 μs. Im Mode = 0 und Mode = 1 wird Vmax ignoriert. Als 32-Bit-Wert ohne Vorzeichen.
	Amax	Beschleunigungsgrenze in Ansteuer-Bits (Galvanometerscanner-Drehwinkel). In Bit/(10 μs)^2. Im Mode = 0 und Mode = 1 wird Amax ignoriert. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen.	
	1	Keine Option "SCANa" vorhanden (**get_last_error** Returncode: RTC6_REJECTED).
	3	Kein SCANahead-System angeschlossen oder das SCANahead-System-Tuning ist nicht aktiv. Es wird kein **get_last_error**-Fehler gesetzt.
	5	Eine Liste ist gerade aktiv (**get_last_error** Returncode: RTC6_BUSY).
	6	HeadNo = 0 und HeadNo >2 (**get_last_error** Returncode: RTC6_PARAM_ERROR).
	7	Der ermittelte Skalierungsfaktor überschreitet 16 oder unterschreitet 1/16 (möglicherweise ist keine Korrekturdatei geladen). Der Skalierungsfaktor wird auf 1 gesetzt.
	8	Die RTC6-Karte reagiert nicht. Möglicherweise wurde noch kein Programm geladen (**get_last_error** Returncode: RTC6_TIMEOUT).
	11	Ein PCI-Fehler ist aufgetreten (**get_last_error** Returncode: RTC6_SEND_ERROR).
Multi-Board-Befehlsname	**n_set_scanahead_params**	
Hinweise	• Vmax und Amax werden zur automatischen Berechnung der Scanner-Delays und Laser-Delays verwendet, siehe **activate_scanahead_autodelays**. • Die Werte für Vmax und Amax sind in Ansteuer-Bits (Galvanometerscanner-Drehwinkel) angegeben. Zur internen Umrechnung in Programmier-Bits (Sample-Werte) werden Informationen aus der Korrekturdatei ausgelesen. Diese muss daher als Parameter TableNo angegeben werden. Die Korrekturdatei muss zum Zeitpunkt der Umrechnung schon auf die RTC6-Karte geladen worden sein, braucht jedoch noch nicht zugewiesen sein. Nicht geladene Korrekturdateien können vom System nicht sicher detektiert werden. Mit **get_scanahead_params**(HeadNo = 256) können die in Programmier-Bits umgerechneten Werte für Vmax und Amax abgefragt werden.	

Kontrollbefehl	set_scanahead_params
Hinweise (Forts.)	• Zu den Fehlerrückgaben, siehe **get_scanahead_params**. • `Mode = 1` führt zu einem Abbruch der **set_scanahead_params**-Ausführung (`Error = 3`), wenn kein SCANahead-System angeschlossen oder dieses nicht aktiv ist. Die SCANahead-Funktionalität wird deaktiviert. Die für `Mode = 2` anzugebenden Parameter werden nicht verwendet! • **set_scanahead_params**`( Mode = 1 )` kann nicht ausgeführt werden, wenn eine automatische Laser-Steuerung aktiv ist, die von einem i*DRIVE*-Scan-System (siehe Glossareintrag auf Seite 28) übertragene Daten verwendet, siehe **set_auto_laser_control**. Diese müssen Sie vorübergehend deaktivieren. • **set_scanahead_params** wartet, bis die letzte Session fertig durchgelaufen ist, d. h., bis HEAD BUSY-Listenausführungsstatus (Bit #23) aus **get_status** nicht mehr gesetzt ist. Höchstens wartet **set_scanahead_params** jedoch 255 × [10 µs]. • **set_scanahead_params** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 600, OUT 600. Letzte Änderung mit Version DLL 605: Datentyp von `Amax`.
Verweise	**get_scanahead_params**, **activate_scanahead_autodelays**, **get_last_error**, **get_status**, **load_correction_file**

Kontrollbefehl	**set_scanahead_speed_control**
Funktion	Nur für SCANahead-Systeme. Steuert die Scanner-Delays entweder abhängig von der Soll-Geschwindigkeit oder der tatsächlich erreichbaren Geschwindigkeit.
Aufruf	**set_scanahead_speed_control**(Mode)
Parameter	Mode Modus. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_scanahead_speed_control**
Hinweise	• Mode mit: – Bit #0 steuert Markierbefehle – Bit #1 steuert Sprungbefehle – Bit #n = 0 steuert die Scanner-Delays abhängig von der Sollgeschwindigkeit – Bit #n = 1 steuert die Scanner-Delays abhängig von der tatsächlich erreichbaren Geschwindigkeit • Default nach **load_program_file**: Mode = 2. • **set_scanahead_speed_control** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 610, OUT 610. Letzte Änderung mit Version OUT 611: Anpassung der Laser-Delays.
Verweise	–

Verzögerter Kurzer Listenbefehl	**set_scanner_delays**
Funktion	Definiert die Scanner-Delays.
Aufruf	`set_scanner_delays( Jump, Mark, Polygon )`
Parameter	`Jump` Scanner-Delay-Typ: Jump-Delay. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 µs. Zulässiger Wertebereich: [0…(2^{32}–1)].
	`Mark` Scanner-Delay-Typ: Mark-Delay. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 µs. Zulässiger Wertebereich: [0…(2^{32}–1)].
	`Polygon` Scanner-Delay-Typ: Polygon-Delay. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 µs. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_set_scanner_delays**
Hinweise	• Siehe auch Kapitel 7.2.2 "Scanner-Delays", Seite 150. • Variable Delays für Sprünge und Polygonzüge (siehe Polygonzug) können mit **set_delay_mode** eingestellt werden. • Die angegebenen Delays werden von der RTC6 automatisch angepasst, um Fehler in der Laser-Ansteuerung zu vermeiden, siehe Abschnitt "Automatische Delay-Anpassung", Seite 158. • Die Default-Einstellung nach **load_program_file** entspricht `set_scanner_delays( 9, 6, 3 )`.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_scanner_delays_ctrl**, **set_delay_mode**, **set_laser_delays**

Kontrollbefehl	**set_scanner_delays_ctrl**	
Funktion	Wie **set_scanner_delays**, aber ein Kontrollbefehl.	
Aufruf	`set_scanner_delays_ctrl( Jump, Mark, Polygon )`	
Parameter	`Jump`	Wie **set_scanner_delays**.
	`Mark`	Wie **set_scanner_delays**.
	`Polygon`	Wie **set_scanner_delays**.
Rückgabe	–	
Multi-Board-Befehlsname	**n_set_scanner_delays_ctrl**	
Hinweise	• Siehe **set_scanner_delays**.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 647, OUT 650.	
Verweise	**set_scanner_delays**	

Kontrollbefehl	**set_serial**
Funktion	Setzt die Startseriennummer für den mit **select_serial_set** zuletzt ausgewählten Seriennummernsatz (= 0 nach **load_program_file**) und setzt die Schrittweite für diesen Seriennummernsatz auf 1.
Aufruf	`set_serial( No )`
Parameter	`No` Seriennummer. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_set_serial**
Hinweise	• **set_serial** ist gleichbedeutend mit **set_serial_step**(`Step` = 1), siehe Hinweise, Seite 817.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_serial_step**, **select_serial_set**

Kontrollbefehl	**set_serial_step**
Funktion	Setzt die Startseriennummer und die Schrittweite für den mit **select_serial_set** zuletzt ausgewählten Seriennummernsatz (= 0 nach **load_program_file**).
Aufruf	`set_serial_step( No, Step )`
Parameter	`No` Seriennummer. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{32}–1)].
	`Step` Schrittweite. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…9999]. Es werden immer nur die letzten 4 Dezimalziffern verwendet.
Multi-Board-Befehlsname	**n_set_serial_step**
Hinweise	• **load_program_file** setzt die Startseriennummer auf 0 und die Schrittweite auf 1. • Wenn bei **mark_serial** oder **mark_serial_abs** mit Mode M_2 = 1 die automatische Seriennummernerhöhung deaktiviert wurde, bleibt die eingestellte Schrittweite ohne Auswirkungen. • Für `Step` = 0 wird nie hochgezählt, außer es handelt sich um eine Leermarkierung (siehe **mark_serial** oder **mark_serial_abs**: `digits` = 0): dann erfolgt immer eine Seriennummernerhöhung um 1. • Zur Verwendung von **set_serial_step** siehe Kapitel 7.5.2 "Seriennummern markieren", Seite 222.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_serial**, **mark_serial_abs**, **set_serial**, **select_serial_set**, **select_serial_set_list**

Normaler Listenbefehl	**set_serial_step_list**
Funktion	Wie **set_serial_step**, aber ein Listenbefehl.
Aufruf	`set_serial_step_list( No, Step )`
Parameter	`No` Wie **set_serial_step**.
	`Step` Wie **set_serial_step**.
Multi-Board-Befehlsname	**n_set_serial_step_list**
Hinweise	• Siehe **set_serial_step**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_serial_step**, **select_serial_set_list**, **get_list_serial**, **mark_serial**, **mark_serial_abs**

Kontrollbefehl	**set_short_cmd_mode_ctrl**
Funktion	Legt das Verhalten für Kurze Listenbefehle (siehe Kurzer Listenbefehl, Seite 339) fest.
Aufruf	`set_short_cmd_mode_ctrl( Mode )`
Parameter	`Mode` Zählung Kurzer Listenbefehle. Als 32-Bit-Wert ohne Vorzeichen. = 0 Aus. = 1 Ein. Default nach **load_program_file**. > 1 Reserviert. **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
Multi-Board-Befehlsname	**n_set_short_cmd_mode_ctrl**
Hinweise	• Für `Mode` = 1 begrenzt die RTC6-Karte die Höchstzahl an ausführbaren Kurzen Listenbefehlen / 10 µs-Takt automatisch, um Takt-Überläufe zu vermeiden. Siehe Kurzer Listenbefehl, Seite 339. • Für `Mode` = 0 sind unbegrenzt viele Kurze Listenbefehle (sowie zusätzlich 1 Normaler Listenbefehl, Seite 339) / 10 µs-Takt möglich. Stellen Sie durch Ausprobieren und **get_overrun**-Aufrufen sicher, dass es nicht zu Takt-Überlaufen kommt. Falls notwendig, initiieren Sie manuell einen 10 µs-Takt-Wechsel mit **list_nop** oder **list_continue**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 646, OUT 649.
Verweise	**set_short_cmd_mode_list**

Normaler Listenbefehl	**set_short_cmd_mode_list**
Funktion	Wie **set_short_cmd_mode_ctrl**, aber ein Listenbefehl.
Aufruf	`set_short_cmd_mode_list( Mode )`
Parameter	`Mode` Wie **set_short_cmd_mode_ctrl**.
Multi-Board-Befehlsname	**n_set_short_cmd_mode_list**
Hinweise	• Für `Mode` > 1 wird **set_short_cmd_mode_list** durch ein **list_nop** ersetzt. • Siehe **set_short_cmd_mode_ctrl**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 646, OUT 649.
Verweise	**set_short_cmd_mode_ctrl**

Kontrollbefehl	**set_sky_writing**
Funktion	Aktiviert Sky-Writing-Modus 1 und stellt die zugehörigen Parameter ein oder deaktiviert Sky-Writing.
Aufruf	`set_sky_writing( Timelag, LaserOnShift )`
Parameter	`Timelag` Wie **set_sky_writing_para**.
	`LaserOnShift` Wie **set_sky_writing_para**.
Multi-Board-Befehlsname	**n_set_sky_writing**
Hinweise	• **set_sky_writing** ist identisch zu `set_sky_writing_para( Timelag, LaserOnShift, Nprev, Npost )` mit `Nprev` = ca. (0,15 × `Timelag`) und `Npost` = ca. (0,1 × `Timelag`). • Siehe auch Hinweise bei **set_sky_writing_para** und Kapitel 7.2.4 "Sky-Writing", Seite 163.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `LaserOnShift` mit 32. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_sky_writing_para**, **set_sky_writing_list**

Kontrollbefehl	**set_sky_writing_limit**
Funktion	Legt den Grenzwert für die Sky-Writing-Umschaltung im Sky-Writing-Modus 3 und Sky-Writing-Modus 4 fest.
Aufruf	`set_sky_writing_limit( Limit )`
Parameter	`Limit` Grenzwert. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–1.0…+1.0]. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_set_sky_writing_limit**
Hinweise	• Zur Verwendung von **set_sky_writing_limit** siehe Kapitel 7.2.4 "Sky-Writing", Seite 163. • `Limit` ist der Cosinus des Grenzwinkels (für die Winkelabweichung zwischen aufeinanderfolgenden Vektoren oder Bögen eines Polygonzugs), ab dem eine Sky-Writing-Bewegung ausgeführt werden soll. • Der Initialisierungswert (nach Programmstart) ist `Limit` = 0 (Grenzwinkel = 90°).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_sky_writing_limit_list**

Unverzögerter Kurzer Listenbefehl	**set_sky_writing_limit_list**
Funktion	Wie **set_sky_writing_limit**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_limit_list( Limit )`
Parameter	`Limit` Wie **set_sky_writing_limit**.
Multi-Board-Befehlsname	**n_set_sky_writing_limit_list**
Hinweise	• Siehe **set_sky_writing_limit**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_sky_writing_limit**

Normaler Listenbefehl	**set_sky_writing_list**
Funktion	Wie **set_sky_writing**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_list( Timelag, LaserOnShift )`
Parameter	`Timelag` Wie **set_sky_writing_para**.
	`LaserOnShift` Wie **set_sky_writing_para**.
Multi-Board-Befehlsname	**n_set_sky_writing_list**
Hinweise	• Siehe **set_sky_writing**, **set_sky_writing_para** und **set_sky_writing_para_list**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `LaserOnShift` mit 32. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_sky_writing**, **set_sky_writing_para**, **set_sky_writing_para_list**

Kontrollbefehl	**set_sky_writing_min_speed_ctrl**
Funktion	Stellt die Mindest-Markiergeschwindigkeit für Sky-Writing ein.
Aufruf	`set_sky_writing_min_speed_ctrl( Speed )`
Parameter	`Speed` Mindest-Markiergeschwindigkeit für Sky-Writing. Wie **set_mark_speed_ctrl**.
Multi-Board-Befehlsname	**n_set_sky_writing_min_speed_ctrl**
Hinweise	• **set_sky_writing_min_speed_ctrl** wirkt nur im: – Sky-Writing-Modus 2 – Sky-Writing-Modus 3 – Sky-Writing-Modus 4 • Sky-Writing mit Mindest-Markiergeschwindigkeit, Seite 171 ist nur verfügbar für: – SCANahead-Systeme (**set_scanahead_params**) mit eingeschalteter automatischen Berechnung der Scanner-Delays und Laser-Delays (**activate_scanahead_autodelays**) • Die Sky-Writing-Bewegungen werden so verkürzt, dass zum Mark-Vektor-Beginn/Ende) nicht mehr die Markiergeschwindigkeit erreicht wird, sondern `Speed`. • Gewöhnliche Sky-Writing-Bewegungen werden ausgeführt für: – `Speed` < 100 – `Speed` ≥ Markiergeschwindigkeit (siehe **set_mark_speed_ctrl**) • **set_sky_writing_min_speed_ctrl** wird nicht ausgeführt, wenn: – nicht zuvor die automatische Berechnung der Scanner-Delays und Laser-Delays eingeschaltet wurde (**get_last_error**-Returncode `RTC6_REJECTED`) – der BUSY-Listenausführungsstatus gesetzt ist (**get_last_error**-Returncode `RTC6_BUSY`) • Für "Spot Distance Control" in Kombination mit Sky-Writing gilt: – set_auto_laser_control mit `Mode` = 2-Erweiterung M = +128 kann mit set_sky_writing_min_speed_ctrl kombiniert werden. Der Energieeintrag wird trotzdem konstant gehalten.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 646, OUT 649.
Verweise	**set_sky_writing_min_speed_list**, **set_mark_speed_ctrl**, **set_sky_writing_scale_ctrl**

Unverzögerter Kurzer Listenbefehl	**set_sky_writing_min_speed_list**
Funktion	Wie **set_sky_writing_min_speed_ctrl**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_min_speed_list( Speed )`
Parameter	`Speed` Wie **set_sky_writing_min_speed_ctrl**.
Multi-Board-Befehlsname	**n_set_sky_writing_min_speed_list**
Hinweise	• Siehe **set_sky_writing_min_speed_ctrl**. • `Speed` = 0 – Deaktiviert Sky-Writing mit Mindest-Markiergeschwindigkeit (solange, bis die Markiergeschwindigkeit nachträglich geändert wird) • `Speed` ≥ Markiergeschwindigkeit – Wie `Speed` = 0
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	**set_sky_writing_min_speed_ctrl**

Kontrollbefehl	**set_sky_writing_mode**
Funktion	Schaltet den Sky-Writing-Modus um.
Aufruf	`set_sky_writing_mode( Mode )`
Parameter	`Mode` Sky-Writing-Modus. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{32}–1)]. = 0: Sky-Writing wird deaktiviert. = 1: Sky-Writing-Modus 1 wird aktiviert. = 2: Sky-Writing-Modus 2 wird aktiviert. = 3: Sky-Writing-Modus 3 wird aktiviert. > 3: Sky-Writing-Modus 4 wird aktiviert.
Multi-Board-Befehlsname	**n_set_sky_writing_mode**
Hinweise	• Sky-Writing-Modus 2, Sky-Writing-Modus 3 und Sky-Writing-Modus 4 können nur dann (mit `Mode` > 1) aktiviert werden, wenn: – Zuvor der Sky-Writing-Modus 1 mit `set_sky_writing_para` (`Timelag` ≥ 1/8) aktiviert wurde – UND dieser immer noch aktiv ist • Nach `set_sky_writing_mode( Mode = 0 )` ist eine nachträgliche Sky-Writing-Reaktivierung: – möglich mit `set_sky_writing_mode( Mode > 0 )` – nicht möglich nach `set_sky_writing_para` (`Timelag` < 1/8) • Jeder Modus-Wechsel mit **set_sky_writing_mode** wird erst mit dem nächsten Listenbefehl wirksam. • Wird Sky-Writing-Modus 1 während der Ausführung einer Liste *reaktiviert* (Umschalten von `Mode` = 0 nach 1), so wird ein bereits angefangener Markier-Befehl noch *ohne* Sky-Writing fertig ausgeführt. • Wird Sky-Writing-Modus 1 während der Ausführung einer Liste *deaktiviert* (Umschalten von `Mode` = 1 nach 0), so wird ein im Sky-Writing-Modus 1 angefangener Markier-Befehl noch im Sky-Writing-Modus 1 beendet und anschließend ein Mark-Delay eingefügt. • Ein Aktivieren oder Deaktivieren von Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 (Umschalten nach oder von `Mode` = 2 oder 3 oder 4) ist nicht möglich, wenn der BUSY-Listenausführungsstatus gesetzt ist. **set_sky_writing_mode** wird dann ignoriert (**get_last_error**-Returncode `RTC6_BUSY`). • **set_sky_writing_mode** wird ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Kein `Mode` > 3.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 646, OUT 649. `Mode` > 3.
Verweise	**set_sky_writing_mode_list**, **set_sky_writing_para**

Normaler Listenbefehl	**set_sky_writing_mode_list**
Funktion	Wie **set_sky_writing_mode**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_mode_list( Mode )`
Parameter	`Mode` Wie **set_sky_writing_mode**.
Multi-Board-Befehlsname	**n_set_sky_writing_mode_list**
Hinweise	• Sky-Writing-Modus 2, Sky-Writing-Modus 3 und Sky-Writing-Modus 4 können mit **set_sky_writing_mode_list** auch innerhalb einer Liste aktiviert oder deaktiviert werden (Umschalten nach oder von `Mode` = 2 oder 3 oder 4) Dabei wird ein bereits angefangener Markier-Befehl mit Sky-Writing-Modus 1 beendet und der nächste Markier-Befehl damit angefangen. • Bei der Deaktivierung des Sky-Writing-Modus 1 mit **set_sky_writing_mode_list** (Umschalten von `Mode` = 1 nach 0) wird ein vor der Sky-Writing-Aktivierung definiertes Mark-Delay eingefügt, wenn kein anderes Delay (z.B. ein Jump-Delay) wirksam ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Kein `Mode` > 3.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 646, OUT 649. `Mode` > 3.
Verweise	**set_sky_writing_mode**

Kontrollbefehl	**set_sky_writing_para**
Funktion	Aktiviert Sky-Writing-Modus 1 und stellt die zugehörigen Parameter ein oder schaltet Sky-Writing aus.
Aufruf	`set_sky_writing_para( Timelag, LaserOnShift, Nprev, Npost )`
Parameter	`Timelag` — Sky-Writing-Parameter. *1,0 entspricht 1 µs*. Der `Timelag`-Wert wird mit einer Auflösung von 1/64 µs verwendet. Als Wert im 64-Bit-IEEE Gleitkommaformat. ≥ 1/4: Sky-Writing-Modus 1 wird aktiviert. Zwischen 1/8 und 1/4 wird Sky-Writing aktiviert, jedoch wird intern `Timelag` = 0 verwendet. < 1/8: Sky-Writing wird deaktiviert.
	`LaserOnShift` — Verschiebung (für positive Werte Verzögerung) des Einschaltzeitpunkts der Lasersteuersignale (siehe Abbildung 46). 1 Bit entspricht 1/64 µs. Als 32-Bit-Wert mit Vorzeichen. Negative Werte kleiner als der gesamte Vorlauf werden zur Laufzeit geclippt, d.h. es gilt automatisch: `LaserOnShift` — Modus ≥ ca. (–40) × `Nprev` — 1 ≥ ca. (–20) × `Nprev` — 2 oder 3
	`Nprev` — Definiert die Dauer der Vorlauf-Phase (siehe Abbildung 46). 1 Bit entspricht 10 µs. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: 0 ≤ `Nprev` ≤ (2^{32}–1). Vorlauf-Dauer [µs] — Modus 20 × `Nprev` — 1 10 × `Nprev` — 2 oder 3
	`Npost` — Definiert die Dauer der Nachlauf-Phase (siehe auch Abbildung 46). 1 Bit entspricht 10 µs. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: 0 ≤ `Npost` ≤ (2^{32}–1). Nachlauf-Dauer [µs] — Modus 20 × `Npost` — 1 10 × `Npost` — 2 oder 3
Multi-Board-Befehlsname	**n_set_sky_writing_para**
Hinweise	• Zum Sky-Writing und zur Bedeutung der Parameter siehe Kapitel 7.2.4 "Sky-Writing", Seite 163. • Für `Nprev` ≥ 65.535 wirkt **set_sky_writing_para** wie **set_sky_writing**: `Nprev` wird auf einen Wert von ca. (0,15 × `Timelag`) eingestellt. • Für `Npost` ≥ 65.535 wirkt **set_sky_writing_para** wie **set_sky_writing**: `Npost` wird auf einen Wert von ca. (0,1 × `Timelag`) eingestellt.

Kontrollbefehl	set_sky_writing_para
Hinweise (Forts.)	• **set_sky_writing_para** kann nicht dazu verwendet werden, zwischen Sky-Writing-Modus 1, Sky-Writing-Modus 2, Sky-Writing-Modus 3 und Sky-Writing-Modus 4 hin und her zu schalten (dafür ist **set_sky_writing_mode** vorgesehen). • Wenn **set_sky_writing_para** aufgerufen wird, wenn *keine* Liste ausgeführt wird oder eine Liste mit **set_wait** angehalten wurde, gilt: – Bei `Timelag` ≥ 1/8 wird der Sky-Writing-Modus 1 aktiviert, wenn vorher kein Sky-Writing aktiv war. Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 bleiben bestehen, der nächste Markier-Befehl wird jedoch immer im Sky-Writing-Modus 1 begonnen – Bei `Timelag` < 1/8 wird Sky-Writing deaktiviert • Wenn **set_sky_writing_para** aufgerufen wird, wenn *eine* Liste ausgeführt wird oder eine Liste mit **pause_list** angehalten wurde, gilt: – Wenn Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 aktiv ist, wird **set_sky_writing_para** nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`) – Wenn kein Sky-Writing oder Sky-Writing-Modus 1 aktiv ist, werden die Parameter von **set_sky_writing_para** mit dem nächsten Listenbefehl wirksam. • Im Sky-Writing-Modus 1 wird ein bereits angefangener Markier-Befehl noch mit den bisherigen Parametern ausgeführt und mit Sky-Writing-Modus 1 beendet • Im Sky-Writing-Modus 1 wird der nächste Markier-Befehl im Sky-Writing-Modus 1 begonnen, wenn `Timelag` ≥ 1/8 ist. Anderenfalls wird er beendet und bei Sky-Writing-Modus 1 ein Mark-Delay eingefügt. • `Nprev` = 0 und/oder `Npost` = 0 werden für Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 intern automatisch auf 1 korrigiert (dies wird nicht als Fehler betrachtet). Wird nachträglich (mit **set_sky_writing_mode**) Sky-Writing-Modus 1 aktiviert, so wird dann wieder `Nprev` = 0 und/oder `Npost` = 0 verwendet.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `LaserOnShift` mit 32. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_sky_writing**, **set_sky_writing_para_list**

Normaler Listenbefehl	**set_sky_writing_para_list**
Funktion	Wie **set_sky_writing_para**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_para_list( Timelag, LaserOnShift, Nprev, Npost )`
Parameter	`Timelag` Wie **set_sky_writing_para**.
	`LaserOnShift` Wie **set_sky_writing_para**.
	`Nprev` Wie **set_sky_writing_para**.
	`Npost` Wie **set_sky_writing_para**.
Multi-Board-Befehlsname	**n_set_sky_writing_para_list**
Hinweise	• **set_sky_writing_para_list** kann auch bei aktiviertem Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 innerhalb einer Liste ausgeführt werden. Dabei wird ein bereits angefangener Markier-Befehl mit Sky-Writing-Modus 1 beendet und der nächste Markier-Befehl damit angefangen. • Mit **set_sky_writing_para_list** kann nicht von Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 dauerhaft in Sky-Writing-Modus 1 umgeschaltet werden. Verwenden Sie dazu **set_sky_writing_mode_list**. • Bei der Deaktivierung des Sky-Writing-Modus 1, Sky-Writing-Modus 2, Sky-Writing-Modus 3 oder Sky-Writing-Modus 4 mit **set_sky_writing_para_list** wird ein vor der Sky-Writing-Aktivierung definiertes Mark-Delay eingefügt, wenn kein anderes Delay (z.B. ein Jump-Delay) wirksam ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `LaserOnShift` mit 32. Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_sky_writing_para**

Kontrollbefehl	**set_sky_writing_scale_ctrl**
Funktion	Stellt den Skalierungsfaktor für Sky-Writing ein.
Aufruf	`set_sky_writing_scale_ctrl( Scale )`
Parameter	`Scale` Skalierungsfaktor. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0,01…2,0] oder 0.
Rückgabe	–
Multi-Board-Befehlsname	**n_set_sky_writing_scale_ctrl**
Hinweise	• **set_sky_writing_min_speed_ctrl** stellt die Mindest-Markiergeschwindigkeit für Sky-Writing ein. • `Scale` < 1.0 – Aktiviert Sky-Writing mit Mindest-Markiergeschwindigkeit – Stellt die Mindest-Markiergeschwindigkeit ein entsprechend `Scale` = Mindest-Markiergeschwindigkeit / Markiergeschwindigkeit • Wenn der Skalierungsfaktor über **set_sky_writing_scale_ctrl** / **set_sky_writing_scale_list** eingestellt wurde, führt ein nachträgliches Ändern der Markiergeschwindigkeit nicht zu einer Neuberechnung. • Dagegen wird bei **set_sky_writing_min_speed_ctrl** / **set_sky_writing_min_speed_list** der Skalierungsfaktor bei jeder Markiergeschwindigkeit-Änderung neu berechnet. • `Scale` = 1.0 – Deaktiviert Sky-Writing mit Skalierung der Ausholbewegung – Deaktiviert Sky-Writing mit Mindest-Markiergeschwindigkeit • `Scale` > 1.0 – Aktiviert Sky-Writing mit Skalierung der Ausholbewegung • `Scale` = 0.0 – Wie `Scale` = 1.0 • Siehe Abschnitt "Sky-Writing mit Skalierung der Ausholbewegung", Seite 173.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	**set_sky_writing_scale_list**, **set_sky_writing_min_speed_ctrl**

Unverzögerter Kurzer Listenbefehl	**set_sky_writing_scale_list**
Funktion	Wie **set_sky_writing_scale_ctrl**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_scale_list( Scale )`
Parameter	`Scale` Wie **set_sky_writing_scale_ctrl**.
Multi-Board-Befehlsname	**n_set_sky_writing_scale_list**
Hinweise	• Siehe **set_sky_writing_scale_ctrl**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	**set_sky_writing_scale_ctrl**

Kontrollbefehl	**set_softstart_level**
Funktion	Ohne Funktion.
Aufruf	`set_softstart_level( Index, Level )`
Parameter	`Index` Als 32-Bit-Wert ohne Vorzeichen. `Level` Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_softstart_level**
Hinweise	• **set_softstart_level** hat keine Auswirkung auf das Anwenderprogramm.
RTC4→RTC6	Ohne Funktion.
RTC5→RTC6	Ohne Funktion.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_softstart_level_list**

Normaler Listenbefehl	**set_softstart_level_list**
Funktion	Ohne Funktion.
Aufruf	`set_softstart_level_list( Index, Level1, Level2, Level3 )`
Parameter	`Index` Wie **set_softstart_level**. `Level1` Wie **set_softstart_level**. `Level2` Wie **set_softstart_level**. `Level3` Wie **set_softstart_level**.
Multi-Board-Befehlsname	**n_set_softstart_level_list**
Hinweise	• Wie **set_softstart_level**.
RTC4→RTC6	Ohne Funktion.
RTC5→RTC6	Ohne Funktion.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_softstart_level**, **set_softstart_mode**, **set_softstart_mode_list**

Kontrollbefehl	**set_softstart_mode**
Funktion	Ohne Funktion.
Aufruf	`set_softstart_mode( Mode, Number, Delay )`
Parameter	`Mode` Als 32-Bit-Wert ohne Vorzeichen. `Number` Als 32-Bit-Wert ohne Vorzeichen. `Delay` Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_softstart_mode**
Hinweise	• **set_softstart_mode** hat keine Auswirkung auf das Anwenderprogramm.
RTC4→RTC6	Ohne Funktion.
RTC5→RTC6	Ohne Funktion.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Variabler Listenbefehl	**set_softstart_mode_list**
Funktion	Ohne Funktion.
Aufruf	`set_softstart_mode_list( Mode, Number, Delay )`
Parameter	`Mode` Wie **set_softstart_mode**. `Number` Wie **set_softstart_mode**. `Delay` Wie **set_softstart_mode**.
Multi-Board-Befehlsname	**n_set_softstart_mode_list**
Hinweise	• Siehe **set_softstart_mode**.
RTC4→RTC6	Neuer Befehl. Ohne Funktion.
RTC5→RTC6	Ohne Funktion.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Kontrollbefehl	**set_standby**
Funktion	Definiert die Ausgabeperiode und die Pulslänge der Standby-Pulse für den "Laser standby"-Betrieb oder – im Laser-Mode 4 und Laser-Mode 6 – der freilaufenden Laser-Signale für den "Laser active"-Betrieb und den "Laser standby"-Betrieb.
Aufruf	`set_standby( HalfPeriod, PulseLength )`
Parameter	`HalfPeriod` *Halbe* Standby-Ausgabeperiode. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 µs. Zulässiger Wertebereich: [0…+(2^{32}–1)].
	`PulseLength` Pulslänge der Standbypulse. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 1/64 µs. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_set_standby**
Hinweise	• Nach einem Hardware-Reset (und nach **load_program_file**) muss zur erstmaligen Aktivierung der Ausgänge LASER1, LASER2 und LASERON **set_laser_control** aufgerufen werden, bevor mit **set_standby** oder **set_standby_list** Standby-Pulse aktiviert werden können, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192. Gleichzeitig wird mit **set_laser_control** der Signalpegel der Standby-Pulse eingestellt. • Die Standbypulse sind in *allen* Laser-Modi (CO_2-Modus, YAG-Modus 1, YAG-Modus 2, YAG-Modus 3, Laser-Mode 4, YAG-Modus 5, Laser-Mode 6) verfügbar. • Die Standbypulse können deaktiviert (ausgeschaltet) werden, indem die Standby-Pulslänge und/oder die -Ausgabeperiode auf Null (Standardeinstellung) gesetzt wird. • Bei `HalfPeriod` muss die *halbe* Standby-Ausgabeperiode angegeben werden, siehe Abbildung 54 und Abbildung 56. • Wenn `PulseLength` größer gleich der Ausgabeperiode (2 × `HalfPeriod`) ist, bleibt der Laser dauernd an. • Der Laser-Mode wird mit **set_laser_mode** eingestellt, siehe Kapitel 7.4 "Laser-Steuerung", Seite 192. Die aktuelle Ausgabeperiode und die Pulslänge der Lasersteuersignale für den "Laser active"-Betrieb (außer im Laser-Mode 4 und Laser-Mode 6) werden mit **set_laser_pulses_ctrl**, **set_laser_pulses** oder **set_laser_timing** eingestellt.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `HalfPeriod` und `PulseLength` mit 8. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_standby_list**, **get_standby**

Verzögerter Kurzer Listenbefehl	**set_standby_list**
Funktion	Wie **set_standby**, aber ein Listenbefehl.
Aufruf	`set_standby_list( HalfPeriod, PulseLength )`
Parameter	`HalfPeriod` Wie **set_standby**.
	`PulseLength` Wie **set_standby**.
Multi-Board-Befehlsname	**n_set_standby_list**
Hinweise	• –
RTC4→RTC6	Wie **set_standby**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_standby**

Kontrollbefehl	**set_start_list**
Funktion	Öffnet den RTC6-Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Eingabe-Zeiger an den Listenanfang der angegebenen Liste (“Liste 1” oder “Liste 2”).
Aufruf	`set_start_list( ListNo )`
Parameter	`ListNo` Listennummer der Liste, in die der Eingabe-Zeiger gesetzt werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [ungerade: “Liste 1”, gerade: “Liste 2”].
Multi-Board-Befehlsname	**n_set_start_list**
Hinweise	• **set_start_list ist gleichbedeutend mit set_start_list_pos** mit `Pos` = 0. • Alternativ können **set_start_list_1** und **set_start_list_2** (ohne Parameter) verwendet werden.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**execute_list**, **read_status**, **set_start_list_pos**

Kontrollbefehl	**set_start_list_1**
Funktion	Wie **set_start_list**.
Aufruf	`set_start_list_1()`
Multi-Board-Befehlsname	**n_set_start_list_1**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_start_list**

Kontrollbefehl	**set_start_list_2**
Funktion	Wie **set_start_list**.
Aufruf	`set_start_list_2()`
Multi-Board-Befehlsname	**n_set_start_list_2**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_start_list**

Kontrollbefehl	**set_start_list_pos**	
Funktion	Öffnet den RTC6-Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Eingabe-Zeiger an die angegebene *relative* Position in der angegebenen Liste (“Liste 1” oder “Liste 2”).	
Aufruf	`set_start_list_pos( ListNo, Pos )`	
Parameter	`ListNo`	Listennummer der Liste, in die der Eingabe-Zeiger gesetzt werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [ungerade: “Liste 1”, gerade: “Liste 2”].
	`Pos`	Position des Eingabe-Zeigers (Offset bezüglich des Listenanfangs der jeweiligen Liste). Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_set_start_list_pos**	
Hinweise	• Der nächste Listenbefehl wird an der angegebenen Adresse gespeichert und alle nachfolgenden Listenbefehle an den darauffolgenden Adressen in der gewählten Liste. • Die angegebene Liste wird unbedingt zum Laden geöffnet. Es erfolgt keine Prüfung, ob die Liste momentan abgearbeitet wird, siehe auch Kapitel 6.4.1 “Listen laden”, Seite 108 und **load_list**. • Der Eingabe-Zeiger kann *nicht* in den geschützten RTC6-Listenspeicher-Bereich “Liste 3” gesetzt werden. Soll der geschützte Bereich (in voller Eigenverantwortung des Anwenders) mit **set_start_list_pos** beschrieben werden, kann dieser Bereich vorübergehend mit `config_list( Mem1, -1 )` der “Liste 2” zugeschlagen werden. Nach Beenden des Schreibvorgangs sollte die Konfiguration zum Schutz des Bereichs aber wieder zurückgesetzt werden: `config_list( Mem1, Mem2 )`. • Für ungerade Werte von `ListNo` wird “Liste 1”, anderenfalls “Liste 2” geöffnet. Dadurch kann ein ständiger Listenwechsel durch fortlaufende Zählung automatisch generiert werden. • Wenn “Liste 2” kein Speicher zugewiesen ist (`Mem2` = 0, siehe **config_list**) wird immer “Liste 1” geöffnet. • Wird `Pos` größer als der Speicherbereich der jeweiligen Liste gewählt (`Pos` > `Mem1` oder `Pos` > `Mem2`), so wird `Pos` auf 0 gesetzt. • Die Listenstatuswerte der gewählten Liste (siehe auch **read_status**) werden wie folgt gesetzt: LOAD-Listenstatus gesetzt, READY-Listenstatus zurückgesetzt, USED-Listenstatus zurückgesetzt. Der LOAD-Listenstatus der jeweils anderen Liste wird zurückgesetzt. • **set_start_list_pos** löst einen Flush der gepufferten Listeneingabe aus, siehe Kapitel 6.4.1 “Listen laden”, Seite 108. • **set_start_list_pos** deckt auch die Spezialisierungen **set_start_list_1**, **set_start_list_2**, **set_start_list** oder **set_input_pointer** ab.	

Kontrollbefehl	set_start_list_pos
Hinweise (Forts.)	• *Vorsicht! Wenn das Ende des jeweiligen RTC6-Listenspeicher-Bereichs erreicht ist, wird der Eingabe-Zeiger automatisch an den Anfang desselben RTC6-Listenspeicher-Bereichs zurückgesetzt.* *Vergewissern Sie sich, dass Sie keine Befehle überschreiben, die in Ihrem Anwenderprogramm noch benötigt werden.*
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**execute_list_pos**, **read_status**

Kontrollbefehl	**set_sub_pointer**	
Funktion	Speichert die absolute Startadresse einer Befehlsliste in der internen Verwaltungstabelle für indizierte Unterprogramme.	
Aufruf	`set_sub_pointer( Index, Pos )`	
Parameter	`Index`	Index des indizierten Unterprogramms, für den die Startadresse `Pos` in die Verwaltungstabelle eingetragen werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023]).
	`Pos`	Absolute Startadresse. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_set_sub_pointer**	
Hinweise	• Für `Index` > 1023 und/oder `Pos` > (2^{23}–1) wird **set_sub_pointer** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Mit **set_sub_pointer** kann ein nichtindiziertes Unterprogramm referenziert werden. Damit wird es zu einem indizierten Unterprogramm und kann mit **save_disk**/**load_disk** geschützt und/oder über den Index aufgerufen werden. • Mit **set_sub_pointer** kann aber auch ein indiziertes Unterprogramm oder Zeichen (oder Zeichenfolge) erneut referenziert werden, so dass es auch über einen zweiten Index aufgerufen werden kann. Zur Indexverwaltung sollte aber bevorzugt **copy_dst_src** verwendet werden. • Die Anfangsadressen von Befehlslisten, die mit **set_sub_pointer** referenziert werden sollen, können – *vor* dem Abspeichern der Befehlslisten – mit **get_input_pointer** ausgelesen werden. • **set_sub_pointer** speichert nur *Startadressen* in der internen Verwaltungstabelle. Erst bei einem anschließenden **save_disk**/**load_disk** wird das indizierte Unterprogramm auch zu einer geschützten Funktion. • `Pos` sollte keine beliebige Adresse innerhalb einer Liste, sondern immer die Anfangsadresse eines tatsächlichen Unterprogramms sein, welches mit **list_return** abgeschlossen worden ist und kein **set_end_of_list** enthält.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**load_sub**	

Kontrollbefehl	**set_text_table_pointer**
Funktion	Speichert die absolute Startadresse einer Befehlsliste in der internen Verwaltungstabelle für indizierte Zeichenfolgen.
Aufruf	`set_text_table_pointer( Index, Pos )`
Parameter	`Index` Index der indizierten Zeichenfolge, für die die Startadresse `Pos` in die Verwaltungstabelle eingetragen werden soll. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…41]. Es gilt die gleiche Zuordnung wie bei **load_text_table**: = 0…9: Ziffern für die Beschriftung von Uhrzeit und Datum [0…9]. = 10…21: Monate [Januar…Dezember]. = 22…28: Wochentage [Sonntag…Samstag]. = 29: Leerzeichen für die Beschriftung von Seriennummern. = 30…39: Ziffern für die Beschriftung von Seriennummern [0…9]. = 40: Text für “a.m.”. = 41: Text für “p.m.”.
	`Pos` Absolute Startadresse. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{23}–1)].
Multi-Board-Befehlsname	**n_set_text_table_pointer**
Hinweise	• Indizierte Zeichenfolgen können bei der Beschriftung von Uhrzeit, Datum und Seriennummern eingesetzt werden, siehe Abschnitt “Indizierte Zeichenfolgen aufrufen”, Seite 123. • Für `Index` > 41 und/oder `Pos` > (2^{23}–1) wird **set_text_table_pointer** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Mit **set_text_table_pointer** kann ein nichtindiziertes Unterprogramm referenziert werden. Damit wird es zu einer indizierten Zeichenfolge und kann mit **save_disk**/**load_disk** geschützt und/oder über den Index aufgerufen werden. • Mit **set_text_table_pointer** kann aber auch ein indiziertes Unterprogramm oder Zeichen (oder Zeichenfolge) erneut referenziert werden, so dass es auch über einen zweiten Index aufgerufen werden kann. Zur Indexverwaltung sollte aber bevorzugt **copy_dst_src** verwendet werden. • Die Anfangsadressen von Befehlslisten, die mit **set_text_table_pointer** referenziert werden sollen, können – *vor* dem Abspeichern der Befehlslisten – mit **get_input_pointer** ausgelesen werden. • **set_text_table_pointer** speichert nur *Startadressen* in der internen Verwaltungstabelle. Erst bei einem anschließenden **save_disk**/**load_disk** wird die indizierte Zeichenfolge auch zu einer geschützten Funktion. • `Pos` sollte keine beliebige Adresse innerhalb einer Liste, sondern immer die Anfangsadresse eines tatsächlichen Unterprogramms sein, welches mit **list_return** abgeschlossen worden ist und kein **set_end_of_list** enthält.

Kontrollbefehl	set_text_table_pointer
RTC4→RTC6	Neuer Befehl. **set_text_table_pointer** ist gleichbedeutend mit **set_char_table**, der schon bei der RTC4 *SCAN*alone-Karte (Standalone-Version der RTC4-Karte) zur Verfügung steht.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**load_text_table**, **mark_date**, **mark_serial**, **mark_time**

Kontrollbefehl	**set_timelag_compensation**
Funktion	Gleicht die unterschiedlichen Schleppverzüge von xy-Achsen und z-Achse an.
Aufruf	`set_timelag_compensation( HeadNoXY, TimelagXY, TimelagZ )`
Parameter	`HeadNoXY` Nummer des xy-Scan-Kopfanschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…2]. `TimelagXY` Schleppverzug der xy-Achsen in [10 µs]. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…255]. `TimelagZ` Schleppverzug der z-Achse in [10 µs]. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…255].
Multi-Board-Befehlsname	**n_set_timelag_compensation**
Hinweise	• Bei unzulässigen Parameterangaben wird der **get_last_error**-Returncode `RTC6_PARAM_ERROR` gesetzt und **set_timelag_compensation** wird nicht ausgeführt. • Wenn eine Listenausführung aktiv ist, wird der **get_last_error**-Returncode `RTC6_BUSY` gesetzt (der Kontrollbefehl **get_status** gibt entweder BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus oder PAUSED-Listenausführungsstatus zurück) und **set_timelag_compensation** wird nicht ausgeführt. • Die schnelleren Achsen werden zu den Langsameren hin verzögert. • **set_timelag_compensation** kann auch mit SCANahead-Systemen verwendet werden. Wurde die RTC6-Karte mittels **set_scanahead_params** zum Ansteuern eines SCANahead-Systems konfiguriert, dann wird `TimelagXY` ignoriert. Statt dessen wird der `PreviewTime`-Wert aus **set_scanahead_params** verwendet. **set_timelag_compensation** wartet automatisch darauf, dass ein HEAD BUSY-Listenausführungsstatus (siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten") zurückgesetzt wird, d.h. die Ausführung von **set_timelag_compensation** kann bis `PreviewTime` dauern. • Bei SCANahead-Systemen wird der Parameter `PreviewTime` aus **set_scanahead_params** automatisch als "Schleppverzug" verwendet. • Bezüglich dem Auswerten von Datenaufzeichnungen (**set_trigger**), siehe Hinweis, Seite 851. • **set_timelag_compensation** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_scanahead_params**, **get_last_error**, **get_status**

Verzögerter Kurzer Listenbefehl	**set_trigger**
Funktion	Startet die Messwertaufzeichnung der angegebenen Signale.
Aufruf	`set_trigger( Period, Signal1, Signal2 )`
Parameter	`Period` Messperiode (zeitlicher Abstand zwischen 2 Datenpaar-Aufzeichnungen). 1 Bit entspricht 10 μs. Zulässiger Wertebereich: [0…(2^{31}–1)]. Bit #30 = 0: Automatischer Stopp der Aufzeichnung beim Listenende. Bit #30 = 1: Kein automatischer Stopp der Aufzeichnung beim Listenende. Bit #31 = 0: Automatischer Stopp bei der max. Kanalgröße, siehe **set_trigger4**/**set_trigger8**. Bit #31 = 1: Endlos-Aufzeichnung (Ringpuffer). Als 32-Bit-Wert ohne Vorzeichen.
	`Signal1` Signaltyp für Speicherkanal 1. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: siehe unten. 0 LASERON-Signal 1 = Lasersteuersignal an. 0 = Lasersteuersignal aus. 1 StatusAX Statussignal der x-Achse, Anschluss für ersten Scan-Kopf. 2 StatusAY Statussignal der y-Achse, Anschluss für ersten Scan-Kopf. 3 Reserviert. 4 StatusBX Statussignal der x-Achse, Anschluss für zweiten Scan-Kopf. 5 StatusBY Statussignal der y-Achse, Anschluss für zweiten Scan-Kopf. 6 Reserviert. 7 SampleX Kartesischer Ansteuerwert für die x-Achse (= bis 4 in 7.3.6, Seite 189). 8 SampleY Kartesischer Ansteuerwert für die y-Achse (= bis 4 in 7.3.6, Seite 189). 9 SampleZ Kartesischer Ansteuerwert für die z-Achse (= bis 4 in 7.3.6, Seite 189).

Verzögerter Kurzer Listenbefehl	set_trigger		
Parameter (Forts.)	Signal1 (Forts.)	10	SampleAX_Corr Korrigierter Ansteuerwert für die x-Achse, Anschluss für ersten Scan-Kopf. (= bis #7 in 7.3.6, Seite 189).
		11	SampleAY_Corr Korrigierter Ansteuerwert für die y-Achse, Anschluss für ersten Scan-Kopf (= bis #7 in 7.3.6, Seite 189).
		12	SampleAZ_Corr Korrigierter Ansteuerwert für die z-Achse, wenn xy am Anschluss für ersten Scan-Kopf angeschlossen sind (= bis #10 in 7.3.6, Seite 189). Identisch zum tatsächlichen Ausgabewert für die z-Achse.
		13	SampleBX_Corr Korrigierter Ansteuerwert für die x-Achse, Anschluss für zweiten Scan-Kopf (= bis #7 in 7.3.6, Seite 189).
		14	SampleBY_Corr Korrigierter Ansteuerwert für die y-Achse, Anschluss für zweiten Scan-Kopf (= bis #7 in 7.3.6, Seite 189).
		15	SampleBZ_Corr Korrigierter Ansteuerwert für die z-Achse, wenn xy am Anschluss für zweiten Scan-Kopf angeschlossen sind (= bis #10 in 7.3.6, Seite 189). Identisch zum tatsächlichen Ausgabewert für die z-Achse.
		16	StatusAX+LASERON-Signal StatusAX wenn Lasersteuersignal an. –524.288 wenn Lasersteuersignal aus.
		17	StatusAY+LASERON-Signal StatusAY wenn Lasersteuersignal an. –524.288 wenn Lasersteuersignal aus.
		18	StatusBX+LASERON-Signal StatusBX wenn Lasersteuersignal an. –524.288 wenn Lasersteuersignal aus.
		19	StatusBY+LASERON-Signal StatusBY wenn Lasersteuersignal an. –524.288 wenn Lasersteuersignal aus.

Verzögerter Kurzer Listenbefehl	**set_trigger**		
Parameter (Forts.)	`Signal1` (Forts.)	20	SampleAX_Out Tatsächlicher Ausgabewert für die x-Achse, Anschluss für ersten Scan-Kopf (= bis #11 in 7.3.6, Seite 189). Nicht für z-Achsen-Ausgabewerte verwendbar.
		21	SampleAY_Out Tatsächlicher Ausgabewert für die y-Achse, Anschluss für ersten Scan-Kopf (= bis #11 in 7.3.6, Seite 189). Nicht für z-Achsen-Ausgabewerte verwendbar.
		22	SampleBX_Out Tatsächlicher Ausgabewert für die x-Achse, Anschluss für zweiten Scan-Kopf (= bis #11 in 7.3.6, Seite 189). Nicht für z-Achsen-Ausgabewerte verwendbar.
		23	SampleBY_Out Tatsächlicher Ausgabewert für die y-Achse, Anschluss für zweiten Scan-Kopf (= bis #11 in 7.3.6, Seite 189). Nicht für z-Achsen-Ausgabewerte verwendbar.
		24	Lasersteuerparameter der “Automatischen Laser-Steuerung“ Siehe **set_auto_laser_control**.
		25	SampleAX_Trans Transformierter Ansteuerwert für die x-Achse, Anschluss für ersten Scan-Kopf (= bis #6 in 7.3.6, Seite 189).
		26	SampleAY_Trans Transformierter Ansteuerwert für die y-Achse, Anschluss für ersten Scan-Kopf (= bis #6 in 7.3.6, Seite 189).
		27	SampleAZ_Trans Transformierter Ansteuerwert für die z-Achse, wenn xy am Anschluss für ersten Scan-Kopf angeschlossen sind (= bis #6 in 7.3.6, Seite 189).
		28	SampleBX_Trans Transformierter Ansteuerwert für die x-Achse, Anschluss für zweiten Scan-Kopf (= bis #6 in 7.3.6, Seite 189).
		29	SampleBY_Trans Transformierter Ansteuerwert für die y-Achse, Anschluss für zweiten Scan-Kopf (= bis #6 in 7.3.6, Seite 189).
		30	SampleBZ_Trans Transformierter Ansteuerwert für die z-Achse, wenn xy am Anschluss für zweiten Scan-Kopf angeschlossen sind (= bis #6 in 7.3.6, Seite 189).

Verzögerter Kurzer Listenbefehl	**set_trigger**		
Parameter (Forts.)	`Signal1` (Forts.)	31	Lasersteuerparameter der Vektorkontrollierte Laser-Steuerung Siehe **set_vector_control**.
		32	Fokusverschiebung Siehe **set_vector_control**, **set_defocus**, **set_defocus_list**.
		33	12-Bit-Ausgabewert am ANALOG OUT1-Ausgang Siehe **set_vector_control** und Kapitel 9.1.4 "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2".
		34	12-Bit-Ausgabewert am ANALOG OUT2-Ausgang Siehe **set_vector_control** und Kapitel 9.1.4 "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2".
		35	Ausgabewert am 16-Bit-Digital-Ausgang Siehe **set_vector_control** und Kapitel 9.1.1 "16-Bit-Digital-Ausgang".
		36	Ausgabewert am 8-Bit-Digital-Ausgang Siehe **set_vector_control** und Kapitel 9.1.2 "8-Bit-Digital-Ausgang".
		37	Pulslänge (PulseLength) der Lasersteuersignale LASER1 und LASER2 Siehe **set_vector_control**.
		38	Ausgabeperiode (HalfPeriod) der Lasersteuersignale LASER1 und LASER2 Siehe **set_vector_control**.
		39	FreeVariable0
		40	FreeVariable1
		41	FreeVariable2
		42	FreeVariable3

Verzögerter Kurzer Listenbefehl	set_trigger		
Parameter (Forts.)	Signal1 (Forts.)	43	Zählerstand von Encoderzähler “Encoder0”
		44	Zählerstand von Encoderzähler “Encoder1”
		45	Markiergeschwindigkeit Aus **set_mark_speed**, **set_mark_speed_ctrl**.
		46	16-Bit-Digital-Eingang (EXTENSION 1-Stiftleiste)
		47	Nur für intelli*WELD* II: Zoomwert
		48	FreeVariable4
		49	FreeVariable5
		50	FreeVariable6
		51	FreeVariable7
		52	32-Bit-“Zeitstempel-Zähler” Siehe Kapitel 8.12 “Zeitmessungen”, Seite 293.

Verzögerter Kurzer Listenbefehl	set_trigger		
Parameter (Forts.)	Signal1 (Forts.)	53	Wobbel-Amplitude Siehe **set_wobbel**, **set_wobbel_mode** und Kapitel 8.4 "Wobbelmode", Seite 244.
		54	ReadAnalogIn Siehe **read_analog_in**.
		55	Skalierter Encoderwert für x
		56	Skalierter Encoderwert für y
		57	Skalierter Encoderwert für z
		58	RS-232
		59	`read_mcbsp( 0 )`
		60	`read_multi_mcbsp( 0 )`
		61	`read_multi_mcbsp( 1 )`
		62	`read_multi_mcbsp( 2 )`

Verzögerter Kurzer Listenbefehl	**set_trigger**
Parameter (Forts.)	`Signal1` (Forts.) 63 StatusAX_Synced Für das Rückkanal-Multiplexing dezidierter Signal-Typ. Der angeforderte Signal-Wert von i*DRIVE*-Scan-System 1, x-Achse wird auf den Hinkanal-Block-Start synchronisiert übertragen und kann deshalb später (im Anwenderprogramm) richtig zugeordnet werden, siehe Kapitel 21.1 "Rückkanal-Multiplexing", Seite 1128. Sub-Frame-Bit 0...Bit 19: Daten. Sub-Frame-Bit 20: Synchronisationsinformation. 64 StatusAY_Synced Wie 64. Jedoch i*DRIVE*-Scan-System 1, y-Achse. 65 StatusBX_Synced Wie 64. Jedoch i*DRIVE*-Scan-System 2, x-Achse. 66 StatusBY_Synced Wie 64. Jedoch i*DRIVE*-Scan-System 2, y-Achse. 67 Status_List Bit #31…Bit #10: Reserviert. Bit #9: BUSY-Listenausführungsstatus, siehe **get_status** Bit #8: LASERON-Signal, entsprechend `Signal1`/`Signal2` = 0 Bit #7…Bit #4: Modulo 16 der Anzahl bisheriger Listenstopps. Wertebereich: 0…15. Bit #3…Bit #0: Modulo 16 der Anzahl bisheriger Listenstopps. Wertebereich: 0…15.
	`Signal2` Wie (analog) `Signal1`.
Multi-Board-Befehlsname	**n_set_trigger**

Verzögerter Kurzer Listenbefehl	**set_trigger**
Hinweise	*Allgemeine Hinweise* • Sind `Signal1` und `Signal2` nicht aus obiger Liste, wird **set_trigger** (außer für `Period` = 0) durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Nach einem **set_trigger** mit `Period` > 0 wird sofort und dann immer im Abstand der angegebenen Messperiode ein Datenpaar (Daten-Dupel) aufgezeichnet. – `Period`(Bit #31 = 0): • Die Messwertaufzeichnung endet automatisch bei 2^{24} Datenpaaren (oder bei 2^{25} Daten-Einträgen, wenn `Signal2` ungültig ist, siehe auch **set_trigger4**/**set_trigger8**). • Sie endet früher, wenn eine der folgenden Situationen eintritt: - es wird **set_trigger** (`Period` = 0) aufgerufen - es wird **set_trigger4**(`Period` = 0) aufgerufen - es wird **set_trigger8**(`Period` = 0) aufgerufen – `Period`(Bit #31 = 1): • Startet eine Endlos-Datenaufzeichnung gemäß `Period` = "Messperiode | 0x80000000" mit 0 < Messperiode < 2^{31}–1. Die Messwertaufzeichnung endet nicht automatisch. Stattdessen beginnt sie jedes Mal von vorne und überschreibt bisher aufgezeichnete Daten ("Ringpuffer"). Die Daten können zwischenzeitlich mit **get_waveform_offset** aus beliebiger Stelle und in beliebigen Paketen ausgelesen werden. **measurement_status** gibt auch hier Auskunft über den aktuellen Stand der Messwertaufzeichnung. • Eine mit **set_trigger** gestartete Messwertaufzeichnung erfolgt nur während der Ausführung einer Liste. • Wenn die Messwertaufzeichnung nicht mit **set_trigger** (`Period` = 0) oder **set_trigger4**(`Period` = 0) beendet wurde, wird sie beim Start einer neuen Liste automatisch fortgesetzt (der Status der Messwertaufzeichnung wird durch **set_end_of_list** nicht zurückgesetzt). Während der Ausführung einer Liste wird der Status durch **set_trigger**(`Period` = 0), **set_trigger4**(`Period` = 0) oder **stop_execution** zurückgesetzt. Mit **stop_trigger** kann der Status zurückgesetzt werden wenn keine Liste ausgeführt wird. • Sample-Werte und vom Scan-System übertragene Statuswerte (`Signal1`, `Signal2` = 1…23, 25…30), die durch **set_trigger** auf der RTC6 gespeichert werden, sind immer im RTC6-20 Bit-Bereich, auch wenn die RTC6-DLL auf den RTC4-Kompatibilitätsmodus eingestellt ist. Die gespeicherten Messwerte können anschließend mit **get_waveform** zur Auswertung an den PC übertragen werden. Es wird im Allgemeinen empfohlen, die Messung vor dem Auslesen der Daten explizit zu beenden. Die Messwerte werden mit **get_waveform** als 32-Bit-Werte an den PC übertragen (siehe Hinweise bei **get_value**). • Der aktuelle Status der Messwertaufzeichnung kann mit **measurement_status** abgerufen werden.

Verzögerter Kurzer Listenbefehl	**set_trigger**
Hinweise (Forts.)	*Allgemeine Hinweise (Forts.)* • Eine gerade laufende Messwertaufzeichnung wird abgeschaltet mit (egal welcher der Folgenden diese gestartet hat): – **set_trigger**(`Period` = 0) – **set_trigger4**(`Period` = 0) – **set_trigger8**(`Period` = 0) • Wird eine Messung mit **set_trigger** (`Period` = 0) oder **set_trigger4**(`Period` = 0) abgebrochen, gehen die zuvor aufgezeichneten Messwerte *nicht* verloren und der Messwertzähler bleibt auf seinem letzten Wert stehen. Dadurch kann auch noch nachträglich mit **measurement_status** die Anzahl der Dateneinträge abgefragt werden (Zeitmessung!). Wird eine Messung dagegen mit **set_trigger** (`Period` > 0) oder **set_trigger4**(`Period` > 0) neu gestartet, so erfolgt ein Reset des Messwertzählers und die bisherigen Daten werden durch die neuen Messwerte überschrieben. Die Fortsetzung einer explizit oder automatisch angehaltenen Messung ist nicht möglich. *Hinweise zu den Daten* • Welche Statussignale vom Scan-System generiert und auf den Statuskanälen zurückgegeben werden hängt vom Typ des Scan-Systems ab. Entnehmen Sie die entsprechenden Informationen dem Handbuch Ihres Scan-Systems. Bei iDRIVE-Scan-Systemen kann mit **control_command** eingestellt werden, welche Informationen auf den Statuskanälen zurückgegeben werden. • Bei Scan-Systemen mit nur einem Statuskanal liefern nur die x-Messsignale sinnvolle Daten. • Bei 3D-Scan-Systemen werden die Ausgabewerte und die Statuswerte der z-Achse jeweils über den Kanal desjenigen Scan-Kopf-Anschlusses übertragen, an dem die z-Achse angeschlossen ist und wie es über **select_cor_table** eingestellt wurde. Beispiel: Ist die z-Achse am x-Kanal des Anschluss für zweiten Scan-Kopf angeschlossen (**select_cor_table**(`HeadA`, 0)), kann ihr Statussignal über StatusBX (`Signal1`/`Signal2` = 4) ausgelesen werden. • Die Signale 12 und 15 sowie 27 und 30 sind jeweils identisch: SampleAZ_Corr = SampleBZ_Corr und SampleAZ_Trans = SampleBZ_Trans. • Statussignale `Status<…> sind` einige 10 μs-Takte später als die Ansteuersignale `Sample<…>`. Siehe auch Abschnitt "Daten auslesen", Seite 225. • Für `Signal1`, `Signal2` = 0, 24, 31…42 und 47…51 werden die Ausgabewerte aus dem jeweils vorherigen Takt aufgezeichnet. Die Datenaufzeichnung unterschiedlicher Signale erfolgt wegen der internen Verarbeitungskette nicht synchron. Je nach dem, wie die Signale kombiniert sind, müssen Sie die Daten nachträglich um einen festen zeitlichen Offset korrigieren.

Verzögerter Kurzer Listenbefehl	**set_trigger**
Hinweise (Forts.)	*Hinweise zu den Daten (Forts.)* • Beachten Sie beim Auswerten von Daten, die bei aktivem Schleppverzug-Ausgleich (**set_timelag_compensation**) aufgezeichnet wurden: – Ein Aufruf von `set_timelag_compensation( TimelagXY < TimelagZ )` verzögert die xy-Achse sowie die Lasersteuersignale um [`TimelagZ` – `TimelagXY`] – Für `Signal1`, `Signal2` = 7, 8, 10, 11, 13, 14, 20, 21 wird jeweils der unverzögerte x-Achsen-/y-Achsen-Ansteuerwert aufgezeichnet – Für `Signal1`, `Signal2` = 20, 21 – wird anders als aufgezeichnet – der tatsächliche Ansteuerwert um [`TimelagZ` – `TimelagXY`] verzögert am Scan-Kopf-Anschluss ausgegeben – Für `Signal1`, `Signal2` = 0 wird das um [`TimelagZ` – `TimelagXY`] verzögerte LASERON-Signal aufgezeichnet • Für `Signal1`, `Signal2` = 24 wird der Signalparameter aufgezeichnet, der durch die "Automatische Laser-Steuerung" (jedoch nicht bei der Vektorkontrollierte Laser-Steuerung) ausgegeben wird (siehe **set_auto_laser_control**). Eine Aufzeichnung erfolgt nur, wenn die "Automatische Laser-Steuerung" aktiviert und während der Laser an ist. Bei ausgeschaltetem Laser wird 0 aufgezeichnet. Bei "Spot Distance Control" hat Signal 24 keine direkte Bedeutung, insbesondere bedeutet es *nicht* "HalfPeriod". • Für `Signal1`, `Signal2` = 31 wird der für die Vektorkontrollierte Laser-Steuerung ausgewählte Signalparameter aufgezeichnet (siehe **set_vector_control**). Eine Aufzeichnung erfolgt auch außerhalb der Ausführung von [*]para[*]-Befehlen, jedoch bleibt dort der Signalwert unverändert. • Pulslänge und Ausgabeperiode (`Signal1`, `Signal2` = 37, 38) werden nur für Signale für den "Laser active"-Betrieb (nicht für Standby-Signale) aufgezeichnet. Im Softstart-Modus (noch nicht implementiert) und Pixelausgabemodus ändern sich Pulslänge oder Ausgabeperiode u.U. schneller als im 10 µs-Takt (Zeitauflösung der Aufzeichnung mit **set_trigger**). Diese Werte können nicht aufgezeichnet werden. • Zu `Signal1`, `Signal2` = 39…42 und 48…51 siehe Kapitel 6.9.1 "Freie Variablen", Seite 138.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: • Die Messung kann mit `Period` = 0 abgebrochen werden. • Die Signale `StatusAZ` und `StatusBZ` (`Signal1`/`Signal2` = 3, 6) sind nicht mehr verfügbar. • Alle Koordinatenwerte sind auf (0|0|0) bezogen (Wertebereich [-2^{19}…(2^{19}–1)]) und nicht mehr Nullpunkt-verschoben.
RTC5→RTC6	Grundsätzlich unveränderte Funktionalität. • Erhöhter Speicherbereich für Datenaufzeichnungen durch Zusammenfassen von Kanälen, siehe **set_trigger4**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Änderung: DLL 610, OUT 610, RBF 615: Endlos-Aufzeichnung. Letzte Änderung DLL 648, OUT 651: `Signal1`, `Signal2` = 63, 64, 65, 66, 67. `Period` Bit #30.
Verweise	**get_waveform**, **get_waveform_offset**, **measurement_status**, **control_command**, **get_value**, **get_values**, **set_mcbsp_out**, **set_mcbsp_out_ptr**, **set_trigger4**, **set_trigger8**

Verzögerter Kurzer Listenbefehl	**set_trigger4**
Funktion	Startet die Messwertaufzeichnung der angegebenen Messsignale (wirkt wie **set_trigger**, jedoch mit bis zu 4 Messsignalen gleichzeitig).
Aufruf	`set_trigger4( Period, Signal1, Signal2, Signal3, Signal4 )`
Parameter	`Period` Wie **set_trigger**.
	`Signal1` Wie **set_trigger**.
	`Signal2` Wie **set_trigger**.
	`Signal3` Wie **set_trigger**.
	`Signal4` Wie **set_trigger**.
Multi-Board-Befehlsname	**n_set_trigger4**
Hinweise	• Siehe Hinweise zu **set_trigger**. • Bei **set_trigger4** stehen pro Messkanal 2^{23} Einträge zur Verfügung, wenn alle 4 Signale im zulässigen Bereich liegen (siehe **set_trigger**). Die Messwertaufzeichnung endet automatisch bei 2^{23} Daten-Quadrupel). • Wenn Signal 3 und Signal 4 unzulässig sind (siehe **set_trigger**), ist **set_trigger4** gleichbedeutend mit **set_trigger**. Die Speicherbereiche von Signal 3 und Signal 4 werden den Speicherbereichen von Signal 1 und Signal 2 zugefügt, so dass 2^{24} Datenpaare aufgezeichnet werden können. Die Messwertaufzeichnung endet automatisch bei 2^{24} Daten*paaren*. • Ist nur Signal 1 im zulässigen Bereich, werden die Speicherbereiche aller 4 Kanäle zu einem Speicherbereich zusammengefasst. Damit können bis zu 2^{25} Datenwerte aufgezeichnet werden. Die Messwertaufzeichnung endet automatisch bei 2^{25} Daten*werten*. • Eine gerade laufende Messwertaufzeichnung wird abgeschaltet mit (egal welcher der 3 Befehle diese gestartet hat): – **set_trigger**(`Period` = 0) – **set_trigger4**(`Period` = 0) – **set_trigger8**(`Period` = 0) • Wird mit **set_trigger**(`Period` > 0) oder **set_trigger4**(`Period` > 0) eine Messwertaufzeichnung neu gestartet, dann erfolgt ein Reset des Messwertzählers. Die bisherigen Daten werden durch die neuen Messwerte überschrieben. Siehe auch die entsprechenden Hinweise unter **set_trigger**. • Wie bei **set_trigger**/**set_trigger4** kann **get_waveform**/**get_waveform_offset** zum Übertragen der aufgezeichneten Werte an den PC verwendet werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Grundsätzlich unveränderte Funktionalität. Erhöhter Speicherbereich für Datenaufzeichnungen durch Zusammenfassen von nicht benutzten Kanälen, Endlos-Aufzeichnung.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 610, OUT 610, RBF 615: siehe **set_trigger**.
Verweise	**set_trigger**, **set_trigger8**, **stop_trigger**, **get_waveform**, **get_waveform_offset**

Verzögerter Kurzer Listenbefehl	**set_trigger8**
Funktion	Startet die Messwertaufzeichnung der angegebenen Messsignale (wirkt wie **set_trigger**, jedoch mit bis zu 8 Messsignalen gleichzeitig).
Aufruf	`set_trigger8( Period, Signal1, Signal2, Signal3, Signal4, Signal5, Signal6, Signal7, Signal8 )`
Parameter	`Period` Wie **set_trigger**.
	`Signal1` Wie **set_trigger**.
	`Signal2` Wie **set_trigger**.
	`Signal3` Wie **set_trigger**.
	`Signal4` Wie **set_trigger**.
	`Signal5` Wie **set_trigger**.
	`Signal6` Wie **set_trigger**.
	`Signal7` Wie **set_trigger**.
	`Signal8` Wie **set_trigger**.
Multi-Board-Befehlsname	**n_set_trigger8**
Hinweise	• Siehe Hinweise zu **set_trigger**. • Bei **set_trigger8** stehen pro Messkanal 2^{22} Einträge zur Verfügung. Die Messwertaufzeichnung endet automatisch bei 2^{22} Daten-Octupel. • Eine gerade laufende Messwertaufzeichnung wird abgeschaltet mit (egal welcher der 3 Befehle diese gestartet hat): – **set_trigger**(`Period` = 0) – **set_trigger4**(`Period` = 0) – **set_trigger8**(`Period` = 0) • Anders als **set_trigger** und **set_trigger4**, vergrößert **set_trigger8** nicht automatisch den Speicherbereich für Datenaufzeichnung durch Zusammenfassen von nicht benutzten Kanälen. Es werden immer höchstens 2^{22} Einträge pro Messkanal aufgezeichnet. • Der Messwertzähler wird zurückgesetzt, wenn eine Messwertaufzeichnung erneut gestartet wird mit: – **set_trigger**(`Period` > 0) – **set_trigger4**(`Period` > 0) – **set_trigger8**(`Period` > 0) Die bisherigen Daten werden mit neuen Daten überschrieben. Siehe auch die entsprechenden Hinweise unter **set_trigger**. • Wie bei **set_trigger**/**set_trigger4** kann **get_waveform**/**get_waveform_offset** zum Übertragen der aufgezeichneten Werte an den PC verwendet werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 635, OUT 636.
Verweise	**set_trigger**, **set_trigger4**, **stop_trigger**, **get_waveform**, **get_waveform_offset**

Unverzögerter Kurzer Listenbefehl	**set_vector_control**	
Funktion	Initialisiert oder deaktiviert die Vektorkontrollierte Laser-Steuerung.	
Aufruf	`set_vector_control( Ctrl, Value )`	
Parameter	`Ctrl`	Steuerparameter zur Initialisierung oder Deaktivierung der Vektorkontrollierte Laser-Steuerung (für `Ctrl` = 1…6 identisch zu **set_auto_laser_control**). Als 32-Bit-Wert ohne Vorzeichen. = 1…8: Definiert, welcher Signalparameter durch die Vektorkontrollierte Laser-Steuerung variiert wird. = 1: 12-Bit-Ausgabewert am ANALOG OUT1-Ausgang. = 2: 12-Bit-Ausgabewert am ANALOG OUT2-Ausgang. = 3: Ausgabewert am 8-Bit-Digital-Ausgang. = 4: Pulslänge (`PulseLength`) der Lasersteuersignale LASER1 und LASER2. = 5: Ausgabeperiode (`HalfPeriod`) der Lasersteuersignale LASER1 und LASER2. = 6: Ausgabewert am 16-Bit-Digital-Ausgang. = 7: Fokusverschiebung (“Defocus”). = 8: Reserviert für intelli*WELD*. = 0 oder > 8: Deaktiviert die Vektorkontrollierte Laser-Steuerung.
	`Value`	Legt den Anfangswert für den über `Ctrl` ausgewählten Parameter fest. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte (überlaufende Werte werden auf die Randwerte geclippt): Für `Ctrl` = 1/2: 12-Bit-Werte [0…4.095]. Für `Ctrl` = 3: 8-Bit-Werte [0…255]. Für `Ctrl` = 4: [0…(2^{32}–1)], 1 Bit entspricht 1/64 μs. Für `Ctrl` = 5: [0…(2^{32}–1)], 1 Bit entspricht 1/64 μs. Für `Ctrl` = 6: 16-Bit-Werte [0…65.535]. Für `Ctrl` = 7: [–524.288…+524.287], `Value` = Fokusverschiebung + 524.288.
Multi-Board-Befehlsname	**n_set_vector_control**	
Hinweise	• Bei ungültigem Wert für `Ctrl` wird die Vektorkontrollierte Laser-Steuerung deaktiviert (`Ctrl` = 0, Initialisierungszustand). Ansonsten wird `Value` als Startwert für den nächsten [*]para[*]-Befehl verwendet. Ungültige Werte für `Value` werden auf die jeweils zulässigen Maximalwerte geclippt. • **set_vector_control** wirkt sich nur auf [*]para[*]-Befehle aus. • Wenn mit **set_auto_laser_control** eine “Automatische Laser-Steuerung” mit dem gleichen Kontrollparameter aktiviert ist (`Ctrl` = 1…6), dann setzt **set_vector_control** den 100%-Wert der “Automatischen Laser-Steuerung” auf den Wert `Value`. Das gilt jedoch nicht für `Ctrl` = 5: **set_auto_laser_control**(`Ctrl` = 7) (“Spot Distance Control”) und die Vektorkontrollierte Laser-Steuerung mit **set_vector_control** (`Ctrl` = 5) (`HalfPeriod`) können *nicht* kombiniert werden.	

Unverzögerter Kurzer Listenbefehl	**set_vector_control**
Hinweise (Forts.)	• Für `Ctrl` = 7 wird bei [*]para[*]-Befehlen die Fokusverschiebung linear (wie mit **set_defocus** oder **set_defocus_list**, siehe Hinweise dort) bis zum angegebenen Endwert variiert (damit die z-Ausgaben ausgegeben werden, muss die Option "3D" freigeschaltet, sowie eine 3D-Korrekturdatei geladen und zugewiesen sein). Stimmt beim ersten [*]para[*]-Befehl die aktuell eingestellte Fokusverschiebung nicht mit dem unter **set_vector_control** angegebenen Anfangswert `Value` überein, wird zu Beginn des Befehls ein "Harter Sprung" in der z-Ausgabe auf `Value` ausgeführt. Die mit **set_defocus** oder **set_defocus_list** festgelegte Einstellung geht verloren. Im Gegensatz zu **set_defocus** oder **set_defocus_list** muss bei **set_vector_control** für die Fokusverschiebung (`Value`) statt eines vorzeichenbehafteten Wertes im Bereich [–524.288…+524.287] ein um 524.288 nach oben verschobener, vorzeichenloser Wert angegeben werden: `Value` = Fokusverschiebung + 524.288. • Die "Automatische Laser-Steuerung" sollte nicht zusammen mit der variablen Laserleistung aus **set_multi_mcbsp_in** oder der "Frei definierbaren Wobbelfigur" verwendet werden.
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 bei `Ctrl` = 7 den für `Value` angegebenen Wert mit 16 (`Value` = Fokusverschiebung + 32.768). Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität. Im RTC5-Kompatibilitätsmodus multipliziert die RTC6 bei `Ctrl` = 7 den für `Value` angegebenen Wert mit 16 (`Value` = Fokusverschiebung + 32.768). Entsprechend verringert sich der zulässige Wertebereich.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 634: `Ctrl` = 8.
Verweise	**set_auto_laser_control**

Kontrollbefehl	**set_verify**
Funktion	Aktiviert oder deaktiviert eine Download-Überprüfung. Siehe Kapitel 6.8.1 "Download-Überprüfung", Seite 134.
Aufruf	`OldVerify = set_verify( Verify )`
Parameter	`Verify` Einstellparameter. Als 32-Bit-Wert ohne Vorzeichen. = 0: Die Überprüfung wird deaktiviert. > 0: Die Überprüfung wird aktiviert.
Rückgabe	Den Einstellparameter `Verify`, der vor Aufruf von **set_verify** aktiv war. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_set_verify**
Hinweise	• Die Download-Zeiten verlängern sich mit aktivierter Überprüfung. • Die Überprüfung von Korrekturdatei-Downloads funktioniert nur bei Korrekturdateien, in die eine Check-Summe eingetragen ist (anderenfalls wird der **get_last_error**-Returncode `RTC6_VERIFY_ERROR` erzeugt). Um sicherzugehen, dass eine Check-Summe eingetragen ist (bei älteren `ct5`-Korrekturdateien ist dies nicht der Fall), sollte die zu ladende Korrekturdatei vor dem Laden mit dem Kontrollbefehl **verify_checksum** überprüft werden. Wenn die Korrekturdatei keine Check-Summe enthält, dann trägt **verify_checksum** sie ein. • **set_verify** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. **set_verify** ändert nur die Einstellungen in der RTC6-DLL für die angegebene (oder Default-) Karte und hat keine Auswirkungen auf die Karte selbst. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **set_verify** weder erzeugt noch verändert. • **set_verify** kann beim Einschalten der Download-Überprüfung einen **get_last_error**-Returncode `RTC6_OUT_OF_MEMORY` erzeugen, wenn eine RTC6-DLL-interne Windows-Speicheranforderung fehlschlägt. In diesem Fall bleibt die Download-Überprüfung abgeschaltet.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_error**, **get_last_error**, **verify_checksum**

Normaler Listenbefehl	**set_wait**
Funktion	Fügt einen nummerierten Unterbrechungspunkt (wait marker) in die Liste ein.
Aufruf	`set_wait( WaitWord )`
Parameter	`WaitWord` Nummer des Unterbrechungspunkts. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…(2^{32}–1)]. 0 wird auf 1 korrigiert.
Multi-Board-Befehlsname	**n_set_wait**
Hinweise	• Die Listenabarbeitung wird an jedem Unterbrechungspunkt unterbrochen und so lange angehalten, bis mit **release_wait** fortgefahren wird. Dadurch können Synchronisierungen realisiert werden. • Die Signale für den "Laser active"-Betrieb werden durch **set_wait** ausgeschaltet und es wird ein gegebenenfalls mit **home_position** oder **home_position_xyz** definierter Home-Jump ausgeführt (für die Dauer des Home-Jumps ist der INTERNAL-BUSY-Listenausführungsstatus gesetzt). Ein Fortfahren mit **execute_list_pos**, **restart_list** oder mit einem externen Start ist in der Folge nicht möglich. Ein **stop_execution** oder Externer Stopp ist dagegen möglich. **release_wait**, **stop_execution** oder ein Externer Stopp heben die Unterdrückung des Starts wieder auf. • Durch **set_wait** wird der (mit **get_status** abrufbare) PAUSED-Listenausführungsstatus gesetzt und der BUSY-Listenausführungsstatus zurückgesetzt. Das umgekehrte erfolgt bei einem anschließenden **release_wait**, siehe auch Kapitel 6.4.3 "Listenausführungsstatus", Seite 111. • Wurde eine Ausführung an einem Unterbrechungspunkt unterbrochen, dann gibt **get_wait_status** die Nummer dieses Unterbrechungspunkts zurück.
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Zusätzlicher PAUSED-Listenausführungsstatus. Dieser wird durch **set_wait** gesetzt.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_wait_status**, **release_wait**, **pause_list**, **stop_list**

Verzögerter Kurzer Listenbefehl	**set_wobbel**
Funktion	Legt die Parameter für eine ellipsenförmige Wobbelbewegung fest. “Wobbel“ ist eine Bewegung der Ausgabeposition, die der normalen Markierbewegung überlagert wird. Siehe Kapitel 8.4 “Wobbelmode“, Seite 244.
Aufruf	`set_wobbel( Transversal, Longitudinal, Freq )`
Parameter	`Transversal` Amplitude der ellipsenförmigen Bewegung *senkrecht* zur momentanen oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung. In Bits. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…131.071 ($=2^{17}$-1)]. Zu große Werte werden geclippt. `Longitudinal` Amplitude der ellipsenförmigen Bewegung *parallel* zur momentanen Bewegungsrichtung oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung. In Bits. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…131.071 ($=2^{17}$-1)]. Zu große Werte werden geclippt. `Freq` Frequenz der Wobbelbewegung in *Hz* (Anzahl der Ellipsen pro Sekunde). Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–6000…6000]. Zu große Werte werden geclippt.
Multi-Board-Befehlsname	**n_set_wobbel**
Hinweise	• Der Wobbelmode kann zum Markieren von Linien unterschiedlicher Linienbreite verwendet werden. Der linearen Markierbewegung wird dazu eine ellipsenförmige Bewegung überlagert, was zu einer spiralförmigen Bewegung des Laserfokus im Bildfeld führt. Mit **set_wobbel_mode** kann alternativ auch eine achtförmige Wobbelbewegung (mit einer in Bewegungsrichtung stehenden oder liegenden Acht) aktiviert werden. Die Linienbreite kann mit geeigneten Werten für die Amplituden und die Frequenz eingestellt werden (Frequenz und Markiergeschwindigkeit sollten aufeinander abgestimmt werden, siehe Kapitel 8.4.1 “Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte“, Seite 246). Auch bei Kreisbögen folgt die Wobbelbewegung immer (Ausnahme siehe unten) der momentanen Markierungsrichtung. Die effektive Linienbreite ist also unabhängig vom kartesischen Winkel immer gleich groß. • Der Wobbelmode wird ausgeschaltet, wenn beide Amplituden und/oder die Frequenz auf Null gesetzt werden (genauer für $-n \le$ `Freq` $\le n$ mit $n = 50000/65536 = 0{,}7629\ldots$). • Die Frequenz ist vorzeichenbehaftet. Bei positiven Werten dreht sich der Wobbelvektor im Uhrzeigersinn, bei negativen Werten entgegen dem Uhrzeigersinn. Dadurch kann die innere und äußere Punktdichte bei Bögen unabhängig von deren eigener Drehrichtung individuell eingestellt werden. • Am Anfang einer Markierung (nach **set_wobbel** oder einem Jump) wird der Wobbel-Startpunkt in der Regel auf denselben Wert relativ zum Vektor/Bogen-Startpunkt gesetzt, bei einem Polygonzug (inkl. Bögen) dagegen kontinuierlich fortgesetzt.

Verzögerter Kurzer Listenbefehl	**set_wobbel**
Hinweise (Forts.)	• Bei identischen Amplituden (`Longitudinal` = `Transversal`) ist der Wobbelstartpunkt fest bezogen auf das Koordinatensystem, also unabhängig von der momentanen Bewegungsrichtung. Mit **set_wobbel_direction** kann eine davon abweichende feste Bezugsrichtung eingestellt werden. • Bei einem Winkel kann es bei elliptischer Wobbelbewegung je nach momentaner Wobbelphase zu mehr oder weniger kleinen Sprüngen bei der Wobbelfortsetzung nach dem Knick kommen. Dies geschieht nicht (= "runde Ecken"), wenn die Wobbelbewegung exakt kreisförmig ist (`Longitudinal` = `Transversal`). • Für `Longitudinal` = 0 ergibt sich eine sinusförmige Wobbelbewegung quer zur Bewegungsrichtung. • Achten Sie bei der Definition der Wobbelfigur und der Frequenz auf die Dynamik des Scan-Kopfs und des Lasers. Anderenfalls ist eine Überhitzung und sogar dauerhafte Beschädigung des Systems möglich, siehe Kapitel 8.4.1 "Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte". • Die momentane Wobbelamplitude kann mit **set_trigger[*]** (Signal 53) aufgezeichnet werden. Das Format der Daten ist ((Transversal << 16) + Longitudinal).
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Weitere Wobbelfiguren: elliptisch, richtungsabhängig einstellbar, siehe auch **set_wobbel_mode**. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `Longitudinal` und `Transversal` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_mark_speed**, **set_wobbel_mode**, **set_wobbel_direction**

Unverzögerter Kurzer Listenbefehl	**set_wobbel_control**
Funktion	Definiert die Laser-Steuerungsparameter für die Laserleistungsvariation bei “Frei definierbaren Wobbelfiguren”.
Aufruf	`set_wobbel_control( Ctrl, Value, MinValue, MaxValue )`
Parameter	`Ctrl` Steuerparameter zur Initialisierung oder Deaktivierung der Laserleistungsvariation. Als 32-Bit-Wert ohne Vorzeichen. = 0: Deaktiviert die Laserleistungsvariation. = 1…6: Definiert, welcher Signalparameter variiert wird. Zur Bedeutung siehe **set_auto_laser_control**. = 7: Deaktiviert die Laserleistungsvariation (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). = 8: Aktiviert eine Laserleistungsvariation, bei der ANALOG OUT1 und ANALOG OUT2 abwechseln. > 8: Wie 7. `Value` Nominelle Laserleistung P0 (100%). Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: siehe **set_auto_laser_control**, maximal jedoch [0…65.535] oder 0xFFFFFFFF (bei `Ctrl` = 1…6) oder 0xFFFF (bei `Ctrl` = 8). Zu große Werte werden geclippt. `MinValue` Grenzwert, der nicht überschritten werden kann, siehe **set_auto_laser_control**. Als 32-Bit-Wert ohne Vorzeichen. Der zulässige Wertebereich hängt vom ausgewählten `Ctrl`-Parameter und von `Value` ab. `MaxValue` Siehe `MinValue`.
Multi-Board-Befehlsname	**n_set_wobbel_control**
Hinweise	• Ein noch anliegender Verzögerter Kurzer Listenbefehl wird noch zuvor ausgeführt. • Für `Ctrl` = 0, `Ctrl` = 7 und `Ctrl` > 8 wird die Laserleistungsvariation deaktiviert (Initialisierungszustand nach **load_program_file**). Die Werte aus der Multi-McBSP-Übertragung werden dann auch nicht verwendet. • `Ctrl` = 8 aktiviert eine Laserleistungsvariation, bei der ANALOG OUT1 und ANALOG OUT2 abwechseln. Die “Frei definierbare Wobbelfigur” beginnt mit ANALOG OUT1. Danach wird bei jedem Mikroschritt gewechselt. Siehe auch **set_wobbel_vector_2**. • Bei `Ctrl` = 8 sind `Value`, `MinValue` und `MaxValue` als 2 unabhängige 16-Bit-Werte zu interpretieren: – Die oberen 16 Bit gelten für ANALOG OUT2 – Die unteren 16 Bit gelten für ANALOG OUT1 • **set_wobbel_control**(`Ctrl` = 8) und **set_wobbel_mode**(`Mode` = 3), siehe Seite 870, können nicht kombiniert werden.

Unverzögerter Kurzer Listenbefehl	**set_wobbel_control**
Hinweise (Fort.)	• Für `Value`, `MinValue`, `MaxValue` gelten dieselben Bedingungen wie bei der "Automatischen Laser-Steuerung" (siehe **set_auto_laser_control**), maximal jedoch [0…65.535]. Initialisierungswerte sind `MinValue` = 0 und `MaxValue` = 0xFFFFFFFF, also keinerlei Einschränkung. • Die Laserleistung kann mit der Vektorkontrollierte Laser-Steuerung kombiniert werden, wenn die entsprechenden `Ctrl`-Parameter gleich gewählt sind. • Wenn eine entlang der Wobbelfigur variierende Laserleistung gewünscht ist, sollte **set_wobbel_control** noch vor der Aktivierung der "Frei definierbaren Wobbelfigur" mit **set_wobbel_mode** ausgeführt werden. Anderenfalls wird keine Laserleistung moduliert, auch dann nicht, wenn in den Datensätzen eine Änderung angegeben ist. • Sonderfall: Für – `Value` = 0xFFFFFFFF (bei `Ctrl` = 1…6) – `Value` = 0xFFFF (bei `Ctrl` = 8) wird die nominelle Laserleistung nicht aus **set_wobbel_control** zugewiesen, sondern aus demjenigen Port entnommen, der dem Signalparameter `Ctrl` zugeordnet ist. Das ist dann der zu diesem Zeitpunkt aktuell vorhandene Inhalt, der über andere Standard-Befehle gesetzt worden ist, z.B. **write_da_1_list** für `Ctrl` = 1. Diese werden noch vor **set_wobbel_control** ausgeführt. Aber Achtung: Mit Ausführung der "Frei definierbaren Wobbelfigur" kann sich der Wert am Port jederzeit ändern. Spätere Aufrufe von **set_wobbel_control** liefern dann andere Werte für die nominelle Laserleistung.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 631, OUT 632: Bei `Ctrl` = 8 wird nun bei jedem Mikroschritt gewechselt (nicht mehr bei jedem **set_wobbel_vector**-Aufruf).
Verweise	**set_wobbel_vector**, **set_multi_mcbsp_in**, **set_multi_mcbsp_in_list**, **set_wobbel_vector_2**

Unverzögerter Kurzer Listenbefehl	**set_wobbel_direction**
Funktion	Definiert einen Richtungsvektor für Wobbelbewegungen.
Aufruf	`set_wobbel_direction( dX, dY )`
Parameter	`dX` x-Komponente des Richtungsvektors. In Bits. Als 32-Bit-Wert mit Vorzeichen.
	`dY` y-Komponente des Richtungsvektors. In Bits. Als 32-Bit-Wert mit Vorzeichen.
Multi-Board-Befehlsname	**n_set_wobbel_direction**
Hinweise	• Für einen Richtungsvektor ungleich 0 (`dX` und/oder `dY` ungleich 0) bezieht sich die Wobbelbewegung (Longitudinal und Transversal) auf *diesen* Richtungsvektor statt auf den momentanen Richtungsvektor. Die Länge des Richtungsvektors ist unerheblich. Die RTC6 normiert den Richtungsvektor. • Für `dX = dY =` 0 ist die Funktion deaktiviert. • Die Richtungsvektor-Einstellung bleibt auch nach dem Ausschalten des Wobbelmodes mit **set_wobbel** oder **set_wobbel_mode** bestehen und wird beim Wiedereinschalten des Wobbelmodes weiter verwendet.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_wobbel**, **set_wobbel_mode**, **set_wobbel_offset**

Verzögerter Kurzer Listenbefehl	**set_wobbel_mode**
Funktion	Schaltet den Wobbelmode ein oder aus und legt die Parameter für eine ellipsenförmige oder achtförmige Wobbelbewegung fest, siehe Kapitel 8.4 "Wobbelmode", Seite 244.
Aufruf	`set_wobbel_mode( Transversal, Longitudinal, Freq, Mode )`
Parameter	`Transversal` Amplitude der Wobbelbewegung Bewegung *senkrecht* zur momentanen oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung. In Bits. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…131.071 (=2^{17}-1)]. Zu große Werte werden geclippt.
	`Longitudinal` Amplitude der Wobbelbewegung *parallel* zur momentanen Bewegungsrichtung oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung. In Bits. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…131.071 (=2^{17}-1)]. Zu große Werte werden geclippt.
	`Freq` Frequenz der Wobbelbewegung in *Hz* (Anzahl der Ellipsen oder Achter pro Sekunde). Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–6000…6000]. Zu große Werte werden geclippt.
	`Mode` Definiert die Wobbelform. Als 32-Bit-Wert mit Vorzeichen. = 0: Ellipsenförmige Wobbelbewegung. < 0: Achtförmige Wobbelbewegung senkrecht zur Bewegungsrichtung (stehende Acht). = 1: Achtförmige Wobbelbewegung parallel zur Bewegungsrichtung (liegende Acht). > 1: "Frei definierbare Wobbelfigur", siehe **set_wobbel_vector**.
Multi-Board-Befehlsname	**n_set_wobbel_mode**
Hinweise	• Für `Mode` = 0 funktioniert **set_wobbel_mode** identisch zu **set_wobbel** (siehe Hinweise dort). • Für `Mode` ≠ 0 wird eine achtförmige Wobbelbewegung aktiviert. Mit einer Achterbewegung kann je nach eingestellten Parametern eine stärkere Bearbeitung in der Linienmitte realisiert werden. Wenn transversale und longitudinale Amplituden identisch gewählt werden, bleibt die Wobbelfigur fest im Raum stehen, ansonsten folgt sie der momentanen Bewegungsrichtung. Wenn bei der Ausführung von **set_wobbel_mode** in einer Liste eine "Frei definierbare Wobbelfigur" definiert ist (siehe **set_wobbel_vector**, und Kapitel 8.4 "Wobbelmode", Seite 244), wählt `Mode` > 1 diese aus, ansonsten die liegende Acht wie bei `Mode` = 1. Wird *nachträglich* eine Wobbelfigur definiert und der Wobbelmode war mit Parameter `Mode` > 1 aktiviert worden, wird automatisch von der liegenden Acht auf die Wobbelfigur umgeschaltet. Aber Achtung: Eine Leistungsvariation entlang der Wobbelfigur ist dann nicht möglich, siehe **set_wobbel_control**.

Verzögerter Kurzer Listenbefehl	**set_wobbel_mode**
Hinweise (Forts.)	• Der Wobbelmode wird ausgeschaltet, wenn beide Amplituden und/oder die Frequenz auf Null gesetzt werden (genauer für –n ≤ `Freq` ≤ n mit n = 50000/65536 = 0,7629…). • Die Frequenz ist vorzeichenbehaftet. Bei positiven Werten dreht sich der Wobbelvektor (in der ersten Achter-Schleife) im Uhrzeigersinn, bei negativen Werten entgegen dem Uhrzeigersinn. Dadurch kann (insbesondere für eine ellipsenförmige Wobbelbewegung) die innere und äußere Punktdichte bei Bögen unabhängig von deren eigener Drehrichtung individuell eingestellt werden. • Am Anfang einer Markierung (nach **set_wobbel** oder einem Jump) wird der Wobbel-Startpunkt in der Regel auf denselben Wert relativ zum Vektor/Bogen-Startpunkt gesetzt, bei einem Polygonzug (inkl. Bögen) dagegen kontinuierlich fortgesetzt. Bei identischen Amplituden (`Longitudinal` = `Transversal`) ist der Wobbelstartpunkt fest bezogen auf das Koordinatensystem, also unabhängig von der momentanen Bewegungsrichtung. • Bei einem Winkel kann es bei einer ellipsenförmigen oder achtförmigen Wobbelbewegung je nach momentaner Wobbelphase zu mehr oder weniger kleinen Sprüngen bei der Wobbelfortsetzung nach dem Knick kommen. Dies erfolgt nicht (= “runde Ecken“), wenn die Wobbelbewegung exakt kreisförmig ist (`Longitudinal` = `Transversal`). • Für `Longitudinal` = 0 ergibt sich eine sinusförmige Wobbelbewegung quer zur Bewegungsrichtung. • Mit der “Frei definierbaren Wobbelfigur” ist der Parameter `Freq` ohne Bedeutung. Er muss jedoch innerhalb der gültigen Grenzen liegen, da sonst der Wobbelmode als solches deaktiviert wird. Das gilt auch für die Parameter `Transversal` und `Longitudinal`. • Wird der Wobbelmode deaktiviert, bleibt eine bereits definierte Wobbelfigur aktiv und wird beim nächsten Einschalten des Wobbelmodes mit `Mode` > 1 wieder verwendet. Das gilt auch dann, wenn die Wobbelfigur zuvor mit **create_dat_file** gespeichert und dann durch **load_program_file** geladen wurde. • Achten Sie bei der Definition der Wobbelfigur und der Frequenz auf die Dynamik des Scan-Kopfs und des Lasers. Anderenfalls ist eine Überhitzung und sogar dauerhafte Beschädigung des Systems möglich, siehe Kapitel 8.4.1 “Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte”.

Verzögerter Kurzer Listenbefehl	**set_wobbel_mode**
Hinweise (Forts.)	• Die momentane Wobbelamplitude kann mit **set_trigger[*]** (Signal 53) aufgezeichnet werden. Das Format der Daten ist ((Transversal << 16) + Longitudinal). • Der Wobbelmode kann nicht kombiniert werden mit: – Sky-Writing – Pixelausgabemodus – Jumps – **laser_on_list**
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `Longitudinal` und `Transversal` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_wobbel**, **set_wobbel_control**, **set_wobbel_offset**, **set_wobbel_vector**

Verzögerter Kurzer Listenbefehl	**set_wobbel_mode_phase**
Funktion	Schaltet einen klassischen Wobbelmode wie mit **set_wobbel_mode** ein oder aus. Legt zusätzlich eine Startphase fest.
Aufruf	`set_wobbel_mode_phase( Transversal, Longitudinal, Freq, Mode, Phase )`
Parameter	`Transversal` Wie bei **set_wobbel_mode**. `Longitudinal` Wie bei **set_wobbel_mode**. `Freq` Wie bei **set_wobbel_mode**. `Mode` Wie bei **set_wobbel_mode**. `Phase` Startphase. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0,0°…360,0°]. Negative Werte werden auf 0,0° geclippt. Zu große positive Werte um entsprechende Vielfache von 360,0° reduziert.
Multi-Board-Befehlsname	**n_set_wobbel_mode_phase**
Hinweise	• `Mode` ≥ 2 (“Frei definierbare Wobbelfigur”) ist bei **set_wobbel_mode_phase** nicht erlaubt (wird auf 1 geclippt). • Die Startphase wird mit einer Auflösung von 16 Bit für den Vollkreis in eine Ganzzahl umgewandelt. • Mit **set_wobbel_mode_phase** wird die Wobbelfigur auch weitergeführt bei: – **jump**-Befehlen – **timed_jump**-Befehlen – **para_jump**-Befehlen – **list_nop** – **long_delay** Die Wobbelfigur folgt dabei der zuletzt gültigen markierten Bewegungsrichtung. • **set_wobbel_direction** und **set_wobbel_offset** sind wirksam.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 614.
Verweise	**set_wobbel_mode**

Verzögerter Kurzer Listenbefehl	**set_wobbel_offset**
Funktion	Definiert eine Verschiebung der Wobbelfigur entlang oder quer zur Bewegungsrichtung.
Aufruf	`set_wobbel_offset( OffsetTrans, OffsetLong )`
Parameter	`OffsetTrans` Transversaler Offset. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: ±32.767. Zu große Werte werden geclippt. Initialisierungswerte nach **load_program_file**: (0,0).
	`OffsetLong` Longitudinaler Offset. Sonst wie `OffsetTrans`.
Multi-Board-Befehlsname	**n_set_wobbel_offset**
Hinweise	• Die Offsets sind sowohl für die "klassischen" Wobbelfiguren (Kreis, Ellipse, Acht) als auch für die "Frei definierbare Wobbelfiguren", siehe Kapitel 8.4 "Wobbelmode", Seite 244, anwendbar. • Die gesamte Wobbelamplitude darf inklusive dieser Offsets den Wertebereich von $\pm(2^{17}-1)$ nicht überschreiten. • Die Offsets werden am Anfang der Wobbel-Markierung als "Harter Sprung" eingestellt, ebenso beim Ausschalten oder Nicht-Benutzen des Wobbelmodes, z.B. bei einem Sprung-Befehl (analog zu longitudinalen Amplituden bei den "klassischen" Wobbelfiguren).
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die Werte für `OffsetTrans` und `OffsetLong` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_wobbel_mode**, **set_wobbel_vector**, **set_wobbel_direction**

Unverzögerter Kurzer Listenbefehl	**set_wobbel_vector**
Funktion	Legt einen linearen Abschnitt einer Wobbelfigur fest.
Aufruf	`set_wobbel_vector( dTrans, dLong, Period, dPower )`
Parameter	`dTrans` Mikroschritt eines linearen Wobbelfigurabschnitts. In Bits. `dLong` Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–256.0…+255.0]. `dTrans` ist die Wobbelauslenkung quer zur Bewegungsrichtung (das ist die Laser-Trajektorie oder die mittels **set_wobbel_direction** definierte Bewegungsrichtung). `dLong` ist entsprechend längs dazu. `Period` Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…65.535]. = 1…65.535: Anzahl der Mikroschritte. = 0: Die Wobbelfigur wird abgeschaltet. `dPower` Mikroschritt der relativen Laserleistung. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–1.0…+1.0]. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_set_wobbel_vector**
Hinweise	• `Period` = 0 ist als Figurenabschnitt unzulässig. `Period` = 0 bedeutet ein explizites Abschalten der “Frei definierbaren Wobbelfigur” (nicht des Wobbelmodes an sich, siehe dazu **set_wobbel_mode**). Nachfolgende Aufrufe von **set_wobbel_vector** beginnen eine neue Wobbelfigur. • Jeder Aufruf von **set_wobbel_vector** fügt unabhängig vom Zeitpunkt des Aufrufs einen neuen Abschnitt am Ende der bisherigen Abschnitte hinzu. • Es können maximal 1023 Wobbelfigurenabschnitte definiert werden. Nach 1023 Abschnitten beginnt die Speicherung automatisch wieder beim ersten Abschnitt und überschreibt diesen und mit jedem weiteren Aufruf auch die folgenden Abschnitte. • Die Wobbelfigur beginnt automatisch mit einem Wobbelvektor (0,0), also direkt auf der eigentlichen Markierung (ein “Harter Sprung” wie bei den klassischen Wobbelfiguren, wenn die Longitudinalamplitude ≠ 0 ist, wird dadurch vermieden) und endet ebenso dort, ohne dass dies explizit angegeben werden muss. • Jeder Wobbelfigurenabschnitt besteht aus einem Vektor. Er wird in Form von Mikroschritten und deren Anzahl angegeben. Jeder Mikroschritt wird zur Laufzeit innerhalb von 10 μs abgearbeitet. Die Parameter `dTrans`, `dLong` werden intern auf 7 Bit Nachkommastellen gerundet. Daher sind bei großen Werten für `Period` Rundungsfehler zu berücksichtigen. Mit positivem Vorzeichen beginnt `dTrans` (wie bei einer kreisförmigen Wobbelfigur) in Bewegungsrichtung nach rechts, `dLong` in Bewegungsrichtung nach vorne. Der Endpunkt des letzten Abschnitts ist gleichbedeutend mit dem Anfangspunkt des ersten Abschnitts. Er endet unabhängig von der jeweiligen Position immer bei (0,0) und wird mit einem harten Sprung (siehe “Harter Sprung”) ausgeführt. Die Laserleistung wird dabei wieder auf die anfängliche nominelle Laserleistung gesetzt.

Unverzögerter Kurzer Listenbefehl	**set_wobbel_vector**
Hinweise (Forts.)	• Die aufsummierte Wobbelamplitude darf an keiner Stelle den Bereich $\pm(2^{17}\text{-}1)$ überschreiten. • Wenn mit **set_wobbel_control** und `set_wobbel_mode( Mode = 2 )` aktiviert, gilt: – Die RTC6 variiert die aktuelle Laserleistung P im Wobbelfigurabschnitt gemäß $$P(i, n) = P_{nom} \times (1 + (\sum_{j=1}^{i-1} dPower_j \times Period_j) + dPower_i \times n)$$ mit $1 \le i \le$ Wobbelvektor-Anzahl und $1 \le n \le Period_i$. n bezeichnet die laufende Nummer des jeweiligen Mikroschritts. Die nominelle Laserleistung P_{nom} wird über den Listenbefehl **set_wobbel_control** eingestellt. – Wenn mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktiviert, wird die nominelle Laserleistung P_{nom} multiplikativ über den Laserleistungsparameter `P` der mehrfachen McBSP-Übertragung verändert: $P_{nom\ McBSP} = P_{nom} \times P / 16.384$. – Dabei ist zu beachten, dass für jede der beiden obigen Korrekturen nur eine maximale Laserleistung von $P_{nom} \times 4.0$ zulässig ist, wobei der maximale Stellbereich des Lasersteuerparameters ebenfalls nicht überschritten werden kann (siehe **set_wobbel_control**). – Bei einer Laserleistungsvariation mit `Ctrl` = 5 (Halfperiod) muss `dPower` mit umgekehrtem Vorzeichen verwendet werden. Im Gegensatz zu der “Automatischen Laser-Steuerung” kann der Multiplikationsfaktor nicht invertiert werden. – Mit einer “leeren” Wobbelfigur `set_wobbel_vector(0.0, 0.0, 1, 0.0)` kann die nominelle Laserleistung P_{nom} auch multiplikativ verändert werden, ohne explizit wobbeln zu müssen (für eine weitere Alternative siehe auch **set_multi_mcbsp_in**). – Beim Ausschalten der Wobbelfigur (nicht beim Deaktivieren über **set_wobbel_mode**) wird die nominelle Laserleistung P_{nom} an dem über `Ctrl` zugeordneten Port ausgegeben, wenn **set_wobbel_control** zuletzt mit `Value` = 0xFFFFFFFF aufgerufen wurde, ansonsten die mit dem externen Faktor aus **set_multi_mcbsp_in** multiplizierte Laserleistung P_{nom}.

Unverzögerter Kurzer Listenbefehl	**set_wobbel_vector** • Wenn mit **set_wobbel_control** und `set_wobbel_mode( Mode = 3 )` aktiviert, gilt: – Die RTC6 variiert die aktuelle Laserleistung P im Wobbelfigurabschnitt gemäß $$P(i, n) = P_{100} \times (Faktor + (\sum_{j=1}^{i-1} dPower_j \times Period_j) + dPower_i \times n)$$ – P100 ist (wie bei `Mode` = 2) die nominelle Laserleistung, wie sie auch ohne Wobbelfigur vorliegen würde. Sie wird über `set_wobbel_control( Ctrl, Value, MinValue, MaxValue )` eingestellt: • Bei `Value` = 0…65.535 wird der angegebene `Value`-Wert verwendet • Bei `Value` = 0xFFFFFFFF wird derjenige Wert übernommen, der zuletzt an den Port `Ctrl` ausgegeben wurde – Faktor ist ein 16-Bit-Wert ohne Vorzeichen mit der Skalierung 16.384 = Eins (Default-Wert). Der Faktor-Anfangswert kann mit `Value` = 0xFF00nnnn und einem nochmaligen **set_wobbel_control**-Aufruf auf den Wert nnnn eingestellt werden. Die Parameter `Ctrl`, `MinValue` und `MaxValue` werden aus dem zuletzt erfolgten **set_wobbel_control**-Aufruf übernommen. – Um die 100%-Leistung zu ändern, kann jederzeit **set_wobbel_control** aufgerufen werden. Der Faktor-Anfangswert wird jedoch nur wirksam, wenn der Wobbelmode *neu* eingeschaltet wird. Zur Laufzeit der Wobbelfigur kann Faktor nicht mit **set_wobbel_control** geändert werden. – Faktor wird bei jedem Mikroschritt additiv um den `dPower` -Wert aus `set_wobbel_vector( dTrans, dLong, Period, dPower )` verändert. Der Maximalwert ist 65.535 (Faktor 4) und kann nicht überschritten werden. – Eine Leistungsmodulation über McBSP (wie bei `Mode` = 2) ist nicht möglich.

Unverzögerter Kurzer Listenbefehl	**set_wobbel_vector**
Hinweise (Forts.)	• Ist beim Ausschalten der Wobbelfigur der Wobbelmode an sich noch aktiv, wird automatisch auf `Mode` = 1 (liegende Acht) umgeschaltet. • Auch der Wobbelmode mit “Frei definierbaren Wobbelfiguren” muss mit **set_wobbel_mode** eingeschaltet werden. • Die Variation der Laserleistung in einer Wobbelfigur kann nicht mit der variablen Laserleistung aus der “Automatischen Laser-Steuerung” oder der Vektorkontrollierte Laser-Steuerung (parametrisierte [*]mark[*]-Befehle) kombiniert werden. Die Wobbelvariation überschreibt die anderen Variationen, wenn der jeweilige `Ctrl`-Parameter derselbe ist. Unterschiedliche `Ctrl`-Parameter sind zwar zulässig, dürften aber kaum einen praktischen Sinn ergeben. • Achten Sie bei der Definition von “Frei definierbaren Wobbelfiguren” darauf, dass die Mikroschritte eines jeden einzelnen Wobbelvektors die maximale Positioniergeschwindigkeit nicht wesentlich überschreiten! Anderenfalls droht unter Umständen eine Überhitzung der Galvanometerscanner, siehe auch Kapitel 8.4.1 “Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte”. • **create_dat_file** speichert eine “Frei definierbare Wobbelfigur”, siehe Hinweis auf Seite 381.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_multi_mcbsp_in**, **set_multi_mcbsp_in_list**, **set_wobbel_control**, **set_wobbel_offset**, **set_wobbel_vector_2**

Unverzögerter Kurzer Listenbefehl	**set_wobbel_vector_2**
Funktion	Wie **set_wobbel_vector**. Jedoch nur für **set_wobbel_control**(`Ctrl` = 8).
Aufruf	`set_wobbel_vector_2( dTrans, dLong, Period, dPower, dPower2, Ctrl )`
Parameter	`dTrans` Wie **set_wobbel_vector**. `dLong` Wie **set_wobbel_vector**. `Period` Wie **set_wobbel_vector**. `dPower` Mikroschritt der relativen Laserleistung. Bezieht sich auf ANALOG OUT1. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–1.0…+1.0]. `dPower2` Mikroschritt der relativen Laserleistung. Bezieht sich auf ANALOG OUT2. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [–1.0…+1.0]. `Ctrl` Reserviert. Als 32-Bit-Wert ohne Vorzeichen. Überlaufende Werte werden auf die Randwerte geclippt.
Multi-Board-Befehlsname	**n_set_wobbel_vector_2**
Hinweise	• **set_wobbel_vector_2** kann nur mit **set_wobbel_control**(`Ctrl` = 8) benutzt werden. • Wie **set_wobbel_vector**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 631, OUT 632.
Verweise	**set_wobbel_vector**

Kontrollbefehl	**set_zoom**
Aufruf	`set_zoom( Zoom )`
Multi-Board-Befehlsname	**n_set_zoom**
Hinweise	• Dieser Befehl ist z.B. im Handbuch für den intelli*WELD* II FT beschrieben.

Unverzögerter Kurzer Listenbefehl	**set_zoom_list**
Aufruf	`set_zoom_list( Zoom )`
Multi-Board-Befehlsname	**n_set_zoom_list**
Hinweise	• Dieser Befehl ist z.B. im Handbuch für den intelli*WELD* II FT beschrieben.

Kontrollbefehl	**simulate_encoder**
Funktion	Aktiviert oder deaktiviert die Encodersimulation für den oder die angegebenen Encoder.
Aufruf	`simulate_encoder( EncoderNo )`
Parameter	`EncoderNo` Nummer des Encoders. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Es werden ENCODER X-Pulse simuliert und damit Encoderzähler "Encoder0" hochgezählt. = 2: Es werden ENCODER Y-Pulse simuliert und damit Encoderzähler "Encoder1" hochgezählt. = 3: Es werden ENCODER X-Pulse *und* ENCODER Y-Pulse simuliert. = 0: Die Encodersimulation wird deaktiviert.
Multi-Board-Befehlsname	**n_simulate_encoder**
Hinweise	• Die Encodersimulation erfolgt über einen internen Takt von 1 MHz, siehe auch Abschnitt "Encodersimulation", Seite 324. • **simulate_encoder** führt keinen Reset der Encoderzähler aus. • Für `EncoderNo` > 3 wird **simulate_encoder** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • **simulate_encoder** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_encoder**, **store_encoder**, **read_encoder**, **wait_for_encoder**, **set_fly_x**, **set_fly_y**, **set_fly_rot**

Normaler Listenbefehl	**simulate_ext_start**
Funktion	Löst nach der angegebenen Streckenverzögerung einen simulierten externen Start aus.
Aufruf	`simulate_ext_start( Delay, EncoderNo )`
Parameter	`Delay` Streckenverzögerung (Zählerschritte des mit `EncoderNo` ausgewählten Encoderzählers). Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}\ldots+(2^{31}-1)]$. `EncoderNo` Nummer des zu verwendenden Encoderzählers. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoderzähler "Encoder0". = 1: Encoderzähler "Encoder1".
Multi-Board-Befehlsname	**n_simulate_ext_start**
Hinweise	• Zu externen Starts siehe Abschnitt "Externer Start", Seite 315. • Die Streckenverzögerung wird in (relativen) Zähleinheiten des ausgewählten Encoderzählers angegeben (die RTC6-Encoderzähler werden durch ein externes oder simuliertes Encodersignal getriggert, siehe Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323). Der Start-Trigger für den Start erfolgt erst, wenn der interne Encoderzähler die angegebene Streckenverzögerung erreicht hat. Externe Starts, die mit **simulate_ext_start** oder einem externen Startsignal angetriggert, aber entsprechend der eingestellten Streckenverzögerung nicht sofort ausgeführt wurden, werden in einer Warteschleife von bis zu 8 Starts gehalten. **simulate_ext_start** vernichtet die bisherige Warteschleife und beginnt eine neue. • Ein Start-Trigger, initiiert durch **simulate_ext_start** oder ein externes Startsignal, löst nur dann einen Start aus, wenn er nicht erfolgt, während der BUSY-Listenausführungsstatus (wie bei Listenausgabe), der INTERNAL-BUSY-Listenausführungsstatus (wie z.B. bei **goto_xy**) und/oder der PAUSED-Listenausführungsstatus (nach **pause_list**, **stop_list** oder **set_wait**) gesetzt ist. Sonst wird Bit #11 des **get_startstop_info**-Rückgabewerts gesetzt. Wird eine ungeeignete Streckenverzögerung (z.B. `Delay = 0`) angegeben, so wird daher kein Start ausgelöst. Wird **simulate_ext_start** als erster Befehl einer Liste verwendet, kann mit **get_startstop_info** überprüft werden, ob die Abarbeitung der Liste auch innerhalb der vorgegebenen Streckenverzögerung abgeschlossen werden kann. • Das Vorzeichen der Streckenverzögerung (Parameter `Delay`) sollte passend zur jeweiligen Encoderzählrichtung ausgewählt werden (diese richtet sich bei externer Triggerung nach der Bewegungsrichtung des Werkstücks). • Streckenverzögerungen werden auch mit **set_ext_start_delay** oder **set_ext_start_delay_list** eingestellt. Bei der Initialisierung (mit **load_program_file**), durch einen externen Stopp und durch **stop_execution** wird die Steckenverzögerung deaktiviert. Sie kann auch mit **set_control_mode**(Bit #2) deaktiviert werden. • **simulate_ext_start** allein führt *keinen* Encoderreset aus. Wenn es zuvor so mit **set_control_mode**(Bit #9) eingestellt wurde, löst dagegen ein Start-Trigger (initiiert durch **simulate_ext_start** oder ein externes Startsignal) einen Encoderreset aus. • Für `EncoderNo` > 1 wird **simulate_ext_start** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).

Normaler Listenbefehl	**simulate_ext_start**
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 den angegebenen Wert für `Delay` mit 16. Entsprechend verringert sich der zulässige Wertebereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**simulate_ext_start_ctrl**, **set_ext_start_delay**, **set_ext_start_delay_list** **set_extstartpos**, **set_extstartpos_list**, **set_control_mode**, **simulate_ext_stop**

Kontrollbefehl	**simulate_ext_start_ctrl**
Funktion	Löst einen simulierten externen Start aus.
Aufruf	`simulate_ext_start_ctrl()`
Multi-Board-Befehlsname	**n_simulate_ext_start_ctrl**
Hinweise	• Zu externen Starts siehe Abschnitt "Externer Start", Seite 315. • Der Start-Trigger für den Start erfolgt erst nach einer zuvor mit **set_ext_start_delay**, **set_ext_start_delay_list** oder **simulate_ext_start** eingestellten Streckenverzögerung. Bei der Initialisierung (mit **load_program_file**), durch einen externen Stopp und durch **stop_execution** wird die Steckenverzögerung deaktiviert. Sie kann auch mit **set_control_mode** (Bit #2) deaktiviert werden. Externe Starts, die mit **simulate_ext_start_ctrl** oder einem externen Startsignal angetriggert, aber entsprechend der eingestellten Streckenverzögerung nicht sofort ausgeführt wurden, werden in einer Warteschleife von bis zu 8 Starts gehalten. **simulate_ext_start_ctrl** vernichtet im Gegensatz zu **simulate_ext_start** die bisherige Warteschleife *nicht*. • Ein Start-Trigger, initiiert durch **simulate_ext_start_ctrl** oder ein externes Startsignal, löst nur dann einen Start aus, wenn er nicht während der Ausgabe einer Liste erfolgt (sonst wird Bit #11 des **get_startstop_info**-Rückgabewerts gesetzt). • **simulate_ext_start_ctrl** führt *keinen* Encoderreset aus. Wenn es zuvor so mit **set_control_mode**(Bit #9) eingestellt wurde, löst dagegen ein Start-Trigger, initiiert durch **simulate_ext_start_ctrl,** **simulate_ext_start** oder ein externes Startsignal, einen Encoderreset aus, falls er auf einen der "Processing on the fly"-Befehle **set_fly_x**, **set_fly_y**, **set_fly_2d** oder **set_fly_rot** folgt. • **simulate_ext_start_ctrl** kann mit **set_control_mode**(Bit #4 = 1) oder **set_control_mode_list**(Bit #4 = 1) deaktiviert werden. • Ab Version DLL 609, OUT 609, RBF 614 gilt: wenn ein Ausführungsstart erlaubt ist, wartet **simulate_ext_start_ctrl** 30 μs, bevor er zurückkehrt. Dadurch ist eine mögliche Timing-Lücke (bei **get_status**) zwischen Befehlsaufruf und tatsächlichem Start geschlossen.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 609, OUT 609, RBF 614: s.o.
Verweise	**simulate_ext_start**

Kontrollbefehl	simulate_ext_stop
Funktion	Löst einen simulierten externen Stopp aus.
Aufruf	`simulate_ext_stop()`
Multi-Board-Befehlsname	**n_simulate_ext_stop**
Hinweise	• Zu externen Stopps, siehe Abschnitt "Externer Stopp", Seite 314. • **simulate_ext_stop** hält sowohl die Master-Karte als auch alle Slave-Karten gleichzeitig an. • Im Gegensatz dazu hält **stop_execution** nur die Master-Karte an.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**simulate_ext_start**, **simulate_ext_start_ctrl**, **stop_execution**, **sync_slaves**

Unverzögerter Kurzer Listenbefehl	**spot_distance**
Funktion	Wie **spot_distance_ctrl**, aber ein Listenbefehl.
Aufruf	`spot_distance( Dist )`
Parameter	`Dist` Wie **spot_distance_ctrl**.
Multi-Board-Befehlsname	**n_spot_distance**
Hinweise	• Siehe **spot_distance_ctrl**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 613.
Verweise	**spot_distance_ctrl**, **set_auto_laser_control**, **set_auto_laser_params**, **set_auto_laser_params_list**

Kontrollbefehl	**spot_distance_ctrl**
Funktion	Legt den geometrischen Pulsabstand für die "Automatische Laser-Steuerung" mit `Ctrl` = 7 fest.
Aufruf	`spot_distance_ctrl( Dist )`
Parameter	`Dist` Pulsabstand. In Bits. Als Wert im 64-Bit-IEEE Gleitkommaformat.
Multi-Board-Befehlsname	**n_spot_distance_ctrl**
Hinweise	• Der Pulsabstand wird mit einer Genauigkeit von 1/40 Bit (Bildfeld-Koordinaten) aufgelöst. • `Dist` = 0 unterdrückt die "Automatische Laser-Steuerung" mit `Ctrl` = 7, aber schaltet sie nicht ab. • `Dist` darf den Maximalwert 26.214 nicht überschreiten. Ein praktischer Wert von `Dist` hängt von der Markiergeschwindigkeit, der Laserfrequenz und dem Scan-System ab.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 609, OUT 609, RBF 613.
Verweise	**spot_distance**, **set_auto_laser_control**, **set_auto_laser_params**, **set_auto_laser_params_list**

Kontrollbefehl	start_loop
Funktion	Aktiviert einen sich wiederholenden, automatischen Listenwechsel.
Aufruf	`start_loop()`
Multi-Board-Befehlsname	**n_start_loop**
Hinweise	• Ein mit **start_loop** aktivierter Listenwechsel oder Ausführungsneustart wird so lange wiederholt, bis die Ausführung durch Aufruf von **quit_loop** beendet wird. • **start_loop** kann zu jedem beliebigen Zeitpunkt aufgerufen werden, wird aber erst mit dem nächsten **set_end_of_list** wirksam. • Wird der automatische Listenwechsel während der Abarbeitung einer Liste aktiviert, fährt die Ausführung bei Erreichen von **set_end_of_list** ohne Verzögerung mit der jeweils anderen Liste fort. Gibt es nur *eine* Liste (`Mem2` = 0, siehe **config_list**), wird die Ausführung der Liste bei Erreichen von **set_end_of_list** bei Pos = 0 (am Listenanfang) fortgesetzt. • Während eine Liste abgearbeitet wird, kann die jeweils andere (aber auch dieselbe) Liste neu geladen werden, siehe Kapitel 6.4.6 "Listen automatisch wechseln", Seite 113. • Damit **start_loop** überhaupt wirkt, muss die gerade aktive Liste unbedingt mit **set_end_of_list** abgeschlossen worden sein, die neue Liste sollte bereits geladen sein oder der Eingabe-Zeiger einen gewissen Vorsprung vor dem Ausgabe-Zeiger haben (anderenfalls werden "alte" Befehle ausgeführt). Wird bei der Listenausführung das Ende der Liste erreicht, ohne dass ein **set_end_of_list** angetroffen wird, so wird mit dem Anfang der aktuellen Liste fortgefahren, nicht mit dem Anfang der jeweils anderen Liste. • Falls während der Abarbeitung einer Liste **start_loop** *und* **auto_change_pos**(`Pos` > 0) aufgerufen werden, so wird beim unmittelbar nächsten **set_end_of_list** der Befehl **auto_change_pos**(`Pos` > 0) ausgeführt, bei den nachfolgenden **start_loop**. • Der aktuelle Listenstatus kann mit **read_status** gelesen werden. • Der aktuelle Listenausführungsstatus kann mit **get_status** gelesen werden. • **start_loop** löst einen Flush der gepufferten Listeneingabe aus, siehe Kapitel 6.4.1 "Listen laden", Seite 108. • **start_loop** ist in Boot-Phase 2 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**quit_loop**

Kontrollbefehl	**stepper_abs**	
Funktion	Löst an den beiden Schrittmotor-Ausgängen Zielpositionsfahrten zu den angegebenen absoluten Zielpositionen aus.	
Aufruf	`stepper_abs( Pos1, Pos2, WaitTime )`	
Parameter	`Pos1`	Absolute Zielposition in CLOCK-Puls-Einheiten für Schrittmotor-Ausgang 1. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [-2^{31}…$+(2^{31}-1)$].
	`Pos2`	Wie `Pos1` (analog).
	`WaitTime`	Legt fest, wann **stepper_abs** spätestens zurückkehrt. Als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1 s*. Zulässiger Wertebereich: [0…$+(2^{32}-1)$].
Multi-Board-Befehlsname	**n_stepper_abs**	
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoren ansteuern", Seite 309. • **stepper_abs** setzt die neuen Zielpositionswerte auch dann, wenn eine bereits zuvor gestartete Zielpositionsfahrt noch läuft: – Liegt `Pos1`/`Pos2` in der aktuellen Bewegungsrichtung dem Wert der internen Positionsvariablen voraus, läuft die Bewegung einfach weiter. Der Busy-Status bleibt gesetzt. – Ist `Pos1`/`Pos2` identisch zum aktuellen Wert der internen Positionsvariablen, stoppt die Bewegung. Der Busy-Status wird zurückgesetzt. – Liegt `Pos1`/`Pos2` in der aktuellen Bewegungsrichtung bereits hinter dem Wert der internen Positionsvariablen zurück, wird die Bewegungsrichtung des entsprechenden Schrittmotors umgedreht, siehe Abschnitt "Hinweise", Seite 309. Der Busy-Status bleibt gesetzt. • Wenn noch keine Zielpositionsfahrt läuft, wird eine gestartet und der Busy-Status gesetzt. • Während der Ausführung einer Referenzfahrt (Init-Status gesetzt, siehe **stepper_init**) wird **stepper_abs** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Falls die CLOCK-Pulsperiode mit **stepper_init**, **stepper_control** oder **stepper_control_list** auf 0 gesetzt wurde, findet am entsprechenden Schrittmotor-Ausgang keine Schrittmotor-Bewegung statt. • Für `WaitTime` = 0 kehrt **stepper_abs** sofort zurück, damit die Anwendung sofort wieder die Kontrolle über das System hat.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**stepper_abs_no**, **stepper_rel**, **stepper_rel_no**, **stepper_abs_list**	

Unverzögerter Kurzer Listenbefehl	**stepper_abs_list**
Funktion	Wie **stepper_abs**, aber ein Listenbefehl und ohne `WaitTime`-Parameter.
Aufruf	`stepper_abs_list( Pos1, Pos2 )`
Parameter	`Pos1` Wie **stepper_abs**.
	`Pos2` Wie **stepper_abs**.
Multi-Board-Befehlsname	**n_stepper_abs_list**
Hinweise	• Siehe **stepper_abs**. • Während der Ausführung einer Referenzfahrt (siehe **stepper_init**) wird die Ausführung von **stepper_abs_list** so lange verzögert, bis die Referenzfahrt beendet ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_abs**

Kontrollbefehl	**stepper_abs_no**
Funktion	Löst en einem Schrittmotor-Ausgang eine Zielpositionsfahrt zur angegebenen absoluten Zielposition aus.
Aufruf	`stepper_abs_no( No, Pos, WaitTime )`
Parameter	`No` Nummer des Schrittmotor-Ausgangs. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Schrittmotor-Ausgang 1. = 2: Schrittmotor-Ausgang 2. Für einen unzulässigen Wert wird **stepper_abs_no** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	`Pos` Absolute Zielposition in CLOCK-Puls-Einheiten. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}…+(2^{31}-1)]$.
	`WaitTime` Legt fest, wann **stepper_abs_no** spätestens zurückkehrt. *1 Bit entspricht 1 s*. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: $[0…+(2^{32}-1)]$.
Multi-Board-Befehlsname	**n_stepper_abs_no**
Hinweise	• Eine Zielpositionsfahrt wird nur am mit `No` angegebenen Schrittmotor-Ausgang ausgeführt. Ansonsten ist **stepper_abs_no** identisch mit **stepper_abs** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_abs**, **stepper_abs_no_list**

Unverzögerter Kurzer Listenbefehl	**stepper_abs_no_list**
Funktion	Wie **stepper_abs_no**, aber ein Listenbefehl und ohne `WaitTime`-Parameter.
Aufruf	`stepper_abs_no_list( No, Pos )`
Parameter	`No` Nummer des Schrittmotor-Ausgangs. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Schrittmotor-Ausgang 1. = 2: Schrittmotor-Ausgang 2. Für einen unzulässigen Wert wird **stepper_abs_no_list** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	`Pos` Wie **stepper_abs_no**.
Multi-Board-Befehlsname	**n_stepper_abs_no_list**
Hinweise	• Siehe **stepper_abs_no**. • Während der Ausführung einer Referenzfahrt (siehe **stepper_init**) wird die Ausführung von **stepper_abs_no_list** so lange verzögert, bis die Referenzfahrt beendet ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_abs_no**

Kontrollbefehl	**stepper_control**
Funktion	Setzt die Pulsperioden für die CLOCK-Signale der Schrittmotorsteuerung.
Aufruf	`stepper_control( Period1, Period2 )`
Parameter	`Period1` Pulsperiode für die CLOCK-Signale von Schrittmotor-Ausgang 1. 1 Bit entspricht 10 μs. Als 32-Bit-Wert mit Vorzeichen. Zulässige Werte: [0…+(2^{24}-1)] oder < 0. Höhere Werte werden geclippt. > 0: Eine bereits laufende CLOCK-Pulsperiode am jeweiligen Schrittmotor-Ausgang wird abgewartet und dann die neue Periode gestartet. = 0: Eine bereits laufende CLOCK-Pulsperiode am jeweiligen Schrittmotor-Ausgang wird abgebrochen (die entsprechende Schrittmotor-Bewegung wird gestoppt, der Init- und/oder Busy-Status für den jeweiligen Schrittmotor-Ausgang wird zurückgesetzt). < 0: die jeweilige Schrittmotor-Steuerung bleibt unverändert.
	`Period2` Pulsperiode für die CLOCK-Signale von Schrittmotor-Ausgang 2. Sonst wie `Period1`.
Multi-Board-Befehlsname	**n_stepper_control**
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoren ansteuern", Seite 309. • `Period1` oder `Period2`= 0 kann für einen Not-Halt genutzt werden, siehe auch Abschnitt "Endlosfahrten anhalten", Seite 311.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_control_list**

Unverzögerter Kurzer Listenbefehl	**stepper_control_list**
Funktion	Wie **stepper_control**, aber ein Listenbefehl.
Aufruf	`stepper_control_list( Period1, Period2 )`
Parameter	`Period1` Wie **stepper_control**.
	`Period2` Wie **stepper_control**.
Multi-Board-Befehlsname	**n_stepper_control_list**
Hinweise	• Siehe **stepper_control**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_control**

Kontrollbefehl	**stepper_disable_switch**
Funktion	Steuert die Verwendung der SWITCH-Signale der Schrittmotorsteuerung.
Aufruf	`stepper_disable_switch( Disable1, Disable2 )`
Parameter	`Disable1` Anweisung, wie die SWITCH-Signale aus Schrittmotor-Eingang 1 verwendet werden sollen. Als 32-Bit-Wert mit Vorzeichen.. Zulässiger Wertebereich: [-2^{32}…$+(2^{32}-1)$]. > 0: Das SWITCH-Signal am Schrittmotor-Eingang wird nicht verwendet. = 0: Das SWITCH-Signal am Schrittmotor-Eingang wird verwendet. < 0: Die Verwendung des Schrittmotor-Eingangssignals bleibt unverändert.
	`Disable2` Wie `Disable1` (analog).
Multi-Board-Befehlsname	**n_stepper_disable_switch**
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoren ansteuern", Seite 309. • Bei normalen Bewegungsfahrten kann der Endschalter ignoriert werden. Das ist z.B. bei kontinuierlich drehenden Rotationsachsen sinnvoll. • Die SWITCH-Signale werden bei durch **stepper_init** initialisierten Bewegungsfahrten immer verwendet.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_init**

Kontrollbefehl	**stepper_enable**
Funktion	Setzt die ENABLE-Signale der Schrittmotorsteuerung.
Aufruf	`stepper_enable( Enable1, Enable2 )`
Parameter	`Enable1` ENABLE-Signal für Schrittmotor-Ausgang 1. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [-2^{32}…$+(2^{32}-1)$]. > 0: Das ENABLE-Signal am Schrittmotor-Ausgang wird gesetzt. $= 0$: Das ENABLE-Signal am Schrittmotor-Ausgang wird zurückgesetzt. < 0: Das Schrittmotor-Ausgangssignal bleibt unverändert.
	`Enable2` Wie `Enable1` (analog).
Multi-Board-Befehlsname	**n_stepper_enable**
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoren ansteuern", Seite 309.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_enable_list**

Unverzögerter Kurzer Listenbefehl	**stepper_enable_list**
Funktion	Wie **stepper_enable**, aber ein Listenbefehl.
Aufruf	`stepper_enable_list( Enable1, Enable2 )`
Parameter	`Enable1` Wie **stepper_enable**.
	`Enable2` Wie **stepper_enable**.
Multi-Board-Befehlsname	**n_stepper_enable_list**
Hinweise	• Siehe **stepper_enable**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_enable**

Kontrollbefehl	stepper_init	
Funktion	Führt eine Schrittmotor-Initialisierung aus.	
Aufruf	`stepper_init( No, Period, Dir, Pos, Tol, Enable, WaitTime )`	
Parameter	`No`	Nummer des Schrittmotor-Ausgangs. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Schrittmotor-Ausgang 1. = 2: Schrittmotor-Ausgang 2. Für einen unzulässigen Wert wird **stepper_init** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	`Period`	Pulsperiode für die CLOCK-Signale. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 μs. Zulässiger Wertebereich: [0…+(2^{24}-1)]. Höhere Werte werden geclippt. Für `Period` = 0 wird keine Referenzfahrt ausgeführt und die Funktion kehrt sofort zurück (siehe Hinweise).
	`Dir`	Richtung der Schrittmotorbewegung bei der Referenzfahrt. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Bereich: [–2^{31}…+(2^{31}-1)]. > 0: Das DIRECTION-Signal wird gesetzt; die interne Positionsvariable wird bei der Referenzfahrt hochgezählt. = 0: Das DIRECTION-Signal wird zurückgesetzt; die interne Positionsvariable wird bei der Referenzfahrt heruntergezählt. < 0: Das DIRECTION-Signal bleibt unverändert, es wird keine Referenzfahrt ausgeführt und die Funktion kehrt sofort zurück.
	`Pos`	Neuer Positionsvariablenwert. In CLOCK-Puls-Einheiten (siehe Hinweise). Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [–2^{31}…+(2^{31}-1)].
	`Tol`	Toleranzwert für die Referenzfahrt (siehe Hinweise). Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…+(2^{32}-1)]. Für `Dir` < 0 und/oder `Period` = 0 ist der Wert `Tol` = 0 irrelevant, ansonsten wird für `Tol` = 0 **stepper_init** nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	`Enable`	ENABLE-Signal. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…+(2^{32}-1)]. = 0: Das ENABLE-Signal wird zurückgesetzt. > 0: Das ENABLE-Signal wird gesetzt.
	`WaitTime`	Legt fest, wann **stepper_init** spätestens zurückkehrt (siehe Hinweise). *1 Bit entspricht 1 s*. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…+(2^{32}-1)].
Multi-Board-Befehlsname	**n_stepper_init**	

Kontrollbefehl	stepper_init
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 “Schrittmotoren ansteuern”, Seite 309. • **stepper_init** stoppt unmittelbar jede zuvor gestartete Schrittmotor-Bewegung des mit `No` angegebenen Schrittmotors. • Das ENABLE-Signal `Enable` wird lediglich durchgereicht und immer entsprechend gesetzt, hat jedoch keinerlei Einfluss auf die internen Abläufe. • Für `Period` > 0 und `Dir` ≥ 0 wird mit Schrittmotor `No` eine Referenzfahrt mit der angegebenen CLOCK-Pulsperiode in die angegebene Richtung gestartet und der Init-Status gesetzt. Der erste Clock-Puls wird erst nach einer vollen CLOCK-Pulsperiode gesetzt. – Ist gleich zu Beginn ein Endschalter aktiviert, versucht die Steuerung in einem Bereich von ±`Tol` um die aktuelle Position herum eine Position mit deaktiviertem Endschalter zu finden, zunächst entgegengesetzt zur angegebenen Richtung. Gelingt dies nicht, bricht die Suche ab. In diesem Fall bleibt das Status-Bit SWITCH gesetzt (siehe **get_stepper_status**). – Wird ein Endschalter während einer Bewegung aktiviert, so stoppt dort die Referenzfahrt. Anschließend wird die Endschalter-Position 4× überfahren und ein Mittelwert für diese Position berechnet. Zuletzt wird der Schrittmotor mit einer normalen Zielpositionsfahrt (Init-Status zurückgesetzt, Busy-Status gesetzt) um den Toleranzwert `Tol` in die entgegengesetzte Richtung gefahren. Dabei wechselt das DIRECTION-Status-Signal, während es bei Suchfahrten mit mehrmals wechselnder Bewegungsrichtung konstant bleibt. Die interne Positionsvariable (für die aktuelle Position) wird auf den über den Parameter `Pos` festgelegten Wert gesetzt. Dieser Wert entspricht also einer gegenüber dem Endschalter um `Tol` verschobenen Position. Mit `Pos` = `Tol` entspricht die mittlere Endschalterposition der Position 0. – Wird *kein* Endschalter gefunden (z.B. weil in der angegebenen Richtung gar kein Endschalter vorhanden ist), so führt der Schrittmotor eine Endlosfahrt aus. **stepper_init** kehrt dann nach `WaitTime` Sekunden zurück, damit die Anwendung wieder die Kontrolle über das System hat. Die Endlosfahrt des Schrittmotors läuft aber so lange weiter, bis sie mit einem erneuten **stepper_init** oder mit **stepper_control**(`Period1`/`Period2` = 0) gestoppt wird. Siehe hierzu auch Abschnitt “Endlosfahrten anhalten”, Seite 311. • Für `Period` = 0, `Dir` < 0 und/oder `WaitTime` = 0 kehrt **stepper_init** sofort zurück, damit die Anwendung sofort wieder die Kontrolle über das System hat. Das Beenden der Referenzfahrt kann dann mit **get_stepper_status** überprüft werden. Während der Suchfahrt ist der Status “Init” (siehe Seite 477) gesetzt und während der abschließenden Zielpositionsfahrt nach (Endschalter + `Tol`) der Status “Busy” (siehe Seite 477). • `Period` = 0 und/oder `Dir` < 0 können für einen Not-Halt genutzt werden. Es wird dann eine zuvor gestartete Schrittmotor-Bewegung gestoppt, aber keine Referenzfahrt ausgeführt. Auch hier wird die Positionsvariable auf den Wert `Pos` gesetzt. Das DIRECTION-Signal bleibt unverändert. • Für `Period` = 0 werden Status “Init” (siehe Seite 477) und Status “Busy” (siehe Seite 477) zurückgesetzt. Es werden keine weiteren Clock-Pulse mehr ausgegeben, bis `Period` wieder auf einen positiven Wert gesetzt wird.

Kontrollbefehl	stepper_init
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_abs**, **stepper_rel_list**

Kontrollbefehl	stepper_rel	
Funktion	Löst an den beiden Schrittmotor-Ausgängen Zielpositionsfahrten zu den angegebenen relativen Positionen aus.	
Aufruf	`stepper_rel( dPos1, dPos2, WaitTime )`	
Parameter	`dPos1`	Relative Zielposition in CLOCK-Puls-Einheiten für Schrittmotor-Ausgang 1. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}…+(2^{31}-1)]$.
	`dPos2`	Relative Zielposition in CLOCK-Puls-Einheiten für Schrittmotor-Ausgang 2. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}…+(2^{31}-1)]$.
	`WaitTime`	Wie **stepper_abs**.
Multi-Board-Befehlsname	**n_stepper_rel**	
Hinweise	• Die Zielpositionen werden relativ zu den aktuellen Positionswerten angegeben. Ansonsten ist **stepper_rel** identisch mit **stepper_abs** (siehe Hinweise dort).	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**stepper_abs**, **stepper_rel_list**	

Unverzögerter Kurzer Listenbefehl	stepper_rel_list	
Funktion	Wie **stepper_rel**, aber ein Listenbefehl und ohne `WaitTime`-Parameter.	
Aufruf	`stepper_rel_list( dPos1, dPos2 )`	
Parameter	`dPos1`	Wie **stepper_rel**.
	`dPos2`	Wie **stepper_rel**.
Multi-Board-Befehlsname	**n_stepper_rel_list**	
Hinweise	• Siehe **stepper_rel**. • Während der Ausführung einer Referenzfahrt (siehe **stepper_init**) wird die Ausführung von **stepper_rel_list** so lange verzögert, bis die Referenzfahrt beendet ist.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**stepper_rel**	

Kontrollbefehl	**stepper_rel_no**
Funktion	Löst an *einem* Schrittmotor-Ausgang eine Zielpositionsfahrt zur angegebenen relativen Position aus.
Aufruf	`stepper_rel_no( No, dPos, WaitTime )`
Parameter	`No` Wie **stepper_abs_no**. `dPos` Relative Position. In CLOCK-Puls-Einheiten. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}\ldots+(2^{31}-1)]$. `WaitTime` Wie **stepper_abs_no**.
Multi-Board-Befehlsname	**n_stepper_rel_no**
Hinweise	• Die Zielposition wird relativ zum aktuellen Positionswert angegeben (der Positionswert wird entsprechend neu gesetzt) und es wird nur am mit `No` angegebenen Schrittmotor-Ausgang eine Zielpositionsfahrt ausgeführt. Ansonsten ist **stepper_rel_no** identisch mit **stepper_abs** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_abs_no**, **stepper_abs**, **stepper_rel_no_list**

Unverzögerter Kurzer Listenbefehl	**stepper_rel_no_list**
Funktion	Wie **stepper_rel_no**, aber ein Listenbefehl und ohne `WaitTime`-Parameter.
Aufruf	`stepper_rel_no_list( No, dPos )`
Parameter	`No` Nummer des Schrittmotor-Ausgangs. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Schrittmotor-Ausgang 1. = 2: Schrittmotor-Ausgang 2. Für einen unzulässigen Wert wird **stepper_rel_no_list** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). `dPos` Wie **stepper_rel_no**.
Multi-Board-Befehlsname	**n_stepper_rel_no_list**
Hinweise	• Siehe **stepper_rel_no**. • Während der Ausführung einer Referenzfahrt (siehe **stepper_init**) wird die Ausführung von **stepper_rel_no_list** so lange verzögert, bis die Referenzfahrt beendet ist.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**stepper_rel_no**

Normaler Listenbefehl	**stepper_wait**
Funktion	Unterbricht die weitere Ausführung einer Liste so lange, bis eine (am angegebenen Schrittmotor-Ausgang) zuvor gestartete Schrittmotor-Bewegung beendet ist.
Aufruf	`stepper_wait( No )`
Parameter	`No` Nummer des Schrittmotor-Ausgangs. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Schrittmotor-Ausgang 1. = 2: Schrittmotor-Ausgang 2. = 0, 3: Beide Schrittmotor-Ausgänge. Es werden nur die 2 niederwertigsten Bits ausgewertet.
Multi-Board-Befehlsname	**n_stepper_wait**
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoren ansteuern", Seite 309. • Falls für den angegebenen Schrittmotor-Ausgang zuvor *keine* Schrittmotor-Bewegung gestartet wurde, benötigt **stepper_wait** trotzdem 10 µs zur Ausführung, bleibt aber ansonsten ohne Auswirkung. • **stepper_wait** beeinflusst *nicht*: – die Signale für den "Laser active"-Betrieb – den Listenstatus – den Listenausführungsstatus
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	–

Kontrollbefehl	stop_execution
Funktion	Beendet die Ausführung der Liste sofort und deaktiviert ohne Verzug die Signale für den "Laser active"-Betrieb.
Aufruf	`stop_execution()`
Multi-Board-Befehlsname	**n_stop_execution**
Hinweise	• **stop_execution** deaktiviert die Signale für den "Laser active"-Betrieb auch dann, wenn keine Liste aktiv ist (**stop_execution** hat aber dann ansonsten keine Auswirkungen; auch hier gilt: **get_last_error**-Returncode `RTC6_BUSY`). • Die Galvanometerscanner bleiben bei **stop_execution** in der aktuellen Position stehen, es sei denn, es wurde zuvor mit **home_position** oder **home_position_xyz** ein Home-Jump definiert (dieser wird ausgeführt). Vor dem Laden einer neuen Liste sollten die Galvanometerscanner deshalb mit **goto_xy** in eine definierte Stellung gedreht werden. • Die externe START-Eingabe wird deaktiviert, siehe Abschnitt "Externer Start", Seite 315. • Die "Processing on the fly"-Korrektur wird ausgeschaltet. • Die BUSY-Listenstatuswerte (siehe **read_status**) und der BUSY-Listenausführungsstatus-Listenausführungsstatus (siehe **get_status**) werden zurückgesetzt. • Eine Liste, die mit **stop_execution** unterbrochen wurde, kann *nicht* fortgesetzt werden. Sie muss mit **execute_list_pos** (o.ä.) neu gestartet werden. Soll eine Liste nur vorübergehend unterbrochen und später fortgesetzt werden, kann **pause_list** verwendet werden. • **stop_execution** wirkt nur auf die angesprochene RTC6-Karte. **stop_execution** wird in einer Master/Slave-Kette nicht zu den Slave-Karten durchgereicht. Wenn alle RTC6-Karten einer Master/Slave-Kette gleichzeitig angehalten werden sollen, dann muss ein **simulate_ext_stop** oder ein Externer Stopp an irgendeine Karte der Master/Slave-Kette ausgegeben werden, siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität. Jedoch: Master/Slave-Änderung.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_startstop_info**

Kontrollbefehl	stop_list
Funktion	Unterbricht die Ausführung der Liste und deaktiviert die Signale für den "Laser active"-Betrieb.
Aufruf	`stop_list()`
Multi-Board-Befehlsname	**n_stop_list**
Hinweise	• **stop_list** ist gleichbedeutend mit **pause_list** (siehe Hinweise dort).
RTC4→RTC6	Grundsätzlich unveränderte Funktionalität. Allerdings: Zusätzlicher PAUSED-Listenausführungsstatus. Dieser wird durch **stop_list** gesetzt.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**pause_list**

Kontrollbefehl	stop_trigger
Funktion	Setzt den mit **measurement_status** auslesbaren Status der Messwertaufzeichnung zurück (auf `Busy`= 0).
Aufruf	`stop_trigger()`
Multi-Board-Befehlsname	**n_stop_trigger**
Hinweise	• **stop_trigger** wird nur benötigt, wenn mit **set_trigger[*]** eine Messwertaufzeichnung gestartet, aber nicht anschließend beendet wurde mit: – **set_trigger**(`Period` = 0) – **set_trigger4**(`Period` = 0) – **set_trigger8**(`Period` = 0) Der Status der Messwertaufzeichnung kann damit auch außerhalb einer aktiven Liste zurückgesetzt werden (siehe Hinweise bei **set_trigger**). • **stop_trigger** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **stop_trigger** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt)
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**measurement_status**, **set_trigger**, **set_trigger4**, **set_trigger8**

Unverzögerter Kurzer Listenbefehl	**store_encoder**
Funktion	Speichert die aktuellen Zählerstände der beiden RTC6-Encoderzähler in einen Zwischenspeicher der RTC6.
Aufruf	`store_encoder( Pos )`
Parameter	`Pos` Speicherposition. Als 32-Bit-Wert ohne Vorzeichen. • Bit #0 = 0: Die Zählerstände werden in Speicherposition 0 gespeichert. • Bit #0 = 1: Die Zählerstände werden in Speicherposition 1 gespeichert.
Multi-Board-Befehlsname	**n_store_encoder**
Hinweise	• **store_encoder** kann u.a. dazu verwendet werden, die exakte Anzahl von Encoderschritten während einer bestimmten Listenausführungszeit zu ermitteln. Dazu können beispielsweise die Zählerstände zu Beginn einer Befehlsliste mit `store_encoder(0)` in Speicherposition 0 und am Ende der Befehlsliste mit `store_encoder(1)` in Speicherposition 1 gespeichert und anschließend mit **read_encoder** ausgelesen werden. • Siehe auch Kapitel 8.6 ""Processing on the fly"", Seite 254 und Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**read_encoder**, **get_encoder**, **set_fly_x**, **set_fly_y**, **set_fly_rot**, **wait_for_encoder**

Kontrollbefehl	**store_program**
Funktion	Standalone-Funktionalität: Speichert (löscht) Daten für das automatische Booten in den (aus dem) NAND-Speicher.
Voraussetzung	RTC6-Software-Paket ≥ V1.7.0 und BIOS-ETH ≥ 26.
Aufruf	`Error = store_program( Mode )`
Parameter	`Mode` = 0: Speichert die Daten für den "Standalone-Basis-Zustand" (s.u.). = 1: Löscht den NAND-Speicher-Inhalt. = 2: Wie = 0, zusätzlich weitere Dateien für den "Standalone-Voll-Zustand" (s.u.). ≥ 3: Wie = 2, aber ohne Listenbefehle. Vorgesehen nur für Remote-Interface-Modus (Listenbefehle können dort nachträglich geladen werden). Als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	`Error` Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. Wert – Beschreibung 0 – Kein Fehler. 1 – NAND-Speicher nicht ansprechbar. 2 – NAND-Speicher-Ende vorzeitig erreicht. 3 – Daten wurden nicht oder nur teilweise im NAND-Speicher gespeichert. 4 – Keine RTC6 Ethernet-Karte.
Multi-Board-Befehlsname	**n_store_program**
Hinweise	• **store_program** ist nur bei RTC6 Ethernet-Karten erlaubt. Anderenfalls wird ein **get_last_error**-Returncode `RTC6_TYPE_REJECTED` erzeugt. • **store_program** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • Während der Ausführung von **store_program** ist der 10 µs-Takt des DSP unterbrochen: – `Mode` = 0 etwa 02 s – `Mode` = 2 etwa 60 s – `Mode` = 3 etwa 10 s • Die Daten für den "Standalone-Basis-Zustand" sind: – `RTC6ETH.out`, `RTC6RBF.rbf` und `RTC6DAT.dat` • Die Daten für den "Standalone-Voll-Zustand" sind: – Die Daten für den "Standalone-Basis-Zustand" (siehe oberer Listenpunkt) – Die erforderlichen Kontrollbefehle, Listenbefehle und Korrekturdateien • Im Fehlerfall wird ein **get_last_error**-Returncode `RTC6_FLASH_ERROR` erzeugt. • Siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 618, OUT 618, RBF 623. Letzte Änderung DLL 643, OUT 645: `Mode` ≥ 3.
Verweise	**read_image_eth**, **write_image_eth**

Kontrollbefehl	store_timestamp_counter
Funktion	Speichert den aktuellen 32-Bit-"Zeitstempel-Zähler"-Wert.
Aufruf	`store_timestamp_counter()`
Parameter	Keine.
Multi-Board-Befehlsname	**n_store_timestamp_counter**
Hinweise	• Siehe Kapitel 8.12 "Zeitmessungen", Seite 293. • Der aktuelle 32-Bit-"Zeitstempel-Zähler"-Wert wird als Zeitreferenz `TimeStampStorage` für **wait_for_timestamp_counter** gespeichert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**store_timestamp_counter_list**, **wait_for_timestamp_counter**

Unverzögerter Kurzer Listenbefehl	store_timestamp_counter_list
Funktion	Wie **store_timestamp_counter**, aber ein Listenbefehl.
Aufruf	`store_timestamp_counter_list()`
Parameter	Keine.
Multi-Board-Befehlsname	**n_store_timestamp_counter_list**
Hinweise	• Siehe **store_timestamp_counter**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**store_timestamp_counter**, **wait_for_timestamp_counter**

Unverzögerter Kurzer Listenbefehl	**sub_call**
Funktion	Bewirkt einen unbedingten Sprung zu einem indizierten Unterprogramm.
Aufruf	`sub_call( Index )`
Parameter	`Index` Index des aufzurufenden indizierten Unterprogramms. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Multi-Board-Befehlsname	**n_sub_call**
Hinweise	• **sub_call** liest die Startadresse des indizierten Unterprogramms entsprechend dem ausgewählten Index aus der internen Verwaltungstabelle und ruft dann **list_call** auf (siehe auch Hinweise dort). **list_call** löst dann den Sprung auf das Unterprogramm aus. • **sub_call** startet sowohl indizierte Unterprogramme im geschützten Bereich (die mit **load_sub**, **load_disk** oder **copy_dst_src** geladen und/oder referenziert wurden) als auch indizierte Unterprogramme im ungeschützten Listenbereich (die mit **set_sub_pointer** oder **copy_dst_src** referenziert wurden). • Ist für den angegebenen Index kein Unterprogramm referenziert, dann unterbleibt der Sprung und die Ausführung fährt mit dem der aufrufenden Stelle unmittelbar folgenden Befehl fort. Gegebenenfalls wird ein **list_continue** ausgeführt. Ob für einen bestimmten Index ein Unterprogramm referenziert ist, kann mit `get_sub_pointer( Index )` abgefragt werden. Falls *kein* Unterprogramm referenziert ist, gibt dieser Befehl einen Wert von "–1" (= 2^{32}–1) zurück. • für `Index` > 1023 wird **sub_call** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Absolute Vektor-Befehle und Bogen-Befehle werden nach Aufruf mit **sub_call** auch absolut ausgeführt. Wenn das Unterprogramm an unterschiedlichen Stellen im Bildfeld ausgeführt werden soll, darf es nur relative [*]mark[*]-Befehle, Arc-Befehle und Sprung-Befehle enthalten oder es ist **sub_call_abs** zu verwenden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_call**, **sub_call_abs**, **sub_call_cond**

Unverzögerter Kurzer Listenbefehl	**sub_call_abs**
Funktion	Bewirkt einen unbedingten Sprung zu einem indizierten Unterprogramm. Im aufgerufenen Unterprogramm werden absolute Vektor-Befehle und Bogen-Befehle mit einem Offset (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) beaufschlagt.
Aufruf	`sub_call_abs( Index )`
Parameter	`Index` Index des aufzurufenden indizierten Unterprogramms. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Multi-Board-Befehlsname	**n_sub_call_abs**
Hinweise	• **sub_call_abs** liest die Startadresse des indizierten Unterprogramms entsprechend dem ausgewählten Index aus der internen Verwaltungstabelle und ruft dann **list_call_abs** auf (siehe auch Hinweise dort). **list_call_abs** löst dann den Sprung auf das Unterprogramm aus. • Enthält ein aufzurufendes Unterprogramm keine absoluten Befehle, besteht zwischen **sub_call_abs** und **sub_call** kein Unterschied.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**sub_call**, **sub_call_abs_cond**

Unverzögerter Kurzer Listenbefehl	**sub_call_abs_cond**
Funktion	*Bedingter Aufruf (AbsCall) eines indizierten Unterprogramms*: **sub_call_abs_cond** führt **sub_call_abs**(`Index`) aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`sub_call_abs_cond( Mask1, Mask0, Index )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
	`Index` Index des aufzurufenden indizierten Unterprogramms. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Multi-Board-Befehlsname	**n_sub_call_abs_cond**
Hinweise	• Siehe **sub_call_abs**. • Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**sub_call_abs**

Unverzögerter Kurzer Listenbefehl	**sub_call_abs_repeat**
Funktion	Bewirkt einen unbedingten Sprung zu einem indizierten Unterprogramm und führt dessen Rumpf mehrmals aus.
Aufruf	`sub_call_abs_repeat( Index, Number )`
Parameter	`Index` Index des aufzurufenden indizierten Unterprogramms (wie bei **sub_call_abs**). `Number` Anzahl der Wiederholungen. 0 wird wie 1 behandelt. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_sub_call_abs_repeat**
Hinweise	• `sub_call_abs( Index )` ist gleichbedeutend mit `sub_call_abs_repeat( Index, 1 )`. • Siehe **sub_call_repeat** und **sub_call_abs**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**sub_call_repeat**, **sub_call_abs**, **sub_call**

Unverzögerter Kurzer Listenbefehl	**sub_call_cond**
Funktion	*Bedingter Aufruf eines indizierten Unterprogramms*: **sub_call_cond** führt **sub_call**(`Index`) aus, falls der aktuelle Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste die folgende Bedingung erfüllt: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (= falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Anderenfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`sub_call_cond( Mask1, Mask0, Index )`
Parameter	`Mask1` 16-Bit-Maske. Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask0` Siehe `Mask1`.
	`Index` Index des aufzurufenden indizierten Unterprogramms. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…1023].
Multi-Board-Befehlsname	**n_sub_call_cond**
Hinweise	• Siehe **sub_call**. • Siehe auch Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**sub_call**

Unverzögerter Kurzer Listenbefehl	**sub_call_repeat**
Funktion	Bewirkt einen unbedingten Sprung zu einem indizierten Unterprogramm und führt dessen Rumpf mehrmals aus.
Aufruf	`sub_call_repeat( Index, Number )`
Parameter	`Index` Index des aufzurufenden indizierten Unterprogramms (wie bei **sub_call**). `Number` Anzahl der Wiederholungen. Als 32-Bit-Wert ohne Vorzeichen. `Number` = 0 wird wie `Number` = 1 behandelt.
Multi-Board-Befehlsname	**n_sub_call_repeat**
Hinweise	• `sub_call( Index )` ist gleichbedeutend mit `sub_call_repeat( Index, 1 )`. • **sub_call_repeat** vermeidet bei der Wiederholung einen Leertakt, der sonst bei Konstruktionen wie `sub_call`...`sub_call` oder `list_repeat`...`sub_call`...`list_until` unvermeidlich ist. • Mit **sub_call_repeat** können z.B. Trajektorien für Anlauf- und Auslaufkurven aus micro_vector[*]-Befehlen mit Figuren aus Unterprogrammen nahtlos aneinander gefügt werden.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**sub_call_abs_repeat**, **sub_call**, **sub_call_abs**, **micro_vector_abs**, **micro_vector_abs_3d**, **micro_vector_rel**, **micro_vector_rel_3d**

Unverzögerter Kurzer Listenbefehl	**switch_ioport**
Funktion	Führt einen relativen Listensprung **list_jump_rel**(`Pos`) aus, dessen Sprungweite `Pos` (>1) erst zur Laufzeit durch den aktuellen Wert (`IOvalue`) am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste festgelegt wird. Es kann festgelegt werden, welche Bits des 16-Bit-Digital-Eingangs dafür ausgewertet werden sollen.
Aufruf	`switch_ioport( MaskBits, ShiftBits )`
Parameter	`MaskBits` Anzahl der unmittelbar nebeneinander liegenden Bits des 16-Bit-Digital-Eingangs, die zur Ermittlung der Sprungweite ausgewertet werden sollen. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1…16]. `ShiftBits` Nummer des niederwertigsten auszuwertenden Bits des 16-Bit-Digital-Eingangs. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…15].
Multi-Board-Befehlsname	**n_switch_ioport**
Hinweise	• Für unzulässige Werte von `MaskBits` oder `ShiftBits` und für (`MaskBits` + `ShiftBits`) > 16 wird **switch_ioport** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Es gilt: `Mask` = ((1 << `MaskBits`) – 1) << `ShiftBits` und `SwitchNo` = (`Mask` & `IOvalue`) >> `ShiftBits`. Es wird dann ein **list_jump_rel**(`Pos`) mit `Pos` = (`SwitchNo` + 1) Listenpositionen ausgeführt. • Die Sprungweite ist mindestens 1, um mögliche Endlosschleifen zu unterbinden, wenn kein Signal anliegt. Sprünge auf die eigene Adresse (`Pos` = 0) sind mit **switch_ioport** nicht möglich, aber simulierbar mit **list_jump_rel**(–1) als unmittelbar nachfolgendem Befehl. • Die Sprungweite kann maximal 2^{16} Listenpositionen betragen. • Siehe auch **list_jump_rel**. • Siehe auch Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81 und Kapitel 9.3.2 "Ausführung Bedingter Befehle", Seite 320.
Beispiel (Pascal)	• Es wird angenommen, dass zur Laufzeit am 16-Bit-Digital-Eingang der aktuelle Wert $F152 beträgt: **switch_ioport**($0008, $0004) führt dann **list_jump_rel**($0016) aus, also einen relativen Listensprung mit einer Sprungweite von 22 Listenpositionen.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**list_jump_rel**, **list_jump_rel_cond**

Kontrollbefehl	**sync_slaves**
Funktion	Ohne Funktion. **sync_slaves** ist ab DLL 614, OUT 614, RBF 619 nicht mehr erforderlich. Synchronisiert alle Slave-Karten (die in einer Master/Slave-Kette mit dem Slave-Anschluss der angesprochenen RTC6-Karte verbunden sind) stabil auf den 10 μs-Takt der angesprochenen RTC6-Karte.
Aufruf	`sync_slaves()`
Multi-Board-Befehlsname	**n_sync_slaves**
Hinweise	• Zur Verwendung von **sync_slaves**, siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127. • Mit RTC6-Software-Paket ≥ V1.5.2 werden die nachfolgend für RTC6-Software-Pakete < V1.5.0 beschriebenen Aktionen nicht mehr ausgeführt. • Für RTC6-Software-Pakete < 1.5.0 (< RBF 619): – SCANLAB empfiehlt, die Synchronisation unmittelbar nach Initialisierung aller Karten (mit **load_program_file** und **load_correction_file**) durchzuführen. Anderenfalls sollten alle beteiligten Karten (d. h. die Master-Karte und die nachgeschalteten Slave-Karten, die sich im Besitz des Anwenderprogramms befinden) bereits vor dem Aufruf von **sync_slaves** angehalten sein. Um Störungen während der Synchronisation zu vermeiden, dürfen außerdem während der Ausführung von **sync_slaves** weder externe Stopp-Signale an die Karten angelegt noch externe Starts ausgelöst werden (dies wird nicht automatisch verhindert). – Während des Ablaufs von **sync_slaves** wird u.a. an die angesprochene Karte ein **simulate_ext_stop** übergeben. Dadurch werden die angesprochene Karte und alle in der Master/Slave-Kette nachgeschalteten Slave-Karten angehalten (auch diejenigen Karten, die sich gegebenenfalls nicht im Besitz des Anwenderprogramms befinden). Benutzer sind selber dafür verantwortlich, dafür zu sorgen, dass eventuell laufende Prozesse dadurch nicht gestört werden. – Nach Ausführung von **sync_slaves** befinden sich die Scan-System-Achsen aller beteiligten Karten entweder in Koordinaten-Mittelstellung (0, 0 [,0]) oder in HomeJump-Position (jeweils unter Beibehaltung etwaiger Offset-Einstellungen z.B. über **set_offset**, **set_defocus** oder **set_hi**). – **sync_slaves** wirkt bezüglich der Synchronisierung nur auf diejenigen RTC6-Karten, die in einer Master/Slave-Kette mit dem Master-Anschluss der angesprochenen RTC6-Karte verbundenen sind, dagegen nicht auf die angesprochene Karte selbst und auch nicht auf am Slave-Anschluss angeschlossene Karten. Sollen alle Slave-Karten einer Master/Slave-Kette auf die Master-Karte synchronisiert werden, dann muss deshalb mit **sync_slaves** die Master-Karte der Master/Slave-Kette angesprochen werden. **sync_slaves** hat keine Auswirkung, wenn am Master-Anschluss der angesprochenen Karte keine Karte angeschlossen ist.

Kontrollbefehl	sync_slaves
Hinweise (Forts.)	– Die Synchronisation der nachgeschalteten Slave-Karten mit **sync_slaves** erfolgt auch dann, wenn der BUSY-Listenausführungsstatus oder der INTERNAL-BUSY-Listenausführungsstatus dieser Karten gesetzt ist (sie werden automatisch angehalten). Synchronisiert werden jedoch nur diejenigen Slave-Karten, für die das Anwenderprogramm eine Zugriffsberechtigung besitzt (der Besitz wird nicht automatisch angefordert). Falls das Anwenderprogramm außer für die angesprochene Karte für keine weitere Karte eine Zugriffsberechtigung besitzt, hat **sync_slaves** keine Auswirkung. – Zusätzlich werden im Ablauf von **sync_slaves** für alle existierenden Karten, die im Besitz des Anwenderprogramms sind (auch für vorgeschaltete Karten), alle **get_startstop_info**-Fehler-Bits gelöscht. • **sync_slaves** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus der angesprochenen Karte gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus der angesprochenen Karte gesetzt ist • **sync_slaves** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus ist gesetzt)
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität. Jedoch: Master/Slave-Funktionalität geändert.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 615: **sync_slaves** hat jetzt keine Funktion mehr.
Verweise	**get_master_slave**, **get_sync_status**, **master_slave_config**

Kontrollbefehl	**time_control_eth**
Funktion	Stellt einen Parameter zur Feinabstimmung der Ganggenauigkeit der Echtzeituhr ein.
Aufruf	`time_control_eth( PPM )`
Parameter	`PPM` Gangabweichung. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: `PPM` / 4,34 = [–64…+63].
Multi-Board-Befehlsname	**n_time_control_eth**
Hinweise	• Die Ausführung von **time_control_eth** kann mehrere 100 µs dauern, siehe **time_update**. • **time_control_eth** wird bei RTC6 PCIe-Karten nicht ausgeführt. • `PPM` bedeutet die Gangabweichung in parts per million. Ein positiver Wert verlangsamt die Uhr.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 612, ETH 612, RBF 617.
Verweise	**time_update**

Normaler Listenbefehl	**time_fix**
Funktion	Speichert die aktuelle Uhrzeit und das aktuelle Datum von RTC-Uhr und -Kalender in einen Zwischenpuffer zur Verwendung bei **mark_date** und **mark_time**.
Aufruf	`time_fix()`
Multi-Board-Befehlsname	**n_time_fix**
Hinweise	• **time_fix** ist gleichbedeutend mit **time_fix_f_off** mit `FirstDay` = 0 und `Offset` = 0 (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**time_fix_f**, **time_fix_f_off**

Normaler Listenbefehl	**time_fix_f**
Funktion	Speichert die aktuelle Uhrzeit und das aktuelle Datum von RTC-Uhr und -Kalender in einen Zwischenpuffer zur Verwendung bei **mark_date** und **mark_time**.
Aufruf	`time_fix_f( FirstDay )`
Parameter	`FirstDay` Wie **time_fix_f_off**.
Multi-Board-Befehlsname	**n_time_fix_f**
Hinweise	• **time_fix_f** ist gleichbedeutend mit **time_fix_f_off** mit `Offset` = 0 (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**time_fix**, **time_fix_f_off**, **time_update**, **mark_time**, **mark_date**

Normaler Listenbefehl	**time_fix_f_off**	
Funktion	Speichert die aktuelle Uhrzeit und das aktuelle Datum von RTC-Uhr und -Kalender oder ein vordatiertes Datum und eine vordatierte Uhrzeit in einen Zwischenpuffer zur Verwendung bei **mark_date** und **mark_time**.	
Aufruf	`time_fix_f_off( FirstDay, Offset )`	
Parameter	`FirstDay`	Legt für die Ermittlung des julianischen Kalendertages aus dem aktuellen Datum des RTC-Kalenders den Beginn der Zählung fest: Die Zählung läuft von `FirstDay` bis `FirstDay` + 364 (+1 für Schaltjahre). Als 32-Bit-Wert ohne Vorzeichen.
	`Offset`	Vordatierung. In Sekunden. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0…(2^{32}–1)].
Multi-Board-Befehlsname	**n_time_fix_f_off**	
Hinweise	• Vor dem Aufruf von **time_fix_f_off**, **time_fix_f** oder **time_fix** sollte (für RTC6-Karten nach jedem **load_program_file** mindestens einmalig) mit **time_update** ein Abgleich zwischen RTC6 und PC-Zeit durchgeführt werden. • Die vollständige Uhrzeit wird durch mehrfachen Aufruf von **mark_time** und das vollständige Datum durch mehrfachen Aufruf von **mark_date** markiert. **time_fix_f_off**, **time_fix_f** oder **time_fix** muss daher *vor* diesen Markierungsbefehlen aufgerufen werden, damit sich die zu markierende Uhrzeit oder das Datum während der Markierung nicht mehr ändern kann. • Werden **time_fix_f_off**, **time_fix_f** oder **time_fix** vor einer Uhrzeit- oder Datumsmarkierung nicht erneut aufgerufen, so wird die zuletzt markierte Zeit ein weiteres Mal markiert. Falls **time_fix_f_off**, **time_fix_f** oder **time_fix** nach einem **load_program_file** noch gar nicht aufgerufen wurden, wird 00:00 Uhr oder 1. Januar 2000 markiert. • Für `Offset` = 0 werden aktuelles Datum und aktuelle Uhrzeit fixiert. Mit einer Vordatierung (`Offset` > 0) kann z.B. ein Haltbarkeitsdatum in Abhängigkeit vom aktuellen Datum fixiert werden. • Eine Rückdatierung (`Offset` < 0) ist nicht möglich.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**time_fix**, **time_fix_f**, **time_update**, **mark_time**, **mark_date**	

Kontrollbefehl	time_update
Funktion	Stellt die 24-Stunden-Uhr und den Kalender der RTC6-Karte auf die aktuelle PC-Zeit ein.
Aufruf	`time_update()`
Multi-Board-Befehlsname	**n_time_update**
Hinweise	• **time_update** muss nach jedem **load_program_file** aufgerufen werden, wenn die 24-Stunden-Uhr und der Kalender der Karte mit der PC-Zeit synchronisiert werden sollen. • Der Basiswert für die interne Zeitzählung wird durch **load_program_file** auf 1. Januar 2000, 00:00 Uhr, durch **time_update** auf die aktuelle PC-Zeit gesetzt. Ein interner Sekundenzähler wird sowohl durch **load_program_file** als auch durch **time_update** auf 0 gesetzt, läuft aber in jedem Fall im quarzgesteuerten 10 μs-Takt weiter. • Vor dem Markieren mit **mark_date** oder **mark_time** muss **time_fix**, **time_fix_f** oder **time_fix_f_off** aufgerufen werden, um die dann jeweils aktuelle Zeit (Summe aus Basiswert und aktuellem Wert des internen Sekundenzählers) festzuhalten und zu formatieren. • Die RTC6 Ethernet-Karte besitzt eine Echtzeit-Uhr. **time_update** muss daher nur einmal aufgerufen werden. **time_update** dauert mehrere hundert μs. Während dieser Zeit ist der 10μs-Takt der RTC6 Ethernet-Karte unterbrochen. **time_update** sollte daher nicht während eine Listenabarbeitung aufgerufen werden. Die Uhr läuft auch bei einem Abschalten der Stromversorgung weiter (> etwa 1 Woche). Siehe auch **time_control_eth**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600. Letzte Änderung DLL 612, ETH 612, RBF 617: Echtzeit-Uhr der RTC6 Ethernet-Karte.
Verweise	**time_fix**, **time_fix_f**, **time_fix_f_off**, **mark_date**, **mark_time**, **time_control_eth**

Normaler Listenbefehl	**timed_arc_abs**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer von der aktuellen Position entlang eines Kreisbogens mit dem angegebenen Winkel um den angegebenen Mittelpunkt (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`timed_arc_abs( X, Y, Angle, T )`
Parameter	`X` Wie **arc_abs**.
	`Y` Wie **arc_abs**.
	`Angle` Wie **arc_abs**.
	`T` Dauer der Kreisbogenmarkierung. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_arc_abs** wie **arc_abs**.
Multi-Board-Befehlsname	**n_timed_arc_abs**
Hinweise	• Im Gegensatz zu **arc_abs** führt **timed_arc_abs** die Markierung nicht mit der (über **set_mark_speed** oder **set_mark_speed_ctrl**) angegebenen Markiergeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikroschritte) der angegebenen Dauer des Kreisbogens angepasst, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286. Die gesamte Markierzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • Siehe auch Hinweise von **arc_abs**.
RTC4→RTC6	Neuer Befehl. Siehe **arc_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Siehe **arc_abs**.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**arc_abs**, **timed_arc_rel**

Normaler Listenbefehl	**timed_arc_rel**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer von der aktuellen Position entlang eines Kreisbogens mit dem angegebenen Winkel um den angegebenen Mittelpunkt (relative Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`timed_arc_rel( dX, dY, Angle, T )`
Parameter	`dX` Wie **arc_rel**.
	`dY` Wie **arc_rel**.
	`Angle` Wie **arc_rel**.
	`T` Dauer der Kreisbogenmarkierung. In µs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_arc_rel** wie **arc_rel**.
Multi-Board-Befehlsname	**n_timed_arc_rel**
Hinweise	• Die Koordinaten des Bogenzentrums werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_arc_rel** analog zu **timed_arc_abs** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `dX` und `dY` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_arc_abs**, **arc_rel**

Normaler Listenbefehl	**timed_jump_abs**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`timed_jump_abs( X, Y, T )`
Parameter	`X` Wie **jump_abs**.
	`Y` Wie **jump_abs**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_jump_abs** wie **jump_abs**.
Multi-Board-Befehlsname	**n_timed_jump_abs**
Hinweise	• Im Gegensatz zu **jump_abs** führt **timed_jump_abs** den Sprung nicht mit der (über **set_jump_speed** oder **set_jump_speed_ctrl**) angegebenen Sprunggeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikroschritte) der angegebenen Sprungdauer des Vektors angepasst, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286. Die gesamte Sprungzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • Siehe auch Hinweise von **jump_abs**.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 die angegebenen Werte für `X` und `Y` mit 16. Entsprechend verringern sich die zulässigen Wertebereiche.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**jump_abs**, **timed_jump_rel**, **timed_jump_abs_3d**

Normaler Listenbefehl	**timed_jump_abs_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_jump_abs_3d** wie **timed_jump_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_jump_abs_3d( X, Y, Z, T )`
Parameter	`X` Wie **jump_abs_3d**.
	`Y` Wie **jump_abs_3d**.
	`Z` Wie **jump_abs_3d**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_jump_abs_3d** wie **jump_abs_3d**.
Multi-Board-Befehlsname	**n_timed_jump_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **timed_jump_abs_3d** gleich wie **timed_jump_abs** (siehe Hinweise dort). • Siehe auch Hinweise von **jump_abs_3d**.
RTC4→RTC6	Neuer Befehl. Siehe **jump_abs_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_jump_abs**, **jump_abs_3d**, **timed_jump_rel_3d**

Normaler Listenbefehl	**timed_jump_rel**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`timed_jump_rel( dX, dY, T )`
Parameter	`dX` Wie **jump_rel**.
	`dY` Wie **jump_rel**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_jump_rel** wie **jump_rel**.
Multi-Board-Befehlsname	**n_timed_jump_rel**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_jump_rel** identisch mit **timed_jump_abs** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Siehe **jump_rel**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_jump_abs**, **jump_rel**, **timed_jump_rel_3d**

Normaler Listenbefehl	**timed_jump_rel_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_jump_rel_3d** wie **timed_jump_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_jump_rel_3d( dX, dY, dZ, T )`
Parameter	`dX` Wie **jump_rel_3d**.
	`dY` Wie **jump_rel_3d**.
	`dZ` Wie **jump_rel_3d**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_jump_rel_3d** wie **jump_rel_3d**.
Multi-Board-Befehlsname	**n_timed_jump_rel_3d**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_jump_rel_3d** identisch mit **timed_jump_abs_3d** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl. Siehe **jump_rel_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_jump_abs_3d**, **jump_rel_3d**, **timed_jump_rel**

Normaler Listenbefehl	**timed_mark_abs**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`timed_mark_abs( X, Y, T )`
Parameter	`X` Wie **mark_abs**.
	`Y` Wie **mark_abs**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_mark_abs** wie **mark_abs**.
Multi-Board-Befehlsname	**n_timed_mark_abs**
Hinweise	• Im Gegensatz zu **mark_abs** führt **timed_mark_abs** die Markierung nicht mit der (über **set_mark_speed** oder **set_mark_speed_ctrl**) angegebenen Markiergeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikroschritte) der angegebenen Dauer des Markvektors angepasst, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286. Die gesamte Markierzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • Siehe auch Hinweise von **mark_abs**.
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Siehe **mark_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**mark_abs**, **timed_mark_rel**, **timed_mark_abs_3d**

Normaler Listenbefehl	**timed_mark_abs_3d**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_mark_abs_3d** wie **timed_mark_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_mark_abs_3d( X, Y, Z, T )`
Parameter	`X` Wie **mark_abs_3d**.
	`Y` Wie **mark_abs_3d**.
	`Z` Wie **mark_abs_3d**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_mark_abs_3d** wie **mark_abs_3d**.
Multi-Board-Befehlsname	**n_timed_mark_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **timed_mark_abs_3d** gleich wie **timed_mark_abs** (siehe Hinweise dort). • Siehe auch Hinweise von **mark_abs_3d**.
RTC4→RTC6	Neuer Befehl. Siehe **mark_abs_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_mark_abs**, **mark_abs_3d**, **timed_mark_rel_3d**

Normaler Listenbefehl	**timed_mark_rel**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld.
Aufruf	`timed_mark_rel( dX, dY, T )`
Parameter	`dX` Wie **mark_rel**.
	`dY` Wie **mark_rel**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_mark_rel** wie **mark_rel**.
Multi-Board-Befehlsname	**n_timed_mark_rel**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_mark_rel** analog zu **timed_mark_abs** (siehe Hinweise dort).
RTC4→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht. Siehe **mark_rel**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_mark_abs**, **mark_rel**, **timed_mark_rel_3d**

Normaler Listenbefehl	**timed_mark_rel_3d**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_mark_rel_3d** wie **timed_mark_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_mark_rel_3d( dX, dY, dZ, T )`
Parameter	`dX` Siehe **mark_rel_3d**.
	`dY` Siehe **mark_rel_3d**.
	`dZ` Siehe **mark_rel_3d**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_mark_rel_3d** wie **mark_rel_3d**.
Multi-Board-Befehlsname	**n_timed_mark_rel_3d**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_mark_rel_3d** identisch mit **timed_mark_abs_3d** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl. Siehe **mark_rel_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_mark_abs_3d**, **timed_mark_abs**, **mark_rel_3d**, **timed_mark_rel**

Normaler Listenbefehl	**timed_para_jump_abs**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld und variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`timed_para_jump_abs( X, Y, P, T )`
Parameter	`X` Wie **para_jump_abs**.
	`Y` Wie **para_jump_abs**.
	`P` Wie **para_jump_abs**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_jump_abs** wie **para_jump_abs**.
Multi-Board-Befehlsname	**n_timed_para_jump_abs**
Hinweise	• Im Gegensatz zu **para_jump_abs** führt **timed_para_jump_abs** den Sprung nicht mit der (über **set_jump_speed** oder **set_jump_speed_ctrl**) angegebenen Sprunggeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikroschritte) der angegebenen Sprungdauer des Vektors angepasst, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286. Die gesamte Sprungzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • **timed_para_jump_abs** benötigt zwei RTC6-Listenspeicherplätze. Der Befehlsteil auf dem ersten RTC6-Listenspeicherplatz wird zur Laufzeit als Kurzer Listenbefehl ausgeführt, anschließend der zweite Befehlsteil als Normaler Listenbefehl. Beide Befehlsteile werden also im selben 10 μs-Takt ausgeführt, wenn nicht weitere kurze Listenbefehle davor ein **list_continue** zwischen beiden Teilen erzwingen. • Siehe auch Hinweise von **para_jump_abs**.
RTC4→RTC6	Neuer Befehl. Siehe **para_jump_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**para_jump_abs**, **timed_jump_abs**, **jump_abs**, **timed_para_jump_rel**, **timed_para_jump_abs_3d**

Multipler Listenbefehl	**timed_para_jump_abs_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld und variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_para_jump_abs_3d** wie **timed_para_jump_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_para_jump_abs_3d( X, Y, Z, P, T )`
Parameter	`X` Wie **para_jump_abs_3d**.
	`Y` Wie **para_jump_abs_3d**.
	`Z` Wie **para_jump_abs_3d**.
	`P` Wie **para_jump_abs_3d**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_jump_abs_3d** wie **para_jump_abs_3d**.
Multi-Board-Befehlsname	**n_timed_para_jump_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **timed_para_jump_abs_3d** gleich wie **timed_para_jump_abs** (siehe Hinweise dort). • **timed_para_jump_abs_3d** belegt 2 RTC6-Listenspeicherplätze für `T` ≥ 5. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem Hauptteil (Normaler Listenbefehl) ausgeführt. • Siehe auch Hinweise von **para_jump_abs_3d**.
RTC4→RTC6	Neuer Befehl. Siehe **para_jump_abs_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_para_jump_abs**, **para_jump_abs_3d**, **jump_abs_3d**, **timed_para_jump_rel_3d**

Normaler Listenbefehl	**timed_para_jump_rel**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld und variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`timed_para_jump_rel( dX, dY, P, T )`
Parameter	`dX` Siehe **para_jump_rel**.
	`dY` Siehe **para_jump_rel**.
	`P` Siehe **para_jump_rel**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_jump_rel** wie **para_jump_rel**.
Multi-Board-Befehlsname	**n_timed_para_jump_rel**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_para_jump_rel** analog zu **timed_para_jump_abs** (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl. Siehe **para_jump_rel**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_para_jump_abs**, **para_jump_rel**, **timed_jump_rel**, **timed_para_jump_rel_3d**

Multipler Listenbefehl	**timed_para_jump_rel_3d**
Funktion	Bewegt den Ausgabepunkt (für den Laserfokus) in der angegebenen Sprungdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld und variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_para_jump_rel_3d** wie **timed_para_jump_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Sprunggeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_para_jump_rel_3d( dX, dY, dZ, P, T )`
Parameter	`dX` Wie **para_jump_rel_3d**.
	`dY` Wie **para_jump_rel_3d**.
	`dZ` Wie **para_jump_rel_3d**.
	`P` Wie **para_jump_rel_3d**.
	`T` Dauer des gesamten Sprung-Vektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_jump_rel_3d** wie **para_jump_rel_3d**.
Multi-Board-Befehlsname	**n_timed_para_jump_rel_3d**
Hinweise	• Die Koordinaten des Sprung-Vektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_para_jump_rel_3d** analog zu **timed_para_jump_abs_3d** (siehe Hinweise dort). • **timed_para_jump_rel_3d** belegt 2 RTC6-Listenspeicherplätze für T ≥ 5. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem Hauptteil (Normaler Listenbefehl) ausgeführt.
RTC4→RTC6	Neuer Befehl. Siehe **para_jump_rel_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_para_jump_abs_3d**, **para_jump_rel_3d**, **timed_jump_rel_3d**, **timed_para_jump_rel**

Normaler Listenbefehl	**timed_para_mark_abs**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 2D-Bildfeld und variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`timed_para_mark_abs( X, Y, P, T )`
Parameter	`X` Wie **para_mark_abs**.
	`Y` Wie **para_mark_abs**.
	`P` Wie **para_mark_abs**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_mark_abs** wie **para_mark_abs**.
Multi-Board-Befehlsname	**n_timed_para_mark_abs**
Hinweise	• Im Gegensatz zu **para_mark_abs** führt **timed_para_mark_abs** die Markierung nicht mit der (über **set_mark_speed** oder **set_mark_speed_ctrl**) angegebenen Markiergeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikroschritte) der angegebenen Dauer des Markvektors angepasst, siehe Kapitel 8.9 "Zeitbemessene Befehle", Seite 286. Die gesamte Markierzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • **timed_para_mark_abs** belegt 2 RTC6-Listenspeicherplätze für `T` ≥ 5. Der Befehlsteil auf dem ersten RTC6-Listenspeicherplatz wird zur Laufzeit als Kurzer Listenbefehl ausgeführt, anschließend der zweite Befehlsteil als Normaler Listenbefehl. Beide Befehlsteile werden also innerhalb desselben 10 μs-Taktes ausgeführt, wenn nicht weitere Kurze Listenbefehle davor ein **list_continue** zwischen beiden Teilen erzwingen. • Siehe auch Hinweise von **para_mark_abs**.
RTC4→RTC6	Neuer Befehl. Siehe **para_mark_abs**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**para_mark_abs**, **timed_mark_abs**, **mark_abs**, **timed_para_mark_rel**, **timed_para_mark_abs_3d**

Multipler Listenbefehl	**timed_para_mark_abs_3d**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im 3D-Bildfeld und variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_para_mark_abs_3d** wie **timed_para_mark_abs**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_para_mark_abs_3d( X, Y, Z, P, T )`
Parameter	`X` Wie **para_mark_abs_3d**.
	`Y` Wie **para_mark_abs_3d**.
	`Z` Wie **para_mark_abs_3d**.
	`P` Wie **para_mark_abs_3d**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_mark_abs_3d** wie **para_mark_abs_3d**.
Multi-Board-Befehlsname	**n_timed_para_mark_abs_3d**
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert **timed_para_mark_abs_3d** gleich wie **timed_para_mark_abs** (siehe Hinweise dort). • **timed_para_mark_abs_3d** belegt 2 RTC6-Listenspeicherplätze für `T` ≥ 5. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem Hauptteil (Normaler Listenbefehl) ausgeführt. • Siehe auch Hinweise von **para_mark_abs_3d**.
RTC4→RTC6	Neuer Befehl. Siehe **para_mark_abs_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_para_mark_abs**, **para_mark_abs_3d**, **mark_abs_3d**, **timed_para_mark_rel_3d**

Normaler Listenbefehl	**timed_para_mark_rel**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 2D-Bildfeld und variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`timed_para_mark_rel( dX, dY, P, T )`
Parameter	`dX` Wie **para_mark_rel**.
	`dY` Wie **para_mark_rel**.
	`P` Wie **para_mark_rel**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_mark_rel** wie **para_mark_rel**.
Multi-Board-Befehlsname	**n_timed_para_mark_rel**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_para_mark_rel** analog zu **timed_para_mark_abs** (siehe Hinweise dort). • Siehe auch Hinweise von **para_mark_rel**.
RTC4→RTC6	Neuer Befehl. Siehe **para_mark_rel**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_para_mark_abs**, **para_mark_rel**, **timed_mark_rel**, **timed_para_mark_rel_3d**

Multipler Listenbefehl	**timed_para_mark_rel_3d**
Funktion	Führt den Laserfokus in der angegebenen Markierdauer entlang eines 3D-Vektors von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im 3D-Bildfeld. Variiert gleichzeitig den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die Option "3D" nicht freigeschaltet ist oder wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt **timed_para_mark_rel_3d** wie **timed_para_mark_rel**. Die Zerlegung in Mikroschritte wird aber wie bei einem 3D-Befehl berechnet, die effektive Markiergeschwindigkeit in der xy-Ebene wird dadurch beeinflusst.
Aufruf	`timed_para_mark_rel_3d( dX, dY, dZ, P, T )`
Parameter	`dX` Wie **para_mark_rel_3d**.
	`dY` Wie **para_mark_rel_3d**.
	`dZ` Wie **para_mark_rel_3d**.
	`P` Wie **para_mark_rel_3d**.
	`T` Dauer des gesamten Markvektors. In μs. Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: [0…167.772.160]. Der Wert wird auf ein ganzzahliges Vielfaches von 10 gerundet, ein zu großer Wert geclippt. Für `T` < 5 wirkt **timed_para_mark_rel_3d** wie **para_mark_rel_3d**).
Multi-Board-Befehlsname	**n_timed_para_mark_rel_3d**
Hinweise	• Die Koordinaten des Markvektor-Endpunkts werden als relative Koordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist **timed_para_mark_rel_3d** analog zu **timed_para_mark_abs_3d** (siehe Hinweise dort). • **timed_para_mark_rel_3d** belegt 2 RTC6-Listenspeicherplätze für `T` ≥ 5. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem Hauptteil (Normaler Listenbefehl) ausgeführt. • Siehe auch Hinweise von **para_mark_rel_3d**.
RTC4→RTC6	Neuer Befehl. Siehe **para_mark_rel_3d**.
RTC5→RTC6	Unveränderte Funktionalität. Zusätzlich: Wertebereich erhöht.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**timed_para_mark_abs_3d**, **para_mark_rel_3d**, **timed_mark_rel_3d**, **timed_para_mark_rel**

Kontrollbefehl	**transform**	
Funktion	Führt eine Rückwärtstransformation von einzelnen Positionswerten durch.	
Aufruf	`TransformErrorCode = transform( &Sig1, &Sig2, Ptr, Code )`	
Parameter und Parameter-Rückgaben	`Sig1`	Parameter: zu transformierender Positionswert. Als Zeiger auf einen 32-Bit-Wert mit Vorzeichen. Parameterrückgabe: transformierter Positionswert. Als 32-Bit-Wert mit Vorzeichen. Die Eingabewerte werden überschrieben.
	`Sig2`	Wie `Sig1` (analog).
Parameter	`Ptr`	Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, in den die Korrektur- und Transformationseinstellungen, die zur Rückwärtstransformation verwendet werden sollen, zuvor mit **upload_transform** übertragen wurden.
	`Code`	Steuert Details der Rückwärtstransformation, insbesondere welche Teiltransformationen durchgeführt werden: Soll eine Teiltransformationen *nicht* durchgeführt werden, so muss das entsprechende Bit (#2…#5) auf 1 gesetzt werden. Als 32-Bit-Wert ohne Vorzeichen. `Code` hat die gleiche Bedeutung wie bei **get_transform**/**get_transform_offset** (`Sig1` entspricht `Ptr1`, `Sig2` entspricht `Ptr2`). Für Bit #0 = 0 werden die beiden angegebenen Positionswerte (`Sig1` und `Sig2`) als xy-Koordinaten rückwärtstransformiert: Bit #1 = 0: Der unter `Sig1` angegebene Wert wird als x-Koordinate rückwärtstransformiert. Der unter `Sig2` angegebene Wert wird als y-Koordinate rückwärtstransformiert. = 1: Der unter `Sig1` angegebene Wert wird als y-Koordinate rückwärtstransformiert. Der unter `Sig2` angegebene Wert wird als x-Koordinate rückwärtstransformiert. Bit #2 = 0: Die Gain/Offset-Korrektur der automatischen Selbstkalibrierung wird rückwärtstransformiert. Bit #3 = 0: Die Bildfeld-Korrektur wird rückwärtstransformiert. Bit #4 = 0: Der Offset der definierten Koordinatentransformation wird rückwärtstransformiert. Bit #5 = 0: Die Gesamtmatrix der definierten Koordinatentransformation wird rückwärtstransformiert. Bit #6 Reserviert. … … Bit #31 Reserviert.

Kontrollbefehl	transform
Parameter (Forts.)	`Code` (Forts.) Für Bit #0 = 1 wird nur einer der beiden angegebenen Positionswerte als z-Koordinate rückwärtstransformiert: Bit #1 = 0: Der unter `Sig1` angegebene Wert wird als z-Koordinate rückwärtstransformiert. `Sig2` bleibt unverändert. = 1: Der unter `Sig2` angegebene Wert wird als z-Koordinate rückwärtstransformiert. `Sig1` bleibt unverändert. Bit #2 = 0: Der mit **set_defocus** oder **set_defocus_list** definierte Offset auf die Fokuslänge wird rückwärtstransformiert. Bit #3 = 0: Die ABC-Korrektur wird rückwärtstransformiert. Bit #4 = 0: Der mit **set_offset_xyz** oder **set_offset_xyz_list** definierte Offset auf die z-Koordinate wird rückwärtstransformiert. Bit #5 Reserviert. … … Bit #31 Reserviert.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Erfolg. 1 `Ptr` = NULL (kein Speicherbereich angegeben). 2 Unter `Ptr` stehen keine gültigen Daten (**upload_transform** wurde nicht ausgeführt). 3 Unter `Ptr` stehen fehlerhafte Daten (eine entsprechende Fehler-Information wurde durch **upload_transform** eingetragen). 4 z-Achsen-Invertierung nicht möglich.
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Zur Rückwärtstransformation von Positionswerten siehe Kapitel 8.1.3 "Positionierung überwachen", Seite 226. • Vor der Ausführung von **transform** muss **upload_transform** ausgeführt worden sein. Außerdem sollten mit **get_values** Positionswerte ausgelesen worden sein. • Falls bei der Ausführung von **transform** ein Fehler auftritt (zurückgegebener Fehlercode > 0), wird keine Transformation ausgeführt (`Sig1` und `Sig2` bleiben dann unverändert). Als Fehler zählen auch `Ptr` = NULL (Fehlercode = 1) oder ein Fehler während der vorherigen Ausführung von **upload_transform** (Fehlercode = 3). • Wird eine Rückwärtstransformation von z-Positionswerten veranlasst (`Code`-Bit #0 = 1), war aber zum Zeitpunkt der zuvor erfolgten Ausführung von **upload_transform** nur eine 2D-Korrekturtabelle zugewiesen, so werden die Offsets auf die Fokuslänge oder die z-Koordinate mit 0 und die Werte A, B, C mit 0, 1, 0 initialisiert (1zu1-Rückwärtstransformation).

Kontrollbefehl	transform
Hinweise (Forts.)	• Bei der Rückwärtstransformation von xy-Positionswerten (`Code`-Bit #0 = 0) wird nur die z = 0-Ebene transformiert. xy-Stretching und z-Defokus infolge von z-Abweichungen, insbesondere bei Nicht-F-Theta-Systemen, werden nicht berücksichtigt. • **transform** greift nicht auf eine RTC6-Karte zu und ist daher auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. Wenn die **upload_transform**-Daten sowie die mit **get_values** oder **get_waveform** ausgelesenen Aufzeichnungsdaten auf dem PC binär gespeichert werden, ist auch ein Offline-Betrieb für **transform** möglich (es muss dann für **transform** keine RTC6-Karte am PCIe-Bus vorhanden sein). • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **transform** weder erzeugt noch verändert.
RTC4→RTC6	Neuer Befehl. Alle rückwärtstransformierten Werte (auch die z-Werte) sind auch im RTC4-Kompatibilitätsmodus im RTC6-20-Bit-Bereich.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**upload_transform**, **get_transform**, **get_transform_offset**, **get_values**

Kontrollbefehl	**uart_config**
Funktion	Konfiguriert die interne UART-Schnittstelle für die angegebenen Baud-Rate.
Aufruf	`RealBaudRate = uart_config( BaudRate )`
Parameter	`BaudRate` Baud-Rate. Als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [160 Bd…12,8 MBd].
Rückgabe	`RealBaudRate`
Multi-Board-Befehlsname	**n_uart_config**
Hinweise	• **uart_config** erweitert **rs232_config** um einen höheren Wertebereich für die Baudrate. • **uart_config** ist gleichbedeutend mit **rs232_config**, gibt aber die nächstmögliche wirklich verwendete Baudrate zurück. • Die Default-Einstellung ist 9.600 Baud. • Die anderen Einstellungen der RS-232-Schnittstelle können nicht verändert werden (Data-Bits: 8, Start-Bits: 1, Stopp-Bits: 1, Parity: None). • Siehe auch Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86. • **uart_config** ist in Boot-Phase 1 zulässig, siehe auch Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 611, OUT 611, RBF 616.
Verweise	**rs232_config**, **rs232_write_data**, **rs232_read_data**

Kontrollbefehl	**upgrade_bios**
Funktion	Flasht das angegebene BIOS auf die RTC6-Karte.
Aufruf	`upgrade_bios( Name )`
Parameter	`Name` Name der BIOS-Datei. Als Zeiger auf einen nullterminierten ANSI-String.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Erfolg. 1 Download-Fehler. 3 BIOS-Flash fehlgeschlagen. 4 Wie 4. 5 Wie 4. 8 Datei nicht gefunden (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). 32 Datei fehlerhaft. 512 DSP-Speicher-Fehler.
Multi-Board-Befehlsname	**n_upgrade_bios**
Hinweise	• BIOS-Dateien für RTC6 PCIe-Karten sind benannt als `RTC6BIOSOUT_*.out`, z.B. `RTC6BIOSOUT_24.out`, Seite 35 • BIOS-Dateien für RTC6 Ethernet-Karten sind benannt als `RTC6BIOSETH_*.out`, z.B.`RTC6BIOSETH_43.out`, Seite 35 • Achtung! Der Flash-Vorgang kann einige Sekunden in Anspruch nehmen. Schalten Sie die Spannungsversorgung der RTC6-Karte in dieser Zeit nicht aus. Anderenfalls wird die RTC6-Karte beschädigt und muss zur Reparatur an SCANLAB eingeschickt werden. • Nach erfolgreichem Flash-Vorgang müssen Sie einen Hardware-Reset durchführen (bei RTC6 Ethernet-Karten alternativ mit **eth_full_reset**).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 647, OUT 650.
Verweise	–

Kontrollbefehl	**upload_transform**
Funktion	Überträgt alle Korrektur- und Transformationseinstellungen, die dem Scan-System aktuell zugeordnet sind, von der RTC6-Karte auf den PC.
Aufruf	`UploadErrorCode = upload_transform( HeadNo, Ptr )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses, dessen Einstellungen ausgelesen werden sollen. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf.
	`Ptr` Zeiger (bei C und C++ Datentyp ULONG_PTR, ein 32-Bit-Wert ohne Vorzeichen oder 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, in den die ausgelesenen Einstellungen übertragen werden sollen.
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. Bit #0 =1: x-Gain = 0 (Gain der automatischen Selbstkalibrierung für Galvanometerscanner 2). Bit #1 =1: y-Gain = 0 (Gain der automatischen Selbstkalibrierung für Galvanometerscanner 1). Bit #2 =1: Die Gesamtmatrix der definierten Koordinatentransformation ist nicht invertierbar. Bit #3 =1: Keine Korrekturtabelle zugewiesen. Bit #4 =1: Die ABC-Werte (z-Achse) sind nicht invertierbar. Bit #5 =1: Fehler beim Lesen der Korrekturtabelle. Bit #6 =1: Parameter-Fehler: unzulässige `HeadNo` oder `Ptr` = 0. Bit #7 =1: Busy-Fehler, Karte hat BUSY-Listenausführungsstatus oder INTERNAL-BUSY-Listenausführungsstatus (**get_last_error**-Returncode `RTC6_BUSY`). Bit #8 Reserviert. … … Bit #31 Reserviert.
Multi-Board-Befehlsname	**n_upload_transform**

Kontrollbefehl	upload_transform
Hinweise	• Die ausgelesenen und übertragenen Daten können für eine Rückwärtstransformation von Ist-Positions-Werten verwendet werden, siehe auch Kapitel 8.1.3 "Positionierung überwachen", Seite 226 mit: – **transform** – **get_transform**/**get_transform_offset** • Zum Speichern eines ausgelesenen Datensatzes muss das Anwenderprogramm jeweils (an einer über `Ptr` angegebenen Adresse) einen Speicherbereich der Größe 528.524 Bytes auf dem PC bereitstellen. • Im Falle eines Fehlers (außer für Bit #6 = 1) wird in `Ptr` eingetragen, dass die Daten fehlerhaft sind. **transform** und **get_transform**/**get_transform_offset** erkennen diese Fehler-Information und führen die Rückwärtstransformation nicht aus (von **transform** wird dann ein entsprechender Fehlercode, bei **get_transform**/**get_transform_offset** ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt). • **upload_transform** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus der angesprochenen Karte gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • **upload_transform** wird auch dann ausgeführt, wenn: – eine Liste mit **set_wait** angehalten wurde (PAUSED-Listenausführungsstatus gesetzt) • Während der Ausführung von **upload_transform** werden externe Starts unterdrückt.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**transform**, **get_transform**, **get_transform_offset**

Kontrollbefehl	verify_checksum
Funktion	Prüft oder erzeugt die Check-Summe einer Korrekturdatei.
Aufruf	`verify_checksum( Name )`
Parameter	`Name` Name der Korrekturdatei. Als Zeiger auf einen nullterminierten ANSI-String.
Rückgabe	Ergebnis der Überprüfung. Als 32-Bit-Wert ohne Vorzeichen. = 0: Kein Fehler (die überprüfte Check-Summe ist OK). = 1: Es wurde eine Check-Summe neu generiert (keine Info über File-Integrität). = 2: Die überprüfte Check-Summe ist fehlerhaft. = 3: Es ist keine Check-Summe ermittelbar (Datei-Fehler etc.).
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.
Hinweise	• Die Überprüfung von Korrekturdatei-Downloads funktioniert nur bei Korrekturdateien, in die eine Check-Summe eingetragen ist (siehe Korrekturdatei laden, Seite 135 und **set_verify**). • **verify_checksum** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC6-Karte verfügbar. • Die Programme `CorrectionFileConverter.exe` (Version 1.04) sowie `correXion5.exe` (Version 1.01) mit `RTC5Base.dll` (Version 1.0.0.4) geben die Check-Summe bereits automatisch mit aus. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133) werden von **verify_checksum** weder erzeugt noch verändert.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**set_verify**

Normaler Listenbefehl	**wait_for_1_axis**
Funktion	"Fly-Erweiterung"-Befehl: Wartet, bis für den angegebenen Modus der Wert über- oder unterschritten wurde.
Aufruf	`wait_for_1_axis( Value, Mode, WaitMode, LaserMode )`
Parameter	`Value` Wert, auf den gewartet werden soll. Als 32-Bit-Wert mit Vorzeichen. `Mode` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. `WaitMode` < 0: Unterschreitung. = 0: Gleichheit. > 0: Überschreitung. Die Galvanometerscanner folgen der Objektbewegung. `WaitMode`+16: Die Galvanometerscanner bleiben stehen. Als 32-Bit-Wert mit Vorzeichen. `LaserMode` = 0: Der Laser bleibt unverändert. > 0: Der Laser wird nach einem LaserOff-Delay ausgeschaltet. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_wait_for_1_axis**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **wait_for_1_axis** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • Abhängig vom eingestellten Modus kann auf einen Encoderwert oder einen (beliebigen) McBSP-Wert gewartet werden. Im Falle von McBSP sollte eine entsprechende "Processing on the fly"-Korrektur eingeschaltet sein. Stellen Sie sicher, dass die Daten im richtigen Format übertragen werden. • Bei `wait_for_1_axis` und `wait_for_2_axes` darf Modus 1…4 nicht angegeben werden. Stattdessen ist Modus 17…20 vorgesehen. Modus 17…18 ist bei einem SCANahead-System zu verwenden, wenn das Warten noch innerhalb der `PreviewTime` erfolgen soll. Außerhalb davon kann Modus 19…20 verwendet werden. Bei intelliSCAN-Systemen sind Modus 17…18 und Modus 19…20 identisch • `LaserMode` = 0: wie bisher. • `LaserMode` > 0: Laser wird nach einem LaserOff-Delay ausgeschaltet. • `WaitMode` ist wie `Mode` von **wait_for_encoder_mode**. • `WaitMode` und `WaitMode`+16 unterscheiden die Galvanometerscanner-Bewegung, nicht **set_fly_2d** und **set_fly_x**/**set_fly_y**. • **wait_for_encoder** mit automatischer positionsabhängiger Richtungsauswahl wird nicht unterstützt. • Bei einem unzulässigen Parameter-Wert wird **wait_for_1_axis** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).

Normaler Listenbefehl	wait_for_1_axis
Hinweise (Forts.)	• Folgende Befehlsaufrufe werden gleichartig ausgeführt: – `wait_for_1_axis( Value, EncoderNo + 19, Mode, 0 )` = `wait_for_encoder_mode( Value, EncoderNo, Mode )` und **set_fly_2d**-Session – `wait_for_1_axis( Value, EncoderNo + 19, Mode+16, 0 )` = `wait_for_encoder_mode( Value, EncoderNo, Mode )` und **set_fly_x**/**set_fly_y**-Session – `wait_for_1_axis( Value, 6, Mode, 0 )` = `wait_for_mcbsp( Axis, Value, Mode )` und **set_fly_x_pos**-Session – `wait_for_1_axis( Value, Axis×4+10, Mode+16, 0 )` = `wait_for_mcbsp( Axis, Value, Mode )` und **set_fly_x_pos**/**set_fly_y_pos**-Session
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**wait_for_2_axes**

Multipler Listenbefehl	**wait_for_2_axes**
Funktion	"Fly-Erweiterung"-Befehl: Wartet, bis für beide Modi (siehe Modus) die Werte innerhalb oder für beide Modi außerhalb des angegebenen Bereichs liegen.
Aufruf	`wait_for_2_axes( ModeX, MinValueX, MaxValueX, ModeY, MinValueY, MaxValueY, WaitMode, LaserMode )`
Parameter	`ModeX` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. `MinValueX` Unterer Grenzwert für den x-Achsen-Modus-Wert, auf den gewartet werden soll. Als 32-Bit-Wert mit Vorzeichen. `MaxValueX` Oberer Grenzwert für den x-Achsen-Modus-Wert, auf den gewartet werden soll. Als 32-Bit-Wert mit Vorzeichen. `ModeY` Modus aus Tabelle 4, Seite 273. Als 32-Bit-Wert ohne Vorzeichen. `MinValueY` Unterer Grenzwert für den y-Achsen-Modus-Wert, auf den gewartet werden soll. Als 32-Bit-Wert mit Vorzeichen. `MaxValueY` Oberer Grenzwert für den y-Achsen-Modus-Wert, auf den gewartet werden soll. Als 32-Bit-Wert mit Vorzeichen. `WaitMode` ≥ 0: Innerhalb der Grenzwerte. < 0: Außerhalb der Grenzwerte. Die Galvanometerscanner folgen der Objektbewegung. `WaitMode`+16: Die Galvanometerscanner bleiben stehen. Als 32-Bit-Wert mit Vorzeichen. `LaserMode` = 0: Der Laser bleibt unverändert. > 0: Der Laser wird nach einem LaserOff-Delay ausgeschaltet Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_wait_for_2_axes**
Hinweise	• Als "Fly-Erweiterung"-Befehl darf **wait_for_2_axes** nicht mit "klassischen" "Processing on the fly"-Befehlen (siehe Fußnote, Seite 254) gemischt verwendet werden. • Siehe Kapitel 8.6 ""Processing on the fly"", Seite 254 und Abschnitt ""Fly-Erweiterung"-Befehle", Seite 271. • **wait_for_2_axes** belegt 2 RTC6-Listenspeicherplätze. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem zweiten Teil als Normaler Listenbefehl ausgeführt. Eventuell anliegende Verzögerte Kurze Listenbefehle werden davor ausgeführt. • Abhängig vom eingestellten Modus kann darauf gewartet werden, ob ein Encoderwert oder ein (beliebiger) McBSP-Wert innerhalb oder außerhalb der Grenzen liegt. Im Falle von McBSP sollte eine entsprechende "Processing on the fly"-Korrektur eingeschaltet sein. Stellen Sie sicher, dass die Daten im richtigen Format übertragen werden. • Bei `wait_for_1_axis` und `wait_for_2_axes` darf Modus 1…4 nicht angegeben werden. Stattdessen ist Modus 17…20 vorgesehen. Modus 17…18 ist bei einem SCANahead-System zu verwenden, wenn das Warten noch innerhalb der `PreviewTime` erfolgen soll. Außerhalb davon kann Modus 19…20 verwendet werden. Bei intelliSCAN-Systemen sind Modus 17…18 und Modus 19…20 identisch.

Multipler Listenbefehl	**wait_for_2_axes**
Hinweise (Forts.)	• `LaserMode` = 0: wie bisher. • `LaserMode` > 0: Laser wird nach einem LaserOff-Delay ausgeschaltet. • `WaitMode` ist wie `Mode` von **wait_for_encoder_mode**. • `WaitMode` und `WaitMode`+16 unterscheiden die Galvanometerscanner-Bewegung, nicht **set_fly_2d** und **set_fly_x**/**set_fly_y**. • Bei einem unzulässigen Parameter-Wert wird **wait_for_2_axes** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Folgende Befehlsaufrufe werden gleichartig ausgeführt: – `wait_for_2_axes( 19, EncXmin, EncXmax, 20, EncYmin, EncYmax, 0, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 0 )` und **set_fly_2d**-Session, intelliSCAN – `wait_for_2_axes( 19, EncXmin, EncXmax, 20, EncYmin, EncYmax, 16, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 0 )` und **set_fly_x**/**set_fly_y**-Session, intelliSCAN – `wait_for_2_axes( 17, EncXmin, EncXmax, 18, EncYmin, EncYmax, 0, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 2 )` und **set_fly_2d**-Session, SCANahead-System – `wait_for_2_axes( 17, EncXmin, EncXmax, 18, EncYmin, EncYmax, 16, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 2 )` und **set_fly_x**/**set_fly_y**-Session, SCANahead-System
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**wait_for_1_axis**

Normaler Listenbefehl	**wait_for_encoder**
Funktion	Wartet, bis am ausgewählten Encoderzähler der angegebene Zählwert erstmalig über- oder unterschritten wurde.
Aufruf	`wait_for_encoder( Value, EncoderNo )`
Parameter	`Value` Zählwert. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}\ldots+(2^{31}-1)]$.
	`EncoderNo` Nummer des zu verwendenden Encoderzählers. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoderzähler “Encoder0”. = 1: Encoderzähler “Encoder1”.
Multi-Board-Befehlsname	**n_wait_for_encoder**
Hinweise	• **wait_for_encoder** ist gleichbedeutend mit **wait_for_encoder_mode** mit Parameter `Mode` = 0 (siehe Hinweise dort).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**wait_for_encoder_mode**

Multipler Listenbefehl	**wait_for_encoder_in_range**
Funktion	Wartet, bis beide Encoderzähler gleichzeitig innerhalb des angegebenen Bereichs liegen (Grenzen eingeschlossen).
Aufruf	`wait_for_encoder_in_range( EncXmin, EncXmax, EncYmin, EncYmax )`
Parameter	`EncXmin` Grenzwert. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}\ldots+(2^{31}-1)]$.
	`EncXmax` Wie `EncXmin` (analog).
	`EncYmin` Wie `EncXmin` (analog).
	`EncYmax` Wie `EncXmin` (analog).
Multi-Board-Befehlsname	**n_wait_for_encoder_in_range**
Hinweise	• Zur Verwendung von **wait_for_encoder_in_range** siehe Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323 und Kapitel 8.6.7 ""Processing on the fly"-Anwendungen synchronisieren", Seite 264. • **wait_for_encoder_in_range** belegt 2 RTC6-Listenspeicherplätze. Der erste Teil wird als Unverzögerter Kurzer Listenbefehl vor dem zweiten Teil als Normaler Listenbefehl ausgeführt. Eventuell anliegende Verzögerte Kurze Listenbefehle werden davor ausgeführt. • Ist der Wert `EncXmin` > `EncXmax`, dann werden beide Werte vertauscht. • Ist der Wert `EncYmin` > `EncYmax`, dann werden beide Werte vertauscht. • Wenn keine encoderbasierte "Processing on the fly"-Korrektur aktiv ist, bewirkt **wait_for_encoder_in_range** lediglich eine Wartezeit (ohne Galvanometerscanner-Bewegung). • Für `EncXmin` = `EncXmax` (oder `EncYmin` = `EncYmax`) kann auf das Erreichen eines konkreten Encoderwerts gewartet werden. • **wait_for_encoder_in_range** ist auch verfügbar, wenn die Option "Processing on the fly" nicht freigeschaltet ist. • **wait_for_encoder_in_range** lässt die Lasersteuersignale unverändert. Soll der Laser während des Wartens aus sein, muss vor **wait_for_encoder_in_range** irgend ein anderer Befehl stehen, der die Signale für den "Laser active"-Betrieb ausschaltet (z.B. ein **list_nop**). • Ob die Galvanometerscanner während des Wartens stehen bleiben oder sich mit den Encoderänderungen mitbewegen, hängt vom aktiven "Processing on the fly"-Modus ab, siehe Kapitel 8.6.7 ""Processing on the fly"-Anwendungen synchronisieren", Seite 264: Mit **set_fly_2d** bewegen sie sich mit, ansonsten bleiben sie stehen.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**wait_for_encoder_mode**, **park_position**, **park_return**

Multipler Listenbefehl	**wait_for_encoder_in_range_mode**
Funktion	Wartet, bis beide Encoderzähler gleichzeitig innerhalb des angegebenen Bereichs liegen (Grenzen eingeschlossen).
Aufruf	`wait_for_encoder_in_range_mode( EncXmin, EncXmax, EncYmin, EncYmax, Mode )`
Parameter	`EncXmin` Grenzwert. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}\ldots+(2^{31}-1)]$.
	`EncXmax` Wie `EncXmin` (analog).
	`EncYmin` Wie `EncXmin` (analog).
	`EncYmax` Wie `EncXmin` (analog).
	`Mode` Modus. Als 32-Bit-Wert mit Vorzeichen. = 0 oder 1: Wartet auf direkte Encoderwerte ("klassisches" Verhalten). = 2: Wartet auf Encoderwerte, die in (**set_scanahead_params** Parameter) `PreviewTime` voraussichtlich anliegen werden (SCANahead-System-Verhalten).
Multi-Board-Befehlsname	**n_wait_for_encoder_in_range_mode**
Hinweise	• Zur Verwendung siehe Kapitel 8.6.7 ""Processing on the fly"-Anwendungen synchronisieren", Seite 264 und Kapitel 9.3.3 "Synchronisation durch Encodersignale", Seite 323. • Siehe auch Hinweise von **wait_for_encoder_in_range**. • Verwenden Sie `Mode` wie bei **wait_for_encoder_mode**: – 0, 1 bei intelliSCAN oder SCANahead-Systemen außerhalb eines Markierprozesses – 2 bei SCANahead-Systemen während eines Markierprozesses • **wait_for_encoder_in_range** ist gleichbedeutend mit `wait_for_encoder_in_range_mode( Mode = 0 )`.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 612, OUT 612, RBF 617.
Verweise	**wait_for_encoder_mode**

Normaler Listenbefehl	**wait_for_encoder_mode**
Funktion	Wartet, bis am ausgewählten Encoderzähler der angegebene Zählwert erstmalig über- oder unterschritten wurde.
Aufruf	`wait_for_encoder_mode( Value, EncoderNo, Mode )`
Parameter	`Value` Zählwert. Als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31}\ldots+(2^{31}-1)]$.
	`EncoderNo` Nummer des zu verwendenden Encoderzählers. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoderzähler “Encoder0”. = 1: Encoderzähler “Encoder1”.
	`Mode` Modus. Als 32-Bit-Wert mit Vorzeichen. = 0: Wartet auf Über-/Unterschreitung (positionsabhängig). > 0: Wartet auf Überschreitung (positionsunabhängig). < 0: Wartet auf Unterschreitung (positionsunabhängig).
Multi-Board-Befehlsname	**n_wait_for_encoder_mode**
Hinweise	• Zur Verwendung von **wait_for_encoder_mode** siehe Kapitel 9.3.3 “Synchronisation durch Encodersignale”, Seite 323. • Für `Mode` = 0 sollten Größe und Vorzeichen des Parameters `Value` passend zur jeweiligen Encoderzählrichtung ausgewählt werden (diese richtet sich bei externer Triggerung nach der Bewegungsrichtung des Werkstücks). Für `Value` > 0 wartet **wait_for_encoder_mode** auf eine Überschreitung, sonst auf eine Unterschreitung. Ist `Value` positiv und bereits kleiner als der aktuelle Encoderzählwert, wartet **wait_for_encoder_mode** auf einen kompletten Zählerumlauf (analoges gilt bei negativem `Value`). Bei einer Zählrate von 1 MHz dauert dies bis zu ca. 36 Minuten! • Für `Mode` ≠ 0 wartet **wait_for_encoder_mode** unabhängig von der aktuellen Position und Bewegungsrichtung auf eine Über-/Unterschreitung des Parameters `Value`. • Für `EncoderNo` > 1 wird **wait_for_encoder_mode** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Wenn keine encoderbasierte “Processing on the fly”-Korrektur aktiv ist, bewirkt **wait_for_encoder_mode** lediglich eine Wartezeit (ohne Galvanometerscanner-Bewegung), wodurch sich (alternativ zu einem externen Start, **set_wait** oder **if_cond**, etc.) eine extern getriggerte Synchronisation realisieren lässt. • **wait_for_encoder_mode** ist auch verfügbar, wenn die Option “Processing on the fly” nicht freigeschaltet ist. • Für `Mode` = 0 ist **wait_for_encoder_mode** gleichbedeutend mit **wait_for_encoder**. • **wait_for_encoder_mode** lässt die Lasersteuersignale unverändert. Soll der Laser während des Wartens aus sein, muss vor **wait_for_encoder_mode** irgend ein anderer Befehl stehen, der die Signale für den “Laser active”-Betrieb ausschaltet, z.B. ein **list_nop**.

Normaler Listenbefehl	**wait_for_encoder_mode**
Hinweise (Forts.)	• Ob die Galvanometerscanner während des Wartens stehen bleiben oder sich mit den Encoderänderungen mitbewegen, hängt vom aktiven “Processing on the fly”-Modus ab (siehe Kapitel 8.6.7 ““Processing on the fly”-Anwendungen synchronisieren”, Seite 264): Mit **set_fly_2d** bewegen sie sich mit, ansonsten bleiben sie stehen. • Mode = 0 oder ±1 wartet auf direkte Encoderwerte: “klassisches” Verhalten. • Mode = ± 2 wartet auf Encoderwerte, die in (**set_scanahead_params** Parameter) `PreviewTime` voraussichtlich anliegen werden: SCANahead-System-Verhalten.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_encoder**, **store_encoder**, **read_encoder**, **simulate_encoder**, **wait_for_encoder**, **wait_for_encoder_in_range**, **park_position**, **park_return**

Normaler Listenbefehl	**wait_for_mcbsp**
Funktion	Wartet, bis der Eingangswert an der McBSP-Schnittstelle den angegebenen Wert erstmalig erreicht, über- oder unterschritten hat.
Aufruf	`wait_for_mcbsp( Axis, Value, Mode )`
Parameter	`Axis` Auswahl, welches Halbwort des Eingangswerts zur Prüfung verwendet wird (s.u.). Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Unteres Halbwort (x-Achse, Galvanometerscanner 2). = 1: Oberes Halbwort (y-Achse, Galvanometerscanner 1).
	`Value` Grenzwert. Als 32-Bit-Wert mit Vorzeichen.
	`Mode` Modus. Als 32-Bit-Wert mit Vorzeichen. = 0: Wartet auf Gleichheit. > 0: Wartet auf Überschreitung. < 0: Wartet auf Unterschreitung.
Multi-Board-Befehlsname	**n_wait_for_mcbsp**
Hinweise	• Zur Verwendung von **wait_for_mcbsp** siehe Kapitel 9.3.4 "Synchronisation und Online-Positionierung mit McBSP-Signalen", Seite 325. • **wait_for_mcbsp** ist vergleichbar zu **wait_for_encoder_mode**, es wird jedoch die McBSP-Schnittstelle abgefragt. • Falls **set_fly_x_pos** und **set_fly_y_pos** *gleichzeitig* aktiviert sind, besteht `Value` aus zwei 16 Bit-Halbworten, siehe Abschnitt "Korrektur über McBSP-Schnittstelle", Seite 257. Nur in diesem Fall hat `Axis` die Bedeutung der Auswahl, welches Halbwort zur Prüfung verwendet wird. In allen anderen Fällen ist `Axis` bedeutungslos und `Value` wird als 32-Bit-Wert mit Vorzeichen interpretiert. • `Axis` muss immer 0 oder 1 sein (auch wenn es bedeutungslos ist). Anderenfalls wird **wait_for_mcbsp** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Wenn weder **set_fly_x_pos**, **set_fly_y_pos** noch **set_fly_rot_pos** aktiviert ist, bewirkt **wait_for_mcbsp** lediglich eine Wartezeit (ohne Galvanometerscanner-Bewegung), wodurch sich (alternativ zu einem externen Start, **set_wait** oder **if_cond**, etc.) eine extern getriggerte Synchronisation realisieren lässt. • **wait_for_mcbsp** ist auch verfügbar, wenn die Option "Processing on the fly" nicht freigeschaltet ist. • **wait_for_mcbsp** lässt die Lasersteuersignale unverändert. Soll der Laser während des Wartens aus sein, muss vor **wait_for_mcbsp** irgend ein anderer Befehl stehen, der die Signale für den "Laser active"-Betrieb ausschaltet (z.B. ein **list_nop**).
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**wait_for_encoder_mode**

Normaler Listenbefehl	**wait_for_timestamp_counter**
Funktion	Wartet auf einen 32-Bit-"Zeitstempel-Zähler"-Wert.
Aufruf	`wait_for_timestamp_counter( TimeStampCounter )`
Parameter	`TimeStampCounter` 32-Bit-Offset, auf den gewartet werden soll. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_wait_for_timestamp_counter**
Hinweise	• **wait_for_timestamp_counter** wartet, bis der aktuelle 32-Bit-"Zeitstempel-Zähler" den Wert `TimeStampStorage` (aus **store_timestamp_counter** / **store_timestamp_counter_list**) + `TimeStampCounter` erreicht hat. • Anliegende Verzögerte Kurze Listenbefehle werden zuvor ausgeführt. • Dadurch können schon beim Laden einer Liste absolute Zeitbezüge ab dem Zeitpunkt `TimeStampStorage` hergestellt werden. Genauigkeit: 10 μs. • Ist der angegebene Zeitpunkt bei Erreichen von **wait_for_timestamp_counter** schon überschritten, wird ein voller 32-Bit-"Zeitstempel-Zähler"-Umlauf abgewartet (Dauer: etwa 12 Stunden). Dies kann mit **stop_execution** oder /STOP abgebrochen werden. • **wait_for_timestamp_counter** ist gleichbedeutend mit **wait_for_timestamp_counter_mode**(`Mode` > 2). • Die Synchronisation mit anderen RTC6-Befehlen, wie **wait_for_encoder** oder bedingten Befehlen, hängt von externer Hardware ab. • Siehe Kapitel 8.12 "Zeitmessungen", Seite 293.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 617, OUT 617, RBF 623.
Verweise	**store_timestamp_counter**, **store_timestamp_counter_list**, **wait_for_timestamp_counter_mode**

Normaler Listenbefehl	**wait_for_timestamp_counter_long**
Funktion	Wartet auf einen absoluten 64-Bit-"Zeitstempel-Zähler"-Wert.
Aufruf	`wait_for_timestamp_counter_long( WaitCounterL, WaitCounterH, MaxWaitTime, Mode )`
Parameter	`WaitCounterL` Unterer Teil des 64-Bit-"Zeitstempel-Zähler"-Werts, auf den gewartet werden soll. Als 32-Bit-Wert ohne Vorzeichen. `WaitCounterH` Oberer Teil des 64-Bit-"Zeitstempel-Zähler"-Werts, auf den gewartet werden soll. Als 32-Bit-Wert ohne Vorzeichen. `MaxWaitTime` Längste Wartezeit. Als 32-Bit-Wert ohne Vorzeichen. 1 Bit entspricht 10 µs. `Mode` Vorgänge nach einem Timeout. Als 32-Bit-Wert ohne Vorzeichen. = 0: RTC6-Befehl ignorieren. Weiter mit dem nächsten Listenbefehl. = 1: Liste mit **stop_execution** abbrechen. = 2: Liste mit **simulate_ext_stop** abbrechen einschließlich Weiterleitung an alle Slaves, siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127 und Kapitel 9.3.1 "Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden", Seite 314. > 2: Endlos warten bis ein Externer Stopp erfolgt.
Multi-Board-Befehlsname	**n_wait_for_timestamp_counter_long**
Hinweise	• **wait_for_timestamp_counter_long** wartet, bis der aktuelle 64-Bit-"Zeitstempel-Zähler" den folgenden Wert erreicht hat: `WaitCounterL` + (`WaitCounterH` << 32) • Dadurch können schon beim Laden einer Liste absolute Zeitbezüge zu einem im Anwenderprogramm festgelegten Synchronisations-Zeitpunkt hergestellt werden. Genauigkeit: 10 µs. • Ein Fehler-Flag wird gesetzt, wenn: – Der angegebene Zeitpunkt beim Erreichen von **wait_for_timestamp_counter_long** schon überschritten ist – Die geforderte Wartezeit um mehr als `MaxWaitTime` jenseits des laufenden 64-Bit-"Zeitstempel-Zähler" liegt Das Fehler-Flag kann mit **get_startstop_info**(Bit #5) ausgelesen werden. Der Parameter `Mode` bestimmt, wie bei einem Timeout fortgefahren werden soll. • Die Synchronisation mit anderen RTC6-Befehlen, wie **wait_for_encoder** oder bedingten Befehlen, hängt von externer Hardware ab. • Siehe Kapitel 8.12 "Zeitmessungen", Seite 293.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 624, OUT 624.
Verweise	**get_timestamp_long**

Normaler Listenbefehl	**wait_for_timestamp_counter_mode**
Funktion	Wie **wait_for_timestamp_counter**. Zusätzlich kann definiert werden, was nach einem Timeout erfolgen soll.
Aufruf	`wait_for_timestamp_counter_mode( TimeStampCounter, Mode )`
Parameter	`TimeStampCounter` 32-Bit-Offset, auf den gewartet werden soll. Als 32-Bit-Wert ohne Vorzeichen.
	`Mode` Vorgänge nach einem Timeout. Als 32-Bit-Wert ohne Vorzeichen. = 0: RTC6-Befehl ignorieren. Weiter mit dem nächsten Listenbefehl. = 1: Liste mit **stop_execution** abbrechen. = 2: Liste mit **simulate_ext_stop** abbrechen einschließlich Weiterleitung an alle Slaves, siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127 und Kapitel 9.3.1 "Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden", Seite 314. > 2: Vollen 32-Bit-"Zeitstempel-Zähler"-Umlauf abwarten (wie bei **wait_for_timestamp_counter**).
Multi-Board-Befehlsname	**n_wait_for_timestamp_counter_mode**
Hinweise	• **wait_for_timestamp_counter_mode** wartet, bis der aktuelle 32-Bit-"Zeitstempel-Zähler" den Wert `TimeStampStorage` (aus **store_timestamp_counter** / **store_timestamp_counter_list**) + `TimeStampCounter` erreicht hat. • Anliegende Verzögerte Kurze Listenbefehle werden zuvor ausgeführt. • Dadurch können schon beim Laden einer Liste absolute Zeitbezüge ab dem Zeitpunkt `TimeStampStorage` hergestellt werden. Genauigkeit: 10 µs. • Ist der angegebene Zeitpunkt bei Erreichen von **wait_for_timestamp_counter_mode** schon überschritten, wird ein Fehler-Flag gesetzt. Dieses kann mit **get_startstop_info**(Bit #5) ausgelesen werden. Mit dem Parameter `Mode` kann eingestellt werden, wie im Fehlerfall fortgefahren werden soll. • Ein Fehler-Flag wird gesetzt, wenn: – Der angegebene Zeitpunkt beim Erreichen von **wait_for_timestamp_counter_mode** schon überschritten ist Das Fehler-Flag kann mit **get_startstop_info**(Bit #5) ausgelesen werden. Der Parameter `Mode` bestimmt, wie bei einem Timeout fortgefahren werden soll. • **wait_for_timestamp_counter** ist gleichbedeutend mit **wait_for_timestamp_counter_mode**(`Mode` > 2). • Die Synchronisation mit anderen RTC6-Befehlen, wie **wait_for_encoder** oder bedingten Befehlen, hängt von externer Hardware ab. • Siehe Kapitel 8.12 "Zeitmessungen", Seite 293.

Normaler Listenbefehl	**wait_for_timestamp_counter_mode**
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 619, OUT 619, RBF 624.
Verweise	**store_timestamp_counter**, **store_timestamp_counter_list**, **wait_for_timestamp_counter**

Kontrollbefehl	**write_8bit_port**
Funktion	Schreibt einen Wert auf den 8-Bit-Digital-Ausgang der EXTENSION 2-Stiftleiste.
Aufruf	`write_8bit_port( Value )`
Parameter	`Value` 8-Bit-Ausgabewert (DATA0…DATA7). Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 8 Bits ausgewertet.
Multi-Board-Befehlsname	**n_write_8bit_port**
Hinweise	• Siehe auch Kapitel 9.1.2 "8-Bit-Digital-Ausgang", Seite 307.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_8bit_port_list**

Verzögerter Kurzer Listenbefehl	**write_8bit_port_list**
Funktion	Wie **write_8bit_port**, aber ein Listenbefehl.
Aufruf	`write_8bit_port_list( Value )`
Parameter	`Value` Wie **write_8bit_port**.
Multi-Board-Befehlsname	**n_write_8bit_port_list**
Hinweise	• Siehe **write_8bit_port**. • Ab Version DLL 602, OUT 602: Wenn der 8-Bit-Digital-Ausgang (z.B. für die Laserleistung) synchron zu den Laserschaltzeiten ausgeführt werden soll, verwenden Sie `set_laser_power( 2, Value )`.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_8bit_port**, **set_laser_power**, **write_port_list**

Kontrollbefehl	write_abc_to_file	
Funktion	Wie **write_abc_to_file_20b**. Jedoch für einen Fokuslängenwert *f* im RTC4-Kompatibilitätsbereich [–32.768…+32.767]).	
Aufruf	`ErrorNo = write_abc_to_file( Name, A, B, C )`	
Parameter	`Name`	Wie **write_abc_to_file_20b**.
	`A`	Koeffizient A der parabolischen Funktion $z_{out} = A + Bf + Cf^2$, die zur Berechnung der z-Ausgabewerte verwendet wird (Fokuslängenwert *f* im RTC4-Kompatibilitätsbereich [–32.768…+32.767]). Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: siehe **load_z_table**.
	`B`	Wie `A` (analog).
	`C`	Wie `A` (analog).
Rückgabe	Wie **write_abc_to_file_20b**.	
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.	
Hinweise	• Wie **write_abc_to_file_20b**. • `write_abc_to_file_20b( A × 16, B, C × 1/16 )` ist gleichbedeutend mit `write_abc_to_file( A, B, C )`.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Unveränderte Funktionalität.	
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.	
Verweise	**read_abc_from_file**	

Kontrollbefehl	write_abc_to_file_20b	
Funktion	Schreibt die ABC-Werte direkt in die angegebene Korrekturdatei im PC. Für einen Fokuslängenwert *f* im RTC6-20-Bit-Bereich [–524.288…+524.287].	
Aufruf	`ErrorNo = write_abc_to_file_20b( Name, A, B, C )`	
Parameter	`Name`	Name der Korrekturdatei. Als Zeiger auf einen nullterminierten ANSI-String.
	`A`	Koeffizient A der parabolischen Funktion $z_{out} = A + Bf + Cf^2$, die zur Berechnung der z-Ausgabewerte verwendet wird (Fokuslängenwert *f* im RTC6-20-Bit-Bereich [–524.288…+524.287]). Als Wert im 64-Bit-IEEE Gleitkommaformat. Zulässiger Wertebereich: siehe **load_z_table_20b**.
	`B`	Wie `A` (analog).
	`C`	Wie `A` (analog).
Rückgabe	`ErrorNo`	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen.
	0	Kein Fehler.
	1	`A` hat den maximal zulässigen Wert überschritten.
	2	`A` hat den minimal zulässigen Wert unterschritten.
	4	`B` hat den maximal zulässigen Wert überschritten.
	8	`B` hat den minimal zulässigen Wert unterschritten.
	16	`C` hat den maximal zulässigen Wert überschritten.
	32	`C` hat den minimal zulässigen Wert unterschritten.
	3	Datei-Open Fehler (leerer String, Datei nicht gefunden, etc.).
	12	Dateifehler (Check-Summe nicht ermittelbar, Datei korrupt).
Multi-Board-Befehlsname	Nicht als Multi-Board-Befehl verfügbar.	
Hinweise	• **write_abc_to_file_20b** ist auch ohne explizite Zugriffsberechtigung auf irgendeine RTC6-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError`, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133, werden von **write_abc_to_file_20b** weder erzeugt noch verändert. • Die ABC-Werte werden bei Fehlercode ≠ 0 oder ≠ 12 nicht ausgegeben. • `write_abc_to_file_20b( A, B, C )` ist gleichbedeutend mit `write_abc_to_file( A × 1/16, B, C × 16 )`.	
RTC4→RTC6	Neuer Befehl.	
RTC5→RTC6	Neuer Befehl.	
Versionsinfo	Verfügbar ab DLL 631, OUT 632.	
Verweise	**read_abc_from_file_20b**	

Kontrollbefehl	write_da_1
Funktion	Siehe **write_da_x**.
Aufruf	`write_da_1( Value )`
Parameter	`Value` 12-Bit-Ausgabewert für den Analog-Ausgang ANALOG OUT1. Als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert. `Value` = 0 entspricht einem Ausgabewert von 0 V. `Value` = 2^{12}–1 entspricht einem Ausgabewert von 10 V.
Multi-Board-Befehlsname	**n_write_da_1**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_da_x**

Verzögerter Kurzer Listenbefehl	write_da_1_list
Funktion	Siehe **write_da_x_list**.
Aufruf	`write_da_1_list( Value )`
Parameter	`Value` Wie **write_da_1**.
Multi-Board-Befehlsname	**n_write_da_1_list**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_da_x_list**

Kontrollbefehl	write_da_2
Funktion	Siehe **write_da_x**.
Aufruf	`write_da_2( Value )`
Parameter	`Value` 12-Bit-Ausgabewert für den Analog-Ausgang ANALOG OUT2. Als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert. `Value` = 0 entspricht einem Ausgabewert von 0 V. `Value` = 2^{12}–1 entspricht einem Ausgabewert von 10 V.
Multi-Board-Befehlsname	**n_write_da_2**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_da_x**

Verzögerter Kurzer Listenbefehl	write_da_2_list
Funktion	Siehe **write_da_x_list**.
Aufruf	`write_da_2_list( Value )`
Parameter	`Value` Wie **write_da_2**.
Multi-Board-Befehlsname	**n_write_da_2_list**
Hinweise	• –
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_da_x_list**

Kontrollbefehl	**write_da_x**
Funktion	Schreibt einen Ausgabewert auf einen der analogen Ausgänge der RTC6.
Aufruf	`write_da_x( x, Value )`
Parameter	`x` Nummer des Analog-Ausgangs. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [1, 2] (1: ANALOG OUT1, 2: ANALOG OUT2).
	`Value` 12-Bit-Ausgabewert für den gewählten Analog-Ausgang. Als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert. `Value` = 0 entspricht einem Ausgabewert von 0 V. `Value` = 2^{12}–1 entspricht einem Ausgabewert von 10 V.
Multi-Board-Befehlsname	**n_write_da_x**
Hinweise	• Siehe auch Kapitel 9.1.4 "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 308. • Der Ausgabebereich an den Analog-Ausgängen ist 0 V…10 V. • Für `x` < 1 oder `x` > 2 wird **write_da_x** ignoriert (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Alternativ zu **write_da_x** können **write_da_1** / **write_da_2** (ohne Parameter `x`) verwendet werden.
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus multipliziert die RTC6 `Value` jeweils mit 4.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_da_x_list**

Verzögerter Kurzer Listenbefehl	**write_da_x_list**
Funktion	Wie **write_da_x**, aber ein Listenbefehl.
Aufruf	`write_da_x_list( x, Value )`
Parameter	`x` Wie **write_da_x**.
	`Value` Wie **write_da_x**.
Multi-Board-Befehlsname	**n_write_da_x_list**
Hinweise	• Für `x` < 1 oder `x` > 2 wird **write_da_x_list** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`). • Ab Version DLL 602, OUT 602: Wenn der ANALOG OUT-Ausgang (z.B. für die Laserleistung) synchron zu den Laserschaltzeiten ausgeführt werden soll, verwenden Sie `set_laser_power( x-1, Value )`.
RTC4→RTC6	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_da_x**, **set_laser_power**, **write_port_list**

Kontrollbefehl	**write_hi_pos**
Funktion	Nur für einige abgekündigte Systeme. Schreibt die angegebenen Werte als ASC-Referenzwerte (= Home-In-Referenzpositionen, nicht zu verwechseln mit Home-In-Positionen) in den Flash-Speicher der RTC6-Karte.
Aufruf	`Result = write_hi_pos ( HeadNo, X1, X2, Y1, Y2 )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Anschluss für ersten Scan-Kopf. = 2: Anschluss für zweiten Scan-Kopf.
	`X1` X1-Referenzposition. In Bits. Als 32-Bit-Wert mit Vorzeichen.
	`X2` Wie `X1` (analog).
	`Y1` Wie `X1` (analog).
	`Y2` Wie `X1` (analog).
Rückgabe	Fehlercode. Als 32-Bit-Wert ohne Vorzeichen. 0 Kein Fehler. Bit #0 =1: Falsche `HeadNo`. Bit #1 =1: Falsche Sensorposition für `X1`. Bit #2 =1: Falsche Sensorposition für `X2`. Bit #3 =1: Falsche Sensorposition für `Y1`. Bit #4 =1: Falsche Sensorposition für `Y2`. Bit #5 =1: Ungültige ASC-Version. Bit #6 =1: Download fehlgeschlagen. Die Werte sind möglicherweise nicht gespeichert.
Multi-Board-Befehlsname	**n_write_hi_pos**
Hinweise	• **write_hi_pos** wird bei einem Wechsel des Scan-Kopfs von RTC6-Karte “A” zu RTC6-Karte “B” benutzt, um dessen Home-In-Referenzpositionen von der RTC6-Karte “A” nach RTC6-Karte “B” zu übertragen. • Lesen Sie mit `get_hi_pos( HeadNo )` die Home-In-Referenzpositionen von RTC6-Karte “A” aus (nur nach **init_rtc6_dll**, siehe **get_hi_pos**-Befehlsbeschreibung). • **write_hi_pos** wird auch ausgeführt, wenn der Scan-Kopf ausgeschaltet ist oder gar kein Scan-Kopf angeschlossen ist. • Auf der RTC6-Karte “B” muss zuletzt mit einem angeschlossenen Scan-Kopf, der mit ASC-Sensoren des Typs 1 ausgestattet ist, **auto_cal**`(Command = 4)` erfolgreich ausgeführt worden sein (oder **auto_cal**`(Command = 0)`, da dieser Befehl implizit **auto_cal**`(Command = 4)` ausführt). Zur Prüfung: `get_auto_cal(HeadNo)` muss 100 zurückliefern. Anderenfalls liefert **write_hi_pos** den Fehler Bit #5 = 1 zurück. Schließen Sie gegebenenfalls den Scan-Kopf an RTC6-Karte “B” an und führen Sie **auto_cal**`(HeadNo, 4)` aus, bevor Sie **write_hi_pos** ausführen. • **write_hi_pos** dauert mehrere hundert µs. Während dieser Zeit ist der 10 µs-Takt der RTC6-Karten unterbrochen.

Kontrollbefehl	write_hi_pos
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**get_hi_pos**, **get_auto_cal**, **auto_cal**

Kontrollbefehl	write_image_eth
Funktion	Standalone-Funktionalität: Liest die Daten für das automatische Booten aus einer Binärdatei auf dem PC und speichert sie in den NAND-Speicher.
Voraussetzung	RTC6-Software-Paket ≥ V1.7.0 und BIOS-ETH ≥ 26.
Aufruf	`Result = write_image_eth( Name )`
Parameter	`Name` Name der Binärdatei. Als Zeiger auf einen nullterminierten ANSI-String.
Rückgabe	`Result` Fehlercode. Wie `Result` von **read_image_eth**. Als 32-Bit-Wert ohne Vorzeichen.
Multi-Board-Befehlsname	**n_write_image_eth**
Hinweise	• **write_image_eth** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_BUSY`), wenn: – der BUSY-Listenausführungsstatus gesetzt ist – der INTERNAL-BUSY-Listenausführungsstatus gesetzt ist • Wenn `Name` nicht geöffnet werden kann, wird ein **get_last_error**-Returncode `RTC6_PARAM_ERROR` erzeugt. • Während **write_image_eth** ausgeführt wird, ist der 10 µs-Takt des DSP für bis zu 2 Minuten unterbrochen. • **write_image_eth** ist nur bei RTC6 Ethernet-Karten erlaubt. Anderenfalls wird ein **get_last_error**-Returncode `RTC6_TYPE_REJECTED` erzeugt. • Siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 618, OUT 618, RBF 623.
Verweise	**read_image_eth**, **store_program**

Kontrollbefehl	write_io_port
Funktion	Schreibt einen Wert auf den 16-Bit-Digital-Ausgang der EXTENSION 1-Stiftleiste.
Aufruf	`write_io_port( Value )`
Parameter	`Value` 16-Bit-Ausgabewert (DIGITAL OUT0…DIGITAL OUT15). Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
Multi-Board-Befehlsname	**n_write_io_port**
Hinweise	• Mit **set_io_cond_list** und **clear_io_cond_list** können individuelle Bits des 16-Bit-Digital-Ausgangs, abhängig vom aktuellen Zustand des 16-Bit-Digital-Eingangs, gesetzt oder zurückgesetzt werden. • Mit **write_io_port_mask** und **write_io_port_mask_list** können auch nur ausgewählte Bits am 16-Bit-Digital-Ausgang verändert werden. • Siehe auch Kapitel 9.1.1 "16-Bit-Digital-Ausgang", Seite 306.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_io_port_list**, **write_io_port_mask**, **write_io_port_mask_list**, **set_io_cond_list**, **clear_io_cond_list**, **get_io_status**

Verzögerter Kurzer Listenbefehl	write_io_port_list
Funktion	Wie **write_io_port**, aber ein Listenbefehl.
Aufruf	`write_io_port_list( Value )`
Parameter	`Value` Wie **write_io_port**.
Multi-Board-Befehlsname	**n_write_io_port_list**
Hinweise	• Siehe **write_io_port**. • Ab Version DLL 602, OUT 602: Wenn der 16-Bit-Digital-Ausgang (z.B. für die Laserleistung) synchron zu den Laserschaltzeiten ausgeführt werden soll, verwenden Sie `set_laser_power( 3, Value )`.
RTC4→RTC6	Unveränderte Funktionalität.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_io_port**, **set_laser_power**, **write_port_list**

Kontrollbefehl	write_io_port_mask
Funktion	Schreibt die über den Parameter `Mask` festgelegten Bits des Wertes `Value` auf den 16-Bit-Digital-Ausgang der EXTENSION 1-Stiftleiste.
Aufruf	`write_io_port_mask( Value, Mask )`
Parameter	`Value` 16-Bit-Ausgabewert (DIGITAL OUT0…DIGITAL OUT15). Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
	`Mask` 16-Bit-Maske (für DIGITAL OUT0…DIGITAL OUT15). Als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
Multi-Board-Befehlsname	**n_write_io_port_mask**
Hinweise	• Der Parameter `Mask` legt fest, *welche* Bits des 16-Bit-Digital-Ausgangs verändert werden, der Parameter `Value` *wie* sie verändert werden. Die in `Mask` nicht gesetzten Bits des 16-Bit-Digital-Ausgangs bleiben unverändert, werden also erneut so ausgegeben, wie sie zuletzt waren. • Für `Mask` = 0xFFFF wirkt **write_io_port_mask** wie **write_io_port**. • Siehe auch Kapitel 9.1.1 "16-Bit-Digital-Ausgang", Seite 306.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_io_port**, **write_io_port_mask_list**

Verzögerter Kurzer Listenbefehl	write_io_port_mask_list
Funktion	Wie **write_io_port_mask**, aber ein Listenbefehl.
Aufruf	`write_io_port_mask_list( Value, Mask )`
Parameter	`Value` Wie **write_io_port_mask**.
	`Mask` Wie **write_io_port_mask**.
Multi-Board-Befehlsname	**n_write_io_port_mask_list**
Hinweise	• Siehe **write_io_port_mask**. • **write_io_port_mask_list** kann nicht für die Laserleistungssteuerung verwendet werden, siehe **write_io_port_list**.
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Unveränderte Funktionalität.
Versionsinfo	Verfügbar ab DLL 600, OUT 600, RBF 600.
Verweise	**write_io_port_mask**, **write_io_port_list**, **set_laser_power**

Unverzögerter Kurzer Listenbefehl	**write_port_list**
Funktion	Schreibt einen Wert auf einen Digital- oder Analog-Ausgang.
Aufruf	`write_port_list( Port, Value, NoDelay )`
Parameter	`Port` Ausgang. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: ANALOG OUT1-Ausgang. Siehe auch Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77. = 1: ANALOG OUT2-Ausgang. Siehe auch Abschnitt "12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2", Seite 77. = 2: 8-Bit-Digital-Ausgang (EXTENSION 2-Stiftleiste). Siehe auch Abschnitt "8-Bit-Digital-Ausgang", Seite 84. = 3: 16-Bit-Digital-Ausgang (EXTENSION 1-Stiftleiste). Siehe auch Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81. = 4: 2-Bit-Digital-Ausgang (LASER-Buchse). Siehe auch Abschnitt "2-Bit-Digital-Ausgang", Seite 77. > 4: **write_port_list** wird nicht ausgeführt (**get_last_error**-Returncode `RTC6_PARAM_ERROR`).
	`Value` Ausgabewert. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: Für `Port` = 0: 12-Bit-Werte [0…4.095]. Für `Port` = 1: 12-Bit-Werte [0…4.095]. Für `Port` = 2: 8-Bit-Werte [0…255]. Für `Port` = 3: 16-Bit-Werte [0…65.535]. Für `Port` = 4: 2-Bit-Werte [0…3]. Überlaufende Werte werden auf die Randwerte geclippt. **get_last_error**-Returncode `RTC6_PARAM_ERROR`.
	`NoDelay` Ausgabeverzögerung. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: `Value` wird erst nach Ablauf der `PreviewTime` ausgegeben. > 0: `Value` wird sofort ausgegeben.
Multi-Board-Befehlsname	**n_write_port_list**

Unverzögerter Kurzer Listenbefehl	**write_port_list**
Hinweise	• `NoDelay` ist nur relevant, wenn mit **set_scanahead_params** die SCANahead-Funktionalität auf der RTC6-Karte aktiviert wurde. • **write_port_list**(`NoDelay` = 0) verhält sich (je nach `Port`) wie: – **write_da_x_list** – **write_8bit_port_list** – **write_io_port_list** – **set_laser_pin_out_list**
RTC4→RTC6	Neuer Befehl.
RTC5→RTC6	Neuer Befehl.
Versionsinfo	Verfügbar ab DLL 634, OUT 635, RBF 634.
Verweise	**set_scanahead_params**

10.3 RTC6-Befehle in anderen Publikationen

Im Header (z.B. RTC6impl.hpp) sind einige RTC6-Befehle veröffentlicht, die nicht Teil dieses Handbuchs sind. Diese sind für einen begrenzten Benutzerkreis oder spezifische Hardware gedacht.

RTC6-Befehl	Entsprechender Multi-Board-Befehl	Hinweise
check_enduring_wobbel	n_check_enduring_wobbel	Reserviert.
enduring_wobbel	n_enduring_wobbel	Reserviert.
eth_error_dump	n_eth_error_dump	Reserviert.
get_home_4_return_time	n_get_home_4_return_time	Reserviert.
get_home_position_4	n_get_home_position_4	Reserviert.
home_position_4	n_home_position_4	Reserviert.
jump_abs_drill	n_jump_abs_drill	Seite 521
jump_abs_drill_2	n_jump_abs_drill_2	Seite 521
jump_abs_drill_3	n_jump_abs_drill_3	Reserviert.
jump_rel_drill	n_jump_rel_drill	Seite 524
jump_rel_drill_2	n_jump_rel_drill_2	Seite 524
jump_rel_drill_3	n_jump_rel_drill_3	Reserviert.
list_jump_rel_ctrl	n_list_jump_rel_ctrl	Reserviert.
load_oct_table_no	n_load_oct_table_no	Reserviert.
load_zoom_correction_file	n_load_zoom_correction_file	Seite 579
micro_vector_set_position	n_micro_vector_set_position	Reserviert.

RTC6-Befehl (Forts.)	Entsprechender Multi-Board-Befehl (Forts.)	Hinweise (Forts.)
multi_axis_flags_list	n_multi_axis_flags_list	Reserviert.
multi_axis_get_config_ctrl	n_multi_axis_get_config_ctrl	Reserviert.
multi_axis_get_error_ctrl	n_multi_axis_get_error_ctrl	Reserviert.
multi_axis_get_values_ctrl	n_multi_axis_get_values_ctrl	Reserviert.
multi_axis_init_ctrl	n_multi_axis_init_ctrl	Reserviert.
multi_axis_micro_vector_scanner_list	n_multi_axis_micro_vector_scanner_list	Reserviert.
multi_axis_micro_vector_stage_list	n_multi_axis_micro_vector_stage_list	Reserviert.
multi_axis_set_config_ctrl	n_multi_axis_set_config_ctrl	Reserviert.
set_enduring_wobbel	n_set_enduring_wobbel	Reserviert.
set_enduring_wobbel_2	n_set_enduring_wobbel_2	Reserviert.
set_enduring_wobbel_3	n_set_enduring_wobbel_3	Reserviert.
set_enduring_wobbel_4	n_set_enduring_wobbel_4	Reserviert.
set_enduring_wobbel_ctrl	n_set_enduring_wobbel_ctrl	Reserviert.
set_home_4_return_time	n_set_home_4_return_time	Reserviert.
set_laser_pulse_form	n_set_laser_pulse_form	Reserviert.
set_scanahead_line_params_ex	n_set_scanahead_line_params_ex	Reserviert.
set_scanahead_line_params_ex_list	n_set_scanahead_line_params_ex_list	Reserviert.
set_zoom	n_set_zoom	Seite 872
set_zoom_list	n_set_zoom_list	Seite 872

10.4 Nicht mehr unterstützte RTC4-Befehle

RTC4-Kontrollbefehl	dsp_start
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	**dsp_start** ist nicht mehr erforderlich. Der DSP startet automatisch nach dem Laden der Programmdatei mit **load_program_file**.

RTC4-Kontrollbefehl	get_xy_pos
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	Ersatz: **get_value**, **get_values**

RTC4-Kontrollbefehl	get_xyz_pos
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	Ersatz: **get_value**, **get_values**

RTC4-Kontrollbefehl	read_pixel_ad
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	Es gibt keinen äquivalenten RTC6-Befehl.

RTC4-Kontrollbefehl	rtc4_count_cards
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	**rtc6_count_cards**

RTC4-Kontrollbefehl	select_list
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	**select_list**(0) kann durch **set_extstartpos**(0), **select_list**(1) durch **set_extstartpos**(Mem1) ersetzt werden. Mem1 ist dabei die Speichergröße von “Liste 1” (oder die absolute Anfangsadresse von “Liste 2”). Mem1 ist nach einer Initialisierung (mit **load_program_file**) 4000, sonst so wie mit **config_list** eingestellt. Mem1 kann (z.B. nach einem Besitzerwechsel der Karte) mit **set_start_list_2** und **get_input_pointer** ermittelt werden.

RTC4-Kontrollbefehl	**set_list_mode**
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	Siehe Kapitel 6.5.4 "RTC4-Ringpuffer-Modus", Seite 124.

RTC4-Kontrollbefehl	**set_piso_control**
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	**set_piso_control** ist bei der RTC6 nicht mehr nötig. Bei einer Datenübertragung gemäß SL2-100-Protokoll ist die bidirektionale Kommunikation zwischen Scan-System und RTC5/RTC6 unabhängig von der Länge des Datenkabels. Bei einer Datenübertragung gemäß XY2-100-Protokoll muss die Synchronisation der Kommunikation zwar weiterhin an die Länge des Datenkabels angepasst werden. Die Anpassung erfolgt nun über die Jumper am XY2-100-Konverter (Zubehör).

RTC4-Kontrollbefehl	**set_softstart_level**
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	–

RTC4-Kontrollbefehl	**set_softstart_mode**
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	–

RTC4-Kontrollbefehl	**set_wobbel_xy**
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	**set_wobbel**, **set_wobbel_mode**

RTC4-Kontrollbefehl	z_out
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	Um den z-Wert in einem 3-Achsen-Scan-System einzustellen, kann **set_defocus** bei der RTC6 mit freigeschalteter Option "3D" verwendet werden. Nach **load_z_table**(0.0, 1.0, 0.0) hat **set_defocus**(Z) die gleiche Wirkung wie **z_out**(Z) (siehe auch **set_offset_xyz**).

RTC4-Kontrollbefehl	z_out_list
Unterstützung	Nicht von der RTC6 unterstützt.
RTC4→RTC6	Nach **load_z_table**(0.0, 1.0, 0.0) hat **set_defocus_list**(Z) die gleiche Wirkung wie **z_out_list**(Z) (siehe auch **set_offset_xyz_list**).

10.5 Nicht mehr unterstützte RTC5-Befehle

RTC5-Kontrollbefehl	free_rtc5_dll
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	free_rtc6_dll

RTC5-Kontrollbefehl	init_rtc5_dll
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	init_rtc6_dll

RTC5-Kontrollbefehl	rtc5_count_cards
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	rtc6_count_cards

RTC5-Kontrollbefehl	set_softstart_level
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	–

RTC5-Listenbefehl	set_softstart_level_list
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	–

RTC5-Kontrollbefehl	set_softstart_mode
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	–

RTC5-Listenbefehl	set_softstart_mode_list
Unterstützung	Nicht von der RTC6 unterstützt.
Ersatz-Befehl	–

11 Demo-Programme

- Das RTC6-Software-Paket enthält derzeit keine Demo-Programme.

12 Fehlersuche

Problem	Behebung
PC bootet nicht	Schalten Sie den PC ab und prüfen Sie die folgenden Aspekte: • Überprüfen Sie, ob die RTC6-Karte richtig im PCIe-Slot steckt. Beachten Sie die Hinweise im Handbuch Ihres PCs. • Überprüfen Sie, ob bei der Montage metallische Fremdkörper ins Gehäuse des PCs gelangt sind. • Überprüfen Sie, ob Kabel oder Steckverbinder abgetrennt wurden.
RTC6-Karte lässt sich nicht ansprechen	• Überprüfen Sie, ob der RTC6-Kartentreiber richtig installiert ist. Siehe Kapitel 5.4 "RTC6-Software installieren", Seite 93. • Überprüfen Sie mit dem mitgelieferten HPGL-Converter-Programm, ob sich die Karte ordnungsgemäß ansprechen lässt. Wenn nein: Überprüfen Sie den Kabeltyp und die Kabellänge. Falls das Scan-System über einen XY2-100-Konverter (Zubehör) angesteuert wird, überprüfen Sie, ob die Lötjumper des XY2-100-Konverter (Zubehör) passend zur Kabellänge eingestellt sind. Siehe Abbildung 14. Wenn ja: Überprüfen Sie, ob die Funktionen der RTC6-DLL richtig in das Anwenderprogramm importiert wurden. Siehe Kapitel 6.2.2 "Befehle importieren", Seite 98.
Anwenderprogramm stürzt ab	• Überprüfen Sie, ob die Treiber richtig installiert sind. Siehe Kapitel 5.4 "RTC6-Software installieren", Seite 93. • Überprüfen Sie, ob die RTC6-Karte fehlerfrei im Anwenderprogramm initialisiert wurde. • Überprüfen Sie die DLL-Importdeklarationen im Anwenderprogramm. Siehe Kapitel 6.2.2 "Befehle importieren", Seite 98.
Fehler in der Scan-Kopf-Steuerung	• Überprüfen Sie, ob der Scan-Kopf über das Datenkabel richtig mit der RTC6-Karte verbunden ist. Vergewissern Sie sich, ob die empfohlenen Spezifikationen für das Datenkabel erfüllt sind. Siehe Kapitel 4.5.3 "Datenkabel (Zubehör)", Seite 73. • Überprüfen Sie, ob die Stromversorgung des Scan-Kopfs korrekt ist. Beachten Sie dazu die Hinweise in der Bedienungsanleitung des Scan-Kopfs. • Überprüfen Sie das Anwenderprogramm.
Fehler in der Laser-Steuerung	• Überprüfen Sie die Funktionsfähigkeit der Schnittstelle zwischen Laser und RTC6-Karte.
Unregelmäßigkeiten in den Markierungsergebnissen	• Überprüfen Sie, ob die Laser- und Scanner-Delays richtig definiert wurden. Siehe Kapitel 7.2.3 "Hinweise zur Optimierung der Delays", Seite 158.
Der Laser ist während Sprung-Befehlen nicht ausgeschaltet	• Überprüfen Sie, ob die Laser- und Scanner-Delays richtig definiert wurden. Siehe Kapitel 7.2 "Delayeinstellungen – Scan-Kopf-Steuerung und Laser-Steuerung abstimmen", Seite 148.

Für eine Fehlersuche sind außerdem die folgenden Befehle hilfreich:

- Mit **get_error** und **get_last_error** kann (fast) jeder Befehl auf eine ordnungsgemäße Ausführung überprüft werden, siehe Kapitel 6.8 "Fehlerbehandlung (Error-Handling)", Seite 133.
- mit **set_verify** kann sichergestellt werden, dass alle Downloads (Befehle, Tabellen) fehlerfrei ausgeführt wurden, siehe Abschnitt "Download-Überprüfung", Seite 134.
- mit **get_value** oder **get_values** können Sie einzelne vom Scan-System zurückgegebene Werte auslesen oder mit **set_trigger[*]** und **get_waveform** eine ganze Serie von Rückgabewerten aufzeichnen, siehe Abschnitt "Statusüberwachung und Diagnose", Seite 191 (bei iDRIVE-Scan-Systemen siehe auch Kapitel 8.1 "iDRIVE-Funktionen", Seite 224).
- mit **set_wait** und **get_wait_status** können Sie überprüfen, ob Programmverzweigungen (bedingte Befehle) wie geplant ausgeführt wurden.
- mit **get_overrun** können Sie überprüfen, ob Laufzeitüberschreitungen des 10 μs-Taktes stattgefunden haben, siehe Abschnitt "Takt-Überläufe", Seite 190.
- mit **get_status** oder **get_out_pointer** können Sie überprüfen, bei welcher Befehlsnummer sich die Programmausführung gerade befindet (z.B. bei "Endlos-Schleifen" durch Zirkelschluss im Programmablauf).
- Mit **get_startstop_info** können Informationen über die Lasersteuersignale, sowie aufgetretene Übertragungsfehler von und zum angeschlossenen Scan-System ausgelesen werden.

Falls einzelne Port-Ausgaben wirkungslos bleiben, überprüfen Sie das Anwenderprogramm daraufhin, ob ein und derselbe Port unmittelbar nacheinander angesprochen wird. Da hierzu meist sogenannte kurze Listenbefehle (siehe Abschnitt "Normaler Listenbefehl, Kurzer Listenbefehl, Variabler Listenbefehl und Multipler Listenbefehl", Seite 339) verwendet werden, überschreibt ein Befehl gegebenenfalls den Ausgabewert des anderen noch innerhalb desselben 10 μs-Taktes. Trennen Sie in dieser Situation die beiden kurzen Befehle durch einen (normalen) Listenbefehl, z.B. **list_nop** oder **list_continue**.

Wenn die Ausführungsdauer (gemessen mit **save_and_restart_timer** und **get_time**) nicht mit Ihrer Rechnung übereinstimmt, überprüfen Sie ihr Anwenderprogramm, ob sogenannte kurze Listenbefehle enthalten sind. Diese benötigen in der Regel keinen eigenen Takt zur Ausführung. Eventuell können auch zusätzliche Scanner-Delays automatisch eingefügt worden sein, um ein (fehlerhaftes) Überkreuzen von LaserOn und LaserOff zu unterbinden, siehe Abschnitt "Automatische Delay-Anpassung", Seite 158.

Sollten Sie die Fehler nicht wie angegeben beheben können, wenden Sie sich an SCANLAB.

13 Kundendienst

13.1 Wartung und Reparaturen

Alle Wartungs- und Reparaturarbeiten an der RTC6-Karte sollten ausschließlich von SCANLAB durchgeführt werden. Bei Manipulationen an der RTC6-Karte erlöschen die Gewährleistungsansprüche.

13.2 Gewährleistung

SCANLAB gewährleistet, dass das Produkt keine Herstellungs- oder Materialfehler aufweist. Der Gewährleistungszeitraum beträgt 12 Monate ab Auslieferung. Reparaturen aufgrund von Gewährleistungsansprüchen erfolgen grundsätzlich bei SCANLAB.

Der Gewährleistungsumfang ist auf die Reparatur oder den Ersatz des von SCANLAB gelieferten Produktes beschränkt.

SCANLAB ist für die Rücklieferung von Gewährleistungsreparaturen verantwortlich, für die Lieferung zu SCANLAB ist der Kunde verantwortlich.

SCANLAB übernimmt keine Gewährleistung,

- wenn das Produkt durch falsche Handhabung oder falschen Einsatz beschädigt wurde,
- für Reparaturen an der RTC6-Karte, die nicht von SCANLAB durchgeführt worden sind,
- für Schäden, die durch eine unsachgemäße Verpackung bei einer Rücklieferung an SCANLAB entstehen.
- bei Manipulationen an der RTC6-Karte,
- für Folgeschäden.

13.3 Kontaktmöglichkeiten zu SCANLAB

SCANLAB GmbH
Siemensstr. 2a
82178 Puchheim
Deutschland

Tel. +49 (89) 800 746-0
Fax +49 (89) 800 746-199

info@scanlab.de
www.scanlab.de

13.4 Entsorgung

Die RTC6-Karte wird von SCANLAB gegen eine Gebühr zurückgenommen und fachgerecht entsorgt.

14 Rechtliches

14.1 EU-Konformitätserklärung – RTC6 PCIe-Karte

EU-Konformitätserklärung

für das Produkt:

RTC4 (PCI); RTC4 PCI-Express; RTC5 (PCI); RTC5 PCI-Express; RTC6 PCI-Express

Der Hersteller

SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Deutschland

erklärt, dass das genannte Produkt die einschlägigen Harmonisierungsrechtsvorschriften der Union erfüllt:

- **2014/30/EU -** Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Harmonisierung der Rechtsvorschriften der Mitgliedstaaten über die elektromagnetische Verträglichkeit (EMV-Richtlinie).
- **2011/65/EU -** Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Beschränkung der Verwendung bestimmter gefährlicher Stoffe in Elektro- und Elektronikgeräten (RoHS-Richtlinie).

Folgende harmonisierte Normen wurden angewandt:

- **EN IEC 61000-6-2:2005** – Elektromagnetische Verträglichkeit (EMV) - Teil 6-2: Fachgrundnormen - Störfestigkeit für Industriebereiche
- **DIN EN 55011:2018-05** - Industrielle, wissenschaftliche und medizinische Geräte - Funkstörungen - Grenzwerte und Messverfahren
- **DIN EN IEC 63000:2019-05** - Technische Dokumentation zur Beurteilung von Elektro- und Elektronikgeräten hinsichtlich der Beschränkung gefährlicher Stoffe

Die alleinige Verantwortung für die Ausstellung dieser Konformitätserklärung trägt der Hersteller.

Dieses Dokument zur Kundeninformation wurde elektronisch ausgestellt und ist daher ohne Unterschrift gültig. Eine unterschriebene Ausführung ist Bestandteil der SCANLAB-internen technischen Dokumentation.

EU Declaration of Conformity

for the product:

RTC4 (PCI); RTC4 PCI-Express; RTC5 (PCI); RTC5 PCI-Express; RTC6 PCI-Express

The manufacturer

SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Germany

declares that the product mentioned above complies with the relevant Union harmonization legislation:

- **2014/30/EU -** Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the harmonisation of the laws of the Member States relating to electromagnetic compatibility (EMC Directive)
- **2011/65/EU** - Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the restriction of the use of certain hazardous substances in electrical and electronic equipment (RoHS Directive).

The following harmonized standards have been applied:

- **EN IEC 61000-6-2:2005** - Electromagnetic compatibility (EMC) - Part 6-2: Generic standards - Immunity for industrial environments
- **DIN EN 55011:2018-05** - Industrial, scientific and medical equipment – Radio-frequency disturbance characteristics - Limits and methods of measurement
- **DIN EN IEC 63000:2019-05** - Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances

This declaration of conformity is issued under the sole responsibility of the manufacturer.

This document, used for customer information, was issued electronically and is therefore valid without a signature. A signed version is part of the SCANLAB internal technical documentation.

SCANLAB GmbH
Puchheim
2021-07-19

© SCANLAB GmbH – 2021/07

14.2 Konformität mit FCC-Bestimmungen

Die RTC6 PCIe-Karte wurde getestet und entspricht den Grenzwerten für Digitalgeräte der Klasse A, gemäß Teil 15 der FCC-Bestimmungen.

Diese Grenzwerte sollen einen angemessenen Schutz gegen schädliche Störungen bieten, wenn die RTC6 PCIe-Karte in einer kommerziellen Umgebung betrieben wird. Die RTC6 PCIe-Karte erzeugt, verwendet und emittiert möglicherweise Hochfrequenzenergie und kann, wenn nicht in Übereinstimmung mit dieser Bedienungsanleitung installiert und verwendet, Störungen im Funkverkehr verursachen. Der Betrieb der RTC6 PCIe-Karte in einem Wohngebiet wird wahrscheinlich schädliche Störungen verursachen; in diesem Fall ist der Benutzer verpflichtet, die Störungen auf eigene Kosten zu beseitigen.

15 Technische Spezifikationen – RTC6 PCIe-Karte

Systemanforderungen

Windows-PC mit freiem PCI-Express-Steckplatz

Betriebssystem	Mit RTC6-Software-Paket ≥ V1.3.0: Microsoft Windows 10, 8, 7 als 32-Bit oder 64-Bit Version. *Windows XP und Windows Vista werden nicht unterstützt!*

Abmessungen

Länge	160 mm
Breite	104,5 mm

Schnittstelle zum PC

PCI-Express x-1, Version 1.0

Scan-System Ansteuerung

Zahl der RTC6-Listenspeicher-Bereiche	Bis zu 3, konfigurierbar
Gesamtkapazität des RTC6-Listenspeichers	8.388.608 Listenplätze
Ausgabeintervall der Mikroschritte	10 μs
Maximaler Wertebereich der Bildfeld-Koordinaten	–524.288…+524.287 (20 Bit mit Vorzeichen)
Virtuelles Bildfeld (z.B. für "Processing on the fly")	–268.435.456… +268.435.455 (28 Bit mit Vorzeichen = 29 Bit)[(a)]

(a) Mit RTC6-Software-Paket ≥ 1.4.0.

Schnittstellen

SCANHEAD-Buchse

Anschluss	9-polige Sub-D-Buchse (weiblich)

2. SCANHEAD-Stiftleiste

Anschluss	10-polige Stiftleiste. • xy-Ausgabewerte nur bei freigeschalteter Option "Zweite-Scan-Kopf-Steuerung" • z-Ausgabewerte nur bei freigeschalteter Option "3D"
Signalübertragung	SL2-100-Protokoll. Über XY2-100-Konverter (Zubehör): XY2-100-Protokoll.

LASER-Buchse

Anschluss	15-polige Sub-D-Buchse (weiblich)
Lasersteuersignale	LASER1, LASER2, LASERON
• TTL-Pegel	5 V, HIGH-aktiv oder LOW-aktiv (programmierbar)
• Max. Last	20 mA
• Bezugspunkt	GND[a] GND2 bei Option "DC/DC-Wandler"

(a) RTC6 PCIe-Karten: GND ist die PC-Masse.
RTC6 Ethernet-Karten: GND ist die Masse am POWER-Stecker.

Ausgang	12-Bit-Analog-Ausgang 1 ANALOG OUT1
• Ausgangsspannungsbereich	0 V…10 V
• Auflösung	12 Bit
• Max. Last	5 mA
• Bezugspunkt	GND[a]

Ausgang	12-Bit-Analog-Ausgang 2 ANALOG OUT2
• Ausgangsspannungsbereich	0 V…10 V
• Auflösung	12 Bit
• Max. Last	5 mA
• Bezugspunkt	GND[a]

Eingang	2-Bit-Digital-Eingang
• LOW-Pegel	< 0,6 V
• HIGH-Pegel	> 2,3 V
• Max. Eingangs-spannungsbereich	–0,5 V…+5,5 V
• Eingangswiderstand (Pull-up)	> 4,7 kΩ
• Bezugspunkt	GND[a]

Ausgang	2-Bit-Digital-Ausgang gepuffert
• LOW-Pegel	< 0,55 V
• HIGH-Pegel	> 3,8 V
• Max. Last	20 mA
• Bezugspunkt	GND[a]

Eingänge für externe Steuersignale	TTL-Pegel LOW-aktiv /START und /STOP sind intern über Pull-up-Widerstände (4,7 kΩ) mit +3,3 V verbunden, siehe Abbildung 18.
• /START	Flanken-sensitiv (HIGH → LOW Übergang) HIGH-Pegel > 2,3 V LOW-Pegel < 0,6 V
• /STOP	Pegel-sensitiv HIGH-Pegel > 2,3 V LOW-Pegel < 0,6 V
• Bezugspunkt	GND[a]

Sonstige Signale	
• BUSY OUT	5 V, TTL HIGH-aktiv max. 10 mA BUSY OUT, Seite 974, BUSY OUT, Seite 975, BUSY OUT, Seite 976 sind identisch
• +5 V	max. 100 mA
• Bezugspunkt	GND[a]

EXTENSION 1-Stiftleiste

Anschluss	40-polige Stiftleiste
	Ausgangssignal-Pegel konfigurierbar über Jumper
Ausgang	16-Bit-Digital-Ausgang
	gepuffert
• LOW-Pegel	< 0,4 V
• HIGH-Pegel	> 2,0 V (3,3 V oder 5 V)
• Max. Last	±8 mA
• Bezugspunkt	GND[a]
LATCH OUT	3,3 V oder 5 V, TTL HIGH-aktiv, 5 µs-Puls, max. 10 mA
SYNC OUT	3,3 V oder 5 V, TTL HIGH-aktiv, Rechtecksignal (5 µs-Puls, 10 µs-Periode), max. 10 mA
Eingang	16-Bit-Digital-Eingang
• LOW-Pegel	< 0,5 V
• HIGH-Pegel	> 2,6 V…24 V
• Max. Eingangs-spannungsbereich	–0,5 V…+26 V
• Eingangswiderstand	> 10 kΩ
• Bezugspunkt	GND[a]
Sonstige Signale	
• BUSY OUT	3,3 V oder 5 V, TTL HIGH-aktiv, max. 10 mA BUSY OUT, Seite 974, BUSY OUT, Seite 975, BUSY OUT, Seite 976 sind identisch
• VCC_OUT	3,3 V oder 5 V, max. 100 mA
• +5 V	max. 100 mA
• Bezugspunkt	GND[a]

EXTENSION 2-Stiftleiste

Anschluss	26-polige Stiftleiste
	konfigurierbar über Jumper
Ausgang	8-Bit-Digital-Ausgang
	gepuffert
• LOW-Pegel	< 0,4 V
• HIGH-Pegel	> 2,0 V
• Max. Last	±8 mA
• Bezugspunkt	GND[a]
• LATCH-Signal	5 V, TTL HIGH-aktiv, 5 µs-Puls, max. 10 mA
Lasersteuersignale	LASERON, LASER1, LASER2 (siehe LASER-Buchse, Seite 974)
Sonstige Signale	
• +5 V	max. 100 mA
• Bezugspunkt	GND[a]

MARKING ON THE FLY-Stiftleiste

Anschluss	16-polige Stiftleiste
2 Encodereingänge für Inkremental-Encoder	ENCODER X(1±,2±) und ENCODER Y(1±,2±), jeweils für 1 Paar von standardisierten differentiellen Eingangssignalen (RS-422). HIGH-Pegel ≥ 2.0 V LOW-Pegel ≤ 0.8 V f ≤ 4 MHz
Ausgang	12-Bit-Analog-Ausgang 2 ANALOG OUT2 (siehe LASER-Buchse, Seite 974)
Eingänge für externe Steuersignale	TTL-Pegel LOW-aktiv /START2 und /STOP2 sind intern über Pull-up-Widerstände (4,7 kΩ) mit +3,3 V verbunden.
• /START2	Flanken-sensitiv (HIGH → LOW Übergang) HIGH-Pegel > 2,3 V LOW-Pegel < 0,6 V
• /STOP2	Pegel-sensitiv HIGH-Pegel > 2,3 V LOW-Pegel < 0,6 V
• Bezugspunkt	GND[(a)]
Sonstige Signale	
• BUSY OUT	5 V, TTL HIGH-aktiv max. 10 mA BUSY OUT, Seite 974, BUSY OUT, Seite 975, BUSY OUT, Seite 976 sind identisch
• +5 V	max. 100 mA
• Bezugspunkt	GND[(a)]

RS232-Stiftleiste

Anschluss	10-polige Stiftleiste
Eingang	RxD
• Spannungsbereich	max. –25 V…+25 V
Ausgang	TxD
• Spannungsbereich	max. –13 V…+13 V
Bezugspunkt	GND[(a)]
Baud-Rate	300…115.200

McBSP/ANALOG-Stiftleiste

Anschluss	10-polige Stiftleiste
McBSP-Schnittstelle	Siehe Abschnitt "McBSP-Schnittstelle", Seite 87.
• Sender-Signalpegel	3,3 V TTL
• Empfänger-Signalpegel	3,3 V oder 5 V TTL
• McBSP-Mode	Single Phase Frame Single Element per Frame 32 Bits per Element Daten-Delay `XDelay` Bits Daten-Delay `RDelay` Bits
• Bezugspunkt	GND[a]
SPI-Schnittstellenfunktionalität	Nein
Analoge Eingänge	ANALOG IN0, ANALOG IN1
• Eingangsspannungsbereich	0 V…10 V
• Eingangsimpedanz	> 5 kΩ
• ADC-Auflösung	12 Bit
• Bezugspunkt	GND[a]

STEPPER MOTOR-Stiftleiste

Anschluss	10-polige Stiftleiste
Signale zum Ansteuern von 1…2 Schrittmotoren:	
• ENABLE-Ausgang	5 V, TTL
• DIRECTION-Ausgang	5 V, TTL
• CLOCK-Ausgang	5 V, TTL HIGH-aktiv, 5 µs-Puls
• SWITCH-Eingang	TTL LOW-aktiv Intern sind SWITCH1-Eingang und SWITCH2-Eingang über 10 kΩ-Pull-up-Widerstände mit +3,3 V verbunden.
• max. Last	25 mA
• Bezugspunkt	GND[a]

16 Anhang A: RTC6 Ethernet-Karte

16.1 Produktübersicht

16.1.1 RTC6 Ethernet-Karte und RTC6 PCIe-Karte – Verwendung und Vergleich

Bei RTC6 Ethernet-Karten und RTC6 PCIe-Karten sind die *grundsätzlichen Funktionen* gleich:

- Echtzeitsteuerung von Laser und Scan-System.
- Übertragung von direkten Befehlen (Kontrollbefehle) vom PC auf die Karte. Diese Befehle werden sofort ausgeführt.
- Übertragung von Listenbefehlen vom PC in den RTC6-Listenspeicher. Die Befehle werden erst nach dem Start der Liste ausgeführt.
- Die Listenausführung erfolgt in Echtzeit.
- Die Berechnung der Sollpositionen erfolgt alle 10 μs.
- Es erfolgt eine Bildfeld-Korrektur der Sollpositionen.
- Die Schnittstelle zum Scan-System benutzt das 20-Bit SL2-100-Protokoll.
- Zur Softwareentwicklung ist in der `RTC6DLL.dll`/`RTC6DLLx64.dll` der komplette Befehlssatz für RTC6 Ethernet-Karten und RTC6 PCIe-Karten enthalten.

Bei RTC6 Ethernet-Karten ist folgendes *anders*:

- Ethernet-Schnittstelle(1) anstatt PCIe-Bus
  - Die Kommunikation mit dem PC erfolgt über das TCP/IP- und UDP-Protokoll.
  - RTC6 Ethernet-Karten funktionieren *ohne* RTC6-Kartentreiber.
  - Die Karte muss nicht in einen PC gesteckt werden.
  - Echtzeit-Uhr, siehe Kapitel 16.2.15 "Echtzeit-Uhr", Seite 1000.
  - Mit RTC6-Software-Paket ≥ V1.7.0 können RTC6 Ethernet-Karten auch ohne PC im Standalone-Betrieb verwendet werden, siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.

(1) Unterstützt Gigabit Ethernet nach 1000BASE-T.

16.1.2 Systemanforderungen

Vorsicht!

- Die RTC6 Ethernet-Karte darf keinesfalls ohne Abschirmung betrieben werden (z.B. außerhalb eines metallischen Gehäuses), um Störeinkopplungen und Störabstrahlungen zu vermeiden.
- Die RTC6 Ethernet-Karte ist ausschließlich für den gewerblichen Gebrauch bestimmt. Sie ist zum Einbau in eine Maschine (in der Regel in ein Lasersystem) vorgesehen und entspricht *nicht* allen Anforderungen an ein gebrauchsfertiges Endprodukt. Nehmen Sie die RTC6 Ethernet-Karte erst dann in Betrieb, wenn sie in eine Maschine eingebaut ist, die den Bestimmungen aller dafür anzuwendenden Richtlinien und Normen (des betreffenden Landes) entspricht. Sie ist **nicht** für den Einsatz als Kinderspielzeug, im Haushalt oder unter widrigen Umgebungsbedingungen (z.B. im Freien) geeignet. Entsprechende Vorsichtsmaßnahmen zur Vermeidung solcher Fehlanwendungen sind von Benutzern zu treffen.
- Die Installation und Inbetriebnahme darf nur durch Personen erfolgen, die über eine ausreichende Qualifikation verfügen, u.a. über Kenntnisse im sicheren Umgang mit elektrischen Geräten. Führen Sie Installations- und Wartungsarbeiten nur bei ausgeschalteter Versorgungsspannung und ausgeschaltetem Laser aus.
- Die RTC6 Ethernet-Karte ist eine Einrichtung der Klasse A. Sie kann im Wohnbereich Funkstörungen verursachen; in diesem Fall kann vom Betreiber verlangt werden, angemessene Maßnahmen durchzuführen.

Hardware

- Abgeschirmtes Gehäuse mit:
  - Möglichkeit zur mechanischen Befestigung incl. Befestigungsmaterial
  - entsprechender thermischer Anbindung (Kühlung) der Karte
  - Stromzufuhr
  - Ethernet-Kabel
  - Netzteil gemäß Spezifikation
  - Kabel mit Stecker für die Spannungsversorgung der RTC6 Ethernet-Karte
  - Flachbandkabel mit Stecker zu den Peripheriegeräten (Laser, Scan-Kopf, Erweiterungen)
- Windows-PC
- Gigabit-Ethernet (bevorzugt), 10/100 Mbit/s-Ethernet

Software

- TCP/IP Protokoll IPv4
- UDP-Protokoll
- Verwendete Ports (Voreinstellung)
  - 63749 (UDP)
  - 63750 (UDP, TCP)

Für die Entwicklung von Anwenderprogrammen:

- DLL/Importdeklarationen (im RTC6-Software-Paket enthalten)

16.1.3 Optionen

Wie bei RTC6 PCIe-Karten, siehe Kapitel 2.6 "Optionen", Seite 39.
Außerdem gibt es nur bei RTC6 Ethernet-Karten:

- **Option "LDSA"**
  – Unterstützung von Standalone-Funktionalität durch laserDESK
  Erlaubt, dass Standalone-Funktionalitäts-bezogene Leistungsmerkmale in laserDESK genutzt werden können. Beispiel: Booten einer RTC6 Ethernet-Karte mit einem unter laserDESK erstellten Boot-Image. Beachten Sie, dass mit laserDESK erstellte Boot-Images *nicht* mit RTC6 Ethernet-Karten ohne Option "LDSA" verwendet werden können. Weitere Informationen siehe laserDESK Online-Hilfe.
  Siehe auch **get_rtc_version**, Bit #11.

16.1.4 Kennzeichnung

Wie bei RTC6 PCIe-Karten, siehe Kapitel 2.1 "Kennzeichnung", Seite 32.

16.1.5 Typenbezeichnung

Wie bei RTC6 PCIe-Karten, siehe Kapitel 2.7 "Jumperkonfigurationen und Typenbezeichnungen", Seite 41.

16.1.6 Auspackanleitung und typischer Lieferumfang

Wie bei RTC6 PCIe-Karten, siehe Kapitel 2.2 "Auspackanleitung und typischer Lieferumfang", Seite 32

16.1.7 Mitgelieferte Software

Wie bei RTC6 PCIe-Karten, siehe Kapitel 2.3 "Mitgeliefertes RTC6-Software-Paket", Seite 32.

16.1.8 Zubehör für die RTC6 Ethernet-Karte

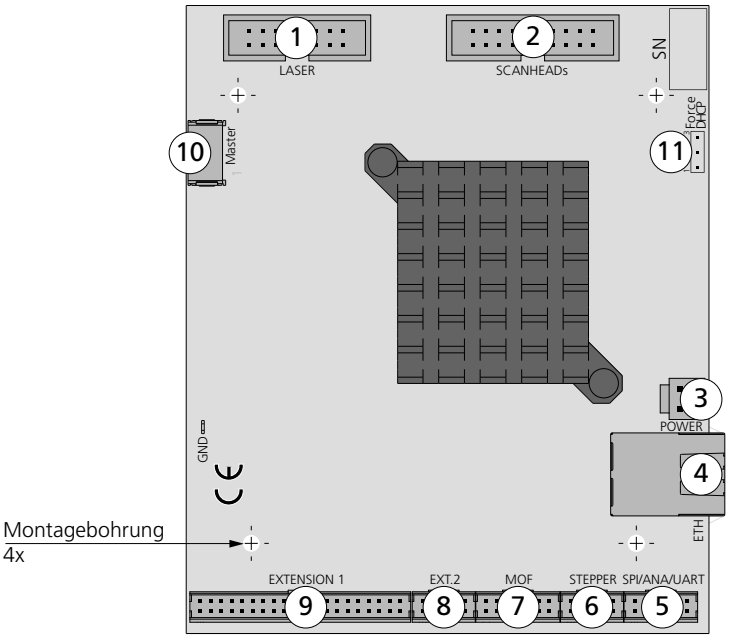
Das in Kapitel 2.8 "Zubehör für die RTC6 PCIe-Karte", Seite 45 genannte Zubehör für RTC6 PCIe-Karten eignet sich (ohne entsprechende Adapter) nicht für RTC6 Ethernet-Karten (wegen abweichender Steckverbinder und Rastermaße).

16.1.9 Ergänzende Software

Wie bei RTC6 PCIe-Karten, siehe Kapitel 2.9 "Ergänzende Software", Seite 48.

16.2 Layout, Schnittstellen, Jumpereinstellungen

16.2.1 Layout – Oberseite

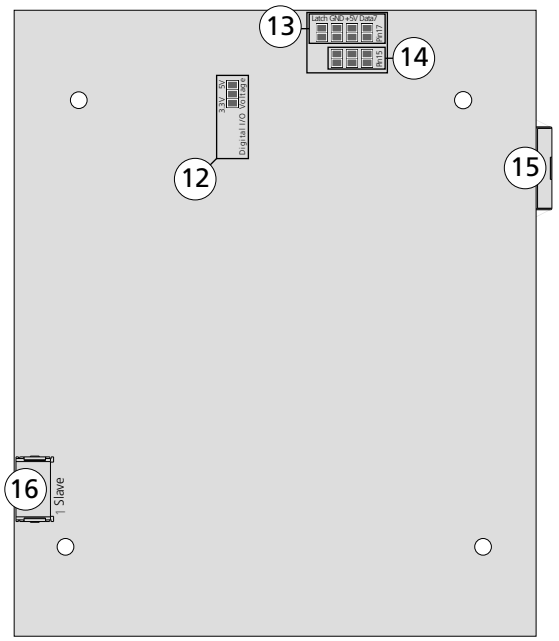


Legende

1. LASER . . . . . 16-polige Stiftleiste. Zum Anschließen des Lasers. Mit digitalen und analogen Ausgängen. Details siehe "LASER-Stiftleiste", Seite 984.
2. SCANHEADs. . . . . 20-polige Stiftleiste. Zum Anschließen des ersten und zweiten Scan-Kopfs. Details siehe "SCANHEADs-Stiftleiste", Seite 987.
3. POWER . . . . . 2-polige Stiftleiste. Für die Spannungsversorgung. Details siehe "POWER-Stiftleiste", Seite 989.
4. ETH . . . . . RJ-48 8P8C-Buchse. Für den Netzwerkanschluss. Details siehe "ETH-Buchse", Seite 989.
5. SPI/ANA/UART . . . . . 12-polige Stiftleiste. Hardware-Schnittstelle für verschiedene Datenaustauschverfahren. Mit analogen Eingängen. Details siehe "SPI/ANA/UART-Stiftleiste", Seite 990.
6. STEPPER. . . . . 10-polige Stiftleiste. Zur Ansteuerung von bis zu zwei Schrittmotoren. Details siehe "STEPPER-Stiftleiste", Seite 991.
7. MOF . . . . . 14-polige Stiftleiste. Hardware-Schnittstelle für Encoderpulse, z.B. für "Processing on the fly"-Anwendungen. Details siehe "MARKING ON THE FLY-Stiftleiste", Seite 85.
8. EXT. 2 . . . . . 10-polige Stiftleiste. Mit einem 8-Bit-Digital-Ausgang. Details siehe "EXT. 2-Stiftleiste", Seite 994.
9. EXTENSION 1. . . . . 40-polige Stiftleiste. Mit einem 16-Bit-Digital-Ausgang und einem 16-Bit-Digital-Eingang. Details siehe EXTENSION 1-Stiftleiste, Seite 993.
10. Master. . . . . 6-polige Stiftleiste. Zum Verbinden mit einer anderen RTC6-Karte zwecks Taktsynchronisierung. Details siehe "Master-Stiftleiste, Slave-Stiftleiste", Seite 996.
11. force DHCP . . . . . Steckjumperfeld zum Ignorieren/nicht Ignorieren der gespeicherten statischen IP-Adresse. Details siehe "Steckjumperfeld 'Force DHCP'", Seite 1000.

RTC6 Ethernet-Karte V1.2. Oberseite.

16.2.2 Layout – Unterseite



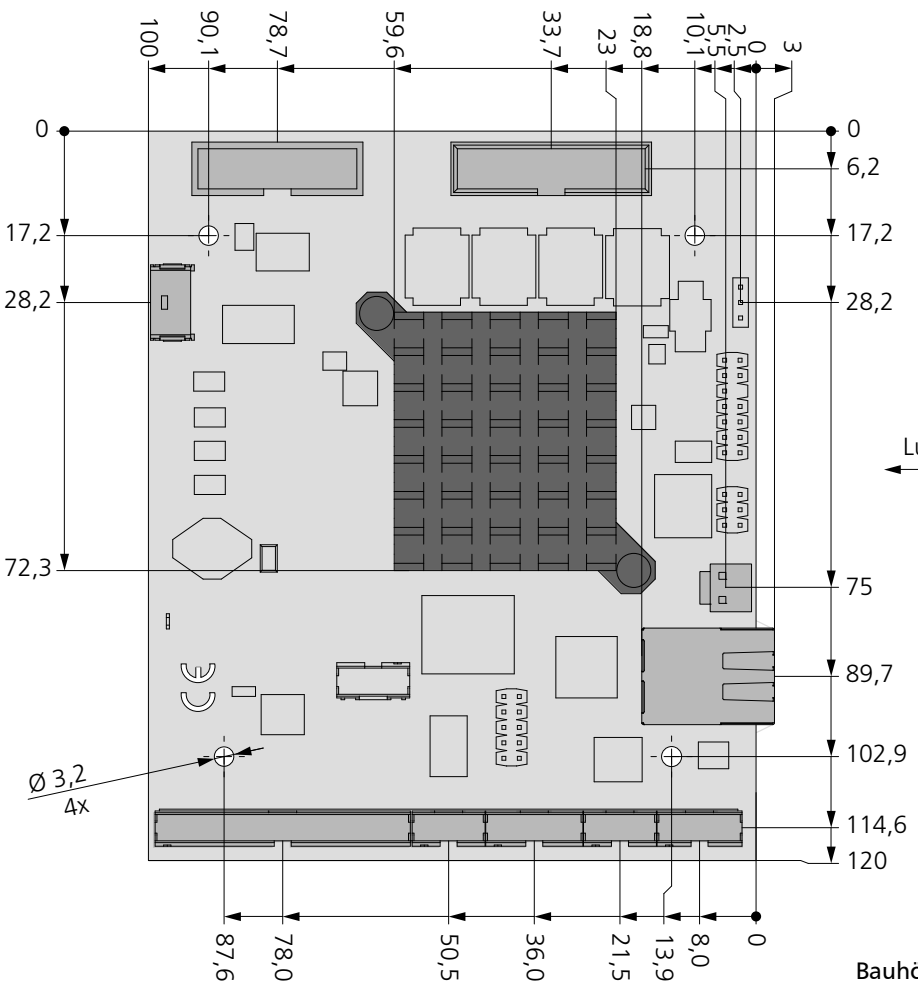
Legende

12. Lötjumperfeld A . . . . . Zum Konfigurieren des Pegels der Ausgangssignale an der EXTENSION 1-Stiftleiste, Seite 993.
Details siehe "Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste", Seite 997.
13. Lötjumperfeld B . . . . . Zum Konfigurieren des Signals an EXT. 2-Stiftleiste Pin (09).
Details siehe "Lötjumperfeld B – Signal an EXT. 2-Stiftleiste Pin (09)", Seite 998.
14. Lötjumperfeld C . . . . . Zum Konfigurieren des Signals an EXT. 2-Stiftleiste Pin (08).
Details siehe "Lötjumperfeld C – Signal an EXT. 2-Stiftleiste Pin (08)", Seite 999.
15. ETH Buchse. . . . . Siehe Abbildung 76.
16. Slave. . . . . 6-polige Stiftleiste. Zum Verbinden mit einer anderen RTC6-Karte zwecks Taktsynchronisierung.
Details siehe "Master-Stiftleiste, Slave-Stiftleiste", Seite 996.

77

RTC6 Ethernet-Karte V1.2. Unterseite.

16.2.3 Abmessungen und Steckverbinderpositionen



Bauhöhe: ca. 28 mm (abhängig vom verbauten Kühlkörper).

78

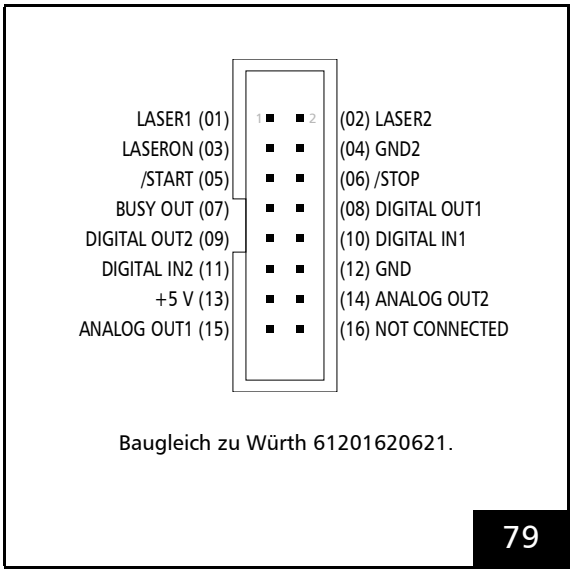
RTC6 Ethernet-Karte V1.2. Abmessungen und Steckverbinderpositionen. Alle Maße in mm.

16.2.4 LASER-Stiftleiste

- Zu den technischen Daten siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", LASER-Stiftleiste, Seite 1066.

Die LASER-Stiftleiste ist 16-polig(1). Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 1.

Die Pinbelegung zeigt Abbildung 79.



Baugleich zu Würth 61201620621.

79

LASER-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm. Bei RTC6 Ethernet-Karten ohne Option "DC/DC-Wandler" sind GND2 und GND identisch. Bei RTC6 Ethernet-Karten mit Option "DC/DC-Wandler" sind GND2 und die Lasersteuersignale galvanisch von GND entkoppelt.

Achtung!

- Wenn Sie die RTC6 Ethernet-Karte in Verbindung mit laserDESK verwenden wollen, beachten Sie folgendes:
  - laserDESK benutzt weitere Pins an der EXTENSION 1-Stiftleiste, Seite 993, um z.B. den Laser zu steuern.
  - Befassen Sie sich deshalb mit der erweiterten Dokumentation zum LASER-Anschluss. Diese finden Sie in der laserDESK-Online-Hilfe (alternativ bei SCANLAB erhältlich oder auch im laserDESK-zip, das Sie von der SCANLAB-Website herunterladen können).

(1) Bei RTC6 PCIe-Karten: 15-polige Sub-D-Buchse, weiblich.

Hinweise

- Bei der RTC6 Ethernet-Karte ist GND die Masse an der POWER-Stiftleiste. Siehe Kapitel 16.2.6 "POWER-Stiftleiste", Seite 989. Bei der RTC6 PCIe-Karte ist GND die PC-Masse.
- Ein geeignetes 0,2 m Kabel (#0116048) ist bei SCANLAB erhältlich, siehe auch Abbildung 80.

Lasersteuersignale

Wie bei RTC6 PCIe-Karten, siehe Kapitel 4.6.1 "LASER-Buchse", Seite 75. Beachten Sie aber, dass sich die Pin-Nummern unterscheiden.

Signal	RTC6 Ethernet -Karte	RTC6 PCIe-Karte
LASERON	Pin (03)	Pin (02)
LASER1	Pin (01)	Pin (01)
LASER2	Pin (02)	Pin (09)

Externe Steuersignale

Wie bei RTC6 PCIe-Karten, siehe Kapitel 4.6.1 "LASER-Buchse", Seite 75.

BUSY-Listenausführungsstatus

Wie bei RTC6 PCIe-Karten, siehe Kapitel 4.6.1 "LASER-Buchse", Seite 75. Beachten Sie aber, dass sich die Pin-Nummern unterscheiden.

Signal	RTC6 Ethernet-Karte	RTC6 PCIe-Karte
BUSY OUT	Pin (07)	Pin (04)

2-Bit-Digital-Eingang, 2-Bit-Digital-Ausgang

Wie bei RTC6 PCIe-Karten, siehe Kapitel 4.6.1 "LASER-Buchse", Seite 75. Beachten Sie aber, dass sich die Pin-Nummern unterscheiden.

Signal	RTC6 Ethernet-Karte	RTC6 PCIe-Karte
DIGITAL IN1	Pin (10)	Pin (13)
DIGITAL IN2	Pin (11)	Pin (06)
DIGITAL OUT1	Pin (08)	Pin (12)
DIGITAL OUT2	Pin (09)	Pin (05)

12-Bit-Analog-Ausgang 1, 12-Bit-Analog-Ausgang 2

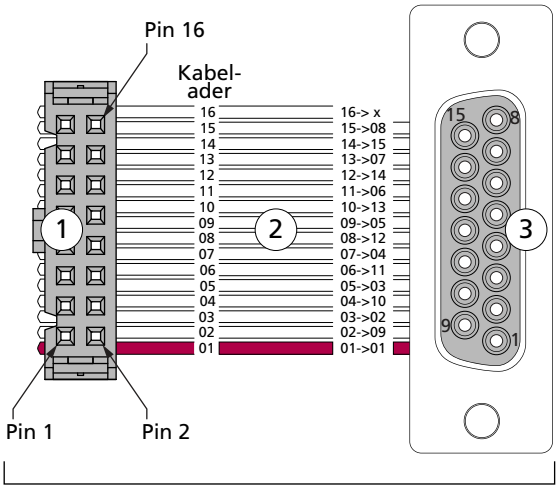
Wie bei RTC6 PCIe-Karten, siehe Kapitel 4.6.1 "LASER-Buchse", Seite 75. Beachten Sie aber, dass sich die Pin-Nummern unterscheiden.

Signal	RTC6 Ethernet-Karte	RTC6 PCIe-Karte
ANALOG OUT1	Pin (15)	Pin (08)
ANALOG OUT2[(a)]	Pin (14)	Pin (15)

(a) Bei der RTC6 PCIe-Karte ist das ANALOG OUT2-Signal zusätzlich auch an der MARKING ON THE FLY-Stiftleiste verfügbar.

Ein-/Ausgangsbeschaltung

Die Ein-/Ausgangsbeschaltung der LASER-Stiftleiste zeigt Abbildung 81. Sie ist mit Ausnahme der Pin-Nummern identisch zu RTC6 PCIe-Karten.



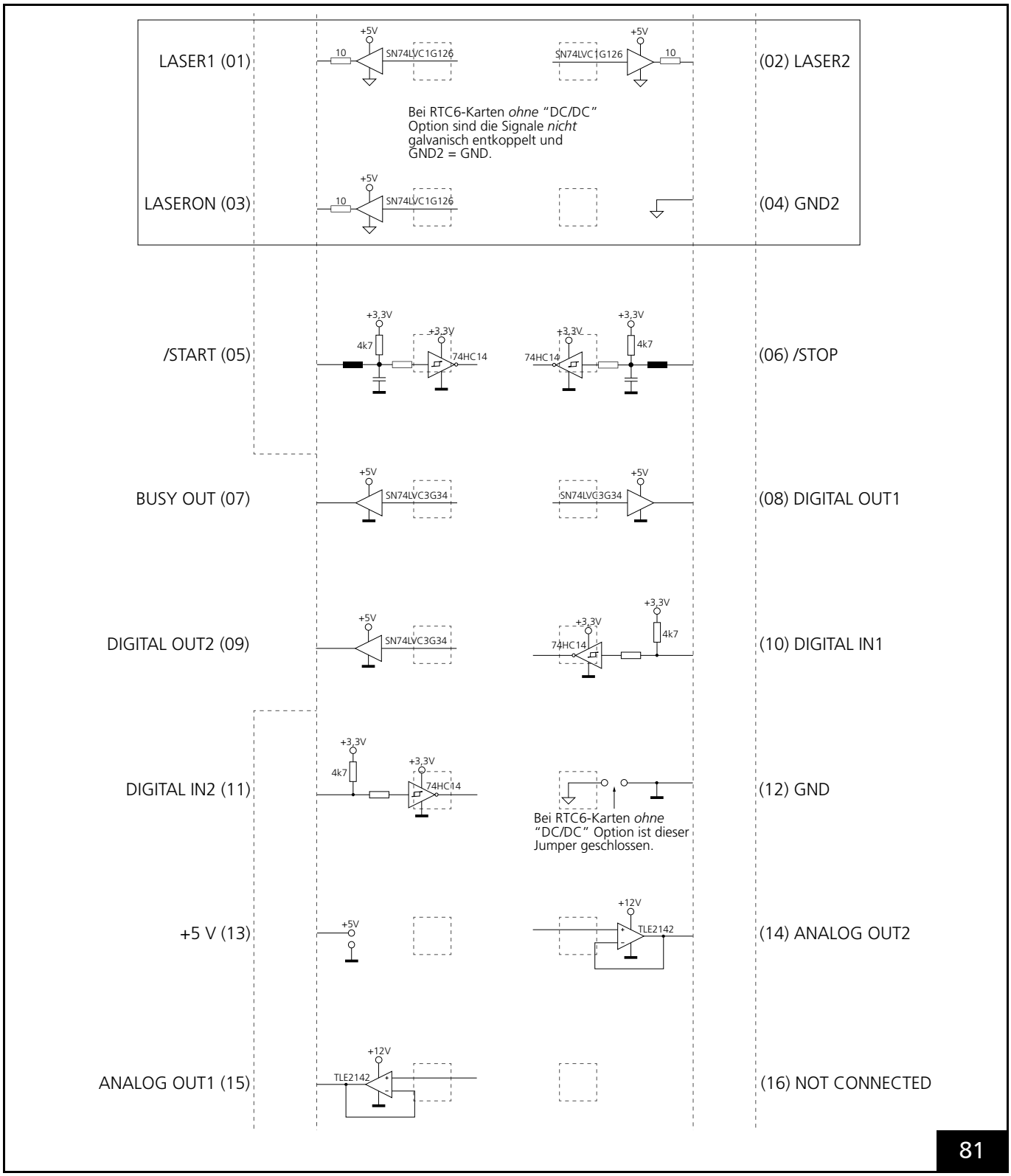
Entspricht funktional Kabel #0116048 (l=0,2 m).

Legende

1. Steckverbinder, 16-polig, weiblich. Für LASER-Stiftleiste, siehe Abbildung 79.
   Baugleich mit Würth 61201623021.
2. Flachbandkabel 26 AWG, 20 Leitungen, Rastermaß 1,27 mm.
3. Steckverbinder, 15-polig Sub-D, weiblich.

80

Kabel (Vorschlag). Die tatsächliche Umsetzung kann auf Grund von Kundenanforderungen abweichend sein.
Die dargestellten Teile sind nicht im Standardlieferumfang der RTC6 Ethernet-Karte enthalten!



81

LASER-Stiftleiste (Ausschnitt, gestrichelt angedeutet): Ein-/Ausgangsbeschaltung.

16.2.5 SCANHEADs-Stiftleiste

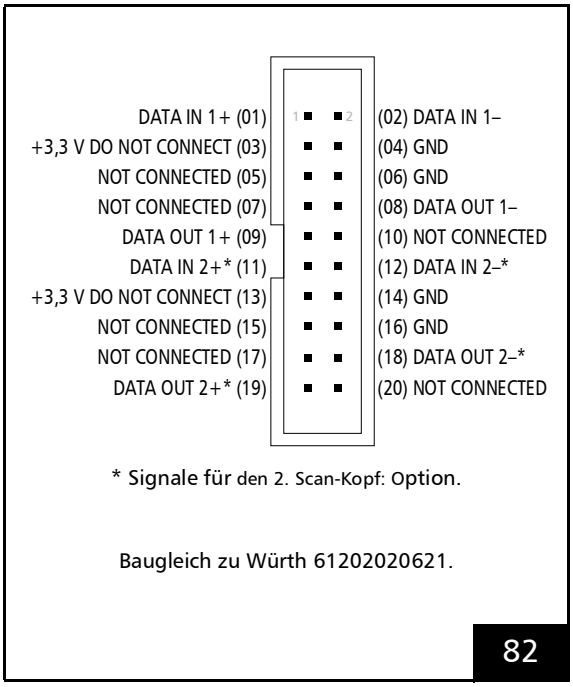
- Zu den technischen Daten siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", SCANHEADs-Stiftleiste, Seite 1066.

Die SCANHEADs-Stiftleiste ist 20-polig. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 2.

Die SCANHEADs-Stiftleiste stellt die gleichen Signale bereit wie die RTC6 PCIe-Karte:

- Pin (01)…Pin (10) wie SCANHEAD-Buchse (Anschluss für ersten Scan-Kopf)
- Pin (11)…Pin (20) wie 2. SCANHEAD-Stiftleiste (Anschluss für zweiten Scan-Kopf)

Die Pinbelegung zeigt Abbildung 82.

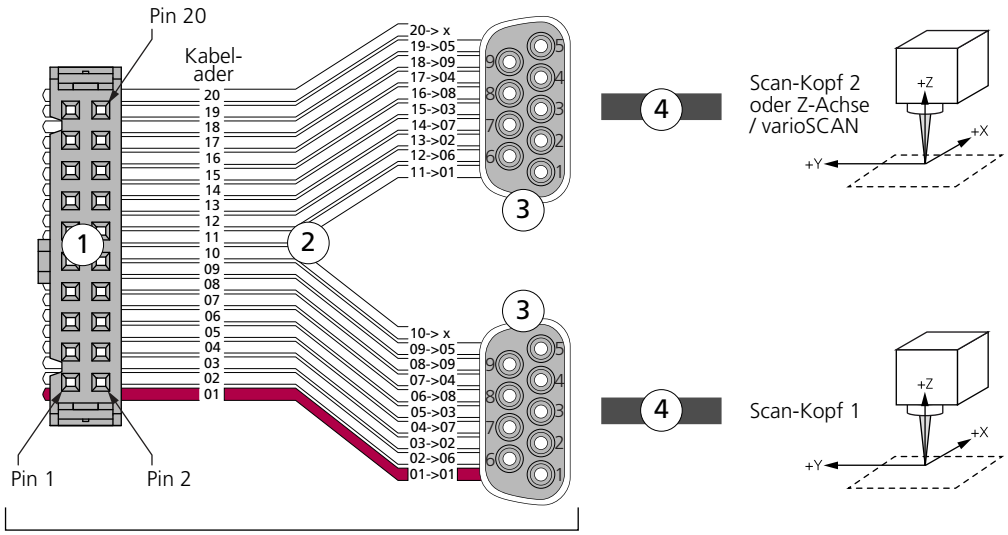


SCANHEADs-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,54 mm.

Ein Beispiel zur Verkabelung zeigt Abbildung 83 (das gezeigte Y-Kabel ist mit der Länge 0,2 m bei SCANLAB erhältlich: #0116050).

Die Ansteuerung von Scan-Systemen (gegebenenfalls über einen XY2-100-Konverter (Zubehör)) erfolgt identisch zur RTC6 PCIe-Karte.

Zu weiteren Informationen siehe Kapitel 4.5 "Schnittstellen zum Scan-System", Seite 69.



Entspricht funktional Y-Kabel #0116050 (l=0,2 m).

Legende

1. Steckverbinder, 20-polig, weiblich. Für SCANHEADs-Stiftleiste, siehe Abbildung 82. Baugleich mit Würth 61202023021.
2. Flachbandkabel 26 AWG, 20 Leitungen, Rastermaß 1,27 mm.
3. Steckverbinder, 9-polig Sub-D, weiblich.
4. Datenkabel für SL2-100-Protokoll, z.B. #0115428.

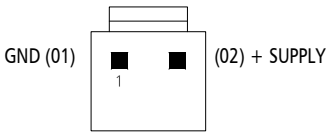
83

Verkabelung (Vorschlag). Die tatsächliche Umsetzung kann auf Grund von Kundenanforderungen abweichend sein. Die dargestellten Teile sind nicht im Standardlieferumfang der RTC6 Ethernet-Karte enthalten!

16.2.6 POWER-Stiftleiste

Die POWER-Stiftleiste ist 2-polig. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 3. Über sie wird der RTC6 Ethernet-Karte die Versorgungsspannung zugeführt.

Die Pinbelegung zeigt Abbildung 84.



Hersteller: JST.
Teile-Nr.: B2P-VH(LF)(SN).

84

POWER-Stiftleiste. Das Rastermaß der Pins ist 3,96 mm.

Pin	Versorgungsspannung
(01) GND	Masse.
(02) + Supply	Siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", Seite 1065.

Hinweise

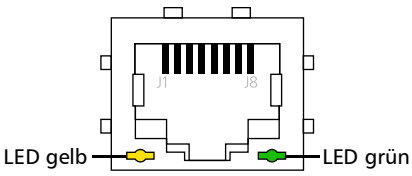
- Die für den Gegen-Stecker benötigten Teile sind *nicht im Lieferumfang enthalten*:
  Hersteller: JST
  1× Steckergehäuse, Teile-Nr. VHR-2N
  2× Crimpkontakt, Teile-Nr. SVH-41T-P1.1.

16.2.7 ETH-Buchse

Die ETH-Buchse ist ein RJ-Steckverbinder vom Typ 8P8C (umgangssprachlich: RJ-45). Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 4. Sie dient dazu, die RTC6 Ethernet-Karte an das Netzwerk anzuschließen(1).

In die Buchse sind folgende LEDs zur Statusanzeige integriert, siehe Abbildung 85:

- LED gelb, Funktion "IP address assigned".
  Die LED leuchtet *dauerhaft*, sobald die RTC6 Ethernet-Karte eine IP-Adresse bezogen hat (statische IP, DHCP oder ab BIOS 24 Link-local Adresse(2)).
  Ohne gespeicherte statische IP zeigt die LED die erfolgreiche IP-Adressvergabe mit DHCP an. Im Falle einer gespeicherten statischen IP bedeutet eine aktive LED nicht zwingend eine im Subnetz gültige IP-Adresse. Die LED *blinkt* bei Netzwerkverkehr ("Traffic"), sobald die Karte von einem PC in Besitz genommen (akquiriert) ist.
- LED grün, Funktion "Link / Traffic".
  Die LED leuchtet *dauerhaft*, sobald eine Verbindung hergestellt ist ("Link"). Das ist gewöhnlich der Fall, wenn das Ethernet-Kabel angesteckt und auf der Gegenseite ein Router oder eine Netzwerkkarte aktiv ist.
  Die LED *blinkt* bei Netzwerkverkehr ("Traffic").



85

ETH-Buchse. RJ-48 8P8C.

(1) Ethernet-Kabel der Kategorie 5e (Cat 5e) empfohlen.

(2) Siehe Seite 1002.

16.2.8 SPI/ANA/UART-Stiftleiste

- Zu den technischen Daten siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", SPI/ANA/UART-Stiftleiste, Seite 1069.

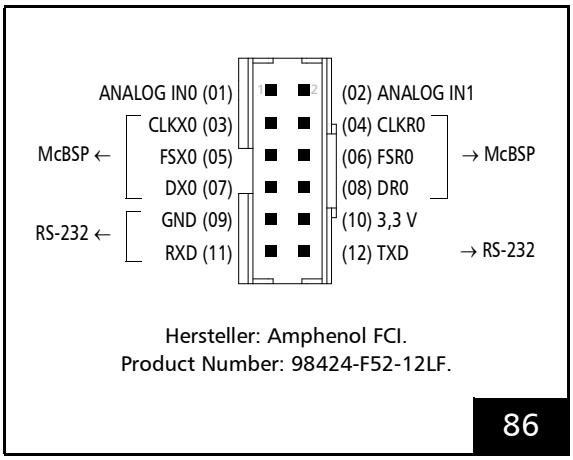
Die SPI/ANA/UART-Stiftleiste ist 12-polig. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 5.

Die SPI/ANA/UART-Stiftleiste ist eine Hardware-Schnittstelle, die für Analog-Eingänge sowie die Datenaustauschverfahren McBSP (Multichannel Buffered Serial Port) und RS-232 vorgesehen ist.

Die SPI/ANA/UART-Stiftleiste entspricht funktional der:

- McBSP/ANALOG-Stiftleiste und RS232-Stiftleiste bei RTC6 PCIe-Karten

Die entsprechenden Pinbelegungen Pinbelegungzeigt Abbildung 86.



SPI/ANA/UART-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,00 mm.

Hinweise

- Die für den Gegen-Stecker benötigten Teile sind *nicht im Lieferumfang enthalten*:
  Hersteller: Amphenol FCI
  Serie: Minitek®
  1× Steckergehäuse, Teile-Nr. 90311-012LF
  12× Crimpkontakt, Teile-Nr. 77138-101LF.

Analog-Eingänge

An der SPI/ANA/UART-Stiftleiste werden 2 Analog-Eingänge bereitgestellt:

- ANALOG IN0
  – Siehe Abschnitt "Analog-Eingänge", Seite 89
- ANALOG IN1
  – Siehe Abschnitt "Analog-Eingänge", Seite 89

Technische Spezifikationen siehe SPI/ANA/UART-Stiftleiste, Seite 1069.

McBSP-Schnittstelle

Siehe Abschnitt "McBSP-Schnittstelle", Seite 87.

RS-232 Schnittstelle

Die RS232-Stiftleisten-Pins der RTC6 PCIe-Karte sind bei RTC6 Ethernet-Karten in die SPI/ANA/UART-Stiftleiste(1) integriert.

Spezifikationen und weitere Informationen wie RTC6 PCIe-Karte, siehe Kapitel 4.6.5 "RS232-Stiftleiste", Seite 86.

Die Signale beziehen sich auf GND, siehe Hinweise, Seite 984.

(1) Die Beschriftung der Stiftleiste lautet "…/UART" (Universal Asynchronous Receiver/Transmitter). Bezieht sich auf die elektronische Schaltung, die die auf der RS-232-Schnittstelle zu übertragenden Daten-Bits und den dazu notwendigen Datenrahmen erzeugt.

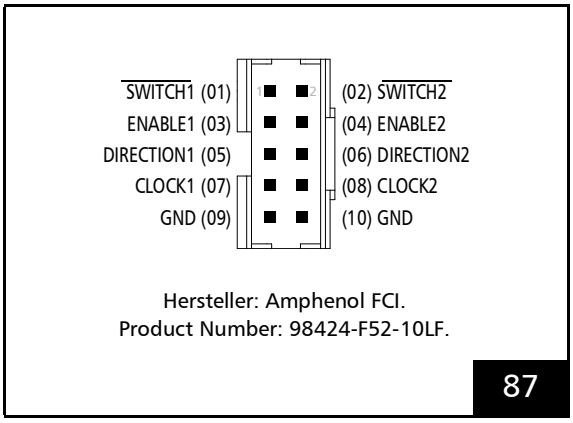
16.2.9 STEPPER-Stiftleiste

- Zu den technischen Daten siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", STEPPER-Stiftleiste, Seite 1069.

Die STEPPER-Stiftleiste ist 10-polig. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 6.

An der STEPPER-Stiftleiste können Signale zur Ansteuerung von bis zu zwei Schrittmotoren ausgegeben werden.

Die Pinbelegung zeigt Abbildung 87.



STEPPER-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,00 mm.

Alle Signale beziehen sich auf GND, siehe Hinweise, Seite 984.

Spezifikationen und weitere Informationen wie RTC6 PCIe-Karte, siehe Kapitel 4.6.7 "STEPPER MOTOR-Stiftleiste", Seite 90.

Hinweise

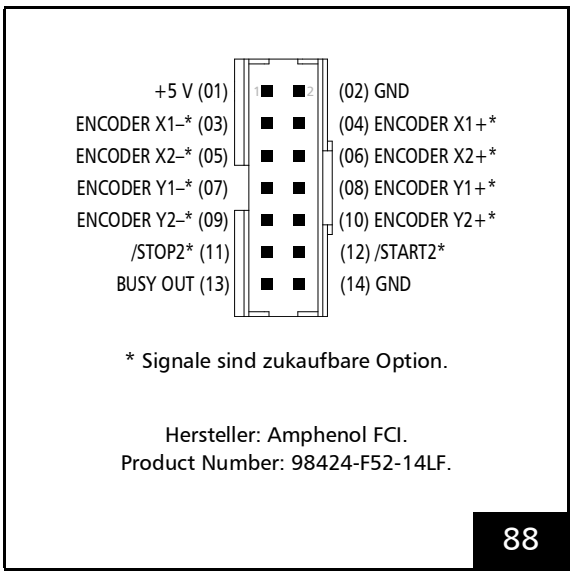
- Die für den Gegen-Stecker benötigten Teile sind *nicht im Lieferumfang enthalten*
  Hersteller: Amphenol FCI
  Serie: Minitek®
  1× Steckergehäuse, Teile-Nr. 90311-010LF
  10× Crimpkontakt, Teile-Nr. 77138-101LF

16.2.10MOF-Stiftleiste

- Zu den technischen Daten siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", MOF-Stiftleiste, Seite 1068.

Die MOF-Stiftleiste(1) ist 14-polig. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 7.

Die Pinbelegung zeigt Abbildung 88.



MOF-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,00 mm.

Hinweise

- Die für den Gegen-Stecker benötigten Teile sind *nicht im Lieferumfang enthalten*:
  Hersteller: Amphenol FCI
  Serie: Minitek®
  1× Steckergehäuse, Teile-Nr. 90311-014LF
  14× Crimpkontakt, Teile-Nr. 77138-101LF.
- Die RTC6 Ethernet-Karte hat an ihrer MOF-Stiftleiste *keinen* Pin für das ANALOG OUT2-Signal, siehe Kapitel 4.6.4 "MARKING ON THE FLY-Stiftleiste", Seite 85.

Encodereingänge

Wie bei RTC6 PCIe-Karte, siehe Abschnitt "Encodereingänge", Seite 85.

Externe Steuersignale

Wie bei RTC6 PCIe-Karte, siehe Abschnitt "Externe Steuersignale", Seite 85.

Die Signale beziehen sich auf GND, siehe Hinweise, Seite 984.

BUSY OUT-Status

Das BUSY OUT-Signal an Pin (13) ist identisch mit dem BUSY OUT-Signal an der LASER-Stiftleiste, Seite 984.
Das Signal bezieht sich auf GND, siehe Hinweise, Seite 984.

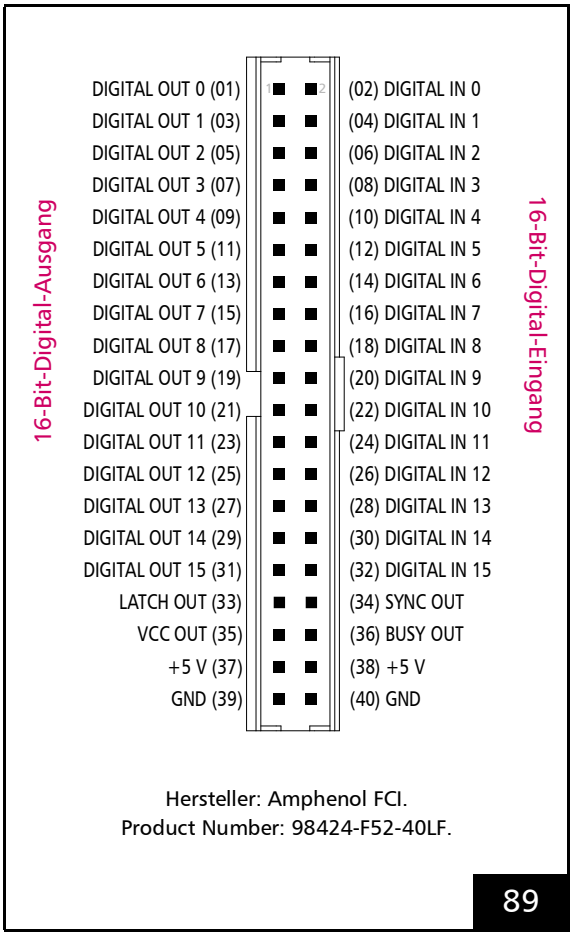
(1) Marking on the fly (= "Processing on the fly").

16.2.11EXTENSION 1-Stiftleiste

- Zu den technischen Daten siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", EXTENSION 1-Stiftleiste, Seite 1067.

Die EXTENSION 1-Stiftleiste ist 40-polig. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 9.

Die Pinbelegung zeigt Abbildung 89.



EXTENSION 1-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,00 mm.

Hinweise

- Die für den Gegen-Stecker benötigten Teile sind *nicht im Lieferumfang enthalten*:
  Hersteller: Amphenol FCI
  Serie: Minitek®
  1× Steckergehäuse, Teile-Nr. 90311-040LF
  40× Crimpkontakt, Teile-Nr. 77138-101LF.
- Die EXTENSION 1-Stiftleiste hat bei RTC6 Ethernet-Karten ein Rastermaß der Pins von 2,00 mm, bei RTC6 PCIe-Karten dagegen 2,54 mm.

Ausgangssignal-Pegel einstellen

Mit dem Lötjumperfeld A auf der Unterseite der RTC6 Ethernet-Karte kann für alle Ausgangssignale der EXTENSION 1-Stiftleiste (DIGITAL OUT 0…DIGITAL OUT 15, LATCH_OUT, SYNC_OUT, BUSY OUT, VCC_OUT) ein Signal-Pegel von 5 V oder 3,3 V eingestellt werden, siehe Abschnitt "Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste", Seite 997.

Der eingestellte Signalpegel wird statisch an Pin (35) ausgegeben: Signal VCC_OUT. VCC_OUT bezieht sich auf GND, siehe Hinweise, Seite 984.
Die maximale Belastung des Signals ist 100 mA.

16-Bit-Digital-Ausgang und 16-Bit-Digital-Eingang

Wie bei RTC6 PCIe-Karte, siehe Abschnitt "16-Bit-Digital-Eingang und 16-Bit-Digital-Ausgang", Seite 81.

Die Signale beziehen sich auf GND, siehe Hinweise, Seite 984.

Synchronisation der Datenübergabe

Wie bei RTC6 PCIe-Karte, siehe Abschnitt "Synchronisation der Datenübergabe", Seite 82.

Die Signale beziehen sich auf GND, siehe Hinweise, Seite 984.

BUSY-Listenausführungsstatus

Das BUSY OUT-Signal an Pin (36) ist identisch mit dem BUSY OUT-Signal an der LASER-Stiftleiste, Seite 984.
Das Signal bezieht sich auf GND, siehe Hinweise, Seite 984.

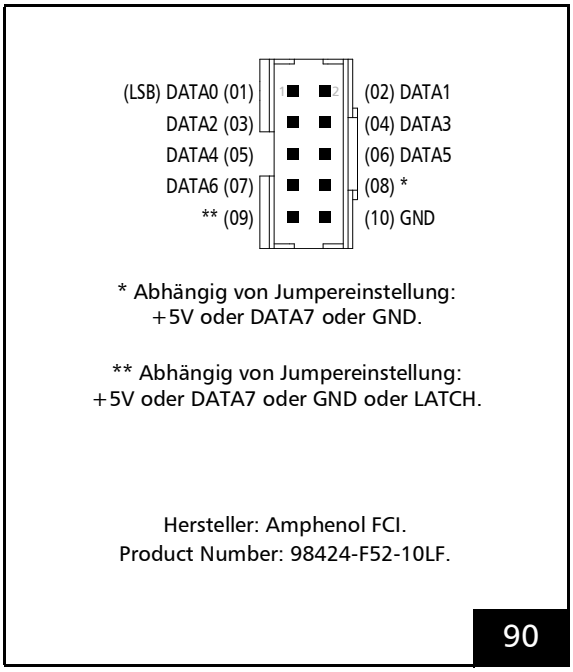
16.2.12EXT. 2-Stiftleiste

- Zu den technischen Daten siehe Kapitel 16.12 "Technische Spezifikationen – RTC6 Ethernet-Karte", EXT. 2-Stiftleiste, Seite 1068.

Die EXT. 2-Stiftleiste ist 10-polig. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 8.

Sie stellt einen gepufferten 8-Bit-Digital-Ausgang bereit: DATA0…DATA7.

Die Pinbelegung zeigt Abbildung 24.



EXT. 2-Stiftleiste: Pinbelegung. Das Rastermaß der Pins ist 2,00 mm.

Auf RTC6 PCIe-Karten ist die Bezeichnung EXTENSION 2.

Neben der Gesamtanzahl der Pins unterscheiden sich die bereitgestellten Signale sowie auch die Pin-Nummern, siehe nachfolgende Tabelle.

Signal	RTC6 Ethernet-Karten, EXT. 2-Stiftleiste 10-polig	RTC6 PCIe-Karten, EXTENSION 2-Stiftleiste, 26-polig
DATA0	Pin (01)	Pin (01)
DATA1	Pin (02)	Pin (03)
DATA2	Pin (03)	Pin (05)
DATA3	Pin (04)	Pin (07)
DATA4	Pin (05)	Pin (09)
DATA5	Pin (06)	Pin (11)
DATA6	Pin (07)	Pin (13)
*, siehe in Abbildung 24	Pin (08)	Pin (15)
**, siehe in Abbildung 24	Pin (09)	Pin (17)
GND	Pin (10)	Pin (02)
+5 V	–	Pin (06), (18), (25)
LASER1	–	Pin (22)
LASER2	–	Pin (19)
GND2	–	Pin (23)

Hinweise

- Die für den Gegen-Stecker benötigten Teile sind *nicht im Lieferumfang enthalten*:
  Hersteller: Amphenol FCI
  Serie: Minitek®
  1× Steckergehäuse, Teile-Nr. 90311-010LF
  10× Crimpkontakt, Teile-Nr. 77138-101LF.
- Pin (08) und Pin (09) können mittels Lötjumper eingestellt werden.
- Eine RTC6 Ethernet-Karte hat an ihrer EXT. 2-Stiftleiste *keine* Pins für LASER1 und LASER2.

Einstellungen mittels Lötjumper

Pin (08) der EXT. 2-Stiftleiste wird über das Lötjumperfeld C konfiguriert, Pin (09) über das Lötjumperfeld B. Beide Lötjumperfelder befinden sich auf der Unterseite der RTC6 Ethernet-Karte, siehe Abbildung 77. Für weitere Informationen siehe Abschnitt "Lötjumperfeld B – Signal an EXT. 2-Stiftleiste Pin (09)", Seite 998 und Abschnitt "Lötjumperfeld C – Signal an EXT. 2-Stiftleiste Pin (08)", Seite 999.

Hinweise

- Ist das DATA7-Bit an Pin (08) zugewiesen, dann ist der volle 8-Bit Ausgangswert am Ausgang (an den Pins (1) bis Pin (08) der EXT. 2-Stiftleiste) verfügbar.
- Ist Pin (08) auf +5 V (HIGH-Pegel) eingestellt, führt das zu einem Offset von 128 für die Ausgabewerte, d. h. die Ausgabewerte liegen zwischen 128…255.
- Ist Pin (08) auf GND (LOW-Pegel) eingestellt, so sind nur Ausgabewerte zwischen 0 und 127 möglich.
- Das DATA7-Bit kann für andere Zwecke verwendet werden, indem es Pin (09) zugewiesen wird.

8-Bit-Digital-Ausgang

Wie bei RTC6 PCIe-Karte, siehe Abschnitt "8-Bit-Digital-Ausgang", Seite 84.

16.2.13Master-Stiftleiste, Slave-Stiftleiste

Die Master-Stiftleiste ist 6-polig, siehe Abbildung 91. Sie befindet sich auf der Oberseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 10.

Die Slave-Stiftleiste ist 6-polig, siehe Abbildung 91. Sie befindet sich auf der Unterseite der RTC6 Ethernet-Karte, siehe in Abbildung 76, Nummer 16.

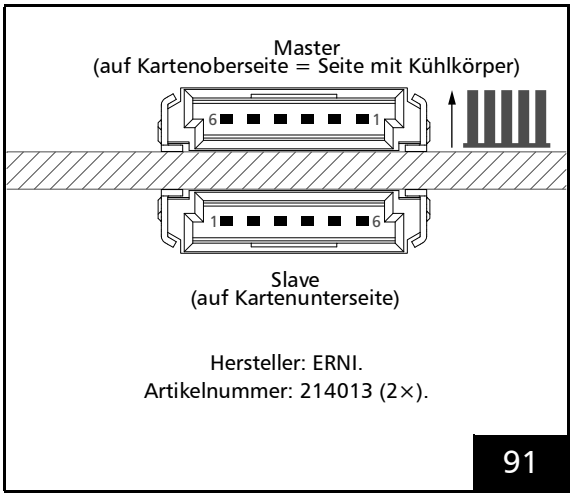
Die beiden Stiftleisten dienen dazu, um die Takte mehrerer RTC6-Karten zu synchronisieren.

Verbindungskabel sind bei SCANLAB erhältlich, siehe Abbildung 9, Seite 68. Die notwendigen Angaben zum Konfektionieren eigener Kabel sind in Abbildung 8, Seite 68 gezeigt.

Zu weiteren Informationen siehe

- Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127
- Kapitel 9.3.1 "Listen mit externen Steuersignalen und Master/Slave-Synchronisation ausführen und beenden", Seite 314.

Für eine ordnungsgemäße Funktion der Master/Slave-Taktsynchronisation ist eine bestimmte Hardware-Revision der RTC6 Ethernet-Karte erforderlich (siehe Aufkleber mit Artikelnummer und Änderungsindex). Wenn Sie eine ältere RTC6 Ethernet-Karte haben oder sich nicht sicher sind, wenden Sie sich an SCANLAB.



Master-Stiftleiste und Slave-Stiftleiste. Das Rastermaß der Pins ist 1,27 mm.

Achtung!

- Master/Slave-verbundene RTC6 Ethernet-Karten werden durch unterschiedliche Potentiale zerstört. Das kann selbst dann der Fall sein, wenn die Karten an die gleiche Spannungsversorgung angeschlossen sind, jedoch die Verkabelung ungünstig ausgeführt ist. Die empfohlene Verkabelung finden Sie in Abbildung 92.
  Um unterschiedliche Potentiale zu vermeiden:
  (1) Stellen Sie sicher, dass die Spannungsversorgung ausgeschaltet ist.
  (5) Verkabeln Sie zuerst eine RTC6 Ethernet-Karte mit der Spannungsversorgung, und dann die andere durch eine Abzweigung dieses Kabels (Länge möglichst kurz halten).
  (2) Schalten Sie erst danach die Spannungsversorgung wieder ein.
- Bevor Sie die Verkabelung von Master/Slave-verbundenen RTC6 Ethernet-Karten trennen, schalten Sie immer unbedingt vorher die Spannungsversorgung aller verbundenen Karten aus.

16.2.14Jumpereinstellungen

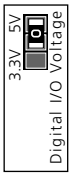
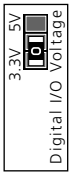
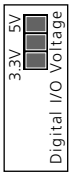
SCANLAB liefert RTC6 Ethernet-Karten in unterschiedlichen Jumperkonfigurationen aus. Die Jumper können nachträglich umkonfiguriert werden. Siehe Kapitel 2.7 "Jumperkonfigurationen und Typenbezeichnungen", Seite 41.

Achtung!

- Nur zulässige Jumpereinstellungen vornehmen. Anderenfalls wird die Karte beschädigt!

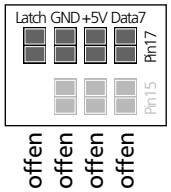
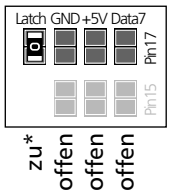
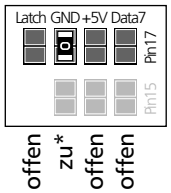
Lötjumperfeld A – Ausgangssignal-Pegel an der EXTENSION 1-Stiftleiste

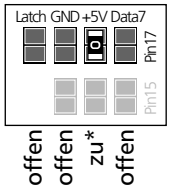
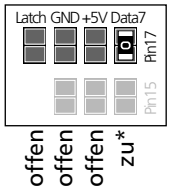
- Position auf der Karte: Unterseite, siehe in Abbildung 77, Nummer 12.
- Zweck: Einstellen des Pegels (5 V oder 3,3 V) für alle Ausgangssignale der EXTENSION 1-Stiftleiste, siehe folgende Tabelle.
- Siehe auch Abschnitt "Ausgangssignal-Pegel einstellen", Seite 81.

Zulässige Jumpereinstellung	**An der** EXTENSION 1-Stiftleiste
 zu* offen	Ausgangssignal-Pegel 5 V.
 offen zu*	Ausgangssignal-Pegel 3,3 V.
 offen offen	Keine Ausgangssignale.
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!	

Lötjumperfeld B – Signal an EXT. 2-Stiftleiste Pin (09)

- Position auf der Karte: Unterseite, siehe in Abbildung 77, Nummer 13.
- Zweck: Einstellen des Signals am Pin (09) der EXT. 2-Stiftleiste, siehe folgende Tabelle.
- Siehe auch "Einstellungen mittels Lötjumper", Seite 83.
- Die Konfigurationen von Lötjumperfeld B und Lötjumperfeld C sind nicht voneinander abhängig.
- Auf der RTC6 Ethernet-Karte zeigt die Bedruckung des Lötjumperfeld B 'Pin 17'. Diese wurde bewusst gewählt, um die Konsistenz mit bisherigen RTC-Karten zu bewahren. Tatsächlich wird bei RTC6 Ethernet-Karten jedoch Pin (09) konfiguriert.

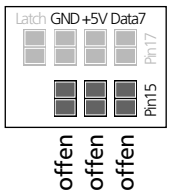
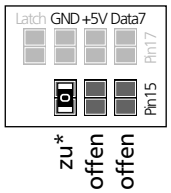
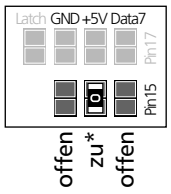
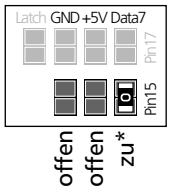
Zulässige Jumpereinstellung	**Ausgabe an der EXT. 2-Stiftleiste Pin (09)**
	Kein Signal.
	LATCH-Signal.
	GROUND (Low-Pegel).
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!	

Zulässige Jumpereinstellung (Forts.)	**Ausgabe an der EXT. 2-Stiftleiste Pin (09) (Forts.)**
	+5 V (High-Pegel).
	DATA7[(a)].
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!	

(a) Synonym: Daten-Bit #7. MSB des 8-Bit-Ausgangswerts.

Lötjumperfeld C – Signal an EXT. 2-Stiftleiste Pin (08)

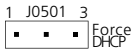
- Position auf der Karte: Unterseite, siehe in Abbildung 77, Nummer 14.
- Zweck: Einstellen des Signals am Pin (08) der EXT. 2-Stiftleiste, siehe folgende Tabelle.
- Siehe auch "Einstellungen mittels Lötjumper", Seite 83.
- Die Konfigurationen von Lötjumperfeld C und Lötjumperfeld B sind nicht voneinander abhängig.
- Auf der RTC6 Ethernet-Karte zeigt die Bedruckung des Lötjumperfeld C 'Pin 15'. Diese wurde bewusst gewählt, um die Konsistenz mit bisherigen RTC-Karten bewahren. Tatsächlich wird bei RTC6 Ethernet-Karten jedoch Pin (08) konfiguriert.

Zulässige Jumpereinstellung	Ausgabe an der EXT. 2-Stiftleiste Pin (08)
	Kein Signal.
	GROUND (Low-Pegel).
	+5 V (High-Pegel).
	DATA7[(a)].
*Achtung: nur *eine einzige* Position in diesem Lötjumperfeld schließen. Andere Kombinationen sind nicht zulässig und beschädigen die Karte!	

(a) Synonym: Daten-Bit #7. MSB des 8-Bit-Ausgangswerts.

Steckjumperfeld 'Force DHCP'

- Position auf der Karte: Oberseite, siehe in Abbildung 76, Nummer 11.
- Zweck: siehe folgende Tabelle.

Zulässige Jumpereinstellung	Auswirkung
 offen zu	Beim Einschalten der RTC6 Ethernet-Karte soll eine IP-Adresse über DHCP bezogen werden:(a) "Force DHCP-Stellung" Es werden für alle konfigurierbaren Netzwerkparameter (statische IP, zugehörige Netzmaske, Gateway, UDP- und TCP-Ports) die Default-Einstellungen verwendet (anstatt der auf der Karte gespeicherten).
 zu offen*	Wenn benutzerdefinierte Netzwerkeinstellungen abgespeichert sind, werden diese beim Einschalten der RTC6 Ethernet-Karte verwendet.
 offen offen	Wie "offen zu".
*Auslieferungszustand.	

(a) Siehe auch Seite 1002.

16.2.15Echtzeit-Uhr

Die RTC6 Ethernet-Karte ist mit einer Echtzeit-Uhr ausgestattet. Ist die Karte stromlos, läuft sie noch etwa > 1 Woche weiter. Bei einem Hardware-Reset wird diese Zeit automatisch übernommen (kein **time_update** erforderlich).

Mit **time_update** wird die Echtzeit-Uhr automatisch mit der PC-Zeit synchronisiert. Die Ganggenauigkeit kann mit **time_control_eth** feinabgestimmt werden.

16.3 Installation und Inbetriebnahme

16.3.1 Hardware-Installation

Achtung!

- Die Karte sollte immer in einer elektrostatisch neutralen Umgebung im mitgelieferten antistatischen Beutel gelagert werden.
- Beachten Sie beim Einbau der Karte die ESD-Schutzvorschriften.
- Berühren Sie die Karte nicht an den elektrischen Kontakten.
- Schützen Sie die Karte vor Feuchtigkeit, Staub, korrosiven Dämpfen und mechanischen Belastungen.

Der Einbau der RTC6 Ethernet-Karte in ein abgeschirmtes Gehäuse erfolgt in den folgenden Arbeitsschritten:

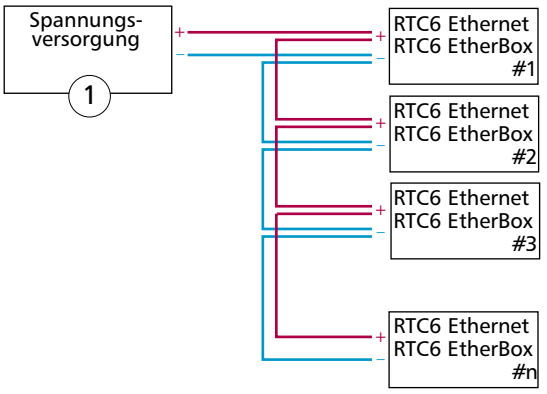
(1) Nehmen Sie die RTC6 Ethernet-Karte aus dem antistatischen Beutel. Berühren Sie die Karte nicht an den elektrischen Kontakten.

(2) Befestigen Sie die RTC6 Ethernet-Karte am dafür vorgesehenen Platz im abgeschirmten Gehäuse.

(3) Verbinden Sie die RTC6 Ethernet-Karte und
- Netzteil mittels Kabel zur Spannungsversorgung
  – Beachten Sie auch Abbildung 92 –
- Ethernet mittels Ethernet-Kabel

(4) Verbinden Sie die RTC6 Ethernet-Karte mittels geeigneter Kabel mit:
- Scan-Kopf
- Laser

(5) Wenn Sie die Signale der RTC6 Ethernet-Karten-Stiftleisten verwenden wollen, dann schließen Sie entsprechende Verbindungskabel an.



Legende

1. Für RTC6 Ethernet-Karte +11 V…+50 V
   Für RTC6 EtherBox +12 V…+30 V

Achtung!

- Zur Spannungsversorgung:
  – "Daisy-Chain"-Verkabelung
  – Keine Stern-Verkabelung
- Gilt unabhängig davon, ob die Geräte Master/Slave-verbunden sind oder nicht.

92

Empfohlene Verkabelung zum Verbinden mehrerer RTC6 Ethernet-Karten/RTC6 EtherBoxen.

16.3.2 Software-Installation

Die Installation und Inbetriebnahme der RTC6 Ethernet-Karte und der zugehörigen Software erfolgt analog zu RTC6 PCIe-Karten, siehe Kapitel 5 "Installation und Inbetriebnahme", Seite 91. Allerdings entfällt Schritt 3, Seite 91:

RTC6 Ethernet-Karten benötigen den RTC6-Kartentreiber nicht.

16.3.3 Mit einem Netzwerk verbinden

Verbinden Sie die RTC6 Ethernet-Karte mit dem gewünschten Netzwerksegment und schalten Sie die Versorgungsspannung ein. Die grüne LED an der ETH-Buchse, siehe Kapitel 16.2.7 "ETH-Buchse", Seite 989, sollte nun aufleuchten und gegebenenfalls blinken.

Ist keine statische IP-Adresse auf der RTC6 Ethernet-Karte gespeichert, ist ein DHCP-Server notwendig, um eine dynamische IP-Adresse zu beziehen. In diesem Fall sollte die gelbe LED nach einigen Sekunden aufleuchten. Damit wird der erfolgreiche Bezug einer IP-Adresse signalisiert.

Ist in "Force DHCP-Stellung" (ab BIOS 24) innerhalb von 60 Sekunden nach dem Einschalten keine IP-Adresse zugewiesen worden, dann wird automatisch die Link-local Adresse 169.254.1.0/16 verwendet.

Wenn eine statische IP-Adresse auf der RTC6 Ethernet-Karte abgespeichert ist, leuchtet die gelbe LED sofort nach Einschalten der RTC6 Ethernet-Karte auf.

Wichtig: Es ist möglich, dass die auf der RTC6 Ethernet-Karte gespeicherte IP-Konfiguration für das angeschlossene Netzwerksegment nicht gültig ist. Auch in diesem Fall leuchtet die gelbe LED auf. Jedoch ist die RTC6 Ethernet-Karte im Netzwerk nicht ansprechbar.

16.4 Hinweise zur Migration und Neu-Programmierung von RTC6-Anwenderprogrammen

RTC6 Ethernet-Karten müssen manuell in die RTC6-Kartenverwaltung eingetragen werden (**init_rtc6_dll** sucht nur nach RTC6 PCIe-Karten), siehe Kapitel 16.5.3 "Über die RTC6-Kartenverwaltung", Seite 1008.

Danach können RTC6 Ethernet-Karten prinzipiell genauso wie RTC6 PCIe-Karten verwendet werden.

Die Kommunikation über Ethernet ist im Allgemeinen unsicherer als eine PCIe-Bus-Verbindung innerhalb eines PCs. Daher sollte mehr Augenmerk auf eventuell auftretende Fehler und deren Fehlerbehandlung gelegt werden, als bei RTC6 PCIe-Karten (siehe **get_last_error**).

16.4.1 RTC6 Ethernet-Karten im Netzwerk finden und Eigenschaften abfragen

Fall 1: Die IP-Adresse der RTC6 Ethernet-Karte ist bekannt

```
card_no = eth_assign_card_ip(ip, 0); // 0: assign to first free number
Result = select_rtc(card_no);         // If Result equals CardNo: Success
…
```

Fall 2: Die IP-Adresse der RTC6 Ethernet-Karte ist nicht bekannt

```
result = eth_search_cards(eth_convert_string_to_ip("192.168.250.1"),
            eth_convert_string_to_ip("255.255.255.0"));
if (result != 0)
{
   card_no = eth_assign_card(1, 0); // take first found card, assign to first free number
   result = select_rtc(card_no);
   if (result == card_no)
   {
            ip = eth_get_ip(); // now IP address is known
            .… // do anything
   }
}
```

16.4.2 Beispiel-Code (C++): Initialisierung mit RTC6 Ethernet-Karten

```
//  Abstract
//      A console application to demonstrate how to initialize RTC6 boards
//
//  Comment
//      This application demonstrates how to properly intialize RTC6 boards.
//      It will enumerate all available PCIe boards as well as search for Ethernet boards in a given subnet.
//
//  Platform
//      Win32 - 32-bit Windows
//      x64   - 64-bit Windows
//
//  Necessary Files
//      RTC6impl.h
//      RTC6DLL.dll for Win32 or RTC6DLLx64.dll for x64
//      RTC6DLL.lib for Win32 or RTC6DLLx64.lib for x64
//      RTC6DAT.dat
//      RTC6OUT.out for PCIe boards and/or RTC6ETH.out for Ethernet boards
//      RTC6RBF.rbf
//      Cor_1to1.ct5 or any other correction file
//
//  Compiler
//      - tested with Microsoft C++ Compiler 19.00.24215.1.

#include "RTC6impl.hpp"

#include <iostream>
#include <array>
#include <string>
#include <conio.h>

using namespace std;

// RTC error codes
const UINT ERROR_NO_ERROR = 0U;
const UINT ERROR_NO_CARD = 1U;
const UINT ERROR_VERSION_MISMATCH = 256U;

// Use current working directory as path
const char* PROGRAM_FILE_PATH = nullptr;
// Use Cor_1to1.ct5 in current working directory as correction file
const char* CORRECTION_FILE_PATH = "./Cor_1to1.ct5";

// Subnet to use for Ethernet board search
const char* ETH_SEARCH_IP = "192.168.250.253";
const char* ETH_SEARCH_NETMASK = "255.255.255.0";


bool LoadProgramAndCorrectionFile(UINT card)
{
   // DAT, ETH/OUT and RBF need to be in the current working directory
   const auto loadProgram = n_load_program_file(card, PROGRAM_FILE_PATH);
   if (loadProgram != ERROR_NO_ERROR)
   {
      cout << "n_load_program_file for card " << card << " failed with error code: " << loadProgram << endl;
      return false;
   }

   // Acquire board for further access
   const auto acquire = acquire_rtc(card);
   if (acquire != card)
   {
      cout << "acquire_rtc for card " << card << " failed with error code: " << acquire << endl;
      return false;
   }
```

```
    // Load 2D correction file as table 1
    const auto loadCorrection = n_load_correction_file(card, CORRECTION_FILE_PATH, 1, 2);
    if (loadCorrection != ERROR_NO_ERROR)
    {
        cout << "n_load_correction_file for card " << card << " failed with error code: " << loadCorrection << endl;
        return false;
    }

    // select_cor_table( 1, 0 ); is done internally. Call select_cor_table, if you want a different setting.

    const auto serialNumber = n_get_serial_number(card);
    cout << "Initialized card " << card << " (SN " << serialNumber << ")" << endl;

    return true;
}

bool InitializeRTC()
{
    // Initialize DLL
    const auto initDLL = init_rtc6_dll();
    if (initDLL != ERROR_NO_ERROR)
    {
        if (initDLL & ERROR_VERSION_MISMATCH)
        {
            // Version mismatch can happen if a board has been previously used with a different software version.
            // This error can be fixed by loading the current program file on to the board.
        }
        else if (initDLL & ERROR_NO_CARD)
        {
            // The RTC6-DLL will return ERROR_NO_CARD if no PCIe board has been found.
            // We can still use Ethernet boards if available.
        }
        else
        {
            cout << "init_rtc6_dll failed with error code: " << initDLL << endl;
            return false;
        }
    }

    // Initialize PCIe boards
    const auto foundCards = rtc6_count_cards();
    for (auto card = 1; card <= foundCards; card++)
    {
        if (!LoadProgramAndCorrectionFile(card))
        {
            return false;
        }
    }
```

```
    // Search for and initialize Ethernet boards
    const auto ethSearchIP = eth_convert_string_to_ip(ETH_SEARCH_IP);
    const auto ethSearchNetmask = eth_convert_string_to_ip(ETH_SEARCH_NETMASK);
    const auto foundEthCards = eth_search_cards(ethSearchIP, ethSearchNetmask);
    if (foundEthCards > 0)
    {
       for (auto searchCard = 1; searchCard <= foundEthCards; searchCard++)
       {
          array<UINT, 16> cardInfo;
          eth_get_card_info_search(searchCard, (UINT)cardInfo.data());

          const auto serialNumber = cardInfo[1];

          // Param 0 automatically assigns the board to the next free index
          const auto card = eth_assign_card(searchCard, 0);
          if (card <= 0)
          {
             cout << "eth_assign_card for SN " << serialNumber << " failed with error code: " << card;
             return false;
          }

          if (!LoadProgramAndCorrectionFile(card))
          {
             return false;
          }
       }
    }

    // At least one PCIe or Ethernet board available
    return (foundCards > 0) || (foundEthCards > 0);
}

int main(int argc, char* argv[])
{
    if (!InitializeRTC())
    {
       return 1;
    }

    // Use boards for marking etc...

    _getch();
    return 0;
}
```

16.5 RTC6 Ethernet-Karten-Befehle und Funktionen

16.5.1 Hinweise zum Arbeiten mit IP-Adressen

Bei den RTC6-Befehlen werden alle IP-Adressen (immer IPv4, IPv6 wird nicht unterstützt) als Dezimalwert im Big-Endian-Format (“Big-Endian Byte-Reihenfolge”) angegeben.

Beispielsweise wird für die IP-Adresse “192.168.250.1” also “33204416” angegeben, siehe dazu folgende Tabelle.

Format	IP-Adresse	Hex-Wert	Dezimalwert
Little-Endian	192.168.250.1[(a)]	0×C0A8FA01	3232299521
Big-Endian	1.250.168.192	0×01FAA8C0	33204416

(a) Übliche Dezimalpunktschreibweise (engl. dotted decimal notation).

eth_convert_ip_to_string wandelt die IP-Adresse in Big-Endian Byte-Reihenfolge in die übliche Dezimalpunktschreibweise um.

eth_convert_string_to_ip wandelt die IP-Adresse von üblicher Dezimalpunktschreibweise in die Big-Endian Byte-Reihenfolge um.

16.5.2 Über die Suche nach RTC6 Ethernet-Karten

Mit **eth_search_cards** kann nach allen im Netzwerk verfügbaren RTC6 Ethernet-Karten gesucht werden. Soll die Suche auf einen bestimmten Adressbereich beschränkt sein, ist **eth_search_cards_range** zu benutzen.

Die Daten aller RTC6 Ethernet-Karten, die innerhalb eines konfigurierbaren Timeout (siehe **eth_set_search_cards_timeout**) geantwortet haben, werden in die Suchergebnisliste (s.u.) eingetragen.

Mit einer Kartensuche können RTC6 Ethernet-Karten mit unbekannter IP-Adresse im Netzwerk identifiziert werden (z.B. weil sie diese durch einen DHCP-Server dynamisch zugewiesen bekommen haben).

Die gefundenen RTC6 Ethernet-Karten werden in einer temporären Liste eingetragen, der Suchergebnisliste.

Index	**Eintrag in der** Suchergebnisliste
1	Informationen[(a)] zu RTC6 Ethernet-Karte 1
2	Informationen[(a)] zu RTC6 Ethernet-Karte 2
n	Informationen[(a)] zu RTC6 Ethernet-Karte n

(a) IP-Adresse, Seriennummer, Verbindungsstatus, etc., siehe **eth_get_card_info_search**.

Für einen angegebenen Suchergebnislisten-Index gibt **eth_get_card_info_search** die verfügbaren Informationen zu der RTC6 Ethernet-Karte zurück, **eth_get_ip_search** dagegen nur die IP-Adresse und **eth_get_serial_search** nur die Seriennummer.

Die Anzahl der gefundenen RTC6 Ethernet-Karten wird zum einen bereits durch **eth_search_cards** zurückgegeben, zum anderen aber auch jederzeit durch den Aufruf von **eth_found_cards** (ohne dass die Suche im Netzwerk wiederholt werden muss).

Mehrere aufeinanderfolgende Aufrufe von **eth_search_cards** können unterschiedliche Ergebnisse liefern, je nachdem, wie viele RTC6 Ethernet-Karten im Netzwerk verfügbar sind, und in welcher zeitlichen Reihenfolge die Antworten eintreffen.

16.5.3 Über die RTC6-Kartenverwaltung

RTC6-Karten werden über eine eindeutige Nummer angesprochen, unter der sie in der RTC6-Kartenverwaltung eingetragen sein müssen.

Die RTC6-Kartenverwaltung ist eine RTC6-DLL-interne Liste aus 255 möglichen RTC6-Karteneinträgen, siehe nachfolgende Tabelle.

Index	RTC6-Karteneintrag
1	RTC6 PCIe-Karte 1
2	RTC6 PCIe-Karte 2
3	RTC6 PCIe-Karte 3
4	"Keine Karte"
5	Informationen zur RTC6 Ethernet-Karte(a)
6	Informationen zur RTC6 Ethernet-Karte(a)
7	"Keine Karte"
…	…
42	Informationen zur RTC6 Ethernet-Karte(a)
43	"Keine Karte"
…	…
255	"Keine Karte"

(a) IP-Adresse, Seriennummer, Verbindungsstatus, etc., siehe **eth_get_card_info_search**.

Am Anfang der Liste stehen RTC6 PCIe-Karten, wenn solche im PC vorhanden sind.

RTC6 PCIe-Karten werden durch **init_rtc6_dll** automatisch gesucht und fortlaufend nummeriert. Die Nummerierung kann nicht verändert werden. **rtc6_count_cards** gibt nur die Anzahl der gefundenen RTC6 PCIe-Karten zurück.

RTC6 Ethernet-Karten müssen dagegen einzeln manuell in die RTC6-Kartenverwaltung eingetragen werden:

- Ist die IP-Adresse bekannt, kann **eth_assign_card_ip** verwendet werden, um eine Karte an einem beliebigen Index zwischen **rtc6_count_cards** + 1…255 einzutragen
- Ist die IP-Adresse nicht bekannt, kann mit **eth_assign_card** eine RTC6 Ethernet-Karte aus der Suchergebnisliste, siehe Kapitel 16.5.2 "Über die Suche nach RTC6 Ethernet-Karten", Seite 1007 an einem frei wählbaren Index eingetragen werden. An diesem Index darf jedoch noch keine RTC6 PCIe-Karte oder RTC6 Ethernet-Karte stehen

eth_max_card gibt den höchsten Index zurück, an dem eine RTC6 Ethernet-Karte in der RTC6-Kartenverwaltung eingetragen ist.

Die Gesamtanzahl der eingetragenen RTC6 Ethernet-Karten gibt dagegen **eth_count_cards** zurück.

Mit **get_card_type** kann der eingetragene Kartentyp abgefragt werden:

- 0 = "Keine Karte"
- 1 = RTC6 PCIe-Karte
- 2 = RTC6 Ethernet-Karte

16.5.4 Verbindung zur RTC6 Ethernet-Karte prüfen

Die Kommunikation über Ethernet ist im Allgemeinen unverlässlicher als eine PCIe-Bus-Verbindung innerhalb eines PCs (einfaches Beispiel: Ethernet-Kabel ist nicht gesteckt). Mit **eth_check_connection** kann einfach geprüft werden, ob die RTC6 Ethernet-Karte antwortet und somit auch die Ethernet-Verbindung noch besteht.

Das Verhalten der RTC6 Ethernet-Karte bei einem Ethernet Link Loss kann mit **eth_configure_link_loss** eingestellt werden. Im `Mode` = 0…2 wird u.a. bei einem Ethernet Link Loss (Erkennungszeit: < 100 ms) Bit #15 von **get_startstop_info** gesetzt. Erst durch den **get_startstop_info**-Aufruf wird Bit #15 wieder auf 0 gesetzt.

16.5.5 Befehlssatz für die RTC6 Ethernet-Karte

- Siehe Kapitel "Kontrollbefehle für RTC6 Ethernet-Karten", Seite 332.

16.6 Sichere Einschalt- und Ausschaltreihenfolge

Beachten Sie beim Einschalten der Komponenten eines Lasersystems die folgende Reihenfolge:

(1) Schalten Sie den Netzwerk-PC ein.

(2) Schalten Sie das Netzteil für die RTC6 Ethernet-Karte ein.

(3) Starten Sie die Steuerungs-Software.

(4) Schalten Sie die benötigten Peripheriegeräte ein.

(5) Schalten Sie die Spannungsversorgung des Scan-Systems ein.

(6) Schalten Sie den Laser ein.

Gehen Sie beim Abschalten des Lasersystems in genau umgekehrter Reihenfolge vor:

(1) Schalten Sie den Laser aus.

(2) Schalten Sie die Spannungsversorgung des Scan-Systems aus.

(3) Schalten Sie die Peripheriegeräte aus.

(4) Beenden Sie die Steuerungs-Software.

(5) Schalten Sie das Netzteil für die RTC6 Ethernet-Karte aus.

(6) Schalten Sie den Netzwerk-PC aus.

Vorsicht!

- Beim Ein- und Ausschalten des PCs können an den Ausgängen der RTC6 Ethernet-Karte kurzzeitige Pegeländerungen, also auch unvorhergesehene Änderungen der Lasersteuersignale stattfinden. Die obige Ein- und Ausschaltreihenfolge muss deswegen unbedingt eingehalten werden, sonst besteht die Gefahr, dass der Laser kurzzeitig unerwartet angeht.
- Das Scan-System muss immer nach dem PC und der Steuerungs-Software eingeschaltet oder vorher wieder ausgeschaltet werden, sonst kann es zu unvorhergesehenen Scan-Bewegungen des Scan-Systems kommen. Der Laser muss immer zuletzt eingeschaltet oder zuerst ausgeschaltet werden, sonst besteht die Gefahr, dass der Laserstrahl in eine unkontrollierte Richtung abgelenkt wird.

16.7 BIOS-ETH-Version upgraden und downgraden

	Anwendungsfall "Upgrade" (Beispiel)	Anwendungsfall "Downgrade" (Beispiel)
Vor Schritt 3	BIOS-ETH 42[a]	BIOS-ETH 43[a]
	↓	↓
Datei in Schritt 3	RTC6BIOSETH_43.out	RTC6BIOSETH_42.out
	↓	↓
Nach Schritt 3	BIOS-ETH 43[a]	BIOS-ETH 42[a]

(a) Feststellbar mit RTC6conf.exe.

Voraussetzungen

- RTC6conf.exe, Seite 35
- Dateien gemäß obiger Tabelle

Vorgehen

(1) Starten Sie RTC6conf.exe[1].

(2) Wählen Sie die gewünschte RTC6 Ethernet-Karte aus. Beachten Sie, dass die aktuelle BIOS-ETH-Version angezeigt wird.

(3) Installieren Sie die Datei (sinngemäß zur Tabelle oben) mittels der Schaltfläche **FLASH BIOS**.

(4) Führen Sie einen Hardware-Reset aus.

(1) Siehe auch RTC6conf.pdf.

16.8 Standalone-Funktionalität

Achtung!

- Standalone-Betriebsmodus-Voraussetzungen
  (1) Auf der RTC6 Ethernet-Karte ist mindestens das BIOS Version 0x26 (BIOS-ETH 26) installiert[a][b].
  (2) Das Anwenderprogramm verwendet RTC6-Software-Paket ≥ V1.7.0, also eine Kombination aus
  - ≥ DLL 618
  - ≥ ETH 618
  - ≥ RBF 623
  - ≥ DAT 603a

(a) Siehe `RTC6conf.exe`.

(b) Siehe auch Kapitel 16.7 "BIOS-ETH-Version upgraden und downgraden", Seite 1011.

- Im Gegensatz zum PC-Betrieb ist das Ziel des Standalone-Betriebsmodus, dass die RTC6 Ethernet-Karte selbstständig ohne angeschlossenen PC arbeitet. Dazu müssen u.a. die abzuarbeitenden Listenbefehle, alle Korrekturtabellen sowie die beim automatischen Booten[1] auszuführenden Kontrollbefehle im NAND-Speicher abgelegt sein.
- Dann kann die RTC6 Ethernet-Karte so konfiguriert werden, dass sie nach einem Hardware-Reset automatisch (abhängig vom mit **store_program**(`Mode`) im NAND-Speicher abgelegten Daten) bootet[1]. Danach ist sie in einem der folgenden Zustände:
  – "Normaler PC-Betriebs-Zustand"
  – "Standalone-Basis-Zustand"
  – "Standalone-Voll-Zustand"
- Der "Normaler PC-Betriebs-Zustand" wird erreicht durch:
  – BIOS < BIOS-ETH 26
  – ≥ BIOS-ETH 26 und **set_eth_boot_control**(0)
  – ≥ BIOS-ETH 26 und **store_program**(1)
  In diesen Fällen muss für den weiteren Betrieb **load_program_file** aufgerufen werden.
- Der "Standalone-Basis-Zustand" wird erreicht durch:
  – BIOS ≥ BIOS-ETH 26, **store_program**(0) und **set_eth_boot_control**(1)
  Ein **load_program_file**-Aufruf ist nicht erforderlich. Siehe auch Kapitel 16.8.3 "Boot-Image", Seite 1015. Der Ablauf für den "Standalone-Basis-Zustand" ist in Kapitel 16.8.1 ""Standalone-Basis-Zustand" herstellen", Seite 1013 beschrieben.
- Der "Standalone-Voll-Zustand" wird erreicht durch:
  – BIOS ≥ BIOS-ETH 26, **store_program**(2) und **set_eth_boot_control**(1)
  Hier ist gegenüber dem "Standalone-Basis-Zustand" zusätzlich auch das Laden von Korrekturdateien nicht mehr erforderlich. Gleiches gilt für das Laden von den Listenbefehlen, die bei **store_program**(2) gespeichert wurden. Außerdem können je nach Bedarf Kontrollbefehle zur Konfiguration[2] automatisch ausgeführt werden. Siehe auch Kapitel 16.8.3 "Boot-Image", Seite 1015. Der Ablauf für den "Standalone-Voll-Zustand" ist in Kapitel 16.8.2 ""Standalone-Voll-Zustand" herstellen", Seite 1014 beschrieben. Im "Standalone-Voll-Zustand" kann die RTC6 Ethernet-Karte nach einem /START ihre Liste selbstständig und ohne angeschlossenen PC abarbeiten.
- Der NAND-Speicher-Inhalt kann als sogenanntes "Boot-Image" mit **read_image_eth** als Binärdatei auf den PC geschrieben und mit **write_image_eth** auf beliebig viele RTC6 Ethernet-Karten kopiert werden, siehe Kapitel 16.8.3 "Boot-Image", Seite 1015.
- Details zum automatischen Booten finden Sie in Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018.
- RTC4 *SCAN*alone-Karten-Benutzer müssen den Sicherheitshinweis zur Encoderzähler-Richtung beachten, siehe Achtung!, Seite 59.

(1) "Automatisches Booten" bedeutet hier, dass außer dem eigentlichen Booten noch zusätzlich Daten aus dem NAND-Speicher ausgelesen werden.

(2) Siehe Kapitel 16.8.4 "Für das automatische Booten zulässige Kontrollbefehle", Seite 1016.

16.8.1 "Standalone-Basis-Zustand" herstellen

Gehen Sie wie folgt vor, um die Karte in den "Standalone-Basis-Zustand" zu versetzen:

(1) Stellen Sie sicher, dass die Standalone-Betriebsmodus-Voraussetzungen, Seite 1012 erfüllt sind.

(2) Rufen Sie **load_program_file** unter Angabe der relevanten Dateien aus dem RTC6-Software-Paket 1.19.1 auf.

(3) Rufen Sie **store_program**(`Mode` = 0) auf. Die Daten für den "Standalone-Basis-Zustand", siehe Seite 893, werden im NAND-Speicher abgelegt.

(4) Rufen Sie **set_eth_boot_control**(`Ctrl` = 1) auf. Somit bootet die Karte nach einem Hardware-Reset automatisch(1)(2).

(5) Führen Sie einen Hardware-Reset aus.
- Die RTC6 Ethernet-Karte bootet automatisch(1)
- Ist nun für den PC-Betrieb bereit
- Nachfolgend müssen Sie **load_program_file** *nicht* aufrufen

Hinweise

- Im Anwenderprogramm müssen die RTC6-Dateien nicht mehr vorgehalten werden. Eine versehentliche falsche RTC6-DLL-Version wird automatisch erkannt, da eine falsche RTC6-DLL zu einem **get_last_error**-Returncode `RTC6_VERSION_MISMATCH` führt (ein **load_program_file** dagegen nicht, solange die RTC6-Dateien mit der RTC6-DLL kompatibel sind).

(1) "Automatisches Booten" bedeutet hier, dass außer dem eigentlichen Booten noch zusätzlich Daten aus dem NAND-Speicher ausgelesen werden.

(2) Dies ist auch die Voreinstellung bei "neuen" (= **set_eth_boot_control**(`Ctrl` = 0) wurde noch nie ausgeführt) RTC6 Ethernet-Karten.

16.8.2 “Standalone-Voll-Zustand” herstellen

Gehen Sie wie folgt vor, um die RTC6 Ethernet-Karte in den “Standalone-Voll-Zustand” zu versetzen.

(1) Voraussetzung: Die Prozedur in Kapitel 16.8.1 ““Standalone-Basis-Zustand” herstellen”, Seite 1013 ist ausgeführt.

(2) Stellen Sie sicher, dass die RTC6 Ethernet-Karte im PC-Betrieb ordnungsgemäß arbeitet.

(3) Erarbeiten Sie sich eine passende Abfolge von Kontrollbefehlen für das automatische Booten(1).
- Die zulässigen Kontrollbefehle finden Sie in Kapitel 16.8.4 ”Für das automatische Booten zulässige Kontrollbefehle”, Seite 1016, getrennt nach Boot-Phase 1 und Boot-Phase 2.
- Rufen Sie **eth_boot_dcmd** vor jedem einzelnen dieser Befehle auf.
- Achten Sie auf die richtige Reihenfolge, beispielsweise
  - sollte **config_list** vor dem Laden der Listenbefehle aufgerufen werden
  - wenn **set_scanahead_params** verwendet wird, muss bereits ein SCANahead-System angeschlossen und eingeschaltet sein
  - wenn eine “Automatische Laser-Steuerung” verwendet wird, darf diese erst nach **set_scanahead_params** aktiviert werden

(4) Testen Sie das Ergebnis aus Schritt 3: Führen Sie es im PC-Betrieb genauso aus, wie es später im Standalone-Betriebsmodus ausgeführt werden soll.

(5) Rufen Sie **load_program_file** auf, um die RTC6 Ethernet-Karte zu initialisieren.

(6) Laden Sie alle erforderlichen Kontrollbefehle, Listenbefehle und Korrekturdateien.

(7) Notfall-Vorsorge, falls beim Speichern ein Fehler auftritt: Rufen Sie **set_eth_boot_control**(0) auf. Somit ist das automatische Booten verhindert.

(8) Führen Sie einen der folgenden Schritte aus:
- Nur für Remote-Interface-Modus –
  Rufen Sie **store_program**(`Mode` = 3) auf.
- Anderenfalls –
  Rufen Sie **store_program**(`Mode` = 2) auf.

(9) Wenn ein Fehler aufgetreten ist:
Wiederholen Sie Schritt 8.

(10) Ist der Vorgang fehlerfrei abgelaufen:
Rufen Sie **set_eth_boot_control**(`Ctrl` = 1) auf. Somit ist das automatische Booten aktiviert.

(11) Führen Sie einen Hardware-Reset aus.
- Die RTC6 Ethernet-Karte ist nun im “Standalone-Voll-Zustand”
- Siehe auch Kapitel 16.8.5 ”Automatisches Booten – Ablauf im Detail”, Seite 1018

Hinweise

- Im “Standalone-Voll-Zustand” ist die RTC6 Ethernet-Karte bereits mit den Kontrollbefehlen für das automatische Booten(2) konfiguriert und mit allen Korrekturtabellen versehen. Falls Listenbefehle gespeichert wurden, sind diese bereit, nach einem /START automatisch ausgeführt zu werden.

(1) Siehe Fußnote, Seite 1012.

(2) Siehe Kapitel 16.8.4 ”Für das automatische Booten zulässige Kontrollbefehle”, Seite 1016.

16.8.3 Boot-Image

- Voraussetzungen: siehe Standalone-Betriebsmodus-Voraussetzungen, Seite 1012

Die Daten im NAND-Speicher können als "Boot-Image" auf den PC geschrieben werden und von dort aus auf beliebige RTC6 Ethernet-Karten (z.B. mehrerer Produktionsmaschinen) kopiert werden:

- **read_image_eth**,
  siehe Boot-Image auf dem PC erstellen
- **write_image_eth**,
  siehe Boot-Image auf die Karte(n) kopieren

Die Eigenschaften der Boot-Images zeigt Tabelle 7 (RTC6-Software-Paket V1.7.0, ungefähre Werte):

Tabelle 7: Eigenschaften von Boot-Images

Boot-Image	Datei-größe	Lese-dauer[a]	Schreib-dauer[b]
für "Standalone-Basis-Zustand"	2 MB	einige Sekunden	einige Sekunden
für "Standalone-Voll-Zustand"	160 MB … 264 MB	bis 2 Minuten	bis 2 Minuten

(a) Mit **read_image_eth**.
(b) Mit **write_image_eth**.

Boot-Image auf dem PC erstellen

- Voraussetzungen: siehe Standalone-Betriebsmodus-Voraussetzungen, Seite 1012

read_image_eth(`Name`) liest den Inhalt des NAND-Speichers aus und schreibt ihn binär in die Datei "`Name`" auf dem PC. Das Anwenderprogramm muss Schreibberechtigung für diese Datei haben. Zur Lesedauer siehe Tabelle 7.
Siehe auch Abschnitt "Vorgehen nach einem vorzeitigen Abbruch der Übertragung", Seite 1015.

Boot-Image auf die Karte(n) kopieren

- Voraussetzungen: siehe Standalone-Betriebsmodus-Voraussetzungen, Seite 1012

write_image_eth(`Name`) liest die Datei "`Name`" ein und schreibt ihren Inhalt in den NAND-Speicher. Das Anwenderprogramm muss Leseberechtigung für diese Datei haben. Zur Schreibdauer siehe Tabelle 7.

(1) Rufen Sie **set_eth_boot_control**(0) auf.
Somit bootet die Karte nach einem Hardware-Reset *nicht* automatisch (empfohlen, da das automatische Booten möglicherweise fehlerhaft sein könnte).

(2) Rufen Sie **write_image_eth**(`Name`) auf und unterbrechen Sie den Vorgang nicht!
Anderenfalls beachten Sie den Abschnitt "Vorgehen nach einem vorzeitigen Abbruch der Übertragung", Seite 1015.

(3) Ist der Vorgang fehlerfrei abgelaufen, rufen Sie **set_eth_boot_control**(1) auf. Anderenfalls wiederholen Sie Schritt 2.

(4) Führen Sie einen Hardware-Reset aus.
Vorausgesetzt es sind die gleichen Optionen freigeschaltet: Diese RTC6 Ethernet-Karte bootet genau so, wie die RTC6 Ethernet-Karte, von der das Boot-Image stammt.

Vorgehen nach einem vorzeitigen Abbruch der Übertragung

Bei einem vorzeitigen Abbruch der Übertragung, z.B. wegen einer Ethernet-Verbindungsunterbrechung, bleibt die RTC6 Ethernet-Karte wahrscheinlich in einem INTERNAL-BUSY-Listenausführungsstatus. Der abgebrochene Vorgang wird nicht fortgesetzt, selbst wenn die Verbindung wieder hergestellt ist:

(1) Rufen Sie entweder **stop_execution** auf oder führen Sie einen /STOP aus, um die Karte aus diesem INTERNAL-BUSY-Listenausführungsstatus zu lösen.

(2) Rufen Sie **set_eth_boot_control**(0) auf.
Somit bootet die Karte nach einem Hardware-Reset *nicht* automatisch.

(3) Führen Sie einen Hardware-Reset aus.

(4) Rufen Sie **load_program_file** auf.

(5) Wenn Sie ein Boot-Image
- auf dem PC erstellen:
  Rufen Sie **read_image_eth**(`Name`) erneut auf.
- auf die Karte kopieren:
  Rufen Sie **write_image_eth**(`Name`) auf.

16.8.4 Für das automatische Booten zulässige Kontrollbefehle

- In jeder Boot-Phase sind nur bestimmte Kontrollbefehle zulässig, siehe Tabelle 8 und Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018:
  - Boot-Phase 1
  - Boot-Phase 2
- Sie erfüllen ein oder mehrere der folgenden Kriterien:
  - Erfordert keine Kommunikation mit dem PC
  - Gibt keine Rückantwort an den PC
  - Es gibt keinen entsprechenden Listenbefehl
- Ausnahmen sind z.B.
  - **set_free_variable** siehe Hinweise dort
  - **set_auto_laser_control** oder **set_scanahead_params**
    - Sie überprüfen zum Zeitpunkt des Aufrufs die angeschlossenen Scan-Systeme
    - Sie speichern nur im Erfolgsfall ausschließlich die endgültig einzustellenden Parameter
  - **set_jump_mode** setzt beim automatischen Booten nur die Tuning-Nummern `VA1`…`JB2` und überprüft nicht, ob diese zum Scan-System passen. Die Tabelle mit den Jump-Delay-Werten `[JumpTable<No>]`, siehe Seite 231 und **load_jump_table_offset**, muss zuvor über **create_dat_file** gespeichert worden sein.
- Beim (späteren) automatischen Booten werden die angeschlossenen Scan-Systeme *nicht* mehr überprüft!
- Boot-Phase 2-Kontrollbefehle sind Kontrollbefehle, die:
  - unbedingt nach einem Boot-Phase 1-Kontrollbefehl ausgeführt werden müssen
  - eine funktionierende Peripherie voraussetzen, z.B. ein i*DRIVE*-Scan-System(1) bei "Automatische Laser-Steuerung".
    Wichtig: Benutzer müssen selber sicherstellen, dass die Peripherie zur Laufzeit tatsächlich funktionsfähig ist

(1) Siehe Glossareintrag auf Seite 28.

Tabelle 8: Für das automatische Booten zulässige Kontrollbefehle (alphabetisch)

Boot-Phase 1

bounce_supp
config_laser_signals
config_list
eth_boot_timeout
eth_set_remote_tgm_format
home_position
home_position_xyz
mcbsp_init
set_control_mode
set_controlpreview_compensation_ctrl
set_extstartpos
set_fly_tracking_error
set_free_variable
set_jump_mode
set_laser_control
set_laser_mode
set_matrix(HeadNo = 4)
set_max_counts
set_mcbsp_freq
set_mcbsp_out_ptr
set_offset_xyz(HeadNo = 4)
set_pulse_picking_length
set_rot_center
set_scanahead_params
set_scanahead_speed_control
set_timelag_compensation
simulate_encoder
uart_config

Boot-Phase 2

control_command
init_fly_2d
select_cor_table
set_auto_laser_control
set_dsp_mode
set_hi
start_loop

16.8.5 Automatisches Booten – Ablauf im Detail

(1) Nach einem Hardware-Reset versucht die RTC6 Ethernet-Karte, die Daten für das automatische Booten aus dem NAND-Speicher zu lesen.(1)
Während dieser Zeit blinkt die gelbe LED gedimmt mit ca. 1 Hz bis zum Ende der Initialisierung.

- Fall 1: Es sind *keine* Daten vorhanden (nach **store_program**(`Mode` = 1)) oder es wurde **set_eth_boot_control**(0) ausgeführt.
  Die Initialisierung wird ohne diese Daten zu Ende geführt. Danach ist die RTC6 Ethernet-Karte im "Normaler PC-Betriebs-Zustand", Seite 1012.
- Fall 2: Es sind nur Daten für den "Standalone-Basis-Zustand", siehe Seite 893, (nach **store_program**(`Mode` = 0)) vorhanden.
  Die RTC6 Ethernet-Karte wird (wie mit **load_program_file**) initialisiert. Danach ist sie unmittelbar für den PC-Betrieb bereit = "Standalone-Basis-Zustand".
  Siehe auch Kapitel 16.8.1 ""Standalone-Basis-Zustand" herstellen", Seite 1013.
- Fall 3: Es sind auch Daten für den "Standalone-Voll-Zustand", siehe Seite 893, vorhanden (nach **store_program**(`Mode` = 2)).
  Dann wird Schritt 2 ausgeführt.

(2) Die RTC6 Ethernet-Karte

- schaltet den BUSY-Pin ein
- liest die "restlichen" Daten für das automatische Booten in den "Standalone-Voll-Zustand", siehe Seite 893) aus dem NAND-Speicher.(1)
- Die nachfolgenden Boot-Schritte 3...6 sind zweiphasig.

(3) In Boot-Phase 1 (Dauer: wenige Sekunden):

- Die Korrekturdatei(en) werden ausgelesen
- Die Boot-Phase 1-Kontrollbefehle werden ausgeführt (in der Reihenfolge, in der sie gespeichert worden sind). Sind solche Befehle mehrfach gespeichert, werden sie entsprechend oft ausgeführt, wobei neuere Daten ältere überschreiben.

(4) Nach Boot-Phase 1:(1)

- Die Initialisierung bleibt stehen
- Der BUSY-Pin wird ausgeschaltet
- Schalten Sie den Laser noch nicht ein!
- Schalten Sie Scan-Systeme und andere Peripherie-Geräte ein.
- Stellen Sie sicher, dass die Peripherie-Geräte betriebsbereit sind. Lösen Sie erst dann mit einem /START Boot-Phase 2 aus
- Sind Ihre Peripherie-Geräte immer nach einer bestimmten Zeit betriebsbereit, können Sie mit dem Boot-Phase 1-Kontrollbefehl **eth_boot_timeout** eine Wartezeit (`TimeOut`) angeben. Nach deren Ablauf wird die Initialisierung automatisch (= ohne expliziten /START) fortgesetzt. Bei einem /START innerhalb der Wartezeit beginnt Boot-Phase 2 sofort. Außerhalb der Wartezeit wird ein /START ignoriert, solange die Boot-Phase 2 noch läuft (siehe BUSY-Pin!). Der /START wird (wie mit **set_control_mode**(`Bit #0` = 1)) automatisch freigegeben. Weitere /STARTs nach einem /STOP muss der Benutzer selber mit **set_control_mode**(`Bit #3` = 1) freigeben.

(1) **eth_get_standalone_status**(&`Status` == 1)

(5) In Boot-Phase 2:[1]
- Der BUSY-Pin wird eingeschaltet
- Der RTC6-Listenspeicher wird ausgelesen (Dauer: etwa 1 Minute)
- Die Boot-Phase 2-Kontrollbefehle werden ausgeführt.
- Die Initialisierung bleibt stehen
- Der BUSY-Pin wird ausgeschaltet

(6) Nach Boot-Phase 2:
- Ist ein Fehler aufgetreten[2] (ein weiterer Betrieb der Karte ist eventuell nicht möglich), blinkt die LED weiter mit etwa 4 Hz
- Wenn die Initialisierung erfolgreich abgeschlossen wurde[3], wird das gedimmte LED-Blinken beendet. Die LED kehrt in den Normal-Zustand zurück, siehe Kapitel 16.2.7 "ETH-Buchse", Seite 989
- Den Ziel-Zustand der RTC6 Ethernet-Karte zeigt **eth_get_standalone_status**(&Mode =1…2).
- An dieser Stelle wäre ein geeigneter Zeitpunkt, auch den Laser einzuschalten (beachten Sie die Hinweise zur Lasersicherheit in Kapitel 3.2 "Lasersicherheit", Seite 65!), sowie weitere Peripherie.
- Die RTC6 Ethernet-Karte wartet endlos auf einen /START, der die Abarbeitung der Liste startet
- Die RTC6 Ethernet-Karte ist auch für den PC-Betrieb bereit.

(1) **eth_get_standalone_status**(&Status == 1)

(2) **eth_get_standalone_status**(&Error = 1)

(3) **eth_get_standalone_status**(&Status =0)

16.9 “High-Performance-Modus”

Bei RTC6 Ethernet-Karten schaltet **eth_set_high_performance_mode** ein (aus) den:

- “High-Performance-Modus”

Der “High-Performance-Modus” wirkt nur bei Listenbefehlen. Ist er eingeschaltet (`Mode` = 1), wartet die RTC6-DLL nicht mehr nach jedem Telegramm auf eine Antwort der RTC6 Ethernet-Karte.

Stattdessen bestätigt die RTC6-DLL nun mehrere Telegramme auf einmal. Daher können mehr Listenbefehle pro Zeit in den RTC6-Listenspeicher geschrieben werden
(= “höhere Listen-Download Performance”).

Die RTC6-DLL wartet immer, bis die RTC6 Ethernet-Karte alle vorherigen Listenbefehle bestätigt hat bei:

- Kontrollbefehl-Aufrufen
- **set_end_of_list**
- **list_return**

Um sicherzustellen, dass alle Listenbefehle in den RTC6-Listenspeicher geschrieben werden, sollte der Listenbefehls-Block immer mit **set_end_of_list** oder **list_return** abgeschlossen werden, wenn:

- Eine Liste gestartet wird mit
  - /Slave-START
  - /START
- Auf einen Listenbefehls-Block nicht direkt ein Kontrollbefehl folgt

Allerdings wird eine nennenswert höhere Rate nur erreicht, wenn Sie:

- Entsprechend größere Listenbefehls-Blöcke schreiben
- Vermeiden, dass durch Aufruf eines Kontrollbefehls die Listenbefehls-Blöcke vorzeitig bestätigt werden

16.10Remote-Interface-Modus

Der Remote-Interface-Modus dient zum Steuern von RTC6 Ethernet-Karten ohne RTC6-DLL(1).

Dazu werden Befehls-Telegramme über das Netzwerk (UDP) an die RTC6 Ethernet-Karte gesendet.

Im RTC6-Software-Paket ist ein C++-Remote-Interface-Wrapper enthalten, der das Anwenden vereinfacht.

Hinweise

- BIOS-ETH auf der RTC6 Ethernet-Karte und RTC6-Dateien müssen aus der selben RTC6-Software-Paket Version, Seite 1081 stammen. Anderenfalls kann ein unerwartetes Verhalten auftreten, z.B. ist das Aktivieren des Remote-Interface-Modus ist nicht möglich.
  - Wenn Sie den Remote-Interface-Modus bereits verwenden und ein neueres RTC6-Software-Paket einsetzen wollen: Vergessen Sie nicht, das BIOS-ETH auf Ihrer RTC6 Ethernet-Karte entsprechend zu aktualisieren. Verwenden Sie dazu `RTC6BIOSETH_*.out` aus dem selben RTC6-Software-Paket.

In diesem Kapitel:

- Voraussetzungen für den Remote-Interface-Modus, Seite 1022
- Einschränkungen im Remote-Interface-Modus, Seite 1022
- Remote-Interface-Modus aktivieren, Seite 1023
- Remote-Interface-Modus deaktivieren – vorübergehend, Seite 1023
- Remote-Interface-Modus deaktivieren – dauerhaft, Seite 1024
- Telegramm-Format RAW, Seite 1025
- Fehlercodes für `LastError`, die nur im Remote-Interface-Modus vorkommen, Seite 1028
- Befehls-Telegramme senden und Antwort-Telegramme empfangen, Seite 1030
- Remote-Befehle, Seite 1030
- Nicht als Remote-Befehl verfügbare RTC6-Befehle, Seite 1054
- Hinweise zur Softwareentwicklung, Seite 1060
- Über den C++-Remote-Interface-Wrapper, Seite 1061

(1) Oder ohne Linux Shared Library, siehe Unterverzeichnis Linux, Seite 34.

16.10.1 Voraussetzungen für den Remote-Interface-Modus

- Neueste RTC6-Software-Paket Version (empfohlen[1])
- RTC6 Ethernet-Karte mit BIOS-ETH aus der neuesten RTC6-Software-Paket Version (empfohlen[1][2])
- Die RTC6 Ethernet-Karte ist konfiguriert für den Standalone-Betriebsmodus im "Standalone-Voll-Zustand", siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.
- In der Boot-Phase 1 wird **eth_set_remote_tgm_format** aufgerufen.

16.10.2 Einschränkungen im Remote-Interface-Modus

- Eine herkömmliche RTC6-DLL-basierte Verbindung zur RTC6 Ethernet-Karte kann nicht hergestellt werden. RTC6-Befehle, die eine Verbindung erfordern, liefern dann **get_error** `RTC6_ACCESS_DENIED` oder **eth_get_error** not_acquired.
- RTC6 Ethernet-Karten können lediglich weiterhin gesucht werden:
  - Über die RTC6-DLL mit **eth_search_cards**
- Die meisten RTC6-Befehle sind in einer entsprechenden Funktionalität auch als Remote-Befehle verfügbar, siehe:
  - Kapitel 16.10.9 "Remote-Befehle", Seite 1030
  - `rtc6_rif_wrapper.h`
- Einige RTC6-Befehle sind *nicht* in einer entsprechenden Funktionalität auch als Remote-Befehle verfügbar, siehe:
  - Kapitel 16.10.10 "Nicht als Remote-Befehl verfügbare RTC6-Befehle", Seite 1054

(1) Der Remote-Interface-Modus war erstmals mit RTC6-Software-Paket ≥ V1.16.1 zusammen mit BIOS-ETH 39 verfügbar und wird seitdem weiterentwickelt, siehe Kapitel 22 "Anhang G: Änderungsindex", Seite 1142.

(2) Siehe auch Kapitel 16.7 "BIOS-ETH-Version upgraden und downgraden", Seite 1011.

16.10.3 Remote-Interface-Modus aktivieren

(1) Konfigurieren Sie die RTC6 Ethernet-Karte für den Standalone-Betriebsmodus im "Standalone-Voll-Zustand", siehe Kapitel 16.8.2 ""Standalone-Voll-Zustand" herstellen", Seite 1014. Verwenden Sie **store_program**(`Mode` = 3) in Schritt 8.

(2) Stellen Sie sicher, dass **eth_set_remote_tgm_format** in der Boot-Phase 1 aufgerufen wird, siehe Schritt 3 in Kapitel 16.8.2 ""Standalone-Voll-Zustand" herstellen", Seite 1014.

(3) Nach jedem Hardware-Reset bootet die RTC6 Ethernet-Karte dann im "Standalone-Voll-Zustand" und wird anschließend automatisch in den Remote-Interface-Modus versetzt.

Hinweise

- Siehe auch Hinweise, Seite 1021.

16.10.4 Remote-Interface-Modus deaktivieren – vorübergehend

Der Remote-Interface-Modus kann vorübergehend – bis zum nächsten Hardware-Reset – deaktiviert werden.

In diesem Zustand kann die RTC6 Ethernet-Karte über die herkömmliche RTC6-DLL-basierte(1) Verbindung gesteuert werden.

- Führen Sie einen der folgenden Schritte aus:
  - Senden Sie ein Befehls-Telegramm gemäß folgender Spezifikation:
    - Der *erste und einzige* `uint32_t`-Wert in TGM_PL_CMD_RAW ist `0x0D15AB1E`(1)
  - Rufen Sie die **Wrapper-Funktion disable_remote_interface** auf
  - Klicken Sie die entsprechende Schaltfläche in `RTC6conf.exe`
  - Unterbrechen Sie das Booten in den "Standalone-Voll-Zustand". Rufen Sie während dieser Zeit (etwa 10 s bei **store_program**(`Mode` = 3) in Schritt 8; anderenfalls ca. 60 s) **load_program_file** auf. Dadurch wird das Booten abgebrochen. Anschließend kann eine herkömmliche RTC6-DLL-basierte Verbindung zur RTC6 Ethernet-Karte hergestellt werden.

(1) Ist der Remote-Interface-Modus deaktiviert: Der erste `uint32_t`-Wert der TGM_PL_ANSW_RAW ist `0xD15AB1ED`.

16.10.5 Remote-Interface-Modus deaktivieren – dauerhaft

Die RTC6 Ethernet-Karte wird in den "Standalone-Voll-Zustand" zurückversetzt, wenn der Remote-Interface-Modus dauerhaft deaktiviert wird.

In diesem Zustand kann die RTC6 Ethernet-Karte über die herkömmliche RTC6-DLL-basierte(1) Verbindung gesteuert werden.

(1) Deaktivieren Sie den Remote-Interface-Modus vorübergehend wie beschrieben in Kapitel 16.10.4 "Remote-Interface-Modus deaktivieren – vorübergehend", Seite 1023.

(2) Führen Sie einen der folgenden Schritte aus:
- Rufen Sie **eth_set_remote_tgm_format**(0) in Boot-Phase 1 auf (oder entfernen Sie **eth_set_remote_tgm_format**(1) aus Boot-Phase 1)
- Stellen Sie den "Standalone-Basis-Zustand" her, siehe Kapitel 16.8.1 ""Standalone-Basis-Zustand" herstellen", Seite 1013 (`store_program`(0))
- Schalten Sie überhaupt den Standalone-Betriebsmodus ab, siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012 (`store_program`(1) oder `set_eth_boot_control`(0)

16.10.6Telegramm-Format RAW(1)

In diesem Kapitel:

- Befehls-Telegramm – Struktur, Seite 1025
- Antwort-Telegramm – Struktur, Seite 1025
- TGM_HEADER, Seite 1026
- TGM_PL_CMD_RAW, Seite 1027
- TGM_PL_ANSW_RAW, Seite 1027

Befehls-Telegramm – Struktur

(Telegramm *an* die RTC6 Ethernet-Karte)

Befehls-Telegramm	
TGM_HEADER	Kopf-Daten
TGM_PL_CMD_RAW	Nutz-Daten • 1 Remote-Kontrollbefehl *ODER* • 1…n Remote-Listenbefehle(a)
TGM_MAX_SIZE(b)	≤ 1.400 Byte

(a) Die entsprechenden RTC6-Befehle werden in dieser Reihenfolge im RTC6-Listenspeicher gespeichert.

(b) In `telegrams.h`. Erlaubte Länge von [TGM_HEADER + TGM_PL_CMD_RAW].

Antwort-Telegramm – Struktur

(Telegramm *von* der RTC6 Ethernet-Karte)

Antwort-Telegramm	
TGM_HEADER	Kopf-Daten
TGM_PL_ANSW_RAW	Nutz-Daten
TGM_MAX_SIZE(a)	≤ 1.400 Byte

(a) In `telegrams.h`. Erlaubte Länge von [TGM_HEADER + TGM_PL_CMD_RAW].

Hinweise

- Auf jedes empfangene Befehls-Telegramm reagiert die RTC6 Ethernet-Karte mit genau 1 Antwort-Telegramm.

(1) enum TGM_FORMAT, Aufzählungskonstante RAW in `telegrams.h`.

TGM_HEADER

(Kopf-Daten des Befehls-Telegramms oder Antwort-Telegramms)

- TGM_HEADER ist Strukturelement von:
  - Befehls-Telegramm
  - Antwort-Telegramm

Datentyp (Forts.)	Name (Forts.)	Beschreibung (Forts.)
`uint32_t` (a)	`length`	Länge von TGM_PL_CMD_RAW oder TGM_PL_ANSW_RAW. In Bytes.
`uint32_t`	`version`	Versionsnummer. Aktuell: 0x01000000 (TGM_VERSION in `telegrams.h`). Muss zur Version passen, die im BIOS gespeichert ist.
`uint32_t`	`seqnum`	Sequenznummer. Zur Flusskontrolle. Antwort-Telegramme enthalten immer den gleichen `seqnum`-Wert. Die RTC6 Ethernet-Karte führt nur Befehls-Telegramme mit `seqnum` > zuletzt empfangene `seqnum` aus. Wenn `seqnum` = letzte empfangene `seqnum` ist, dann wird nur das letzte Antwort-Telegramm erneut gesendet. Beachten Sie Seite 1060.
`uint32_t`	`type`	Telegramm-Typ. • 0 TYPE_COMMAND = Befehls-Telegramm • 1 TYPE_ANSWER = Antwort-Telegramm
`uint32_t`	`format`	Telegramm-Format. Muss mit dem über **eth_set_remote_tgm_format** eingestellten Wert übereinstimmen.

(a) Synonym: `unsigned int`. 32-Bit-Wert ohne Vorzeichen: $[0…+(23^2–1)]$.

TGM_PL_CMD_RAW

(Befehls-Telegramm-Nutz-Daten im Telegramm-Format RAW)

- TGM_PL_CMD_RAW definiert das Format der Befehls-Telegramm-Nutz-Daten
- Beachten Sie die weiteren Informationen in Befehls-Telegramm – Struktur, Seite 1025. ff.

Datentyp	Name	Beschreibung
uint32_t	1. ID	ID[(a)] des 1. Remote-Befehls. *ODER*: 0x12345678, siehe Seite 1060.
n × uint32_t	Parameter 1. ID	Befehls-Telegramm-Parameter von 1. ID[(b)].
uint32_t	2. ID	…
n × uint32_t	Parameter 2. ID	…
…	…	…

(a) IDs von Remote-Kontrollbefehlen, siehe Seite 1031 ff.
IDs von Remote-Listenbefehlen, siehe Seite 1042 ff.

(b) Befehls-Telegramm-Parameter von Remote-Kontrollbefehlen, siehe Seite 1031 ff.
Befehls-Telegramm-Parameter von Remote-Listenbefehlen, siehe Seite 1042 ff.

TGM_PL_ANSW_RAW

(Antwort-Telegramm-Nutz-Daten im Telegramm-Format RAW)

- TGM_PL_ANSW_RAW definiert das Format der Antwort-Telegramm-Nutz-Daten

Datentyp	Name	Beschreibung
uint32_t	LastError 1. ID	LastError von 1. ID. Siehe **get_last_error**.
uint32_t	1. ID	ID[(a)] des 1. Remote-Befehls.
n × uint32_t	Antwort 1. ID	Antwort-Telegramm-Parameter von 1. ID[(b)].
uint32_t	LastError 2. ID	…
uint32_t	2. ID	…
n × uint32_t	Antwort 2. ID	…
…	…	…

(a) IDs von Remote-Kontrollbefehlen, siehe Seite 1031 ff.
IDs von Remote-Listenbefehlen, siehe Seite 1042 ff.

(b) Antwort-Telegramm-Parameter von Remote-Kontrollbefehlen siehe ab Seite 1031.
Antwort-Telegramm-Parameter von Remote-Listenbefehlen siehe ab Seite 1042.

16.10.7 Fehlercodes für LastError, die nur im Remote-Interface-Modus vorkommen

Nur im Remote-Interface-Modus wird **get_error** Bit #31 gesetzt (ungültiges Befehls-Telegramm).

Ist der Befehls-Telegramm-TGM_HEADER fehlerhaft, wird die 1. ID nicht ausgewertet. `LastError 1. ID` ist dann der einzige `uint32_t`-Wert in TGM_PL_ANSW_RAW.

Es können mehrere Fehler-Bits gleichzeitig gesetzt sein. Sie können abgefragt werden mit:

- Remote-Kontrollbefehl
  `148 R_DC_GET_REMOTE_ERROR`
- **Wrapper-Funktion get_remote_error**

Fehler-Bit	Beschreibung	Entsprechende Aufzählungskonstante in enum TGM_ERROR[a]
Bit #0 = 0 … Bit #6 = 0	Kein Fehler.	`ERROR_NO_ERROR`
Bit #0 = 1	Ungültige Befehls-Telegramm-Größe. Muss sein: > TGM_HEADER[a] oder ≤ TGM_MAX_SIZE[a].	`ERROR_TGM_SIZE`
Bit #1 = 1	`version`[a]-Wert im Befehls-Telegramm-Header ist nicht kompatibel mit BIOS. Stellen Sie sicher, dass C++-Remote-Interface-Wrapper und BIOS aus dem selben RTC6-Software-Paket stammen.	`ERROR_HEADER_VERSION`
Bit #2 = 1	Ungültiger `length`[a]-Wert. Die empfangene Telegramm-Größe stimmt nicht mit der TGM_PL_CMD_RAW-Größe überein.	`ERROR_HEADER_LENGTH`
Bit #3 = 1	Ungültiger `type`[a]-Wert. Muss sein: 0 bei einem Befehls-Telegramm.	`ERROR_HEADER_TYPE`
Bit #4 = 1	Ungültiger `format`[a]-Wert. Muss sein: Selber Wert, der vorher mit **eth_set_remote_tgm_format** eingestellt wurde.	`ERROR_HEADER_FORMAT`
Bit #5 = 1	Ungültige Remote-Befehls-ID. Muss sein: Eine gültige Remote-Befehls-ID aus enum REMOTE_ID[a].	`ERROR_PAYLOAD_UNKNOWN_CMD_ID`
Bit #6 = 1	Ungültige Nutz-Daten. Muss sein: 1 Remote-Kontrollbefehl *oder* 1…n Remote-Listenbefehl(e).	`ERROR_PAYLOAD_INVALID`

(a) Siehe `telegrams.h`.

Beispiele

(1) TGM_PL_CMD_RAW enthält mehrere Remote-Kontrollbefehle und ist deshalb ungültig

– Aufruf

```
Payload (R_DC_GOTO_XYZ, …,
R_DC_SET_MARK_SPEED, …)
```

– Antwort

(LastError Bit #31, R_DC_SET_MARK_SPEED)

– Erläuterung

- 76 R_DC_GOTO_XYZ wird nicht ausgeführt
- 72 R_DC_SET_MARK_SPEED wird nicht ausgeführt
- 148 R_DC_GET_REMOTE_ERROR liefert Bit #6 = 1 zurück

(2) TGM_PL_CMD_RAW enthält sowohl Remote-Listenbefehle als auch Remote-Kontrollbefehle und ist deshalb ungültig

– Aufruf

```
Payload (R_LC_JUMP_XY_ABS, …,
R_LC_END_OF_LIST, R_DC_EXECUTE_LIST_POS, …)
```

– Antwort

(LastError Bit #31, R_DC_EXECUTE_LIST_POS)

– Erläuterung

- Die entsprechenden RTC6-Befehle werden nicht im RTC6-Listenspeicher gespeichert
- 15 R_DC_EXECUTE_LIST_POS wird nicht ausgeführt
- 148 R_DC_GET_REMOTE_ERROR liefert Bit #6 = 1 zurück

16.10.8 Befehls-Telegramme senden und Antwort-Telegramme empfangen

- Befehls-Telegramme müssen via UDP an die RTC6 Ethernet-Karte gesendet werden. Der UPD-Port ist der gleiche wie der für die herkömmliche Übertragung(1).
- Die konkrete Implementierung hängt vom Betriebssystem und der Programmiersprache (siehe zugehörige Socket-APIs) ab und kann deswegen hier nicht beschrieben werden. Im Verzeichnis Demo Files finden Sie ein Code-Beispiel (C++) für die Winsock-API und Linux Socket-API.
- Bei UDP werden verloren gegangene Pakete nicht automatisch erkannt. Zudem ist die Übertragungsreihenfolge nicht garantiert. Deswegen sollten Sie nach jedem Befehls-Telegramm explizit auf das entsprechende Antwort-Telegramm warten und dieses prüfen.
- Fügen Sie so viele Remote-Listenbefehle wie möglich in ein Befehls-Telegramm ein. So verringern Sie die Gesamtanzahl an Telegrammen.
- Sie müssen Prüfungen zur Flusskontrolle (die normalerweise von der RTC6-DLL(2) durchgeführt werden) manuell implementieren. Beispiel: Der RTC6-Listenspeicher soll befüllt werden und das Antwort-Telegramm geht verloren. Das heißt nicht, dass die RTC6-Befehle nicht gespeichert wurden. Fragen Sie in diesem Fall die Input-Pointer-Position manuell ab. Wenn Sie sichergehen wollen, dass die RTC6-Befehle ohne Fehler gespeichert wurden: setzen Sie den Input-Pointer zurück und senden Sie das Befehls-Telegramm erneut.

16.10.9 Remote-Befehle

- Remote-Kontrollbefehle sind:
  – ID 0…ID 255
- Remote-Listenbefehle sind:
  – ID 256…ID 511
- Die meisten Remote-Befehle haben eine funktionale Entsprechung in genau einem RTC6-Befehl.
- Manche Remote-Befehle decken mit ihren Parametern und internen Behandlung gleich mehrere RTC6-Befehle ab. Beispiel:
  `ID 312 R_LC_MARK_XYZT_ABS` deckt sowohl **mark_abs** als auch **timed_mark_abs_3d** ab (**mark_abs** wird intern als **timed_mark_abs_3d** mit `Z` = 0x7FFFFFFF und `T` = 0 behandelt).
- Wichtig: Remote-Listenbefehle werden nicht gepuffert. Daher gelten Aussagen zu RTC6-Befehlen (in anderen Kapiteln), dass ein Flush ausgelöst wird, im Remote-Interface-Modus-Kontext nicht.
- Die Typ-Bezeichnungen `UINT` und `LONG` entsprechen denen aus den Importdeklarationen für C/C++.

`UINT`	`typedef uint32_t`
`LONG`	`typedef int32_t`

- Weitere Datentypen bei Remote-Befehlen sind:

`CHAR`	8-Bit-Einzelzeichen.
`DOUBLE`	64-Bit-IEEE Gleitkommaformat.

- Das Antwort-Telegramm unterscheidet nicht zwischen (normaler) “Rückgabe” und “Parameterrückgaben”. Eine “Parameterrückgabe” folgt unmittelbar auf die (normale) Rückgabe.

(1) `UDPexcl` von **eth_set_port_numbers**, `UDPexcl` von **eth_get_port_numbers**.

(2) Oder Linux Shared Library, siehe Unterverzeichnis Linux, Seite 34.

Remote-Kontrollbefehle

- Siehe auch Remote-Listenbefehle, Seite 1042.

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
0	R_DC_NO_COMMAND	-	Reserviert.	-
1	R_DC_CONFIG_LIST	UINT Mem1 UINT Mem2	config_list(Mem1, Mem2).	-
2	R_DC_GET_CONFIG_LIST	-	get_config_list().	-
3	R_DC_SET_START_LIST_POS	UINT ListNo UINT Pos	set_start_list_pos(ListNo, Pos). Für UINT ListNo = 1: set_start_list_1(). Für UINT ListNo = 2: set_start_list_2(). Für UINT Pos = 0: set_start_list(ListNo).	-
4	R_DC_GET_INPUT_POINTER	-	get_input_pointer().	UINT Pos
5	R_DC_GET_LIST_SPACE	-	get_list_space().	UINT ListSpace
6	R_DC_LOAD_LIST_POS	UINT ListNo UINT Pos	load_list(ListNo, Pos).	UINT NumberOfList
7	R_DC_LOAD_SUB	UINT Index	load_sub(Index).	-
8	R_DC_LOAD_CHAR	UINT Char	load_char(Char).	-
9	R_DC_LOAD_TEXT_TABLE	UINT Index	load_text_table(Index).	-
10	R_DC_SET_EXTSTART_POS	UINT Pos	set_extstartpos(Pos).	-
11	R_DC_SET_MAX_COUNT	UINT Counts	set_max_counts(Counts).	-
12	R_DC_GET_COUNTS	-	get_counts().	UINT Counts
13	R_DC_STORE_TIMESTAMP	-	store_timestamp_counter().	-
14	R_DC_SIMULATE_EXT_START	-	simulate_ext_start_ctrl().	-
15	R_DC_EXECUTE_LIST_POS	UINT ListNo UINT Pos	execute_list_pos(ListNo, Pos). Für UINT ListNo = 1: execute_list_1(). Für UINT ListNo = 2: execute_list_2(). Für UINT Pos = 0: execute_list(ListNo).	-
16	R_DC_STOP_EXECUTION	-	stop_execution().	-
17	R_DC_SIMULATE_EXT_STOP	-	simulate_ext_stop().	-
18	R_DC_EXTERN_START_DELAY	LONG Delay UINT EncoderNo	set_ext_start_delay(Delay, EncoderNo).	-
19	R_DC_AUTO_CHANGE_POS	UINT Pos	auto_change_pos(Pos). Für UINT Pos = 0: auto_change().	-
20	R_DC_START_LOOP	-	start_loop().	-
21	R_DC_QUIT_LOOP	-	quit_loop().	-
22	R_DC_PAUSE_LIST	-	pause_list().	-
23	R_DC_RESTART_LIST	-	restart_list().	-
24	R_DC_RELEASE_WAIT	-	release_wait().	-
25	R_DC_GET_WAIT_STATUS	-	get_wait_status().	UINT WaitStatus

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
26	R_DC_SET_PAUSE_LIST_COND	UINT Mask1 UINT Mask0 UINT Mode	Für UINT Mode = 0: set_pause_list_cond(Mask1, Mask0). Für UINT Mode = 1: set_pause_list_not_cond(Mask1, Mask0).	-
27	R_DC_GET_TIME	-	get_time().	DOUBLE TimerValue
28	R_DC_GET_LAP_TIME	-	get_lap_time().	DOUBLE TimerValue
29	R_DC_SET_CONTROL_MODE	UINT Mode	set_control_mode(Mode).	-
30	R_DC_READ_STATUS	-	read_status().	UINT Listenstatus
31	R_DC_GET_STATUS	-	get_status(&Status, &Pos).	UINT Status UINT Pos
32	R_DC_GET_STARTSTOP_INFO	-	get_startstop_info().	UINT Info
33	R_DC_GET_OVERRUN	-	get_overrun().	UINT NoOfOverruns
34	R_DC_CONTROL_COMMAND	UINT Head UINT Axis UINT Data	control_command(Head, Axis, Data).	-
35	R_DC_GET_VALUE	UINT Signal	get_value(Signal).	UINT Value
36	R_DC_GET_VALUES	UINT SignalPtr[4]	get_values(SignalPtr, &ResultPtr).	UINT ResultPtr [4]
37	R_DC_GET_HEAD_STATUS	UINT HeadNo	get_head_status(Head).	UINT Statuswort
38	R_DC_GET_HEAD_PARA	UINT HeadNo UINT ParaNo	get_head_para(HeadNo, ParaNo).	DOUBLE HeadPara
39	R_DC_GET_TABLE_PARA	UINT TableNo UINT ParaNo	get_table_para(TableNo, ParaNo).	DOUBLE TablePara
40	R_DC_GET_Z_DISTANCE	LONG X LONG Y LONG Z	get_z_distance(X, Y, Z).	LONG ZDistance
41	R_DC_GET_GALVO_CONTROLS	UINT InPtr[5] = {X, Y, Z, Defocus, Zoom}	get_galvo_controls(InPtr, OutPtr).	UINT OutPtr[4] = {XA, YA, XB, YB}
42	R_DC_SET_LASER_CONTROL	UINT Ctrl	set_laser_control(Ctrl).	-
43	R_DC_SET_LASER_MODE	UINT Mode	set_laser_mode(Mode).	-
44	R_DC_ENABLE_LASER	-	enable_laser().	-
45	R_DC_DISABLE_LASER	-	disable_laser().	-
46	R_DC_LASER_SIGNAL_ON	-	laser_signal_on().	-
47	R_DC_LASER_SIGNAL_OFF	-	laser_signal_off().	-
48	R_DC_SET_QSWITCH_DELAY	UINT Delay	set_qswitch_delay(Delay).	-
49	R_DC_SET_LASER_TIMING	UINT HalfPeriod UINT PulseLength	set_laser_pulses_ctrl(HalfPeriod, PulseLength).	-
50	R_DC_SET_FIRST_PULSE_KILLER	UINT Length	set_firstpulse_killer(Length).	-
51	R_DC_SET_LASER_PIN_OUT	UINT Pins	set_laser_pin_out(Pins).	-
52	R_DC_GET_LASER_PIN_IN	-	get_laser_pin_in().	UINT LaserPinIn

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
53	R_DC_SET_STANDBY_TIMING	UINT HalfPeriod UINT PulseLength	set_standby(HalfPeriod, PulseLength).	-
54	R_DC_GET_STANDBY	-	get_standby(&HalfPeriod, &PulseLength).	UINT HalfPeriod UINT PulseLength
55	R_DC_PULSE_PICKING	UINT No	set_pulse_picking(No).	-
56	R_DC_PULSE_PICKING_LENGTH	UINT Length	set_pulse_picking_length(Length).	-
57	R_DC_CONFIG_LASER_SIGNALS	UINT Config	config_laser_signals(Config).	-
58	R_DC_SET_AUTO_LASER_CONTROL	UINT Ctrl UINT Value UINT Mode UINT MinValue UINT MaxValue	set_auto_laser_control(Ctrl, Value, Mode, MinValue, MaxValue).	UINT ErrorCode
59	R_DC_SET_AUTO_LASER_PARAMS	UINT Ctrl UINT Value UINT MinValue UINT MaxValue	ErrorCode = set_auto_laser_params(Ctrl, Value, MinValue, MaxValue)	-
60	R_DC_SPOT_DISTANCE	DOUBLE Dist	spot_distance_ctrl(Dist).	-
61	R_DC_SET_ENCODER_SPEED	UINT EncoderNo DOUBLE Speed DOUBLE Smooth	set_encoder_speed_ctrl(EncoderNo, Speed, Smooth).	-
62	R_DC_SET_DEFAULT_PIXEL	UINT PulseLength	set_default_pixel(PulseLength).	-
63	R_DC_SET_PORT_DEFAULT	UINT Port UINT Value	set_port_default(Port, Value).	-
64	R_DC_WRITE_8BIT_PORT	UINT Value	write_8bit_port(Value).	-
65	R_DC_WRITE_IO_PORT_MASK	UINT Value UINT Mask	write_io_port_mask(Value, Mask). Für UINT Mask = 0xFFFF: write_io_port(Value).	-
66	R_DC_GET_IO_STATUS	-	get_io_status().	UINT IOStatus
67	R_DC_READ_IO_PORT	-	read_io_port().	UINT IOPort
68	R_DC_READ_IO_PORT_BUF	UINT Index	read_io_port_buffer(Index, &Value, &XPos, &YPos, &Time).	UINT CurrentIndex UINT Value LONG XPos LONG YPos UINT Time
69	R_DC_WRITE_DA_X	UINT x UINT Value	write_da_x(x, Value). Für UINT x = 1: write_da_1(Value). Für UINT x = 2: write_da_2(Value).	-
70	R_DC_READ_ANALOG_IN	-	read_analog_in().	UINT AnalogValue
71	R_DC_SET_JUMP_SPEED	DOUBLE Speed	set_jump_speed_ctrl(Speed).	-
72	R_DC_SET_MARK_SPEED	DOUBLE Speed	set_mark_speed_ctrl(Speed).	-
73	R_DC_SET_DEFOCUS	LONG Shift	set_defocus(Shift).	-
74	R_DC_SET_DEFOCUS_OFFSET	LONG Shift	set_defocus_offset(Shift).	-

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
75	R_DC_SET_DELAY_MODE	UINT VarPoly UINT DirectMode3D UINT EdgeLevel UINT MinJumpDelay UINT JumpLengthLimit	set_delay_mode(VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit).	-
76	R_DC_GOTO_XYZ	LONG X LONG Y LONG Z	goto_xyz(X, Y, Z). Für LONG Z = 0x7FFFFFFF: goto_xy(X, Y).	-
77	R_DC_SET_OFFSET_XYZ	UINT HeadNo LONG X LONG Y LONG Z UINT at_once	set_offset_xyz(HeadNo, XOffset, YOffset, ZOffset, at_once). Für LONG Z = 0x7FFFFFFF: set_offset(HeadNo, XOffset, YOffset, at_once).	-
78	R_DC_HOME_POSITION_XYZ	LONG XHome LONG YHome LONG ZHome	home_position_xyz(XHome, YHome, ZHome). Für LONG ZHome = 0x7FFFFFFF: home_position(XHome, YHome).	-
79	R_DC_TIME_UPDATE	UINT DateTime[8] = {Year, Month, Day, YearDay, WeekDay, Hours, Minutes, Seconds} siehe 1, Seite 1041	time_update() zum Zeitpunkt UINT DateTime[8].	-
80	R_DC_TIME_CONTROL_ETH	DOUBLE PPM	time_control_eth(PPM).	-
81	R_DC_SET_SERIAL	UINT No UINT Step	set_serial_step(No, Step).	-
82	R_DC_SELECT_SERIAL_SET	UINT No	select_serial_set(No).	-
83	R_DC_GET_SERIAL	-	get_serial().	DOUBLE Current-SerialNo
84	R_DC_GET_LIST_SERIAL	-	get_list_serial(&Set).	DOUBLE LastMarked-SerialNo UINT Set
85	R_DC_SET_ZOOM	UINT Zoom	set_zoom(Zoom).	-
86	R_DC_SET_MATRIX	UINT HeadNo DOUBLE M11 DOUBLE M12 DOUBLE M21	set_matrix(HeadNo, M11, M12, M21, M22, at_once).	DOUBLE M22 UINT at_once
87	R_DC_SET_ANGLE	UINT HeadNo DOUBLE Angle UINT at_once	set_angle(HeadNo, Angle, at_once).	-
88	R_DC_SET_SCALE	UINT HeadNo DOUBLE Scale UINT at_once	set_scale(HeadNo, Scale, at_once).	-
89	R_DC_SIMULATE_ENCODER	UINT EncoderNo	simulate_encoder(EncoderNo).	-
90	R_DC_READ_ENCODER	-	read_encoder(&Encoder0_0, &Encoder1_0, &Encoder0_1, &Encoder1_1).	LONG Encoder0_0 LONG Encoder1_0 LONG Encoder0_1 LONG Encoder1_1

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
91	R_DC_GET_ENCODER	-	get_encoder(&Encoder0, &Encoder1).	LONG Encoder0 LONG Encoder1
92	R_DC_INIT_FLY_2D	LONG OffsetX LONG OffsetY UINT No	init_fly_2d(OffsetX, OffsetY, No).	-
93	R_DC_GET_FLY_2D_OFFSET	-	get_fly_2d_offset(&OffsetX, &OffsetY).	LONG OffsetX LONG OffsetY
94	R_DC_GET_MARKING_INFO	-	get_marking_info().	UINT MarkingInfo
95	R_DC_ROT_CENTER	LONG X LONG Y	set_rot_center(X, Y).	-
96	R_DC_SET_MCBSP_FREQ	UINT Freq	set_mcbsp_freq(Freq).	UINT mcbsp_freq
97	R_DC_MCBSP_INIT	UINT XDelay UINT RDelay	mcbsp_init(XDelay, RDelay).	-
98	R_DC_SET_MCBSP_MODE	UINT Axis DOUBLE Scale	Für UINT Axis = 1: set_mcbsp_x(Scale). Für UINT Axis = 2: set_mcbsp_y(Scale). Für UINT Axis = 4: set_mcbsp_rot(Resolution) mit Resolution = DOUBLE Scale. Für UINT Axis = 0 und DOUBLE Scale = egal: set_mcbsp_matrix().	-
99	R_DC_SET_MCBSP_MODE_GLOBAL	UINT Axis DOUBLE Scale	Für UINT Axis = 1 set_mcbsp_global_x(Scale). Für UINT Axis = 2 set_mcbsp_global_y(Scale). Für UINT Axis = 4 set_mcbsp_global_rot(Resolution) mit Resolution = DOUBLE Scale. Für UINT Axis = 0 und DOUBLE Scale = egal: set_mcbsp_global_matrix().	-
100	R_DC_SET_MCBSP_IN	UINT Mode DOUBLE Scale	set_mcbsp_in(Mode, Scale).	-
101	R_DC_SET_MULTI_MCBSP_IN	UINT Ctrl UINT P UINT Mode	set_multi_mcbsp_in(Ctrl, P, Mode).	-
102	R_DC_READ_MCBSP	UINT No	read_mcbsp(No).	UINT mcbsp_value
103	R_DC_READ_MULTI_MCBSP	UINT No	read_multi_mcbsp(No).	UINT mcbsp_value
104	R_DC_APPLY_MCBSP	UINT HeadNo UINT at_once	apply_mcbsp(HeadNo, at_once).	-
105	R_DC_GET_SCANAHEAD_PARAMS	UINT HeadNo	get_scanahead_params(HeadNo, &Preview, &Amax, &Vmax).	UINT Error UINT Preview UINT Amax UINT Vmax
106	R_DC_SCANAHEAD_ACTIVATE_AUTO	UINT Mode	activate_scanahead_autodelays(Mode).	UINT CurrentMode
107	R_DC_SET_SCANAHEAD_LASER_SHIFT	LONG dLasOn LONG dLasOff	set_scanahead_laser_shifts(dLasOn, dLasOff).	-
108	R_DC_SET_SCANAHEAD_LINE_PARAMS	UINT CornerScale UINT EndScale UINT AccScale UINT JumpScale	set_scanahead_line_params (CornerScale, EndScale, AccScale).	-

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
109	R_DC_SET_SCANAHEAD_PARAMS	UINT Mode UINT HeadNo UINT TableNo UINT Preview UINT Vmax DOUBLE Amax	set_scanahead_params(Mode, HeadNo, TableNo, Preview, Vmax, Amax).	UINT Error
110	R_DC_SET_SCANAHEAD_SPEED_CTRL	UINT Mode	set_scanahead_speed_control(Mode).	-
111	R_DC_STEPPER_INIT	UINT No UINT Period LONG Dir LONG Pos UINT Tol UINT Enable UINT WaitTime	stepper_init(No, Period, Dir, Pos, Tol, Enable, WaitTime).	-
112	R_DC_STEPPER_ENABLE	LONG Enable1 LONG Enable2	stepper_enable(Enable1, Enable2).	-
113	R_DC_STEPPER_DISABLE_SWITCH	LONG Disable1 LONG Disable2	stepper_disable_switch(Disable1, Disable2).	-
114	R_DC_STEPPER_CTRL	LONG Period1 LONG Period2	stepper_control(Period1, Period2).	-
115	R_DC_STEPPER_NO_ABS	UINT No LONG Pos UINT WaitTime	stepper_abs_no(No, Pos, WaitTime).	-
116	R_DC_STEPPER_NO_REL	UINT No LONG dPos UINT WaitTime	stepper_rel_no(No, dPos, WaitTime).	-
117	R_DC_STEPPER_ABS	LONG Pos1 LONG Pos2 UINT WaitTime	stepper_abs(Pos1, Pos2, WaitTime).	-
118	R_DC_STEPPER_REL	LONG dPos1 LONG dPos2 UINT WaitTime	stepper_rel(dPos1, dPos2, WaitTime).	-
119	R_DC_GET_STEPPER_STATUS	-	get_stepper_status(&Status1, &Pos1, &Status2, &Pos2).	UINT Status1 LONG Pos1 UINT Status2 LONG Pos2
120	R_DC_SET_SKY_WRITING_PARA	DOUBLE Timelag LONG LaserOnShift UINT Nprev UINT Npost	set_sky_writing_para(Timelag, LaserOnShift, Nprev, Npost). Für UINT Nprev, UINT Npost ≥ 65.535: set_sky_writing(Timelag, LaserOnShift).	-
121	R_DC_SET_SKY_WRITING_ANGLE	DOUBLE Limit	set_sky_writing_limit(Limit).	-
122	R_DC_SET_SKY_WRITING_MODE	UINT Mode	set_sky_writing_mode(Mode).	-
123	R_DC_UART_CONFIG	UINT BaudRate	uart_config(BaudRate).	UINT RealBaudRate
124	R_DC_RS232_WRITE_DATA	UINT Data	rs232_write_data(Data).	-
125	R_DC_RS232_WRITE_TEXT	CHAR Text[N] siehe 2, Seite 1041	rs232_write_text(pData).	-

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
126	R_DC_RS232_READ_DATA	-	rs232_read_data().	UINT RS232Data
127	R_DC_BOUNCE_SUPP	UINT Length	bounce_supp(Length).	-
128	R_DC_SET_HI	UINT HeadNo DOUBLE GalvoGainX DOUBLE GalvoGainY LONG GalvoOffsetX LONG GalvoOffsetY	set_hi(HeadNo, GalvoGainX, GalvoGainY, GalvoOffsetX, GalvoOffsetY).	-
129	R_DC_SET_JUMP_MODE	LONG Flag UINT Length LONG VA1 LONG VA2 LONG VB1 LONG VB2 LONG JA1 LONG JA2 LONG JB1 LONG JB2	set_jump_mode(Flag, Length, VA1, VA2, VB1, VB2, JA1, JA2, JB1, JB2).	-
130	R_DC_SELECT_COR_TABLE	UINT HeadA UINT HeadB	select_cor_table(HeadA, HeadB).	-
131	R_DC_SET_FREE_VAR	UINT VarNo UINT Value	set_free_variable(No, Value).	-
132	R_DC_GET_FREE_VAR	UINT VarNo	get_free_variable(No).	UINT Value
133	R_DC_SET_MCBSP_OUT_PTR	UINT Num UINT SignalPtr[Num]	set_mcbsp_out_ptr(Number, SignalPtr).	-
134	R_DC_PERIODIC_TOGGLE	UINT Port UINT Mask UINT P1 UINT P2 UINT Count UINT Start	periodic_toggle(Port, Mask, P1, P2, Count, Start).	-
135	R_DC_ENDURING_WOBBEL_1	UINT CenterX UINT CenterY UINT CenterZ UINT LimitHi UINT LimitLo DOUBLE ScaleX DOUBLE ScaleY DOUBLE ScaleZ	Reserviert.	-
136	R_DC_ENDURING_WOBBEL_2	UINT CenterX UINT CenterY UINT CenterZ UINT LimitHi UINT LimitLo DOUBLE ScaleX DOUBLE ScaleY DOUBLE ScaleZ	Reserviert.	-
137	R_DC_STOP_TRIGGER	-	stop_trigger().	-
138	R_DC_MEASUREMENT_STATUS	-	measurement_status(&Busy, &Pos).	UINT Busy UINT Pos

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
139	R_DC_GET_WAVEFORM	UINT Channel UINT Offset UINT Number	get_waveform_offset(Channel, Offset, Number, Ptr).	UINT Index UINT Data[Number] siehe 3, Seite 1041
140	R_DC_SET_VERIFY	UINT Verify	set_verify(Verify).	UINT OldVerify
141	R_DC_GET_HEX_VERSION	-	get_hex_version().	UINT HexVersion
142	R_DC_GET_RTC_VERSION	-	get_rtc_version().	UINT RTCVersion
143	R_DC_GET_BIOS_VERSION	-	get_bios_version().	UINT BiosVersion
144	R_DC_GET_SERIAL_NUMBER	-	get_serial_number().	UINT SerialNumber
145	R_DC_SET_INPUT_POINTER	UINT Pos	set_input_pointer(Pos).	-
146	R_DC_GET_LIST_POINTER	-	get_list_pointer(&ListNo, &Pos).	UINT ListNo UINT Pos
147	R_DC_EXECUTE_AT_POINTER	UINT Pos	execute_at_pointer(Pos).	-
148	R_DC_GET_REMOTE_ERROR	-	Ein entsprechender RTC6-Befehl existiert nicht, da get_error Bit #31 nur im Remote-Interface-Modus auftreten kann.	UINT Error siehe Fehler-Bit, Seite 1028
149	R_DC_SET_LASER_PULSE_SYNC	UINT Mode UINT Delay	set_laser_pulse_sync(Mode, Delay).	-
150	R_DC_GET_TIMESTAMP_LONG	-	get_timestamp_long(&TimeStampL, &TimeStampH).	UINT TimestampLow UINT TimestampHigh
151	R_DC_CLEAR_FLY_OVERFLOW	UINT Mode	clear_fly_overflow_ctrl(Mode).	-
152	R_DC_GET_TEMPERATURE	-	get_temperature().	DOUBLE Temperature
153	R_DC_SET_MCBSP_OUT_OIE	UINT Signal1 UINT Signal2	set_mcbsp_out_oie_ctrl(Signal1, Signal2).	-
154	R_DC_LOAD_CORRECTION_FILE	UINT Offset UINT Size CHAR Data[Size]	load_correction_file(…). Siehe 4, Seite 1041	UINT Error
155	R_DC_NUMBER_OF_COR_TABLES	UINT Number	number_of_correction_tables(Number).	-
156	R_DC_LOAD_Z_TABLE	DOUBLE A DOUBLE B DOUBLE C UINT No	load_z_table_no(A, B, C, No).	UINT Error
157	R_DC_LOAD_OCT_TABLE	DOUBLE A DOUBLE B UINT No	Reserviert.	UINT Error

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
158	R_DC_SET_TIMELAG_COMPENSATION	UINT HeadNo UINT TimelagXY UINT TimelagZ	set_timelag_compensation(HeadNoXY, TimelagXY, TimelagZ).	-
159	R_DC_SET_ENCODER_FILTER	UINT Length UINT Mode	set_encoder_filter_ctrl(Length, Mode).	-
160	R_DC_GET_MASTER_SLAVE	-	get_master_slave().	UINT MasterSlaveStatus
161	R_DC_GET_SYNC_STATUS	-	get_sync_status().	UINT SyncStatus
162	R_DC_MASTER_SLAVE_CONFIG	UINT Config	master_slave_config(Config).	-
163	R_DC_GET_STATIC_IP	-	eth_get_static_ip(&Ip, &NetMask, &Gateway).	UINT Ip UINT Netmask UINT Gateway
164	R_DC_SET_STATIC_IP	UINT Ip UINT Netmask UINT Gateway	eth_set_static_ip(Ip, NetMask, Gateway).	UINT Error
165	R_DC_GET_PORT_NUMBERS	-	eth_get_port_numbers(&UDPsearch, &UDPexcl, &TCP).	UINT UDPsearch UINT UDPexcl UINT TCP
166	R_DC_SET_PORT_NUMBERS	UINT UDPsearch UINT UDPexcl UINT TCP	eth_set_port_numbers(UDPsearch, UDPexcl, TCP).	UINT Error
167	R_DC_CONFIGURE_LINK_LOSS	UINT Mode	eth_configure_link_loss(Mode).	-
168	R_DC_SET_SKY_WRITING_MIN_SPEED	DOUBLE Speed	set_sky_writing_min_speed_ctrl(Speed).	-
169	R_DC_SET_CTRL_PREVIEW_COMP	UINT ControlPreview UINT Mode	set_controlpreview_compensation_ctrl(ControlPreview, Mode).	-
170	R_DC_SET_FLY_TRACKING_ERROR	UINT TrackingErrorX UINT TrackingErrorY	set_fly_tracking_error(TrackingErrorX, TrackingErrorY).	-
171	R_DC_SET_LASER_DELAYS	LONG LaserOnDelay UINT LaserOffDelay	set_laser_delays_ctrl(LaserOnDelay, LaserOffDelay).	-
172	R_DC_SET_SCANNER_DELAYS	UINT Jump UINT Mark UINT Polygon	set_scanner_delays_ctrl(Jump, Mark, Polygon).	-
173	R_DC_SET_SKY_WRITING_SCALE	DOUBLE Scale	set_sky_writing_scale_ctrl(Scale).	-
174	R_DC_GET_EXTENDED_HEAD_STATUS_VALUE	UINT Head UINT Axis UINT Value	get_extended_head_status_value_ctrl(Head, Axis, Value).	UINT Value

Remote-Kontrollbefehle, Seite 1031

ID	Remote-Kontrollbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
175	R_DC_GET_EXTENDED_HEAD_STATUS_BLOCK	UINT Head UINT Axis	get_extended_head_status_block_ctrl(Head, Axis, Ptr).	UINT Block[256]
176	R_DC_GET_RCM_VALUE	UINT Head UINT Axis UINT Value	get_rcm_value_ctrl(Head, Axis, Value).	UINT Value
177	R_DC_MASTER_SLAVE_SET_OFFSET	UINT Mode UINT Ctrl	master_slave_set_offset_ctrl(Mode, Ctrl)	…
…	…	…	…	…
255	-	-	-	-

Remote-Kontrollbefehle, Seite 1031

Hinweise

(1) Zu 79 R_DC_TIME_UPDATE.

```
// Beispiel-Code für Windows (C++)

time64_t dt( _time64( NULL ) );
tm today;
(void) _localtime64_s( &today, &dt
UINT DateTime[8]

DateTime[ E_YEAR    ] = today.tm_year + 1900;

// 1-12
DateTime[ E_MONTH   ] = today.tm_mon + 1;

// 1…31
DateTime[ E_DAY     ] = today.tm_mday;

// 0-364/365
DateTime[ E_YEARDAY ] = today.tm_yday;

// 1-7. 1 = Sunday
DateTime[ E_WEEKDAY ] = today.tm_wday + 1;

// 0-23
DateTime[ E_HOURS   ] = today.tm_hour;

// 0-59
DateTime[ E_MINUTES ] = today.tm_min;

// 0-59
DateTime[ E_SECONDS ] = today.tm_sec;
```

(2) Zu 125 R_DC_RS232_WRITE_TEXT, CHAR Text[N].
Sie müssen N so wählen, dass die Telegramm-Gesamtlänge TGM_MAX_SIZE nicht überschreitet. Verteilen Sie gegebenenfalls den Text auf mehrere Remote-Befehle. Sie müssen jeden der Texte mit "\0" abschließen.

(3) Zu 139 R_DC_GET_WAVEFORM.
Number von UINT Data[Number] muss ≤ 343(1) sein. Anderenfalls liefert LastError 1. ID (in TGM_PL_ANSW_RAW) RTC6_PARAM_ERROR.
Größere Datenmengen müssen Sie mit mehreren 139 R_DC_GET_WAVEFORM-Aufrufen auslesen, siehe **get_waveform_offset**-Implementierung in rtc6_rif_wrapper.cpp.

(4) Zu 154 R_DC_LOAD_CORRECTION_FILE.
load_correction_file muss im Remote-Interface-Modus in mehreren Schritten implementiert werden, siehe auch rtc6_rif_wrapper.cpp.
In Schritt 1 muss der gesamte Inhalt der Korrekturdatei eingelesen und an die RTC6 Ethernet-Karte gesendet werden. Die Datenmenge ist zu groß für ein einziges Befehls-Telegramm. Daher muss dies über mehrere Befehls-Telegramme erfolgen. UINT Offset gibt den Byte-Offset innerhalb der Datei an. UINT Size gibt die Anzahl an Bytes des aktuellen Blocks an. Darauf folgen n Bytes an Daten (n = Size).
Schritt 2 folgt erst, wenn der gesamte Korrekturdatei-Inhalt auf die RTC6 Ethernet-Karte geladen ist. Dafür muss UINT Offset auf –1 (UINT_MAX, siehe telegrams.h) gesetzt werden. In UINT Size muss die Korrekturdatei-Nummer (No) in die unteren 16 Bits und die Dimension (Dim) in die oberen 16 Bits geschrieben werden. Es folgen keine weiteren Daten im Schritt 2. Der Aufruf entspricht
load_correction_file(Name = Dateipfad, No, Dim).

(1) Herleitung: 4 Byte je 1 Waveform-Wert.
[5 × 4 Byte TGM_HEADER] +
[1 × 4 Byte LastError 1. ID] +
[1 × 4 Byte Remote-Befehl 1. ID] + [343 × 4 Byte] =
[≤ 1.400 Byte TGM_MAX_SIZE].

Remote-Listenbefehle

- Siehe auch Remote-Kontrollbefehle, Seite 1031.

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
256	R_LC_NO_COMMAND	-	list_nop().	-
257	R_LC_CONTINUE	-	list_continue().	-
258	R_LC_NEXT	-	list_next().	-
259	R_LC_END_OF_LIST	-	set_end_of_list().	-
260	R_LC_LIST_RETURN	-	list_return().	-
261	R_LC_REPEAT	-	list_repeat().	-
262	R_LC_UNTIL	UINT Number	list_until(Number).	-
263	R_LC_LIST_JUMP_POS	UINT Pos	list_jump_pos(Pos).	-
264	R_LC_LIST_JUMP_REL	LONG Pos	list_jump_rel(Pos).	-
265	R_LC_SET_WAIT	UINT WaitWord	set_wait(WaitWord).	-
266	R_LC_LONG_DELAY	UINT Delay	long_delay(Delay).	-
267	R_LC_SET_EXTSTART_POS	UINT Pos	set_extstartpos_list(Pos).	-
268	R_LC_SIMULATE_EXTERN_START	LONG Delay UINT EncoderNo	simulate_ext_start(Delay, EncoderNo).	-
269	R_LC_LIST_CALL	UINT Pos UINT Number UINT AbsCall	Für UINT AbsCall = 1: list_call_abs_repeat(Pos, Number). Für UINT Number = 0 oder 1: list_call_abs(Pos). Für UINT AbsCall = 0: list_call_repeat(Pos, Number). Für UINT Number = 0 oder 1: list_call(Pos).	-
270	R_LC_SUB_CALL	UINT Index UINT Number UINT AbsCall	Für UINT AbsCall = 1: sub_call_abs_repeat(Index, Number). Für UINT Number = 0 oder 1: sub_call_abs(Index). Für UINT AbsCall = 0: sub_call_repeat(Index, Number). Für UINT Number = 0 oder 1: sub_call(Index).	-
271	R_LC_IF_COND	UINT Mask1 UINT Mask0 UINT Mode	Für UINT Mode = 0: if_cond(Mask1, Mask0). Für UINT Mode = 1: if_not_cond(Mask1, Mask0).	-
272	R_LC_IF_PIN_COND	UINT Mask1 UINT Mask0 UINT Mode	Für UINT Mode = 0: if_pin_cond(Mask1, Mask0). Für UINT Mode = 1: if_not_pin_cond(Mask1, Mask0).	-
273	R_LC_CLEAR_IO_COND	UINT Mask1 UINT Mask0 UINT MaskClear	clear_io_cond_list(Mask1, Mask0, MaskClear).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
274	R_LC_SET_IO_COND	UINT Mask1 UINT Mask0 UINT MaskSet	set_io_cond_list(Mask1, Mask0, MaskSet).	-
275	R_LC_SWITCH_IOPORT	UINT MaskBits UINT BitShifts	switch_ioport(MaskBits, ShiftBits).	-
276	R_LC_LIST_CALL_COND	UINT Mask1 UINT Mask2 UINT Pos UINT AbsCall	Für UINT AbsCall = 1: list_call_abs_cond(Mask1, Mask0, Pos). Für UINT AbsCall = 0: list_call_cond(Mask1, Mask0, Pos).	-
277	R_LC_SUB_CALL_COND	UINT Mask1 UINT Mask2 UINT Index UINT AbsCall	Für UINT AbsCall = 1: sub_call_abs_cond(Mask1, Mask0, Index). Für UINT AbsCall = 0: sub_call_cond(Mask1, Mask0, Index).	-
278	R_LC_LIST_JUMP_POS_COND	UINT Mask1 UINT Mask0 UINT Pos	list_jump_pos_cond(Mask1, Mask0, Pos).	-
279	R_LC_LIST_JUMP_REL_COND	UINT Mask1 UINT Mask0 LONG Pos	list_jump_rel_cond(Mask1, Mask0, Pos).	-
280	R_LC_CONFIG_LASER_SIGNALS	UINT Config	config_laser_signals_list(Config).	-
281	R_LC_LASER_SIGNAL_ON	-	laser_signal_on_list().	-
282	R_LC_LASER_SIGNAL_OFF	-	laser_signal_off_list().	-
283	R_LC_SET_LASER_TIMING	UINT HalfPeriod UINT PulseLength	set_laser_pulses(HalfPeriod, PulseLength).	-
284	R_LC_SET_FIRST_PULSE_KILLER	UINT Length	set_firstpulse_killer_list(Length).	-
285	R_LC_SET_QSWITCH_DELAY	UINT Delay	set_qswitch_delay_list(Delay).	-
286	R_LC_SET_LASER_PIN_OUT	UINT Pins	set_laser_pin_out_list(Pins).	-
287	R_LC_PULSE_PICKING	UINT No	set_pulse_picking_list(No).	-
288	R_LC_SET_STANDBY_TIMING	UINT HalfPeriod UINT PulseLength	set_standby_list(HalfPeriod, PulseLength).	-
289	R_LC_SET_AUTO_LASER_PARAMS	UINT Ctrl UINT Value UINT MinValue UINT MaxValue	ErrorCode = set_auto_laser_params_list(Ctrl, Value, MinValue, MaxValue).	-
290	R_LC_LASER_POWER	UINT Port UINT Power	set_laser_power(Port, Power).	-
291	R_LC_SPOT_DISTANCE	DOUBLE Dist	spot_distance(Dist).	-
292	R_LC_SET_ENCODER_SPEED	UINT Encoder DOUBLE Speed DOUBLE Smooth	set_encoder_speed(EncoderNo, Speed, Smooth).	-
293	R_LC_SET_DEFAULT_PIXEL	UINT PulseLength	set_default_pixel_list(PulseLength).	-
294	R_LC_SET_PORT_DEFAULT	UINT Port UINT Value	set_port_default_list(Port, Value).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
295	R_LC_SET_PIXEL_LINE	UINT Channel UINT HalfPeriod DOUBLE dX DOUBLE dY DOUBLE dZ	set_pixel_line_3d(Channel, HalfPeriod, dX, dY, dZ). Für DOUBLE dZ = 0: set_pixel_line(Channel, HalfPeriod, dX, dY).	-
296	R_LC_SET_N_PIXEL_DATA	UINT PortOutValue1 UINT PortOutValue2 UINT Number	set_n_pixel(PortOutValue1, PortOutValue2, Number). Für UINT Number = 1: set_pixel(PortOutValue1, PortOutValue2).	-
297	R_LC_WRITE_8BIT_PORT	UINT Value	write_8bit_port(Value).	-
298	R_LC_WRITE_IO_PORT_MASK	UINT Value UINT Mask	write_io_port_mask_list(Value, Mask). Für UINT Mask = 0xFFFF write_io_port_list(Value).	-
299	R_LC_READ_IO_PORT	-	read_io_port_list().	-
300	R_LC_WRITE_DA_X	UINT x UINT Value	write_da_x_list(x, Value). Für UINT x = 1: write_da_1_list(Value). Für UINT x = 2: write_da_2_list(Value).	-
301	R_LC_SET_MARK_SPEED	DOUBLE Speed	set_mark_speed(Speed).	-
302	R_LC_SET_JUMP_SPEED	DOUBLE Speed	set_jump_speed(Speed).	-
303	R_LC_SET_SCANNER_DELAYS	UINT Jump UINT Mark UINT Polygon	set_scanner_delays(Jump, Mark, Polygon).	-
304	R_LC_SET_LASER_DELAYS	LONG LaserOnDelay UINT LaserOffDelay	set_laser_delays(LaserOnDelay, LaserOffDelay).	-
305	R_LC_SET_DEFOCUS	LONG Shift	set_defocus_list(Shift).	-
306	R_LC_SET_DEFOCUS_OFFSET	LONG Shift	set_defocus_offset_list(Shift).	-
307	R_LC_SET_ZOOM	UINT Zoom	set_zoom_list(Zoom).	-
308	R_LC_SET_OFFSET_XYZ	UINT HeadNo LONG XOffset LONG YOffset LONG ZOffset UINT at_once	set_offset_xyz_list(HeadNo, XOffset, YOffset, ZOffset, at_once). Für LONG ZOffset = 0x7FFFFFFF: set_offset_list(HeadNo, XOffset, YOffset, at_once).	-
309	R_LC_SET_MATRIX	UINT HeadNo UINT Ind1 UINT Ind2 DOUBLE Mij UINT at_once	set_matrix_list(HeadNo, Ind1, Ind2, Mij, at_once).	-
310	R_LC_SET_ANGLE	UINT HeadNo DOUBLE Angle UINT at_once	set_angle_list(HeadNo, Angle, at_once).	-
311	R_LC_SET_SCALE	UINT HeadNo DOUBLE Scale UINT at_once	set_scale_list(HeadNo, Scale, at_once).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
312	R_LC_MARK_XYZT_ABS	LONG X LONG Y LONG Z DOUBLE T	timed_mark_abs_3d(X, Y, Z, T). Für LONG Z = 0x7FFFFFFF: timed_mark_abs(X, Y, T). Für DOUBLE T = 0: mark_abs_3d(X, Y, Z). Für LONG Z = 0x7FFFFFFF, DOUBLE T = 0: mark_abs(X, Y).	-
313	R_LC_MARK_XYZT_REL	LONG dX LONG dY LONG dZ DOUBLE T	timed_mark_rel_3d(dX, dY, dZ, T). Für LONG dZ = 0: timed_mark_rel(dX, dY, T). Für DOUBLE T = 0: mark_rel_3d(dX, dY, dZ). Für LONG dZ = 0 und DOUBLE T = 0: mark_rel(dX, dY).	-
314	R_LC_ARC_XYAT_ABS	LONG X LONG Y DOUBLE Angle DOUBLE T	timed_arc_abs(X, Y, Angle, T). Für DOUBLE T = 0: arc_abs(X, Y, Angle).	-
315	R_LC_ARC_XYAT_REL	LONG dX LONG dY DOUBLE Angle DOUBLE T	timed_arc_rel(dX, dY, Angle, T). Für DOUBLE T = 0: arc_rel(dX, dY, Angle).	-
316	R_LC_ARC_XYZA_ABS	LONG X LONG Y LONG Z DOUBLE Angle	arc_abs_3d(X, Y, Z, Angle).	-
317	R_LC_ARC_XYZA_REL	LONG dX LONG dY LONG dZ DOUBLE Angle	arc_rel_3d(dX, dY, dZ, Angle).	-
318	R_LC_SET_ELLIPSE	UINT A UINT B DOUBLE Phi0 DOUBLE Phi	set_ellipse(a, b, Phi0, Phi).	-
319	R_LC_ELLIPSE_ABS	LONG X LONG Y DOUBLE Alpha	mark_ellipse_abs(X, Y, Alpha).	-
320	R_LC_ELLIPSE_REL	LONG dX LONG dY DOUBLE Alpha	mark_ellipse_rel(dX, dY, Alpha).	-
321	R_LC_JUMP_XY_ABS	LONG X LONG Y	jump_abs(X, Y).	-
322	R_LC_JUMP_XY_REL	LONG dX LONG dY	jump_rel(dX, dY).	-
323	R_LC_JUMP_XYZT_ABS	LONG X LONG Y LONG Z DOUBLE T	timed_jump_abs_3d(X, Y, Z, T). Für LONG Z = 0x7FFFFFFF: timed_jump_abs(X, Y, T). Für DOUBLE T = 0: jump_abs_3d(X, Y, Z).	-
324	R_LC_JUMP_XYZT_REL	LONG dX LONG dY LONG dZ DOUBLE T	timed_jump_rel_3d(dX, dY, dZ, T). Für LONG dZ = 0: timed_jump_rel(dX, dY, T). Für DOUBLE T = 0: jump_rel_3d(dX, dY, dZ).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
325	R_LC_JUMP_XY_DRILL_ABS	LONG X LONG Y UINT DrillTime LONG XOff LONG YOff	jump_abs_drill_2(X, Y, DrillTime, XOff, YOff). Für LONG XOff = LONG YOff = 0: jump_abs_drill(X, Y, DrillTime).	-
326	R_LC_JUMP_XY_DRILL_REL	LONG dX LONG dY UINT DrillTime LONG XOff LONG YOff	jump_rel_drill_2(dX, dY, DrillTime, XOff, YOff). Für LONG XOff = LONG YOff = 0: jump_rel_drill(dX, dY, DrillTime).	-
327	R_LC_SET_VECTOR_CONTROL	UINT Ctrl UINT Value	set_vector_control(Ctrl, Value).	-
328	R_LC_LASER_ON_PULSES_PARA	UINT Period UINT Pulses UINT P	para_laser_on_pulses_list(Period, Pulses, P). Für UINT Pulses > 65.535: laser_on_list(Period). Für UINT P = $2^{32}-1$: laser_on_pulses_list(Period, Pulses).	-
329	R_LC_MARK_XYZP_ABS	LONG X LONG Y LONG Z UINT P	para_mark_abs_3d(X, Y, Z, P). Für LONG Z = 0x7FFFFFFF: para_mark_abs(X, Y, P).	-
330	R_LC_MARK_XYZP_REL	LONG dX LONG dY LONG dZ UINT P	para_mark_rel_3d(dX, dY, dZ, P). Für LONG dZ = 0: para_mark_rel(dX, dY, P).	-
331	R_LC_MARK_XYZPT_ABS	LONG X LONG Y LONG Z UINT P DOUBLE T	timed_para_mark_abs_3d(X, Y, Z, P, T). Für LONG Z = 0x7FFFFFFF: timed_para_mark_abs(X, Y, P, T).	-
332	R_LC_MARK_XYZPT_REL	LONG dX LONG dY LONG dZ UINT P DOUBLE T	timed_para_mark_rel_3d(dX, dY, dZ, P, T). Für LONG dZ = 0: timed_para_mark_rel(dX, dY, P, T).	-
333	R_LC_JUMP_XYZP_ABS	LONG X LONG Y LONG Z UINT P	para_jump_abs_3d(X, Y, Z, P). Für LONG Z = 0x7FFFFFFF: para_jump_abs(X, Y, P).	-
334	R_LC_JUMP_XYZP_REL	LONG dX LONG dY LONG dZ UINT P	para_jump_rel_3d(dX, dY, dZ, P). Für LONG dZ = 0: para_jump_rel(dX, dY, P).	-
335	R_LC_JUMP_XYZPT_ABS	LONG X LONG Y LONG Z UINT P DOUBLE T	timed_para_jump_abs_3d(X, Y, Z, P, T). Für LONG Z = 0x7FFFFFFF: timed_para_jump_abs(X, Y, P, T).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
336	R_LC_JUMP_XYZPT_REL	LONG dX LONG dY LONG dZ UINT P DOUBLE T	timed_para_jump_rel_3d(dX, dY, dZ, P, T). Für LONG dZ = 0: timed_para_jump_rel(dX, dY, P, T).	-
337	R_LC_MICRO_VECTOR_ABS_3D	LONG X LONG Y LONG Z LONG LasOn LONG LasOff	micro_vector_abs_3d(X, Y, Z, LasOn, LasOff). Für LONG Z = 0x7FFFFFFF: micro_vector_abs(X, Y, LasOn, LasOff).	-
338	R_LC_MICRO_VECTOR_REL_3D	LONG dX LONG dY LONG dZ LONG LasOn LONG LasOff	micro_vector_rel_3d(dX, dY, dZ, LasOn, LasOff). Für LONG dZ = 0: micro_vector_rel(dX, dY, LasOn, LasOff).	-
339	R_LC_MARK_CHAR	UINT Char UINT AbsCall	Für UINT AbsCall = 1: mark_char_abs(Char). Für UINT AbsCall = 0: mark_char(Char).	-
340	R_LC_SELECT_CHAR_SET	UINT No	select_char_set(No).	-
341	R_LC_MARK_TEXT	CHAR Text[N] siehe 1, Seite 1053 UINT AbsCall	Für UINT AbsCall = 1: mark_text_abs(Text). Für UINT AbsCall = 0: mark_text(Text).	-
342	R_LC_MARK_SERIAL	UINT Mode UINT Digits UINT AbsCall	Für UINT AbsCall = 1: mark_serial_abs(Mode, Digits). Für UINT AbsCall = 0: mark_serial(Mode, Digits).	-
343	R_LC_SET_SERIAL	UINT No UINT Step	set_serial_step_list(No, Step).	-
344	R_LC_SELECT_SERIAL_SET	UINT No	select_serial_set_list(No).	-
345	R_LC_TIME_FIX_F_OFF	UINT FirstDay UINT Offset	time_fix_f_off(FirstDay, Offset). Für UINT FirstDay = 0: time_fix(). Für UINT Offset = 0: time_fix_f(FirstDay).	-
346	R_LC_MARK_DATE	UINT Part UINT Mode UINT AbsCall	Für UINT AbsCall = 1: mark_date_abs(Part, Mode). Für UINT AbsCall = 0: mark_date(Part, Mode).	-
347	R_LC_MARK_TIME	UINT Part UINT Mode UINT AbsCall	Für UINT AbsCall = 1: mark_time_abs(Part, Mode). Für UINT AbsCall = 0: mark_time(Part, Mode).	-
348	R_LC_SET_WOBBEL_MODE_PHASE	UINT Transversal UINT Longitudinal DOUBLE Freq LONG Mode DOUBLE Phase	set_wobbel_mode_phase(Transversal, Longitudinal, Freq, Mode, Phase).	-
349	R_LC_SET_WOBBEL_MODE	UINT Transversal UINT Longitudinal DOUBLE Freq LONG Mode	set_wobbel_mode(Transversal, Longitudinal, Freq, Mode). Für LONG Mode = 0: set_wobbel(Transversal, Longitudinal, Freq).	-
350	R_LC_SET_WOBBEL_DIR	LONG dX LONG dY	set_wobbel_direction(dX, dY).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
351	R_LC_SET_WOBBEL_CONTROL	UINT Ctrl UINT Value UINT MinValue UINT MaxValue	set_wobbel_control(Ctrl, Value, MinValue, MaxValue).	-
352	R_LC_SET_WOBBEL_VECTOR	DOUBLE dTrans DOUBLE dLong UINT Period DOUBLE dPower	set_wobbel_vector(dTrans, dLong, Period, dPower).	-
353	R_LC_SET_WOBBEL_OFFSET	LONG OffsetTrans LONG OffsetLong	set_wobbel_offset(OffsetTrans, OffsetLong).	-
354	R_LC_SET_SKY_WRITING_PARA	DOUBLE Timelag LONG LasrOnShift UINT Nprev UINT Npost	set_sky_writing_para_list(Timelag, LaserOnShift, Nprev, Npost). Für UINT Nprev = UINT Npost = 65.535: set_sky_writing_list(Timelag, LaserOnShift).	-
355	R_LC_SET_SKY_WRITING_ANGLE	DOUBLE Limit	set_sky_writing_limit_list(Limit).	-
356	R_LC_SET_SKY_WRITING_MODE	UINT Mode	set_sky_writing_mode_list(Mode).	-
357	R_LC_SET_CONTROL_MODE	UINT Mode	set_control_mode_list(Mode).	-
358	R_LC_STORE_ENCODER	UINT Pos	store_encoder(Pos).	-
359	R_LC_START_TIMER	-	save_and_restart_timer().	-
360	R_LC_STORE_TIMESTAMP	-	store_timestamp_counter().	-
361	R_LC_WAIT_FOR_TIMESTAMP	UINT Counter UINT Mode	wait_for_timestamp_counter_mode(TimeStampCounter, Mode). Für UINT Mode > 2: wait_for_timestamp_counter(TimeStampCounter).	-
362	R_LC_SET_TRIGGER	UINT Period UINT Signal1 UINT Signal2 UINT Signal3 UINT Signal4	set_trigger4(Period, Signal1, Signal2, Signal3, Signal4). Für UINT Signal3 und/oder UINT Signal4 unzulässig: set_trigger(Period, Signal1, Signal2).	-
363	R_LC_EXTERN_START_DELAY	LONG Delay UINT EncoderNo	set_ext_start_delay_list(Delay, EncoderNo).	-
364	R_LC_SET_FLY	DOUBLE Scale UINT Axis UINT No	Für UINT Axis = 1 und UINT No = egal: set_fly_x(ScaleX) mit ScaleX = DOUBLE Scale. Für UINT Axis = 2 und UINT No = egal: set_fly_y(ScaleY) mit ScaleY = DOUBLE Scale. Für UINT Axis = 3 und UINT No = EncoderNo: set_fly_z(ScaleZ, EncoderNo) mit [ScaleZ = DOUBLE Scale, UINT No] und EncoderNo = UINT No. Für UINT Axis = 4 und UINT No = egal: set_fly_rot(Resolution) mit Resolution = DOUBLE Scale.	-
365	R_LC_ROT_CENTER	LONG X LONG Y	set_rot_center_list(X, Y).	-
366	R_LC_SET_FLY_2D	DOUBLE ScaleX DOUBLE ScaleY	set_fly_2d(ScaleX, ScaleY).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
367	R_LC_FLY_RETURN_Z	LONG X LONG Y LONG Z	fly_return_z(X, Y, Z). Für LONG Z = 0x7FFFFFFF: fly_return(X, Y).	-
368	R_LC_SET_FLY_LIMITS	LONG Xmin LONG Xmax LONG Ymin LONG Ymax	set_fly_limits(Xmin, Xmax, Ymin, Ymax).	-
369	R_LC_SET_FLY_LIMITS_Z	LONG Zmin LONG Zmax	set_fly_limits_z(Zmin, Zmax).	-
370	R_LC_IF_FLY_OVERFLOW	LONG Mode UINT Axis	Für UINT Axis = 1: if_fly_x_overflow(Mode). Für UINT Axis = 2: if_fly_y_overflow(Mode). Für UINT Axis = 3: if_fly_z_overflow(Mode).	-
371	R_LC_IF_NOT_FLY_OVERFLOW	LONG Mode UINT Axis	Für UINT Axis = 1: if_not_fly_x_overflow(Mode). Für UINT Axis = 2: if_not_fly_y_overflow(Mode). Für UINT Axis = 3: if_not_fly_z_overflow(Mode).	-
372	R_LC_CLEAR_FLY_OVERFLOW	UINT Mode	clear_fly_overflow(Mode).	-
373	R_LC_PARK_POSITION	UINT Mode LONG X LONG Y	park_position(Mode, X, Y).	-
374	R_LC_PARK_RETURN	UINT Mode LONG X LONG Y	park_return(Mode, X, Y).	-
375	R_LC_ACTIVATE_FLY_ENC_XY	DOUBLE ScaleX DOUBLE ScaleY LONG EncX LONG EncY	activate_fly_xy_encoder(ScaleX, ScaleY, EncX, EncY). Für LONG EncX = LONG EncY = 0: activate_fly_xy(ScaleX, ScaleY).	-
376	R_LC_ACTIVATE_FLY_ENC_2D	DOUBLE ScaleX DOUBLE ScaleY LONG EncX LONG EncY	activate_fly_2d_encoder(ScaleX, ScaleY, EncX, EncY). Für LONG EncX = LONG EncY = 0: activate_fly_2d(ScaleX, ScaleY).	-
377	R_LC_WAIT_FOR_ENCODER_MODE	LONG Value UINT Encoder UINT Mode	wait_for_encoder_mode(Value, EncoderNo, Mode). UINT Mode = 0: wait_for_encoder(Value, EncoderNo).	-
378	R_LC_WAIT_FOR_ENCODER_RANGE	LONG EncXmin LONG EncXmax LONG EncYmin LONG EncYmax UINT Mode	wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, Mode). Für UINT Mode = 0: wait_for_encoder_in_range(EncXmin, EncXmax, EncYmin, EncYmax).	-
379	R_LC_IF_NOT_ACTIVATED	-	if_not_activated().	-
380	R_LC_RANGE_CHECKING	UINT Head UINT Mode UINT Data	range_checking(HeadNo, Mode, Data).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
381	R_LC_SET_FLY_POS	DOUBLE Scale UINT Axis	Für UINT Axis = 1: set_fly_x_pos(ScaleX) mit ScaleX = DOUBLE Scale. Für UINT Axis = 2: set_fly_y_pos(ScaleY) mit ScaleY = DOUBLE Scale. Für UINT Axis = 4: set_fly_rot_pos(Resolution) mit DOUBLE Scale = Resolution.	-
382	R_LC_WAIT_FOR_MCBSP	UINT Axis LONG Value LONG Mode	wait_for_mcbsp(Axis, Value, Mode).	-
383	R_LC_SET_MCBSP_IN	UINT Mode DOUBLE Scale	set_mcbsp_in_list(Mode, Scale).	-
384	R_LC_SET_MULTI_MCBSP_IN	UINT Ctrl UINT P UINT Mode	set_multi_mcbsp_in_list(Ctrl, P, Mode).	-
385	R_LC_SET_MCBSP_MODE	UINT Axis DOUBLE Scale	Für UINT Axis = 1: set_mcbsp_x_list(Scale). Für UINT Axis = 2: set_mcbsp_y_list(Scale). Für UINT Axis = 4: set_mcbsp_rot_list(Resolution)mit Resolution = DOUBLE Scale). Für UINT Axis = 0 und DOUBLE Scale = egal: set_mcbsp_matrix_list().	-
386	R_LC_APPLY_MCBSP	UINT HeadNo UINT at_once	apply_mcbsp_list(HeadNo, at_once).	-
387	R_LC_SET_MCBSP_MODE_GLOBAL	UINT Axis DOUBLE Scale	Für UINT Axis = 1: set_mcbsp_global_x_list(Scale). Für UINT Axis = 2: set_mcbsp_global_y_list(Scale). Für UINT Axis = 4: set_mcbsp_global_rot_list(Resolution) mit Resolution = DOUBLE Scale. Für UINT Axis = 0 und DOUBLE Scale = egal: set_mcbsp_global_matrix_list().	-
388	R_LC_SET_FLY_1	UINT Axis UINT Mode DOUBLE Scale	set_fly_1_axis(Axis, Mode, Scale).	-
389	R_LC_FLY_RETURN_1	UINT Axis LONG RetPos1	fly_return_1_axis(Axis, RetPos1).	-
390	R_LC_WAIT_FOR_1	LONG Value UINT Mode LONG WaitMode UINT LaserMode	wait_for_1_axis(Value, Mode, WaitMode, LaserMode).	-
391	R_LC_ACTIVATE_FLY_1	UINT Axis UINT Mode DOUBLE Scale LONG Offset	activate_fly_1_axis(Axis, Mode, Scale, Offset).	-
392	R_LC_PARK_POSITION_1	UINT Mode UINT Axisv LONG ParkPos	park_position_1_axis(Mode, Axis, ParkPos).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
393	R_LC_PARK_RETURN_1	UINT Mode UINT Axis LONG RetPos	park_return_1_axis(Mode, Axis, RetPos).	-
394	R_LC_SET_FLY_2	UINT Axis1 UINT Mode1 DOUBLE Scale1 UINT Axis2 UINT Mode2 DOUBLE Scale2	set_fly_2_axes(Axis1, Mode1, Scale1, Axis2, Mode2, Scale2).	-
395	R_LC_FLY_RETURN_2	UINT Axis2 LONG RetPos1 UINT Axis2 LONG RetPos2	set_fly_2_axes(Axis1, Mode1, Scale1, Axis2, Mode2, Scale2).	-
396	R_LC_WAIT_FOR_2	UINT ModeX LONG MinValueX LONG MaxValueX UINT ModeY LONG MinValueY LONG MaxValueY LONG WaitMode UINT LaserMode	wait_for_2_axes(ModeX, MinValueX, MaxValueX, ModeY, MinValueY, MaxValueY, WaitMode, LaserMode).	-
397	R_LC_ACTIVATE_FLY_2	UINT ModeX DOUBLE ScaleX LONG OffsetX UINT ModeY DOUBLE ScaleY LONG OffsetY	activate_fly_2_axes(ModeX, ScaleX, OffsetX, ModeY, ScaleY, OffsetY).	-
398	R_LC_PARK_POSITION_2	UINT Mode LONG ParkPosX LONG ParkPosY	park_position_2_axes(Mode, ParkPosX, ParkPosY).	-
399	R_LC_PARK_RETURN_2	UINT Mode LONG RetPosX LONG RetPosY	park_return_2_axes(Mode, RetPosX, RetPosY).	-
400	R_LC_SET_FLY_3	UINT ModeX DOUBLE ScaleX UINT ModeY DOUBLE ScaleY UINT ModeZ DOUBLE ScaleZ	set_fly_3_axes(ModeX, ScaleX, ModeY, ScaleY, ModeZ, ScaleZ).	-
401	R_LC_FLY_RETURN_3	LONG RetPosX LONG RetPosY LONG RetPosZ	fly_return_3_axes(RetPosX, RetPosY, RetPosZ).	-
402	R_LC_RS232_WRITE_TEXT	CHAR Text[N] siehe 2, Seite 1053	rs232_write_text_list(pData) mit pData = CHAR Text[N].	-
403	R_LC_SET_MCBSP_OUT	UINT Signal1 UINT Signal2	set_mcbsp_out(Signal1, Signal2).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
404	R_LC_PERIODIC_TOGGLE	UINT Port UINT Mask UINT P1 UINT P2 UINT Count UINT Stop	periodic_toggle_list(Port, Mask, P1, P2, Count, Start).	-
405	R_LC_SET_FREE_VAR	UINT VarNo UINT Value	set_free_variable_list(No, Value) mit No = UINT VarNo.	-
406	R_LC_SELECT_COR_TABLE	UINT HeadA UINT HeadB	select_cor_table_list(HeadA, HeadB).	-
407	R_LC_SET_DELAY_MODE	UINT VarPoly UINT DirectMove3D UINT EdgeLevel UINT MinJumpDelay UINT JumpLengthLimit	set_delay_mode_list(VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit).	-
408	R_LC_SCANAHEAD_ACTIVATE_AUTO	LONG Mode	activate_scanahead_autodelays_list(Mode).	-
409	R_LC_SET_SCANAHEAD_LASER_SHIFT	LONG dLasOn LONG dLasOff	set_scanahead_laser_shifts_list(dLasOn, dLasOff).	-
410	R_LC_SET_SCANAHEAD_LINE_PARAMS	UINT CornerScale UINT EndScale UINT AccScale UINT JumpScale	Reserviert.	-
411	R_LC_STEPPER_ENABLE	LONG Enable1 LONG Enable2	stepper_enable_list(Enable1, Enable2).	-
412	R_LC_STEPPER_CTRL	LONG Period1 LONG Period2	stepper_control_list(Period1, Period2).	-
413	R_LC_STEPPER_NO_ABS	UINT No LONG Pos	stepper_abs_no_list(No, Pos).	-
414	R_LC_STEPPER_NO_REL	UINT No LONG dPos	stepper_rel_no_list(No, dPos).	-
415	R_LC_STEPPER_ABS	LONG Pos1 LONG Pos2	stepper_abs_list(Pos1, Pos2).	-
416	R_LC_STEPPER_REL	LONG dPos1 LONG dPos2	stepper_rel_list(dPos1, dPos2).	-
417	R_LC_STEPPER_WAIT	UINT No	stepper_wait(No).	-
418	R_LC_CAMMING	UINT FirstPos UINT NPos UINT EncoderNo UINT Ctrl DOUBLE Scale	camming (FirstPos, NPos, EncoderNo, Ctrl, Scale, Code).	-
419	R_LC_SET_JUMP_MODE	LONG Flag	set_jump_mode_list(Flag).	-
420	R_LC_ENDURING_WOBBEL	-	Reserviert.	-
421	R_LC_FLY_PREDICTION	UINT PredictionX UINT PredictionY	fly_prediction(PredictionX, PredictionY).	-

Remote-Listenbefehle, Seite 1042

ID	Remote-Listenbefehl	Befehls-Telegramm-Parameter	Funktionalität	Antwort-Telegramm-Parameter
422	R_LC_WAIT_FOR_TIMESTAMP_LONG	UINT WaitCounterL UINT WaitCounterH UINT MaxWaitTime UINT Mode	wait_for_timestamp_counter_long(WaitCounterL, WaitCounterH, MaxWaitTime, Mode).	-
423	R_LC_SET_TRIGGER_8	UINT Period UINT Signals[8]	set_trigger8(Period, Signal1, Signal2, Signal3, Signal4, Signal5, Signal6, Signal7, Signal8).	-
424	R_LC_SET_WOBBEL_VECTOR_2	DOUBLE dTrans DOUBLE dLong UINT Period DOUBLE dPower DOUBLE dPower2 UINT Ctrl	set_wobbel_vector_2(dTrans, dLong, Period, dPower, dPower2, Ctrl).	-
425	R_LC_SET_MCBSP_OUT_PTR	UINT Num UINT SignalPtr[Num]	set_mcbsp_out_ptr_list(Number, SignalPtr).	-
426	R_LC_SET_MCBSP_OUT_OIE	UINT Signal1 UINT Signal2	set_mcbsp_out_oie_list(Signal1, Signal2).	-
427	R_LC_WRITE_PORT	UINT Port UINT Value UINT NoDelay	write_port_list(Port, Value, NoDelay).	-
428	R_LC_FLY_DISABLE	-	fly_disable_list( ).	-
429	R_LC_INIT_FLY_2D	LONG OffsetX LONG OffsetY UINT No	init_fly_2d_list(OffsetX, OffsetY, No).	-
430	R_LC_SET_DEFOCUS_2	LONG Shift UINT Mode	set_defocus_2_list(Shift, Mode).	-
431	R_LC_SET_SKY_WRITING_MIN_SPEED	DOUBLE MinSpeed	set_sky_writing_min_speed_list(Speed)	-
432	R_LC_SET_SKY_WRITING_SCALE	DOUBLE Scale	set_sky_writing_scale_list(Scale)	-
…	…	…	…	…
511	-	-	-	-

Remote-Listenbefehle, Seite 1042

Hinweise

(1) Zu 341 R_LC_MARK_TEXT, CHAR Text[N].

Sie müssen N so wählen, dass die Telegramm-Gesamtlänge TGM_MAX_SIZE nicht überschreitet. Verteilen Sie gegebenenfalls den Text auf mehrere Remote-Befehle. Sie müssen jeden der Texte mit “\0” abschließen.

(2) Zu 402 R_LC_RS232_WRITE_TEXT, CHAR Text[N].

Sie müssen N so wählen, dass die Telegramm-Gesamtlänge TGM_MAX_SIZE nicht überschreitet. Verteilen Sie gegebenenfalls den Text auf mehrere Remote-Befehle. Sie müssen jeden der Texte mit “\0” abschließen.

16.10.10Nicht als Remote-Befehl verfügbare RTC6-Befehle

RTC6-Befehl	Begründung, Alternative, Kommentar
acquire_rtc	Im Remote-Interface-Modus nicht relevant: Zugriffsrechte.
auto_cal	Im Remote-Interface-Modus nicht relevant: Nur für einige abgekündigte Systeme.
copy_dst_src	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
create_dat_file	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
eth_assign_card	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
eth_assign_card_ip	Im Remote-Interface-Modus nicht relevant: RTC6-Kartenverwaltung. Bei Bedarf: Ist durch den Benutzer selber zu implementieren.
eth_boot_dcmd	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.
eth_boot_timeout	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.
eth_check_connection	Ist durch den Benutzer selber zu implementieren. Sende beliebigen Kontrollbefehl und prüfe Antwort.
eth_convert_ip_to_string	Bei Bedarf: Ist durch den Benutzer selber zu implementieren.
eth_convert_string_to_ip	Bei Bedarf: Ist durch den Benutzer selber zu implementieren.
eth_count_cards	Im Remote-Interface-Modus nicht relevant: RTC6-Kartenverwaltung. Bei Bedarf: Ist durch den Benutzer selber zu implementieren.
eth_found_cards	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
eth_full_reset	–
eth_get_card_info	Im Remote-Interface-Modus nicht relevant: RTC6-Kartenverwaltung.
eth_get_card_info_search	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
eth_get_com_timeouts	Veraltet.
eth_get_com_timeouts_auto	Ist durch den Benutzer selber zu implementieren. Siehe im C++-Remote-Interface-Wrapper: **eth_set_com_timeouts_auto**.
eth_get_error	Ist durch den Benutzer selber zu implementieren. Siehe im C++-Remote-Interface-Wrapper: **get_network_error**.

RTC6-Befehl (Forts.)	Begründung, Alternative, Kommentar (Forts.)
eth_get_ip	Im Remote-Interface-Modus nicht relevant: RTC6-Kartenverwaltung.
eth_get_ip_search	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
eth_get_last_error	Ist durch den Benutzer selber zu implementieren. Siehe im C++-Remote-Interface-Wrapper: **get_network_error**.
eth_get_serial_search	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
eth_get_standalone_status	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.
eth_max_card	Im Remote-Interface-Modus nicht relevant: RTC6-Kartenverwaltung. Bei Bedarf: Ist durch den Benutzer selber zu implementieren.
eth_remove_card	Im Remote-Interface-Modus nicht relevant: RTC6-Kartenverwaltung. Bei Bedarf: Ist durch den Benutzer selber zu implementieren.
eth_search_cards	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
eth_search_cards_range	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
eth_set_com_timeouts	Veraltet.
eth_set_com_timeouts_auto	Ist durch den Benutzer selber zu implementieren. Siehe im C++-Remote-Interface-Wrapper: **eth_set_com_timeouts_auto**.
eth_set_high_performance_mode	Bei Bedarf: Ist durch den Benutzer selber zu implementieren.
eth_set_remote_tgm_format	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.
eth_set_search_cards_timeout	Im Remote-Interface-Modus nicht relevant: Suche nach RTC6-Karten.
free_rtc6_dll	Im Remote-Interface-Modus nicht relevant: RTC6-DLL.
get_auto_cal	Im Remote-Interface-Modus nicht relevant: Nur für einige abgekündigte Systeme.
get_card_type	Im Remote-Interface-Modus nicht relevant: nur RTC6 Ethernet-Karten sind unterstützt.
get_char_pointer	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.

RTC6-Befehl (Forts.)	Begründung, Alternative, Kommentar (Forts.)
get_dll_version	Im Remote-Interface-Modus nicht relevant: RTC6-DLL.
get_error	Ist durch den Benutzer selber zu implementieren. Siehe im C++-Remote-Interface-Wrapper: **get_error**.
get_hi_data	Im Remote-Interface-Modus nicht relevant: Nur für einige abgekündigte Systeme.
get_hi_pos	Im Remote-Interface-Modus nicht relevant: Nur für einige abgekündigte Systeme.
get_jump_table	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
get_last_error	Ist durch den Benutzer selber zu implementieren. Siehe im C++-Remote-Interface-Wrapper: **get_last_error**.
get_mcbsp_list	Ohne Funktion.
get_out_pointer	**get_status**
get_rtc_mode	Bei Bedarf: Ist durch den Benutzer selber zu implementieren. Zusätzlich: Bei jedem RTC6-Befehl ist die Zeile RTC4→RTC6 oder RTC5→RTC6 zu beachten.
get_sub_pointer	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
get_text_table_pointer	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
get_transform	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
get_transform_offset	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
init_rtc6_dll	Im Remote-Interface-Modus nicht relevant: RTC6-DLL.
load_auto_laser_control	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
load_disk	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
load_fly_2d_table	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
load_jump_table	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.

RTC6-Befehl (Forts.)	Begründung, Alternative, Kommentar (Forts.)
load_jump_table_offset	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
load_position_control	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
load_program_file	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022 und deswegen bereits ausgeführt worden.
load_stretch_table	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
load_varpolydelay	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
load_zoom_correction_file	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
move_to	Im Remote-Interface-Modus nicht relevant: Spezialbefehl.
read_abc_from_file	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
read_abc_from_file_20b	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
read_image_eth	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.
read_user_data	Ohne Funktion.
regulation3	Im Remote-Interface-Modus nicht relevant: Spezialbefehl.
release_rtc	Im Remote-Interface-Modus nicht relevant: Zugriffsrechte.
reset_error	Ist durch den Benutzer selber zu implementieren. Siehe im C++-Remote-Interface-Wrapper: **reset_error**.
rs232_config	uart_config
rtc6_count_cards	Im Remote-Interface-Modus nicht relevant: nur RTC6 Ethernet-Karten werden unterstützt.
save_disk	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
select_rtc	Im Remote-Interface-Modus nicht relevant: Zugriffsrechte. Im Remote-Interface-Modus nicht relevant: Multi-Board-Befehle.

RTC6-Befehl (Forts.)	Begründung, Alternative, Kommentar (Forts.)
send_user_data	Ohne Funktion.
set_char_pointer	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
set_char_table	**set_text_table_pointer**
set_dsp_mode	Im Remote-Interface-Modus nicht relevant: Nur für einige abgekündigte Systeme.
set_duty_cycle_table	Im Remote-Interface-Modus nicht relevant: Spezialbefehl.
set_eth_boot_control	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.
set_jump_table	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
set_laser_timing	**set_laser_pulses**
set_laser_timing_table	Im Remote-Interface-Modus nicht relevant: Spezialbefehl.
set_list_jump	**list_jump_pos**
set_rtc4_mode	Bei Bedarf: Ist durch den Benutzer selber zu implementieren. Zusätzlich: Bei jedem RTC6-Befehl ist die Zeile RTC4→RTC6 zu beachten.
set_rtc5_mode	Bei Bedarf: Ist durch den Benutzer selber zu implementieren. Zusätzlich: Bei jedem RTC6-Befehl ist die Zeile RTC5→RTC6 zu beachten.
set_rtc6_mode	Ist im Remote-Interface-Modus der Standardfall.
set_softstart_level	Ohne Funktion.
set_softstart_level_list	Ohne Funktion.
set_softstart_mode	Ohne Funktion.
set_softstart_mode_list	Ohne Funktion.

RTC6-Befehl (Forts.)	Begründung, Alternative, Kommentar (Forts.)
set_sub_pointer	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
set_text_table_pointer	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus ist RTC6-Listenspeicher-Fragmentierung zu vermeiden.
store_program	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.
sync_slaves	Ohne Funktion.
transform	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
upgrade_bios	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem.
upload_transform	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
verify_checksum	Im Remote-Interface-Modus nicht relevant: Zugriffe auf Dateisystem. Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022 (dort prüft **load_program_file** die Check-Summe).
write_abc_to_file	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
write_abc_to_file_20b	Ist durch den Benutzer selber zu implementieren. Mit z.B. CalibrationLibrary.
write_hi_pos	Im Remote-Interface-Modus nicht relevant: Nur für einige abgekündigte Systeme.
write_image_eth	Im Remote-Interface-Modus nicht relevant: Standalone-Betriebsmodus ist Voraussetzung, Seite 1022.

16.10.11 Hinweise zur Softwareentwicklung

- Beim Start des Anwenderprogramms sollte die initiale `seqnum` festgelegt werden(1). Senden Sie dazu ein Befehls-Telegramm mit 0x12345678 als 1. `uint32_t`-Wert in TGM_PL_CMD_RAW. Die RTC6 Ethernet-Karte liefert dann die zuletzt ausgeführte `seqnum` zurück.

(1) Im C++-Remote-Interface-Wrapper wird die initiale `seqnum` automatisch festgelegt.

16.10.12Über den C++-Remote-Interface-Wrapper

Die Gesamt-Funktionalität aus

- `rtc6_rif_wrapper.cpp`
  (C++-Remote-Interface-Wrapper-Source-Datei. Enthält eine abstrakte Klasse, in der das Senden und Empfangen von Befehls-Telegrammen implementiert ist)
- `rtc6_rif_wrapper.h`
  (C++-Remote-Interface-Wrapper-Header-Datei)
- `telegrams.h`
  (Header-Datei für `rtc6_rif_wrapper.h`. Definiert die Struktur der Befehls-Telegramme sowie die Remote-Befehl-IDs)

wird in diesem Dokument bezeichnet als:

- C++-Remote-Interface-Wrapper

Der C++-Remote-Interface-Wrapper bietet die grundlegende Logik an für:

- Das Erzeugen von Befehls-Telegrammen
- Das Senden(1)(2) von Befehls-Telegrammen
- Das Empfangen von Befehls-Telegrammen
- Die Behandlung von Wiederholungsversuchen (Retry-Handling)
- Eine einfache Fehlererkennung

Damit das Senden und Empfangen von Befehls-Telegrammen funktioniert, müssen Sie eine plattformspezifische Implementierung für folgende virtuelle Socket-Funktionen bereitstellen, siehe Abbildung 93 sowie die Beispiel-Implementierung `example_1.cpp` aus dem RTC6-Software-Paket:

- `send`
- `recv`
- `select`

(1) Befehls-Telegramme müssen über UDP an die RTC6 Ethernet-Karte gesendet werden. Der zu verwendende Port ist der gleiche wie der der herkömmlichen Übertragung (**eth_get_port_numbers**(`UDPexcl`), **eth_set_port_numbers**(`UDPexcl`)).

(2) Nur 1 Listenbefehl (nicht mehrere) je Befehls-Telegramm.

Der C++-Remote-Interface-Wrapper bietet eine ähnliche Schnittstelle an wie `RTC6impl.h` und `RTC6impl.hpp`.

Die Signaturen der Wrapper-Funktionen (in `rtc6_rif_wrapper.h`) und RTC6-DLL sind weitgehend gleich.

Nur im C++-Remote-Interface-Wrapper stehen zur Verfügung:

- Wrapper-Funktion **get_remote_error**
- Wrapper-Funktion **disable_remote_interface**
- Wrapper-Funktion **get_network_error**

Nicht im C++-Remote-Interface-Wrapper stehen zur Verfügung:

- RTC6-Befehl **eth_get_last_error**
- RTC6-Befehl **eth_get_error**

  Ersatz ist die Wrapper-Funktion get_network_error mit 2 vordefinierten Fehlercodes, die vom C++-Remote-Interface-Wrapper gesetzt werden können:
  - `NETWORK_ERR_TIMEOUT`
    Es ist ein Timeout beim Empfang eines Antwort-Telegramms aufgetreten, siehe auch **eth_set_com_timeouts_auto**.
  - `NETWORK_ERR_WRONG_ID`
    Im Antwort-Telegramm und Befehls-Telegramm stimmen die Wrapper-Funktions-IDs nicht überein.

  Die restlichen Fehlercodes können Sie selber definieren und entsprechen dem letzten Rückgabewert von `send` / `recv` / `select`.

Hinweise

- Im C++-Remote-Interface-Wrapper wird die initiale `seqnum` automatisch festgelegt.

So benutzen Sie den C++-Remote-Interface-Wrapper:

(1) Erstellen Sie eine abgeleitete Klasse von NetworkAdapter.

(2) Erstellen Sie einen Datagrammsocket (UDP) mit der IP-Adresse der gewünschten RTC6 Ethernet-Karte und Port 63750[1].

(3) Implementieren Sie die plattformspezifische Socket-Funktionalität in den virtuellen Funktionen `send` / `recv` / `select`.

(4) Erstellen Sie ein RTC-Objekt und übergeben Sie die NetworkAdapter-Implementierung

```
// Pseudo-Code (nicht vollständig).

// Senden der angegebenen Anzahl von Bytes.
// Bei Erfolg wird 0 zurückgegeben, andernfalls ein negativer Fehlercode.
   virtual int32_t send(const char* buf, size_t len) = 0;

// Empfängt bis zu len Bytes und schreibt sie in buf.
// Bei Erfolg wird 0 zurückgegeben, andernfalls ein negativer Fehlercode.
// Wenn das empfangene Telegramm verworfen werden soll, wird ein positiver Wert zurückgegeben.
   virtual int32_t recv(char* buf, size_t len) = 0;

// Ermitteln des Lesestatus. Rückgabe der Anzahl der verfügbaren Lesevorgänge.
// Rückgabe 0, wenn nach timeout_us Mikrosekunden keine Daten verfügbar sind,
// andernfalls ein negativer Fehlercode.
   virtual int32_t select(long timeout_us) = 0;
```

93

Siehe Schritt 3 sowie Erläuterungen, Seite 1061.

16.11Rechtliches

16.11.1EU-Konformitätserklärung – RTC6 Ethernet-Karte

EU-Konformitätserklärung

für das Produkt:

RTC4 Ethernet, RTC6 Ethernet

Der Hersteller

SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Deutschland

erklärt, dass das genannte Produkt die einschlägigen Harmonisierungsrechtsvorschriften der Union erfüllt:

- **2011/65/EU -** Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Beschränkung der Verwendung bestimmter gefährlicher Stoffe in Elektro- und Elektronikgeräten (RoHS-Richtlinie).

Folgende harmonisierte Normen wurden angewandt:

- **DIN EN IEC 63000:2019-05** - Technische Dokumentation zur Beurteilung von Elektro- und Elektronikgeräten hinsichtlich der Beschränkung gefährlicher Stoffe

Die alleinige Verantwortung für die Ausstellung dieser Konformitätserklärung trägt der Hersteller.

Dieses Dokument zur Kundeninformation wurde elektronisch ausgestellt und ist daher ohne Unterschrift gültig. Eine unterschriebene Ausführung ist Bestandteil der SCANLAB-internen technischen Dokumentation.

EU Declaration of Conformity

for the product:

RTC4 Ethernet, RTC6 Ethernet

The manufacturer

SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Germany

declares that the product mentioned above complies with the relevant Union harmonization legislation:

- **2011/65/EU** - Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the restriction of the use of certain hazardous substances in electrical and electronic equipment (RoHS Directive).

The following harmonized standards have been applied:

- **DIN EN IEC 63000:2019-05** - Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances

This declaration of conformity is issued under the sole responsibility of the manufacturer.

This document, used for customer information, was issued electronically and is therefore valid without a signature. A signed version is part of the SCANLAB internal technical documentation.

SCANLAB GmbH
Puchheim

2021-07-19

© SCANLAB GmbH – 2021/07

16.11.2Konformität mit FCC-Bestimmungen

Die RTC6 Ethernet-Karte wurde getestet und entspricht den Grenzwerten für Digitalgeräte der Klasse A, gemäß Teil 15 der FCC-Bestimmungen.

Diese Grenzwerte sollen einen angemessenen Schutz gegen schädliche Störungen bieten, wenn die RTC6 Ethernet-Karte in einer kommerziellen Umgebung betrieben wird. Die RTC6 Ethernet-Karte erzeugt, verwendet und emittiert möglicherweise Hochfrequenzenergie und kann, wenn nicht in Übereinstimmung mit dieser Bedienungsanleitung installiert und verwendet, Störungen im Funkverkehr verursachen. Der Betrieb der RTC6 Ethernet-Karte in einem Wohngebiet wird wahrscheinlich schädliche Störungen verursachen; in diesem Fall ist der Benutzer verpflichtet, die Störungen auf eigene Kosten zu beseitigen.

16.11.3TI-RTOS

In der RTC6 Ethernet-Karte wird TI-RTOS verwendet.

TI-RTOS is licenced under the BSD licence:
Copyright (c) 2000-2018, Texas Instruments Incorporated
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Texas Instruments Incorporated nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

16.12Technische Spezifikationen – RTC6 Ethernet-Karte

Systemanforderungen

Windows-PC mit Ethernet-Schnittstelle (Gigabit-Ethernet (bevorzugt), 10/100 Mbit/s-Ethernet)

Betriebssystem	Microsoft Windows 10, 8, 7 als 32-Bit oder 64-Bit Version.

Abmessungen (ohne eingesteckte Steckverbinder)

Länge	120 mm
Breite	103 mm
Höhe	ca. 30 mm

Sonstige Anschlüsse und Spezifikationen

Versorgungs-spannungs	+11 V…+50 V
Maximale Leistungsaufnahme, ohne angeschlossene Peripheriegeräte	< 10 W

Um die Verlustleistung gering zu halten, sollte eine niedrige Eingangsspannung gewählt werden.

Betriebstemperatur

- mit natürlicher Konvektion: 10 °C…50 °C
- mit erzwungener Konvektion: 10 °C…60 °C

Der Benutzer hat für eine ausreichende Kühlung der RTC6 Ethernet-Karte zu sorgen.

Standalone-Funktionalität	Ja, siehe Kapitel 16.8 "Standalone-Funktionalität", Seite 1012.

Schnittstelle zum Netzwerk

Ethernet	Gigabit-Ethernet (bevorzugt), 10/100 Mbit/s-Ethernet
IP-Adressen-Bezug	• Durch einen DHCP-Server • Link-local Adresse (ab BIOS 24)[a] • Statische IP-Adresse (auf der Karte speicherbar)
Benötigte lokale Ports[b]	• 63749 (UDP) • 63750 (UDP, TCP)

(a) Siehe Seite 1002.

(b) Konfigurierbar mit **eth_set_port_numbers**.

Scan-System Ansteuerung

Zahl der RTC6-Listenspeicher-Bereiche	Bis zu 3, konfigurierbar
Gesamtkapazität des RTC6-Listenspeichers	8.388.608 Listenplätze
Ausgabeintervall der Mikroschritte	10 μs
Maximaler Wertebereich der Bildfeld-Koordinaten	–524.288…+524.287 (20 Bit mit Vorzeichen).
Virtuelles Bildfeld, z.B. für "Processing on the fly"	–268.435.456… +268.435.455 (28 Bit mit Vorzeichen = 29 Bit)[a]

(a) Mit RTC6-Software-Paket ≥ 1.4.0.

Schnittstellen

SCANHEADs-Stiftleiste

Anschluss	20-polige Stiftleiste mit Rastermaß 2,54 mm[(a)]. Die Signale für den ersten Scan-Kopf werden an Pin (01)…Pin (10) ausgegeben. Die Signale für den zweiten Scan-Kopf werden an Pin (11)…Pin (20) ausgegeben: • xy-Ausgabewerte nur bei freigeschalteter Option “Zweite-Scan-Kopf-Steuerung” • z-Ausgabewerte nur bei freigeschalteter Option “3D”
Signalübertragung	SL2-100-Protokoll. Über XY2-100-Konverter (Zubehör): XY2-100-Protokoll

(a) *Nicht* wie RTC6 PCIe-Karte.

LASER-Stiftleiste

Anschluss	16-polige Stiftleiste mit Rastermaß 2,54 mm[(a)]
Lasersteuersignale	LASER1, LASER2, LASERON
• TTL-Pegel	5 V, HIGH-aktiv oder LOW-aktivprogrammierbar
• Max. Last	20 mA
• Bezugspunkt	GND bei Karten ohne Option “DC/DC-Wandler”. GND2 bei Karten mit Option “DC/DC-Wandler”
Analoge Ausgänge	ANALOG OUT1, ANALOG OUT2[(b)]
• Ausgangsspannungsbereich	0 V…10 V
• Auflösung	12 Bit
• Max. Last	5 mA
• Bezugspunkt	GND
Digitaler Eingang	2 Bits
• LOW-Pegel	< 0,6 V
• HIGH-Pegel	> 2,3 V
• Max. Eingangsspannungsbereich	–0,5 V…+5,5 V
• Eingangswiderstand (Pull-up)	> 4,7 kΩ
• Bezugspunkt	GND
Digitaler Ausgang	2 Bits, gepuffert
• LOW-Pegel	< 0,55 V
• HIGH-Pegel	> 3,8 V
• Max. Last	20 mA
• Bezugspunkt	GND

(a) *Nicht* wie RTC6 PCIe-Karte.

(b) Bei RTC6 PCIe-Karten ist das ANALOG OUT2-Signal zusätzlich auch an der MARKING ON THE FLY-Stiftleiste verfügbar.

LASER-Stiftleiste (Forts.)

Eingänge für externe Start- und Stoppsignale	TTL LOW-aktiv, intern über Pull-up-Widerstände (4,7 kΩ) mit +3,3 V verbunden
• /START	Flanken-sensitiv
• /STOP	Pegel-sensitiv
• Bezugspunkt	GND
Sonstige Signale	
• BUSY OUT	5 V, TTL HIGH-aktiv, max. 10 mA, Bezugspunkt GND
• +5 V	Max. 100 mA
• Bezugspunkt	GND

EXTENSION 1-Stiftleiste

Anschluss	40-polige Stiftleiste mit Rastermaß 2,00 mm[(a)]. Ausgangssignal-Pegel einstellbar über Lötjumperfeld A
Digitaler Ausgang	16 Bits, gepuffert
• LOW-Pegel	< 0,4 V
• HIGH-Pegel	> 2,0 V (3,3 V oder 5 V)
• Max. Last	±8 mA
• Bezugspunkt	GND
• LATCH-Signal	3,3 V oder 5 V, TTL HIGH-aktiv, 5 µs-Puls, max. 10 mA
Digitaler Eingang	16 Bit, geschützt
• LOW-Pegel	< 0,5 V
• HIGH-Pegel	> 2,6 V…24 V
• Max. Eingangs-spannungsbereich	–0,5 V…+26 V
• Eingangswiderstand	> 10 kΩ
• Bezugspunkt	GND
• SYNC-Signal	3,3 V oder 5 V, TTL HIGH-aktiv, Rechtecksignal (5 µs-Puls, 10 µs-Periode), max. 10 mA
Sonstige Signale	
• BUSY OUT	3,3 V oder 5 V, TTL HIGH-aktiv, max. 10 mA
• VCC	3,3 V oder 5 V, max. 100 mA
• +5 V	max. 100 mA
• Bezugspunkt	GND

(a) Bei RTC6 PCIe-Karten: Rastermaß 2,54 mm.

EXT. 2-Stiftleiste

Anschluss	10-polige Stiftleiste mit Rastermaß 2,00 mm. Über Lötjumperfeld B ist Pin (09)[(a)], und Pin (08)[(b)] über Lötjumperfeld C einstellbar
Digitaler Ausgang	8 Bits, gepuffert
• LOW-Pegel	< 0,4 V
• HIGH-Pegel	> 2,0 V
• Max. Last	±8 mA
• Bezugspunkt	GND
• LATCH-Signal	5 V, TTL HIGH-aktiv, 5 µs-Puls, max. 10 mA
Lasersteuersignale	Keine[(c)(d)]
Sonstige Signale	
• +5 V	max. 100 mA
• Bezugspunkt	GND

(a) Pin (09) der RTC6 Ethernet-Karten = Pin (17) der RTC6 PCIe-Karten.

(b) Pin (08) der RTC6 Ethernet-Karten = Pin (15) der RTC6 PCIe-Karten.

(c) *Nicht* wie RTC6 PCIe-Karte.

(d) Bei RTC6 PCIe-Karten: LASER1 und LASER2.

MOF-Stiftleiste[(a)]

Anschluss	14-polige Stiftleiste mit Rastermaß 2,00 mm[(a)]
2 Encodereingänge für Inkremental-Encoder	ENCODER X(1±, 2±) und ENCODER Y(1±, 2±), jeweils für 1 Paar von standardisierten differentiellen Eingangssignalen (RS-422). HIGH-Pegel ≥ 2,0 V LOW-Pegel ≤ 0,8 V f ≤ 4 MHz
Analoger Ausgang	Keiner[(b)(c)]
Eingänge für externe Start- und Stoppsignale	/START2 /STOP2 Wie /START, /STOP der LASER-Stiftleiste, Seite 1066
Sonstige Signale	
• BUSY OUT	Identisch mit BUSY OUT der LASER-Stiftleiste, Seite 1066
• +5 V	Max. 100 mA
• Bezugspunkt	GND

(a) Bei RTC6 PCIe-Karten: 16-polige Stiftleiste mit Rastermaß 2,54 mm. Außerdem ist die Bedruckung auf der Karte MARKING ON THE FLY.

(b) *Nicht* wie RTC6 PCIe-Karte.

(c) Die RTC6 Ethernet-Karte hat an ihrer MOF-Stiftleiste keinen Pin für das ANALOG OUT2-Signal.

SPI/ANA/UART-Stiftleiste

Bei RTC6 Ethernet-Karten sind die Pins von McBSP/ANALOG-Stiftleiste und RS232-Stiftleiste der RTC6 PCIe-Karte in die SPI/ANA/UART-Stiftleiste integriert.

Anschluss	10-polige Stiftleiste mit Rastermaß 2,00 mm.
Analoge Eingänge	ANALOG IN0, ANALOG IN1
• Eingangsspannungsbereich	0 V…10 V
• Eingangsimpedanz	> 5 kΩ[a]
• ADC-Auflösung	12 Bit
• Bezugspunkt	GND[b]
Pins für RS-232	
Eingang	RxD
• Spannungsbereich	Max. –25 V…+25 V
Ausgang	TxD
• Spannungsbereich	Max. –13 V…+13 V
Bezugspunkt	GND
Baud-Rate	300…115.200

Pins für McBSP siehe auch Abschnitt "McBSP-Schnittstelle", Seite 87

• Sender-Signalpegel	3,3 V TTL
• Empfänger-Signalpegel	3,3 V oder 5 V TTL
• McBSP-Mode	Single Phase Frame Single Element per Frame 32 Bits per Element Daten-Delay `XDelay` Bits Daten-Delay `RDelay` Bits
• Bezugspunkt	GND
SPI-Schnittstellenfunktionalität	Nein.

(a) *Nicht* wie RTC6 PCIe-Karte.

(b) Siehe Hinweise, Seite 984.

STEPPER-Stiftleiste

Anschluss	10-polige Stiftleiste mit Rastermaß 2,00 mm.

Signale zur Ansteuerung von bis zu 2 Schrittmotoren:

• Ausgänge ENABLE1, ENABLE2, DIRECTION1, DIRECTION2	5 V, TTL
• Ausgänge CLOCK1, CLOCK2	5 V, TTL HIGH-aktiv, 5 µs-Puls
• Eingänge SWITCH1, SWITCH2	TTL LOW-aktiv, intern über Pull-up-Widerstände (10 kΩ) mit +3,3 V verbunden
• max. Last	25 mA
• Bezugspunkt	GND

17 Anhang B: UFPM-Erweiterungskarte

Die UFPM-Erweiterungskarte[1] wurde für RTC6 PCIe-Karten entwickelt[2].

Den Zusammenbau zeigt Abbildung 94, Abmessungen und Details Abbildung 95.

Die UFPM-Erweiterungskarte wandelt mittels eines schnellen Digital-Analog-Wandlers 8-Bit-Digital-Signale in Analogspannungswerte.

Sie wird empfohlen, wenn:

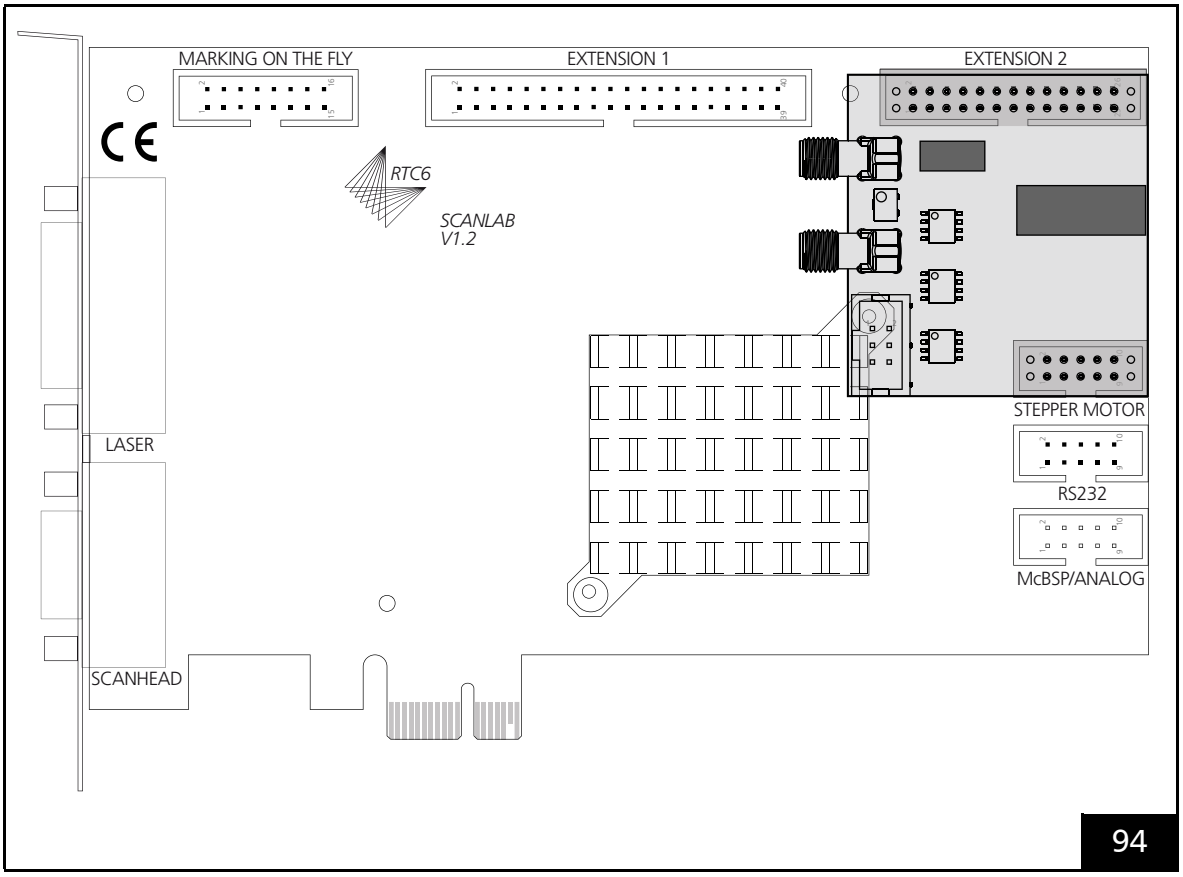
- *Analog* angesteuerte Laser betrieben werden *und*
- Pixelausgabefrequenzen ≥ 100 kHz[3][4] erreicht werden sollen (siehe **set_pixel_line**)

(1) Z.B. #0137980, #0140965, #0152803.

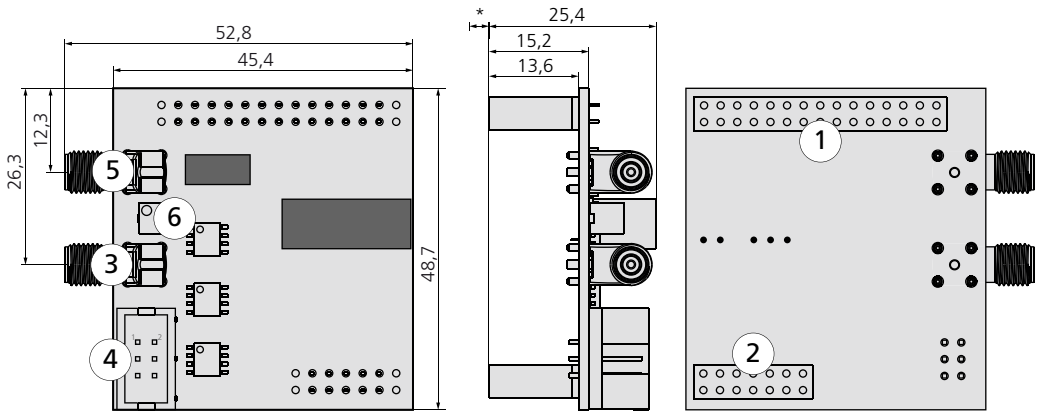
(2) Kann aus mechanischen Gründen nicht mit RTC6 Ethernet-Karten verwendet werden.

(3) Option "UFPM" für Pixelausgabefrequenzen 800 kHz…3,2 MHz erforderlich.

(4) Die UFPM-Erweiterungskarte unterstützt selbstverständlich auch Pixelausgabefrequenzen < 100 kHz.



UFPM-Erweiterungskarte: Zusammenbau mit der RTC6 PCIe-Karte.
Hinweis: Das Slotblech für Zweiten Scan-Kopf (Zubehör) (#0115132) kann nicht verwendet werden.



Legende

* 3 mm (Abstand zur RTC6-Leiterplattenoberfläche im montierten Zustand).

1 . . . .J1. 30-polige Buchse zur Verbindung mit der 26-poligen EXTENSION 2-Stiftleiste der RTC6 PCIe-Karte. Die äußeren 4 Pins von Anschluss J1 sind nicht verbunden und verhindern lediglich ein versetztes Aufstecken der Karte.

2 . . . .J2. 14-polige Buchse zur Verbindung mit der 10-poligen STEPPER MOTOR-Stiftleiste der RTC6 PCIe-Karte. Die Buchse J2 dient ausschließlich zur mechanischen Fixierung auf der RTC-Karte.

3 . . . .J3. SMA-Buchse mit Analog-Ausgang ANALOG OUTA.

4 . . . .J4. 6-polige Stiftleiste mit Analog-Ausgängen ANALOG OUTA und ANALOG OUTB.

5 . . . .J5. SMA-Buchse mit Analog-Ausgang ANALOG OUTB.

6 . . . .P3. Potentiometer zum Feinabgleich der 5 V-Ausgangsamplitude der Analog-Ausgänge ANALOG OUTA und ANALOG OUTB.

95

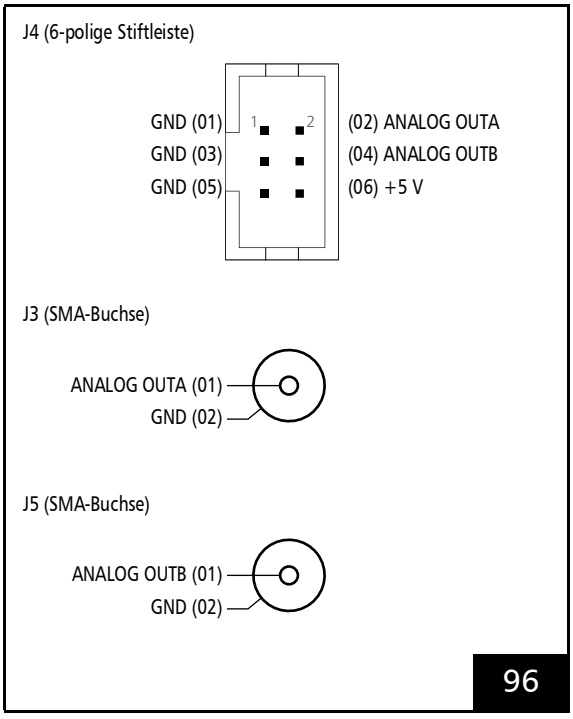
UFPM-Erweiterungskarte: Abmessungen und Details. Alle Maße in mm.

Mit der UFPM-Erweiterungskarte kann die Variation der Laserleistung von Pixel zu Pixel über Analogspannungswerte in 8-Bit Auflösung gesteuert werden. Ausgangsspannungsbereich: 0 V…5 V.

Dazu müssen die Ausgaben an den Port = 3 (8-Bit-Digital-Ausgang an der EXTENSION 2-Stiftleiste) gesendet werden, siehe **set_pixel_line**.

Für alle Pixelausgabemodi inklusive deren Erweiterungen ist die Laserpulsdauer vor Beginn der Pixelzeile über **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** mit mindestens 1/64 μs Dauer anzugeben (auch wenn die Pulsdauer nicht zur Laser-Steuerung verwendet wird). Anderenfalls werden an LASER1 keine Pulse ausgegeben.

Die UFPM-Erweiterungskarte passt die LATCH-Signaldauer an die Pixelfrequenz an (etwa `HalfPeriod`…5 µs). Sie wandelt 8-Bit-Digitalwerte synchron in entsprechende Analogwerte von 0 V…5 V um, und stellt diese an ihren Analog-Ausgängen ANALOG OUTA und ANALOG OUTB zur Verfügung. Die Pinbelegungen der Anschlüsse auf der UFPM-Erweiterungskarte zeigt Abbildung 96. Die exakte Ausgangsamplitude kann über das Potentiometer P3 feineingestellt werden, siehe Abbildung 95.



UFPM-Erweiterungskarte: Pinbelegungen der Anschlüsse.

Software-Voraussetzungen

- Die UFPM-Erweiterungskarte wird standardmäßig durch das RTC6-Software-Paket unterstützt.

Hardware-Voraussetzungen

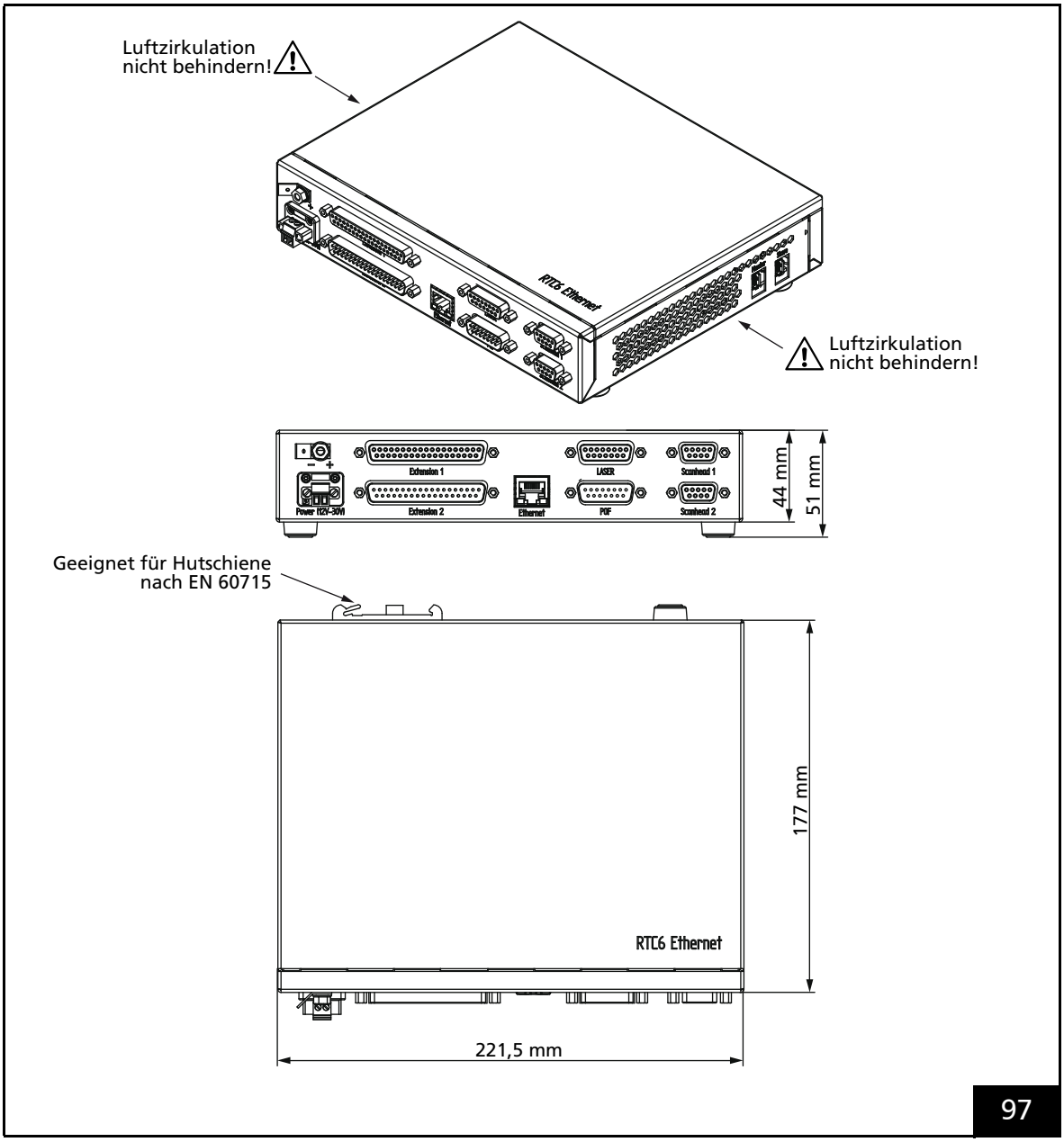
- Die Pixelwerte werden als 8-Bit-Digitalwerte am 8-Bit-Digital-Ausgang der EXTENSION 2-Stiftleiste der RTC6 PCIe-Karte ausgegeben. Damit die Pixelwerte (vollständig) sowie das Latch-Signal an der EXTENSION 2-Stiftleiste ausgegeben werden, müssen die folgenden Lötjumper auf der RTC6 PCIe-Karte geschlossen sein(1):
  - DATA7 im Lötjumperfeld C (= das DATA7-Signal wird an Pin 15 der EXTENSION 2-Stiftleiste ausgegeben)
  - LATCH im Lötjumperfeld B (= das LATCH-Signal wird an Pin 17 der EXTENSION 2-Stiftleiste ausgegeben)

Technische Spezifikationen

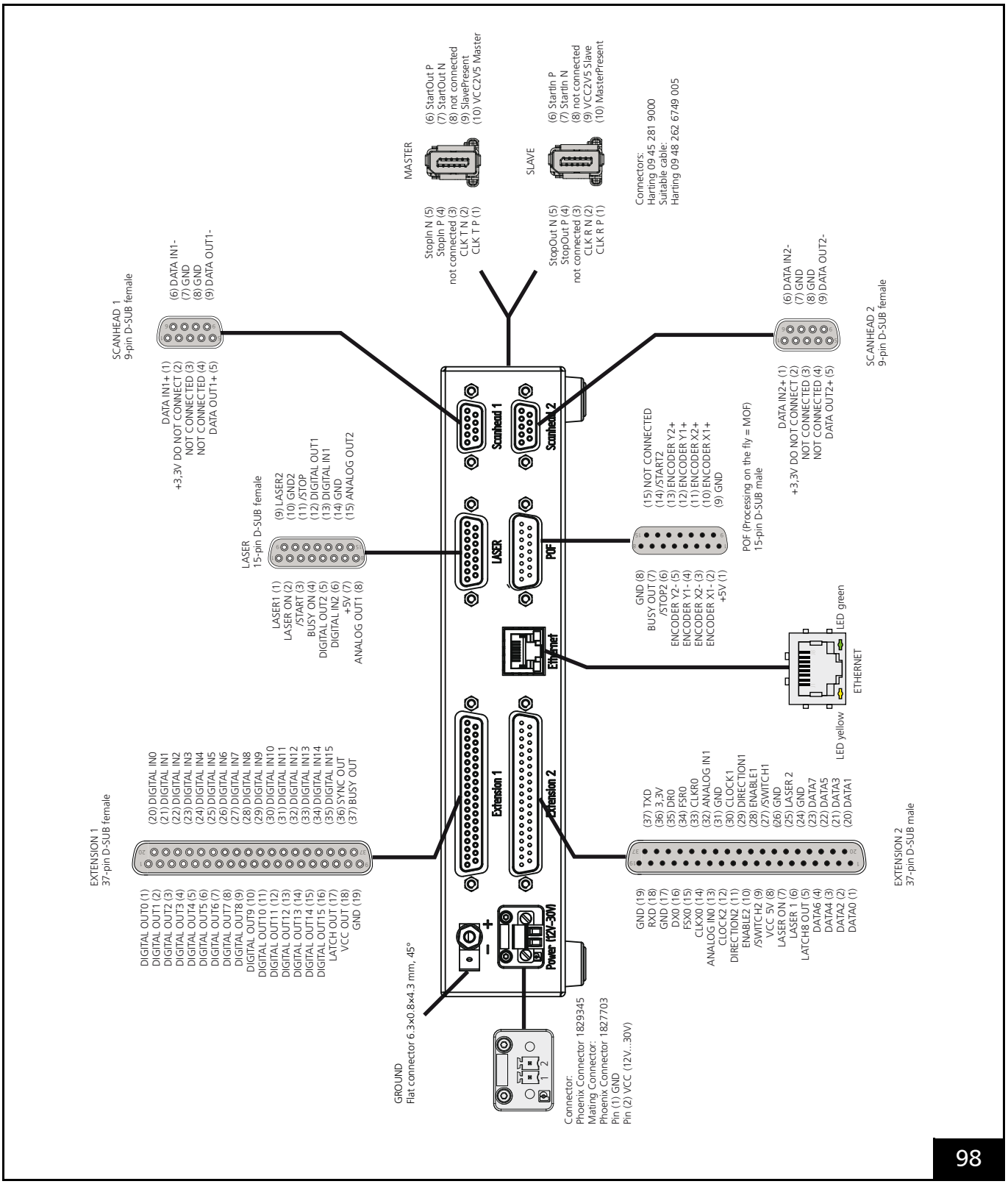
- Abmessungen
  - Länge 52,8 mm
  - Breite 48,7 mm
- Analoge Ausgänge ANALOG OUTA, ANALOG OUTB
  - Anschlüsse 1 6-polige Stiftleiste, 2 SMA-Buchsen (koaxiale Steckverbinder)
  - Ausgangsspannungsbereich 0 V…5 V
  - Auflösung 8 Bit
  - Max. Last 5 mA
  - Bezugspunkt GND

(1) Entspricht RTC6 PCIe-Karten TYPE n24.

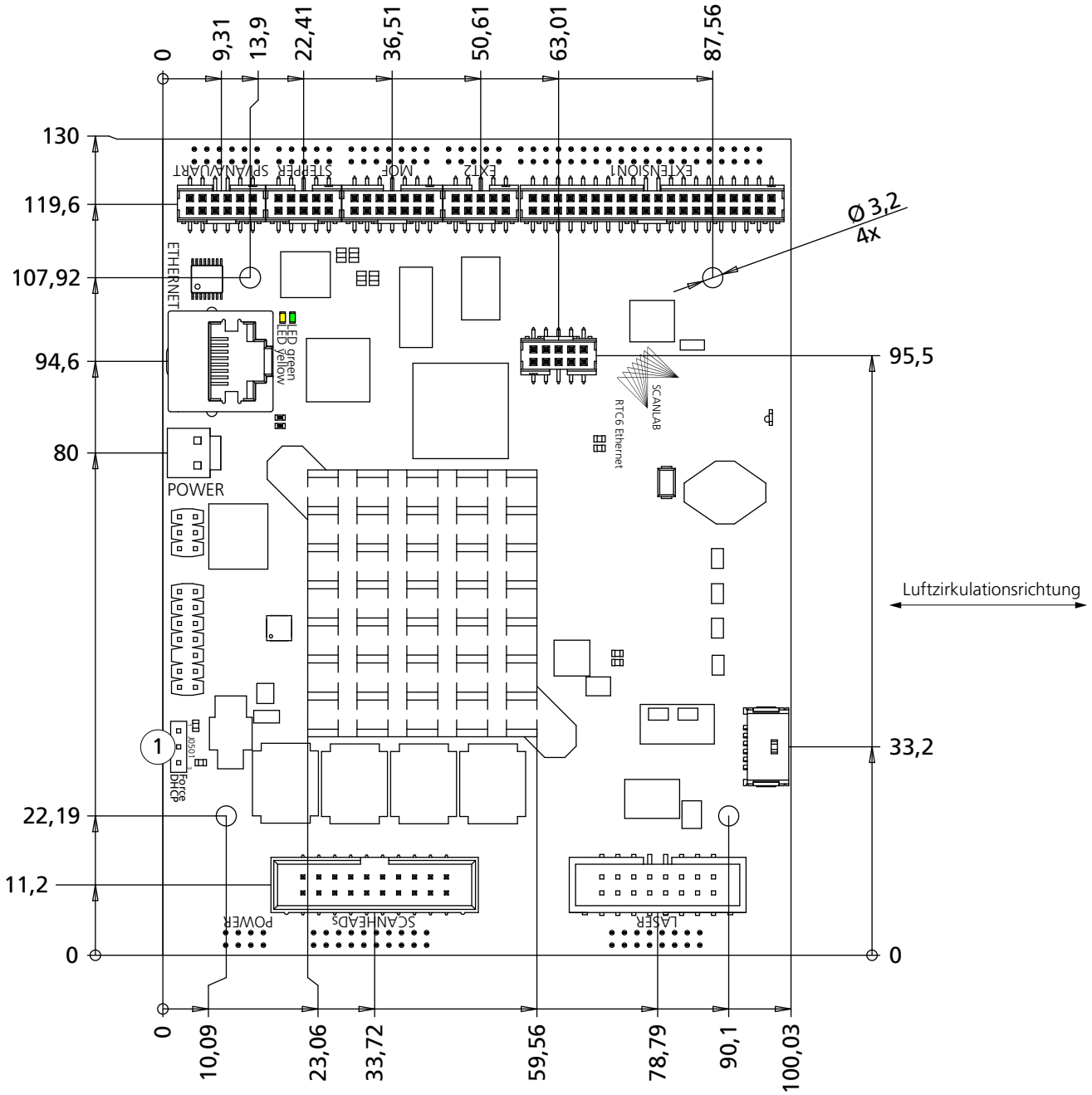
18 Anhang C: RTC6 EtherBox



RTC6 EtherBox. Abmessungen. Z.B. #0148520.



RTC6 EtherBox: Pinbelegungen.



Legende

1. Force DHCP. . . . . Steckjumperfeld zum Ignorieren/nicht Ignorieren der gespeicherten statischen IP-Adresse. Details wie "Steckjumperfeld 'Force DHCP'", Seite 1000.

RTC6 EtherBox: Verbaute RTC6 Ethernet-Karte. Oberseite. Abmessungen und Steckverbinderpositionen. Alle Maße in mm.

18.1 Montage

Achtung!

- Die Karte wird bei einer Überhitzung irreversibel zerstört. Stellen Sie sicher, dass die perforierten Seiten der RTC6 EtherBox niemals verschlossen oder abgedeckt sind.

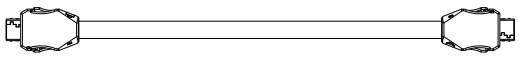
- Verwenden Sie den Hutschienenhalter an der Rückseite, siehe Abbildung 97, um die RTC6 EtherBox in eine Hutschiene einzuhängen. Alternativ stellen Sie das Gerät frei auf.

18.2 Verkabelung

Vorsicht!

- Auch wenn die beiden RTC6 EtherBoxen an dieselbe Spannungsversorgung angeschlossen sind, können durch eine ungünstige Verkabelung an den Karten unterschiedliche Potentiale vorherrschen und die Karten zerstören. Um unterschiedliche Potentiale zu vermeiden:
  - Verbinden Sie eine RTC6 EtherBox mit der Spannungsversorgung und verbinden Sie die zweite RTC6 EtherBox durch eine Abzweigung am Kabel mit derselben Spannungsversorgung.
  - Halten Sie dabei die Verbindungsstrecke von einer RTC6 EtherBox zur nächsten so kurz wie möglich.
  - Führen Sie erst die Verkabelung korrekt aus und schalten Sie danach die Spannungsversorgung ein.
  - Schalten Sie unbedingt erst die Spannungsversorgung aus, bevor Sie die Verkabelung trennen.

- Die Spannungsversorgung erfolgt über einen Phoenix-Stecker, siehe Abbildung 98. Verwenden Sie dazu den mitgelieferten Phoenix-Gegenstecker (Hersteller-Artikelnummer: 1827703).
- SCANLAB empfiehlt, das Gerät zu erden. Verbinden Sie dazu die Erdung mit dem Flachstecker (6,3×0,8×4,3 mm).
- Verbinden Sie den PC über ein Ethernet-Kabel mit der RTC6 EtherBox.
- Schließen Sie Ihre gewünschten Geräte an den jeweiligen Anschlüssen an. Beachten Sie dabei die Pinbelegungen der Anschlüsse, siehe Abbildung 98.
- Verwenden Sie für die elektrische Einbindung der RTC6 EtherBox nur die Anschlüsse an der Außenseite des Gehäuses. Verwenden Sie keine Pins auf der eingebauten RTC6 Ethernet-Karte (siehe Abbildung 99), die nicht von SCANLAB mit Anschlüssen am Gehäuse der RTC6 EtherBox verbunden sind.
- Verbinden Sie den MASTER-Anschluss/SLAVE-Anschluss der RTC6 EtherBox *nicht* mit dem separater RTC6-Karten, weil die Pinbelegungen unterschiedlich sind.
- Wenn Sie mehrere RTC6 Ethernet-Karten/RTC6 EtherBoxen mit Spannung aus nur einer Quelle versorgen, müssen Sie die Verkabelung so ausführen, wie in Abbildung 92 gezeigt.
- Wenn Sie mehrere RTC6 EtherBoxen zusammen(1) verwenden, achten Sie auf eine korrekte Master/Slave-Verbindung. Ein geeignetes Kabel ist bei SCANLAB erhältlich, siehe Abbildung 100.



#0149964 l = 200 mm

100

SCANLAB Master/Slave-Verbindungskabel für RTC6 EtherBoxen.

(1) Siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127.

18.3 Inbetriebnahme und Verwendung

- Bei der Inbetriebnahme der RTC6 Ethernet-Karte muss eine statische IP-Adresse bezogen werden. Dazu müssen Sie den Steckjumper **Force DHCP** an der RTC Ethernet-Karte umstellen:

  (1) Stellen Sie sicher, dass die RTC6 EtherBox von der Spannungsversorgung getrennt ist.

  (2) Schrauben Sie die zwei Torx-Schrauben an der Rückseite RTC6 EtherBox auf und nehmen Sie den Gehäusedeckel ab.

  (3) Setzen Sie den Jumper **'Force DHCP'**, siehe auch Abbildung 99, in "Force DHCP-Stellung":

  "Force DHCP-Stellung"

  (4) Setzen Sie den Gehäusedeckel wieder auf und schrauben die zwei Torx-Schrauben aus Schritt 2 wieder ein.

  (5) Verbinden Sie die RTC6 Ethernet-Karte wieder mit der Stromversorgung und dem Netzwerk, siehe auch Kapitel 16.3.3 "Mit einem Netzwerk verbinden", Seite 1002.

  (6) Alles weitere zur Installation und Verwendung der RTC6 Ethernet-Karte (in der RTC6 EtherBox) finden Sie im Hauptteil des RTC6-Handbuchs.

18.4 Rechtliches

18.4.1 EU-Konformitätserklärung – RTC6 EtherBox

EU-Konformitätserklärung

für das Produkt:

RTC6 EtherBox

Der Hersteller

SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Deutschland

erklärt, dass das genannte Produkt die einschlägigen Harmonisierungsrechtsvorschriften der Union erfüllt:

- **2011/65/EU -** Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Beschränkung der Verwendung bestimmter gefährlicher Stoffe in Elektro- und Elektronikgeräten (RoHS-Richtlinie).

Folgende harmonisierte Normen wurden angewandt:

- **DIN EN IEC 63000:2019-05** - Technische Dokumentation zur Beurteilung von Elektro- und Elektronikgeräten hinsichtlich der Beschränkung gefährlicher Stoffe

Die alleinige Verantwortung für die Ausstellung dieser Konformitätserklärung trägt der Hersteller.

Dieses Dokument zur Kundeninformation wurde elektronisch ausgestellt und ist daher ohne Unterschrift gültig. Eine unterschriebene Ausführung ist Bestandteil der SCANLAB-internen technischen Dokumentation.

EU Declaration of Conformity

for the product:

RTC6 EtherBox

The manufacturer

SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Germany

declares that the product mentioned above complies with the relevant Union harmonization legislation:

- **2011/65/EU** - Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the restriction of the use of certain hazardous substances in electrical and electronic equipment (RoHS Directive).

The following harmonized standards have been applied:

- **DIN EN IEC 63000:2019-05** - Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances

This declaration of conformity is issued under the sole responsibility of the manufacturer.

This document, used for customer information, was issued electronically and is therefore valid without a signature. A signed version is part of the SCANLAB internal technical documentation.

SCANLAB GmbH
Puchheim

2021-07-19

© SCANLAB GmbH – 2021/07

18.4.2 Bestätigung elektromagnetische Verträglichkeit (EMV)

Bestätigung elektromagnetische Verträglichkeit (EMV)

Das Produkt:

RTC6 EtherBox

wurde bezüglich der elektromagnetischen Verträglichkeit (EMV) wie folgt geprüft:

☒	Ein baugleiches System wurde getestet und erfüllt die Anforderungen der harmonisierten Normen **EN 55011:2016** **EN 61000-6-2:2005 +AC:2005** **EN 61000-3-2:2014** **EN 61000-3-3:2013**
☐	Ein technisch vergleichbares System wurde getestet und erfüllt die Anforderungen der harmonisierten Normen [Bezeichnung]
☐	Die EMV-relevanten Komponenten des Systems wurden einzeln oder in einer anderen Konfiguration getestet und erfüllten hierbei die Anforderungen der harmonisierten Normen [Bezeichnung]
☐	Ein baugleiches System wurde in einem EMV-kompatiblen Gehäuse getestet und erfüllte in diesem Zustand die Anforderungen der harmonisierten Normen [Bezeichnung]

Bemerkungen:

./.

SCANLAB GmbH
Puchheim

2023-07-06

Dieses Dokument wurde elektronisch ausgestellt und ist daher ohne Unterschrift gültig.

07/2023

18.4.3 Konformität mit FCC-Bestimmungen

Die RTC6 EtherBox wurde getestet und entspricht den Grenzwerten für Digitalgeräte der Klasse A, gemäß Teil 15 der FCC-Bestimmungen.

Diese Grenzwerte sollen einen angemessenen Schutz gegen schädliche Störungen bieten, wenn die RTC6 EtherBox in einer kommerziellen Umgebung betrieben wird. Die RTC6 EtherBox erzeugt, verwendet und emittiert möglicherweise Hochfrequenzenergie und kann, wenn nicht in Übereinstimmung mit dieser Bedienungsanleitung installiert und verwendet, Störungen im Funkverkehr verursachen. Der Betrieb der RTC6 EtherBox in einem Wohngebiet wird wahrscheinlich schädliche Störungen verursachen; in diesem Fall ist der Benutzer verpflichtet, die Störungen auf eigene Kosten zu beseitigen.

18.5 Technische Spezifikationen – RTC6 EtherBox

Abmessungen

Siehe Abbildung 98.

Gewicht

1,4 kg

Elektrische Spezifikationen

Versorgungsspannung (a)	+12 V…+30 V
Maximale Leistungsaufnahme, ohne angeschlossene Peripheriegeräte	< 12 W

(a) Kein Netzteil im Lieferumfang.

19 Anhang D: RTC6-Software-Pakete – Historie

Quelle: `versions.md`.

RTC6-Software-Paket Version	DLL (a)	OUT (a)	ETH (a)	RBF (a)	DAT (a)	BIOS (b)	BIOS-ETH (c)	Datum
1.19.1	648	651	651	643	604	24	44	2024-10-25
1.19.0	647	650	650	642	604	24	43	2024-08-02
1.18.0	646	649	649	641	604	23	42	2024-02-16
1.17.0	644	647	647	640	604	23	40	2023-11-24
1.16.3	643	646	646	639	604	23	40	2023-09-29
1.16.2	643	646	646	639	604	23	39	2023-09-15
1.16.1	643	645	645	639	604	23	39	2023-07-21
1.15.5	639	640	640	637	604	23	37	2023-02-15
1.15.4	638	639	639	637	604	23	36	2023-01-20
1.15.3	638	638	638	636	604	23	36	2022-12-22
1.15.2	637	637	637	636	604	23	36	2022-11-24
1.15.0	635	636	636	635	604	23	36	2022-09-16
1.14.1	634	635	635	634	604	23	35	2022-07-22
1.14.0	633	634	634	633	604	23	34	2022-06-03
1.13.0	632	633	633	633	604	23	33	2022-02-18
1.12.0	631	632	632	632	604	23	33	2021-12-22
1.11.0	630	631	631	631	603a	23	32	2021-11-12
1.10.0	629	630	630	631	603a	23	31	2021-10-15
1.9.0	628	629	629	630	603a	23	30	2021-09-03
1.7.12	627	628	628	629	603a	23	29	2021-07-09
1.7.11	626	627	627	629	603a	23	28	2021-04-30
1.7.10	625	626	626	628	603a	23	28	2021-02-26
1.7.9	625	625	625	628	603a	23	28	2021-02-05
1.7.8	624	624	624	627	603a	23	28	2021-01-22
1.7.7	623	623	623	626	603a	23	28	2020-12-11
1.7.6	622	622	622	625	603a	23	28	2020-10-02
1.7.5	621	621	621	625	603a	23	27	2020-07-24
1.7.4	620	620	620	625	603a	23	27	2020-07-10
1.7.3	619	619	619	624	603a	23	27	2020-06-19
1.7.1	618	618	618	623	603a	23	26	2020-03-19
1.7.0	618	618	618	623	603a	23	26	2020-03-13

RTC6-Software-Paket Version	DLL (a)	OUT (a)	ETH (a)	RBF (a)	DAT (a)	BIOS (b)	BIOS-ETH (c)	Datum
1.6.1	617	617	617	623	603a	23	25	2020-02-07
1.6.0	616	616	616	622	603a	23	25	2019-11-25
1.5.2	615	615	615	621	603a	23	25	2019-09-11
1.5.0	614	614	614	619	603a	23	25	2019-07-26
1.4.4	611	611	611	615	603	23	24	2019-03-01
1.4.2	610	610	610	615	602	23	23	2018-11-15
1.4.1	609	609	609	614	601	23	23	2018-08-03
1.3.3	608	608	608	612	601	22	22	2018-04-19
1.3.2	608	608	608	611	601	22	22	2018-01-23
1.3.1	607	607	607	611	601	22	*	2017-11-09
1.3.0	606	606	*(d)	611	601	21	*	2017-09-11

(a) Siehe Tabelle Seite 25.

(b) `RTC6BIOSOUT_*.out`, siehe Seite 35.

(c) `RTC6BIOSETH_*.out`, siehe Seite 35.

(d) Kann nicht durch Benutzer upgedatet werden oder Version kann nicht ausgelesen werden.

20 Anhang E: i*DRIVE*-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale

- Zur Benutzung siehe **control_command**.
- Beispiel:
  `control_command( Data` = 0501_H)
  d.h. $Code_{HIGH}$ = 05_H (**SetMode**) und $Code_{LOW}$ = 01_H fordert an, dass das i*DRIVE*-Scan-System den zu übertragenden Datentyp auf "Ist-Position" umstellen soll. Das i*DRIVE*-Scan-System wertet die Anforderung aus und überträgt – im Erfolgsfall – nachfolgend die "Ist-Position" an die RTC-Karte (die Notation in diesem Handbuch lautet in diesem Fall "Signal 0501_H").
- Die folgende Referenz wird in mehreren RTC-Handbüchern benutzt und ist deshalb generisch.

$Code_{HIGH}$ (hex)	
05	**SetMode** Mit **SetMode** kann ausgewählt werden, welchen Datentyp das Scan-System (für die ausgewählte Achse) übertragen soll. Siehe auch Kapitel 8.1.2 "Datensignal-Übertragungsverhalten des Scan-Systems anpassen", Seite 225. Jeder Parameterwert $Code_{LOW}$ entspricht einem bestimmten Datentyp. Standardeinstellung: $Code_{LOW}$ = 00_H (XY2-100-Statuswort). Siehe auch "Zu SetMode", Seite 1124. Siehe auch "Zu SetMode, SetControlDefinitionMode und SetEchoMode", Seite 1124. i*DRIVE*-Scan-Systeme mit XY2-100-Schnittstelle übertragen einen 16-Bit-Wert mit Vorzeichen. i*DRIVE*-Scan-Systeme mit SL2-100-Schnittstelle übertragen einen 20-Bit-Wert mit Vorzeichen. Z.B. excelli*SCAN*-Scan-Köpfe. i*DRIVE*-Scan-Systeme mit XY2-100 Enhanced-Schnittstelle übertragen einen 16-Bit-Wert mit Vorzeichen. Bei diesen wandelt der XY2-100-Konverter (Zubehör) den Wert (durch Multiplizieren mit 16) in einen 20-Bit-Wert mit Vorzeichen um. In diesem Dokument: 0500_H; 0501_H; 0502_H; 0503_H; 0504_H; 0505_H; 0506_H; 0507_H; 0508_H; 0509_H; 0510_H; 0511_H; 0512_H; 0513_H; 0514_H; 0515_H; 0516_H; 0517_H; 0518_H; 0519_H; $051A_H$; $051B_H$; $051C_H$; $051D_H$; $051E_H$; $051F_H$; 0520_H; 0521_H; 0522_H; 0523_H; 0524_H; 0525_H; 0526_H; 0527_H; 0528_H; 0529_H; $052A_H$; $052B_H$; $052C_H$; $052D_H$; $052E_H$; $052F_H$; 0530_H; 0531_H; 0532_H; 0533_H; 0534_H; 0535_H; 0536_H; 0537_H; 0538_H; 0539_H; $053A_H$; $053B_H$; $053C_H$; $053D_H$; $053E_H$; $053F_H$; 0549_H; $055E_H$; 0564_H; 0565_H; 0574_H;

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	00	Ab Scan-System-Firmware ≥ 2001. XY2-100-Statuswort		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #15 (MSB)	Bit #19 (MSB)	Wie [Bit #7|Bit #11]. = 1: Die Scan-System-Regelung ist bereit für Eingabedaten ("PowerOK", entspricht eigentlich einem "ServoOK"). Bei Scan-Systemen mit Sensorik sowie Scan-Systemen mit Interlock: zusätzlich liegt kein Interlock-Fehler vor.
		Bit #14	Bit #18	Wie [Bit #6|Bit #10]. = 1: Die System-Temperatur ist im optimalen Bereich ("TempOK"). Bei Scan-Systemen mit Sensorik: zusätzlich liegt keine Warnung vor.
		Bit #13	Bit #17	= 1.
		Bit #12	Bit #16	Wie [Bit #4|Bit #8]. = 1: Die Positionsfehler der y-Achse sind im erlaubten Bereich ("PosAck"-Signal).
		Bit #11	Bit #15	Wie [Bit #3|Bit #7]. = 1: Die Positionsfehler der x-Achse sind im erlaubten Bereich ("PosAck"-Signal).
		Bit #10	Bit #14	Wie [Bit #2|Bit #6]. = 1. Nur intelli*WELD*: Reserviert. Bei Scan-Systemen mit Sensoren zur automatischen Selbstkalibrierung: Reserviert.
		Bit #9	Bit #13	= 0.
		Bit #8	Bit #12	= 1.
		Bit #7	Bit #11	Wie [Bit #15|Bit #19].
		Bit #6	Bit #10	Wie [Bit #14|Bit #18].
		Bit #5	Bit #9	= 1.
		Bit #4	Bit #8	Wie [Bit #12|Bit #16].
		Bit #3	Bit #7	Wie [Bit #11|Bit #15].
		Bit #2	Bit #6	Wie [Bit #10|Bit #14].
		Bit #1	Bit #5	= 0.
		Bit #0	Bit #4	= 1.
		–	Bit #3	= 0.
		–	Bit #2	= 0.
		–	Bit #1	= 0.
		–	Bit #0	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	01	Ab Scan-System-Firmware ≥ 2001. Ist-Position (Winkelstellung der Galvanometerscanner)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In Bits. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In Bits. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	02	Ab Scan-System-Firmware ≥ 2001. Soll-Position (Winkelstellung der Galvanometerscanner) Ab Scan-System-Firmware ≥ 5050. Bei excelli*SCAN*-Scan-Köpfen: Vorausberechnete Soll-Position (Winkelstellung der Galvanometerscanner). Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.1.6 "Timing bei SCANahead-Systemen", Seite 20 und Abbildung 6.		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In Bits. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In Bits. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	03	Ab Scan-System-Firmware ≥ 2001. Positionsfehler (= Soll-Position 0502_H – Ist-Position 0501_H) Ab Scan-System-Firmware ≥ 5050. Bei excelli*SCAN*-Scan-Köpfen: Trajektorienfehler. Siehe Glossareintrag.		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In Bits. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In Bits. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	04	Ab Scan-System-Firmware ≥ 2001. Ist-Strom (Endstufenstrom)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In mA. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 16^{-1} mA. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	05	Ab Scan-System-Firmware ≥ 2001. Relative Galvanometerscanner-Ansteuerung Nicht intelli*cube*.		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In Promille. Wertebereich mit 16-Bit-Protokoll: [–1.000…+1.000]. Einheit mit 20-Bit-Protokoll: In 16^{-1} Promille. Wertebereich mit 20-Bit-Protokoll: [–16.000…+16.000]. Der Rückgabewert 16 entspricht 1‰.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	06	Ab Scan-System-Firmware ≥ 2001. Ist-Geschwindigkeit (Winkelgeschwindigkeit)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In bit/ms. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In bit/ms. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	07	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	08	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	09	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	10	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	11	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	12	Ab Scan-System-Firmware ≥ 2001. Nur intelli*WELD* mit vario*SCAN* $_{de}$ FCi. Nicht: intelli*WELD* mit vario*SCAN* FC. Ist-Position der z-Achse Kann nur über den Kanal des Scan-Kopf-Anschlusses ausgelesen werden, an dem die z-Achse angeschlossen ist.		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In Bits. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In Bits. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	13	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	14	Ab Scan-System-Firmware ≥ 2001. Temperatur des Galvanometerscanners		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-1} °C. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 160^{-1} °C. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	15	Ab Scan-System-Firmware ≥ 2001. Temperatur der Servokarte Ab Scan-System-Firmware ≥ 5050. Achse 1 (x-Achse): Temperatur der Servokarte Achse 2 (y-Achse): Temperatur der Servo-Aufsteckkarte für das Auswerten der Encodersignale		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-1} °C. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 160^{-1} °C. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	16	Ab Scan-System-Firmware ≥ 2001. AGC-Spannung (Versorgungsspannung Positionsdetektor)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-2} V.. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 1600^{-1} V. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	17	Ab Scan-System-Firmware ≥ 2001. DSP Core-Versorgungsspannung		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-2} V. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 1600^{-1} V. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	18	Ab Scan-System-Firmware ≥ 2001. DSP IO-Spannung		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-2} V. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 1600^{-1} V. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	19	Ab Scan-System-Firmware ≥ 2001. Spannung Analogteil		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-2} V. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 1600^{-1} V. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	1A	Ab Scan-System-Firmware ≥ 2001. Versorgungsspannung Analog-Digital-Wandler		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-2} V. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 1600^{-1} V. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	1B	Ab Scan-System-Firmware ≥ 2001. AGC-Strom (PD-Versorgungsstrom)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	Einheit mit 16-Bit-Protokoll: In mA. Wertebereich mit 16-Bit-Protokoll: [–32.768…+32.767]. Einheit mit 20-Bit-Protokoll: In 16^{-1} mA. Wertebereich mit 20-Bit-Protokoll: [–524.288…+524.287].

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	1C	Ab Scan-System-Firmware ≥ 2066. Hardware-Version der Servokarte		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #0 … Bit #19	16-Bit Protokoll: 2nnn = DSCB 5021 = ISB1 5031 = ISB2 6011 = microISB 20-Bit Protokoll: 2nnn×16 = DSCB 5021×16 = ISB1 5031×16 = ISB2 6011×16 = microISB

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	1D	Ab Scan-System-Firmware ≥ 2001. Relevant für Scan-Systeme mit Galvanometerscanner-Heizung. Relative Heizleistung der Heizung des jeweiligen Galvanometerscanners Nicht relevant für intelli*cube*. Nicht relevant für dyn*AXIS* XS. Nicht relevant für Scan-Systeme mit wassergekühlten Galvanometerscannern.		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In Promille. Wertebereich mit 16-Bit-Protokoll: [0…1.000]. Einheit mit 20-Bit-Protokoll: In Promille. Wertebereich mit 20-Bit-Protokoll: [0…1.000].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	1E	Ab Scan-System-Firmware ≥ 2001. Seriennummer, untere 16 Bits		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: Keine. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: Keine. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	1F	Ab Scan-System-Firmware ≥ 2001. Seriennummer, obere 16 Bits		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: Keine. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: Keine. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	20	Ab Scan-System-Firmware ≥ 2001. SCANLAB-Artikelnummer, untere 16 Bits		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: Keine. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: Keine. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	21	Ab Scan-System-Firmware ≥ 2001. SCANLAB-Artikelnummer, obere 16 Bits		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: Keine. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: Keine. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	22	Ab Scan-System-Firmware ≥ 2001. Versionsnummer der Scan-System-Firmware		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: Keine. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: Keine. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	23	Ab Scan-System-Firmware ≥ 2001. Kalibrierwinkel Mechanischer Scan-Winkel (±) bei 96% des maximalen oder minimalen Ansteuerwerts. Bei 16-Bit Protokoll±31457 Bit. Bei 20-Bit Protokoll ±503316 Bit.		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-3} Grad. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: In 10^{-3} Grad. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	24	Ab Scan-System-Firmware ≥ 2001. Apertur		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In mm. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: In mm. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	25	Ab Scan-System-Firmware ≥ 2001. Wellenlänge		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In nm. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: In nm. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	26	Ab Scan-System-Firmware ≥ 2078. Tuningnummer		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #8 … Bit #15	Bit #12 … Bit #19	Starteinstellung.
		Bit #0 … Bit #7	Bit #4 … Bit #11	Aktuelle Einstellung.
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

<table>
<tr><td>$Code_{HIGH}$ (hex)</td><td>$Code_{LOW}$ (hex)</td><td colspan="3">Vom Scan-System übertragener Datentyp</td></tr>
<tr><td rowspan="5">05</td><td rowspan="5">27</td><td colspan="3">Ab Scan-System-Firmware ≥ 2078.
Nur nach <code>control_command</code>(<code>Data</code> = $17FF_H$): Nummer des zwischengespeicherten Datentyps</td></tr>
<tr><td>16-Bit Protokoll</td><td>20-Bit Protokoll</td><td></td></tr>
<tr><td>Bit #8
…
Bit #15</td><td>Bit #12
…
Bit #19</td><td>= 0.</td></tr>
<tr><td>Bit #0
…
Bit #7</td><td>Bit #4
…
Bit #11</td><td>Zwischengespeicherte Einstellung.</td></tr>
<tr><td>–</td><td>Bit #0
…
Bit #3</td><td>= 0.</td></tr>
</table>

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!

Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	28	Ab Scan-System-Firmware ≥ 2060. Diagnose-Flags, untere 16 Bits		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	29	Ab Scan-System-Firmware ≥ 2060. Diagnose-Flags, obere 16 Bits		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	2A	Ab Scan-System-Firmware ≥ 2001. Stopp-Ereigniscode Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Wertebereich mit 20-Bit-Protokoll: [0…65.535].		
		16-Bit Protokoll	20-Bit Protokoll	
		(hex)	(hex)	
		0001	00010	Der Galvanometerscanner hat eine kritische Position erreicht.
		0002	00020	Fehler beim AD-Wandler.
		0003	00030	Temperatur im Scan-System ist über dem zulässigen Höchstwert.
		0004	00040	Spannungseinbruch der externen Versorgungsspannung.
		0005	00050	Die Flags erfüllen nicht die Betriebsvoraussetzung.
		0006 … 000C	00060 … 000C0	Reserviert. … Reserviert.
		000D	000D0	Reserviert.
		000E	000E0	Position Acknowledge Time-out (Sollposition lange nicht erreicht).
		000F	000F0	Reserviert.
		0014	00140	Reserviert.
		0015	00150	Reserviert.
		0016	00160	Reserviert.
		0017	00170	Reserviert.
		0018	00180	Reserviert.
		0019	00190	Reserviert.
		001A	001A0	Reserviert.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!

Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	2B	Ab Scan-System-Firmware ≥ 2060. Flags von letztem Stopp-Event, untere 16 Bit		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	2C	Ab Scan-System-Firmware ≥ 2060. Flags von letztem Stopp-Event, obere 16 Bit		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	2D	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	2E	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	2F	Ab Scan-System-Firmware ≥ 2061. Betriebszeit (Sekunden)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In s. Wertebereich mit 16-Bit-Protokoll: [0…59]. Einheit mit 20-Bit-Protokoll: In s. Wertebereich mit 20-Bit-Protokoll: [0…59].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	30	Ab Scan-System-Firmware ≥ 2061. Betriebszeit (Minuten)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In min. Wertebereich mit 16-Bit-Protokoll: [0…59]. Einheit mit 20-Bit-Protokoll: In min. Wertebereich mit 20-Bit-Protokoll: [0…59].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	31	Ab Scan-System-Firmware ≥ 2061. Betriebszeit (Stunden)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In h. Wertebereich mit 16-Bit-Protokoll: [0…23]. Einheit mit 20-Bit-Protokoll: In h. Wertebereich mit 20-Bit-Protokoll: [0…23].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	32	Ab Scan-System-Firmware ≥ 2061. Betriebszeit (Tage)		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In d. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: In d. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	33	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: 3,3 V Sensorplatinen-Betriebsspannung		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 10^{-3} V. Wertebereich mit 16-Bit-Protokoll: [0…7.000]. Einheit mit 20-Bit-Protokoll: In 10^{-3} V. Wertebereich mit 20-Bit-Protokoll: [0…7.000].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	34	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: Sensorplatinen-Betriebstemperatur		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0.1 °C. Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Einheit mit 20-Bit-Protokoll: In 0.1 °C. Wertebereich mit 20-Bit-Protokoll: [0…65.535].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	35	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: Spülgasdurchflussrate		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: [(Spülgasdurchflussrate – 4 l/min) * 109,89 min/l]. Wertebereich mit 16-Bit-Protokoll: [0…2.250]. Einheit mit 20-Bit-Protokoll: [Spülgasdurchflussrate – 4 l/min) * 109,89 min/l]. Wertebereich mit 20-Bit-Protokoll: [0…2.250].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	36	Ab Scan-System-Firmware ≥ 206x. [Temperatur von Spiegel 2 + 26,6 °C]		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0,05 °C. Wertebereich mit 16-Bit-Protokoll: [0…1.992]. Einheit mit 20-Bit-Protokoll: In 0,05 °C. Wertebereich mit 20-Bit-Protokoll: [0…1.992].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	37	Ab Scan-System-Firmware ≥ 206x. [Temperatur von Spiegel 1 + 26,6 °C]		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0,05 °C. Wertebereich mit 16-Bit-Protokoll: [0…1.992]. Einheit mit 20-Bit-Protokoll: In 0,05 °C. Wertebereich mit 20-Bit-Protokoll: [0…1.992].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	38	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: Schutzglas-Temperatur		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0,044 °C. Wertebereich mit 16-Bit-Protokoll: [0…2.250]. Einheit mit 20-Bit-Protokoll: In 0,044 °C. Wertebereich mit 20-Bit-Protokoll: [0…2.250].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	39	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: Kollimatortemperatur		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0,05 °C. Wertebereich mit 16-Bit-Protokoll: [0…2.250]. Einheit mit 20-Bit-Protokoll: In 0,05 °C. Wertebereich mit 20-Bit-Protokoll: [0…2.250].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	3A	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: Temperatur des Galvanometerscanner-Halters		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0,05 °C. Wertebereich mit 16-Bit-Protokoll: [0…2.250]. Einheit mit 20-Bit-Protokoll: In 0,05 °C. Wertebereich mit 20-Bit-Protokoll: [0…2.250].
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	3B	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: [Kühlflussrate + 0,93 l × min^{-1}]		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0,00525 l × min-1. Wertebereich mit 16-Bit-Protokoll: [0…2.250]. Einheit mit 20-Bit-Protokoll: [0…2.250]. Wertebereich mit 20-Bit-Protokoll: In 0,00525 l × min-1.
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	3C	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: Schutzglas-Streulichtwert		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #0 … Bit #15	Bit #4 … Bit #19	Einheit mit 16-Bit-Protokoll: In 0,00444 V. Wertebereich mit 16-Bit-Protokoll: [0…2.250]. Einheit mit 20-Bit-Protokoll: [0…2.250]. Wertebereich mit 20-Bit-Protokoll: In 0,00444 V.
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	3D	Ab Scan-System-Firmware ≥ 206x. Nur intelli*WELD*: Sensor-Platinen-Flags Wertebereich mit 16-Bit-Protokoll: [0…65.535]. Wertebereich mit 20-Bit-Protokoll: [0…65.535].		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #15 (MSB)	Bit #19 (MSB)	= 1: Temperaturwert des Schutzglasfensters außerhalb [0…1.882].
		Bit #14	Bit #18	= 1: Temperaturwert von Spiegel 1 außerhalb [0…2.250].
		Bit #13	Bit #17	= 1: Temperaturwert von Spiegel 2 außerhalb [0…2.250].
		Bit #10	Bit #14	= 1: Temperaturwert am Strahlaustritt außerhalb [0…1.200]. Außerdem wird ein INTERLOCK-Fehler ausgelöst.
		Bit #3	Bit #7	= 1: Schutzglasfenster-Streulichtwert außerhalb [0…2.250].
		Bit #2	Bit #6	= 1: Kühlflussrate außerhalb [750…2.250].
		Bit #1	Bit #5	= 1: Temperaturwert des Galvanometerscanner-Halters außerhalb [0…1.600]. Außerdem wird ein INTERLOCK-Fehler ausgelöst.
		Bit #0	Bit #4	= 1: Kollimatortemperaturwert außerhalb [0…2.000]. Außerdem wird ein INTERLOCK-Fehler ausgelöst.
		–	Bit #0 … Bit #3	= 0. … = 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!

Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	3E	Reserviert.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	3F	Ab Scan-System-Firmware ≥ 2072. Eingestellter Skalierungsfaktor für Positionswerte		
		16-Bit Protokoll	20-Bit Protokoll	
		Bit #2 … Bit #15	Bit #6 … Bit #19	Reserviert.
		Bit #0 … Bit #1	Bit #4 … Bit #5	Skalierungsfaktor = $1/2^n$. 16-Bit Protokoll n = 2 × Bit #1 + Bit #0. 20-Bit Protokoll n = 2 × Bit #5 + Bit #4.
		–	Bit #0 … Bit #3	= 0.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	49	Ab Scan-System-Firmware ≥ 5000. Versorgungsspannung (üblicherweise 30 V oder 48 V)		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	5E	Ab Scan-System-Firmware ≥ 5000. Anzahl der "Regulärer Betrieb → Fehlerzustand"-Übergänge		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp
05	64	Ab Scan-System-Firmware ≥ 5100. Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.3 "Erweiterte Einstellungen für "Spot Distance Control"", Seite 24: `TimeShift` relativ zu der von SCANLAB voreingestellten Verschiebung.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp
05	65	Ab Scan-System-Firmware ≥ 5100. Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.3 "Erweiterte Einstellungen für "Spot Distance Control"", Seite 24: `TimeShift` gesamt.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	$Code_{LOW}$ (hex)	Vom Scan-System übertragener Datentyp		
05	74	Ab Scan-System-Firmware ≥ 51*7 + < 5090. Für Rückkanal-Multiplexing. Zur Nutzung in Verbindung mit $Code_{HIGH}$ (hex) = 65_H und $Code_{HIGH}$ (hex) = 66_H. Siehe entsprechendes Kapitel inkl. Beispiel-Code.		
		16-Bit Protokoll	20-Bit Protokoll	–
		Bit #0 … Bit #15	Bit #0 … Bit #19	–

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

<table>
<tr><td>Code_{HIGH} (hex)</td><td colspan="2"></td></tr>
<tr><td>0A</td><td colspan="2">Ab Scan-System-Firmware ≥ 2078.
UpdatePermanentMemory
UpdatePermanentMemory veranlasst, dass das Scan-System das aktuelle Regelverhalten als Startverhalten für nachfolgende Neustarts oder Resets einstellt.
Siehe auch Kapitel 8.1.8 “Startverhalten anpassen”, Seite 235.
Ab Scan-System-Firmware ≥ 5050.
Nur excelliSCAN:
excelliSCAN-Scan-Köpfe unterstützen kein UpdatePermanentMemory.</td></tr>
<tr><td></td><td>Code_{LOW} (hex)</td><td></td></tr>
<tr><td></td><td>00</td><td>Einziger zulässiger Wert.</td></tr>
</table>

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

<table>
<tr><td>Code_{HIGH} (hex)</td><td colspan="4"></td></tr>
<tr><td>0E</td><td colspan="4">Ab Scan-System-Firmware ≥ 206x.
SetControlDefinitionMode
SetControlDefinitionMode veranlasst, dass das Scan-System Informationen über ein bestimmtes Tuning (oder den entsprechenden Regelalgorithmus) über den ausgewählten Statuskanal überträgt.
Siehe auch “Zu SetMode, SetControlDefinitionMode und SetEchoMode”, Seite 1124.</td></tr>
<tr><td></td><td>Code_{LOW} (hex)</td><td colspan="3">Vom Scan-System übertragener Datentyp</td></tr>
<tr><td></td><td>00
01
02
03</td><td colspan="3">Ab Scan-System-Firmware ≥ 206x.
Tuningnummer</td></tr>
<tr><td></td><td></td><td>16-Bit Protokoll</td><td>20-Bit Protokoll</td><td></td></tr>
<tr><td></td><td></td><td>Bit #15
(MSB)</td><td>Bit #19
(MSB)</td><td>= 0: Tuning vorhanden.
= 1: Tuning nicht vorhanden.</td></tr>
<tr><td></td><td></td><td>Bit #4
…
Bit #14</td><td>Bit #8
…
Bit #18</td><td>Reserviert.</td></tr>
<tr><td></td><td></td><td>Bit #0
…
Bit #3</td><td>Bit #4
…
Bit #7</td><td>Tuningtyp
= 0: Vektortuning.
= 1: Sprungtuning.
= 2: Reserviert.
= 3: Reserviert.
= 4: Ab Scan-System-Firmware ≥ 5050.
SCANahead-Regelung.</td></tr>
<tr><td></td><td></td><td>–</td><td>Bit #0
…
Bit #3</td><td>= 0.</td></tr>
</table>

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!

Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)		
11	Ab Scan-System-Firmware ≥ 2078. **SelectControlDefinition** Nur Scan-Systeme, die mit mehreren Tunings (oder entsprechenden Regelalgorithmen) ausgestattet sind. **SelectControlDefinition** veranlasst, dass das Scan-System auf ein bestimmtes Tuning umstellt. Siehe auch Kapitel 8.1.4 "Tuning (Dynamikeinstellung) auswählen", Seite 228. $Code_{LOW}$ = 00…03 gibt die Tuningnummer an. Ab Scan-System-Firmware ≥ 5050. Nur excelli*SCAN*: excelli*SCAN*-Scan-Köpfe unterstützen kein **SelectControlDefinition**. Bei excelli*SCAN*-Scan-Köpfen gibt es nur ein Tuning.	
	$Code_{LOW}$ (hex)	
	00	Tuning 0 (= Standardeinstellung)
	01	Tuning 1
	02	Tuning 2
	03	Tuning 3

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	
12	Ab Scan-System-Firmware ≥ 2072. **SetPositionScale** **SetPositionScale** veranlasst, dass das Scan-System auf einen bestimmten Skalierungsfaktor umstellt. Die Scan-System-Regelelektronik multipliziert die von der RTC-Karte empfangenen Soll-Positions-Werte mit diesem Skalierungsfaktor. Die vom Scan-System an die RTC-Karte zurückgesendeten Ist-Positions-Werte werden automatisch durch den selben Skalierungsfaktor dividiert. So sind Soll-Position und Ist-Position immer passend zueinander skaliert. Wichtig: Einen Skalierungsfaktor ≠ 1 können Sie durch einen Vergleich von Soll-Position und Ist-Position nicht erkennen. Siehe auch Kapitel 8.1.7 "Effektive Kalibrierung anpassen", Seite 234. Wichtig! Sie müssen den Skalierungsfaktor in 2 Schritten ändern: 1. Führen Sie **SetPositionScale** mit $Code_{LOW}$ = 83_H aus. 2. Führen Sie **SetPositionScale** mit $Code_{LOW}$ = [00…03] (= dem gewünschten Wert) aus. Siehe auch "Zu SetPositionScale", Seite 1125. Ab Scan-System-Firmware ≥ 5050. Nur *SCAN*ahead-Systeme: *SCAN*ahead-Systeme unterstützen kein **SetPositionScale**. Bei *SCAN*ahead-Systemen kann die effektive Kalibrierung nicht geändert werden.

$Code_{LOW}$ (hex)	
00	Skalierungsfaktor-Wert 1/1 (= keine Skalierung; Standardeinstellung)
01	Skalierungsfaktor-Wert 1/2
02	Skalierungsfaktor-Wert 1/4
03	Skalierungsfaktor-Wert 1/8
83	Bereitschaft für Skalierungsfaktor-Änderung herstellen.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)		
15	Ab Scan-System-Firmware ≥ 2066. **SetPosAcknowledgeLevel** **SetPosAcknowledgeLevel** veranlasst, dass das Scan-System auf einen bestimmten PosAck-Grenzwert umstellt. Siehe auch Kapitel 8.1.6 "PosAck-Grenzwert anpassen", Seite 234. Siehe auch "Zu SetPosAcknowledgeLevel", Seite 1125.	
	$Code_{LOW}$ (hex)	
	00 … FF	Gewünschter PosAck-Grenzwert in LSB16, 16-Bit Protokoll. In Bits.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)		
17	Ab Scan-System-Firmware ≥ 2078. **Store/RestoreTransmissionMode** **Store/RestoreTransmissionMode** veranlasst mit $Code_{LOW}$ = FF_H, dass das Scan-System den aktuell zur Übertragung ausgewählten Datentyp zwischenspeichert: $Code_{LOW}$ = FF_H. Zu einem späteren Zeitpunkt kann er dann mit $Code_{LOW}$ = 00_H wieder ausgewählt werden. Siehe auch `Data` = 0527_H. Siehe auch "Zu Store/RestoreTransmissionMode", Seite 1125.	
	$Code_{LOW}$ (hex)	
	FF	Zwischenspeichern
	00	Wieder auswählen

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	
21	Ab Scan-System-Firmware ≥ 2078. **SetEchoMode** **SetEchoMode** dient zum Überprüfen der Datenübertragung. Siehe auch Kapitel 8.1.9 "Fehlerdiagnose und Funktionstest", Seite 235. **SetEchoMode** übergibt mit $Code_{LOW}$ den gewünschten 8-Bit-Wert an das Scan-System. In der Folge überträgt das Scan-System am entsprechenden Statuskanal einen 16-Bit-Wert (16-Bit Protokoll) oder 20-Bit-Wert (20-Bit Protokoll) an die RTC-Karte. War die Datenübertragung fehlerfrei, sind bei diesem: • die oberen 8 Bit und $Code_{LOW}$ identisch • beim 16-Bit-Wert die unteren / beim 20-Bit-Wert die nächst niedrigeren 8 Bit und der Komplementärwert von $Code_{LOW}$ (NOT $Code_{LOW}$) identisch Beispiel mit 16-Bit Protokoll: Für `control_command(1, 1, 0x210A)` liefert `(get_value(1) AND 0xFFFF)` bei fehlerfreier Datenübertragung den Wert 0x0AF5 zurück. Beispiel mit 20-Bit Protokoll: Für `control_command( 1, 1, 0x210A )` liefert `(get_value(1) AND 0xFFFF0)` bei fehlerfreier Datenübertragung den Wert 0x0AF50 zurück. Siehe auch "Zu SetMode, SetControlDefinitionMode und SetEchoMode", Seite 1124.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)		
40	Ab Scan-System-Firmware ≥ 5102 + ≥ 5112. Nur excelli*SCAN*: **ResetTrAck** **ResetTrAck** veranlasst, dass das Scan-System die TrAck-Signal-Bits zurücksetzt. Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.1.3 "TrAck-Signal", Seite 16. Zu weiteren Informationen, z.B. über das Konfigurieren des TrAck-Grenzwerts, siehe das entsprechende Scan-System-Handbuch. Hinweis: Der TrAck-Grenzwert kann nicht für nachfolgende Neustarts oder Resets gespeichert werden.excelli*SCAN*-Scan-Köpfe unterstützen kein **UpdatePermanentMemory**.	
	$Code_{LOW}$ (hex)	
	3A	Nur excelli*SCAN*: Setze TrAck-Signal-Bits zurück.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	
50	Ab Scan-System-Firmware ≥ 5100. Nur excelli*SCAN*: Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.3 "Erweiterte Einstellungen für "Spot Distance Control"", Seite 24: dT_{high}.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)	
51	Ab Scan-System-Firmware ≥ 5100. Nur excelli*SCAN*: Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.3 "Erweiterte Einstellungen für "Spot Distance Control"", Seite 24: dT_{low}.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)		
65	Ab Scan-System-Firmware ≥ 51*7 + < 5090. Für Rückkanal-Multiplexing. Zur Nutzung in Verbindung mit $Code_{HIGH}$ (hex) = 66_H und 0574_H. Siehe entsprechendes Kapitel inkl. Beispiel-Code.	
	$Code_{LOW}$ (hex)	
	0N	Setzt die Rückkanal-Multiplexing-Block-Länge auf N+1 (d.h. größter Index = N). In anderen Worten: die Anzahl der Rückkanal-Multiplexing-Werte, die das i*DRIVE*-Scan-System übertragen soll. Zulässige Werte: 1; 2; 3; 4; 6; 8; 12; 16 (= ganzzahliger Teiler von 192 ≥ 16).
	8N	Setzt den Rückkanal-Multiplexing-Block-Index auf N, der als nächstes zu Konfigurieren ist. In anderen Worten: Definiert die Stelle, an der das Signal übertragen wird.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

$Code_{HIGH}$ (hex)		
66	Ab Scan-System-Firmware ≥ 51*7 + < 5090. Für Rückkanal-Multiplexing. Zur Nutzung in Verbindung mit $Code_{HIGH}$ (hex) = 65_H und 0574_H. Siehe entsprechendes Kapitel inkl. Beispiel-Code.	
	$Code_{LOW}$ (hex)	
	NN	Setzt den Rückgabewert für den aktuellen Index auf NN = Schreibt den Signal-Wert an diesen Index. Für NN können Sie die gleichen $Code_{LOW}$-Werte einsetzen wie bei **SetMode**, siehe $Code_{LOW}$ = 00_H (XY2-100-Statuswort), $Code_{LOW}$ = 01_H (Ist-Position (Winkelstellung der Galvanometerscanner)) usw.

Achtung! Generische Information. Möglicherweise für Ihr Scan-System unvollständig oder sogar unzutreffend. Konsultieren Sie immer das entsprechende Scan-System-Handbuch!
Modul Rev.1.0.6 de-DE

Änderungsindex		
Modul Rev.1.0.5 de-DE	0536_H	Nicht nur bei intelli*WELD*.
Modul Rev.1.0.5 de-DE	0537_H	Nicht nur bei intelli*WELD*.
Modul Rev.1.0.6 de-DE	0500_H	[Bit #15 | Bit #19]. Redaktionelle Erweiterung.
Modul Rev.1.0.6 de-DE	0500_H	[Bit #14 | Bit #18]. Redaktionelle Erweiterung.

Hinweise (RTC6)

- Scan-System-Firmware a.bb.c wird als abbc ausgelesen, z.B. entspricht 5100 der Version 5.10.0, und 5054 der Version 5.05.4.

- Zu **SetMode**, **SetControlDefinitionMode** und **SetEchoMode**
  - Der mit `control_command`($Code_{HIGH}$ = 05_H, $0E_H$ oder 21_H oder `Data` = 1700_H) ausgewählte Datentyp wird so lange übertragen, bis ein anderer ausgewählt wird.
  - Die vom Scan-System an die RTC-Karte übertragenen Daten können gelesen werden mit:
  - **get_value**
  - **get_values**
  - **set_trigger[*]**
  - **get_waveform**
    Die ersten Daten nach einem Umschalten der Datenquelle werden aufgrund der seriellen Übertragungszeiten erst nach einer kurzen zeitlichen Verzögerung übertragen. Daher kann zwischen dem Umschaltzeitpunkt und dem erstmaligen Ausgeben des neuen Datentyps eine Verzögerungszeit von bis zu 60 µs entstehen. **control_command** fügt nach einem Umschalten der Datenquelle automatisch immer eine Wartezeit von 60 µs ein (die oben genannten Befehle lesen in jedem Fall korrekte Rückgabewerte aus).
  - Alle vom Scan-System an die RTC6 übertragenen Daten werden als 20-Bit-Werte mit Vorzeichen übergeben. Dies gilt auch dann, wenn die RTC6-DLL auf den RTC4-Kompatibilitätsmodus eingestellt ist und auch für Scan-Systeme ohne SL2-100-Schnittstelle, die über einen XY2-100-Konverter (Zubehör) angesteuert werden. Beim Auslesen der zurückgegebenen Daten mit **get_value**, **get_values** oder **get_waveform** werden jedoch generell 32-Bit-Werte mit Vorzeichen an den PC übertragen (zur Auswertung siehe Hinweise bei **get_value**).
  - Bei Scan-Systemen mit SL2-100-Schnittstelle liest **get_head_status** unabhängig von einer mit `control_command`($Code_{HIGH}$ = 05_H, $0E_H$ oder 21_H oder `Data` = 1700_H) getroffenen Einstellung das XY2-100-Statuswort aus, das hier zusätzlich zu den ausgewählten Daten übertragen wird. Bei iDRIVE-Scan-Systemen (ohne integrierte SL2-100-Schnittstelle), die über einen XY2-100-Konverter (Zubehör) mit der RTC6 angesteuert werden, liefert **get_head_status** dagegen das XY2-100-Statuswort nur dann zurück, wenn dieses Signal auch zuvor mit **control_command** zur Übertragung ausgewählt wurde, siehe auch Abschnitt "Daten auslesen", Seite 225.
  - Nach einem Neustart oder Reset des Scan-Systems kann es ca. 5 s dauern, bis Daten vom Scan-System übertragen werden, siehe auch **get_value**. Nach einem Neustart oder Reset wird immer das XY2-100-Statuswort übertragen.

- Zu **SetMode**
  - Die vom Scan-System übertragenen Sollpositionswerte (Winkelstellung) entsprechen den tatsächlichen Ausgabewerten `Sample<AX…BY>_Out` bei **set_trigger[*]**. Die zurückgegebenen 20-Bit-Sollpositionswerte der z-Achse im RTC5-Kompatibilitätsmodus (und im RTC4-Kompatibilitätsmodus auch die der x-Achse und y-Achse) sind zusätzlich um einen Faktor 16 gegenüber den dafür angegebenen 16-Bit-Koordinatenwerten skaliert.
  - Die an die RTC6 zurückgegebenen Winkelgrößen in Bit (Ist-Position, Soll-Position, Positionsfehler) oder Bit/ms (Ist-Geschwindigkeit) können mit Hilfe des Kalibrierwinkels des Scan-Systems in die Maßeinheiten ° oder °/ms umgerechnet werden. Der Kalibrierwinkel kann mit `Data` = 0523_H ausgelesen werden.
  - Die genauen Werte für die in der Tabelle angegebenen internen Spannungen können abhängig vom Typ und der Version des jeweiligen Scan-Systems variieren.
  - intelli*WELD* Scan-Systeme können mit unterschiedlich vielen Sensoren ausgeliefert werden. Beachten Sie deshalb das Handbuch des jeweiligen Scan-Systems. In diesem Handbuch sind die Befehlscodes für derzeit verfügbare Sensoren aufgelistet.

- Zu **SetPositionScale**
  - Wenn der Skalierungsfaktor für die Positionswerte verändert wird, dann muss ein im Anwenderprogramm verwendeter Kalibrierfaktor zur Berechnung der RTC6-Ansteuerwerte (in Bits) aus den gewünschten Positionswerten im Bildfeld (in mm), siehe Kapitel 7.3.2 "Bildfeld-Größe und Bildfeld-Kalibrierung", Seite 175, umgekehrt proportional skaliert werden. Außerdem sollten die Sprunggeschwindigkeit und Markiergeschwindigkeit angepasst werden.
  - Mit `control_command`(`Data` = $053F_H$) kann der aktuell eingestellte Skalierungsfaktor ausgelesen werden.

- Zu **SetPosAcknowledgeLevel**
  - Der Grenzwert muss bezogen auf einen 16-Bit-Ansteuerbereich angegeben werden.
  - Das Startverhalten ist standardmäßig so voreingestellt, dass das Scan-System nach jedem Neustart oder Reset einen PosAck-Grenzwert von 0,28% des *vollen* Winkelbereichs einer Achse (LSB16 oder 1/65536, $Code_{LOW}$ = 183 = $B7_H$) einstellt. Wenn ein anderer Grenzwert gewünscht wird, dann muss er – für jede Achse separat – neu eingestellt werden ($Code_{HIGH}$ = 15_H). SCANLAB empfiehlt nur Grenzwerte ($Code_{LOW}$) > 14_H (d.h. 0,03% des vollen Positionsbereichs). Niedrigere Werte können zu häufigem Abschalten des Systems aufgrund von Position Acknowledge Time-outs (Sollposition lange nicht erreicht) führen.

- Zu **Store/RestoreTransmissionMode**
  - `control_command`(`Data` = 1700_H) bleibt wirkungslos, wenn nach dem letztmaligen Aufruf von `control_command`(`Data` = $17FF_H$) ein Neustart oder Reset durchgeführt wurde.

21 Anhang F: Über Rückkanal-Multiplexing und Erweiterter Scan-Kopf-Status

Nicht-Rückkanal-Multiplexing-fähige *iDRIVE*-Scan-Systeme transportieren je 10 µs-Takt nur 1 Signaltyp auf dem Rückkanal, also z.B.:

- *Entweder* die Galvanometerscanner-Position
- *Oder* ein "anderes" Signal(1)

Mehr Möglichkeiten der Signal-Übertragung auf dem Rückkanal eröffnen sich durch die Einführung von Zeitmultiplexverfahren.

iDRIVE-Scan-Systeme, die diese unterstützen, können nunmehr gleichzeitig mehrere Signaltypen je Rückkanal-Multiplexing-Block(2)(a) in einem gemeinsamen Übertragungskanal (hier: der Rückkanal) übertragen(3).

Das Kriterium zum Unterscheiden der Signalübertragungsmöglichkeiten ist die zeitliche Auflösung:

- Signale mit hoher Aktualisierungsrate, z.B. Positionen.
  Diese werden in einer möglichst hohen zeitlichen Auflösung benötigt (= jeder Block(2)).
  Dafür steht das Rückkanal-Multiplexing, Seite 1128 (RCM, Return Channel Multiplexing) zur Verfügung.
- Signale mit niedriger Aktualisierungsrate, z.B. Seriennummer.
  Diese werden nur in einer geringen zeitlichen Auflösung benötigt (= nicht jeder Block(2)).
  Bereitgestellt werden sie über den Erweiterter Scan-Kopf-Status, Seite 1134.

(1) Würde zudem in – aus Benutzer-Sicht – in nicht brauchbaren Zykluszeiten abgerufen.

(2) Nicht: "SL2-100-Block", siehe Kurzinformation SL2-100-Protokoll, Seite 1127.

(3) Allerdings geht dieses gleichzeitige Übertragen mehrerer Signale zu Lasten einer geringeren zeitlichen Auflösung.

Kurzinformation SL2-100-Protokoll

Das SL2-100-Protokoll:

- Ist bidirektional
- Ist seriell
- Erfordert 2 getrennte Kanäle
  - Hinkanal
    (hier aus Sicht der RTC-Karte bezeichnet)
    - 1 Kanal für die Richtung RTC-Karte → i*DRIVE*-Scan-System
    - In Abbildung 15: Kabel-Ader von Pin 5 → Pin 1 (blau) und Kabel-Ader von Pin 9 → Pin 6
  - Rückkanal
    (hier aus Sicht der RTC-Karte bezeichnet)
    - 1 Kanal für die Richtung i*DRIVE*-Scan-System → RTC-Karte
    - In Abbildung 15: Kabel-Ader von Pin 5 → Pin 1 (rot) und Kabel-Ader von Pin 9 → Pin 6
- Die Daten im Hinkanal und Rückkanal werden in Blöcken transportiert
  ("SL2-100-Block", "Übertragungsblock")
- Beim SL2-100-Protokoll sind Hinkanal und Rückkanal vollständig voneinander unabhängig
- Dementsprechend haben beide ihre eigenen SL2-100-Blöcke
  (= SL2-100-Hinkanal-Block und SL2-100-Rückkanal-Block)
- Ein Blockstart in die eine Richtung bedeutet nicht automatisch ein Blockstart in die andere Richtung[(a)]
- Es gilt:
  - 1 Block besteht aus 192 Frames = "Block-Länge"
  - Jeder Block beginnt mit einer (durch die RTC-Karte erzeugten) Präambel
  - 1 Frame besteht aus 2 Sub-Frames
  - 1 Frame wird in 10 μs übertragen
  - 1 Sub-Frame enthält
    - 20 Bit Nutzdaten
      (Ansteuerwerte oder Statuswerte)
    - 12 Bit weitere Informationen
      (z.B. zur Synchronisation)

(a) Beim Rückkanal-Multiplexing wird daher nur auf den Start des Rückkanal-Blocks synchronisiert. Rückkanal-Multiplexingist vom Hinkanal an sich unabhängig. Außerdem ist die maximale Rückkanal-Multiplexing-Blocklänge 16.

21.1 Rückkanal-Multiplexing

Voraussetzungen

- Senderseitig
  SCANLAB i*DRIVE*-Scan-System
  – Muss das SL2-100-Protokoll nativ unterstützen d.h. keinen Konverter benutzen!
  – Muss eine Scan-System-Firmware installiert haben, die Folgendes unterstützt:(1)
    - `control_command`(`Data` = 65nn$_H$)
    - `control_command`(`Data` = 66nn$_H$)
    - `control_command`(`Data` = 0574$_H$)
- Empfängerseitig
  RTC6-Karte (auch: RTC6 Ethernet-Karte)
  RTC6-Software-Paket ≥ 1.19.1

Beim Rückkanal-Multiplexing überträgt das i*DRIVE*-Scan-System (= Sender) nun mehrere Signale zeitlich nacheinander (= Zeitmultiplexing) im Rückkanal-Block. Zur Unterscheidbarkeit wird im Folgenden dieser modifizierte SL2-100-Block als "Rückkanal-Multiplexing-Block" bezeichnet.

Rückkanal-Multiplexing ist für Signale entwickelt, die sich häufig ändern (= hohe Aktualisierungsrate). Für solche wird eine möglichst hohe (= jeder Block(2)) zeitliche Auflösung gefordert, wie z.B.

- Soll-Position
- Ist-Position
- Ist-Geschwindigkeit

Es gilt:

- Jedes Signal wird mit der vollen Auflösung von 20 Bit je 10 µs-Takt übertragen
- Bei n Signalen ist die zeitliche Auflösung deswegen 100 kHz / n

Mit Rückkanal-Multiplexing können Sie die gleichen Signale übertragen lassen wie mit **SetMode**. Die *möglichen* Code$_{LOW}$-Werte (und deren Bedeutung) sind also gleich.

Grundsätzliches Vorgehen

(1) Mit entsprechenden `control_command`-Aufrufen(3) legen Sie fest:
  – Die zu übertragenden Signale
    - deren Abfolge
    - deren jeweilige "Block-Länge"

  Damit die Daten (mittels Präambel(2)) aus Hinkanal-Block und Rückkanal-Block synchronisierbar sind, müssen Sie die Rückkanal-Multiplexing-"Block-Länge" einstellen. Zulässige Werte: 1; 2; 3; 4; 6; 8; 12; 16 (= ganzzahliger Teiler von 192 ≥ 16). Siehe Kapitel 21.1.1 "Konfigurieren, welche Werte übertragen werden sollen (Beispiel-Code)", Seite 1129

(2) Übertragung (durch das iDRIVE-Scan-System) auslösen.

(3) Aufzeichnen durch RTC-Karte mit **set_trigger[*]**-Aufruf
  – dezidierter Signaltyp (63, 64, 65, 66)

(4) Auslesen mit **get_waveform** oder Data-Streaming.

(5) Werte zuordnen (Demultiplex).
  Sie müssen das Demultiplexing selber implementieren. Dazu müssen Sie u.a. wissen, welche Konfiguration ("Block-Länge", Signale inkl. Reihenfolge) in Schritt 1 eingestellt ist.

Hinweise

- Rückkanal-Multiplexing und "Spot Distance Control" sind *nicht* kompatibel.

(1) Anderenfalls verhält sich Rückkanal-Multiplexing undefiniert.

(2) Siehe Kurzinformation SL2-100-Protokoll, Seite 1127.

(3) `control_command` wirkt immer nur auf 1 Achse, siehe Funktionssignatur `control_command`(`Head`, `Axis`, `Data`).

21.1.1 Konfigurieren, welche Werte übertragen werden sollen (Beispiel-Code)

Folgender Beispiel-Code stellt mit control_command-Aufrufen das i*DRIVE*-Scan-System 1 so ein, dass es mittels Rückkanal-Multiplexing-Signal-Werte der x-Achse überträgt.

```
// Pseudo-Code, nicht ausführbar

// Gebe für die x-Achse von iDRIVE-Scan-System 1
// 4 Signal-Werte aus.
// Die Rückkanal-Multiplexing-"Block-Länge"
// muss ein ganzzahliger Teiler von 192 ≤ 16 sein:
// 1; 2; 3; 4; 6; 8; 12; 16.
// Rückkanal-Multiplexing-
// "Block-Länge" = 4
control_command( 1, 1, 0x6503 );

// Signal 0 konfigurieren = Konfiguriere den Index 0
// An welcher Stelle wird der erste Datentyp übertragen
control_command( 1, 1, 0x6580 );
// Signal 0 = 0x01 (Ist-Position)
control_command( 1, 1, 0x6601 );

// Signal 1 konfigurieren = Konfiguriere den Index 1
// An welcher Stelle wird der zweite Datentyp übertragen
control_command( 1, 1, 0x6581 );
// Signal 1 = 0x02 (Soll-Position)
control_command( 1, 1, 0x6602 );

// Signal 2 konfigurieren = Konfiguriere den Index 2
control_command( 1, 1, 0x6582 );
// Signal 2 = 0x06 (Ist-Geschwindigkeit)
control_command( 1, 1, 0x6606 );

// Signal 3 konfigurieren = Konfiguriere den Index 3
control_command( 1, 1, 0x6583 );
// Signal 3 = 0x04 (Ist-Endstufen-Strom)
control_command( 1, 1, 0x6604 );

// Übertragung durch das iDRIVE-Scan-System starten
control_command( 1, 1, 0x0574 );

// Aufzeichnen durch RTC-Karte wie üblich
// jedoch mit dezidiertem Signaltyp (63, 64, 65, 66).
// set_trigger/set_trigger4/set_trigger8-Aufruf

// Auf den PC herunterladen (auslesen) wie üblich
// get_waveform-Aufruf

// Demultiplexing-Abschnitt, siehe Kapitel 21.1.3 "Werte zuordnen (Demultiplex)", Seite 1132
```

Folgende Tabelle zeigt beispielhaft einen Ausschnitt aus einer Rückkanal-Multiplexing-Datenaufzeichnung.

Waveform-Index	Hinkanal-Block	Rückkanal-Block[a]	Daten	Signal
…	…	…	…	…
123456	190	2	0x000150	Ist-Geschwindigkeit
123457	191	3	0x000160	Ist-Endstufenstrom
123458	0	0	0x100040	Ist-Position
123459	1	1	0x000080	Soll-Position
123460	2	2	0x000150	Ist-Geschwindigkeit
123461	3	3	0x000160	Ist-Endstufenstrom
…	…	…	…	…

(a) Die eingestellte Rückkanal-Multiplexing-"Block-Länge" ist "4".

- Die Spalte `Waveform-Index` verdeutlicht, dass es sich um einen Ausschnitt aus einer größeren Menge an aufgezeichneten Daten handelt (also z.B. einige hunderttausend Datenpunkte). Es werden die Einträge an Stelle `123456`…`123461` betrachtet.
- Die Spalte `Hinkanal-Block` zeigt den gedachten Frame-Zähler des SL2-100-Rückkanals. Dieser wird am Anfang jedes Blocks wieder auf 0 gesetzt.
- Die Spalte `Rückkanal-Block` enthält den gedachten Index des Rückkanal-Multiplexing-Rückkanals. Zum SL2-100-Blockstart (Spalte `Hinkanal-Block` = 0) ist er 0, danach zählt er bis zum höchsten Index (N bei `control_command`(`Data` = $650N_H$)) hoch, und wird dann wieder auf 0 gesetzt.
- Die Spalte `Daten` enthält die eigentlichen ausgelesenen Daten.
- Die Spalte `Signal` nennt, um welche Daten es sich handelt.

21.1.2 Übertragung (durch das i*DRIVE*-Scan-System) auslösen

- Um die Übertragung der Signale durch das i*DRIVE*-Scan-System zu starten, rufen Sie **SetMode** mit der Kanal-Nummer 0x74 auf:
  `control_command`(1, 1, 0x0574); (1)
  Daraufhin überträgt das i*DRIVE*-Scan-System die konfigurierten Werte im Rundlauf-Verfahren (siehe Wikipedia-Artikel; englisch: Round-Robin).

Hinweise

- Es wird immer der Wert zum aktuellen 10 µs-Takt übertragen. Wenn Sie Ist-Position und Soll-Position abwechselnd übertragen lassen, stammen die Werte aus unterschiedlichen 10 µs-Takten. Daher können Sie hier *nicht* auf den Positionsfehler schließen!
- Der Rückkanal-Multiplexing-Block-Längen-Wert muss sein: 1; 2; 3; 4; 6; 8; 12; 16 (= ganzzahliger Teiler von 192 ≤ 16).
  Der Block-Beginn von Hinkanal sowie Rückkanal erfolgt gleichzeitig. Die "Block-Länge" beim SL2-100-Protokoll ist 192. Demnach wird der Rückkanal-Multiplexing-Block abgeschnitten, wenn seine Länge nicht durch 192 teilbar ist (potenzielle Fehlerquelle!).

(1) `control_command` wirkt immer nur auf 1 Achse, siehe Funktionssignatur `control_command`(`Head`, `Axis`, `Data`).

21.1.3 Werte zuordnen (Demultiplex)

Damit Sie später die Werte (anhand der Blockstarts) wieder richtig zuordnen können, stellt **set_trigger** zum Aufzeichnen der Rückkanal-Multiplexing-Werte dezidierte Signal-Typen bereit (ab RTC6-Software-Paket ≥ 1.19.1):

- Signaltyp 63
  = StatusAX_Synced
- Signaltyp 64
  = StatusAY_Synced
- Signaltyp 65
  = StatusBX_Synced
- Signaltyp 66
  = StatusBY_Synced

Nach dem entsprechenden **set_trigger[*]**-Aufruf ist in Bit #19…Bit #0 (eines Sub-Frames) der angeforderte Signal-Wert zu finden.

Wenn alle 192 Werte Bit #20 = 1 gesetzt sind (sonst gilt: Bit #20 = 0), fängt ein neuer Rückkanal-Block an.

So können Sie mit den entsprechenden Hinkanal-Blöcken synchronisieren.

Sie können die Werte selber mit einer benutzerdefinierten Funktion zuordnen (“die Reihe an Zahlen auflösen”), oder statt dessen verwenden:

- **demux_rcm_ctrl**,
  siehe Beispiel-Code (C), Seite 1133

Benötigen Sie nur einen einzelnen Wert, verwenden Sie:

- **get_rcm_value_ctrl**,
  siehe Beispiel-Code (C), Seite 1133

Beispiel-Code (C)

```
// Pseudo-Code, nicht ausführbar
// Es wird angenommen, dass das Scan-System konfiguriert ist gemäß:
// Kapitel 21.1.1 ”Konfigurieren, welche Werte übertragen werden sollen (Beispiel-Code)”, Seite 1129

// Liste zur Datenaufzeichnung erstellen
set_start_list_1();
set_trigger(1, 63, 0);
long_delay(10000);
set_end_of_list();
// Liste ausführen und Ende abwarten
execute_list_1();
UINT status, pos;
do {
   get_status(&status, &pos);
} while (status != 0);

// Daten auslesen
UINT rcm_data[10000];
get_waveform(1, 10000, rcm_data);
// Demultiplexen
UINT actual_positions[2500];
UINT n_actual_positions = demux_rcm_ctrl(rcm_data, 10000, 4, 0, 0, actual_positions);

// Einzelnen Wert auslesen
UINT set_position = get_rcm_value_ctrl(1, 1, 1);
```

21.2 Erweiterter Scan-Kopf-Status

Voraussetzungen

- Sender-seitig
  SCANLAB i*DRIVE*-Scan-System
  - Scan-System-Firmware ≥ 51*7
  - Scan-System-Firmware < 5900
    (Die Versionsnummer der Scan-System-Firmware wird übertragen nach: `control_command( 0x0522 );`)
- Empfänger-seitig
  RTC6-Software-Paket ≥ 1.19.1

Signale, die sich relativ seltener oder gar nicht ändern (= niedrige Aktualisierungsrate), sind z.B.

- Temperaturen
- Spannungen
- Seriennummer
- Firmware
- Konfiguration von Rückkanal-Multiplexing

Für solche Signale ist eine niedrigere (= nicht jeder Block(2)) zeitliche Auflösung ausreichend. Sie werden mittels Erweiterter Scan-Kopf-Status übertragen. Die vollständige Signal-Liste finden Sie in Kapitel 21.2.1 "Erweiterter Scan-Kopf-Status-Signal-IDs", Seite 1135. Die Übertragung dieser Signale durch das i*DRIVE*-Scan-System ist standardmäßig aktiv (siehe Voraussetzungen). Benutzer-seitig ist dazu keine Maßnahme notwendig.

Für Erweiterter Scan-Kopf-Status wird ein vormals nicht genutztes Bit des SL2-100-Protokolls verwendet. Daher gilt:

- Ein Signal (aus Kapitel 21.2.1 "Erweiterter Scan-Kopf-Status-Signal-IDs", Seite 1135) benötigt 320 µs Übertragungszeit
  (= 1 Bit je 10 µs-Takt).
  Entsprechend würden 46 (256) verschiedene Signale alle 14,72 ms (81,92 ms) aktualisiert werden.

Zum Abfragen von Erweiterter Scan-Kopf-Status-Signal-Werten stehen zur Verfügung, siehe Voraussetzungen:

- Kontrollbefehl
  get_extended_head_status_value_ctrl
  Gibt nur einen einzelnen Erweiterter Scan-Kopf-Status-Signal-Wert einer Achse zurück.
  Beispiel-Code:

```
UINT Aperture =
get_extended_head_status_value_ctrl( 1, 1, 0x0
D );
UINT Wavelength =
get_extended_head_status_value_ctrl( 1, 1, 0x0
E );
```

- Kontrollbefehl
  get_extended_head_status_block_ctrl
  Schreibt alle möglichen Erweiterter Scan-Kopf-Status-Signal-Werte einer Achse in das Array `Ptr`.
  Beispiel-Code:

```
UINT Buffer[256];
get_extended_head_status_block_ctrl( 1, 1, (ULONG
_PTR) Ptr );
UINT Aperture = Ptr[0x0D];
UINT Wavelength = Ptr[0x0E];
```

Hinweise

- Erweiterter Scan-Kopf-Status und Data-Streaming sind *nicht* kompatibel.
- Erweiterter Scan-Kopf-Status-Werte können also *nicht* mit **set_trigger[*]** durch die RTC-Karte aufgezeichnet und dann mit **get_waveform** auf den PC heruntergeladen (ausgelesen) werden. Stattdessen wird ein Teil der entsprechenden Daten in den Metadaten eines jeden Data-Streaming-Datenpaket mitgesendet.

21.2.1 Erweiterter Scan-Kopf-Status-Signal-IDs

Einige der mit Erweiterter Scan-Kopf-Status abfragbaren Signale haben eine Entsprechung mit SetMode (unter dem 20-Bit Protokoll), siehe folgende Tabelle.

Erweiterter Scan-Kopf-Status-Signal-ID (hex)	SetMode-Entsprechung (hex)	Beschreibung
00	0514	Temperatur des Galvanometerscanners.
01	0515	Temperatur der Servokarte.
02	0516	AGC-Spannung (Versorgungsspannung Positionsdetektor).
03	0517	DSP Core-Versorgungsspannung.
04	0518	DSP IO-Spannung.
05	0519	Spannung Analogteil.
06	051A	Versorgungsspannung Analog-Digital-Wandler.
07	051B	AGC-Strom (PD-Versorgungsstrom).
–	051C	–
–	051D	–
08	051E	Seriennummer, untere 16 Bits.
09	051F	Seriennummer, obere 16 Bits.

Erweiterter Scan-Kopf-Status-Signal-ID (hex)	SetMode-Entsprechung (hex)	Beschreibung
0A	0520	SCANLAB-Artikelnummer, untere 16 Bits.
0B	0521	SCANLAB-Artikelnummer, obere 16 Bits.
-	0522	-
0C	0523	Kalibrierwinkel.
0D	0524	Apertur.
0E	0525	Wellenlänge.
0F	0526	Tuningnummer.
-	0527	-
-	052D	-
-	052E	-
10	052F	Betriebszeit (Sekunden).
11	0530	Betriebszeit (Minuten).
12	0531	Betriebszeit (Stunden).
13	0532	Betriebszeit (Tage).
-	0533	-
-	0534	-

Erweiterter Scan-Kopf-Status-Signal-ID (hex)	SetMode-Entsprechung (hex)	Beschreibung
–	0535	–
–	0536	–
–	0537	–
–	0538	–
–	0539	–
–	053A	–
–	053B	–
–	053C	–
–	053D	–
–	053E	–
14	053F	Eingestellter Skalierungsfaktor für Positionswerte.
15	0549	Versorgungsspannung (üblicherweise 30 V oder 48 V).
16	055E	Anzahl der "Regulärer Betrieb → Fehlerzustand"-Übergänge.
17	0564	Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.3 "Erweiterte Einstellungen für "Spot Distance Control"", Seite 24: TimeShift relativ zu der von SCANLAB voreingestellten Verschiebung.
18	0565	Siehe Handbuch "excelliSCAN-Scan-Köpfe – Funktionsprinzip der SCANahead-Regelung und Ansteuerung mit RTC6-Karten", Kapitel 2.3 "Erweiterte Einstellungen für "Spot Distance Control"", Seite 24: TimeShift gesamt.

Erweiterter Scan-Kopf-Status-Signal-ID (hex)	SetMode-Entsprechung (hex)	Beschreibung
–	0574	–
19	–	Bit #19…Bit #18: Reserviert. Bit #17…Bit #12: Maximale Anzahl an Rückkanal-Multiplexing-Signalen. Bit #11…Bit #06: Index für Rückkanal-Multiplexing-Konfiguration. Bit #05…Bit #00: Maximaler Index für Rückkanal-Multiplexing-Konfiguration (Block-Länge -- 1).
1A	–	Einstellungen für Tuning 0. Entspricht control_command(0x0E00).
1B	–	Einstellungen für Tuning 1. Entspricht control_command(0x0E01).
1C	–	Einstellungen für Tuning 2. Entspricht control_command(0x0E02).
1D	–	Einstellungen für Tuning 3. Entspricht control_command(0x0E03).
1E	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 1. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 0.
1F	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 3. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 2.
20	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 5. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 4.
21	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 7. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 6.
22	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 9. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 8.

Erweiterter Scan-Kopf-Status-Signal-ID (hex)	SetMode-Entsprechung (hex)	Beschreibung
23	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 11. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 10.
24	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 13. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 12.
25	–	Bit #15…Bit #08: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 15. Bit #07…Bit #00: Return Channel Multiplexing Siehe Rückkanal-Multiplexing, Seite 1128.-Kanal 14.
26	–	Aktuelle **SetMode**-Einstellung.
27	0528	Diagnose-Flags, untere 16 Bits.
28	0529	Diagnose-Flags, obere 16 Bits.
29	052A	Stopp-Ereigniscode.
2A	052B	Flags von letztem Stopp-Event, untere 16 Bit.
2B	052C	Flags von letztem Stopp-Event, obere 16 Bit.
2C	056E	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Vorhandensein.
2D	056A	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Fehler.
2E	0571	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Warnung.
2F	5600	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 0.
30	5601	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 1.

Erweiterter Scan-Kopf-Status-Signal-ID (hex)	SetMode-Entsprechung (hex)	Beschreibung
31	5602	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 2.
32	5603	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 3.
33	5604	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 4.
34	5605	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 5.
35	5606	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 6.
36	5607	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 7.
37	5608	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 8.
38	5609	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 9.
39	560A	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 10.
3A	560B	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 11.
3B	560C	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 12.
3C	560D	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 13.
3D	560E	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 14.
3E	560F	Nur bestimmte Scan-Systeme mit integrierten Sensoren: Sensor-Messwert 15.
–	0500	–
–	0501	–
–	0502	–

Erweiterter Scan-Kopf-Status-Signal-ID (hex)	SetMode-Entsprechung (hex)	Beschreibung
–	0503	–
–	0504	–
–	0505	–
–	0506	–
–	0507	–
–	0508	–
–	0509	–
–	0510	–
–	0511	–
–	0512	–
–	0513	–

22 Anhang G: Änderungsindex

Nachfolgend genannt sind Änderungen an diesem Handbuch aufgrund technischer Weiterentwicklung des Produkts sowie wesentliche redaktionelle Änderungen.

In diesem Kapitel:

- Änderungen nach Dokument-Revision 1.0.0 de-DE von Dokument-Revision 0.0, Seite 1143
- Änderungen nach Dokument-Revision 1.0.1 de-DE von Dokument-Revision 1.0.0 de-DE, Seite 1143
- Änderungen nach Dokument-Revision 1.0.2 de-DE von Dokument-Revision 1.0.1 de-DE, Seite 1144
- Änderungen nach Dokument-Revision 1.0.3 de-DE von Dokument-Revision 1.0.2 de-DE, Seite 1145
- Änderungen nach Dokument-Revision 1.0.4 de-DE von Dokument-Revision 1.0.3 de-DE, Seite 1147
- Änderungen nach Dokument-Revision 1.0.5 de-DE von Dokument-Revision 1.0.4 de-DE, Seite 1148
- Änderungen nach Dokument-Revision 1.0.6 de-DE von Dokument-Revision 1.0.5 de-DE, Seite 1149
- Änderungen nach Dokument-Revision 1.0.7 de-DE von Dokument-Revision 1.0.6 de-DE, Seite 1149
- Änderungen nach Dokument-Revision 1.0.8 de-DE von Dokument-Revision 1.0.7 de-DE, Seite 1150
- Änderungen nach Dokument-Revision 1.0.9 de-DE von Dokument-Revision 1.0.8 de-DE, Seite 1150
- Änderungen nach Dokument-Revision 1.0.10 de-DE von Dokument-Revision 1.0.9 de-DE, Seite 1151
- Änderungen nach Dokument-Revision 1.0.11 de-DE von Dokument-Revision 1.0.10 de-DE, Seite 1151
- Änderungen nach Dokument-Revision 1.0.12 de-DE von Dokument-Revision 1.0.11 de-DE, Seite 1152
- Änderungen nach Dokument-Revision 1.0.13 de-DE von Dokument-Revision 1.0.12 de-DE, Seite 1152
- Änderungen nach Dokument-Revision 1.0.14 de-DE von Dokument-Revision 1.0.13 de-DE, Seite 1153
- Änderungen nach Dokument-Revision 1.0.15 de-DE von Dokument-Revision 1.0.14 de-DE, Seite 1154
- Änderungen nach Dokument-Revision 1.0.16 de-DE von Dokument-Revision 1.0.15 de-DE, Seite 1155
- Änderungen nach Dokument-Revision 1.0.17 de-DE von Dokument-Revision 1.0.16 de-DE, Seite 1156
- Änderungen nach Dokument-Revision 1.0.18 de-DE von Dokument-Revision 1.0.17 de-DE, Seite 1157
- Änderungen nach Dokument-Revision 1.0.19 de-DE von Dokument-Revision 1.0.18 de-DE, Seite 1158
- Änderungen nach Dokument-Revision 1.0.20 de-DE von Dokument-Revision 1.0.19 de-DE, Seite 1159
- Änderungen nach Dokument-Revision 1.1.0 de-DE von Dokument-Revision 1.0.20 de-DE, Seite 1161
- Änderungen nach Dokument-Revision 1.1.1 de-DE von Dokument-Revision 1.1.0 de-DE, Seite 1163
- Änderungen nach Dokument-Revision 1.1.2 de-DE von Dokument-Revision 1.1.1 de-DE, Seite 1165

Änderungen nach Dokument-Revision 1.0.0 de-DE von Dokument-Revision 0.0

Wo	Warum & Was
Global	Erste Dokument-Revision. Dokument-Revision • 1.0.0 de-DE gilt für das RTC6-Software-Paket • V1.5.0[(a)]
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.1 de-DE von Dokument-Revision 1.0.0 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.1 de-DE gilt für das RTC6-Software-Paket • V1.5.2[(a)]
eth_get_com_timeouts, Seite 401	Software-Änderung.[(a)] Reine RTC6-DLL-Einstellungen sind jetzt ohne Zugriff auf eine RTC6 Ethernet-Karte möglich.
eth_set_com_timeouts, Seite 416	Wie **eth_get_com_timeouts**.
load_disk, Seite 554	Software-Änderung.[(a)] Jetzt mit Versionskontrolle.
range_checking, Seite 636	Software-Änderung.[(a)] `Mode` = 2 kommt hinzu: ein **simulate_ext_stop** wird an alle Slave-Karten weitergeleitet.
save_disk, Seite 661	Wie **load_disk**.
set_wobbel_vector, Seite 868	Redaktionelle Erweiterung. “Frei definierbare Wobbelfigur” mit **set_wobbel_control** und `set_wobbel_mode( Mode = 3 )` aktivieren, siehe Seite 870.
sync_slaves, Seite 902	Software-Änderung.[(a)] Hat jetzt keine Funktion mehr.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.2 de-DE von Dokument-Revision 1.0.1 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.2 de-DE gilt für das RTC6-Software-Paket • V1.6.0[a]
Abschnitt "Verzeichnis iSCANCfg", Seite 33	Software-Änderung.[a] `iSCANcfg6.exe` wird im RTC6-Software-Paket durch das generische `iSCANcfg.exe` ersetzt.
Kapitel 7.4.1 "Lasersteuersignale freigeben, aktivieren und schalten"	Redaktionelle Änderung. Abbildung 53, Seite 193.
control_command, Seite 377	Redaktionelle Änderung. Beschreibung von "PowerOK" und "TempOK" geändert, siehe XY2-100-Statuswort. Außerdem wurde die Beschreibung von 0528_H und 0529_H gelöscht (diese Datensignaltypen sind nicht für Benutzer vorgesehen).
get_error, Seite 439	Redaktionelle Änderung. Fehler-Bit `Bit #15` nicht mehr "reserviert", siehe Seite 441.
Kapitel 16.2.13 "Master-Stiftleiste, Slave-Stiftleiste"	Redaktionelle Änderung: Neuer Sicherheitshinweis zu Master/Slave-verbundenen RTC6 Ethernet-Karten, siehe Seite 996.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch `RTC6_Aenderungshistorie.pdf`.

Änderungen nach Dokument-Revision 1.0.3 de-DE von Dokument-Revision 1.0.2 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.3 de-DE gilt für das RTC6-Software-Paket • RTC6-Software-Paket V1.6.1[(a)]
Global	Redaktionelle Änderung. Der bisher verwendete Begriff "Online-Positionierung" lautet (aufgrund der Einführung der "Globale Online-Positionierung") – wo zutreffend – jetzt "Lokale Online-Positionierung", z.B. in Kapitel 8.3.1 ""Lokale Online-Positionierung"", Seite 240.
Kapitel 8.3.2 ""Globale Online-Positionierung"", Seite 243	Redaktionelle Erweiterung. Beschreibung der "Globale Online-Positionierung".
Kapitel 8.4 "Wobbelmode"	Redaktionelle Erweiterung. Abschnitt "Beispiel-Code (C++)", Seite 245.
Kapitel 8.6.12 ""Fly-Erweiterung"-Befehle", Seite 271	Redaktionelle Erweiterung. Beschreibung der "Fly-Erweiterung"-Befehle.
activate_fly_1_axis, Seite 348	Software-Änderung.[(a)] Neuer Befehl.
activate_fly_2_axes, Seite 349	Software-Änderung.[(a)] Neuer Befehl.
control_command, Seite 377	Redaktionelle Änderung. Die Beschreibung von $052B_H$ und $052C_H$ wurde gelöscht (diese Datensignaltypen sind nicht für Benutzer vorgesehen).
fly_return_1_axis, Seite 429	Software-Änderung.[(a)] Neuer Befehl.
fly_return_2_axes, Seite 430	Software-Änderung.[(a)] Neuer Befehl.
fly_return_3_axes, Seite 431	Software-Änderung.[(a)] Neuer Befehl.
get_startstop_info, Seite 473	Software-Änderung.[(a)] Neues Bit #14.
park_position_1_axis, Seite 626	Software-Änderung.[(a)] Neuer Befehl.
park_position_2_axes, Seite 627	Software-Änderung.[(a)] Neuer Befehl.
park_return_1_axis, Seite 630	Software-Änderung.[(a)] Neuer Befehl.
park_return_2_axes, Seite 631	Software-Änderung.[(a)] Neuer Befehl.
set_fly_1_axis, Seite 710	Software-Änderung.[(a)] Neuer Befehl.
set_fly_2_axes, Seite 711	Software-Änderung.[(a)] Neuer Befehl.
set_fly_3_axes, Seite 715	Software-Änderung.[(a)] Neuer Befehl.
set_mcbsp_global_matrix, Seite 762	Software-Änderung.[(a)] Neuer Befehl.
set_mcbsp_global_matrix_list, Seite 763	Software-Änderung.[(a)] Neuer Befehl.
set_mcbsp_global_rot, Seite 764	Software-Änderung.[(a)] Neuer Befehl.
set_mcbsp_global_rot_list, Seite 764	Software-Änderung.[(a)] Neuer Befehl.
set_mcbsp_global_x, Seite 765	Software-Änderung.[(a)] Neuer Befehl.

Wo (Forts.)	Warum & Was (Forts.)
set_mcbsp_global_x_list, Seite 765	Software-Änderung.[a] Neuer Befehl.
set_mcbsp_global_y, Seite 766	Software-Änderung.[a] Neuer Befehl.
set_mcbsp_global_y_list, Seite 766	Software-Änderung.[a] Neuer Befehl.
store_timestamp_counter, Seite 894	Software-Änderung.[a] Neuer Befehl.
store_timestamp_counter_list, Seite 894	Software-Änderung.[a] Neuer Befehl.
wait_for_1_axis, Seite 934	Software-Änderung.[a] Neuer Befehl.
wait_for_2_axes, Seite 936	Software-Änderung.[a] Neuer Befehl.
wait_for_timestamp_counter, Seite 944	Software-Änderung.[a] Neuer Befehl.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.4 de-DE von Dokument-Revision 1.0.3 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.4 de-DE gilt für das RTC6-Software-Paket • V1.7.0[(a)]
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[(a)] Neue Datei. ≥ BIOS-ETH 26 ist eine Voraussetzung für die Nutzung der Standalone-Funktionalität.
eth_boot_dcmd, Seite 389	Software-Änderung.[(a)] Neuer Befehl.
set_eth_boot_control, Seite 705	Software-Änderung.[(a)] Neuer Befehl.
eth_boot_timeout, Seite 389	Software-Änderung.[(a)] Neuer Befehl.
set_free_variable, Seite 729	Software-Änderung.[(a)] Geänderter Befehl.
read_image_eth, Seite 642	Software-Änderung.[(a)] Neuer Befehl.
store_program, Seite 893	Software-Änderung.[(a)] Neuer Befehl.
write_image_eth, Seite 956	Software-Änderung.[(a)] Neuer Befehl.
Kapitel 16.2.15 "Echtzeit-Uhr", Seite 1000	Redaktionelle Erweiterung. Beschreibung der Echtzeit-Uhr für RTC6 Ethernet-Karten.
Kapitel 16.8 "Standalone-Funktionalität", Seite 1012	Software-Änderung.[(a)] Neue Funktionalität für RTC6 Ethernet-Karten: Standalone-Funktionalität.
Kapitel 18 "Anhang C: RTC6 EtherBox", Seite 1073	Redaktionelle Erweiterung. Beschreibung der RTC6 EtherBox.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.5 de-DE von Dokument-Revision 1.0.4 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.5 de-DE gilt für das RTC6-Software-Paket • V1.7.5[a]
Kapitel 4.6.1 "LASER-Buchse", Seite 75	Redaktionelle Erweiterung.Hinweis für laserDESK-Benutzer auf externe Dokumente.
eth_boot_timeout, Seite 389	Redaktionelle Änderung. Befehlsname lautet **eth_boot_timeout** (nicht: **set_eth_boot_timeout**).
eth_get_com_timeouts_auto, Seite 402	Software-Änderung.[a] Neuer Befehl.
eth_set_com_timeouts, Seite 416	Software-Änderung.[a] Geänderter Befehl.
eth_set_com_timeouts_auto, Seite 417	Software-Änderung.[a] Neuer Befehl.
get_head_status, Seite 449	Redaktionelle Änderung. Beschreibung berichtigt.
para_mark_abs, Seite 619	Software-Änderung.[a] Geänderter Befehl.
wait_for_timestamp_counter_mode, Seite 946	Software-Änderung.[a] Neuer Befehl.
Kapitel 16.2.4 "LASER-Stiftleiste", Seite 984	Redaktionelle Erweiterung. Hinweis für laserDESK-Benutzer auf externe Dokumente.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.6 de-DE von Dokument-Revision 1.0.5 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.6 de-DE gilt für das RTC6-Software-Paket • V1.7.6[a]
Abschnitt "Verzeichnis RTC6 Files", Seite 34	Software-Änderung.[a] Andere interne Struktur des RTC6-Software-Pakets sowie weitere Dateien.
get_startstop_info, Seite 473	Software-Änderung.[a] Neues Bit #15.
set_laser_delays, Seite 743	Redaktionelle Änderung. Default-Einstellung nach **load_program_file** berichtigt.
wait_for_encoder_mode, Seite 941	Redaktionelle Änderung. Zulässiger Wertebereich von `Value` berichtigt.
wait_for_encoder_in_range_mode, Seite 940	Redaktionelle Änderung. Zulässiger Wertebereich von `EncXmin` berichtigt.
eth_configure_link_loss, Seite 393	Software-Änderung.[a] Neuer Befehl.
Kapitel 16.5.4 "Verbindung zur RTC6 Ethernet-Karte prüfen", Seite 1009	Redaktionelle Erweiterung.
Kapitel 16.11.3 "TI-RTOS", Seite 1064	Redaktionelle Erweiterung. Die RTC6 Ethernet-Karte verwendet Drittanbieter-Software.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch `RTC6_Aenderungshistorie.pdf`.

Änderungen nach Dokument-Revision 1.0.7 de-DE von Dokument-Revision 1.0.6 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.7 de-DE gilt für das RTC6-Software-Paket • V1.7.7[a]
load_program_file, Seite 567	Redaktionelle Erweiterung.
Kapitel 18 "Anhang C: RTC6 EtherBox", Seite 1073	Hardware-Änderung. Abbildung 97, Abbildung 98.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch `RTC6_Aenderungshistorie.pdf`.

Änderungen nach Dokument-Revision 1.0.8 de-DE von Dokument-Revision 1.0.7 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.8 de-DE gilt für das RTC6-Software-Paket • V1.7.8[(a)]
Kapitel 8.12 "Zeitmessungen", Seite 293	Redaktionelle Erweiterung. 64-Bit-"Zeitstempel-Zähler".
get_timestamp_long, Seite 485	Software-Änderung.[(a)] Neuer Befehl.
wait_for_timestamp_counter_long, Seite 945	Software-Änderung.[(a)] Neuer Befehl.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.9 de-DE von Dokument-Revision 1.0.8 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.9 de-DE gilt für das RTC6-Software-Paket • V1.7.9[(a)]
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Redaktionelle Erweiterung.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.10 de-DE von Dokument-Revision 1.0.9 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.10 de-DE gilt für das RTC6-Software-Paket • V1.7.11[(a)]
Kapitel 7.4.11 "Puls-Synchronisations-Modus", Seite 220	Software-Änderung.[(a)]
Kapitel 8.2 "Koordinatentransformationen", Seite 236	Redaktionelle Erweiterung. Hinweise zur Datenaufzeichnung, Seite 238.
eth_get_last_error, Seite 406	Software-Änderung.[(a)] Neues Bit 27.
regulation3, Seite 651	Software-Änderung.[(a)] Neuer Befehl.
set_defocus_list, Seite 690	Redaktionelle Erweiterung. Neuer Hinweis, Seite 690.
set_duty_cycle_table, Seite 698	Software-Änderung.[(a)] Neuer Befehl.
set_laser_pulse_sync, Seite 749	Software-Änderung.[(a)] Neuer Befehl.
set_laser_timing_table, Seite 753	Software-Änderung.[(a)] Neuer Befehl.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.11 de-DE von Dokument-Revision 1.0.10 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.11 de-DE gilt für das RTC6-Software-Paket • V1.7.11[(a)]
Kapitel 18 "Anhang C: RTC6 EtherBox", Seite 1073	Produkt umbenannt von "RTC6 Ethernet Plug-in+Box" zu "RTC6 EtherBox". Gelöscht: #0145963, #0146152, #0146155, #0146368, #0146371. Hinzugefügt: #0148520. #0148519.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.12 de-DE von Dokument-Revision 1.0.11 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.12 de-DE gilt für das RTC6-Software-Paket • V1.7.12[(a)]
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[(a)] BIOS-ETH 29.
Kapitel "Mindest-BIOS-ETH-Version sicherstellen"	Software-Änderung.[(a)] BIOS-ETH 29.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.13 de-DE von Dokument-Revision 1.0.12 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.13 de-DE gilt für das RTC6-Software-Paket • V1.7.12[(a)]
control_command, Seite 377	Redaktionelle Änderung. $052A_H$: Stop-Ereigniscode $000D0_H$ ist nicht für Benutzer vorgesehen.
Kapitel 14.1 "EU-Konformitätserklärung – RTC6 PCIe-Karte", Seite 971	Redaktionelle Änderung. Ersetzt das Kapitel "Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV)".
Kapitel 16.11.1 "EU-Konformitätserklärung – RTC6 Ethernet-Karte", Seite 1063	Redaktionelle Änderung. Ersetzt das Kapitel "Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV)".
Kapitel 18.4.1 "EU-Konformitätserklärung – RTC6 EtherBox", Seite 1078	Redaktionelle Erweiterung.
Kapitel 18.4.3 "Konformität mit FCC-Bestimmungen", Seite 1080	Redaktionelle Erweiterung.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.14 de-DE von Dokument-Revision 1.0.13 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.14 de-DE gilt für das RTC6-Software-Paket • V1.9.0[(a)]
Abschnitt "Unterverzeichnis Linux", Seite 34	Software-Änderung.[(a)] Neues Unterverzeichnis.
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[(a)] BIOS-ETH 30.
clear_fly_overflow, Seite 371	Redaktionelle Änderung. `Mode` = 63 (nicht: = 15).
clear_fly_overflow_ctrl, Seite 371	Software-Änderung.[(a)] Neuer Befehl.
eth_get_standalone_status, Seite 410	Software-Änderung.[(a)] Neuer Befehl.
get_rtc_version, Seite 467	Software-Änderung.[(a)] Bit #11.
set_sky_writing, Seite 819	Redaktionelle Erweiterung. RTC5→RTC6.
set_sky_writing_list, Seite 821	Redaktionelle Erweiterung. RTC5→RTC6.
set_sky_writing_para, Seite 826	Redaktionelle Erweiterung. RTC5→RTC6.
set_sky_writing_para_list, Seite 828	Redaktionelle Erweiterung. RTC5→RTC6.
Kapitel 16.1.3 "Optionen", Seite 980	Neue Option "LDSA" (nur RTC6 Ethernet-Karten).
Kapitel 16.8.5 "Automatisches Booten – Ablauf im Detail", Seite 1018	Redaktionelle Erweiterung.
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.[(a)] V1.9.0.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch `RTC6_Aenderungshistorie.pdf`.

Änderungen nach Dokument-Revision 1.0.15 de-DE von Dokument-Revision 1.0.14 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.15 de-DE gilt für das RTC6-Software-Paket • V1.9.0[(a)]
Kapitel 2.10 "Hinweise an RTC4-Benutzer", Seite 49	Redaktionelle Erweiterung. Encoderzähler-Richtung von RTC4-Karten, siehe Achtung!, Seite 59.
control_command, Seite 377	Redaktionelle Änderung. Inhalte verschoben nach Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083.
list_repeat, Seite 544	Redaktionelle Erweiterung. Neuer Hinweis, Seite 544.
Kapitel 16.3.1 "Hardware-Installation", Seite 1001	Redaktionelle Erweiterung. Abbildung 92: Stromversorgung mehrerer RTC6 Ethernet-Karten/RTC6 EtherBoxen.
Kapitel 18.5 "Technische Spezifikationen – RTC6 EtherBox", Seite 1080	Redaktionelle Erweiterung.
Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083	Redaktionelle Erweiterung.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.16 de-DE von Dokument-Revision 1.0.15 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.16 de-DE gilt für das RTC6-Software-Paket • V1.11.0[a]
Kapitel 1 "Über dieses Handbuch", Seite 25	Redaktionelle Änderung.
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[a] BIOS-ETH 32.
acquire_rtc, Seite 346	Redaktionelle Erweiterung. Neuer Hinweis, Seite 347.
set_wobbel_control, Seite 860	Software-Änderung.[a] `Ctrl` = 7, `Ctrl` = 8.
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.[a] V1.10.0, V1.11.0.
Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083	Redaktionelle Änderung. Modul Rev.1.0.1 de-DE.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch `RTC6_Aenderungshistorie.pdf`.

Änderungen nach Dokument-Revision 1.0.17 de-DE von Dokument-Revision 1.0.16 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.17 de-DE gilt für das RTC6-Software-Paket • V1.11.0[(a)]
Kapitel 2.8.6 "Slotblech mit 15-poliger Sub-D-Buchse und 9-poliger Sub-D-Buchse", Seite 47	Redaktionelle Erweiterung. #0130209.
Kapitel 16.3.1 "Hardware-Installation", Seite 1001	Redaktionelle Änderung. Abbildung 92.
Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083	Redaktionelle Änderung. Modul Rev.1.0.2 de-DE.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.0.18 de-DE von Dokument-Revision 1.0.17 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.18 de-DE gilt für das RTC6-Software-Paket • V1.12.0[a]
Kapitel 1 "Über dieses Handbuch", Seite 25	Redaktionelle Änderung.
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[a] BIOS-ETH 33.
Abschnitt "z-Achsen-Kalibrierung überprüfen", Seite 178	Redaktionelle Erweiterung. Hinweise, Seite 179 und RTC6-Befehle aus V1.12.0.
get_temperature, Seite 482	Software-Änderung.[a] Neuer Befehl.
load_z_table_20b, Seite 576	Software-Änderung.[a] Neuer Befehl.
load_z_table_no_20b, Seite 578	Software-Änderung.[a] Neuer Befehl.
read_abc_from_file_20b, Seite 639	Software-Änderung.[a] Neuer Befehl.
set_wobbel_control, Seite 860	Software-Änderung.[a] Bei `Ctrl` = 8 wird nun bei jedem Mikroschritt gewechselt (nicht mehr bei jedem **set_wobbel_vector**-Aufruf).
set_wobbel_vector_2, Seite 872	Software-Änderung.[a] Neuer Befehl.
write_abc_to_file_20b, Seite 950	Software-Änderung.[a] Neuer Befehl.
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.[a] V1.12.0.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch `RTC6_Aenderungshistorie.pdf`.

Änderungen nach Dokument-Revision 1.0.19 de-DE von Dokument-Revision 1.0.18 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.19 de-DE gilt für das RTC6-Software-Paket • V1.13.0[a]
Kapitel 1 "Über dieses Handbuch", Seite 25	Redaktionelle Änderung.
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[a] BIOS-ETH 33.
Kapitel 7.3.4 "3D-Bildfeld", Seite 179	Redaktionelle Erweiterung. ABC-Werte aus der `*_ReadMe.txt`-Datei zur Korrekturdatei sind als 16-Bit-Werte zu verstehen.
load_z_table_20b, Seite 576	Redaktionelle Änderung. Zulässiger Wertebereich für `A`, `B`, `C`.
load_z_table_no_20b, Seite 578	Redaktionelle Änderung. Zulässiger Wertebereich für `A`, `B`, `C`.
set_mcbsp_out_ptr, Seite 775	Software-Änderung.[a] `Number`-Bit 31.
set_mcbsp_out_ptr_list, Seite 777	Software-Änderung.[a] Neuer Befehl.
set_wobbel_vector, Seite 868	Software-Änderung.[a] Berechnung für die Variation der aktuellen Laserleistung P nach `set_wobbel_mode`(`Mode` = 2): Formel, Seite 869 (statt: P = P0 × (1 + `dPower` × n)).
set_wobbel_vector, Seite 868	Software-Änderung.[a] Berechnung für die Variation der aktuellen Laserleistung P nach `set_wobbel_mode`(`Mode` = 3): Formel, Seite 870 (statt: P = P100 × (Faktor + `dPower` × n).
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.[a] V1.13.0.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch `RTC6_Aenderungshistorie.pdf`.

Änderungen nach Dokument-Revision 1.0.20 de-DE von Dokument-Revision 1.0.19 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.0.20 de-DE gilt für das RTC6-Software-Paket • V1.14.1[a]
Kapitel 1 "Über dieses Handbuch", Seite 25	Redaktionelle Änderung.
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[a] BIOS-ETH 35.
Kapitel 4.4 "Master-Stiftleiste, Slave-Stiftleiste", Seite 68	Hardware-Änderung. #0149963 (vorher: #0117241).
eth_set_high_performance_mode, Seite 418	Software-Änderung.[a] Neuer Befehl.
get_error, Seite 439	Software-Änderung. Fehler-Bit Bit #17 nicht mehr "reserviert", siehe Seite 441.
get_list_pointer, Seite 457	Redaktionelle Erweiterung. Neuer Hinweis, Seite 457.
set_laser_control, Seite 740	Redaktionelle Erweiterung. Präzisierung bei Bit #3 und Bit #4.
set_mcbsp_out_oie_ctrl, Seite 773	Software-Änderung.[a] Neuer Befehl.
set_mcbsp_out_oie_list, Seite 774	Software-Änderung.[a] Neuer Befehl.
set_vector_control, Seite 854	Software-Änderung. `Ctrl` = 8.
wait_for_encoder_mode, Seite 941	Redaktionelle Änderung. **wait_for_encoder_mode** ist Kategorie "Normaler Listenbefehl" (nicht: "Multipler Listenbefehl").
write_port_list, Seite 959	Software-Änderung.[a] Neuer Befehl.
Kapitel 16.9 ""High-Performance-Modus"", Seite 1020	Software-Änderung.[a] Neue Funktionalität für RTC6 Ethernet-Karten: "High-Performance-Modus".

Wo (Forts.)	Warum & Was (Forts.)
Kapitel 18.2 "Verkabelung", Seite 1076	Hardware-Änderung. #0149964 (vorher: #0148519).
Kapitel 18.5 "Technische Spezifikationen – RTC6 EtherBox", Seite 1080	Gewicht, Seite 1080.
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.[(a)] V1.14.1.
Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083	Redaktionelle Änderung. Modul Rev.1.0.6 de-DE.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.1.0 de-DE von Dokument-Revision 1.0.20 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.1.0 de-DE gilt für das RTC6-Software-Paket • 1.16.3[a]
Kapitel 1 "Über dieses Handbuch", Seite 25	Software-Änderung.[a]
Kapitel 1.2 "Weiterführende Dokumente", Seite 26	Redaktionelle Erweiterung. Handbuch "StreamParser-DLL – Schnittstelle zur Anwendungsprogrammierung"
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[a] BIOS 23, BIOS-ETH 40.
Kapitel 5.5 "RTC6 PCIe-Karten-BIOS upgraden und downgraden", Seite 94	Redaktionelle Erweiterung.
Kapitel 6.4.3 "Listenausführungsstatus", Seite 111	Redaktionelle Erweiterung. HEAD BUSY-Listenausführungsstatus.
Kapitel 8.13 "SCANahead-Funktionalität", Seite 294	Redaktionelle Erweiterung.
Abschnitt "Eingänge für externe Encodersignale", Seite 324	Software-Änderung.[a]
eth_config_waveform_streaming_ctrl, Seite 392	Software-Änderung.[a] Neuer Befehl.
eth_set_remote_tgm_format, Seite 420	Software-Änderung.[a] Neuer Befehl.
get_error, Seite 439	Redaktionelle Erweiterung. Bit #31.
get_transform, Seite 486	Software-Änderung.[a] Siehe Versionsinfo, Seite 486.
get_transform_offset, Seite 487	Software-Änderung.[a] Neuer Befehl.
get_waveform, Seite 496	Software-Änderung.[a] Siehe Versionsinfo, Seite 496.
get_waveform_offset, Seite 497	Software-Änderung.[a] Siehe Versionsinfo, Seite 497.
init_fly_2d_list, Seite 516	Software-Änderung.[a] Neuer Befehl.
set_auto_laser_control, Seite 674	Software-Änderung.[a] Siehe Versionsinfo, Seite 678. M = +64.
set_auto_laser_params, Seite 679	Redaktionelle Erweiterung. Rückgabe.

Wo (Forts.)	Warum & Was (Forts.)
set_ellipse, Seite 699	Software-Änderung.[(a)] Siehe Versionsinfo, Seite 699.
set_encoder_filter_ctrl, Seite 700	Software-Änderung.[(a)] Neuer Befehl.
set_fly_1_axis, Seite 710	Redaktionelle Erweiterung. Parameter für Skalierungsfaktor wird für Modus 8, 12, 16 ignoriert, Seite 710.
set_fly_2_axes, Seite 711	Redaktionelle Erweiterung. Parameter für Skalierungsfaktor wird für Modus 8, 12, 16 ignoriert, Seite 711.
set_fly_3_axes, Seite 715	Redaktionelle Erweiterung. Parameter für Skalierungsfaktor wird für Modus 8, 12, 16 ignoriert, Seite 715.
set_scanahead_params, Seite 810	Redaktionelle Änderung. Bei Fehlercode 3 wird kein **get_last_error**-Fehler gesetzt.
set_trigger, Seite 842	Redaktionelle Erweiterung. Neuer Hinweis, Seite 851.
set_trigger8, Seite 853	Software-Änderung.[(a)] Neuer Befehl.
store_program, Seite 893	Software-Änderung.[(a)] Siehe Versionsinfo, Seite 893.
Kapitel 16.9 ""High-Performance-Modus"", Seite 1020	Software-Änderung.[(a)] Präzisierung.
Kapitel 16.10 "Remote-Interface-Modus", Seite 1021	Software-Änderung.[(a)] Neue Funktionalität für RTC6 Ethernet-Karten: Remote-Interface-Modus.
Kapitel 18.4.2 "Bestätigung elektromagnetische Verträglichkeit (EMV)", Seite 1079	Redaktionelle Erweiterung.
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.[(a)] V1.16.3.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.1.1 de-DE von Dokument-Revision 1.1.0 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.1.1 de-DE gilt für das RTC6-Software-Paket • 1.18.0[a]
Kapitel 1 "Über dieses Handbuch", Seite 25	Software-Änderung.
Abschnitt "Unterverzeichnis Linux", Seite 34	Software-Änderung.[a] Paket für Debian Bookworm (Debian Stretch nicht mehr unterstützt).
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[a] BIOS 23, BIOS-ETH 42.
Abschnitt "Sky-Writing-Modus 4", Seite 166	Software-Änderung.[a] Neue Funktionalität für RTC6-Karten: Sky-Writing-Modus 4.
Abschnitt "Sky-Writing mit Mindest-Markiergeschwindigkeit", Seite 171	Software-Änderung.[a]
Abschnitt "2D-Positions-abhängige Laser-Steuerung", Seite 218	Software-Änderung.[a] Neue Funktionalität für RTC6-Karten: 2D-Positions-abhängige Laser-Steuerung.
Kapitel 8.6.10 "Schleppverzugskompensation von Encoderwerten für "Processing on the fly"-Anwendungen", Seite 269	Software-Änderung.[a]
fly_disable_list, Seite 426	Software-Änderung.[a] Neuer Befehl.
fly_prediction, Seite 427	Software-Änderung.[a] Neuer Befehl.
get_marking_info, Seite 459	Redaktionelle Änderung. Text von `Bit #17` und `Bit #18` war vertauscht.
load_position_control_2d_ctrl, Seite 565	Software-Änderung.[a] Neuer Befehl.
set_auto_laser_control, Seite 674	Software-Änderung.[a] Siehe Versionsinfo, Seite 678. M = +128.
set_auto_laser_control, Seite 674	Redaktionelle Erweiterung. Hinweis, Seite 676.
set_controlpreview_compensation_ctrl, Seite 687	Software-Änderung.[a] Neuer Befehl.

Wo (Forts.)	Warum & Was (Forts.)
set_defocus_2_list, Seite 691	Software-Änderung.[a] Neuer Befehl.
set_encoder_filter_ctrl, Seite 700	Software-Änderung.[a] Siehe Versionsinfo, Seite 701.
set_fly_tracking_error, Seite 721	Software-Änderung.[a] Befehl jetzt mit Funktionalität.
set_short_cmd_mode_ctrl, Seite 818	Software-Änderung.[a] Neuer Befehl.
set_short_cmd_mode_list, Seite 818	Software-Änderung.[a] Neuer Befehl.
set_sky_writing_min_speed_ctrl, Seite 822	Software-Änderung.[a] Neuer Befehl.
set_sky_writing_mode, Seite 824	Software-Änderung.[a] Mode > 3 (Sky-Writing-Modus 4).
set_sky_writing_mode_list, Seite 825	Software-Änderung.[a] Mode > 3 (Sky-Writing-Modus 4).
upload_transform, Seite 931	Redaktionelle Änderung. Anwenderprogramm muss einen Speicherbereich der Größe 528.524 Bytes auf dem PC bereitstellen (nicht: 528.520 Bytes), Seite 932.
Abschnitt "RTC6-Befehle in anderen Publikationen", Seite 961	Redaktionelle Erweiterung.
Abschnitt "Remote-Kontrollbefehle", Seite 1031	Software-Änderung.[a] Neu: ID 158…ID 170.
Abschnitt "Remote-Listenbefehle", Seite 1042	Software-Änderung.[a] Neu: ID 428, ID 429, ID 430.
Kapitel 16.10.10 "Nicht als Remote-Befehl verfügbare RTC6-Befehle", Seite 1054	Redaktionelle Erweiterung.
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.[a] V1.18.0.
Kapitel 20 "Anhang E: iDRIVE-Scan-Systeme – Steuerbefehle und an RTC-Karten übertragene Signale", Seite 1083	Redaktionelle Änderung. Modul Rev.1.0.6 de-DE.
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch RTC6_Aenderungshistorie.pdf.

Änderungen nach Dokument-Revision 1.1.2 de-DE von Dokument-Revision 1.1.1 de-DE

Wo	Warum & Was
Global	Dokument-Revision • 1.1.2 de-DE gilt für das RTC6-Software-Paket • 1.19.1[a]
Kapitel 1 "Über dieses Handbuch", Seite 25	Software-Änderung.[a] V1.19.1.
Kapitel 2.3 "Mitgeliefertes RTC6-Software-Paket", Seite 32	Software-Änderung.[a] Dateien aus dem Stammverzeichnis gelöscht:RTC6_Release_Notes_<SW-V>.pdf, IMPORTANT_RELEASE_NOTES.txt, Liesmich.txt, Readme.txt. Siehe Zeile Abschnitt "Verzeichnis RTC6 Files", Seite 34.
Abschnitt "Verzeichnis RTC6 Files", Seite 34	Software-Änderung.[a] Neue Dateien. release_notes_de.md, release_notes_en.md, IMPORTANT_RELEASE_NOTES.txt, changelog.md, liesmich.txt, readme.txt, RTC6_Release_Notes.pdf, WICHTIGE_RELEASE_HINWEISE.txt, versions.md.
Abschnitt "Verzeichnis RTC6 Files", Seite 34	Software-Änderung.[a] Umbenannte Dateien. RTC6_Aenderungshistorie.pdf (vorher: RTC6_Aenderungshistorie_<SW-V>.pdf), RTC6_RevisionHistory.pdf (vorher: RTC6_RevisionHistory_<SW-V>.pdf).
Abschnitt "Verzeichnis RTC6 Tools", Seite 35	Software-Änderung.[a] BIOS 24, BIOS-ETH 43.
Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 127	Software-Änderung.[a]
Abschnitt "Sky-Writing mit Mindest-Markiergeschwindigkeit", Seite 171	Redaktionelle Erweiterung. Abbildung 47.
Abschnitt "Sky-Writing mit Skalierung der Ausholbewegung", Seite 173	Software-Änderung.[a]
Kapitel 7.4.7 "Softstart-Modus (noch nicht implementiert)", Seite 205	Redaktionelle Erweiterung.

Wo (Forts.)	Warum & Was (Forts.)
demux_rcm_ctrl, Seite 383	Software-Änderung.[(a)] Neuer Befehl.
eth_full_reset, Seite 398	Software-Änderung.[(a)] Neuer Befehl.
eth_get_card_info, Seite 399	Software-Änderung.[(a)] Neue Karteninformation [14] mit ≥ BIOS-ETH 43, siehe Hinweise, Seite 399.
eth_set_remote_tgm_format, Seite 420	Software-Änderung.[(a)] Siehe Versionsinfo, Seite 420.
get_extended_head_status_block_ctrl, Seite 443	Software-Änderung.[(a)] Neuer Befehl.
get_extended_head_status_value_ctrl, Seite 444	Software-Änderung.[(a)] Neuer Befehl.
get_rcm_value_ctrl, Seite 465	Software-Änderung.[(a)] Neuer Befehl.
master_slave_set_offset_ctrl, Seite 601	Software-Änderung.[(a)] Neuer Befehl.
move_to, Seite 610	Redaktionelle Erweiterung. Befehlsbeschreibung jetzt auch auf Deutsch vorhanden (vorher: nur auf Englisch).
set_laser_delays_ctrl, Seite 744	Software-Änderung.[(a)] Neuer Befehl.
set_scanner_delays_ctrl, Seite 815	Software-Änderung.[(a)] Neuer Befehl.
set_sky_writing_min_speed_list, Seite 823	Software-Änderung.[(a)] Neuer Befehl.
set_sky_writing_scale_ctrl, Seite 829	Software-Änderung.[(a)] Neuer Befehl.
set_sky_writing_scale_list, Seite 830	Software-Änderung.[(a)] Neuer Befehl.
set_trigger, Seite 842	Software-Änderung.[(a)] Siehe Versionsinfo, Seite 851.

Wo (Forts.)	Warum & Was (Forts.)
upgrade_bios, Seite 930	Software-Änderung.[(a)] Neuer Befehl.
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **micro_vector_quad_axis**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **micro_vector_quad_axis_v**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **micro_vector_quad_axis_v_2**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **multi_axis_config**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **multi_axis_flags**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_micro_vector_quad_axis**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_micro_vector_quad_axis_v**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_micro_vector_quad_axis_v_2**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_multi_axis_config**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_multi_axis_flags**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_quad_axis_get_status**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_quad_axis_get_values**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **n_quad_axis_init**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **quad_axis_get_status**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **quad_axis_get_values**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[(a)] Gelöscht: **quad_axis_init**

Wo (Forts.)	Warum & Was (Forts.)
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_flags_list**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_get_config_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_get_error_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_get_values_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_init_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_micro_vector_scanner_list**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_micro_vector_stage_list**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **multi_axis_set_config_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_flags_list**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_get_config_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_get_error_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_get_values_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_init_ctrl**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_micro_vector_scanner_list**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_micro_vector_stage_list**
RTC6-Befehle in anderen Publikationen, Seite 961	Software-Änderung.[a] Neu: **n_multi_axis_set_config_ctrl**

Wo (Forts.)	Warum & Was (Forts.)
Kapitel 16.7 "BIOS-ETH-Version upgraden und downgraden", Seite 1011	Redaktionelle Erweiterung.
Kapitel 16.10 "Remote-Interface-Modus", Seite 1021	Redaktionelle Erweiterung. Hinweise, Seite 1021.
Kapitel 16.10.1 "Voraussetzungen für den Remote-Interface-Modus", Seite 1022	Redaktionelle Erweiterung. Präzisierung.
Kapitel "Mindest-BIOS-ETH-Version sicherstellen"	Redaktionelle Änderung. Gelöscht.
Remote-Kontrollbefehle, Seite 1031	Software-Änderung.(a) Neu: ID 171…ID 177.
Remote-Listenbefehle, Seite 1042	Software-Änderung.(a) Neu: ID 431, ID 432.
Kapitel 16.10.10 "Nicht als Remote-Befehl verfügbare RTC6-Befehle", Seite 1054	Software-Änderung.(a) **upgrade_bios**, **eth_full_reset**.
Kapitel 19 "Anhang D: RTC6-Software-Pakete – Historie", Seite 1081	Software-Änderung.(a) V1.19.1.
Kapitel 21 "Anhang F: Über Rückkanal-Multiplexing und Erweiterter Scan-Kopf-Status", Seite 1126	Software-Änderung.(a)
Anhang G: Änderungsindex, Seite 1142	

(a) Siehe auch changelog.md und versions.md.

Notizen