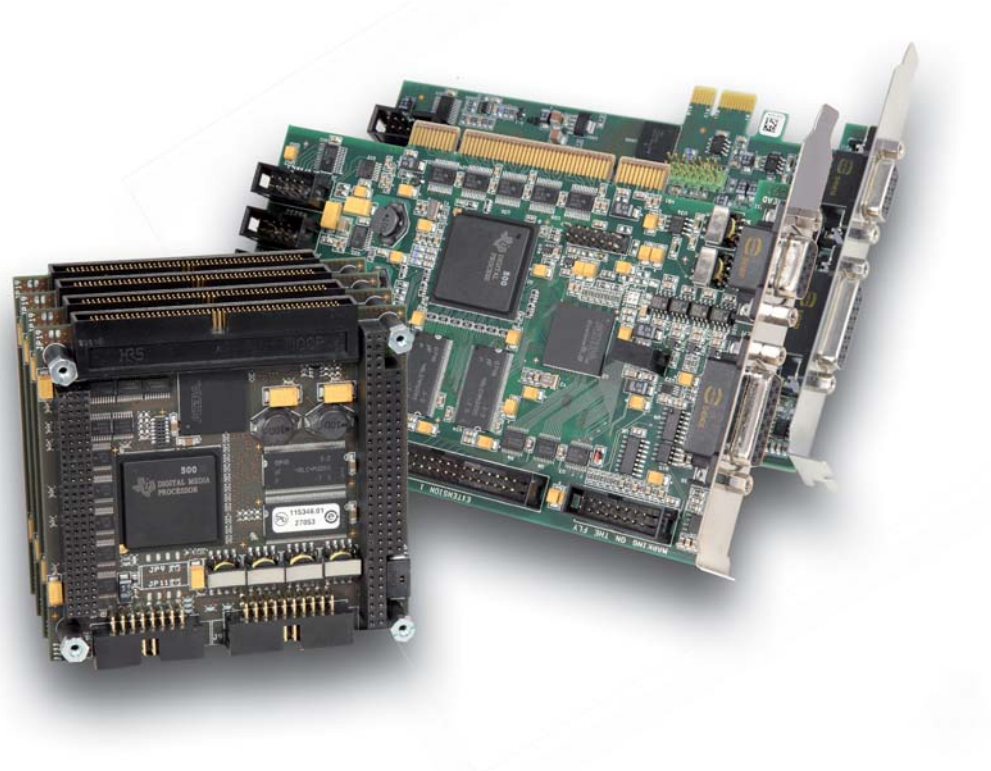
SCANLAB
innovators for industry

Installation und Inbetriebnahme

RTC5 PCI-Karte, RTC5 PCI-Express-Karte, RTC5 PC/104-*Plus*-Karte und RTC5 PCIe/104-Karte

für die Scan-Kopf- und Lasersteuerung in Echtzeit



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 2017
(Doc. Rev. 1.13.0 d - 15. Dezember 2017)

SCANLAB behält sich vor, dieses Dokument jederzeit und ohne Ankündigung inhaltlich zu aktualisieren.
Kein Teil dieser Bedienungsanleitung 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.
RTC ist ein eingetragenes Warenzeichen von SCANLAB GmbH.
PC/104-*Plus* und PCIe/104 sind eingetragene Warenzeichen des "PC/104 Embedded Consortium".
Andere erwähnte Marken unterliegen dem Markenschutz der jeweiligen Markeninhaber.

Inhalt

1 Einleitung 21
1.1 Hersteller 21
1.2 Kennzeichnung 21
1.3 Auspackanleitung und typischer Lieferumfang 22
1.3.1 Mitgelieferte Software 22

2 Produktübersicht 24
2.1 Bestimmungsgemäße Verwendung 24
2.2 Systemanforderungen 26
2.2.1 Hardware 26
2.2.2 Software 26
2.3 Optionale Funktionalitäten 27
2.4 Jumperkonfigurationen und Typenbezeichnung 28
2.4.1 Jumper JP1: Auswahl des Ausgangssignal-Pegels an der EXTENSION 1-Stiftleiste 28
2.4.2 Jumper JP2–JP8: Konfiguration von Pin15 und Pin17 an der EXTENSION 2-Stiftleiste 28
2.4.3 Typenbezeichnung 29
2.5 Zubehör 29
2.5.1 XY2-100-Konverter 29
2.5.2 Datenkabel 29
2.5.3 Laseradapter 29
2.5.4 PCI-Slotbleche 29
2.5.5 ADC-Aufsteckplatine 30
2.5.6 RTC5 varioSCAN FLEX Extension-Karte 30
2.6 Ergänzende Software 30
2.7 Hinweise an Benutzer der RTC4-Karte 31
2.7.1 Hardware-Änderungen 31
Steuerung von Scan-Systemen 31
Lasersteuerung 31
EXTENSION 1-Stiftleiste 31
EXTENSION 2-Stiftleiste 32
MARKING ON THE FLY-Stiftleiste 32
Sonstige Schnittstellen 32
2.7.2 Überarbeiten von RTC4-Programmcodes für die RTC5 32
Veränderte Initialisierung 32
Befehlsänderungen 32
Erhöhte Parameter-Auflösung 33
Verändertes Zeitverhalten 34
2.7.3 Neue und geänderte Funktionalitäten 35
Schnittstelle zum PC 35
Steuerung von Scan-Systemen 35
Lasersteuerung 36
Schnittstellen zur Peripherie 36
Allgemeine Programmierung 37
Laser-Markieren 37
Spezielle Funktionen 38

3 Sicherheit bei Installation und Inbetriebnahme 39
3.1 Betriebssicherheit 39
3.2 Lasersicherheit 39

4 Layout und Schnittstellen 40
4.1 Stecker- und Jumperpositionen 40
4.2 Schnittstelle zum PC 41
4.2.1 Master/Slave-Synchronisation 41
4.3 Schnittstellen zum Scan-System 42
4.3.1 Scan-Kopf-Anschlüsse und Übertragungsprotokoll 42
Primärer Scan-Kopf-Anschluss 42
Sekundärer Scan-Kopf-Anschluss (optional aktiviert) 43
4.3.2 XY2-100-Konverter (optional) 43
4.3.3 Datenkabel (Zubehör) 45
4.4 Schnittstellen zum Laser und zur Peripherie 47
4.4.1 Laseranschluss-Buchse 47
Lasersignale 47
Externe Steuersignale 47
BUSY-Status 47
Digitaler Eingang und Ausgang 48
Analoge Ausgänge 48
Ausgangsbeschaltung 48
Laseradapter (optional) 50
4.4.2 EXTENSION 1-Stiftleiste 51
Auswahl des Ausgangssignal-Pegels 51
16-Bit-Digital-Eingang und -Ausgang 51
Synchronisation der Datenübergabe 51
BUSY-Status 51
4.4.3 EXTENSION 2-Stiftleiste 52
Jumper-Einstellung 52
Lasersignale 52
8-Bit-Digital-Ausgang 52
4.4.4 MARKING ON THE FLY-Stiftleiste 53
Encoder-Eingänge 53
Externe Steuersignale 53
Analoger Ausgang 53
BUSY-Status 53
Slotblech (optional) 53
4.4.5 RS232-Stiftleiste 54
4.4.6 SPI / I2C-Stiftleiste 54
Verwendung als McBSP-Schnittstelle 54
Verwendung als I2C-Schnittstelle 57
Verwendung als SPI-Schnittstelle 57
4.4.7 STEPPER MOTOR-Stiftleiste (Schrittmotoransteuerung) 58
4.4.8 Analog-Eingänge (Sonderzubehör) 59
5 Installation und Inbetriebnahme 60
5.1 Jumpereinstellungen 60
5.2 Einbau der RTC5-Karte 60
5.3 Installation der Treiber 61
5.3.1 Software-Paket-Upgrades 62
5.3.2 Tausch von RTC-Karten und Update von RTC-Karten-Treibern 62
5.4 Installation der RTC5-Software 62
5.5 Sichere Einschalt- und Ausschaltreihenfolge 63
5.6 Funktionstest 63
5.7 Anwenderprogramme und Demoprogramme 64
5.8 Austauschen von RTC5-Karten 64

6 Programmierung von Anwenderprogrammen 65
6.1 Grundlegendes zum RTC5 Software-Konzept 65
6.1.1 Ansteuerung von Scan-Systemen und Lasern – Beispiel zur Einführung 65
6.1.2 Listenbefehle und Kontrollbefehle 65
6.2 Initialisierung und Programmstart 66
6.2.1 DLL-Aufruf 66
6.2.2 Import von Befehlen 66
Pascal 66
C 67
C++ 67
C# 67
6.2.3 Initialisierung der DLL und Kartenverwaltung 68
6.2.4 Inbetriebnahme der RTC5-Karte 69
Initialisierung der Karte 69
Konfiguration der Karte 69
Initialisierung der Scan-System-Ansteuerung 69
Initialisierung der Lasersteuerung 70
Laden und Abarbeiten von Listen 70
6.2.5 Beispiel-Code 71
6.3 Listenspeicher 73
6.3.1 Listen und geschützter Speicherbereich 73
Listen 73
Geschützter Speicherbereich 73
6.3.2 Konfiguration des Listenspeichers 74
6.4 Listen-Handling 75
6.4.1 Listen laden 75
“Unbedingtes” Laden 75
Laden mit Schutzfunktion 75
Abschließen von Listen 76
6.4.2 Listenstatus 76
6.4.3 Listenausführungsstatus 77
6.4.4 Listen ausführen und beenden 78
6.4.5 Unterbrechen von Listen zur Synchronisation der Ausgabe 79
6.4.6 Automatischer Listenwechsel 79
Einmaliger Listenwechsel 79
Alternierender Listenwechsel 80
6.5 Strukturierte Programmierung 81
6.5.1 Unterprogramme 81
Nichtindizierte Unterprogramme 81
Indizierte Unterprogramme 82
Aufruf von Unterprogrammen 83
Nachträgliches Schützen und Konvertieren von nichtindizierten Unterprogrammen 84
Entschützen von Unterprogrammen 85
Indexverwaltung und Defragmentierung 85
6.5.2 Zeichensätze und Zeichenfolgen 87
Definition von indizierten Zeichensätzen 87
Aufruf von indizierten Zeichen 88
Definition von indizierten Zeichenfolgen für Uhrzeit, Datum und Seriennummer 88
Aufruf von indizierten Zeichenfolgen 89
Verwalten von indizierten Zeichen und Zeichenfolgen 89
6.5.3 Sprünge 89
6.5.4 Ringpuffer-Modus 90

6.5.5 Schleifen ... 91
6.6 Verwendung mehrerer RTC5-Karten in einem Rechner ... 92
6.6.1 Multi-Board-Programmierung ... 92
6.6.2 Sequentielle Programmierung ... 93
6.6.3 Master/Slave-Betrieb ... 93
Initialisierung ... 93
Synchronisierung der Takt-Phasen ... 94
Anpassung der Kurzbefehlszählung (ab Version DLL 523, OUT 524) ... 94
Synchrone Starts und Stopps ... 94
Beispiel-Code ... 95
6.7 Verwendung durch mehrere Anwenderprogramme ... 96
6.7.1 Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm ... 96
6.8 Error-Handling ... 98
6.8.1 Download-Überprüfung ... 99
6.8.2 Überprüfung auf Überläufe ... 99
6.8.3 Beispiel-Code ... 100
6.9 Verschiedenes ... 102
6.9.1 Freie Variablen ... 102

7 Grundlegende Funktionen zur Scan-Kopf- und Lasersteuerung ... 103
7.1 Markieren von Punkten, Linien und Bögen ... 103
7.1.1 Markieren mit Vektor- und Bogenbefehlen ... 103
Jump-Befehle ... 104
Mark-Befehle ... 104
Arc-Befehle ... 104
Ellipse-Befehle ... 105
Para-Befehle ... 106
7.1.2 Mikrovektorisierung ... 107
7.1.3 Markieren von Punkten ... 108
7.1.4 Beispiel-Code ... 109
7.2 Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung ... 111
7.2.1 Laser-Delays ... 111
LaserOn-Delay ... 111
LaserOff-Delay ... 112
7.2.2 Scanner-Delays ... 113
Jump-Delay ... 113
Variable Jump-Delays ... 114
Mark-Delay ... 115
Polygon-Delay ... 117
Variables Polygon-Delay ... 118
Benutzerdefinierte variable Polygon-Delays ... 120
7.2.3 Hinweise zur Optimierung der Delays ... 121
Empfohlene Reihenfolge ... 121
Automatische Delay-Anpassung ... 121
Mögliche Fehler beim Optimieren der Delays ... 124
7.2.4 Sky-Writing ... 127
Modus 1 ... 127
Modus 2 ... 128
Modus 3 ... 128
Synchronisation ... 129
7.3 Scan-Kopf-Steuerung ... 134
7.3.1 Referenzsystem ... 134
7.3.2 Bildfeld-Größe und -Kalibrierung ... 134

Typisches Bildfeld ... 135
RTC4-Kompatibilitätsmodus ... 135
7.3.3 Virtuelles Bildfeld ... 135
7.3.4 Bildfeldkorrektur und Korrekturtabellen ... 136
Bildfeldverzerrung ... 136
Feldkorrektur-Algorithmus ... 137
Aktivieren der Bildfeldkorrektur ... 137
2D- und 3D-Korrekturdateien ... 137
Laden von Korrekturtabellen ... 137
Zuweisung von geladenen Korrekturtabellen ... 138
1to1-Korrekturtabellen ... 139
Inverse Tabellen ... 139
Datei-Kopf von Korrekturdateien ... 140
Konvertierung von Korrekturdateien ... 141
7.3.5 Ausgabewerte an das Scan-System ... 141
Berechnung ... 141
Wertebereich ... 142
Vorausberechnung und Diagnose ... 142
Takt-Überlauf ... 142
7.3.6 Statusüberwachung und Diagnose ... 143
Statusinformationen vom Scan-System ... 143
7.4 Lasersteuerung ... 144
7.4.1 Freigeben, Aktivieren und Schalten der Lasersteuersignale ... 144
“Laser active”-Signale ... 145
“Laser standby”-Signale ... 145
Automatische Lasersignalunterdrückung bei Status-Fehlern des Scan-Systems ... 146
Automatische Lasersignalunterdrückung bei Bereichsüberschreitung der Galvanometerscanner ... 146
7.4.2 Konfigurieren der Laseranschluss-Buchse (LASER) ... 146
7.4.3 CO_2-Version ... 147
7.4.4 YAG-Versionen ... 149
Q-Switch-Signal ... 149
FirstPulseKiller-Signal ... 150
Unterschiede zwischen den YAG-Versionen ... 150
Lampenstrom (Laserleistung) ... 150
7.4.5 Softstart-Modus ... 152
7.4.6 Lasermode 4 ... 154
7.4.7 Lasermode 6 ... 155
7.4.8 Pulse-Picking-Lasermodus ... 157
7.4.9 Automatische Lasersteuerung ... 159
Positionsabhängige Lasersteuerung ... 161
Geschwindigkeitsabhängige Lasersteuerung ... 165
Vektorkontrollierte Lasersteuerung ... 166
Encodergeschwindigkeitsabhängige Lasersteuerung ... 168
7.4.10 Ausgabe-Synchronisation ... 169
7.5 Markieren von Uhrzeit, Datum und Seriennummern ... 170
7.5.1 Markieren von Uhrzeit und Datum ... 170
7.5.2 Markieren von Seriennummern ... 170
8 Erweiterte Funktionen zur Scan-Kopf- und Lasersteuerung ... 172
8.1 i*DRIVE*-Funktionen ... 172
8.1.1 Übertragungsprotokoll ... 172
8.1.2 Anpassen des Rückmeldeverhaltens ... 173

Auswahl von Datentypen ... 173
Daten auslesen ... 173
8.1.3 Überwachung der Positionierung ... 174
8.1.4 Anpassen der dynamischen Einstellung (Tuning) ... 176
8.1.5 Sprungmodus ... 176
Funktionsweise ... 176
Voraussetzung und Aktivierung ... 177
Sprunglängenabhängige Jump-Delays ... 178
Experimentelle Ermittlung von Jump-Delay-Werten ... 178
Hinweise zum Laden der ermittelten Jump-Delay-Werte ... 179
Automatische Ermittlung der Jump-Delay-Tabelle ... 180
8.1.6 Anpassen des PosAcknowledge-Schwellwerts ... 181
8.1.7 Anpassen der effektiven Kalibrierung ... 181
8.1.8 Anpassen des Startverhaltens ... 182
8.1.9 Fehlerdiagnose und Funktionstest ... 182
8.2 Koordinatentransformationen ... 183
8.3 Online-Positionierung ... 187
Konfigurieren der Online-Positionierung ... 187
8.4 Wobbelmode ... 189
8.4.1 Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte ... 190
“Klassische” Wobbelfiguren ... 190
“Frei Definierbare Wobbelfiguren” ... 191
8.5 Verwendung mehrerer verschiedener Korrekturtabellen ... 192
8.5.1 Konfiguration mit zwei 2D-Scan-Systemen ... 192
8.5.2 Mehrere Korrekturtabellen mit einem Scan-System ... 192
8.6 Ansteuerung eines 3-Achsen-Scan-Systems (optional) ... 193
8.6.1 Verwendung ... 193
8.6.2 Anschluss und Initialisierung ... 193
8.6.3 3D-Markierbefehle ... 194
8.6.4 Justage ... 196
Überprüfen der Z-Achsen-Kalibrierung ... 196
Test für 3-Achsen-Scan-Systeme mit F-Theta-Objektiv ... 197
8.6.5 Erweiterte 3D-Korrektur ... 198
8.7 “Processing on the fly” (optional) ... 199
8.7.1 Verwendung und Initialisierung ... 199
Übersicht ... 200
8.7.2 Kompensation linearer Bewegungen ... 201
Korrektur über Encoder-Zähler ... 201
Korrektur über McBSP/SPI-Schnittstelle ... 202
Korrektur über McBSP/SPI-Schnittstelle mit zusätzlicher McBSP/SPI-Eingabe ... 203
Korrektur über McBSP/SPI-Schnittstelle mit erweiterter McBSP/SPI-Eingabe ... 203
8.7.3 Kompensation von Rotationsbewegungen ... 204
Korrektur über Encoder-Zähler ... 204
Korrektur über McBSP/SPI-Schnittstelle ... 205
Korrektur über McBSP/SPI-Schnittstelle mit zusätzlicher McBSP/SPI-Eingabe ... 205
8.7.4 Kompensation zweidimensionaler Bewegungen ... 206
2D-Encoder-Kompensation für XY-Tische ... 206
Koordinatentransformationen im virtuellen Bildfeld ... 208
8.7.5 Deaktivieren der “Processing on the fly”-Korrektur ... 209
8.7.6 Virtuelles Bildfeld ... 210
8.7.7 Synchronisation von “Processing on the fly”-Anwendungen ... 211
8.7.8 Encoder-Resets ... 213

8.7.9 Überwachung der “Processing on the fly”-Korrektur 213
Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527) 214
8.7.10 Schleppverzugskompensation von Encoderwerten für “Processing on the fly”-Anwendungen 215
8.7.11 “Processing on the fly”-Korrektur für die Z-Achse — “FlyZ-Korrektur” (ab Version DLL 530, OUT 531) 215
8.8 Pixelausgabemodus – Rasterbilder (Bitmaps) scannen 217
8.8.1 Funktionsprinzip 217
8.8.2 Softwarebefehle 217
8.8.3 Lasersteuerung 218
8.8.4 Synchronisation 219
8.9 Mikrovektor-Befehle 221
8.10 Zeitbemessene Vektor- und Bogenbefehle 223
8.11 Automatische Selbstkalibrierung 224
8.11.1 Verwendung zur Drift-Kompensation 224
8.11.2 Funktionsweise 224
8.11.3 Ermittlung von Referenzwerten 226
8.11.4 Kalibrierung während der Anwendung 226
Automatische Selbstkalibrierung 226
Kundeneigene Kalibrierung 227
Weitere Hinweise zur Kalibrierung 227
8.12 Camming 229
8.13 Zeitmessungen 231

9 Programmierung von Peripherie-Schnittstellen 232
9.1 Signal-Ausgabe 232
9.1.1 16 Bit-Digital-Ausgang 232
9.1.2 8 Bit-Digital-Ausgang 233
9.1.3 2 Bit-Digital-Ausgang 233
9.1.4 12-Bit-Analog-Ausgänge 233
9.1.5 Schrittmotoransteuerung 234
Ausgangssignale 234
Referenzfahrten und Positions-Initialisierung 235
Zielpositionsfahrten 235
Rücklesen von Signalen und Statuswerten 236
Anhalten von Endlosfahrten 236
9.1.6 RS232-Schnittstelle 237
9.1.7 McBSP/SPI-Schnittstelle 237
9.2 Signal-Eingabe 238
9.2.1 16 Bit-Digital-Eingang 238
9.2.2 2 Bit-Digital-Eingang 238
9.2.3 RS232-Schnittstelle 238
9.2.4 McBSP/SPI Schnittstelle 238
9.3 Steuerung mit externen Signalen 239
9.3.1 Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation 239
Externer Listen-Stopp 239
Externer Listen-Start 240
Externer Listen-Start mit Streckenverzögerung 241
Regelmäßige (periodische) externe Listen-Starts 242
9.3.2 Bedingte Befehlsausführung 244
Programmierbeispiele 245

9.3.3 Synchronisation durch Encoder-Signale ... 246
Verwendung ... 246
Eingänge für externe Encoder-Signale ... 247
Encoder-Simulation ... 247
9.3.4 Synchronisation und Online-Positionierung mit McBSP/SPI-Signalen ... 248
9.4 Periodische I/O-Signale ... 249

10 Befehle und Funktionen ... 250
10.1 Übersicht ... 250
10.1.1 Nomenklatur ... 250
Multi-Board-Befehle ... 250
Normale, kurze, variable und multiple Listenbefehle ... 250
Unverzögerte und verzögerte kurze Listenbefehle (ab Version OUT 515) ... 251
Multiple Listenbefehle ... 252
10.1.2 Kompatibilität ... 252
10.1.3 Versionsinformationen ... 253
10.1.4 Optionale Funktionen ... 253
10.1.5 Kontrollbefehle ... 254
10.1.6 Listenbefehle ... 257
10.1.7 Datentypen ... 260
10.2 RTC5 Befehlssatz ... 261
acquire_rtc ... 261
activate_fly_2d ... 262
activate_fly_xy ... 263
apply_mcbsp ... 264
apply_mcbsp_list ... 265
arc_abs ... 266
arc_abs_3d ... 267
arc_rel ... 268
arc_rel_3d ... 269
auto_cal ... 270
auto_change ... 273
auto_change_pos ... 274
bounce_supp ... 275
camming ... 276
clear_fly_overflow ... 278
clear_io_cond_list ... 279
config_laser_signals ... 280
config_laser_signals_list ... 280
config_list ... 281
control_command ... 283
copy_dst_src ... 293
disable_laser ... 294
enable_laser ... 294
execute_at_pointer ... 295
execute_list ... 295
execute_list_1 ... 295
execute_list_2 ... 295
execute_list_pos ... 296
fly_return ... 297
fly_return_z ... 298
free_rtc5_dll ... 299
get_auto_cal ... 299

get_char_pointer 300
get_config_list 300
get_counts 301
get_dll_version 301
get_encoder 301
get_error 302
get_fly_2d_offset 304
get_free_variable 304
get_galvo_controls 305
get_head_para 306
get_head_status 307
get_hex_version 308
get_hi_data 309
get_hi_pos 309
get_input_pointer 310
get_io_status 310
get_jump_table 311
get_lap_time 311
get_laser_pin_in 311
get_last_error 312
get_list_pointer 312
get_list_serial 313
get_list_space 313
get_marking_info 314
get_master_slave 316
get_mcbsp 317
get_mcbsp_list 317
get_out_pointer 318
get_overrun 318
get_rtc_mode 318
get_rtc_version 319
get_serial 319
get_serial_number 320
get_standby 320
get_startstop_info 321
get_status 323
get_stepper_status 324
get_sub_pointer 325
get_sync_status 326
get_table_para 327
get_text_table_pointer 328
get_time 328
get_transform 329
get_value 332
get_values 334
get_wait_status 335
get_waveform 336
get_z_distance 337
goto_xy 338
goto_xyz 339
home_position 340
home_position_xyz 341

if_cond 342
if_fly_x_overflow 342
if_fly_y_overflow 343
if_fly_z_overflow 343
if_not_activated 344
if_not_cond 344
if_not_fly_x_overflow 345
if_not_fly_y_overflow 345
if_not_fly_z_overflow 346
if_not_pin_cond 346
if_pin_cond 347
init_fly_2d 347
init_rtc5_dll 348
jump_abs 350
jump_abs_3d 351
jump_rel 352
jump_rel_3d 353
laser_on_list 354
laser_on_pulses_list 355
laser_signal_off 356
laser_signal_off_list 356
laser_signal_on 357
laser_signal_on_list 357
list_call 358
list_call_abs 359
list_call_abs_cond 360
list_call_abs_repeat 360
list_call_cond 361
list_call_repeat 362
list_continue 363
list_jump_cond 364
list_jump_pos 365
list_jump_pos_cond 366
list_jump_rel 367
list_jump_rel_cond 368
list_next 369
list_nop 369
list_repeat 370
list_return 371
list_until 372
load_auto_laser_control 373
load_char 374
load_correction_file 375
load_disk 379
load_fly_2d_table 381
load_jump_table 382
load_jump_table_offset 383
load_list 385
load_position_control 387
load_program_file 389
load_stretch_table 392
load_sub 393

load_text_table 394
load_varpolydelay 395
load_z_table 396
long_delay 397
mark_abs 398
mark_abs_3d 399
mark_char 400
mark_char_abs 401
mark_date 402
mark_date_abs 404
mark_ellipse_abs 405
mark_ellipse_rel 406
mark_rel 407
mark_rel_3d 408
mark_serial 409
mark_serial_abs 411
mark_text 412
mark_text_abs 413
mark_time 413
mark_time_abs 415
mcbsp_init 415
mcbsp_init_spi 416
measurement_status 417
micro_vector_abs 418
micro_vector_abs_3d 419
micro_vector_rel 420
micro_vector_rel_3d 421
move_to 422
para_jump_abs 422
para_jump_abs_3d 423
para_jump_rel 424
para_jump_rel_3d 425
para_laser_on_pulses_list 426
para_mark_abs 427
para_mark_abs_3d 428
para_mark_rel 429
para_mark_rel_3d 430
park_position 431
park_return 432
pause_list 434
periodic_toggle 434
periodic_toggle_list 435
quit_loop 436
range_checking 436
read_abc_from_file 438
read_analog_in 439
read_encoder 440
read_io_port 440
read_io_port_buffer 441
read_io_port_list 442
read_mcbsp 443
read_multi_mcbsp 444

read_status 445
read_user_data 447
release_rtc 448
release_wait 449
reset_error 449
restart_list 450
rs232_config 450
rs232_read_data 451
rs232_write_data 452
rs232_write_text 452
rs232_write_text_list 453
rtc5_count_cards 453
save_and_restart_timer 454
save_disk 455
select_char_set 456
select_cor_table 457
select_cor_table_list 459
select_rtc 460
select_serial_set 461
select_serial_set_list 461
send_user_data 461
set_angle 462
set_angle_list 463
set_auto_laser_control 464
set_auto_laser_params 467
set_auto_laser_params_list 467
set_char_pointer 468
set_char_table 469
set_control_mode 470
set_control_mode_list 472
set_default_pixel 472
set_default_pixel_list 472
set_defocus 473
set_defocus_list 474
set_delay_mode 475
set_delay_mode_list 476
set_dsp_mode 476
set_ellipse 477
set_encoder_speed 478
set_encoder_speed_ctrl 479
set_end_of_list 480
set_extstartpos 481
set_extstartpos_list 481
set_ext_start_delay 482
set_ext_start_delay_list 483
set_firstpulse_killer 483
set_firstpulse_killer_list 483
set_fly_2d 484
set_fly_limits 485
set_fly_limits_z 486
set_fly_rot 487
set_fly_rot_pos 488

set_fly_tracking_error 489
set_fly_x 490
set_fly_x_pos 491
set_fly_y 492
set_fly_y_pos 493
set_fly_z 494
set_free_variable 495
set_free_variable_list 495
set_hi 496
set_input_pointer 497
set_io_cond_list 497
set_jump_mode 498
set_jump_mode_list 501
set_jump_speed 502
set_jump_speed_ctrl 502
set_jump_table 503
set_laser_control 504
set_laser_delays 507
set_laser_mode 508
set_laser_off_default 508
set_laser_pin_out 509
set_laser_pin_out_list 509
set_laser_pulses 510
set_laser_pulses_ctrl 511
set_laser_timing 512
set_list_jump 513
set_mark_speed 514
set_mark_speed_ctrl 514
set_matrix 515
set_matrix_list 516
set_max_counts 517
set_mcbsp_freq 517
set_mcbsp_in 518
set_mcbsp_in_list 519
set_mcbsp_matrix 520
set_mcbsp_matrix_list 521
set_mcbsp_out 521
set_mcbsp_out_ptr 522
set_mcbsp_rot 523
set_mcbsp_rot_list 523
set_mcbsp_x 524
set_mcbsp_x_list 524
set_mcbsp_y 525
set_mcbsp_y_list 525
set_multi_mcbsp_in 526
set_multi_mcbsp_in_list 528
set_n_pixel 529
set_offset 530
set_offset_list 530
set_offset_xyz 531
set_offset_xyz_list 532
set_pause_list_cond 533

set_pixel 534
set_pixel_line 535
set_pixel_line_3d 536
set_port_default 537
set_pulse_picking 538
set_pulse_picking_length 538
set_pulse_picking_list 539
set_qswitch_delay 539
set_qswitch_delay_list 539
set_rot_center 540
set_rot_center_list 540
set_rtc4_mode 541
set_rtc5_mode 541
set_scale 542
set_scale_list 543
set_scanner_delays 544
set_serial 544
set_serial_step 545
set_serial_step_list 546
set_sky_writing 546
set_sky_writing_limit 547
set_sky_writing_limit_list 547
set_sky_writing_list 547
set_sky_writing_mode 548
set_sky_writing_mode_list 549
set_sky_writing_para 550
set_sky_writing_para_list 552
set_softstart_level 553
set_softstart_level_list 554
set_softstart_mode 555
set_softstart_mode_list 556
set_standby 557
set_standby_list 558
set_start_list 558
set_start_list_1 558
set_start_list_2 558
set_start_list_pos 559
set_sub_pointer 560
set_text_table_pointer 561
set_trigger 562
set_trigger4 568
set_vector_control 569
set_verify 571
set_wait 572
set_wobbel 573
set_wobbel_control 575
set_wobbel_direction 576
set_wobbel_mode 577
set_wobbel_offset 579
set_wobbel_vector 580
simulate_encoder 582
simulate_ext_start 583

simulate_ext_start_ctrl 584
simulate_ext_stop 584
start_loop 585
stepper_abs 586
stepper_abs_list 587
stepper_abs_no 587
stepper_abs_no_list 588
stepper_control 588
stepper_control_list 589
stepper_disable_switch 589
stepper_enable 590
stepper_enable_list 590
stepper_init 591
stepper_rel 593
stepper_rel_list 593
stepper_rel_no 594
stepper_rel_no_list 594
stepper_wait 595
stop_execution 596
stop_list 596
stop_trigger 597
store_encoder 597
sub_call 598
sub_call_abs 599
sub_call_abs_cond 599
sub_call_abs_repeat 600
sub_call_cond 600
sub_call_repeat 601
switch_ioport 602
sync_slaves 603
time_fix 604
time_fix_f 604
time_fix_f_off 605
time_update 606
timed_arc_abs 607
timed_arc_rel 608
timed_jump_abs 609
timed_jump_abs_3d 610
timed_jump_rel 611
timed_jump_rel_3d 612
timed_mark_abs 613
timed_mark_abs_3d 614
timed_mark_rel 615
timed_mark_rel_3d 616
timed_para_jump_abs 617
timed_para_jump_abs_3d 619
timed_para_jump_rel 620
timed_para_jump_rel_3d 621
timed_para_mark_abs 622
timed_para_mark_abs_3d 624
timed_para_mark_rel 625
timed_para_mark_rel_3d 626

transform ... 627
upload_transform ... 630
verify_checksum ... 631
wait_for_encoder ... 631
wait_for_encoder_in_range ... 632
wait_for_encoder_mode ... 633
wait_for_mcbsp ... 634
write_8bit_port ... 634
write_8bit_port_list ... 635
write_abc_to_file ... 635
write_da_1 ... 636
write_da_1_list ... 636
write_da_2 ... 636
write_da_2_list ... 637
write_da_x ... 637
write_da_x_list ... 638
write_hi_pos ... 639
write_io_port ... 640
write_io_port_list ... 640
write_io_port_mask ... 641
write_io_port_mask_list ... 641
10.3 Nicht mehr unterstützte RTC2/RTC3/RTC4-Befehle ... 642

11 Demo-Programme ... 647

12 Fehlersuche ... 648

13 Kundendienst ... 650
13.1 Wartung und Reparaturen ... 650
13.2 Gewährleistung ... 650
13.3 Kontaktmöglichkeiten zu SCANLAB ... 650
13.4 Entsorgung ... 650

14 Technische Spezifikationen der RTC5 PCI-Karte ... 651
14.1 Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV) ... 654
14.2 Konformität mit FCC-Bestimmungen ... 654

15 Anhang A: Die RTC5 PC/104-*Plus*-Karte ... 655
15.1 Produktübersicht ... 655
15.1.1 Verwendung – Vergleich mit der RTC5 PCI-Karte ... 655
15.1.2 Systemanforderungen ... 655
Hardware ... 655
Software ... 656
15.1.3 Optionale Funktionalitäten ... 656
15.1.4 Kennzeichnung ... 656
15.1.5 Typenbezeichnung ... 656
15.1.6 Auspackanleitung und typischer Lieferumfang ... 657
Mitgelieferte Software ... 657
15.1.7 Optionale Zusatzkomponenten ... 657
15.1.8 Ergänzende Software ... 657
15.2 Layout und Schnittstellen ... 658
15.2.1 Stecker- und Jumperpositionen ... 658
15.2.2 Schnittstelle zur CPU-Karte ... 660
15.2.3 Master/Slave-Synchronisation ... 660
15.2.4 Schnittstellen zum Scan-System ... 660

15.2.5 Schnittstellen zum Laser und zur Peripherie ... 661
LASER-Stecker ... 661
MULTI-Stecker ... 661
15.3 Installation und Inbetriebnahme ... 664
15.3.1 Sicherheit bei Installation und Inbetriebnahme ... 664
15.3.2 Jumpereinstellungen ... 665
Jumper 17+18: Binäre Festlegung der Stapeladresse ... 665
JP19: +5 V Spannungsversorgung ... 666
JP10+12: 10 V Ausgangsspannung für ANALOG OUT1 und ANALOG OUT2 ... 666
Jumper JP1: Auswahl des Ausgangssignal-Pegels am MULTI-Stecker ... 667
Jumper JP2-JP8: Konfiguration von Pin B40 und Pin B42 am MULTI-Stecker ... 667
15.3.3 Einbau der RTC5 PC/104-*Plus*-Karte ... 668
15.3.4 Installation der Treiber und RTC5-Software ... 670
15.3.5 Funktionstest, Anwenderprogramme und Demoprogramme ... 670
15.4 Technische Spezifikationen der RTC5 PC/104-*Plus*-Karte ... 671

16 Anhang B: Die RTC5 PCI-Express-Karte ... 672
16.1 Produktübersicht ... 672
16.1.1 Verwendung – Vergleich mit der RTC5 PCI-Karte ... 672
16.1.2 Systemanforderungen ... 672
Hardware ... 672
Software ... 672
16.1.3 Optionale Funktionalitäten ... 672
16.1.4 Kennzeichnung ... 673
16.1.5 Typenbezeichnung ... 673
16.1.6 Auspackanleitung und typischer Lieferumfang ... 673
Mitgelieferte Software ... 673
16.1.7 Optionale Zusatzkomponenten ... 673
16.1.8 Ergänzende Software ... 673
16.2 Layout und Schnittstellen ... 674
16.2.1 Stecker- und Jumperpositionen ... 674
16.2.2 Schnittstelle zum PC ... 675
16.2.3 McBSP/SPI-Schnittstelle und Analog-Eingänge ... 676
McBSP/SPI-Schnittstelle ... 676
Analog-Eingänge ... 676
16.3 Installation und Inbetriebnahme ... 676
16.4 Technische Spezifikationen der RTC5 PCI-Express-Karte ... 677
16.5 Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV) ... 677
16.6 Konformität mit FCC-Bestimmungen ... 677

17 Anhang C: Die RTC5 PCIe/104-Karte ... 678
17.1 Produktübersicht ... 678
17.1.1 Verwendung – Vergleich mit der RTC5 PCI-Karte ... 678
17.1.2 Vergleich mit der RTC5 PC/104-*Plus*-Karte ... 678
17.1.3 Systemanforderungen ... 679
Hardware ... 679
Software ... 679
17.1.4 Optionale Funktionalitäten ... 679
17.1.5 Optionaler Anschluss ... 679
17.1.6 Kennzeichnung ... 679
17.1.7 Typenbezeichnung ... 680
17.1.8 Auspackanleitung und typischer Lieferumfang ... 680
Mitgelieferte Software ... 680

17.1.9 Optionale Zusatzkomponenten 680
17.1.10Ergänzende Software 680
17.2 Layout und Schnittstellen 681
17.2.1 Stecker- und Jumperpositionen 681
17.2.2 Schnittstelle zur CPU-Karte 682
17.2.3 Master/Slave-Synchronisation 682
17.2.4 Schnittstellen zum Scan-System 683
17.2.5 Schnittstellen zum Laser und zur Peripherie 683
17.3 Installation und Inbetriebnahme 683
17.4 Technische Spezifikationen der RTC5 PCIe/104-Karte 684
18 Änderungsindex 685

1 Einleitung

Diese Anleitung beschreibt die SCANLAB RTC5-Karten und ihre Verwendung zur synchronen Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten.

Im Haupteil wird dazu beispielhaft auf die RTC5 PCI-Karte Bezug genommen. Zur Produktübersicht siehe Seite 24.

Im Anhang werden zusätzlich weitere SCANLAB RTC5-Hardware-Varianten beschrieben:

- die RTC5 PC/104-*Plus*-Karte in Anhang A, Seite 655,
- die RTC5 PCI-Express-Karte in Anhang B, Seite 672,
- die RTC5 PCIe/104-Karte in Anhang C, Seite 678.

Die Anleitung ist Bestandteil des Produkts. Lesen Sie diese Anweisungen sorgfältig durch, bevor Sie mit der Installation und Inbetriebnahme beginnen. Beachten Sie insbesondere alle Sicherheitshinweise dieser Anleitung. Bei Fragen zum Inhalt kontaktieren Sie SCANLAB (siehe Kapitel 1.1 "Hersteller").

Halten Sie die Anleitung für Wartung, Pflege, Reparatur und Entsorgung verfügbar. Bei einer Veräußerung der RTC5-Karte sollten Sie diese Anleitung weitergeben.

SCANLAB behält sich vor, diese Anleitung jederzeit und ohne Ankündigung inhaltlich zu aktualisieren.

Diese Bedienungsanleitung bezieht sich auf die folgenden Versionen der RTC5-Software und -Firmware:

- DLL-Datei (`RTC5DLL.dll`) Version 514 oder höher,
- DSP-Programmdatei (`RTC5OUT.out`) Version 513 oder höher,
- RTC5-Firmware-Datei (`RTC5RBF.rbf`) Version 511 oder höher.

Die Versionsnummern der ausgelieferten DLL- und Programmdateien sind in den Dateinamen der entsprechenden `zip`-Dateien enthalten (siehe Seite 23).

Nach Installation können Sie die Versionsnummern Ihrer aktuellen Software und Firmware mit den Befehlen **get_dll_version**, **get_hex_version** und **get_rtc_version** abrufen.

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 Kennzeichnung

Die Seriennummer der RTC5 PCI-Karte finden Sie auf einem weißem Aufkleber auf der Karte (Beschriftung "RTC SN…").

Die Artikelnummer und die Konfiguration der Karte werden auf dem jeweiligen Lieferschein aufgeführt (siehe Seite 27 und Seite 28).

1.3 Auspackanleitung und typischer Lieferumfang

- Nehmen Sie die RTC5-Karte vorsichtig aus der Verpackung.
- Bewahren Sie sowohl die Umverpackung als auch den antistatischen Beutel, in dem die RTC5 ausgeliefert wird, auf, damit die Karte im Fall einer Reparatur wieder optimal verpackt an SCANLAB geschickt werden kann.
- 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 RTC5-Karte und einer Daten-CD (mit der zugehörigen Software (s. u.) und diesem Manual). Unter Umständen sind aber auch Zusatzkomponenten wie Datenkabel oder Konverter enthalten, siehe Kapitel 2.5 "Zubehör", Seite 29.

1.3.1 Mitgelieferte Software

Das mitgelieferte RTC5 Software-Paket enthält Treiber für die 32-Bit und 64-Bit-Versionen der Betriebssysteme Microsoft Windows 10, 8, 7, Vista, XP SP2, XP SP3.

Die Daten-CD enthält alle Dateien in ungezippter Version. Das komplette Software-Paket wird aber zur Kennzeichnung der Programmversion (und damit auch zur einfacheren Verwaltung unterschiedlicher Programmversionen, siehe auch Abschnitt "Änderungshistorie", Seite 23) zusätzlich in gezippter Version mitausgeliefert:

- `RTC5_Software_<Datum>.zip`

Alle Dateien des Software-Pakets sind im Folgenden aufgelistet.

Inhalts- und Installationsbeschreibung

- `Readme.txt`
  Beschreibung in englischer Sprache
- `Liesmich.txt`
  Beschreibung in deutscher Sprache

Korrekturdateien

- `cor_1to1.CT5`
  Korrekturdatei
- i.d.R. weitere Korrekturdateien (`D2_XXX.CT5`, `D3_YYY.CT5`) und Readme-Dateien

CorrectionFileConverter-Programm

- Das Win32-basierte Programm `CorrectionFileConverter.exe` ermöglicht das Konvertieren von RTC4-Korrekturdateien in RTC5-Korrekturdateien und umgekehrt. Das Paket enthält auch die zugehörigen Handbücher in deutscher und englischer Sprache.

Demo-Dateien

- Quellcodes in C von Demo1…Demo7 (`Demo1.cpp`…`Demo7.cpp`)
- Demo1 als ausführbare Datei für 32-Bit sowie 64-Bit kompiliert
- Quellcode in C# von Demo3 (`Class1.cs`)
- Projektgenerierungsdatei für CMake (`CMakeLists.txt`) sowie die zugehörige Include-Datei (`RTC_Variables.cmake`)(1)

HPGL Converter-Programm

- Win32-basiertes HPGL Demo-Programm. Benötigt die Programm-Dateien und die 32-Bit-DLL im gleichen Verzeichnis.

iSCANConfig-Programm

- Das Programm `iSCANcfg5.exe` ist ein Win32-basiertes Diagnose- und Konfigurationsprogramm für i*DRIVE*-Scan-Systeme (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario$SCAN_{de}$). Benötigt die Programm-Dateien und die 32-Bit-DLL im gleichen Verzeichnis. Die zugehörigen Handbücher in Deutsch und Englisch sind enthalten.

(1) Das sind CMake-Dateien (cross-platform make) zur einfachen Erstellung ausführbarer Demo-Programme. CMake (https://cmake.org) ist ein freies und open-source plattformunabhängiges Programmierwerkzeug für die Entwicklung und Erstellung von Software. Mit CMake werden aus Skriptdateien (`CMakeLists.txt`) Makefiles und Projekte für viele integrierte Entwicklungsumgebungen und Compiler erzeugt.



Windows-Treiber

- `RTC5DRV.sys`, `RTC5DRVx64.sys`, `RTC5DRVx86.sys`
  RTC5-Treiber-Dateien
- `RTC5DRV.inf`
  Installationsdatei (Setup-Information)
- `RTC5DRV.cat`, `rtc5drvx64.cat`, `rtc5drvx86.cat`
  Sicherheitskatalog-Dateien
- `amd64/WdfCoInstaller01009.dll`, `x86/WdfCoInstaller01009.dll`
  Installations-Assistenten-Hilfsdateien
- `AfterInstallation/ScanlabClassChecker.cmd`,
  Security-Installations-Script mit Hinweisen in `ReadMe_ScanlabClassChecker.pdf`

RTC5-Dateien

- DLL-Dateien:
  - `RTC5DLL.dll`, `RTC5DLLx64.dll`
    Win32- bzw. Win64-basierte RTC5 dynamische Link Library
- Programm-Dateien:
  - `RTC5OUT.out`
    DSP-Programmdatei
  - `RTC5RBF.rbf`
    Firmwaredatei
  - `RTC5DAT.dat`
    binäre Hilfsdatei
- Utility-Dateien für C, C++ und C#:
  - `RTC5DLL.lib`, `RTC5DLLx64.lib`
    Visual C++ Import Libraries zum impliziten Linken der DLL für Win32- bzw. Win64-basierte Anwenderprogramme
  - `RTC5expl.c`
    C-Funktionen zum DLL-Handling für das explizite Linken
  - `RTC5expl.h`
    C-Funktionsprototypen der RTC5 zum expliziten Linken der DLL
  - `RTC5impl.h`
    C-Funktionsprototypen der RTC5 zum impliziten Linken der DLL
  - `RTC5impl.hpp`
    C++-Funktionsprototypen der RTC5 zum impliziten Linken der DLL
  - `RTC5Wrap.cs`
    Importdeklarationen der Wrapper-Klasse zum impliziten Linken der DLL unter C#
- Utility-Datei für Delphi:
  - `RTC5Import.pas`
    Importdeklarationen für Delphi

Die RTC5-Dateien werden zur Kennzeichnung der Programmversion (und damit auch zur einfacheren Verwaltung unterschiedlicher Programmversionen) zusätzlich in gezippter Version mitausgeliefert:

- `RTC5DAT_<Version>.zip`
- `RTC5DLL_<Version>.zip`
  (enthält DLL und zugehörige Utility-Dateien)
- `RTC5OUT_<Version>.zip`
- `RTC5RBF_<Version>.zip`

Die Programm-Dateien und die DLL können nicht in beliebigen Programmversionen miteinander kombiniert werden. Die `zip`-Dateien enthalten deshalb auch jeweils eine Text-Datei mit entsprechenden Versions- Informationen.

Änderungshistorie

Beschreibung von Änderungen in der RTC5-Software

- `RTC5_Software_RevisionHistory_<Datum>.pdf`
  Beschreibung in englischer Sprache
- `RTC5_Software_Aenderungshistorie_<Datum>.pdf`
  Beschreibung in deutscher Sprache

Hinweise

- Ältere RTC5-Softwarepakete enthalten auch MS Visual Studio Laufzeitbibliotheken für C und C++. Ab der Version DLL 527 werden diese nicht mehr benötigt.

2 Produktübersicht

2.1 Bestimmungsgemäße Verwendung

Die RTC5 PCI-Karte von SCANLAB ermöglicht zusammen mit der dazugehörigen Software die synchrone Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten in Echtzeit über einen Windows-PC mit PCI-Bus-Interface. Der mitgelieferte Treiber stellt einen umfassenden Satz an Steuerbefehlen zur Verfügung, die eine schnelle und flexible Programmierung von Laser-Scan-Prozessen ermöglichen.

Die RTC5 verfügt über ein digitales Signalprozessorsystem (DSP), das bei der Umsetzung der Steuerbefehle auch komplexere Signalbearbeitungsprozesse und Rechenschritte ausführt, etwa für die gleichzeitige Ansteuerung zweier Scan-Systeme oder für Koordinatentransformationen. Zusätzlich können Steuerbefehle auch auf der RTC5 gespeichert und ihre Ausführung dann zu einem späteren Zeitpunkt gestartet werden. Die Ausführung der Befehle durch die RTC5 kann dann 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.

Zur Ansteuerung von Lasern verfügt die RTC5 über geeignete Schnittstellen, an denen Lasersteuersignale ausgegeben werden, die über Steuerbefehle an die jeweiligen Anforderungen angepasst werden können. Der Benutzer kann u.a. zwischen sieben verschiedenen Lasersteuerungsversionen wählen und die Signalparameter, z.B. den Signal-Pegel (High-aktiv oder Low-aktiv) oder die Ausgabefrequenz geeignet einstellen. Mit der RTC5 können alle gebräuchlichen Lasertypen angesteuert werden.

Zur Ansteuerung von Peripheriegeräten und zur Einbindung externer Steuersignale verfügt die RTC5 über eine Reihe von Schnittstellen (u.a. einen 16-Bit-Digital-Eingang, einen 16-Bit-Digital-Ausgang, zwei 12-Bit-Analog-Ausgänge und ein RS232-Interface) und entsprechende Softwarebefehle.

Es können beliebig viele RTC5-Karten in einem Rechner gleichzeitig (bei Bedarf auch Master/Slave-synchronisiert) betrieben werden.

Außerdem erlaubt die DLL der RTC5 sowohl Multi-Threading als auch Multi-Processing. Es können also auch beliebig viele Anwenderprogramme gleichzeitig verwendet werden. Ein und dieselbe Karte kann allerdings nicht gleichzeitig von mehreren Anwenderprogramme verwendet werden. Mehrere Threads *eines* Anwenderprogramms können zwar ein und dieselbe Karte verwenden, aber nicht gleichzeitig Befehle senden.

Die RTC5 PCI-Karte kann in unterschiedlichen Konfigurationen ausgeliefert werden, siehe Kapitel 2.3 “Optionale Funktionalitäten”, Seite 27 und Kapitel 2.4 “Jumperkonfigurationen und Typenbezeichnung”, Seite 28.

Die Schnittstellen der RTC5-Karte werden ab Seite 40, die Installation und Inbetriebnahme von RTC5-Karten ab Seite 60 und die Programmierung ab Seite 65 beschrieben. Die einzelnen Befehlsbeschreibungen sind ab Seite 250 aufgelistet.

Eine Übersicht über die technischen Spezifikationen der RTC5 PCI-Karte finden Sie auf Seite 651.





Achtung!

- Die RTC5-Karte darf auf keinen Fall außerhalb eines PCs betrieben werden.
- Die RTC5-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 RTC5 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 RTC5-Karte ist eine Einrichtung der Klasse A. Sie kann im Wohnbereich Funkstörungen verursachen; in diesem Fall kann vom Betreiber verlangt werden, angemessene Maßnahmen durchzuführen.

2.2 Systemanforderungen

2.2.1 Hardware

Die RTC5 PCI-Karte kann in jeden Windows-PC mit PCI-Bus und mindestens einem freien PCI-Steckplatz installiert werden.
Die Abmessungen sind in Abb. 2 dargestellt.

RTC5 PCI-Karten, die Master/Slave-synchronisiert werden sollen, müssen in benachbarte PCI-Steckplätze gesteckt werden.

2.2.2 Software

Im Lieferumfang zur RTC5-Karte sind Treiber und DLL-Dateien für Microsoft Windows-Betriebssysteme enthalten. Die RTC5-Treiber unterstützen die Plug-and-Play-Fähigkeit der RTC5 und den simultanen Betrieb von beliebig vielen RTC5-Karten. Die RTC5-DLL-Dateien enthalten die Funktionen, die mit RTC5-Befehlen aufgerufen werden können.

- RTC5-Treiber und RTC5-DLL-Dateien aus den RTC5-Software-Paketen Version 2013_02_21 und später (mit Version ≥ DLL 535) sind sowohl für die 32-Bit-Versionen als auch die 64-Bit-Versionen von Microsoft Windows 10, 8, 7, Vista, XP SP2 und XP SP3 vorgesehen.
  RTC5-DLL-Dateien aus diesen Paketen unterstützen Microsoft Windows 2000 und XP ≤ SP1 nicht mehr.
- RTC5-Treiber und RTC5-DLL-Dateien aus den RTC5-Software-Paketen Version 2011_09_29 bis 2012_09_05 (mit Version DLL 528…DLL 535) sind sowohl für die 32-Bit-Version als auch die 64-Bit-Version von Microsoft Windows 7, Vista, XP SP2 und XP SP3 vorgesehen.
  RTC5-DLL-Dateien aus diesen Paketen unterstützen Microsoft Windows 2000 und XP ≤ SP1 nicht mehr.
- RTC5-Treiber und RTC5-DLL-Dateien aus den RTC5-Software-Paketen Version 2011_07_27 und früher (mit Version ≤ DLL 527; diese haben einen geringeren Funktionsumfang) sind für die 32-Bit-Versionen von Microsoft Windows 7, Vista, XP und 2000 vorgesehen.

Vorsicht!

- Die RTC5 PCI-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Busses abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren (siehe auch Hinweis unten).

Hinweise

- RTC5-Software-Pakete ab Version 2013_02_21 enthalten die (neuere) WDF(1)-Treiber Version 6.1.7600.16385.
  Frühere Pakete enthalten (ältere) WDM(2)-Treiber der Versionen 1.0.4.0 oder 2.0.6.0.
- RTC5-DLL-Dateien Version DLL ≤ 533sind *nicht* mit den WDF-Treibern kompatibel.
- RTC5-DLL-Dateien Version DLL ≥ 535 sind auch mit den WDM-Treibern kompatibel.
- Der neue WDF-Treiber bietet gegenüber einem alten WDM-Treiber folgende neue Funktionalität: Wenn **init_rtc5_dll** aufgerufen wird, unterbindet der Treiber eine automatische Aktivierung eines Standby- oder Sleep-Modus (dauerhaft, bis wieder ein System-Neustart erfolgt). Dadurch können die RTC5-Karten weiter autonom Listen abarbeiten, auch wenn die initiierende Software bereits beendet ist.
  Manuell oder durch entladene Akkus erzwungene Standby- oder Sleep-Modi können jedoch nicht verhindert werden. Danach sind geladene Listen und sonstige Einstellungen der RTC5-Karte verloren. Nach einem Wake-Up ist die Karte evtl. nicht mehr korrekt ansprechbar. Benutzer müssen Standby- oder Sleep-Modi vor dem erstmaligen Aufruf von **init_rtc5_dll** selbst verhindern.

(1) Windows Driver Framework
(2) Windows Driver Model

2.3 Optionale Funktionalitäten

Die RTC5 kann mit einigen Funktionalitäten ausgestattet werden, die in der Standard-Version nicht enthalten sind. Die Funktionalitäten müssen von SCANLAB freigeschaltet bzw. installiert werden. Die folgenden Optionen sind erhältlich:

- **"Processing on the fly"**
  Wenn die "Processing on the fly"-Option freigeschaltet ist, kann eine "Processing on the fly"-Korrektur aktiviert werden (siehe Seite 199). Die entsprechenden Befehle sind ansonsten nicht oder nur eingeschränkt nutzbar.
- **Steuerung eines 3-Achsen-Scan-Systems**
  Wenn die 3D-Option der RTC5 freigeschaltet ist, kann über die beiden Scan-Kopf-Anschlüsse der RTC5-Karte synchron zu den X- und Y-Achsen eines Scan-Systems (in der Regel zwei Galvanometerscanner) auch eine dritte Achse (Z-Achse, z. B. ein vario*SCAN* als dynamische Fokussiereinheit) angesteuert werden (siehe Seite 193). Die entsprechenden 3D-Befehle sind ansonsten nicht oder nur eingeschränkt nutzbar.
- **Synchrone Ansteuerung zweier Scan-Systeme**
  Wenn die Option "Zweite-Scan-Kopf-Steuerung" der RTC5 freigeschaltet ist, können über die beiden Scan-Kopf-Anschlüsse der RTC5-Karte zwei XY-Scan-Systeme simultan angesteuert werden (siehe auch Seite 42 und Seite 192).
- **Optokoppler**
  RTC5-Karten mit dieser Option werden werksseitig zusätzlich mit DC/DC-Wandlern ausgestattet. Die Lasersteuersignale LASERON, LASER1 und LASER2 an der LASER-Buchse und an der EXTENSION 2-Stiftleiste sind deshalb galvanisch von der PC-Masse entkoppelt (siehe Seite 47 und Seite 52).

Hinweise

- Aus der Artikelnummer einer RTC5 ergibt sich, welche Optionen freigeschaltet bzw. installiert sind. Freigeschaltete bzw. installierte Optionen werden auf dem Lieferschein aufgezählt und dort wie folgt benannt:
  – "Fly" ("Processing on the fly"-Option)
  – "3D" (3D-Option)
  – "SSHC" (Option "Zweite-Scan-Kopf-Steuerung")
  – "DCDC" (Optokoppler)
- Ob die "Processing on the fly"-Option, die 3D-Option und/oder die Option "Zweite-Scan-Kopf-Steuerung" freigeschaltet sind, kann auch mit dem Befehl **get_rtc_version** abgefragt werden.
- Falls Sie eine oder mehrere der Optionen "Processing on the fly", "3D" und/oder "Zweite Scan-Kopf-Steuerung" benötigen, diese aber auf Ihrer Karte nicht freigeschaltet sind, so können die benötigten Optionen auch nachträglich mit einer speziellen Upgrade-Software freigeschaltet werden. Diese Upgrade-Software ist bei SCANLAB erhältlich. Zur Erstellung der Software wird die Seriennummer Ihrer RTC5-Karte benötigt.
  Für eine nachträgliche Installation von Optokopplern muss die RTC5-Karte dagegen an SCANLAB eingeschickt werden.
- Zur optischen Datenübertragung zwischen RTC5 und Scan-System (über eine Kunststoff-Faser) wird ein entsprechendes Datenkabel, aber weder auf Seite der RTC5 noch auf Seite des Scan-Systems ein spezielles (optisches) Dateninterface benötigt (siehe Seite 45).

2.4 Jumperkonfigurationen und Typenbezeichnung

SCANLAB liefert die RTC5 in unterschiedlichen Jumperkonfigurationen aus. Die Jumper **JP1-JP8** sind Lötverbindungen. Benutzer können sie mit einem Lötkolben auch nachträglich umkonfigurieren.

Vorsicht!

- Achten Sie bei einer Änderung der Jumpereinstellung darauf, die Elektronik auf der Karte nicht zu beschädigen!

2.4.1 Jumper JP1: Auswahl des Ausgangssignal-Pegels an der EXTENSION 1-Stiftleiste

Mit dem Jumper **JP1** auf der Rückseite der RTC5, siehe Abb. 3, kann für alle Ausgangssignale der EXTENSION 1-Stiftleiste zwischen einem Signal-Pegel von 5 V oder 3,3 V ausgewählt werden.
Siehe auch "Auswahl des Ausgangssignal-Pegels", Seite 51.

Jumper JP1		Signal-Pegel
Position 1-2		5 V
Position 2-3		3,3 V
offen		keine Ausgangssignale

Vorsicht!

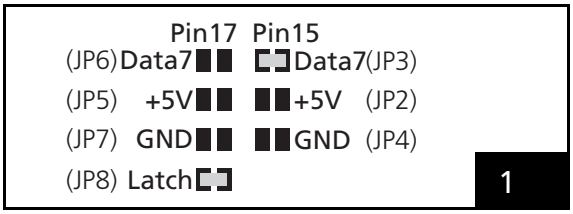
- Achten Sie darauf, dass nicht alle drei Pins des Jumpers verbunden werden. Andernfalls wird die Elektronik der RTC5-Karte beschädigt.

2.4.2 Jumper JP2–JP8: Konfiguration von Pin15 und Pin17 an der EXTENSION 2-Stiftleiste

Über die Jumper **JP2-JP8** auf der Rückseite der RTC5, siehe Abb. 3, können Pin (15) und (17) der EXTENSION 2-Stiftleiste konfiguriert werden.
Siehe auch "Jumper-Einstellung", Seite 52.

Das höchstwertige Bit (DATA7) des 8-Bit-Ausgangswertes kann Pin (15) oder Pin (17) zugewiesen werden. Alternativ können beide Pins auch permanent auf +5 V (HIGH) oder GND (LOW-Pegel) gelegt werden. Pin (17) kann alternativ auch das LATCH-Signal zugeordnet werden. Die Zuweisung erfolgt durch Schließen des entsprechend beschrifteten Jumpers.

Abb. 1 zeigt als Beispiel die Jumper-Einstellung, mit der DATA7 Pin (15) und das LATCH-Signal Pin (17) zugeordnet wird.



Beispielkonfiguration für Jumper JP2-JP8:
Pin (15): DATA7, Pin (17): LATCH-Signal

Vorsicht!

- Achten Sie darauf, dass nicht mehr als *einer* der drei Jumper für Pin (15) geschlossen ist und auch nicht mehr als *einer* der vier Jumper für Pin (17). Andernfalls wird die Elektronik der RTC5-Karte beschädigt.

2.4.3 Typenbezeichnung

Aus der Artikelnummer einer RTC5 ergibt sich, in welcher Jumperkonfiguration die Karte ausgeliefert wird. Die jeweilige Jumperkonfiguration wird zusätzlich durch eine dreiziffrige Typenbezeichung dargestellt:

Ziffer 1	=0: Jumper JP1 offen (kein Signal) =1: Jumper JP1 in Position 1-2 (5 V) =2: Jumper JP1 in Position 2-3 (3,3 V)
Ziffer 2 (betr. Pin 15)	=0: Jumper JP2-4 offen (kein Signal) =1: Jumper JP2 geschlossen (+5 V) =2: Jumper JP3 geschlossen (DATA7) =3: Jumper JP4 geschlossen (GND)
Ziffer 3 (betr. Pin 17)	=0: Jumper JP5-8 offen (kein Signal) =1: Jumper JP5 geschlossen (+5 V) =2: Jumper JP6 geschlossen (DATA7) =3: Jumper JP7 geschlossen (GND) =4: Jumper JP8 geschlossen (LATCH)

Beispiele

- “Typ 000”: alle Jumper offen
- “Typ 124”: JP1 in Position 1-2 (5 V Signal-Pegel), JP3 geschlossen (DATA7 an Pin 15), JP8 geschlossen (LATCH an Pin 17)

2.5 Zubehör

Zusätzlich zur RTC5 PCI-Karte und zum zugehörigen Software-Paket ist bei SCANLAB das folgende Zubehör erhältlich (in Kombination mit der RTC5 sollten nur Hardwareerweiterungen von SCANLAB verwendet werden):

2.5.1 XY2-100-Konverter

Mit dem XY2-100-Konverter von SCANLAB können mit der RTC5-Karte auch Scan-Systeme angesteuert werden, die mit einem herkömmlichen XY2-100-Interface ausgestattet sind (siehe Seite 43).

2.5.2 Datenkabel

Zur Verbindung der RTC5-Karte mit Scan-Systemen sind bei SCANLAB geeignete Datenkabel verschiedener Länge – wahlweise Datenkabel zur elektrischen Datenübertragung oder Lichtwellenleiterkabel zur optischen Datenübertragung – erhältlich (siehe Seite 45).

2.5.3 Laseradapter

Wenn der optionale Laseradapter von SCANLAB auf die 15-polige LASER-Buchse der RTC5 aufgesteckt wird, dann stehen über eine 9-polige Sub-D-Buchse die gleichen Signale (mit identischer Pinbelegung) wie an der 9-poligen LASER-Buchse der RTC4 zur Verfügung (siehe Seite 50).

2.5.4 PCI-Slotbleche

Für den Anschluss eines zweiten Scan-Kopfs oder einer Z-Achse an den sekundären Scan-Kopf-Anschluss (10-polige Stiftleiste 2. SCAN HEAD, siehe Seite 43) ist bei SCANLAB ein Slotblech mit einer 9-poligen Sub-D-Buchse erhältlich (siehe Seite 43).

Zur Verwendung der Eingänge und Signale an der MARKING ON THE FLY-Stiftleiste (16-polige Stiftleiste, siehe Seite 53) ist bei SCANLAB ein Slotblech mit einer 15-poligen Sub-D-Buchse erhältlich (siehe Seite 53).

2.5.5 ADC-Aufsteckplatine

Für die RTC5 PCI-Karte – *nicht* dagegen für RTC5 PCI-Express-Karte, RTC5 PC/104-*Plus*-Karte und RTC5 PCIe/104-Karte – ist bei SCANLAB eine ADC-Aufsteckplatine (Artikelnummer 121126) erhältlich.

Damit verfügt die RTC5 PCI-Karte dann über zwei 10 V-Analog-Eingänge (siehe Seite 59).

Die RTC5 PCI-Express-Karte verfügt auch ohne Aufsteckplatine über zwei 10 V-Analog-Eingänge (siehe Seite 676).

2.5.6 RTC5 varioSCAN FLEX Extension-Karte

Für die RTC5-Karten ist bei SCANLAB die Erweiterungskarte RTC5 vario*SCAN* 40 FLEX Extension (Artikelnummer 128683) erhältlich(1).

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 vario*SCAN FLEX* Extension for the RTC5 PC Interface Board”.

Hinweise

- Die Erweiterungskarte RTC5 vario*SCAN* 40 FLEX Extension (Artikelnummer 128683) unterstützt *nicht* den Befehl **move_to**(2).

2.6 Ergänzende Software

Zur individuellen Anpassung von RTC-Korrekturdateien entsprechend der Daten einer Testmessung ist bei SCANLAB Software mit zugehörigem Handbuch aus der correXion-Reihe erhältlich (siehe auch Seite 137).

Mit der SCANLAB laserDESK-Software können ohne eigene Softwareentwicklung Laserbeschriftungs- und Lasermaterialbearbeitungsprogramme erstellt und ausgeführt werden. Fast jede RTC5-Funktionalität wird über die grafische Benutzeroberfläche bereitgestellt. SCANLAB laserDESK integriert das Bildauswertesystem *SCAN*align, so dass kamerabasierte Positionskorrekturen und Qualitätskontrolle direkt in den Laserjobs verwaltet und angewendet werden können.

(1) Im Gegensatz zur RTC Step Motor Extension-Karte (Artikelnummer 112097) – die auch mit der RTC5-Karte verwendet werden kann – bietet sie daneben den Vorteil, dass die EXTENSION 1-Stiftleiste frei bleibt.

(2) Der Befehl **move_to** wurde für die RTC Step Motor Extension-Karte (Artikelnummer 112097) eingeführt.

2.7 Hinweise an Benutzer der RTC4-Karte

Dieses Kapitel gibt eine Übersicht über die wichtigsten Änderungen, die bei der RTC5 PCI-Karte im Vergleich zur RTC4 PCI-Karte eingeführt wurden.

Unter anderem wird in Kapitel 2.7.2 "Überarbeiten von RTC4-Programmcodes für die RTC5", Seite 32 auch eine mögliche Vorgehensweise zum Überarbeiten von RTC4-Programmcodes für die RTC5 beschrieben.

Auch in den weiteren Kapiteln dieses Handbuchs werden an verschiedenen Stellen Hinweise an RTC5-Benutzer, die vorher eine RTC4 benutzt haben, gegeben. Insbesondere bei den einzelnen Befehlsbeschreibungen wird auf relevante Änderungen hingewiesen.

2.7.1 Hardware-Änderungen

Beim Wechsel von der RTC4 zur RTC5 müssen die folgenden Hardware-Änderungen für eine korrekte Verkabelung mit den Systemkomponenten berücksichtigt werden.

Steuerung von Scan-Systemen

- Zur Ansteuerung von Scan-Systemen mit XY2-100- oder XY2-100 Enhanced-Schnittstelle wird zusätzlich ein XY2-100-Konverter benötigt (bei SCANLAB erhältlich – siehe Seite 43).
- Scan-Systeme mit XY2-100-O-Schnittstelle (zur optischen Datenübertragung) können nicht mit der RTC5 angesteuert werden.
- Zur Ansteuerung von Scan-Systemen mit SL2-100-Schnittstelle wird ein anderes Datenkabel benötigt (bei SCANLAB erhältlich – siehe Seite 45).
- Zur Verwendung des sekundären Scan-Kopf-Anschlusses wird ein anderes Adapterkabel benötigt (bei SCANLAB mit Slotblech erhältlich – siehe Seite 43).
- Zur Ansteuerung eines 3-Achsen-Scan-Systems müssen beide Scan-Kopf-Anschlüsse verwendet werden (und die 3D-Korrekturdatei korrekt zugewiesen werden) (siehe Seite 193).
- Für die simultane Ansteuerung zweier 3-Achsen-Scan-Systeme werden zwei RTC5-Karten benötigt (siehe Seite 193).

Lasersteuerung

- Die Sub-D-LASER-Buchse am RTC5 Slotblech ist 15-polig (an der RTC4 dagegen 9-polig). Deren Pinbelegung kann nicht über Jumper konfiguriert werden.
  - Die Jumper X6 und X7 der RTC4 werden auf der RTC5 nicht benötigt, weil ohnehin alle Signale an der RTC5-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 RTC5 nicht mehr bereitgestellt).
- Soll der Laser über das gleiche (9-polige) Kabel mit der RTC5 verbunden werden wie zuvor mit der RTC4, so wird ein Adapter mit 9-poliger Sub-D-Buchse benötigt (bei SCANLAB erhältlich).
  - Beim Laseradapter von SCANLAB (siehe Seite 50) werden zwei Jumper zu Konfiguration der Pinbelegung bereitgestellt (JP1 entspricht Jumper X7 der RTC4, JP2 entspricht Jumper X6 der RTC4).
- Der Signalpegel der Lasersteuersignale wird nicht mehr über eine Jumper-Konfiguration festgelegt, sondern kann/muss über einen Software-Befehl festgelegt werden (siehe **set_laser_control**).
  - Jumper X10 der RTC4 wird auf der RTC5 nicht mehr bereitgestellt.

EXTENSION 1-Stiftleiste

- Die EXTENSION 1-Stiftleiste der RTC5 ist – bis auf die zusätzlich bereitgestellten Signale an den Pins 33-35 – identisch zur EXTENSION 1-Stiftleiste der RTC4 (siehe Seite 51).
- Bei der RTC5 kann (anders als bei der RTC4) über einen Jumper für alle Ausgangssignale der EXTENSION 1-Stiftleiste zwischen einem Signal-Pegel von 5 V oder 3,3 V ausgewählt werden (siehe Seite 51).

EXTENSION 2-Stiftleiste

- Die EXTENSION 2-Stiftleiste der RTC4 wird auf der RTC5 nicht mehr bereitgestellt. Entsprechend kann auch keine I/O-Erweiterungskarte mehr an die RTC5 angeschlossen werden.
- Die EXTENSION 2-Stiftleiste der RTC5 ist – bis auf das optionale LATCH-Signal an Pin 17 – identisch zur LASER EXTENSION-Stiftleiste der RTC4 (siehe Seite 52).
  - Zur Konfiguration der Pinbelegung werden auf der RTC5 die Jumper **JP2-JP8** bereitgestellt (diese entsprechen den Jumper X8 und X9 auf der RTC4) (siehe auch Seite 52).

MARKING ON THE FLY-Stiftleiste

- Die MARKING ON THE FLY-Stiftleiste der RTC5 ist identisch zur MARKING ON THE FLY-Stiftleiste der RTC4 (siehe Seite 53).

Sonstige Schnittstellen

- Die Anforderungen an die PCI-Schnittstelle sind gegenüber der RTC4 unverändert.
- PCI-Express-, PC/104-*Plus* und PCIe/104-Versionen sind nur für die RTC5 erhältlich; eine Standalone- und Ethernet-Version derzeit nur für die RTC4.
- Die folgenden Schnittstellen werden nur auf der RTC5 bereitgestellt:
  - MASTER und SLAVE (siehe Seite 41)
  - RS232 (siehe Seite 54)
  - SPI / I²C / McBSP (siehe Seite 54)
  - STEPPER MOTOR (siehe Seite 58)

2.7.2 Überarbeiten von RTC4-Programmcodes für die RTC5

Anwenderprogramme, die für die RTC4 geschrieben wurden, können für die RTC5 nur dann verwendet werden, wenn der Programmcode geeignet überarbeitet wird. Dies gilt auch dann, wenn der eigentliche Ablauf des Programms unverändert bleiben und keine der neuen Funktionalitäten der RTC5 in Anspruch genommen werden soll.

Veränderte Initialisierung

Der Initialisierungsteil des Programmcodes muss mindestens wie folgt überarbeitet werden:

- Zu Beginn des Programms muss ein **init_rtc5_dll**-Befehl zur Initialisierung der DLL und Kartenverwaltung eingefügt werden (siehe Seite 68).
- Die Dateien zur Initialisierung der Karte mit **load_program_file** müssen anders bereitgestellt und aufgerufen werden 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 Seite 136).
- Zur Initialisierung der Lasersteuerung muss unbedingt zusätzlich der Befehl **set_laser_control** eingefügt werden (siehe Seite 144), selbst wenn nur Standby-Signale ausgegeben werden sollen.

Befehlsänderungen

Alle nicht mehr unterstützten RTC4-Befehle müssen aus dem Programmcode entfernt bzw. ersetzt werden (siehe hierzu auch Kapitel 10.3 ab Seite 642).

Geänderte und erweiterte RTC4-Befehle müssen gegebenenfalls anders in das Programm eingebunden werden (z.B. durch Anpassung von übergebenen Parameterwerten oder durch veränderte Auswertung rückgegebener Werte). Auf relevante Änderungen bei den weiterhin unterstützten Befehlen wird bei den einzelnen Befehlsbeschreibungen (in Kapitel 10.2) jeweils in der Zeile "RTC4→ RTC5" hingewiesen.

Es folgt eine Liste der RTC4-Befehle, die ersetzt bzw. überprüft werden müssen:

- **aut_change** nicht mehr unterstützt
- **auto_cal** geändert
- **auto_change_pos** geändert
- **control_command** geändert
- **dsp_start** nicht mehr 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 mehr unterstützt
- **get_xyz_pos** nicht mehr unterstützt
- **goto_xy** geändert
- **goto_xyz** geändert
- **list_jump_cond** geändert (abhängig von der Speicherkonfiguration)
- **list_nop** geändert
- **load_cor** nicht mehr unterstützt
- **load_correction_file** geändert
- **load_pro** nicht mehr unterstützt
- **load_program_file** geändert
- **read_pixel_ad** nicht mehr unterstützt
- **read_status** geändert
- **rtc3_count_cards / rtc4_count_cards** nicht mehr unterstützt
- **select_cor_table** geändert
- **select_list** nicht mehr unterstützt
- **select_rtc** geändert
- **set_control_mode** erweitert
- **set_control_mode_list** erweitert
- **set_laser_mode** erweitert
- **set_laser_timing** Firmware geändert
- **set_list_mode** nicht mehr unterstützt
- **set_matrix** geändert
- **set_matrix_list** geändert
- **set_offset** geändert
- **set_offset_list** geändert
- **set_piso_control** nicht mehr unterstützt
- **set_pixel** geändert
- **set_pixel_line** geändert
- **set_softstart_mode** geändert
- **set_trigger** erweitert
- **set_wobbel** geändert
- **set_wobbel_xy** nicht mehr unterstützt
- **timed_jump_abs** geändert
- **timed_jump_rel** geändert
- **timed_mark_abs** geändert
- **timed_mark_rel** geändert
- **z_out** nicht mehr unterstützt
- **z_out_list** nicht mehr unterstützt

Erhöhte Parameter-Auflösung

Zusätzlich muss beim Wechsel von der RTC4 zur RTC5 – 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- und y-Koordinatenwerte für das reale Bildfeld bei der RTC5 mit einer Auflösung von 20 Bit (gegenüber 16 Bit bei der RTC4) angegeben. Für eine bestimmte Ausgabeposition (in mm) müssen also 16-mal höhere Parameterwerte angegeben werden (siehe Kapitel 7.3.2, Seite 134). Insbesondere für "Processing on the fly"-Anwendungen steht bei der RTC5 darüber hinaus ein erweiterter, virtueller Wertebereich – insgesamt also ein virtuelles 24-Bit-Bildfeld – zur Verfügung (siehe Kapitel 7.3.3, Seite 135).
- Bei Befehlen wie **write_da_x** werden analoge Ausgabewerte bei der RTC5 mit einer Auflösung von 12 Bit (gegenüber 10 Bit bei der RTC4) angegeben. Für einen bestimmten Ausgabewert (in V) muss also ein 4-mal höherer Parameterwert angegeben werden.
- Beim Befehl **set_laser_timing** werden Ausgabeperiode und Pulslänge bei der RTC5 mit einer Auflösung von 1/64 µs (gegenüber 1/8 µs bzw. 1 µs bei der RTC4) angegeben. Für einen bestimmten Zeitwert (in µs) muss also ein 8-mal bzw. 64-mal höherer Parameterwert angegeben werden (zur Taktfrequenz siehe Befehlsbeschreibung).
- Beim Befehl **set_laser_delays** werden die Laser-Delays bei der RTC5 mit einer Auflösung von 0,5 µs (gegenüber 1 µs bei der RTC4) angegeben. Für einen bestimmten Delaywert (in µs) muss also ein 2-mal höherer Parameterwert angegeben werden.

Hier reicht es jedoch in der Regel, den Betriebsmodus der RTC5-DLL mit **set_rtc4_mode** in den RTC4-Kompatibilitätsmodus zu schalten. Dann führt die RTC5-DLL eine entsprechende Umrechnung der Para-

meterwerte automatisch durch, so dass viele Befehlsfolgen aus RTC4-Programmcodes (etwa zur Definition von Beschriftungsmustern) unverändert für die RTC5 übernommen werden können.

Der RTC4-Kompatibilitätsmodus wirkt auf die folgenden RTC4-Befehle (bei allen Befehlen wird gegebenenfalls in der Zeile "RTC4→ RTC5" 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 bzw. geänderte RTC4-Befehle muss allerdings unabhängig davon durchgeführt werden, ob das Programm im RTC5-Modus oder im RTC4-Kompatibilitätsmodus ausgeführt werden soll.

Verändertes Zeitverhalten

Die im Folgenden aufgelisteten RTC4-Listenbefehle werden von der RTC5 als sogenannte "kurze Listenbefehle" abgearbeitet. Hier kann sich bei der Programmausführung ein verändertes Zeitverhalten ergeben (siehe Seite 250).

- **clear_io_cond_list**
- **list_call**
- **list_call_cond**
- **list_jump_cond** (geändert, abhängig von der Speicherkonfiguration)
- **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** (Firmware 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 Seite 121).

2.7.3 Neue und geänderte Funktionalitäten

Schnittstelle zum PC

- Die Treiber der RTC5 unterstützen den simultanen Betrieb von beliebig vielen RTC5-Karten in einem PC (siehe Seite 92).
- Stecker und Befehle zur Master/Slave-Synchronisierung mehrerer RTC5-Karten (siehe Seite 41)

Steuerung von Scan-Systemen

- Neue Schnittstelle zum Scan-System (siehe Seite 42):
  - 9-polige Sub-D-Buchse am RTC5 Slotblech und 10-polige Stiftleiste on-board
  - SL2-100-Übertragungsprotokoll
  - 2 Datenkanäle je Scan-Kopf-Anschluss
  - Galvanische Potentialtrennung
  - 20 Bit Positionsauflösung (siehe Seite 134)
  - Erweiterte Statusrückgabe vom Scan-System (siehe Seite 143)
  - Kontroll- und Statuskanäle zum effektiveren Datenaustausch mit i*DRIVE*-Scan-Systemen (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario$SCAN_{de}$)
- Zur Datenübertragung gemäß XY2-100-Protokoll steht der XY2-100-Konverter zur Verfügung (siehe Seite 43). Die Versorgungsspannung für den Konverter wird von der RTC5 zur Verfügung gestellt.
  - 25-polige Sub-D-Buchse
  - 16 Bit Positionsauflösung
  - Statusrückgabe gemäß XY2-100- bzw. XY2-100 Enhanced-Protokoll
  - Synchronisationsanpassung für längere Datenkabel über Lötjumper im XY2-100-Konverter
- Zur optischen Datenübertragung zwischen RTC5 und Scan-System wird keine spezielle Variante der RTC5 (mit XY2-100-O Interface) benötigt. Die optische Datenübertragung kann über ein Datenkabel von SCANLAB mit Lichtwellenleiter-Umsetzung im Sub-D-Stecker-Gehäuse realisiert werden (siehe Seite 45).
- Ansteuerung eines 3-Achsen-Scan-Systems (siehe Seite 193):
  - nur durch Verwendung beider Scan-Kopf-Anschlüsse und durch korrekte Zuweisung einer 3D-Korrekturdatei
  - Für die simultane Ansteuerung zweier 3-Achsen-Scan-Systeme werden zwei RTC5-Karten benötigt.
- Bildfeldkorrektur
  - Es werden neue Korrekturdateien benötigt (siehe Seite 136):
    > Dateinamenerweiterung ".ct5"
    > Korrektur mit höherer Auflösung
    > Auslesbare Informationen zur Korrekturdatei im Datei-Kopf
    > RTC2/3/4-Korrekturdateien (.ctb) müssen für die RTC5 neu berechnet oder mit einem mitgelieferten Programm konvertiert werden.
  - Es können bis zu vier Korrekturdateien auf die RTC5 geladen werden.
  - Erweiterte 3D-Bildfeldkorrektur mit Stretch-Korrektur-Tabellen (siehe Seite 198)
- Koordinatentransformationen (siehe Seite 183):
  - keine Transformationen (Rotation, Verschiebung, Streckung) der Korrekturdatei beim Download mehr
  - Matrixtransformationen werden erst nach der Mikrovektorisierung angewendet, d.h. die Markiergeschwindigkeit kann sich durch die Transformation ändern.
  - 24-Bit-Vektorkoordinaten: Objekte größer als Bildfeld möglich (virtuelles Bildfeld, siehe Seite 135)
  - Ansammeln von Transformationsdefinitionen und Ausführen bei einem späteren Befehl
  - Online-Positionierung (siehe Seite 187)
- Überwachung der Positionierung bei i*DRIVE*-Scan-Systemen durch Rücktransformation von Ist-Positions-Werten (siehe Seite 174)
- Automatische Selbstkalibrierung (siehe Seite 224):
  - Optimierung der bisherigen Funktionen
  - ASC-Hardware-Überprüfung
- Sprungmodus (siehe Seite 176)
- Ausgabe-Synchronisation (siehe Seite 169).

Lasersteuerung

- Der Signalpegel der Lasersteuersignale wird nicht mehr über eine Jumper-Konfiguration festgelegt, sondern kann/muss über einen Software-Befehl festgelegt werden (siehe **set_laser_control**).
- 15-polige Sub-D-LASER-Buchse mit allen Lasersignalen am RTC5 Slotblech (siehe Seite 47), 9-polige Sub-D-Buchse nur über optionalen Laseradapter (siehe Seite 50)
- 15-polige Sub-D-LASER-Buchse per Software-Befehl konfigurierbar (siehe Seite 146).
- Lasersteuersignale mit 15 ns zeitlicher Auflösung und 20 mA Ausgangsstrom
- Standby-Signale auch in YAG-Versionen (siehe Seite 149)
- YAG-Version 5: Zeit zwischen FirstPulseKiller-Signal und erstem Laserpuls in der YAG-Version frei programmierbar (siehe Seite 149)
- Lasermode 6: LASERON-Signal mit freilaufendem LASER1-Signal synchronisiert (siehe Seite 155)
- Pulse-Picking-Lasermodus (siehe Seite 157)
- Laser-Pulsperiode, -Pulslänge oder ein Analogausgang können auch innerhalb eines Polygonzugs zwischen zwei Vektoren programmiert werden (siehe “kurze Listenbefehle” Seite 250).
- Befehle zur positionsabhängigen, geschwindigkeitsabhängigen, vektorkontrollierten und encodergeschwindigkeitsabhängigen Lasersteuerung (siehe Seite 159)

Schnittstellen zur Peripherie

- 16-Bit-Digital-Ausgang (siehe Seite 51 und Seite 232):
  - Ausgangssignal-Pegel über einen Jumper einstellbar (3,3 V oder 5 V)
  - LATCH-Signal zur Synchronisation der Datenübergabe
- 8 Bit-Digital-Ausgang (siehe Seite 52 und Seite 233):
  - 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 Seite 48 und Seite 233):
  - 12 Bit Auflösung
  - 0…10 V (0 V … 2,50 V nicht mehr möglich)
  - “Stopp-Ausgabewert” einstellbar
- 16-Bit-Digital-Eingang (siehe Seite 51 und Seite 238).
  - SYNC-Signal zur Synchronisation der Datenübergabe
- Programmierbare Entprellung externer Startsignale (siehe **bounce_supp** und Seite 240).
- Regelmäßige (periodische) externe Listen-Starts (siehe Seite 242)
- Neue Schnittstellen:
  - Ein 2-Bit-Digital-Eingang und ein 2-Bit-Digital-Ausgang auf der LASER-Buchse (siehe Seite 48)
  - RS232-Schnittstelle auf 10-poliger Stiftleiste (siehe Seite 54)
  - Schrittmotorsignale für 2 Motoren auf 10-poliger Stiftleiste (siehe Seite 58)
  - McBSP, I^2C und SPI (Serielle Peripherie-Schnittstelle) auf 10-poliger Stiftleiste (siehe Seite 54)
- Für die RTC5 gibt es keine IO-Erweiterungskarte mehr. Die Stiftleiste zum Anschließen einer Erweiterungskarte fehlt entsprechend auf der RTC5 (auf der RTC4 heißt diese Stiftleiste “EXTENSION 2”; die EXTENSION 2-Stiftleiste auf der RTC5 entspricht der “LASER EXTENSION” Stiftleiste auf der RTC4).

Allgemeine Programmierung

- Utility-Dateien für C, C++, C# und Delphi (siehe Seite 66). Keine Utility-Dateien mehr für Basic.
- Befehle zur Änderung der Zugriffsberechtigungen für RTC5-Karten (siehe Seite 96)
- Verbessertes und erweitertes Listen-Handling (siehe Seite 75)
  - Listenspeicher mit 1.048.576 Speicherplätzen
  - Listenspeicher frei konfigurierbar (in 2 Listenbereiche und einen geschützten Speicherbereich)
  - Definition von geschützten Unterprogrammen
  - Laden von Listen mit Schutzfunktion
  - Schleifen in Listen und Unterprogrammen
  - kein Ringpuffer-Modus mehr (kann aber alternativ realisiert werden)
  - Listenstatus und Listenausführungsstatus
- "Kurze" Listenbefehle (z. B. zur Änderung von Geschwindigkeit, Analogausgang, IO-Port etc.) können ohne Zeitverlust ausgeführt werden (es können mehrere "kurze" Listenbefehle innerhalb eines 10 µs-Takts ausgeführt werden, siehe Seite 250).
- Funktionen zur Fehlerbehandlung und Download-Überprüfung (siehe Seite 98)

Laser-Markieren

- Vektoren und Bögen
  - Befehle zur Markierung von Ellipsen (siehe Seite 105)
  - Befehle zur Markierung von Spiralen (siehe Seite 194)
  - Zeitbemessene Bogen- und 3D-Vektorbefehle (siehe Seite 223)
  - Para-Mark- und Para-Jump-Befehle zur vektorkontrollierten Lasersteuerung (siehe Seite 166)
- Zeichen und Texte
  - Definition von Zeichensätzen und Zeichenfolgen (siehe Seite 87)
  - Befehle zum Markieren einzelner Zeichen und von Texten (mit auswählbarem Zeichensatz, siehe Seite 88)
  - Funktionen zur Markierung von Uhrzeit, Datum und Seriennummern (mit auswählbarem Zeichensatz und mit auswählbarem Seriennummernsatz, siehe Seite 170)
- Mikrovektor-Befehle (direkte Ausführung von Vektoren ohne Mikrovektorisierung)

Spezielle Funktionen

- Synchronisation der Steuerung von Scan-System und Laser
  - Automatische Delay-Anpassung (siehe Seite 121)
  - Sky-Writing (siehe Seite 127)
- Pixelausgabemodus (Scannen von Bitmaps, siehe Seite 217)
  - Pixelfrequenzen bis 300 kHz, unabhängig vom 10 µs-Takt
  - 15 ns zeitliche Auflösung
  - 0–100% Laser-Pulslänge
  - kein Pixel-Mode 0 mehr
  - kein Einscannen von Analogspannungen mehr
  - Pixelbeschriftung auf schrägen Flächen
- Befehle zur Konditionierung beliebiger Listenbefehle (siehe Seite 244)
- 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. “Frei definierbare Wobbelfiguren” sind (im Gegensatz zu den “klassischen” Wobbelfiguren) zusätzlich mit der Möglichkeit einer Laserleistungsvariation ausgestattet (siehe Seite 189)
- Camming-Funktionalität (siehe **camming**)
- Erweiterte Möglichkeiten zur Signalaufzeichnung (siehe **set_trigger** und **set_trigger4**)
- “Processing on the fly” (siehe Seite 199)
  - zwei Encodereingänge (RS422) mit 32 Bit-Zähler für eine “Processing on the fly”-Korrektur mit Encoder-Signalen auf 2 Achsen (siehe Seite 246); alternativ: Korrektur mit McBSP-Signalen (siehe Seite 248)
  - 24-Bit-Vektorkoordinaten: Objekte größer als Bildfeld möglich (virtuelles Bildfeld, siehe Seite 135 und Seite 210)
  - bis zu 8 Objekte innerhalb der “Processing on the fly”-Streckenverzögerung (siehe “Externer Listen-Start”, Seite 240)
  - Exakter “externer Listen-Start”:
    Der Encoder-Zähler kann – sofern 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-Tisch)
  - Encodergesteuerte “Processing on the fly”-Korrektur für die Z-Achse (“FlyZ-Korrektur”, siehe Seite 215)

3 Sicherheit bei Installation und Inbetriebnahme

Lesen Sie diese Betriebsanweisungen vollständig, bevor Sie mit der Installation und Inbetriebnahme der RTC5 PCI-Karte beginnen.

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

Im Handbuch werden die folgenden Symbole verwendet, um Vorsichtsmaßnahmen im Text herauszuheben:



Hinweise, die auf eine gesundheitliche Gefahr aufmerksam machen, beginnen mit dem Signalwort **"Achtung!"** und sind durch ein Warndreieck gekennzeichnet.



Hinweise, die eine sachgemäße Verwendung des Gerätes empfehlen oder vor möglichen Beschädigungen warnen, beginnen mit dem Signalwort **"Vorsicht!"** und sind durch einen Kreis mit einem Kreuz gekennzeichnet.

3.1 Betriebssicherheit



Vorsicht!

- 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.
- Vermeiden Sie bei Lagerung und Betrieb der RTC5 elektromagnetische Felder und statische Elektrizität. Die Elektronik der Karte kann durch solche Felder zerstört werden. Lagern Sie die Karte stets im mitgelieferten antistatischen Beutel.



Vorsicht!

- 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 RTC5 wird zur Steuerung 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.



Achtung!

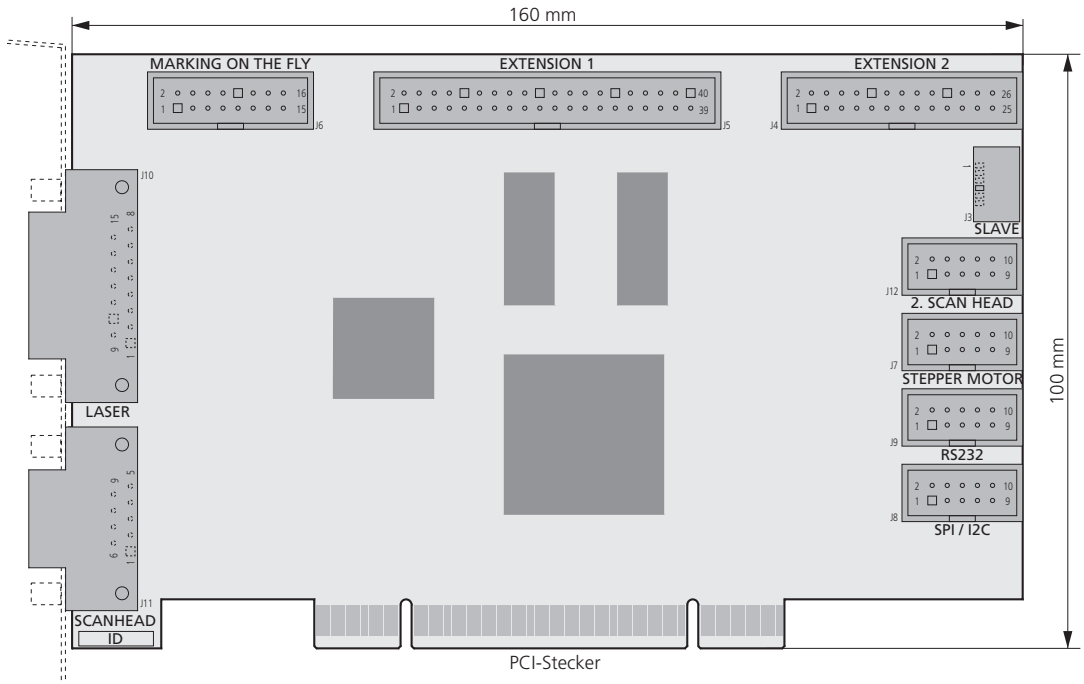
- Sicherheitsvorschriften können von Land zu Land variieren. Die entsprechenden Vorschriften müssen bekannt sein und angewandt werden. Es liegt in der Verantwortung des Kunden, die lokalen Vorschriften zu erfüllen.
- Beachten Sie unbedingt die Hinweise bezüglich der Lasersicherheit, die in der Bedienungsanleitung zum Scan-System im Kapitel "Sicherheit bei Installation und Inbetriebnahme" aufgeführt sind.
- **Schalten Sie immer zuerst den PC und erst dann den Laser an. Andernfalls 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 Layout und Schnittstellen

4.1 Stecker- und Jumperpositionen

Abb. 2 und Abb. 3 zeigen die Positionen der Anschlüsse und Jumper auf der Vorderseite und Rückseite der RTC5 PCI-Karte.

Alle Anschlüsse und Jumpereinstellungen werden in den folgenden Abschnitten beschrieben.

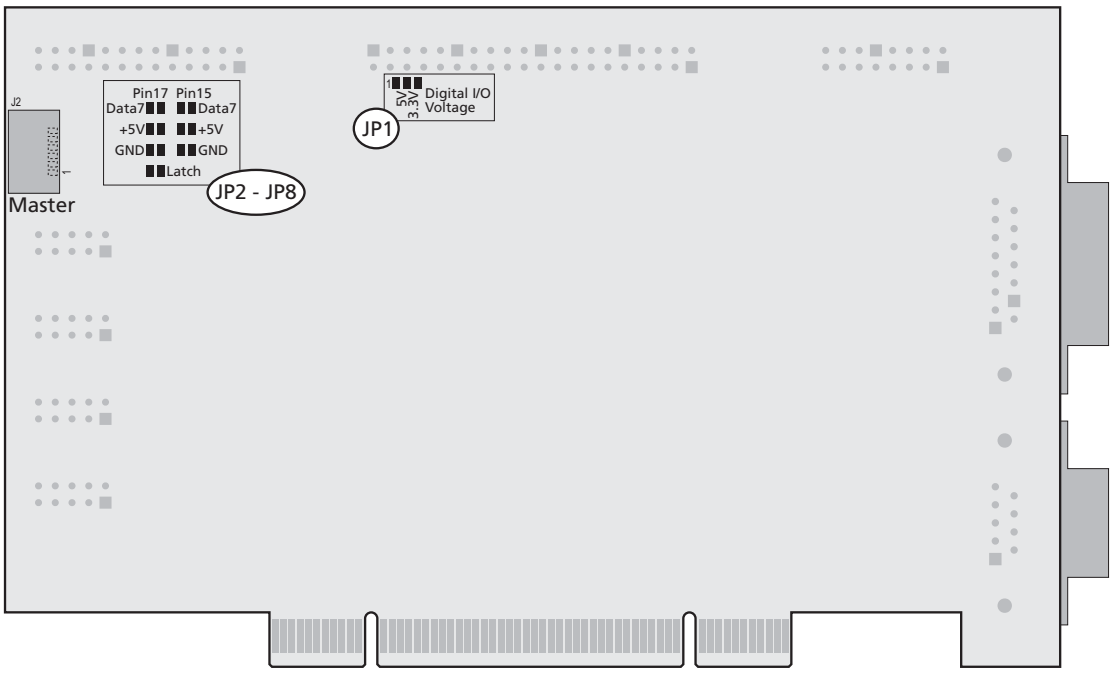


Legende

SCAN HEAD . . . . . .Primärer Scan-Kopf-Anschluss (9-polige Sub-D-Buchse, Pinbelegung Seite 42)
LASER . . . . . .Laseranschluss, sowie digitale und analoge Ausgänge (15-polige Sub-D-Buchse, Pinbelegung Seite 47)
MARKING ON THE FLY . . . . .16-polige Stiftleiste mit Encoder-Eingängen, (optional) für "Processing on the fly"-Anwendungen (Pinbelegung Seite 53)
EXTENSION 1 . . . . . .40-polige Stiftleiste mit einem 16 Bit-Digital-Eingang und einem 16 Bit-Digital-Ausgang (Pinbelegung Seite 51)
EXTENSION 2 . . . . . .26-polige Stiftleiste mit einem 8-Bit-Digital-Ausgang, z. B. für eine alternative Laseransteuerung (Pinbelegung Seite 52)
SLAVE . . . . . .6-poliger SLAVE-Anschluss (siehe "Master/Slave-Synchronisation", Seite 41)
2. SCAN HEAD. . . . . .10-polige Stiftleiste für einen zweiten Scan-Kopf (optional aktiviert, Pinbelegung Seite 43)
STEPPER MOTOR . . . . . .10-polige Stiftleiste zur Ansteuerung zweier Schrittmotoren (Pinbelegung Seite 58)
RS232 . . . . . .10-polige Stiftleiste zur Ansteuerung eines externen Geräts per RS232-Schnittstelle (Pinbelegung Seite 54)
SPI / I2C. . . . . .10-polige Stiftleiste (serielle Erweiterungsschnittstelle, Pinbelegung Seite 54)
ID . . . . . .Artikelnummer
PCI-Stecker . . . . . .Schnittstelle zum PC (siehe Seite 41)

2

Layout der RTC5 PCI-Karte (Vorderseite)



Legende

Master . . . . . 6-poliger MASTER-Anschluss (siehe "Master/Slave-Synchronisation", Seite 41)
JP1 . . . . . Jumper zur Auswahl des Ausgangssignal-Pegels an der EXTENSION 1-Stiftleiste (siehe Seite 51)
JP2 - JP8 . . . . Jumper zur Konfiguration der EXTENSION 2-Stiftleiste (siehe Seite 52)

3

Layout der RTC5 PCI-Karte (Rückseite)

4.2 Schnittstelle zum PC

Schnittstelle zum PC ist der PCI-Stecker der RTC5.

Die RTC5 kann in jeden Windows-PC mit PCI-Bus und mindestens einem freien PCI-Steckplatz installiert werden.

Vorsicht!

- Die RTC5-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Siehe hierzu auch Hinweis auf Seite 26.

4.2.1 Master/Slave-Synchronisation

Sollen mehrere RTC5-Karten mit synchroner Taktung in einem PC betrieben werden, dann müssen die RTC5-Karten zuerst paarweise über die MASTER- und SLAVE-Anschlüsse miteinander verbunden und dann in benachbarte PCI-Steckplätze gesteckt werden. Verbinden Sie jeweils den MASTER-Anschluss einer Karte mit dem SLAVE-Anschluss einer anderen Karte. Geeignete Verbindungskabel sind bei SCANLAB erhältlich.

Siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93 und "Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation", Seite 239.

4.3 Schnittstellen zum Scan-System

4.3.1 Scan-Kopf-Anschlüsse und Übertragungsprotokoll

Zur digitalen Ansteuerung von Scan-Systemen stehen der primäre Scan-Kopf-Anschluss SCAN HEAD und der optional aktivierte sekundäre Scan-Kopf-Anschluss 2. SCAN HEAD zur Verfügung (siehe Abb. 2). 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 erfolgt in beide Richtungen gemäß dem SL2-100-Protokoll. Zur Konvertierung der Signale für eine Datenübertragung gemäß dem XY2-100-Protokoll ist bei SCANLAB ein entsprechender XY2-100-Konverter erhältlich (siehe "XY2-100-Konverter (optional)", Seite 43).

Wenn weder die Option "Zweite-Scan-Kopf-Steuerung" noch die 3D-Option freigeschaltet ist (diese Optionen müssen von SCANLAB freigeschaltet werden), werden nur am primären Scan-Kopf-Anschluss Signale für ein XY-Scan-System ausgegeben.

Ist die Option "Zweite-Scan-Kopf-Steuerung" (aber nicht die 3D-Option) freigeschaltet, werden am sekundären Scan-Kopf-Anschluss die gleichen Signaltypen ausgegeben wie am primären Scan-Kopf-Anschluss. Es können dann mit einer RTC5 zwei XY-Scan-Systeme simultan angesteuert werden.

Ist dagegen die 3D-Option (aber nicht die Option "Zweite-Scan-Kopf-Steuerung") freigeschaltet, dann kann über die beiden Scan-Kopf-Anschlüsse ein 3-Achsen-Scan-System angesteuert werden, wenn dem primären Scan-Kopf-Anschluss eine 3D-Korrekturtabelle zugewiesen wird. Am primären Scan-Kopf-Anschluss werden dann die Signale für den XY-Scan-Kopf und über beide Kanäle des sekundären Scan-Kopf-Anschlusses die Signale für die dritte Achse (Z-Achse) ausgegeben.

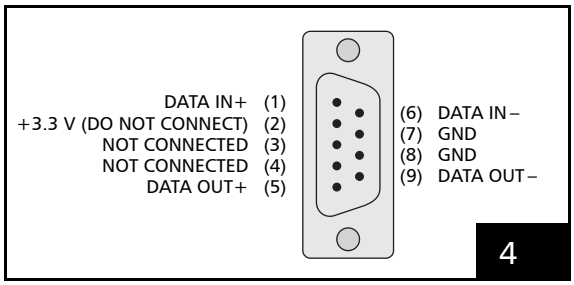
Sind *beide* Optionen ("Zweite-Scan-Kopf-Steuerung" *und* 3D) freigeschaltet, entscheidet der Benutzer bei der Zuweisung der Korrekturtabellen darüber, welche Signale (XY oder Z) an welchem Anschluss ausgegeben werden (siehe auch "2D- und 3D-Korrekturdateien", Seite 137).

Wenn mehrere RTC5-Karten mit freigeschalteter 3D-Option in einen PC eingebaut werden, dann können auch entsprechend viele 3-Achsen-Systeme gleichzeitig angesteuert werden.

Bei Verwendung der Master/Slave-Funktionalität können in einem PC auch mehrere RTC5-Karten mit synchroner Taktung betrieben werden, sofern auf sämtlichen Karten der Befehl **load_program_file** ausgeführt worden ist (siehe "Master/Slave-Synchronisation", Seite 41 und "Initialisierung der Karte", Seite 69).

Primärer Scan-Kopf-Anschluss

Abb. 4 zeigt die Pinbelegung des primären Scan-Kopf-Anschlusses SCAN HEAD (9-polige Sub-D-Buchse).

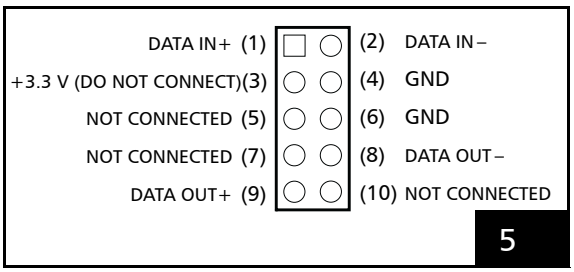


Pinbelegung der 9-poligen Sub-D-Buchse (SCAN HEAD) für den Scan-Kopf-Anschluss

Am differentiellen DATA OUT-Ausgang werden die Ansteuerwerte für das Scan-System ausgegeben und am differentiellen DATA IN-Eingang die Statussignale empfangen, die vom Scan-System zurückgegeben werden. An Pin 2 wird 3,3 V als Betriebsspannung für den optionalen XY2-100-Konverter oder für einen POF-Konverter zur optischen Datenübertragung (POF = Polymer Optical Fiber) von SCANLAB bereitgestellt. Für andere Zwecke sollte diese Spannung nicht verwendet werden.

Sekundärer Scan-Kopf-Anschluss (optional aktiviert)

Die Pinbelegung des sekundären Scan-Kopf-Anschlusses “2. SCAN HEAD” (10-polige Stiftleiste) ist in Abb. 5 dargestellt.



Pinbelegung der Stiftleiste “2. SCAN HEAD” (on-board)

Die Signale am sekundären Scan-Kopf-Anschluss sind optional und müssen durch SCANLAB zur Ansteuerung eines zweiten 2-Achsen-Scan-Systems (durch Freischaltung der Option “Zweite-Scan-Kopf-Steuerung”) und/oder zur Ansteuerung eines 3-Achsen-Scan-Systems (durch Freischaltung der 3D-Option) aktiviert werden (siehe oben und Kapitel 2.3 “Optionale Funktionalitäten”, Seite 27).

Nach der Aktivierung werden, abhängig von den zugewiesenen Korrekturtabellen (s. o.) die gleichen Signaltypen ausgegeben wie am primären Scan-Kopf-Anschluss.

Slotblech (optional)

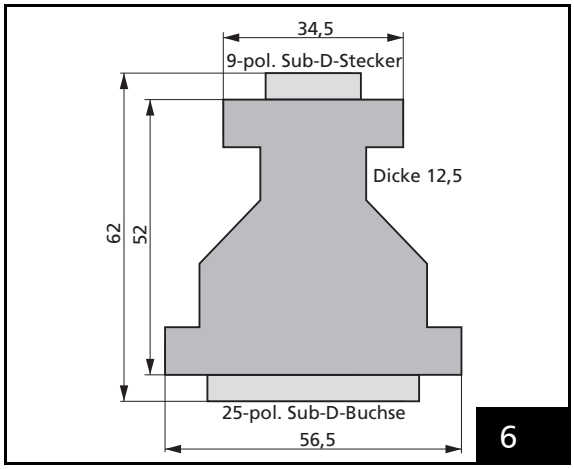
Für den Anschluss eines zweiten Scan-Kopfs oder einer Z-Achse empfiehlt SCANLAB den Gebrauch eines zusätzlichen Slotblechs. Ein geeignetes Slotblech mit 9-poligem Sub-D-Anschluss und einer Pinbelegung wie beim primären Scan-Kopf-Anschluss (siehe Abb. 4) ist bei SCANLAB erhältlich.

4.3.2 XY2-100-Konverter (optional)

Der optionale XY2-100-Konverter von SCANLAB konvertiert die RTC5-Ansteuersignale gemäß SL2-100-Protokoll (20 Bit) in Ansteuersignale gemäß XY2-100-Protokoll (16 Bit) und Scan-System-Statussignale gemäß XY2-100-Protokoll in Statussignale gemäß SL2-100-Protokoll (siehe Seite 143).

Der XY2-100-Konverter verursacht bei der Scan-System-Ansteuerung eine Laufzeitverzögerung von 10 µs. Diese Laufzeitverzögerung wird ausgeglichen, wenn der Benutzer das LaserOn-Delay und das LaserOff-Delay um jeweils 10 µs vergrößert (siehe **set_laser_delays**).

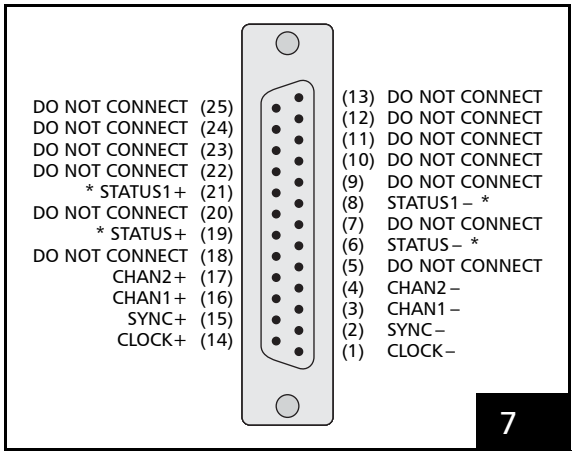
Abb. 6 zeigt die Abmessungen des XY2-100-Konverters.



Abmessungen des XY2-100-Konverters

Der 9-polige Sub-D-Stecker des XY2-100-Konverters sollte direkt in den primären Scan-Kopf-Anschluss der RTC5 eingesteckt bzw. – über ein möglichst kurzes 1:1-Kabel – mit den entsprechenden Pins des sekundären Scan-Kopf-Anschlusses der RTC5 verbunden werden (Pinbelegung siehe Abb. 4 und Abb. 5).

Die 25-polige Sub-D-Buchse des XY2-100-Konverters ist mit den Scan-Köpfen kompatibel, die mit einem XY2-100-Standardinterface ausgestattet sind. Die Pinbelegung ist in Abb. 7 dargestellt.

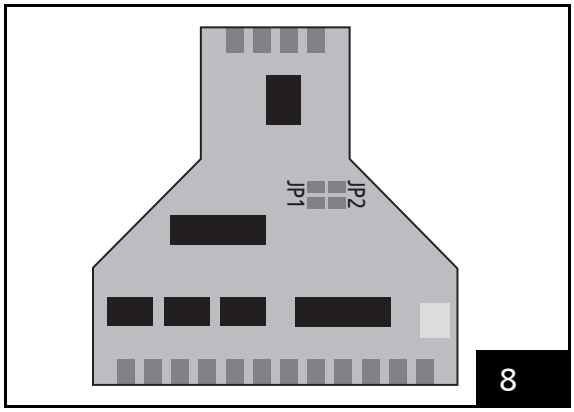


Pinbelegung der 25-poligen Sub-D-Buchse des XY2-100-Konverters

* Bei i*DRIVE*-Scan-System (intelli*SCAN*, intelli*SCAN*$_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario*SCAN*$_{de}$) 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 dem XY2-100-Standard zurückgegeben werden.

Falls zur Datenübertragung zwischen XY2-100-Konverter und Scan-System sehr lange Datenkabel verwendet werden, dann sollte die Synchronisation der bidirektionalen Kommunikation zwischen Scan-Kopf und RTC5 angepasst werden. Dies kann über zwei Lötjumper im XY2-100-Konverter bewerkstelligt werden. Dazu muss das Gehäuse des Konverters vorsichtig (z. B. mit Hilfe eines Schraubenziehers) an seinen vier Clip-Verriegelungen geöffnet werden. Die Lötjumper JP1 und JP2 befinden sich auf der Konverter-Platine zwischen den beiden Sub-D-Anschlüssen. Abb. 8 zeigt die Positionen der Jumper auf der Platine. In der Tabelle darunter sind die möglichen Jumpereinstellungen und die zugehörigen Kabellängen aufgelistet. Andere Jumpereinstellungen sowie Kabellängen über 25 m werden nicht empfohlen.



Position der Lötjumper JP1 und JP2 auf der Platine des XY2-100-Konverters

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

* Der empfohlene Kabellängenbereich für die jeweilige Jumperkonfiguration hängt u. a. vom verwendeten Kabeltyp ab und kann von den oben genannten Werten abweichen.

4.3.3 Datenkabel (Zubehör)

Zur Übertragung der Datensignale zwischen RTC5 bzw. XY2-100-Konverter und Scan-System sind bei SCANLAB auch geeignete Datenkabel erhältlich.

Zur Ansteuerung von SCANLAB Scan-Systemen, die mit einem SL2-100-Interface (9-polige Sub-D-Buchse) ausgestattet sind, sind wahlweise Datenkabel zur elektrischen Datenübertragung oder Lichtwellenleiterkabel zur optischen Datenübertragung erhältlich. Auch die Lichtwellenleiterkabel werden über 9-polige Sub-D-Stecker angeschlossen. Die optische Umsetzung (POF-Konvertierung) für die optische Datenübertragung erfolgt dabei in den Sub-D-Steckern. Die Betriebsspannung für die POF-Konvertierung wird an den RTC5-Scan-Kopf-Anschlüssen und am digitalen Interface des Scan-Systems bereitgestellt.

Für SCANLAB Scan-Systeme, die mit einem herkömmlichen XY2-100-Interface (25-polige Sub-D-Buchse) ausgestattet sind, sind nur Datenkabel zur elektrischen Datenübertragung erhältlich. Scan-Systeme, die mit einem optischen Dateninterface (XY2-100-O) ausgestattet sind, können nicht mit der RTC5 angesteuert werden.

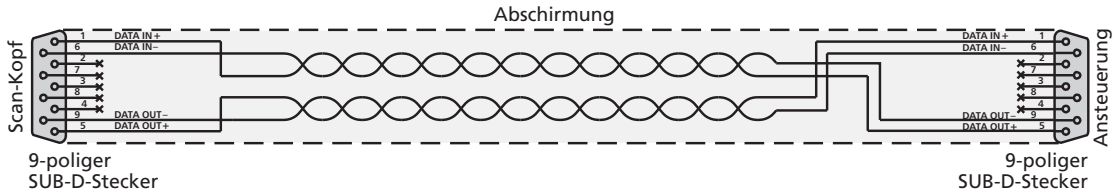
Datenkabel sind in der Regel nicht im Lieferumfang enthalten. SCANLAB empfiehlt die folgende Konfektionierung (für die elektrische Datenübertragung):

- Für die Datenübertragung gemäß SL2-100-Protokoll 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 RTC5 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 sollte das Kabel an beiden Enden mit 25-poligen Sub-D-Steckern mit identischer Pinbelegung ausgestattet sein. Die fünf (bzw. sechs) Kanäle SYNC±, CHAN1±, CHAN2±, STATUS± (und STATUS1±) und CLOCK± sollten aus verdrillten Kabelpaaren bestehen. Zusammen mit einem XY2-100-Konverter mit Standard-Jumper-Konfiguration (s. o.) sollte das Kabel nicht länger als 10 m sein. Falls ein längeres Datenkabel benötigt wird, dann muss u. U. die Lötjumper-Einstellung des XY2-100-Konverters angepasst werden (siehe "XY2-100-Konverter (optional)", Seite 43). Kabellängen über 25 m werden nicht empfohlen.
- Das Datenkabel muss über eine koaxiale Abschirmung verfügen, die als Kupfergeflecht ausgeführt ist.
- Die Sub-D-Stecker müssen über ein voll abgeschirmtes Metallgehäuse verfügen.
- Die Abschirmung des Kabels darf *nicht* als Drahtverbindung an das Gehäuse ausgeführt sein, sondern muss koaxial über geschirmte Griffschellen übertragen werden.
- Für die Datenübertragung gemäß XY2-100-Protokoll muss am ansteuerseitigen Kabelende des Datenkabels ein Ferrit des Typs "Würth WE 742 711 32" o. ä. angebracht werden.

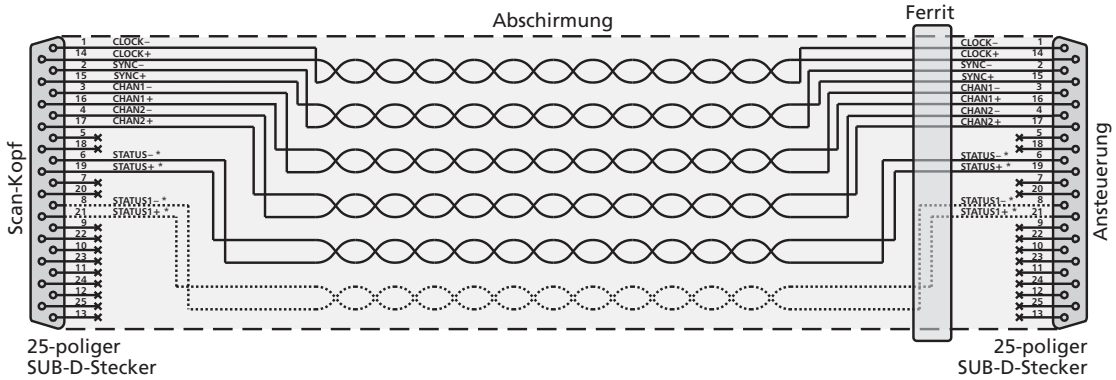
Das Datenkabel und die Pinbelegung zeigt Abb. 9.

Einige Scan-Köpfe verwenden nur einen einzelnen Anschluss für die Bereitstellung der Betriebsspannung und der Datensignale. Für diese Scan-Köpfe empfiehlt SCANLAB eine Kabelausführung zu implementieren, bei der die Daten in einem separaten Kabel übertragen werden, das wie bereits beschrieben konfektioniert sein sollte.

Datenübertragung gemäß SL2-100-Protokoll



Datenübertragung gemäß XY2-100-Protokoll



* Bei i*DRIVE*-Scan-Systemen (intelli*SCAN*, intelli*SCAN*$_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario*SCAN*$_{de}$) 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 wird der STATUS1± Kanal nicht benötigt.

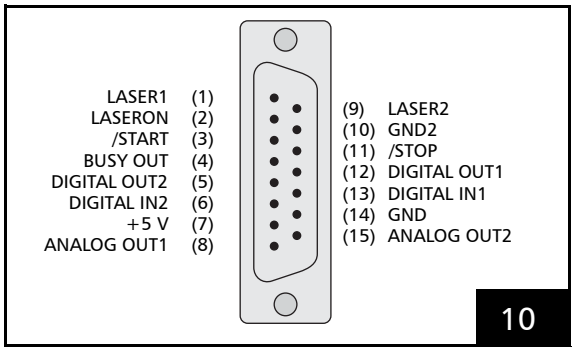
9

Layout des Datenkabels und Pinbelegung

4.4 Schnittstellen zum Laser und zur Peripherie

4.4.1 Laseranschluss-Buchse

Abb. 10 zeigt die Pinbelegung der 15-poligen Sub-D-Buchse für den Laseranschluss (LASER).



Pinbelegung der 15-poligen Sub-D-Buchse für den Laseranschluss (LASER)

Lasersignale

Die Laserausgangssignale LASER1 und LASER2 auf den Pins (1) und (9) unterscheiden sich je nach ausgewählter Lasersteuerungsversion, siehe die entsprechenden Zeitdiagramme in Kapitel 7.4 "Lasersteuerung", Seite 144:

	LASER1	LASER2
CO_2-Version	Modulations-signal 1 bzw. Standbysignal	Modulations-signal 2 bzw. Standbysignal
YAG-Version 1-3, 5	Q-Switch-Signal	FirstPulseKiller-Signal
Lasermode 4	Modulations-signal bzw. Standbysignal	FirstPulseKiller-Signal
Lasermode 6	Modulations-signal bzw. Standbysignal	–

Alle Laserausgangssignale (LASERON, LASER1 und LASER2) werden permanent von der RTC5-Karte generiert. Es sind digitale TTL-Pegel-Signale und beziehen sich auf GND2.

Bei RTC5-Karten mit der Option "Optokoppler" sind GND2 und die Laserausgangssignale galvanisch von der PC-Masse (GND) entkoppelt. Andernfalls sind die Massen GND2 und GND identisch, siehe Kapitel 2.3 "Optionale Funktionalitäten", Seite 27.

Die maximale Belastung der Signale beträgt 20 mA.

Die Laserausgangssignale können mit dem Befehl **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-Lasersignal-Pegel gemäß der Spezifikation Ihrer Lasersteuerung. Beachten Sie die Dokumentation Ihres Lasers.

Mit **config_laser_signals** und **config_laser_signals_list** können Pins (1), (2) und (9) der Laseranschluss-Buchse konfiguriert werden, siehe Seite 146.

Externe Steuersignale

Die externen Steuersignale /START und /STOP (TTL-Pegel, low-aktiv) haben die PC-Masse (GND) als Bezugspunkt. Beide Eingangssignale sind intern über Pull-up-Widerstände mit +3,3 V verbunden. Siehe auch Kapitel 9.3.1 "Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation", Seite 239.

BUSY-Status

An Pin (4) wird der BUSY-Status als BUSY OUT-Signal ausgegeben. Das BUSY OUT-Signal steht auf HIGH, wenn der BUSY-Status gesetzt ist, siehe Kapitel 6.4.3 "Listenausführungsstatus", Seite 77. Das Signal bezieht sich auf GND.

Digitaler Eingang und Ausgang

Die RTC5 stellt einen 2-Bit-Digital-Eingang (DIGITAL IN1 und DIGITAL IN2) und einen gepufferten 2-Bit-Digital-Ausgang (DIGITAL OUT1 und DIGITAL OUT2) bereit.

Zur Programmierung des Eingangs und Ausgangs siehe "2 Bit-Digital-Eingang", Seite 238 und "2 Bit-Digital-Ausgang", Seite 233.

Signaleingänge:

- LOW-Pegel < 0,6 V, HIGH-Pegel > 2,3 V
- Max. Eingangsspannungsbereich: –0,5 V … +5,5 V
- Eingangswiderstand (Pull-up) > 4,7 kΩ

Signalausgänge:

- LOW-Pegel < 0,55 V, HIGH-Pegel > 3,8 V,
- Die maximale Belastung der Signale ist jeweils 20 mA

Die Signale beziehen sich auf GND.

Analoge Ausgänge

Die RTC5 stellt zwei vielseitig verwendbare analoge Ausgänge, die 12-Bit-Ausgänge ANALOG OUT1 und ANALOG OUT2, bereit(1).

Zur Programmierung der Ausgänge siehe "12-Bit-Analog-Ausgänge", Seite 233.

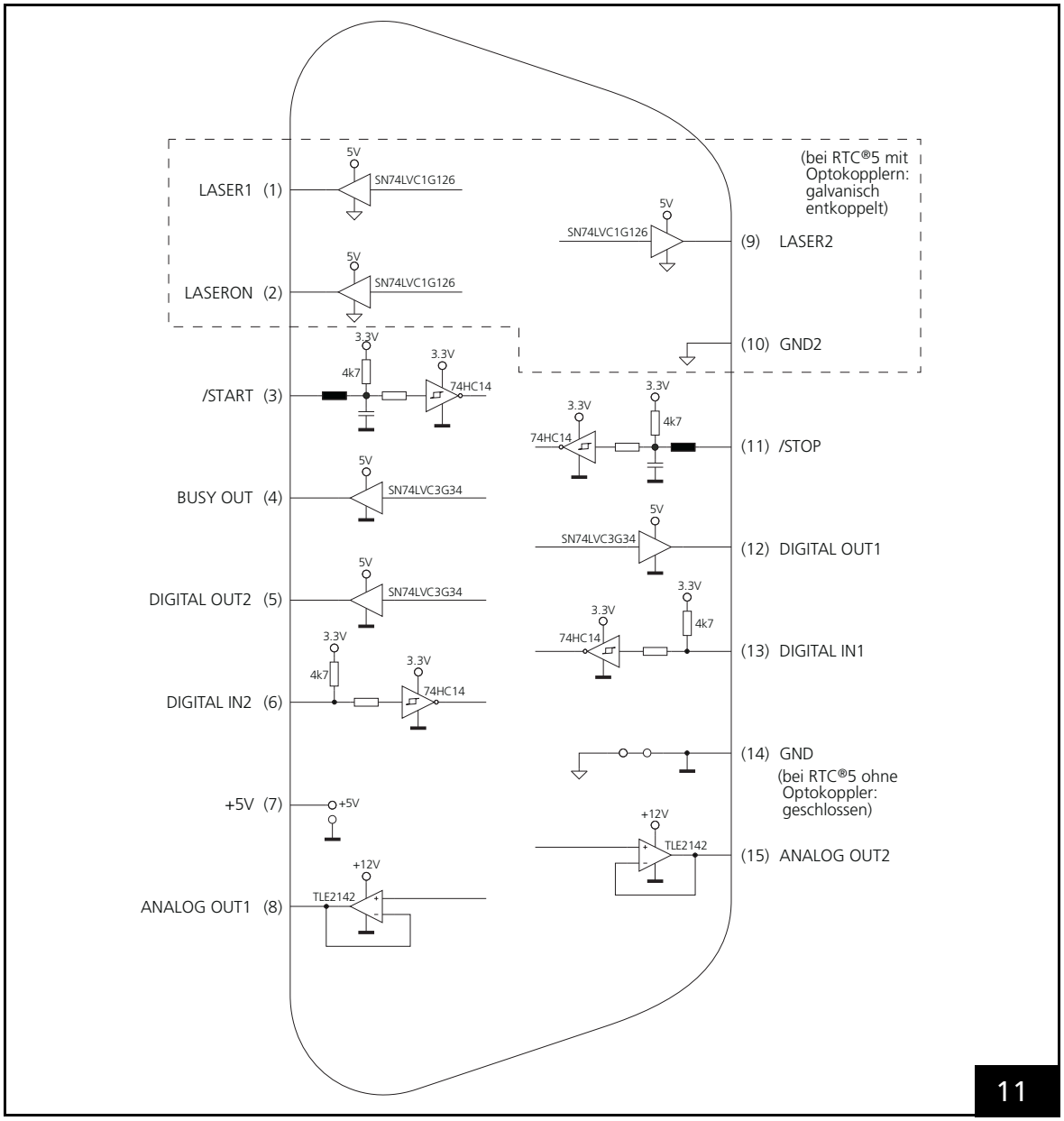
Der Ausgangsspannungsbereich beider Ausgänge ist auf 0 V … 10 V eingestellt.

Beide Signale haben die PC-Masse (GND) als Bezugspunkt. Die maximale Belastung der Signale ist jeweils 5 mA.

Ausgangsbeschaltung

Abb. 11 zeigt die Ausgangsbeschaltung der 15-poligen Sub-D-Laseranschluss-Buchse (LASER).

(1) Das ANALOG OUT2-Signal ist auch über die MARKING ON THE FLY-Stiftleiste verfügbar (siehe Seite 53).



11

Ausgangsbeschaltung der 15-poligen Sub-D-Laseranschluss-Buchse (LASER)

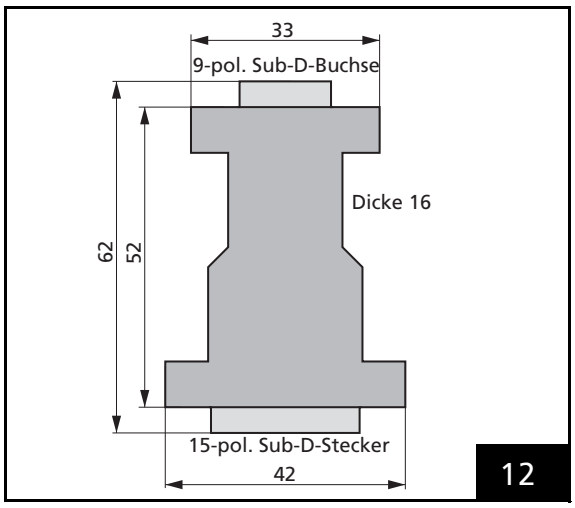
Laseradapter (optional)

Wenn der optionale Laseradapter von SCANLAB auf die 15-polige LASER-Buchse der RTC5 aufgesteckt wird, dann stehen über eine 9-polige Sub-D-Buchse die gleichen Signale (mit identischer Pinbelegung) wie an der 9-poligen LASER-Buchse der RTC4 zur Verfügung.

Hinweise

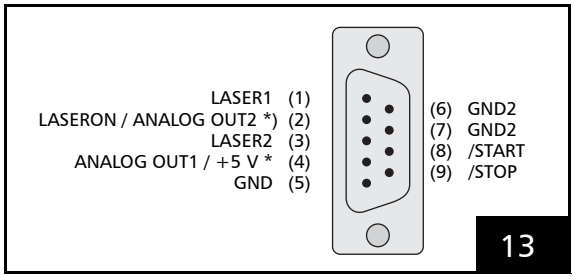
- Der Laseradapter kann *nicht* direkt auf die RTC5 aufgesteckt werden, wenn dort bereits ein XY2-100-Konverter aufgesteckt ist.
- Das BUSY OUT-Signal, der 2-Bit-Digital-Eingang und der 2-Bit-Digital-Ausgang stehen *nicht* über die 9-polige Sub-D-Buchse des Laseradapters zur Verfügung.

Abb. 12 zeigt die Abmessungen des Laseradapters.



Abmessungen des optionalen Laseradapters

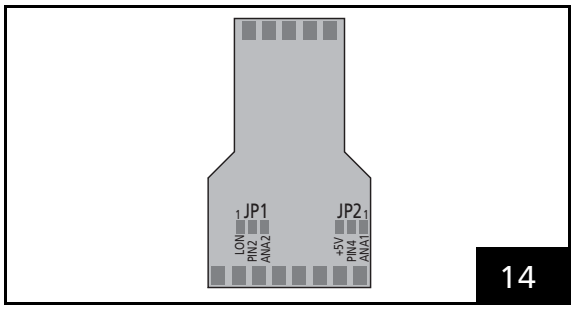
Die Pinbelegung der 9-poligen Sub-D-Buchse ist in Abb. 13 dargestellt.



Pinbelegung der 9-poligen Sub-D-Buchse des Laseradapters

* abhängig von der Einstellung der Jumper auf der Platine des Laseradapters

Die Signale an den Pins (2) und (4) der 9-poligen Sub-D-Buchse können über zwei Lötjumper im Laseradapter ausgewählt werden. Dazu muss das Gehäuse des Laseradapters vorsichtig (z. B. mit Hilfe eines Schraubenziehers) an seinen vier Clip-Verriegelungen geöffnet werden. Die Lötjumper JP1 und JP2 befinden sich auf der Platine des Laseradapters zwischen den beiden Sub-D-Anschlüssen. Abb. 14 zeigt die Position der Jumper auf der Platine. In der Tabelle darunter sind die möglichen Jumpereinstellungen aufgelistet.

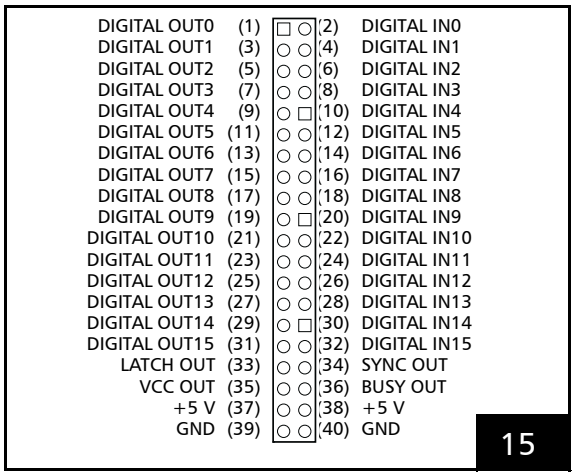


Position der Lötjumper JP1 und JP2 auf der Platine des Laseradapters

Jumper JP1 (Signal an Pin 2)	Position 1-2 LASERON (Standard-Konfiguration)	Position 2-3 ANALOG OUT2
Jumper JP2 (Signal an Pin 4)	Position 1-2 ANALOG OUT1 (Standard-Konfiguration)	Position 2-3 +5 V

4.4.2 EXTENSION 1-Stiftleiste

Abb. 15 zeigt die Pinbelegung der 40-poligen EXTENSION 1-Stiftleiste.



Pinbelegung der 40-poligen EXTENSION 1-Stiftleiste (on-board)

Auswahl des Ausgangssignal-Pegels

Mit dem Jumper **JP1** auf der Rückseite der RTC5 (siehe Abb. 3) kann für alle Ausgangssignale der EXTENSION 1-Stiftleiste (DIGITAL OUT0-15, LATCH_OUT, SYNC_OUT, BUSY OUT, VCC_OUT) zwischen einem Signal-Pegel von 5 V oder 3,3 V ausgewählt werden (siehe Kapitel 2.4.1 "Jumper JP1: Auswahl des Ausgangssignal-Pegels an der EXTENSION 1-Stiftleiste", Seite 28).

Als Kontrollmöglichkeit für den Benutzer wird der ausgewählte Signal-Pegel zusätzlich konstant an Pin (35) ausgegeben: Signal VCC_OUT. VCC_OUT hat die PC-Masse (GND) als Bezugspunkt. Die maximale Belastung des Signals ist 100 mA.

16-Bit-Digital-Eingang und -Ausgang

An der 40-poligen EXTENSION 1-Stiftleiste wird ein geschützter 16 Bit-Digital-Eingang (TTL-Pegel) und ein gepufferter 16 Bit-Digital-Ausgang (TTL-Pegel) bereitgestellt (siehe Abb. 15). Der Ausgangssignal-Pegel muss über eine Jumpereinstellung konfiguriert werden (s. o.).

Zur Programmierung des Eingangs und Ausgangs siehe "16 Bit-Digital-Eingang", Seite 238, "16 Bit-Digital-Ausgang", Seite 232 und "Bedingte Befehlsausführung", Seite 244.

Die Eingangs- und Ausgangssignale haben die PC-Masse (GND) als Bezugspunkt. Die maximale Belastung der Ausgangssignale ist 8 mA.

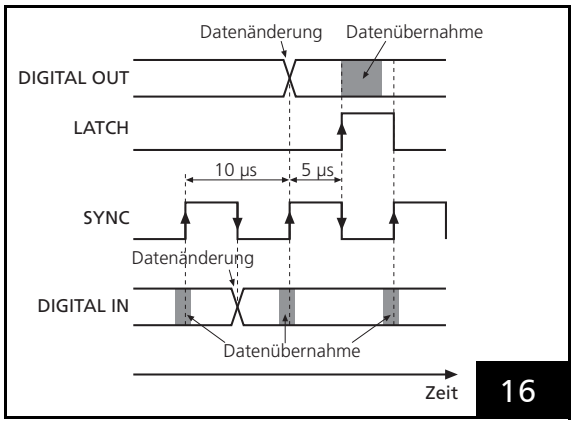
Synchronisation der Datenübergabe

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

Das LATCH-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 RTC5 erzeugt, wenn der Ausgangswert am 16-Bit-Digital-Ausgang geändert 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 geändert wurde (siehe Abb. 16).

Zur Synchronisation der Datenübergabe am 16-Bit-Digital-Eingang wird an Pin (34) das SYNC-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-Signals vorgenommen werden. Der DSP der RTC5 übernimmt den aktuell angelegten Wert immer mit der steigenden Flanke des SYNC-Signals (siehe Abb. 16).

Die Synchronisationssignale haben die PC-Masse (GND) als Bezugspunkt. Die maximale Belastung ist 10 mA.



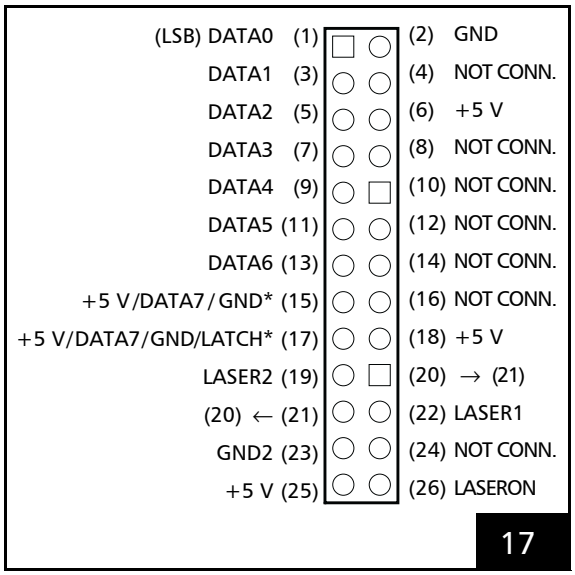
Synchronisation der Datenübergabe mit LATCH- oder SYNC-Signal

BUSY-Status

Das BUSY OUT-Signal an Pin (36) ist identisch mit dem BUSY OUT-Signal an der LASER-Buchse (siehe Seite 47). Das Signal bezieht sich auf GND.

4.4.3 EXTENSION 2-Stiftleiste

Die 26-polige EXTENSION 2-Stiftleiste auf der RTC5-Karte (auf der RTC4 ist die entsprechende Stiftleiste mit LASER EXTENSION bezeichnet) stellt einen gepufferten digitalen 8-Bit-Ausgang (DATA0 bis DATA7) bereit. Pin (15) und Pin (17) werden über die Jumpereinstellung konfiguriert.



Pinbelegung der EXTENSION 2-Stiftleiste (on-board)
* abhängig von der Jumpereinstellung

Jumper-Einstellung

Pin (15) und (17) der EXTENSION 2-Stiftleiste werden über die Jumper **JP2-JP8** auf der Rückseite der RTC5 (siehe Abb. 3) konfiguriert. Das höchstwertige Bit (DATA7) des 8-Bit-Ausgangswertes kann Pin (15) oder Pin (17) zugewiesen werden. Alternativ können beide Pins auch permanent auf +5 V (HIGH) oder GND (LOW-Pegel) gelegt werden. Pin (17) kann alternativ auch das LATCH-Signal zugeordnet werden (s.u.). Die Zuweisung erfolgt durch Schließen des entsprechend beschrifteten Jumpers (siehe Seite 28).

Beispiele

- Wenn das DATA7-Bit Pin (15) zugewiesen wird, dann ist der volle 8-Bit Ausgangswert am Ausgang (an den ungeradzahligen Pins (1) bis (15) der EXTENSION 2-Stiftleiste) verfügbar.
- Wird Pin (15) permanent auf HIGH gelegt, so führt das zu einem Offset von 128 für die Ausgabewerte, d.h. die Ausgabewerte liegen zwischen 128 und 255.
- Wird Pin (15) auf LOW gelegt, so sind nur Ausgabewerte zwischen 0 und 127 möglich.
- Das DATA7-Bit kann für andere Zwecke verwendet werden, indem es Pin (17) zugewiesen wird.

Lasersignale

Die Ausgangssignale LASER1 und LASER2 hängen wie bei der LASER-Buchse von der ausgewählten Lasersteuerungsversion ab (siehe Abschnitt "Lasersignale", Seite 47). Die Signale beziehen sich wie bei der LASER-Buchse auf GND2.

Bei RTC5-Karten mit der Option "Optokoppler" sind GND2 und die Laserausgangssignale galvanisch von der PC-Masse (GND) entkoppelt. Andernfalls sind die Massen GND2 und GND identisch (siehe Kapitel 2.3 "Optionale Funktionalitäten", Seite 27).

8-Bit-Digital-Ausgang

Der gepufferte 8-Bit-Digital-Ausgang (TTL-Pegel) ist für einen YAG-Laser mit digitaler Steuerung des Lampenstroms vorgesehen, kann aber natürlich auch für andere Zwecke eingesetzt werden. Vor der ersten Wertezuweisung befindet er sich im (hochohmigen) tristate-Zustand.

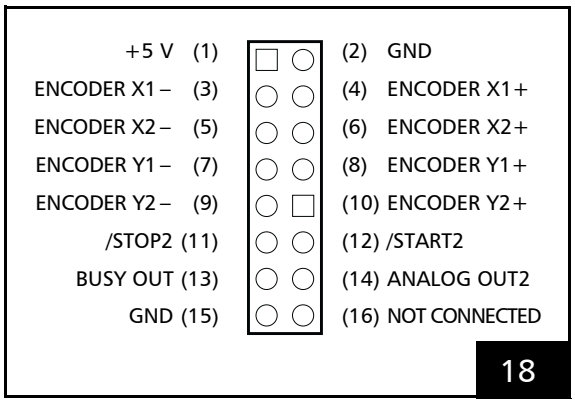
Zur Programmierung des Ausgangs siehe "8 Bit-Digital-Ausgang", Seite 233.

Das höchstwertige Bit (DATA7) des Ausgabewerts kann für Sonderzwecke eingesetzt werden, z.B. zur Steuerung eines Shutters. Dazu kann es einem anderem Pin auf der EXTENSION 2-Stiftleiste zugewiesen werden (s.o.).

Bei entsprechender Jumper-Einstellung (s.o.) wird an Pin (17) ein LATCH-Signal ausgegeben. Das LATCH-Signal (ein 5 µs-Puls, high aktiv) wird automatisch von der RTC5 erzeugt, wenn der Wert am 8-Bit-Digital-Ausgang geändert wird. Falls am 8-Bit-Digital-Ausgang mehrere Bits simultan als Datenwort (also nicht unabhängig voneinander) übertragen werden, dann sollte das LATCH-Signal zur Synchronisation der Datenübergabe verwendet werden (etwa als Triggersignal für den Laser zur Übernahme eines geänderten Lampenstromwertes). 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 geändert wurde (siehe auch Abb. 16). Das LATCH-Signal hat die PC-Masse (GND) als Bezugspunkt. Die maximale Belastung ist 10 mA.

4.4.4 MARKING ON THE FLY-Stiftleiste

Die MARKING ON THE FLY-Stiftleiste stellt u. a. Encoder-Eingänge zur Einbindung von Encoder-Signalen zur Verfügung. Zusammen mit der optionalen "Processing on the fly"-Funktionalität kann damit eine Lasermaterialbearbeitung bewegter Werkstücke (z. B. auf einem Förderband oder auf einer rotierenden Scheibe) realisiert werden (siehe Seite 199). Die Pinbelegung ist in Abb. 18 dargestellt.



Pinbelegung der Stiftleiste MARKING ON THE FLY (on-board)

Encoder-Eingänge

Die RTC5 ist mit zwei Encoder-Eingängen (ENCODER X und ENCODER Y) ausgestattet, die jeweils für ein Paar (1, 2) von standardisierten differentiellen Eingangssignalen (RS422) vorgesehen sind (siehe auch Seite 247).

Externe Steuersignale

Die externen Steuersignale /START2 und /STOP2 (TTL-Pegel, low-aktiv) haben die PC-Masse (GND) als Bezugspunkt. Beide Eingangssignale sind intern über Pull-up-Widerstände mit +3,3 V verbunden. Siehe auch Kapitel 9.3.1 "Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation", Seite 239).

Analoger Ausgang

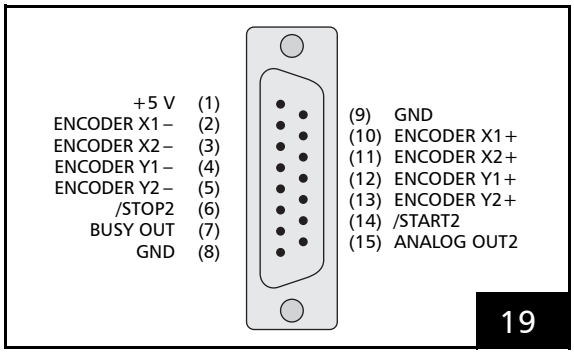
Der 12-Bit-Analogausgang ANALOG OUT2 an Pin (14) ist identisch mit dem Analogausgang ANALOG OUT2 an der LASER-Buchse (siehe Seite 48). Das Ausgangssignal bezieht sich auf GND.

BUSY-Status

Das BUSY OUT-Signal an Pin (13) ist identisch mit dem BUSY OUT-Signal an der LASER-Buchse (siehe Seite 47). Das Signal bezieht sich auf GND.

Slotblech (optional)

Zur Verwendung der Eingänge und Signale an der MARKING ON THE FLY-Stiftleiste empfiehlt SCANLAB den Gebrauch eines zusätzlichen Slotblechs. Ein geeignetes Slotblech mit einer 15-poligen Sub-D-Buchse ist bei SCANLAB erhältlich. Die Pinbelegung der Sub-D-Buchse zeigt Abb. 19.



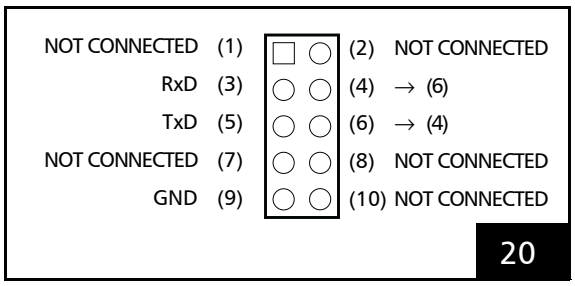
Pinbelegung der 15-poligen Sub-D-Buchse des optionalen MARKING ON THE FLY-Slotblechs

4.4.5 RS232-Stiftleiste

Zur Ansteuerung externer Geräte wird an der "RS232" Stiftleiste eine RS232-Schnittstelle bereitgestellt. Die Pinbelegung ist in Abb. 20 dargestellt.

Max. Eingangsspannungsbereich: –25 V … +25 V
Max. Ausgangsspannungsbereich: –13 V … +13 V

Die Signale beziehen sich auf die PC-Masse GND.



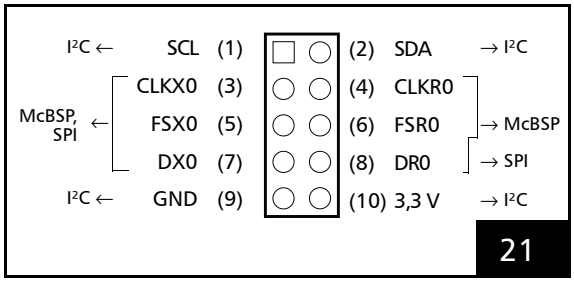
Pinbelegung der Stiftleiste "RS232" (on-board)

Mit **rs232_config** kann die Baud-Rate eingestellt werden (Voreinstellung: 9600 Baud). Die anderen Parameter können nicht verändert werden (Data-Bits: 8, Start-Bits: 1, Stopp-Bits: 1, Parity: None).

Zum Ausgeben und Einlesen von Daten über die RS232-Schnittstelle siehe Seite 237 und Seite 238.

4.4.6 SPI / I²C-Stiftleiste

Die "SPI / I2C"-Stiftleiste ist eine Hardware-Schnittstelle, die für verschiedene Datenaustauschverfahren vorgesehen ist: McBSP (Multi channel Buffered Serial Port), SPI (Serial Peripheral Interface) sowie I²C (Inter-Integrated Circuit). Abb. 21 zeigt die entsprechenden Pinbelegungen. Abhängig vom verwendeten Verfahren wird jedoch nur ein Teil der Pins tatsächlich verwendet.



Pinbelegung der Stiftleiste "SPI / I2C" (on-board). Die für die verschiedenen Datenaustauschverfahren relevanten Pins sind bezeichnet.

Verwendung als McBSP-Schnittstelle

Als Datenaustauschverfahren an der "SPI / I2C"-Stiftleiste ist McBSP werksseitig voreingestellt.
Die "SPI / I2C"-Stiftleiste wird automatisch mit `DataDelay = 1` und einer Taktfrequenz von 8 MHz initialisiert. Für McBSP sind die Pins (3), (5), (7) für die ausgehenden Signale und die Pins (4), (6), (8) für die eingehenden Signale relevant.

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

- das Taktsignal CLKX an Pin (3)
- alle 10 μs ein Frame-Synchronization-Signal FSX an Pin (5)
- ein Datensignal DX an Pin (7) (Voreinstellung: SampleY|SampleX), siehe **set_mcbsp_out**.

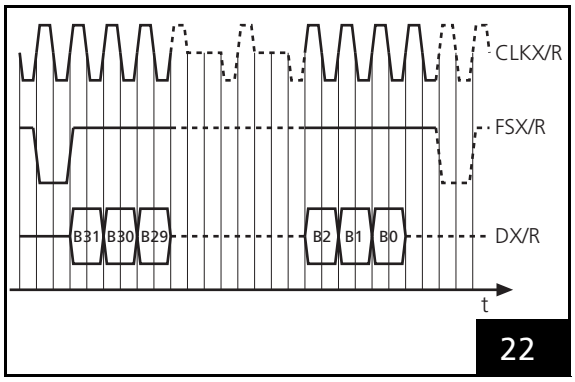
Die McBSP-Schnittstellenfunktionalität samt zugehörige Befehle werden u. a. bereitgestellt, um Positionssignale direkt in Anwendungen einbinden zu können.

- Bei "Processing on the fly"-Anwendungen können beispielsweise die Positionssignale eines Roboterarms dazu verwendet werden, eine Lasermaterialbearbeitung mit bewegten Scan-Systemen zu realisieren (siehe Seite 199 und Seite 248) – alternativ zu einer über Encoder-Signale gesteuerten "Processing on the fly"-Bearbeitung von bewegten Werkstücken.
- Alternativ können die über die McBSP-Schnittstelle eingehenden Signale auch dazu verwendet werden, das Scan-System mit kontrollierbarem Timing über dem Werkstück auszurichten (siehe "Online-Positionierung", Seite 187 und Seite 248).

Zum Ausgeben und Einlesen von Daten mit der McBSP-Schnittstellenfunktionalität siehe auch Kapitel 9.1.7 "McBSP/SPI-Schnittstelle", Seite 237 und Kapitel 9.2.4 "McBSP/SPI Schnittstelle", Seite 238.

Bei der 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 die PC-Masse GND.



Timing-Diagramm der McBSP-Signale bei 1 Bit Daten-Delay (Voreinstellung; `XDelay` = `RDelay` = 1)

RTC5 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
  - DataDelay N Bit
- Das Timing-Diagramm der McBSP-Signale ist in Abb. 22 dargestellt. Bei steigender Taktflanke wird ein Frame-Synchronization-Signal (low-aktiv) erzeugt, das nur einen Takt (1 Datenbit) lang ist. Die 32 Datenbits werden nach `XDelay`-Takten mit jeweils steigender Taktflanke in der Reihenfolge Bit31...Bit0 gesendet. Die Sendefrequenz beträgt in der Standardeinstellung 8 MHz (4 µs pro Datenwort). Sie kann aber auch mit **set_mcbsp_freq** auf einen Wert zwischen 4 MHz und 16 MHz eingestellt werden.
- Mit **set_mcbsp_out** und **set_mcbsp_out_ptr** kann ausgewählt werden, welche Datensignale ausgegeben werden. Die Signale werden permanent im 10 µs-Takt ausgegeben.

RTC5 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
  - DataDelay N Bit
- Das Timing-Diagramm der McBSP-Signale ist in Abb. 22 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 Datenbits in der Reihenfolge Bit31…Bit0 detektiert, sofern sie mit steigender Flanke gesendet werden. Beachten Sie dazu auch die folgenden Hinweise:
  - Die Bitfrequenz (Empfangsfrequenz) wird ausschließlich aus den eingehenden Taktsignalen bestimmt und darf maximal 16 MHz betragen.
  - Nach dem letzten Datenbit Bit0 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.
- Ab OUT 526: Nach einem **load_program_file** werden die McBSP-Daten intern permanent abgeholt und zwischengespeichert, sobald (neue) Daten übertragen werden. Neuere Datenworte überschreiben ältere.
  - Nach einem Reset mit **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 der Online-Positionierung mit **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** bzw. **set_mcbsp_matrix** werden die Eingabewerte an die internen Speicherstellen 1 oder 2 übertragen (siehe Seite 187).
  - Der jeweils zuletzt vollständig an eine Speicherstelle übertragene Wert kann mit **read_mcbsp** von der jeweiligen Speicherstelle ausgelesen werden.

  Das jeweils erste Frame-Synchronization-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen. Gegebenenfalls sollte immer erst ein Dummy-Wert zur Schnittstelle geschickt werden (das gilt sowohl für "Processing on the fly"-Anwendungen als auch die Online-Positionierung). Mit **read_mcbsp** kann dann auf eine erfolgreiche Übertragung geprüft werden.
- Bei Verwendung der Version OUT 525 oder älter, siehe auch Versionsinfo unter **get_mcbsp** und **get_mcbsp_list**.
- Ab OUT 532: 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.
- Ab OUT 537: 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.

Verwendung als I²C-Schnittstelle

Die “SPI / I2C”-Stiftleiste wird beim Einsatz der optionalen ADC-Aufsteckplatine benutzt, um die Digitalwerte an den DSP zu übertragen, siehe auch Kapitel 4.4.8 ”Analog-Eingänge (Sonderzubehör)”, Seite 59.

Die I²C-Taktsignale (SCL an Pin (1)) und I²C-Datensignale (SDA an Pin (2)) sind derzeit nicht anderweitig frei verfügbar.

Verwendung als SPI-Schnittstelle

Die “SPI / I2C”-Stiftleiste kann bei Bedarf für eine serielle synchrone Datenübertragung eingestellt werden. Es wird dann die SPI-Funktionalität statt der McBSP-Funktionalität (Voreinstellung) verwendet. Dazu kann jederzeit der Befehl **mcbsp_init_spi** aufgerufen werden. Eine laufende Datenübertragung wird dadurch abgebrochen. Mit **mcbsp_init** kann der vorherige Zustand (McBSP-Funktionalität) wieder hergestellt werden.

Nach der Initialisierung kann pro 10 µs nur ein Datenwort übertragen werden (siehe z. B. **set_mcbsp_in**).

Die voreingestellte Übertragungsfrequenz ist 8 MHz (4 µs pro Datenwort). Alternativ kann ein Wert aus dem Bereich von 4…16 MHz eingestellt werden. Dazu wird der Befehl **set_mcbsp_freq** aufgerufen (siehe auch **mcbsp_init** oder **mcbsp_init_spi**).

Nach dem Aufruf des Befehls **mcbsp_init_spi** fungiert die RTC5 im Sinne des SPI-Standards als SPI-Master Device. Die RTC5 ist ein Master Device für nur einen SPI-Slave. Für den Betrieb sind notwendig:

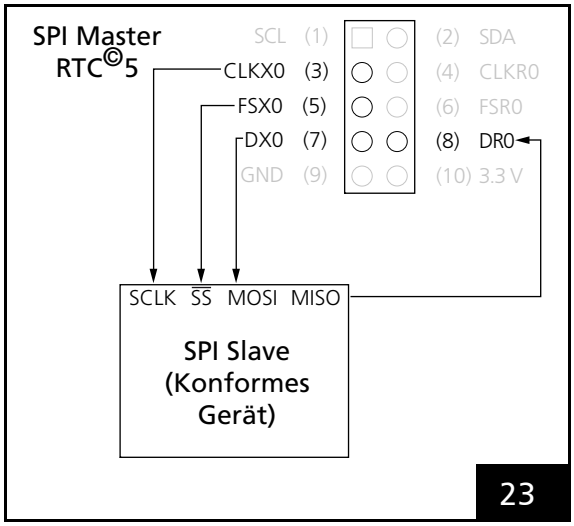
- 1 Steuerleitung für das Taktsignal (CLKX0 an Pin (3))
- 1 Steuerleitung für das low-aktive Slave-Select-Signal (FSX0 an Pin (5))
- 1 Datenleitung - aus Sicht des Masters - ausgehend (DX0 an Pin (7))
- 1 Datenleitung - aus Sicht des Masters - eingehend (DR0 an Pin (8))

Pin (4) CLKR0 und Pin (6) FSR0 werden nicht angeschlossen.

Für alle Pegelsignale gilt die Spezifikation 3,3 V TTL oder 5 V TTL.

Die Signalabfolge ist wie beim McBSP-Mode, jedoch sind `XDelay` und `RDelay` *immer* = 1.

Zur Verschaltung der RTC5 mit einem SPI-Slave Device siehe Abb. 23.



RTC5 als SPI-Master mit SPI-Slave – Signale und Verschaltung. SCLK = Serial Clock, MOSI = Master Output - Slave Input, MISO = Master Input - Slave Output, SS = Slave Select. Anmerkung: In der Literatur sind die relevanten Abkürzungen und Benennungen uneinheitlich, d.h. es gibt für die hier Gewählten eine Vielzahl von Synonymen.

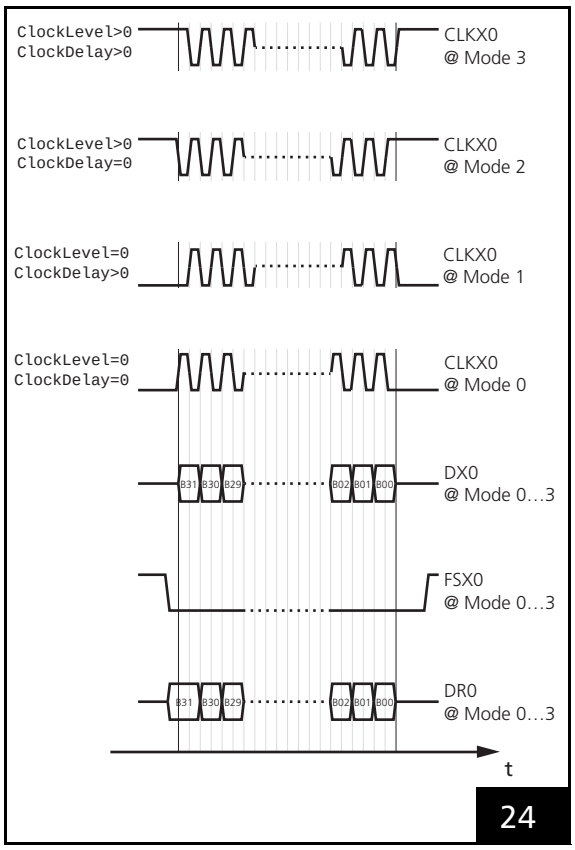
Im Gegensatz zur McBSP-Datenübertragung erfolgen bei SPI Dateneingang und Datenausgang synchron mit einem gemeinsamen, vom SPI-Master generierten Taktsignal. Das SPI Timing-Diagramm ist in Abb. 24 dargestellt.

Mit dem Befehl **mcbsp_init_spi** können die Taktsignaleigenschaften ’Polarität’ (`ClockLevel`) und ’Phase’ (`ClockDelay`) eingestellt werden. Daraus ergeben sich die gemäß SPI-Standard mit Mode 0...Mode 3 bezeichneten Übertragungsformate, siehe nachfolgende Tabelle.

SPI-Mode	Polarität des Taktsignals	Phase des Taktsignals
0	ist LOW wenn inaktiv – `ClockLevel=0`	ist zeitgleich mit den Datenbits – `ClockPhase=0`
1	ist LOW wenn inaktiv – `ClockLevel=0`	ist um ein halbes Taktsignal verzögert – `ClockPhase>0`
2	ist HIGH wenn inaktiv – `ClockLevel>0`	ist zeitgleich mit den Datenbits – `ClockPhase=0`
3	ist HIGH wenn inaktiv – `ClockLevel>0`	ist um ein halbes Taktsignal verzögert – `ClockPhase>0`

Alle Signale beziehen sich auf die PC-Masse GND.

Im Gegensatz zur McBSP-Konfiguration sind alle Signale unmittelbar nach der Initialisierung an den entsprechenden Pins statisch und die RTC5 generiert kein permanentes Taktsignal.

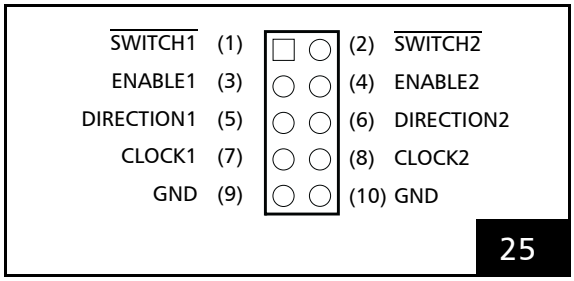


Timing-Diagramm der SPI-Signale. Das Taktsignal ist bei Modus 0, 1, 2 und 3 unterschiedlich in Bezug auf Polarität und Phase.
Der Abstand der Gitterlinien ist ein *halbes* Taktsignal.

4.4.7 STEPPER MOTOR-Stiftleiste (Schrittmotoransteuerung)

An der "STEPPER MOTOR"-Stiftleiste können Signale zur Ansteuerung von zwei Schrittmotoren zur Verfügung gestellt werden(1).

Die Pinbelegung ist in Abb. 25 dargestellt.



Pinbelegung der Stiftleiste "STEPPER MOTOR" (on-board)

Alle Signale beziehen sich auf die PC-Masse GND.

Die Ausgangssignale (ENABLE, DIRECTION und CLOCK) sind TTL-Pegel-Signale (5 V). Als CLOCK-Signal generiert die RTC5 periodisch high-aktive 5 µs-Pulse (zwischen diesen Pulsen ist das CLOCK-Signal permanent 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.

Die SWITCH-Eingänge (low-aktiv) sind intern über 10 kΩ-Pull-up-Widerstände mit +3,3 V verbunden. Damit können Endschaltersignale (3,3 V oder 5 V TTL-Signale oder Eingang auf Masse ziehen) eingebunden werden.

Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoransteuerung", Seite 234.

(1) Schrittmotorsignale stehen nur für RTC5-Karten mit DSP-Versionsnummer 2 und größer (siehe **get_rtc_version** Bits #16-23) zur Verfügung. Bei älteren RTC5-Karten werden die Schrittmotorbefehle nicht ausgeführt.

4.4.8 Analog-Eingänge (Sonderzubehör)

Wird die RTC5 PCI-Karte mit der ADC-Aufsteckplatine (Sonderzubehör nur für die RTC5 PCI-Karte, Artikelnummer 121126) bestückt, dann verfügt sie über die zwei 10 V-Analog-Eingänge ANALOG IN0 und ANALOG IN1 (zu ANALOG IN2 siehe Abschnitt Hinweise, Seite 59).

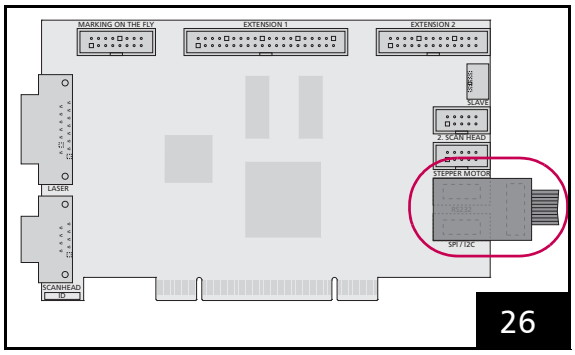
Die an den beiden Analog-Eingängen angelegten Spannungen werden unaufgefordert in einer Endlos-Serie automatisch konvertiert (Dauer ca. 0,3 ms) und als 12-Bit-Digitalwerte zum DSP übertragen (I^2C-Bus mit 400 kHz, fast mode).

Die angelegten Werte können mit dem Kontrollbefehl **read_analog_in** jederzeit ausgelesen werden.

Ab Version DLL 543, OUT 543, RBF 524 können die analogen Eingabewerte (ANALOG IN0 und ANALOG IN1, siehe **read_analog_in**) mit dem Triggersignal 54 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**). Die Old-Bits werden jedoch nicht aufgezeichnet. Das Format der Daten ist ((ANALOG IN1 << 16) + ANALOG IN0).

Installation und Pinbelegung

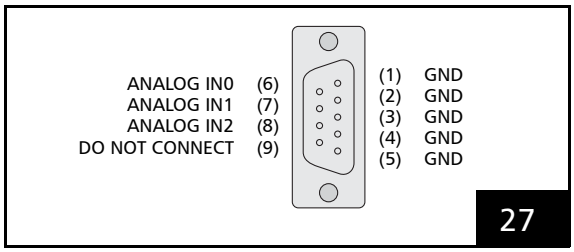
Die ADC-Aufsteckplatine muss auf die beiden Stiftleisten "SPI / I2C" und "RS232" der RTC5 PCI-Karte aufgesteckt werden. Achten Sie dabei auf die korrekte Orientierung der Aufsteckplatine (siehe Abb. 26): das angelötete Flachbandkabel muss nach außen zeigen.



ADC-Aufsteckplatine auf RTC5 PCI-Karte

Intern ist die ADC-Aufsteckplatine nur mit den Pins 1, 2, 9 und 10 der "SPI / I2C" Stiftleiste verbunden. Die Verbindung zur "RS232" Stiftleiste dient nur zur mechanischen Stabilität.

Die Analog-Eingänge stehen dann über einen 9-poligen Sub-D-Stecker am angelöteten Flachbandkabel bereit. Dessen Pinbelegung zeigt Abb. 27.



ADC-Aufsteckplatine: Pinbelegung des 9-poligen Sub-D-Steckers

Spezifikationen

- Eingangsspannungsbereich: 0 V … 10 V
- Eingangsimpedanz: 4 kΩ
- ADC-Auflösung: 12 Bit

Die Eingangs-Signale beziehen sich auf GND.

Hinweise

- Die ADC-Aufsteckplatine stellt auch noch einen *dritten* 10 V-Analog-Eingang ANALOG IN2 mit identischen Spezifikationen zur Verfügung (siehe Abb. 27). Jedoch können die Eingangswerte dieses dritten Analog-Eingangs mit **read_analog_in** weder ausgelesen noch mit **set_trigger** oder **set_trigger4** aufgezeichnet werden.

5 Installation und Inbetriebnahme

Die Installation der RTC5 erfolgt in den folgenden Schritten (gehen Sie in der vorgeschlagenen Reihenfolge vor):

(1) Konfiguration der RTC5-Jumper (siehe Seite 60)

(2) Hardware-Installation: Einbau der RTC5-Karte (siehe Seite 60)

(3) Installation der Treiber (siehe Seite 61)

(4) Installation der RTC5-Software (siehe Seite 62)

(5) Beachten Sie beim Starten des Laser-Systems die sichere Einschaltreihenfolge (siehe Seite 63).

(6) Für einen Funktionstest nach der Installation kann z. B. das mitgelieferte HPGL-Converter-Programm verwendet werden (siehe Seite 63).

5.1 Jumpereinstellungen

Die RTC5 kann von SCANLAB in unterschiedlichen Jumperkonfigurationen ausgeliefert werden. Die Jumper sind Lötverbindungen und können vom Kunden mit einem Lötkolben auch nachträglich verändert werden (siehe Seite 28).

Wenn Sie die werkseigene Einstellung nicht verändern wollen, so fahren Sie mit der Installation bei Kapitel 5.2 "Einbau der RTC5-Karte", Seite 60 fort.

Vorsicht!

- Achten Sie bei einer Änderung der Jumpereinstellung darauf, die Elektronik auf der Karte nicht zu beschädigen!

5.2 Einbau der RTC5-Karte

Vorsicht!

- Die RTC5 sollte stets 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 RTC5 nicht an den elektrischen Kontakten.
- Schützen Sie die Karte vor Feuchtigkeit, Staub, korrosiven Dämpfen und mechanischen Belastungen.

Der Einbau der RTC5 in den PC erfolgt in den folgenden Arbeitsschritten:

(1) Entfernen Sie alle Disketten und CDs aus den Laufwerken Ihres PCs.

(2) Fahren Sie das Betriebssystem herunter, schalten Sie den Computer 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 PCI-Steckplatzes. Von der Oberseite des Steckplatzes aus muss eine Höhe von mindestens 105 mm verfügbar sein.

(6) Nehmen Sie die RTC5 aus dem antistatischen Beutel. Berühren Sie die Karte nicht an den Kontaktstellen.

(7) Wenn Sie die Signale der RTC5-Stiftleisten verwenden wollen, dann schließen Sie entsprechende Verbindungskabel an. SCANLAB empfiehlt den Gebrauch zusätzlicher Slotbleche mit geeigneten Steckern, so dass die Signale auch bei geschlossenem PC-Gehäuse zugänglich sind. Alle Anschlüsse und Signale der RTC5 sind in Kapitel 4 "Layout und Schnittstellen", Seite 40 beschrieben.

(8) Stecken Sie die RTC5 in den vorgesehenen PCI-Steckplatz. Beachten Sie dabei die Hinweise im Handbuch Ihres Rechners(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 RTC5 – gegebenenfalls unter Einbindung des XY2-100-Konverters – über ein Datenkabel mit dem Scan-System (siehe "Schnittstellen zum Scan-System", Seite 42).

(12) Verbinden Sie die 15-polige Sub-D-Buchse an der RTC5 über ein geeignetes Interface mit dem Laser (siehe "Schnittstellen zum Laser und zur Peripherie", Seite 47).

(13) Verbinden Sie gegebenenfalls eine oder mehrere RTC5-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 Computer wieder an.

5.3 Installation der Treiber

Zur Installation der RTC5-Treiber für Windows 10, 8, 7, Vista, XP SP2, XP SP3 gehen Sie wie folgt vor:

- Nachdem Sie die RTC5-Karte in den PC eingebaut haben, starten Sie den Computer.

Unter Windows 7, Vista, XP SP2, XP SP3:

- Wenn der "Add Hardware Wizard" nicht automatisch startet, dann starten Sie ihn vom Control-Panel (Systemsteuerung) aus.
- Spezifizieren Sie im "Add Hardware Wizard" das Verzeichnis, in dem sich die RTC5-Treiber befinden. Während der Installation wählt dann das Betriebssystem automatisch die geeignete Treiber-Datei (32- oder 64-Bit) aus.

Unter Windows 10, 8:

- 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 die RTC5-Treiber befinden.

Vorsicht!

- Die RTC5-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Busses abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Deaktivieren Sie insbesondere den Windows-Sleep-Modus (dieser ist bei einigen Windows-Betriebssystemen standardmäßig voreingestellt). Siehe hierzu auch Hinweis auf Seite 26.

(1) Sollen mehrere RTC5-Karten mit Master/Slave-Synchronisation in einem PC betrieben werden, dann müssen die RTC5-Karten zuerst paarweise über die MASTER- und SLAVE-Anschlüsse miteinander verbunden und dann in benachbarte PCI-Steckplätze gesteckt werden (siehe auch "Master/Slave-Synchronisation", Seite 41).

5.3.1 Software-Paket-Upgrades

Wenn Sie bisher ein RTC5-Software-Paket Version 2012_09_05 oder früher (enthält noch WDM-Treiber) benutzen, und auf ein RTC5-Software-Paket Version 2013_02_21 oder später (enthält schon WDF-Treiber) umstellen wollen, gehen Sie wie folgt vor:

(1) Installieren Sie zunächst nur die RTC5-Software-Dateien, siehe Kapitel 5.4 "Installation der RTC5-Software", Seite 62.

(2) Prüfen Sie die korrekte Funktionsfähigkeit Ihrer Anwendung.

(3) Installieren Sie erst dann den neuen Treiber (das ist ein WDF-Treiber). Ein schneller Rücktausch auf ältere RTC5-Software-Dateien ist dann nämlich nicht mehr möglich, ohne auch den Treiber rückzutauschen (siehe auch Hinweise auf Seite 26).

(4) Rechtsklicken Sie auf das mitgelieferte Script `ScanlabClassChecker.cmd` und wählen aus dem Kontextmenü 'Als Administrator ausführen'. Weitere Informationen finden Sie in der Datei `ReadMe_ScanlabClassChecker.pdf`.

5.3.2 Tausch von RTC-Karten und Update von RTC-Karten-Treibern

Nach dem Tausch von RTC-Karten des Typs RTC3, RTC4, RTC *SCAN*alone, RTC5 gegen eine RTC5 (auch: RTC6 PCI-Express-Karte) oder um einen (veralteten) WDM-Treiber auf einen (neueren) WDF-Treiber zu aktualisieren, gehen Sie wie folgt vor:

(1) Installieren Sie den aktuellen Treiber für die neue Karte (das ist ein WDF-Treiber; herunterladbar von der SCANLAB Homepage).

(2) Rechts-klicken Sie auf das mitgelieferte Script `ScanlabClassChecker.cmd` und wählen aus dem Kontextmenü 'Als Administrator ausführen'. Weitere Informationen finden Sie in der Datei `ReadMe_ScanlabClassChecker.pdf`.

5.4 Installation der RTC5-Software

Zusätzlich zu den Treibern muss die RTC5-Software installiert (auf die Festplatte des PCs kopiert bzw. entpackt) werden. Sie besteht aus den folgenden Dateien:

- `RTC5DLL.dll` oder `RTC5DLLx64.dll` (anwendungsunterstützende Win32- oder Win64-basierte DLL)
- `RTC5OUT.out` (DSP-Programmdatei)
- `RTC5RBF.rbf` (Firmwaredatei)
- `RTC5DAT.dat` (binäre Hilfsdatei)
- (`*.CT5`) (Korrekturdatei/en)

Die Dateien sind im RTC5-Software-Paket enthalten. Zur Kennzeichnung und Verwaltung unterschiedlicher Softwareversionen werden einige der Dateien zusätzlich gezippt ausgeliefert (die `zip`-Dateinamen `RTC5<…>_<Version>.zip` enthalten die jeweiligen Versionsnummern).

Führen Sie die folgenden Schritte aus, um die RTC5-Software zu installieren:

- Kopieren bzw. entpacken Sie die Dateien `RTC5DLL.dll` (oder `RTC5DLLx64.dll`), `RTC5OUT.out`, `RTC5RBF.rbf` und `RTC5DAT.dat` (der gewünschten Version) auf die Festplatte Ihres PCs(1).
  Beachten Sie beim Austauschen einzelner Software-Dateien, dass die Dateien nicht in beliebigen Programmversionen miteinander kombiniert werden können. Die `zip`-Dateien enthalten jeweils eine Text-Datei mit entsprechenden Versions-Informationen.
- Kopieren Sie die Korrekturdatei(en) (`*.CT5`) auf die Festplatte Ihres PCs (bestehende Korrekturdateien können weiterhin genutzt werden; speziell angepasste Korrekturdateien nicht überschreiben!).

SCANLAB empfiehlt, die Dateien in dem Verzeichnis zu speichern, aus dem die Anwendungs-Software gestartet wird. Beim Laden der Korrekturdateien muss dann nur der Dateiname und nicht der Pfad angegeben werden.

(1) MS Visual Studio Laufzeitbibliotheken für C und C++ werden ab der Version DLL 527 nicht mehr benötigt.

5.5 Sichere Einschalt- und Ausschaltreihenfolge

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

(1) Schalten Sie den PC, in den die RTC5-Karte eingebaut ist, ein und starten Sie die Steuerungs-Software.

(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 RTC5-Karte aus.



Achtung!

- Beim Ein- und Ausschalten des PCs können an den Ausgängen der RTC5-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 bzw. vorher wieder ausgeschaltet werden, sonst kann es zu unvorhergesehenen Scan-Bewegungen des Scan-Systems kommen. Der Laser muss immer zuletzt eingeschaltet bzw. zuerst ausgeschaltet werden, sonst besteht die Gefahr, dass der Laserstrahl in eine unkontrollierte Richtung abgelenkt wird.

5.6 Funktionstest



Achtung!

- **Schalten Sie immer zuerst den PC und die Scan-Kopfstromversorgung an, dann erst den Laser. Andernfalls 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 "HPGL Converter-Programm", Seite 22. Dieses Programm erlaubt es, Grafikdateien im HPGL-Format von Hewlett Packard (Vektorgrafik-Plotter-Dateien `*.PLT`) zu laden und an die RTC5 auszugeben.

(1) Kopieren Sie den HPGL-Converter und die mitgelieferte HPGL-Demo-Datei in das gleiche Verzeichnis wie die RTC5-DLL, die RTC5-DSP-Programmdatei(en) und die Korrekturdatei(en).

(2) Starten Sie den HPGL-Converter.

(3) Wählen Sie unter `Options|Correction` eine Korrekturdatei aus und unter `File|HPGL-File` eine Plotdatei (Demo-Datei).

(4) Mit `Mark|Start Marking` wird die Ausgabe gestartet.

5.7 Anwenderprogramme und Demoprogramme

Die DLLs für RTC5-Anwenderprogramme (`RTC5DLL.dll`, `RTC5DLLx64.dll`) unterstützen die RTC5 unter den 32-Bit- und 64-Bit-Microsoft-Betriebssystemen Windows 10, 8, 7, Vista, XP SP2, XP SP3. Die DLLs stellen alle benötigten Funktionen für den Betrieb der RTC5 zur Verfügung.

Durch Aufruf der gewünschten DLL (Win32- oder Win64-basiert, s.u.) und Initialisierung ihrer Funktionen können die Funktionen in Anwenderprogramme eingebunden werden. Die Programmierung von Anwenderprogrammen wird in Kapitel 6 ausführlich beschrieben. Kapitel 6.2, Seite 66 beschreibt u.a. die Aufrufkonvention der DLL und zeigt, wie Funktionen und Prozeduren der DLL in Anwenderprogrammen initialisiert werden, wenn die Programme in Pascal, C, C++ oder C# geschrieben werden.

Die 64-Bit-Variante des RTC5-Treibers unterstützt auf einem 64-Bit-Betriebssystem sowohl Funktions-Aufrufe aus Win64-basierten als auch aus Win32-basierten Anwenderprogrammen(1).

Bereits existierende Win32-basierte Anwenderprogramme zur RTC5 sind damit auch auf 64-Bit-Systemen ausführbar, wenn die im Paket enthaltene Win32-basierte `RTC5DLL.dll` verwendet wird. Für Win64-basierte Anwenderprogramme ist im Paket die Win64-basierte DLL `RTC5DLLx64.dll` enthalten. Falls ein Anwenderprogramm zu dieser DLL Verbindung durch implizites Linken aufnehmen soll, ist sie mit der Win64-basierten Import-Library `RTC5DLLx64.LIB` zu linken.

Zum einfacheren Einstieg in die Programmierung von Anwenderprogrammen sind auf Seite 71, Seite 100, und Seite 109 Beispiel-Codes aufgelistet und im RTC5-Software-Paket eine Reihe von Demoprogrammen enthalten (siehe Seite 647).

Vorsicht!

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

(1) Der Treiber unterscheidet, ob ein Funktionsaufruf von einer 32-Bit oder einer 64-Bit-DLL kommt.

5.8 Austauschen von RTC5-Karten

Wird eine bereits installierte RTC5-Karte älteren Typs durch eine RTC5-Karte neueren Typs ersetzt, dann sollte auch eine Software-Aktualisierung durchgeführt werden. Dies gilt insbesondere dann, wenn eine RTC5-Karte mit der DSP-Versionsnummer 0 durch eine RTC5-Karte mit der DSP-Versionsnummer 2 ersetzt wird (siehe **get_rtc_version** Bits #16-23). Das passende RTC5-Software-Paket finden Sie auf der mitgelieferten CD. Von der SCANLAB-Website können Sie auch immer das neueste RTC5-Software-Paket herunterladen.

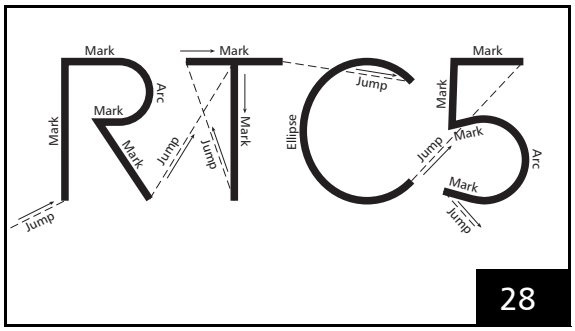
Siehe auch Kapitel 5.3.2 "Tausch von RTC-Karten und Update von RTC-Karten-Treibern", Seite 62.

6 Programmierung von Anwenderprogrammen

6.1 Grundlegendes zum RTC5 Software-Konzept

6.1.1 Ansteuerung von Scan-Systemen und Lasern – Beispiel zur Einführung

Die SCANLAB RTC5 PCI-Karte und die dazugehörige Software sind für die Ansteuerung von Scan-Systemen und Lasern konzipiert. Zur Veranschaulichung des Funktionsprinzips ist in Abb. 28 ein einfaches Beispiel eines Lasermarkierungs-Prozesses(1) dargestellt.



Beispiel zur Lasermarkierung

Der Schriftzug ist aus geraden Liniensegmenten (Vektoren) und Bogensegmenten zusammengesetzt. Zum Abfahren dieser Segmente stellt die RTC5-DLL einen entsprechenden Satz an Jump-, Mark-, Arc- und Ellipse-Befehlen zur Verfügung. Jeder dieser Befehle beschreibt einen einzelnen Vektor bzw. Bogen. Weitere Softwarebefehle ermöglichen die Steuerung des Lasers während des Markierungsprozesses. Die RTC5 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.

Mit entsprechenden Softwarebefehlen ist über die RTC5 auch der aktuelle Zustand eines Scan-Kopfs abrufbar.

Für die Ansteuerung des Lasers stellt die RTC5 verschiedene analoge und digitale Signalausgänge zur freien Verfügung, deren sich der Benutzer eigenverantwortlich entsprechend seiner gewünschten Lasersteuerung bedienen kann.

6.1.2 Listenbefehle und Kontrollbefehle

Der Befehlssatz der RTC5 besteht aus *Kontrollbefehlen* und *Listenbefehlen*.

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

Bevor **Listenbefehle** an die RTC5 übergeben werden, muss mit einem Kontrollbefehl der Input-Pointer definiert werden, an den die nachfolgenden Listenbefehle übergeben werden (dies entspricht dem Eröffnen einer Liste, siehe Kapitel 6.4.1 "Listen laden", Seite 75). Listenbefehle, die dann an die RTC5 übergeben werden, werden nicht sofort ausgeführt, sondern zunächst in einem Listenspeicher zwischengespeichert. Erst nach Starten der Liste werden die Befehle aus dem Listenspeicher in Echtzeit abgearbeitet.

Einige Listenbefehle, sogenannte kurze Listenbefehle, benötigen keinen vollen 10 µs-Takt zur Befehlsausführung, sondern können zusammen mit dem jeweils nachfolgenden Listenbefehl unmittelbar nacheinander innerhalb eines 10 µs-Takts ausgeführt werden, allerdings ohne dass dazwischen ein Kontrollbefehl ausgeführt werden kann. Dadurch verringert sich die gesamte Listenausführungszeit.

Zu den Listenbefehlen zählen sowohl Jump-, Mark- und Bogenbefehle als auch Befehle zur Einstellung verschiedener Scan-Parameter wie der Laserleistung, der Sprung- und Markiergeschwindigkeit.

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

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

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

Alle RTC5-Befehle werden ausführlich in Kapitel 10 "Befehle und Funktionen" beschrieben. In Kapitel 10.1, Seite 250 ist ein tabellarischer Überblick der Befehle zusammengestellt.

6.2 Initialisierung und Programmstart

Zur Verwendung der Befehle und Funktionen der RTC5 in einem Anwenderprogramm

- muss eine vollständige Installation – der Hardware, der RTC5-Treiber und der RTC5-Software – durchgeführt sein (siehe "Installation und Inbetriebnahme", Seite 60),
- muss die gewünschte DLL (Win32- oder Win64-basiert) vom Anwenderprogramm aufgerufen werden (siehe "DLL-Aufruf", Seite 66) und
- müssen die Funktionen und Prozeduren der DLL im Anwenderprogramm initialisiert werden (siehe "Import von Befehlen", Seite 66).

Zu Beginn eines jeden Anwenderprogramms müssen Befehle aufgerufen werden, mit denen

- die DLL für das aufrufende Anwenderprogramm initialisiert wird und die Zugriffsberechtigungen für die installierten RTC5-Karten vergeben werden (siehe "Initialisierung der DLL und Kartenverwaltung", Seite 68) und
- die installierten RTC5-Karten selbst initialisiert und in den gewünschten Grundzustand versetzt werden (siehe "Inbetriebnahme der RTC5-Karte", Seite 69).

Erst danach kann das Anwenderprogramm die gewünschten Befehlslisten auf den Listenspeicher der RTC5 laden und zur Abarbeitung starten.

Die aufgelisteten Schritte werden in den folgenden Abschnitten im Einzelnen erläutert und im letzten Abschnitt dieses Unterkapitels (siehe Seite 71) noch einmal in einem einfachen Beispielprogramm zusammengefasst.

6.2.1 DLL-Aufruf

Damit Befehle und Funktionen der DLL in einem Anwenderprogramm verwendet werden können, muss die gewünschte DLL (Win32- oder Win64-basiert) wie unter "Installation und Inbetriebnahme", Seite 60 beschrieben bereitgestellt werden und vom Anwenderprogramm aufgerufen werden.

Die DLL-Aufrufkonvention heißt **stdcall**. Das Struktur-Alignment ist 4 Byte (für Win32) oder 8 Byte (für Win64).

6.2.2 Import von Befehlen

Um den Import von Befehlen aus der DLL in eine C, C++, C# oder Pascal-Anwendung zu ermöglichen, enthält das RTC5-Software-Paket entsprechende Hilfsdateien. Die folgenden Abschnitte beschreiben, wie diese Dateien in den verschiedenen Softwareumgebungen verwendet werden(1).

Vorsicht!

- Einige der Befehle, die in den Hilfsdateien enthalten sind, können nur in der entsprechenden optionalen Hardwarekonfiguration der RTC5 verwendet werden (siehe Kapitel 2.3 "Optionale Funktionalitäten", Seite 27).
  Achten Sie darauf, nur diejenigen Befehle zu verwenden, die von Ihrer RTC5-Version unterstützt werden. (Die einzelnen Befehle werden in Kapitel 10 beschrieben.)
  Mit dem Befehl **get_rtc_version** können Sie die entsprechenden Informationen bzgl. der gegenwärtig auf Ihrer RTC5-Karte installierten Optionen abrufen.

Pascal

Verwenden Sie die Datei `RTC5Import.pas` als eine Unit und rufen Sie die benötigten RTC5-Befehle auf, z. B.
goto_xy(1000, 2500);

für einen Sprung zur Position 1000, 2500.

(1) Außer beim impliziten Linken unter C/C++ ist die Vorgehensweise unabhängig von der verwendeten DLL (Win32- oder Win64-basiert). Die Formate und Wertebereiche der bei den RTC5-Befehlen zu verwendenden Datentypen (siehe Seite 260) ist aber in jedem Fall zu beachten.

C

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

Implizites Linken

Zum impliziten Linken fügen Sie die Kopfdatei `RTC5impl.h` ein und linken Sie (für Win32-basierte Anwenderprogramme) mit der (C) Import-Bibliothek `RTC5DLL.LIB` (oder für Win64-basierte Anwenderprogramme mit `RTC5DLLx64.LIB`), um eine ausführbare Datei zu erzeugen.

Rufen Sie die benötigten RTC5-Befehle auf wie

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

für einen Sprung zur Position 1000, 2500.

Explizites Linken

Zum expliziten Linken fügen Sie die Kopfdatei `RTC5expl.h` ein. Bevor Sie irgendeine RTC5-Funktion aufrufen, initialisieren Sie die DLL durch Aufruf der Funktion `RTC5open` (die in der Datei `RTC5expl.c` definiert wird). Wenn Sie die RTC5-Sitzung beendet haben, dann schließen Sie die DLL durch Aufruf der Funktion `RTC5close` (wird auch in der Datei `RTC5expl.c` definiert). Eine ausführbare Datei erzeugen Sie durch Linken mit der Datei `RTC5EXPL.OBJ`, die Sie aus dem Quellcode der Datei `RTC5expl.c` generieren können.

Rufen Sie die benötigten RTC5-Befehle auf wie

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

für einen Sprung zur Position 1000, 2500.

Vor- und Nachteile von Implizitem und Explizitem Linken

	Implizites Linken	Explizites Linken
Benötigte Datei	`RRTC5impl.h` bzw. `RTC5impl.hpp`, `RTC5DLL.LIB` oder `RTC5DLLx64.LIB`	`RTC5expl.h`, `RTC5expl.c`
Vorteile	einfachste Linkmethode	Das Linken der Anwendung mit einer Import-Bibliothek ist unnötig.
Nachteile	Man muss die Anwendung mit einer Compiler-spezifischen Import-Bibliothek linken.	Man muss die DLL initialisieren (`RTC5open`) und schließen (`RTC5close`).

C++

Wenn Sie unter C++ für die Werterückgabe an Funktionsparameter Referenzen statt Pointer (Zeiger) verwenden wollen, z.B. `UINT&Pos` statt `UINT*Pos`, dann verwenden Sie die Datei `RTC5impl.hpp` statt `RTC5impl.h`. Der Import von Befehlen erfolgt ansonsten wie in C (s.o.).

C#

Zum impliziten Linken der DLL unter C# werden Importdeklarationen der Wrapper-Klasse zur Verfügung gestellt.
Ab DLL 537: Die Wrapper-Klasse unterstützt jetzt auch die Option 'Any CPU' zur simultanen Verwendung für 32-Bit und 64-Bit Programme.

6.2.3 Initialisierung der DLL und Kartenverwaltung(1)

Es können beliebig viele RTC5-Karten in einem Rechner gleichzeitig verwendet werden. Außerdem erlaubt die RTC5-DLL sowohl Multi-Threading als auch Multi-Processing. Es können also auch beliebig viele Anwenderprogramme gleichzeitig verwendet werden.

Ein und dieselbe Karte kann allerdings nicht gleichzeitig von mehreren Anwenderprogrammen verwendet werden. Zugriffsberechtigungen (auch zeitweise) auf die vorhandenen Karten werden von der DLL exklusiv vergeben. Mehrere Threads *eines* Anwenderprogramms können zwar ein und dieselbe Karte verwenden, aber nicht gleichzeitig Befehle senden (die DLL serialisiert die Befehlsaufrufe automatisch).

Die Verwendung unterschiedlicher Karten durch unterschiedliche Anwenderprogramme wird von der DLL über eine DLL-interne Kartenverwaltung koordiniert. Die Kartenverwaltung wird durch den Befehl **init_rtc5_dll** initiiert. Dieser Befehl muss – auch wenn nur *eine* RTC5-Karte von nur *einem* Anwenderprogramm verwendet werden soll – zu Beginn eines jeden Anwenderprogramms aufgerufen werden, damit überhaupt eine RTC5-Karte ansprechbar ist.

Der Befehl **init_rtc5_dll**:

- sucht nach sämtlichen vorhandenen RTC5-Karten
- baut die entsprechende Verwaltung auf
- vergibt dem Anwenderprogramm automatisch die Zugriffsberechtigung für die gefundenen Karten, sofern die Zugriffsberechtigung nicht bereits an ein anderes Anwenderprogramm vergeben wurde
- vergibt DLL-interne Nummern für alle gefundenen RTC5-Karten (wichtig für Multi-Board-Befehle)
- macht eine der Karten zur aktiven Karte und damit zum Ziel für Nicht-Multi-Board-Befehle

Für eine genaue Befehlsbeschreibung siehe Seite 348.

Weitere Befehle ermöglichen eine nachträgliche Änderung der Zugriffsberechtigungen und einen Wechsel der aktiven Karte. Die Verwendung mehrerer Karten wird in Kapitel 6.6 "Verwendung mehrerer RTC5-Karten in einem Rechner", Seite 92, die Verwendung durch mehrere Anwenderprogramme in Kapitel 6.7 "Verwendung durch mehrere Anwenderprogramme", Seite 96 beschrieben.

Nach der Initialisierung der DLL mit **init_rtc5_dll** kann zusätzlich zwischen zwei DLL-Betriebsmodi ausgewählt werden (siehe **set_rtc4_mode** und **set_rtc5_mode**). Standardmäßig ist der RTC5-Modus voreingestellt. Der RTC4-Kompatibilitätsmodus wird zur Verfügung gestellt, damit Anwenderprogramme, die für die RTC4 geschrieben sind, auch von der RTC5 (zu großen Teilen) ohne Programmcode-Änderung abgearbeitet werden können. Voraussetzung hierbei ist allerdings, dass der Programmcode nur RTC4-Befehle enthält, die mit unveränderter Funktionalität auch als RTC5-Befehle existieren. In der Befehlsauflistung dieses Handbuchs (ab Seite 261) wird gegebenenfalls jeweils in der Zeile "RTC4→ RTC5" auf entsprechende Änderungen hingewiesen (siehe auch Abschnitt "Erhöhte Parameter-Auflösung", Seite 33).

(1) Alle RTC5-Befehle können sowohl unter 32-Bit als auch unter 64-Bit Windows identisch verwendet werden. Die Formate und Wertebereiche der zu verwendenden Datentypen (siehe Seite 260) ist aber in jedem Fall zu beachten.

6.2.4 Inbetriebnahme der RTC5-Karte

Am Anfang eines jeden RTC5 Anwenderprogramms sollten – nach erfolgter Initialisierung der DLL (siehe vorhergehender Abschnitt) – die folgenden Schritte ausgeführt werden, um die RTC5 in Betrieb zu nehmen:

Initialisierung der Karte

(1) Rufen Sie den Befehl **load_program_file** auf. Dieser Befehl setzt die RTC5 zurück (dies entspricht einem Hardware-Reset), initialisiert die Speicherkonfiguration (in der Standard-Konfiguration), führt einen DSP-Speicher-Check aus, lädt die Firmware (`RTC5RBF.rbf`), die Programmdatei `RTC5OUT.out` und eine binäre Hilfsdatei (`RTC5DAT.dat`) und startet den Signalprozessor. Nach Ausführung des Befehls ist die Position des Scanners automatisch auf den Nullpunkt (0|0) eingestellt (der Laserfokus ist also im Zentrum des Bildfeldes) und die Lasersteuerung ist *deaktiviert*. Die RTC5-DLL und die Programmdateien `RTC5OUT.out`, `RTC5RBF.rbf` und `RTC5DAT.dat` können nicht in beliebigen Programmversionen miteinander kombiniert werden. Der Befehl **load_program_file** führt eine entsprechende Versionsprüfung durch. Liegt ein Versionsfehler vor (Fehlercode 7 und **get_last_error**-Returncode `RTC5_VERSION_MISMATCH`), wird die Karte wie mit **release_rtc** freigegeben und damit für weitere Befehle gesperrt, außer für solche, die keine Zugriffsberechtigung erfordern. RTC5-Karten können nur in Betrieb genommen werden, wenn alle Software-Dateien in einer zulässigen Versionskombination bereitgestellt werden (siehe Kapitel 5.4 "Installation der RTC5-Software", Seite 62). Wenn die Versionsprüfung fehlschlägt und die Karte nicht von einem anderen Anwenderprogramm belegt ist, kann mit dem Multi-Boardbefehl **n_load_program_file** eine korrekte Programmversion geladen und anschließend die Karte mit **select_rtc** oder **acquire_rtc** in Besitz genommen werden.
Wenn mehrere RTC5-Karten als Master und Slave miteinander verbunden sind, dann muss auf allen Karten der Befehl **load_program_file** erteilt worden sein und sollte eine Synchronisierung der Takt-Phasen durchgeführt werden, bevor einzelne Karten mit weiteren Befehlen initialisiert und betrieben werden (siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93).

Konfiguration der Karte

(2) Konfigurieren Sie falls nötig mit **config_list** den RTC5-Listenspeicher. Der Listenspeicher ist von SCANLAB bei Auslieferung standardmäßig so vorkonfiguriert, dass die Speicherbereiche "Liste 1" und "Liste 2" jeweils 4.000 Listenbefehle aufnehmen können. Der geschützte Speicherbereich "Liste 3" umfasst die restlichen 1.040.576 der 2^{20} Speicherplätze.

Initialisierung der Scan-System-Ansteuerung

(3) Laden Sie mit **load_correction_file** die benötigten Korrekturdatei(en) auf die RTC5 (das Laden einer Korrekturtabelle kann vor oder nach **load_program_file**, sollte aber unbedingt vor **select_cor_table** erfolgen(1); es sollte zumindest eine 1:1-Korrekturdatei geladen werden). Informationen über die Verwendung mehrerer verschiedener Korrekturtabellen finden Sie in Kapitel 8.5, Seite 192.

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

Die vorgesehene Bildfeldkorrektur wird allerdings erst durch einen nachfolgenden **select_cor_table**-, **select_cor_table_list**-, Sprung- oder Markierbefehl wirksam.

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

(6) Laden Sie falls nötig (mit **load_varpolydelay**) eine Tabelle für das variable Polygon-Delay.

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

(1) Die Korrekturtabellen Nummer 3 und 4 werden durch **load_program_file** gelöscht.

(2) Ab Version DLL 521, OUT 521 ruft **load_correction_file** (wenn der Befehl nach **load_program_file** erfolgt) nach dem Laden einer Korrekturtabelle automatisch **select_cor_table** auf (siehe Hinweise auf Seite 138).

Initialisierung der Lasersteuerung

(7) Wählen Sie (mit **set_laser_mode**) eine Laser-Version.

(8) Setzen Sie (mit **set_laser_control**) die Laserausgangssignale passend zu Ihrem Lasersystem.

(9) Legen Sie (mit **set_firstpulse_killer**) die Länge des FirstPulseKiller-Signals fest (nur in der YAG-Version).

(10) Legen Sie (mit **set_qswitch_delay**) das Delay für das Q-Switch-Signal fest (nur bei den YAG-Versionen, insbesondere bei der YAG-Version 5).

(11) Legen Sie (mit **set_standby**) Werte für die Standby-Pulse fest (üblicherweise nur in der CO_2-Version).

(12) Geben Sie gegebenenfalls die Lasersteuer-signale frei (siehe **enable_laser**), sofern 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", Seite 109 oder Abschnitt ""Laser active"-Signale", Seite 145.

Laden und Abarbeiten von Listen

(13) Laden Sie die Liste(n).

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

Vorsicht!

- 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

Der grundsätzliche Aufbau eines Anwenderprogrammes zur Initialisierung der DLL und der RTC5 ist im Folgenden beispielhaft als C-Quellcode für eine Konsolenanwendung (Win32-Umgebung) dargestellt (für komplette Demoprogramme siehe Seite 647).

Benötigte Dateien: `RTC5impl.h`, `RTC5DLL.LIB` (bei implizitem Linken) bzw. `RTC5expl.h`, `RTC5expl.c` (bei explizitem Linken). Zusätzlich muss die DLL vom Programm aufgerufen werden (siehe Seite 66). Falls das Betriebssystem die `RTC5DLL.dll` beim Programm-Startup nicht findet, antwortet es mit einer entsprechenden Fehlermeldung und beendet das Programm.

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

// RTC5 Kopfdatei zum impliziten Linken zur RTC5DLL.dll (linken Sie zusätzlich mit der (Visual C++) Import-Bibliothek
// RTC5DLL.LIB, um eine ausführbare Datei zu erzeugen):
#include "RTC5impl.h"
// Alternativ: RTC5 Kopfdatei zum expliziten Linken zur RTC5DLL.dll (linken Sie zusätzlich mit der Datei RTC5EXPL.OBJ, die
// Sie aus dem Quellcode der Datei RTC5expl.c generieren können, um eine ausführbare Datei zu erzeugen):
//#include "RTC5expl.h"

void __cdecl main( void*, void* )
{

   // nur bei explizitem Linken:
   // if ( RTC5open() ) // Fehler detektiert (RTC5open gibt 0 zurück, wenn kein Fehler auftritt)
   // {
   // printf( "Error: RTC5DLL.dll not found\n" );
   // terminateDLL();
   // return;
   // }

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

   UINT  ErrorCode;

   // Initialisierung der DLL (der folgende Befehl sollte als erster RTC5 Befehl aufgerufen werden)
   ErrorCode = init_rtc5_dll();

   // Direkt auf den init_rtc5_dll-Befehl sollte ein Programmteil eingebaut werden, mit dem ein Fehler während der
   // Initialisierung aufgefangen wird, z. B. für den Fall, dass die gewünschte RTC5-Karte nicht gefunden wird, wenn der
   // Zugriff verweigert wird oder wenn ein weiterer Fehler (z. B. ein VersionMismatch-Fehler auftritt. Ein entsprechender
   // Beispielcode ist auf Kapitel 6.8.3 "Beispiel-Code", Seite 100 aufgeführt.

   // Initialisierung der RTC5:
   // - Auswählen der RTC5-Karte Nummer 1 als die aktive RTC5-Karte diesem Anwenderprogramm
   // - Falls gewünscht: Auswählen des RTC4-Kompatibilitätsmodus als Betriebsmodus (Default: RTC5-Modus)
   // - Stoppen einer eventuellen Listenausführung auf der RTC5-Karte Nummer 1 falls diese Karte zuvor durch ein anderes
   //   Anwenderprogramm verwendet wurde, könnte noch eine Liste laufen (ausgeführt werden). load_program_file und
   //   load_correction_file könnten dann nicht ausgeführt werden).
   // - Aufrufen von load_program_file zum Resetten 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", Seite 100)
   // - Zurücksetzen aller vorherigen Fehlercodes (stop_execution könnte einen RTC5_TIMEOUT- oder RTC5_BUSY-Fehler
   //   verursacht haben)
   // - Konfigurieren des RTC5-Listenspeichers, Default: 4000 Speicherplätze für Liste 1, 4000 für Liste 2).
```

```
   (void) select_rtc( 1 );
   set_rtc4_mode();
   stop_execution();
   ErrorCode = load_program_file( 0 ); // pPath = 0: Pfad des aktuellen Arbeitsverzeichnisses
   if ( ErrorCode )
   {
      printf( "Program file loading error: %d\n", ErrorCode );
      free_rtc5_dll();
      return;
   }
   reset_error( -1 );
   config_list( 4000, 4000 );
   // Nachfolgend kann nun der Programmcode zur Definition des Laser-Scanprozesses eingefügt werden.
   // Ein Beispielcode ist auf Seite 109 aufgeführt.

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

void terminateDLL()
{
   printf( "- Press any key to shut down \n" );
   while( !kbhit() );
   (void) getch();
   printf( "\n" );
   // Schließen der RTC5.DLL
   free_rtc5_dll();
   // nur bei explizitem Linken:
   // RTC5close();
}
```

6.3 Listenspeicher

Der Listenspeicher dient zum Zwischenspeichern von Listenbefehlen. Bevor Listenbefehle an die RTC5 übergeben werden, muss mit einem Kontrollbefehl der Input-Pointer definiert werden, an den die nachfolgenden Listenbefehle übergeben werden (dies entspricht dem Eröffnen einer Liste, siehe "Listen laden", Seite 75). Durch weitere Kontrollbefehle kann das Abarbeiten der übergebenen Listenbefehle gestartet werden.

6.3.1 Listen und geschützter Speicherbereich

Der Listenspeicher der RTC5 umfasst 1 M (= 1.048.576 = 2^{20}) Speicherpositionen. Der Speicher ist prinzipiell in drei Bereiche einstellbarer Größe eingeteilt (konfiguriert):

- zwei Listen-Speicherbereiche ("Liste 1" und "Liste 2"), im Folgenden auch kurz mit "Listen" benannt, und
- ein dritter "geschützter Speicherbereich" ("Liste 3"), der vor einem versehentlichen Überschreiben durch Laden von normalen Befehlslisten geschützt ist.

Listen

Die beiden Listenbereiche ("Liste 1" und "Liste 2") können prinzipiell identisch verwendet werden wie die beiden Listenspeicher der RTC4-, RTC3- oder RTC2-Karten, etwa für das kontinuierliche Laden und Abarbeiten von Befehlslisten.

Bei der RTC5 steht allerdings ein höherer Speicherbereich zur Verfügung und die Größe der einzelnen Speicherbereiche ist frei konfigurierbar (siehe "Konfiguration des Listenspeichers", Seite 74).

Der Befehlsumfang für das Listen-Handling wurde für die RTC5 erweitert und wird ausführlich in Kapitel 6.4 "Listen-Handling", Seite 75 beschrieben.

Geschützter Speicherbereich

Der dritte Speicherbereich ("Liste 3") – neu bei der RTC5 – ist zur geschützten Speicherung immer wieder benötigter Listenbefehlssequenzen in Form von Unterprogrammen oder Zeichensatzdefinitionen vorgesehen. Er ist vor einem versehentlichen Überschreiben durch Laden von normalen Befehlslisten geschützt.

Prinzipiell gibt es dabei zwei Möglichkeiten, diese Schutzfunktion zu nutzen:

(1) Es werden Unterprogramme – so wie es auch bereits bei den Vorgängerkarten der RTC5 möglich war – in die oberen Speicherplätze des Listenbereichs geschrieben und diese Speicherplätze – diese Möglichkeit ist neu bei der RTC5 – nachträglich dem geschützten Speicherbereich zugewiesen. Der Aufruf eines solchen Unterprogramms erfolgt – sowohl zu Beginn im Listenbereich als auch später im geschützten Speicherbereich – unter Angabe der absoluten Speicherposition.

(2) Unterprogramme bzw. Zeichensatzdefinitionen werden über spezielle Befehle als indizierte Unterprogramme bzw. Zeichensätze direkt in den geschützten Speicherbereich geladen. Ihr Aufruf erfolgt dann unter Angabe des entsprechenden Indexes. Dies macht u. U. die spätere Nutzung durch den Benutzer leichter. Indizierte Zeichensätze können aber auch zusätzlich beispielsweise mit dem neu definierten **mark_text**-Befehl direkt zum Markieren von Texten verwendet werden.

SCANLAB empfiehlt dringend, die beiden Nutzungsarten *nicht* miteinander zu vermischen. Ansonsten kann es gegebenenfalls auch im geschützten Speicherbereich zu einem unbeabsichtigten Datenverlust durch Überschreiben kommen.

Damit man nicht dauerhaft auf die eine oder die andere Nutzungsart festgelegt ist, sind im Befehlsumfang der RTC5 entsprechende Konvertierungs-Befehle enthalten.

Die Definition von Unterprogrammen und Zeichensätzen, deren Verwaltung und die Möglichkeiten zur nachträglichen Konvertierung sind ausführlich in Kapitel 6.5 "Strukturierte Programmierung", Seite 81 beschrieben.

6.3.2 Konfiguration des Listenspeichers

Aus Kompatibilitätsgründen ist der RTC5-Listenspeicher von SCANLAB bei Auslieferung standardmäßig so vorkonfiguriert, dass die Speicherbereiche “Liste 1” und “Liste 2” jeweils 4.000 Listenbefehle aufnehmen können.

Der geschützte Speicherbereich “Liste 3” umfasst dann die restlichen 1.040.576 der 2^{20} Speicherplätze (siehe Konfiguration (a) in Abb. 29). Benutzer können mit dem Befehl **config_list** die Speicherbereiche aber auch umkonfigurieren.

Benötigt ein Anwenderprogramm z. B. nur *eine* Liste, so kann der gesamte RTC5-Speicher auch wie ein einzelner Listenspeicher mit einer Gesamtkapazität von 2^{20} Eintragungen behandelt werden (siehe Konfiguration (e) in Abb. 29). In diesem Fall kann eine Liste (“Liste 1”) mit bis zu 2^{20} Befehlen geladen werden.

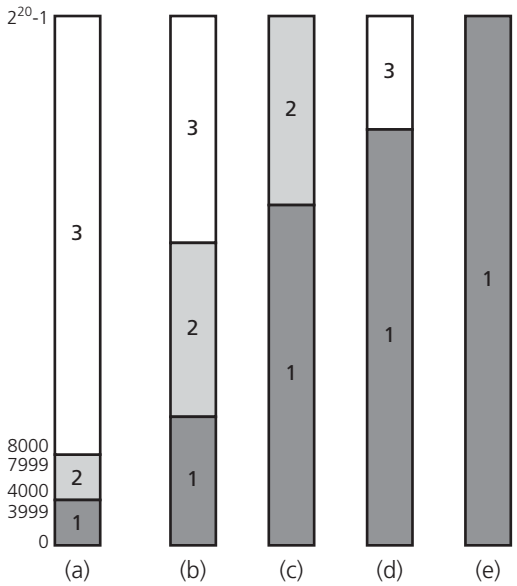
Generell werden bei der Konfiguration die Speicherplätze mit den niedrigeren Speicherplatznummern “Liste 1”, die mittleren Speicherplätze “Liste 2” und die höheren Speicherplätze “Liste 3” zugeordnet.

Bei der Konfiguration der Speicherbereiche ist zu beachten, dass “Liste 1” mindestens *einen* Speicherplatz enthalten muss und dass die Summe der Speicherplätze von “Liste 1” und “Liste 2” die Zahl 2^{20} nicht überschreiten kann. Ansonsten können die Größen von “Liste 1” und “Liste 2” beliebig konfiguriert werden (es kann also z. B. auch eine Speicherplatzgröße 0 für “Liste 2” gewählt werden, siehe Konfiguration (d) oder (e) in Abb. 29). Dem geschützten Speicherbereich “Liste 3” werden bei jeder Konfiguration automatisch die restlichen, nicht an “Liste 1” oder “Liste 2” vergebenen Speicherplätze zugeteilt.

Bei der Konfiguration erfolgt keine Änderung des Speicherinhalts. Ein wiederholter Aufruf mit verschiedenen Parametern ist daher unschädlich.

Beim nachträglichen Ändern der Konfiguration sollte man aber u. a. das Folgende beachten:

- Listengrenzen sollten nicht so verschoben werden, dass sie hinterher mitten durch ein Unterprogramm verlaufen.
- Wird ein Bereich von “Liste 3” dem Listenbereich (“Liste 1” oder “Liste 2”) zugeordnet, so wird der Schutz für diesen Bereich aufgehoben.
- Legale Sprungadressen in Sprungbefehlen werden u. U. durch Ändern der Konfiguration illegal (siehe “Sprünge”, Seite 89).
- Eventuell muss nach einer Vergrößerung des geschützten Speicherbereichs (“Liste 3”) zusätzlich eine Defragmentierung durchgeführt werden, damit der neu zugewiesene Speicherbereich auch nutzbar ist (siehe “Indexverwaltung und Defragmentierung” auf Seite 85).



1 “Liste 1” (Listen-Speicherbereich)
2 “Liste 2” (Listen-Speicherbereich)
3 “Liste 3” (geschützter Speicherbereich)
(a) Standard-Konfiguration (links die Nummern der begrenzenden Speicherplätze der drei Speicherbereiche):
`config_list(4000, 4000)`
(b) `config_list(Mem1, Mem2)`
mit `Mem1` > 0, `Mem2` ≤ 2^{20} - `Mem1`
(c) `config_list(Mem1, -1)` mit `Mem1` > 0
(d) `config_list(Mem1, 0)` mit `Mem1` > 0
(e) `config_list(-1, x)` mit `x` beliebig

29

Beispiele für zulässige Listenspeicher-Konfigurationen

6.4 Listen-Handling

Die beiden Listenbereiche (“Liste 1” und “Liste 2”) dienen als Zwischenspeicher für ein kontinuierliches Laden und Abarbeiten von Listenbefehlen. In diesem Kapitel werden die Befehle beschrieben, mit denen dieser Datentransfer gesteuert wird.

6.4.1 Listen laden

Listen werden mit den Befehlen **set_start_list_pos**, **load_list** oder weiteren Kontrollbefehlen (siehe unten) zum Beschreiben mit Listenbefehlen freigegeben. Dabei wird für die ausgewählte Liste ein Input-Pointer definiert, an den die nachfolgenden Listenbefehle übergeben werden.

Listen sind in sich geschlossene Blöcke von Speicherplätzen für Listenbefehle, d.h. wird beim Laden einer Liste das Ende der Liste erreicht und der Input-Pointer nicht auf die jeweils andere Liste gesetzt, so wird der Input-Pointer automatisch auf den Anfang der aktuellen Liste zurückgesetzt und das Laden wird dort fortgesetzt. Es erfolgt niemals ein automatischer Wechsel des Input-Pointers in eine andere Liste, insbesondere nicht in den geschützten Speicherbereich (“Liste 3”).

Prinzipiell werden beim Laden von Listenbefehlen auf die Listenspeicherplätze die dort zuvor gespeicherten Listenbefehle überschrieben, auch wenn diese noch nicht abgearbeitet und sogar wenn sie momentan ausgeführt werden. Man sollte sich vergewissern, dass keine Befehle überschrieben werden, die noch in der Anwendung gebraucht werden (s.u.).

Der PCI-Transfer der Listenbefehle in den 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 die Befehle **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** erteilt werden, wird automatisch ein Flush ausgeführt, damit die gepufferten Listenbefehle in den Listenspeicher transferiert werden. 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

Durch den Befehl **set_start_list(_1/_2)** wird der Input-Pointer an den Anfang der ausgewählten Liste und durch die Befehle **set_start_list_pos** oder **set_input_pointer** an die spezifizierte Adresse der ausgewählten Liste gesetzt. Der jeweils nächste Listenbefehl wird – unabhängig vom aktuellen Status der ausgewählten Liste (siehe “Listenstatus”, Seite 76) – an diese Adresse geschrieben.

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

Laden mit Schutzfunktion

Durch den Befehl **load_list** wird der Input-Pointer zur Initiierung des Lade-Vorgangs wie beim **set_start_list_pos**-Befehl an die spezifizierte Adresse der ausgewählten Liste gesetzt, allerdings nur dann, wenn die ausgewählte Liste nicht aktuell in Verwendung ist. Alternativ kann man den Input-Pointer mit **load_list** auch einfach in eine gerade nicht aktive bzw. bereits abgearbeitete Liste setzen lassen (die jeweils geeignete Liste wird dabei von der RTC5 automatisch ermittelt).

Der Returnwert des **load_list**-Befehls gibt zusätzlich darüber Auskunft, ob bzw. in welcher Liste ein Lade-Vorgang erfolgreich initiiert wurde. Andernfalls wird der Input-Pointer auf eine illegale Position gesetzt. Im letzteren Fall ist so lange keine Eingabe von weiteren Listenbefehlen möglich, bis der Input-Pointer (z.B. mit Hilfe eines erneuten **load_list**-Befehls mit positivem Resultat oder mit Hilfe eines **set_start_list_pos**-Befehls) wieder gültig gesetzt wird.

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

Der **load_list**-Befehl hilft u.a. beim alternierenden Listenwechsel, um gezielt auf das Abarbeiten einer Liste zu warten (siehe “Alternierender Listenwechsel”, Seite 80).

Abschließen von Listen

Befehlslisten können, müssen aber nicht notwendigerweise mit einem **set_end_of_list**-Befehl abgeschlossen werden.

Beim Abarbeiten von nicht abgeschlossenen Befehlslisten wird allerdings nach Erreichen der letztmöglichen Position des Output-Pointers 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 dem **set_end_of_list**-Befehl abgeschlossen wurde (siehe "Automatischer Listenwechsel", Seite 79).

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

6.4.2 Listenstatus

Listen werden – abhängig vom Status der Befehlseingabe und Befehlsausgabe – bestimmte Statuswerte zugeordnet. Die aktuellen Listenstatuswerte können mit dem Kontrollbefehl **read_status** – für beide Listen separat – abgefragt werden.

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

Hinweise

- Wenn der Listenstatus während der Abarbeitung eines Unterprogramms im geschützten Speicherbereich (“Liste 3”) abgefragt wird, so wird der Status derjenigen Liste (“Liste 1” oder “Liste 2”) zurückgegeben, in der sich der Output-Pointer 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** bzw. einem externen Listen-Stopp wird für beide Listen der Status USED 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 Listen-Stopp) einer Liste `ListNo` wieder explizit den Status USED zuweisen wollen, laden Sie an einer unbenutzten Stelle `Pos` dieser Liste durch **set_start_list_pos**(`ListNo`, `Pos`) einen **set_end_of_list**-Befehl und führen Sie **execute_list_pos**(`ListNo`, `Pos`) aus. Wenn gerade keine andere Liste aktiv war, hat die Liste `ListNo` nun den Status USED.
- Bei der Interpretation der mit **read_status** rückgelesenen Statuswerte sollte immer der jeweils programmierte Lade- bzw. Ausführungsvorgang der Listen berücksichtigt werden (siehe Befehlsbeschreibung).

6.4.3 Listenausführungsstatus

Zusätzlich zu den Listenstatuswerten werden drei Statuswerte bereitgestellt, die darüber Auskunft geben, ob eine der beiden Listen aktuell ausgeführt wird (BUSY), ob deren Ausführung gerade unterbrochen ist (PAUSED), bzw. ob die RTC5 mit Kontrollbefehlen außerhalb der Listenausführung beschäftigt ist (INTERNAL-BUSY). Diese Statuswerte können mit dem Kontrollbefehl **get_status** ausgelesen werden.

- Der BUSY-Status ist gesetzt, wenn aktuell eine Liste abgearbeitet wird oder wenn eine Liste mit den Kontrollbefehlen **pause_list** oder **stop_list** angehalten wurde. Der BUSY-Status ist dagegen nicht gesetzt, wenn eine Liste mit dem Listenbefehl **set_wait** angehalten wurde (und wird erst durch einen anschließenden **release_wait**-Befehl erneut gesetzt).
- Der PAUSED-Status ist gesetzt, wenn eine Liste mit **pause_list**, **stop_list** oder **set_wait** angehalten wurde. Durch einen anschließenden **restart_list**- oder **release_wait**-Befehl wird der PAUSED-Status wieder rückgesetzt (siehe auch “Unterbrechen von Listen zur Synchronisation der Ausgabe”, Seite 79).
- Der INTERNAL-BUSY-Status ist gesetzt, wenn die RTC5 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). INTERNAL-BUSY-Status und BUSY-Status können nicht gleichzeitig gesetzt sein.

Hinweise

- Der BUSY-Status ist auch als BUSY OUT-Signal an der 15-poligen Sub-D-LASER-Buchse (siehe Abb. 10), an der EXTENSION 1-Stiftleiste (siehe Abb. 15) und an der MARKING ON THE FLY Stiftleiste (siehe Abb. 18) verfügbar.
- Einige Kontrollbefehle werden ignoriert (nicht ausgeführt), wenn der BUSY-Status und/oder INTERNAL-BUSY-Status gesetzt sind (z. B. **auto_cal**, **goto_xy**, **load_correction_file**) oder werden bei gesetztem INTERNAL-BUSY-Status erst dann verzögert ausgeführt, nachdem der INTERNAL-BUSY-Status 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 entweder mit dem Kontrollbefehl **execute_list** oder durch ein externes Startsignal (siehe Abschnitt ”Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation”, Seite 239) gestartet werden.

Zum Starten der Listenausgabe an einer bestimmten Adresse kann der Befehl **execute_at_pointer** verwendet werden. Wenn ein externes Startsignal (siehe Abschnitt ”Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation”, Seite 239) verwendet wird, ermöglicht der Befehl **set_extstartpos_list** die Definition einer Startadresse für den externen Listen-Start.

Die RTC5 startet die Ausgabe sofort. Während die Befehle der Liste dann der Reihe nach im 10 µs-Takt ausgeführt werden, können weiterhin Kontrollbefehle an die RTC5 übergeben werden, die dann unmittelbar ausgeführt werden, 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). Gestartet werden kann die zweite Liste allerdings erst, nachdem die erste Liste abgearbeitet wurde. Während der Abarbeitung einer Liste wird der Befehl **execute_list** oder ein externes Startsignal ignoriert.

Die Ausführung einer Liste kann aber auch – etwa zur Implementierung eines Notstopps – abgebrochen werden. Sobald der Befehl **stop_execution** aufgerufen oder ein externes Stoppsignal an die RTC5 transferiert wird, werden die aktuell ausgeführte Liste sofort abgebrochen und die “Laser active”-Lasersteuersignale abgeschaltet (aber nicht deaktiviert). Mit dem **range_checking**-Befehl kann das Abarbeiten einer Liste (wie mit **stop_execution**) ebenso automatisch beendet werden.

Wenn beim Abarbeiten einer Liste das Ende der Liste erreicht wird, ohne dass ein **set_end_of_list**-Befehl gefunden wurde, wird mit dem Beginn der aktuellen Liste fortgefahren. Das wiederholt sich so lange, bis der Befehl **stop_execution** aufgerufen oder ein externes Stoppsignal an die RTC5 transferiert wird.

Falls andererseits beim Abarbeiten einer Liste ein **set_end_of_list**-Befehl erreicht wird, so endet die Ausführung der Liste, es sei denn es wurde zuvor ein **auto_change**-, **auto_change_pos**- oder **start_loop**-Befehl erteilt. In letzterem Fall erfolgt ein Listenwechsel (siehe ”Automatischer Listenwechsel”, Seite 79). Der Wechsel erfolgt ausschließlich bei Erreichen eines **set_end_of_list**-Befehls.

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 Ausführung von Listen auch von externen Signalen gestartet und gestoppt werden kann, verfügt die RTC5 über entsprechende Signal-Eingänge (siehe ”Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation”, Seite 239).

6.4.5 Unterbrechen von Listen zur Synchronisation der Ausgabe

Der Listenbefehl **set_wait** ermöglicht es, nummerierte Unterbrechungspunkte (wait markers) in eine Liste zu integrieren. Erreicht die RTC5 bei der Abarbeitung einer Liste (siehe "Strukturierte Programmierung", Seite 81) einen Unterbrechungspunkt, so wird die Ausgabe der Liste vorübergehend unterbrochen und der Laser ausgeschaltet.

Der Befehl **get_wait_status** überprüft, ob die Ausgabe momentan an einem der Unterbrechungspunkte angehalten wurde. Ist dies der Fall, gibt der Befehl **get_wait_status** die zugehörige Nummer (wait_word), sonst den Wert Null, zurück.

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

Durch den **set_wait**-Befehl wird der (mit **get_status** abrufbare) PAUSED-Status gesetzt und der BUSY-Status rückgesetzt. Das umgekehrte geschieht bei einem anschließenden **release_wait**-Befehl.

Die Unterbrechung einer Liste zu einem beliebigen Zeitpunkt der Ausführung erreicht man mit dem Kontrollbefehl **pause_list** (oder mit dem synonymen Befehl **stop_list**) und eine Wiederaufnahme durch den Befehl **restart_list**. Durch **pause_list** werden die "Laser active"-Lasersteuersignale unterdrückt(1) und das Scan-System bleibt im zuletzt vorgegebenen Zustand stehen – gegebenenfalls auch mitten in einer Mikrovektorisierung. Nach dem anschließenden **restart_list**-Befehl setzt das Scan-System die vorgesehenen Bewegungen (des aktuellen Befehls) fort und die Lasersteuersignale werden gegebenenfalls wieder freigegeben (der Laser wird mit den Standard-Laser-Parametern gestartet, also ohne eventuell vor **pause_list** festgelegte Softstart-Parameter o.ä.).

Durch den **pause_list**-Befehl wird der (mit **get_status** abrufbare) PAUSED-Status gesetzt und durch den **restart_list**-Befehl rückgesetzt. Der BUSY-Status wird von beiden Befehlen unverändert gelassen.

6.4.6 Automatischer Listenwechsel

Wenn bei der Speicherkonfiguration zwei Listenspeicherbereiche ("Liste 1" und "Liste 2") eingerichtet werden (siehe Seite 74), so kann parallel zur Abarbeitung einer Befehlsliste eine zweite Liste geladen werden. Dabei dauert es in der Regel wesentlich weniger lang, eine Liste in den Speicher zu schreiben, als sie auszuführen. Damit ist auch die kontinuierliche Ausführung von beliebig langen, aber in Befehlsblöcke aufgeteilten Listen möglich.

Die kontinuierliche Befehlsausgabe beim notwendigen Wechsel zwischen den beiden Listen kann dabei – wie im Folgenden beschrieben – mit Hilfe einiger Befehle über einen automatischen Listenwechsel sichergestellt werden.

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

Wird die Größe des zweiten Speicherbereichs ("Liste 2") bei der Speicherkonfiguration auf 0 gesetzt, so führen alle automatischen Listenwechsel-Befehle zu einer Wiederholung von "Liste 1" (gegebenenfalls ab der angegebenen Position).

Einmaliger Listenwechsel

Mit den Befehlen **auto_change** oder **auto_change_pos** aktiviert man einen einmaligen, automatischen Listenwechsel zwischen "Liste 1" und "Liste 2". Nach Abarbeiten der gerade aktuellen Liste (wenn der **set_end_of_list**-Befehl erreicht wird) wird dadurch automatisch die Ausgabe der nächsten Liste gestartet. Bei Verwendung des Befehls **auto_change** wird die nächste Liste an der Position 0 gestartet, bei Verwendung des Befehls **auto_change_pos** an der angegebenen Startposition (Listenspeicheradresse als Offset bzgl. des Listenanfangs).

(1) Bei unterdrückten Signalen sind die Lasersignal-Ausgänge im (hochohmigen) tristate-Zustand.

Alternierender Listenwechsel

Alternativ kann die kontinuierliche Befehlsausgabe über eine sich wiederholende alternierende Ausgabe der beiden Listen realisiert werden. Dazu muss der Befehl **start_loop** aufgerufen werden. Dies führt dann zu einer automatischen, kontinuierlichen, alternierenden Ausgabe der beiden Listen, vorausgesetzt, beide Listen werden am Ende jeweils mit dem **set_end_of_list**-Befehl abgeschlossen. Die alternierende Ausgabe wiederholt sich so lange, bis der Befehl **quit_loop** aufgerufen wird. Der Befehl **quit_loop** bricht die kontinuierliche Ausgabe 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 Ausgabe zweier konstanter Listen auch eine ständig inhaltlich wechselnde Liste ausführbar.

Hinweise

- Befehle zum Starten eines einmaligen, automatischen Listenwechsels und der **start_loop**-Befehl zum Starten eines alternierenden Listenwechsels können zu einem beliebigen Zeitpunkt aufgerufen werden.
- 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 der **load_list**-Befehl. Dieser Befehl startet das Laden einer Liste nur dann, wenn die ausgewählte Liste nicht aktuell in Verwendung bzw. bereits abgearbeitet ist (siehe "Listen laden", Seite 75).
- Außerdem sollten die jeweils neuen Listen bereits mindestens zu einem gewissen Anteil geladen sein, bevor der Listenwechsel stattfindet. Andernfalls werden u. U. "alte" Befehle ausgeführt. Der Input-Pointer sollte immer einen ausreichenden Vorsprung gegenüber dem Output-Pointer haben, da der PCI-Transfer der Listenbefehle in den Listenspeicher gepuffert erfolgt (siehe Seite 75) und evtl. sogenannte kurze Listenbefehle verwendet werden können (siehe Seite 65).
- Der Ringpuffer-Modus wird von der RTC5 nicht mehr unterstützt. Effektiv lässt sich dieser Betriebsmodus aber auch mit Hilfe des oben beschriebenen alternierenden Listenwechsels und des **load_list**-Befehls realisieren.

6.5 Strukturierte Programmierung

Zur Unterstützung einer strukturierten Programmierung und Ausgabe von Listenbefehlen enthält der Befehlssatz der RTC5 eine Reihe von Möglichkeiten zur Definition von Unterprogrammen und Zeichensätzen, sowie Listenbefehle zur Steuerung des Programmflusses. Dies wird im Folgenden näher beschrieben.

6.5.1 Unterprogramme

Unterprogramme können als Listenbefehlssequenzen prinzipiell an jeder beliebigen Stelle des Listenspeichers stehen. Sie sollten bevorzugt in das obere Ende des Listenbereichs geschrieben werden (siehe "Geschützter Speicherbereich", Seite 73). Es sollten keine Listengrenzen mitten durch Unterprogramme verlaufen.

Unterprogramme müssen mit einem **list_return**-Befehl abgeschlossen werden.

Es können indizierte und (wie auch bereits bei den Vorgängerkarten der RTC5) nichtindizierte Unterprogramme definiert werden.

Nichtindizierte Unterprogramme

Nichtindizierte Unterprogramme werden wie normale Listenbefehlssequenzen mit den Listen-Lade-Befehlen (siehe "Listen laden", Seite 75) in einen Listenbereich ("Liste 1" oder "Liste 2") geladen, müssen jeweils mit einem **list_return**-Befehl abgeschlossen werden und werden mit dem **list_call**-Befehl unter Angabe der absoluten Speicheradresse aufgerufen.

Nach Abarbeiten eines Unterprogramms einschließlich des abschließenden **list_return**-Befehls wird mit dem der aufrufenden Stelle nachfolgenden Befehl fortgefahren.

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

Hinweise

- Nichtindizierte Unterprogramme können nicht in den geschützten Speicherbereich ("Liste 3") geschrieben werden, können aber nachträglich geschützt werden (siehe auch "Nachträgliches Schützen und Konvertieren von nichtindizierten Unterprogrammen", Seite 84). Damit das Unterprogramm zu diesem Zweck mit dem Befehl **set_sub_pointer** referenziert werden kann, muss allerdings seine Anfangsadresse bekannt sein. Vor dem Laden eines nichtindizierten Unterprogramms in den Listenbereich ("Liste 1" oder "Liste 2") sollte daher immer die Startadresse mit dem Befehl **get_input_pointer** ausgelesen werden.
- Achten Sie darauf, dass im normalen Ablauf einer Liste oder im Rumpf eines Unterprogrammes kein **list_return** angetroffen wird, für das es keinen entsprechenden Unterprogrammaufruf gegeben hat. Bei verschachtelten Unterprogrammaufrufen wird die Integrität der Verschachtelung zerstört. Ist kein Unterprogrammaufruf mehr aktiv, wird die Listenabarbeitung an der absoluten Position 0 fortgesetzt.
  Ab Version OUT 540 gilt: 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 mit dem Befehl fortgefahren, der **list_return** nachfolgt. Es wird nicht mit dem Befehl fortgefahren, der **list_call** nachfolgt (dieser würde das Unterprogramm erneut mit einem unkorrelierten **list_return** abarbeiten).

Indizierte Unterprogramme

Bei der RTC5 können zusätzlich zu nichtindizierten Unterprogrammen auch indizierte Unterprogramme definiert werden. Mit dem **load_sub**-Befehl wird ein durch nachfolgende Listenbefehle definiertes Unterprogramm mit dem gewünschten Index versehen und in den geschützten Speicherbereich (“Liste 3”) geladen. Indizierte Unterprogramme müssen jeweils mit einem **list_return**-Befehl abgeschlossen werden (sonst werden sie nicht gespeichert) und werden mit dem **sub_call**-Befehl unter Angabe des Indexes aufgerufen.

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

Es können 1024 indizierte Unterprogramme gespeichert werden. Die Speicherverwaltung der indizierten Unterprogramme erfolgt intern. Die RTC5 speichert für jedes indizierte Unterprogramm den Index, die zugehörige Speicheradresse (Startposition) und die Anzahl der Listenbefehle in einer internen Verwaltungstabelle. Die Indexverwaltung von indizierten Unterprogrammen durch den Benutzer wird im Abschnitt “Indexverwaltung und Defragmentierung”, Seite 85 beschrieben.

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

Ein indiziertes Unterprogramm wird mit dem **load_sub**-Befehl nur dann gespeichert,

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

Nach einem **list_return**-Befehl wird der Input-Pointer illegal, evtl. direkt nachfolgende Listenbefehle werden nicht mehr gespeichert.

Indizierte Unterprogramme werden mit **load_sub**-Befehlen in der Reihenfolge der Eintragung in “Liste 3” geschrieben. Die kleinstmögliche Startadresse wird dabei – entsprechend der bereits eingetragenen indizierten Unterprogramme (und Zeichensätze) – automatisch ermittelt. Wird ein indiziertes Unterprogramm mit einem bereits bestehenden Index abgespeichert, so wird das vorgehend so indizierte Unterprogramm nicht überschrieben, sondern es bleibt zunächst im geschützten Speicherbereich (“Liste 3”) gespeichert, ist dann aber nicht mehr referenziert und kann daher nicht mehr mit **sub_call** über seinen Index, allerdings weiterhin über seine absolute Speicheradresse aufgerufen werden.

Ob für einen bestimmten Index ein Unterprogramm referenziert ist, kann mit **get_sub_pointer** abgefragt werden. Falls kein Unterprogramm referenziert ist, gibt der Befehl einen Wert von “–1” (d.h. 2^{32}–1) zurück.

Damit ein indiziertes Unterprogramm in den geschützten Speicherbereich geladen werden kann, wenn dieser bereits mit indizierten Unterprogrammen vollgeschrieben ist, muss zuvor der geschützte Speicherbereich mit **config_list** entsprechend vergrößert und mit **save_disk**/**load_disk** defragmentiert werden (das Vergrößern allein ist nicht ausreichend, siehe Seite 85).

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

- Ein **set_end_of_list**-Befehl in einem indizierten Unterprogramm wird durch einen **list_nop**-Befehl ersetzt.
- Absolute Sprünge in den geschützten Speicherbereich (“Liste 3”) und aus dem geschützten Speicherbereich heraus werden beim Abarbeiten ignoriert (siehe ”Sprünge“, Seite 89). Absolute Sprünge können daher nicht in indizierten Unterprogrammen verwendet werden.
- Ebenso ignoriert werden beim Abarbeiten des Unterprogramms relative Sprünge, die über die Grenzen des indizierten Unterprogramms hinausgehen, und Sprungbefehle, die einen Sprung auf sich selbst veranlassen.

Aufruf von Unterprogrammen

Unterprogramme können nicht direkt sondern nur über Listenbefehle aus einer Liste oder aus einem Unterprogramm heraus gestartet werden.

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

“Normaler” Aufruf

Wie bereits oben beschrieben können nichtindizierte Unterprogramme mit dem **list_call**-Befehl unter Angabe der absoluten Speicheradresse aufgerufen werden.

Indizierte Unterprogramme können sowohl mit dem **sub_call**-Befehl unter Angabe des Indexes als auch mit dem **list_call**-Befehl unter Angabe der absoluten Speicheradresse aufgerufen werden.

Hinweise

- Bei der Indexverwaltung oder Defragmentierung (siehe Seite 85) ändert sich u.U. die absolute Speicheradresse eines indizierten Unterprogramms. Vom Aufruf eines indizierten Unterprogramms mit dem **list_call**-Befehl ist daher eher abzuraten.

“AbsCalls”

Bei den Vorgängerkarten der RTC5 dürfen Unterprogramme nur *relative* Vektor- und Bogenbefehle (siehe Seite 103) enthalten, falls die entsprechenden Prozesse an unterschiedlichen Stellen im Bildfeld wiederholt werden sollen.

Bei der RTC5 wird diese Einschränkung durch sogenannte “AbsCalls” aufgehoben. Bei “AbsCalls” wird die momentane Position durch Setzen des Offsets übergeben, der dann bei der nachfolgenden Ausführung von absoluten Vektor- und Bogenbefehlen im Unterprogramm verwendet wird. 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 die Befehle **list_call_abs**, **sub_call_abs**.

Bedingte Aufrufe

Damit das Aufrufen von Unterprogrammen (“normale” Aufrufe und “AbsCalls”) auch von externen Steuersignalen abhängig gemacht werden kann, gibt es die Möglichkeit, mit zusätzlichen Befehlen bedingte Verzweigungen in den Programmablauf einzubauen (siehe “Bedingte Befehlsausführung”, Seite 244).

Aufrufe mit wiederholter Ausführung

Mit den Befehlen **sub_call_repeat** und **sub_call_abs_repeat** kann der Rumpf des Unterprogramms automatisch wiederholt ausgeführt werden.

Nachträgliches Schützen und Konvertieren von nichtindizierten Unterprogrammen

Nichtindizierte Unterprogramme können (direkt) nur in einen Listenbereich (“Liste 1” oder “Liste 2”), also nicht in den geschützten Speicherbereich (“Liste 3”) geschrieben werden. Prinzipiell gibt es aber zwei Methoden sie nachträglich zu schützen:

(1) Änderung der Konfiguration:

Der Speicherbereich des Listenbereichs, in den das nichtindizierte Unterprogramm geschrieben wurde (und sofern dieser auch im oberen Bereich des Speichers liegt), wird mit dem **config_list**-Befehl dem geschützten Speicherbereich zugeordnet. Das auf diese Weise nachträglich geschützte Unterprogramm ist weiterhin ein nichtindiziertes Unterprogramm (mit unveränderter Speicheradresse) und kann nach wie vor mit dem **list_call**-Befehl aufgerufen werden.

Damit nichtindizierte Unterprogramme auf diese Weise auch in den obersten Bereich des Listenspeichers gebracht werden können, müssen die entsprechenden Speicherbereiche gegebenenfalls vor dem Laden des nichtindizierten Unterprogramms mit **config_list** vorübergehend dem Listenbereich zugeordnet werden.

(2) Konvertieren in indizierte Unterprogramme:
Mit dem **set_sub_pointer**-Befehl wird ein nichtindiziertes Unterprogramm referenziert (d.h. dem nichtindizierten Unterprogramm wird ein Index zugewiesen und es wird mit in die Speicherverwaltung der indizierten Unterprogramme aufgenommen). Mit **save_disk**/**load_disk** kann es anschließend als indiziertes Unterprogramm in den geschützten Speicherbereich (“Liste 3”) kopiert werden (mit **save_disk** werden automatisch alle indizierte Unterprogramme und alle über **set_sub_pointer** nachträglich referenzierte nichtindizierte Unterprogramme in Indexreihenfolge binär auf einen PC-Datenträger gespeichert und mit **load_disk** von dort wieder in den geschützten Speicherbereich geschrieben). Bei einem nachfolgenden Aufruf über den Index mit **sub_call** wird dann nicht das Original, sondern die Kopie im geschützten Speicherbereich 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 Speicherbereich und aus dem geschützten Speicherbereich heraus verboten sind (siehe “Sprünge”, Seite 89).

Bei der Konvertierung in indizierte Unterprogramme (Methode 2) müssen alle Programmierregeln für indizierte Unterprogramme beachtet werden (siehe Seite 83).

Wenden Sie möglichst immer nur eine der beiden Methoden an, um zu vermeiden, dass es zu einem unbeabsichtigten Datenverlust im geschützten Speicherbereich 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 deshalb alle evtl. im geschützten Speicherbereich befindlichen nichtindizierten Unterprogramme mit Methode 2 in indizierte Unterprogramme konvertieren, bevor mit dem **load_sub**-Befehl das erste indizierte Unterprogramm definiert wird.

Beachten Sie hierzu auch die folgenden Hinweise:

- Wird bei Methode 1 ein Teil des geschützten Speicherbereichs zum Überschreiben entschützt, so riskiert man dabei, dass indizierte Unterprogramme oder bereits zuvor geschützte Unterprogramme überschrieben werden, die noch benötigt werden.
- Nachträglich nach Methode 1 geschützte nichtindizierte Unterprogramme werden u.U. durch einen nachfolgenden **load_sub**- oder **load_disk**-Befehl unbeabsichtigt überschrieben.
- Der **set_sub_pointer**-Befehl verknüpft den angegebenen Index auch dann mit der angegebenen Startadresse, wenn zuvor bereits ein indiziertes Unterprogramm für diesen Index definiert wurde. Das ursprünglich so indizierte Unterprogramm ist dann nicht mehr referenziert und kann nicht mehr über Index aufgerufen werden.

- Wenn Methode 2 angewendet wird, dann sollte sie auch vollständig angewendet werden. Wenn nämlich nur der **set_sub_pointer**-Befehl 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 geschieht erst, wenn das Unterprogramm nachträglich mit **save_disk**/**load_disk** als indiziertes Unterprogramm in den geschützten Speicherbereich kopiert wird. Auch die Anzahl der Listenbefehle des Unterprogramms wird erst mit **save_disk**/**load_disk** in die interne Verwaltungstabelle eingetragen.
- Nicht referenzierte nichtindizierte Unterprogramme werden von **save_disk** nicht berücksichtigt, also auch nicht die über Methode 1 nachträglich geschützten nichtindizierten Unterprogramme im geschützten Speicherbereich. Im geschützten Speicherbereich werden sie u. U. durch **load_disk** überschrieben.
- Unzulässige Befehle (z. B. **set_end_of_list**) werden bei **save_disk**/**load_disk** automatisch durch **list_nop**-Befehle ersetzt.
- Indizierte Unterprogramme, die mit **copy_dst_src** mehrfach referenziert werden, werden bei einem anschließenden **save_disk**/**load_disk** im Listenspeicher dupliziert. Gegebenenfalls kann es dadurch zu einem Speicherüberlauf im geschützten Speicherbereich kommen.
- Vor Ausführen des **load_disk**-Befehls sollte sichergestellt werden, dass der geschützte Speicherbereich (“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.

Entschützen von Unterprogrammen

Der Schutz für Unterprogramme, die im geschützten Speicherbereich gespeichert sind, wird aufgehoben, indem der entsprechende Speicherbereich mit **config_list** dem Listenbereich (“Liste 1” oder “Liste 2”) zugeordnet wird.

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

Indexverwaltung und Defragmentierung

Damit indizierte Unterprogramme dupliziert, umnummeriert oder ineinander konvertiert werden können, wird der Befehl **copy_dst_src** bereitgestellt. Durch diesen Befehl wird ein indiziertes Unterprogramm ein weiteres Mal referenziert (kann also dann mit einem weiteren Index aufgerufen werden). Der Befehl ändert nur die entsprechenden Einträge in der internen Verwaltungstabelle und lässt den Speicherinhalt des Listenspeichers unverändert.

Überflüssige Referenzierungen (überflüssige Einträge in der internen Verwaltungstabelle) können mit dem **load_sub**-Befehl zusammen mit einem unmittelbar nachfolgenden **list_return**-Befehl gelöscht werden. Auch hier erfolgt das Löschen nur in der internen Verwaltungstabelle und die Listenbefehle des zuvor referenzierten indizierten Unterprogramms bleiben weiterhin im Listenspeicher stehen.

Ob für einen bestimmten Index ein Unterprogramm referenziert ist, kann mit **get_sub_pointer** abgefragt werden. Falls kein Unterprogramm referenziert ist, gibt der Befehl einen Wert von “–1” (d.h. 2^{32}–1) zurück.

Ein echtes Duplikat eines indizierten Unterprogramms im geschützten Speicherbereich des Listenspeichers kann – nachfolgend auf **copy_dst_src** – mit **save_disk**/**load_disk** erstellt werden. Mehrfach referenzierte Unterprogramme werden dadurch entsprechend oft in den Listenspeicher geschrieben. Man sollte darauf achten, dass es dabei zu keinem Speicherüberlauf im geschützten Speicherbereich kommt.

Nicht mehr benötigte Unterprogramme, die im geschützten Speicherbereich auf niedrigeren Speicherplätzen liegen als noch benötigte Unterprogramme, blockieren u.U. Speicherplätze für weitere indizierte Unterprogramme, weil der **load_sub**-Befehl ein neues indiziertes Unterprogramm immer hinter dem referenzierten indizierten Unterprogramm mit der höchsten Speicheradresse einträgt. Aus dem gleichen Grund schafft ein nachträgliches Vergrößern des geschützten Speicherbereichs mit **config_list** zunächst keine weiteren nutzbaren Speicherplätze für das Abspeichern von indizierten Unterprogrammen (der geschützte Speicherbereich kann immer nur nach unten, nicht aber nach oben vergrößert werden).

Abhilfe schafft hier eine Defragmentierung mit **save_disk**/**load_disk**. Dabei werden alle indizierte Unterprogramme und alle über **set_sub_pointer** nachträglich referenzierte nichtindizierte Unterprogramme beginnend bei den untersten Speicherplätzen des geschützten Speicherbereichs in Indexreihenfolge neu in den geschützten Speicherbereich geschrieben. Die hierbei freigebliebenen oberen Speicherplätze können dann für weitere indizierte Unterprogramme genutzt werden.

Hinweise

- Vor Ausführen des **load_disk**-Befehls sollte sichergestellt werden, dass der geschützte Speicherbereich (“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** entreferenziert werden.
- Bei der Indexverwaltung oder Defragmentierung ändert sich u.U. die absolute Speicheradresse eines indizierten Unterprogramms. Vom Aufruf eines indizierten Unterprogramms mit dem **list_call**-Befehl ist daher abzuraten.
- **save_disk** speichert die Unterprogramme ab der referenzierten Adresse bis zum nächstmöglichen **list_return**. Relative Sprünge werden dabei nicht ausgewertet. Verwenden Sie daher keine Verzweigungen zu unterschiedlichen **list_return**-Befehlen, sondern führen Sie die Verzweigungen bereits vor dem einzigen **list_return**-Befehl zusammen.

6.5.2 Zeichensätze und Zeichenfolgen

Für Beschriftungsaufgaben ist es sinnvoll, Befehlslisten in Form von separaten Unterprogrammen im Listenspeicher der RTC5 anzulegen, mit denen festgelegt ist, wie die benötigten Zeichen bzw. Zeichenfolgen vom Scan-System markiert werden sollen.

Um die Verwaltung von Zeichen und Zeichenfolgen sowie deren Verwendung beim Markieren zu erleichtern, bietet die RTC5 die Möglichkeit, indizierte Zeichendefinitionen und Zeichenfolgedefinitionen im geschützten Speicherbereich (“Liste 3”) abzuspeichern und mit einfachen Befehlen aufzurufen.

Indizierte Zeichendefinitionen und Zeichenfolgedefinitionen sind prinzipiell nichts anderes als indizierte Unterprogramme, werden aber mit eigenen Befehlen definiert und aufgerufen und werden von der RTC5 mit Hilfe einer eigenen internen Verwaltungstabelle – separat von indizierten Unterprogrammen – verwaltet.

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

Definition von indizierten Zeichensätzen

Eine Listenbefehlssequenz zur Definition eines Zeichens kann mit dem **load_char**-Befehl als indiziertes Zeichen direkt in den geschützten Speicherbereich geladen werden (die dabei automatisch vergebene Speicheradresse kann mit dem Befehl **get_char_pointer** abgefragt werden). Alternativ kann aber auch ein nichtindiziertes Unterprogramm nachträglich mit dem **set_char_pointer**-Befehl referenziert und dann anschließend mit **save_disk**/**load_disk** als indiziertes Zeichen in den geschützten Speicherbereich kopiert werden.

Die RTC5 verwaltet bis zu 4 Zeichensätze mit jeweils 256 indizierten Zeichen.

Ansonsten gelten die gleichen Regeln wie für indizierte Unterprogramme (siehe “Indizierte Unterprogramme”, Seite 82 und “Nachträgliches Schützen und Konvertieren von nichtindizierten Unterprogrammen”, Seite 84).

Hinweise

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

Aufruf von indizierten Zeichen

Das Markieren eines einzelnen Zeichens wird mit dem **mark_char**-Befehl (bzw. 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 dem Befehl **mark_serial** aufgerufen werden (siehe "Markieren von Uhrzeit, Datum und Seriennummern" auf Seite 170).

Das Markieren ganzer Texte kann mit **mark_text** (bzw. mit dem "AbsCall"-Befehl **mark_text_abs**) gestartet werden. Der gewünschte Zeichensatz kann dabei zuvor mit dem **select_char_set**-Befehl ausgewählt werden. Beim Laden eines **mark_text**-Befehls wird der zu markierende Text, sofern er aus mehr als 12 Zeichen besteht, in Blöcke zu je 12 Zeichen zerteilt und für jeden Block ein separater **mark_text**-Befehl in den Listenspeicher geschrieben. Man sollte darauf achten, dass es dabei zu keinem unerwünschten Speicherüberlauf des jeweiligen Speicherbereichs kommt.

Definition von indizierten Zeichenfolgen für Uhrzeit, Datum und Seriennummer

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 der **mark_char**- und **mark_text**-Befehle – zuvor definierte Zeichensätze verwendet werden.

Eine Listenbefehlssequenz zur Definition einer Zeichenfolge kann mit dem **load_text_table**-Befehl als indizierte Zeichenfolge direkt in den geschützten Speicherbereich geladen werden (die dabei automatisch vergebene Speicheradresse kann mit dem Befehl **get_text_table_pointer** abgefragt werden). Alternativ kann aber auch ein nichtindiziertes Unterprogramm nachträglich mit dem **set_text_table_pointer**-Befehl referenziert und dann anschließend mit **save_disk**/**load_disk** als indizierte Zeichenfolge in den geschützten Speicherbereich kopiert werden.

Die RTC5 verwaltet bis zu 42 indizierte Zeichenfolgen.

Ansonsten gelten die gleichen Regeln wie für indizierte Unterprogramme (siehe "Indizierte Unterprogramme", Seite 82 und "Nachträgliches Schützen und Konvertieren von nichtindizierten Unterprogrammen", Seite 84).

Hinweise

- Der Befehl **set_char_table** ist synonym zum Befehl **set_text_table_pointer**.

Aufruf von indizierten Zeichenfolgen

Indizierte Zeichenfolgen können zur Beschriftung von Uhrzeit, Datum und Seriennummern mit den Befehlen **mark_time**, **mark_date** und **mark_serial** (bzw. mit den "AbsCall"-Befehlen **mark_time_abs**, **mark_date_abs** und **mark_serial_abs**) aufgerufen werden (siehe "Markieren von Uhrzeit, Datum und Seriennummern" auf Seite 170).

Verwalten von indizierten Zeichen und Zeichenfolgen

Die interne Indexverwaltung von indizierten Zeichen und Zeichenfolgen durch die RTC5 erfolgt gemeinsam über eine interne Verwaltungstabelle, aber separat von der internen Indexverwaltung für indizierte Unterprogramme.

Die Indexverwaltung durch den Benutzer (umnummerieren, duplizieren, …) erfolgt analog zur Indexverwaltung von indizierten Unterprogrammen (siehe "Indexverwaltung und Defragmentierung" auf Seite 85) mit den Befehlen **copy_dst_src**, **load_char**, **load_text_table**, **get_char_pointer**, **get_text_table_pointer** und **save_disk**/**load_disk**. Bei einer Defragmentierung des geschützten Speicherbereichs werden auch die indizierten Zeichen und Zeichenfolgen mit einbezogen.

6.5.3 Sprünge

Durch die Befehle **list_jump_pos** (synonym zu **set_list_jump**) und **list_jump_rel** können Sprünge zu einer angegebenen Adresse definiert werden, die im Programmablauf von der RTC5 ausgeführt werden.

Beim Befehl **list_jump_pos** kann eine absolute Speicheradresse innerhalb des konfigurierten Listenbereichs ("Liste 1" und "Liste 2") angegeben werden. Sprünge in den geschützten Speicherbereich ("Liste 3") und aus dem geschützten Speicherbereich heraus sind bei diesem Befehl verboten. Ein **list_jump_pos**-Befehl mit einer solchen unerlaubten Sprungadresse wird beim Abarbeiten ignoriert.

Beim Befehl **list_jump_rel** können Sprungweiten (relative Speicheradressen) angegeben werden. Dieser Befehl kann in allen Speicherbereichen, also auch im geschützten Speicherbereich ("Liste 3"), verwendet werden. Bei der Angabe der Sprungadresse muss aber darauf geachtet werden, dass der Sprung nicht über die Grenzen des jeweiligen Speicherbereichs hinaus erfolgt. Andernfalls wird der Befehl von der RTC5 beim Abarbeiten ignoriert.

Falls der Befehl 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 RTC5 beim Abarbeiten des Unterprogramms ignoriert.

Hinweise

- Beim Umkonfigurieren des Listenspeichers oder bei der Konvertierung eines Unterprogramms kann es passieren, dass eine ursprünglich legale Sprungadresse aufgrund neuer Listengrenzen oder einer veränderten Speicherposition des Unterprogramms zu einer illegalen Sprungadresse wird. In diesem Fall wird der entsprechende Sprungbefehl von der RTC5 ignoriert und das Programm funktioniert vermutlich nicht mehr wie gewünscht. Bei der Programmierung mit Sprungbefehlen muss daher mit einer entsprechenden Sorgfalt vorgegangen werden.
- Wenn konditionale Sprungbefehle verwendet werden, hängt die Ausführung eines Sprungs von einem externen Steuersignal ab (siehe "Bedingte Befehlsausführung", Seite 244).
- Sprungbefehle 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.
  Konditionale Sprungbefehle wie `list_jump_rel_cond( Mask1, Mask0, 0 )` sind dagegen erlaubt, z. B. zum Warten auf die Bestätigung eines Signals.

6.5.4 Ringpuffer-Modus

Der Ringpuffer-Modus (engl. circular queue mode), der bei der RTC3/RTC4 mit dem Befehl **set_list_mode** aktiviert werden konnte, wird nicht von der RTC5 unterstützt.

Effektiv lässt sich dieser Betriebsmodus aber bei der RTC5 einfach mit Hilfe eines alternierenden Listenwechsels und des Befehls **load_list** realisieren.

Mit `load_list ( 3, 0 )` ist dabei – wie im RTC3/RTC4-Ringpuffer-Modus – sichergestellt, dass neue Befehle nur in eine bereits abgearbeitete Liste (die nicht BUSY ist) geladen werden, ohne dass die Nummer der Liste explizit angegeben werden muss, und dass das kontinuierliche Abarbeiten von Listenbefehlen nicht durch das gleichzeitige Nachladen neuer Befehle gestört wird (siehe auch "Alternierender Listenwechsel", Seite 80 und "Laden mit Schutzfunktion", Seite 75).

Benutzer, die einfach nur – wie im RTC3/RTC4-Ringpuffer-Modus – die volle Kapazität des Listenspeichers einer einzigen Liste zuweisen wollen, können dies einfach mit dem Befehl **config_list** realisieren.

6.5.5 Schleifen

Mit Listensprüngen (siehe Seite 89) und bedingten Sprüngen (siehe Seite 244) 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.

Die Befehle **list_repeat** und **list_until** müssen immer paarweise verwendet werden. Ungepaarte bzw. überzählige Befehle (**list_until** ohne zugehöriges **list_repeat** sowie **list_repeat**-Anweisungen, die zu einer Verschachtelungstiefe über 8 führen würden) werden ggf. 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. Über Unterprogrammgrenzen hinweg funktionieren Schleifen nicht, da Listensprünge in ein Unterprogramm hinein oder aus einem Unterprogramm heraus nicht zulässig sind (siehe Seite 89).

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, sofern 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 angefangene Schleifen können nicht mehr beendet werden, da das nächste **list_until** dann kein zugehöriges **list_repeat** mehr vorfindet.

set_end_of_list löscht die gesamte Schleifenverwaltung, sofern kein automatischer Listenwechsel anliegt, **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, sofern die Unterprogramme nicht selbst ungepaarte **list_repeat**/**list_until**-Befehle enthalten).

6.6 Verwendung mehrerer RTC5-Karten in einem Rechner

Die RTC5-DLL unterstützt die simultane Ansteuerung von beliebig vielen RTC5-Karten in einem PC.

Die RTC5 -Karten arbeiten völlig unabhängig voneinander. Die Befehlslisten aller Karten können jederzeit geladen und ausgegeben werden.

Der Befehlssatz der RTC5 ermöglicht zwei Methoden, Anwenderprogramme zu schreiben, wenn mehrere RTC5-Karten verwendet werden:

- die Multi-Board-Programmierung und
- die sequentielle Programmierung.

6.6.1 Multi-Board-Programmierung

Bei der Multi-Board-Programmierung werden die Multi-Board-Versionen (Befehlsnamen mit dem Vorsatz `n_`) der RTC5-Befehle verwendet.

Jeder Multi-Board-Befehl erlaubt die Angabe der Nummer derjenigen RTC5-Karte, an die der Befehl übergeben werden soll.

Gehen Sie wie folgt vor, um einen Multi-Board-Befehl einzusetzen:

- Stellen Sie dem Befehlsnamen den Vorsatz `n_` voran und
- fügen Sie die Nummer derjenigen RTC5-Karte, an die der Befehl übergeben werden soll, als ersten Parameter (32-Bit-Wert ohne Vorzeichen) ein. Der restliche Parametersatz des Befehls bleibt der gleiche wie beim Single-Board-Befehl.

Die installierten RTC5-Karten sind entsprechend der bei der Initialisierung gefundenen Reihenfolge (beginnend bei 1) durchnummeriert. Auf welche der RTC5-Karten sich die Nummern beziehen, lässt sich mit dem Multi-Board-Befehl **n_get_serial_number** (siehe Seite 320) herausfinden (siehe auch Beispiel (3), unten). Wieviele RTC5-Karten von der RTC5-DLL gesehen werden kann man mit dem Befehl **rtc5_count_cards** überprüfen.

Hinweise

- Übersteigt die angegebene Nummer die Zahl der bei der Initialisierung gefundenen RTC5-Karten (siehe **rtc5_count_cards**) oder wird 0 angegeben (reale Karten zählen ab 1), werden Multi-Board-Befehle an die aktive Karte ausgegeben.

Alle Multi-Board-Befehle sind in Kapitel 10.1, Seite 250 aufgeführt. Es wird für fast alle Single-Board-Befehle auch ein zugehöriger Multi-Board-Befehl bereitgestellt. Bei Befehlen, bei denen das nicht der Fall ist, wird in der Befehlsauflistung (ab Seite 261) explizit darauf hingewiesen.

Beispiele in Pascal

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

schreibt einen Jump-Befehl zum Punkt (500, 500) in die aktuelle Liste der RTC5-Karte Nr.1.

(2) `n_execute_list(RTC5_no, list_no)`
führt die Liste mit der Nummer `list_no` (1 oder 2) auf der RTC5-Karte mit der Nummer aus, die durch die Variable `RTC5_no` spezifiziert wird.

(3) `sn_1 := n_get_serial_number(1)`
gibt die Seriennummer von RTC5-Karte Nr.1 zurück.

6.6.2 Sequentielle Programmierung

Bei der sequentiellen Programmierung wird jeweils eine der eingesteckten RTC5-Karten mit dem Befehl **select_rtc** für das jeweilige Anwenderprogramm aktiviert. Bei dieser Programmiermethode können die gewöhnlichen Single-Board-Befehle verwendet werden. Multi-Board-Befehle werden durch den Befehl **select_rtc** nicht beeinflusst (sofern die Kartennummer gültig ist, s.o.). Alle Befehle, die auf einen **select_rtc**-Befehl folgen, werden so lange an die ausgewählte Karte übergeben, bis der Befehl **select_rtc** erneut verwendet wird. Achten Sie darauf, dass die angegebene RTC5-Nummer nicht größer ist als die Anzahl der RTC5-Karten. Siehe auch **rtc5_count_cards**.

Seien Sie vorsichtig, wenn ein Prozess mehrere Karten über verschiedene Threads bedient, denn der Befehl **select_rtc** leitet die Ausgabe aller gegenwärtig laufenden Threads eines Prozesses (nicht derjenigen eines anderen Prozesses) sofort an die spezifizierte RTC5-Karte weiter. In Multi-Threaded-Anwendungen kann dies zu Programmierfehlern führen. Für solche Anwendungen sollten möglichst nur die Multi-Board-Befehle verwendet werden, siehe Kapitel 6.6.1 "Multi-Board-Programmierung", Seite 92.

Vorsicht!

- Der Befehl **select_rtc** bestimmt die aktive RTC5-Karte für alle gegenwärtig laufenden Threads eines Prozesses (Anwenderprogramms).
  In Multi-Threaded-Anwendungen kann dies zu Programmierfehlern führen. Für solche Anwendungen sollten möglichst nur Multi-Board-Befehle verwendet werden.

6.6.3 Master/Slave-Betrieb

Sollen mehrere RTC5-Karten mit synchroner Taktung in einem PC betrieben werden, dann müssen die RTC5-Karten zuerst paarweise über die MASTER- und SLAVE-Anschlüsse miteinander verbunden und dann in benachbarte PCI-Steckplätze gesteckt werden. Verbinden Sie jeweils den MASTER-Anschluss einer Karte mit dem SLAVE-Anschluss einer anderen Karte. Geeignete Verbindungskabel sind bei SCANLAB erhältlich.

In einer Master/Slave-Kette ist automatisch diejenige RTC5-Karte die Master-Karte, an deren MASTER-Anschluss eine weitere und an deren SLAVE-Anschluss keine weitere RTC5-Karte angeschlossen ist. Alle anderen RTC5-Karten sind Slave-Karten.

Mit **get_master_slave** kann für jede RTC5-Karte der Master/Slave-Status abgefragt werden, d.h. ob die Karte als Master-, Slave- oder Einzelkarte betrieben wird.

Für ein Quellcodebeispiel auf eine Prüfung, welche RTC5-Karte Master, und welche RTC5-Karte Slave ist, siehe Abschnitt "Beispiel-Code", Seite 95.

Initialisierung

Auf allen RTC5-Karten einer Master/Slave-Kette müssen die Befehle **load_program_file** und **load_correction_file** erteilt worden sein. Die synchrone Taktung mit stabiler Phasenlage der Master/Slave-Kette wird durch die erste nicht-initialisierte Karte unterbrochen. Eine RTC5-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 RTC5-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 dem Befehl **sync_slaves** synchronisiert werden. Der Befehl muss dazu einmalig 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.

Nach der Synchronisierung ist die Takt-Phase jeder Karte um ca. 0,16 µs (reproduzierbar) gegenüber derjenigen der vorgeschalteten Karte verzögert. Ohne die Synchronisierung kann die Verzögerung bis zu 10 µs betragen. Mit **get_sync_status** kann überprüft werden, ob eine Slave-Karte synchron zur Master-Karte (bzw. ihrer Vorgängerkarte in der Master/Slave-Kette) ist.

Hinweise

- Die Encoder-Signale 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 0,16 μs berücksichtigt werden.

Anpassung der Kurzbefehlszählung (ab Version DLL 523, OUT 524)

Die maximal erlaubte Anzahl von unmittelbar aufeinander folgenden kurzen Listenbefehlen innerhalb eines 10 µs-Taktes hängt von der DSP-Version ab. Damit die Abarbeitung von kurzen Listenbefehlen auch bei Verwendung von mehreren RTC5-Karten unterschiedlicher DSP-Versionen in einer Master/Slave-Kette synchron abläuft, reduziert der **sync_slaves**-Befehl (sofern er an die Master-Karte geschickt wird) alle schnelleren Karten der Master/Slave-Kette bzgl. der Kurzbefehlszählung auf die langsamste Karte (d.h. auf die Karte mit der kleinsten DSP-Version).

Hinweise

- Die CPU-Frequenz selbst wird nicht angeglichen, sondern lediglich die Zählung der kurzen Listenbefehle.
- Mit dem Befehl **set_dsp_mode** kann eine entsprechende Anpassung auch für jede RTC5-Karte separat eingestellt werden.
- Es ist aber darauf zu achten, das einige Befehle (z.B. **mark_ellipse_abs**) nur auf den Karten mit höherer DSP-Version verfügbar sind. Eine Angleichung der Kurzbefehlszählung ändert an diesen Bedingungen nichts. Verwenden Sie nur solche Befehle, die auch für die Karte mit der kleinsten DSP-Version verfügbar sind, damit die Master/Slave-Kette synchron bleibt.

Synchrone Starts und Stopps

Externe Listen-Starts (sofern mit **set_control_mode** freigegeben) und externe Listen-Stopps werden in einer Master/Slave-Kette von einer Karte zu allen nachgeschalteten Slave-Karten durchgereicht. Durch ein externes Start-Signal, ein **simulate_ext_start** oder **simulate_ext_start_ctrl** an der Master-Karte kann daher ein synchroner Listen-Start aller Karten (mit vorgebbaren Streckenverzögerungen) ausgelöst werden; und durch ein externes Stopp-Signal oder ein **simulate_ext_stop** an der Master-Karte ein synchroner Listen-Stopp.

Listen-Starts mit **execute_list** oder **execute_at_pointer**, sowie Listen-Stopps mit **stop_execution** werden dagegen nicht durchgereicht und müssen auch an Master/Slave-synchronisierten Karten separat ausgeführt werden.

Siehe auch "Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation", Seite 239.

Beispiel-Code

Der folgende Delphi-Quellcode zeigt beispielhaft, wie man prüft, welche RTC5-Karte Master, und welche Slave ist.

Der Code muss in ein Anwenderprogramm eingebunden werden (siehe Seite 71).

```
if init_rtc5_dll()   <> 0 then halt; // Initialize the RTC5 DLL
if rtc5_count_cards() <> 2 then halt; // Are 2 RTC5 boards in the PC?

for CardNo := 1 to 2 do // Load firmware and 3D correction files onto both boards
begin
  n_stop_execution(CardNo); // Stop RTC if a task is currently still running
  if n_load_program_file(CardNo, nil) <> 0 then halt;
  if n_load_correction_file(CardNo, 'D3_nnn.ct5', 1, 3) <> 0 then halt;
end;

// 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

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); // For external control, you must use the master
// board's control inputs
n_sync_slaves(Master); // Synchronize master and slave boards
// check snychronization status at any time
// provide an external /START or a simulated external start before checking
n_simulate_ext_start_ctrl(Master);
Result = n_get_sync_status(Master); // must be 640
Result = n_get_sync_status(Slave);  // must be <11
```

6.7 Verwendung durch mehrere Anwenderprogramme

Die Verwendung der installierten RTC5-Karten durch unterschiedliche Anwenderprogramme wird von der DLL über eine DLL-interne Kartenverwaltung koordiniert. Die Kartenverwaltung wird durch den Befehl **init_rtc5_dll** initiiert.

Dieser Befehl vergibt einem Anwenderprogramm automatisch mit **acquire_rtc** die Zugriffsberechtigung für die gefundenen Karten, sofern die Zugriffsberechtigung nicht bereits an ein anderes Anwenderprogramm vergeben wurde (es können zwar beliebig viele RTC5-Karten oder Anwenderprogramme gleichzeitig verwendet werden, ein und dieselbe Karte kann aber nicht gleichzeitig von mehreren Anwenderprogrammen verwendet werden). Zugriffsberechtigungen (auch zeitweise) auf die vorhandenen Karten werden von der DLL exklusiv vergeben. Mehrere Threads *eines* Anwenderprogramms können zwar ein und dieselbe Karte verwenden, aber nicht gleichzeitig Befehle senden (die DLL serialisiert die Befehlsaufrufe automatisch).

Ohne Zugangsberechtigung ist eine Karte von einem Anwenderprogramm nur mit ausschließlich 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 von einem anderen Anwenderprogramm beansprucht werden, nachdem die Karte mit **release_rtc** oder **free_rtc5_dll** explizit von dem zugriffsberechtigten Anwenderprogramm freigegeben wurde. Die Übernahme einer freigegebenen Karte durch ein Anwenderprogramm erfolgt dann mit **acquire_rtc** (bzw. **init_rtc5_dll** oder **select_rtc**).

Bei Übernahme einer Karte mit **acquire_rtc** (bzw. **init_rtc5_dll** oder **select_rtc**) erfolgt eine Versionsprüfung für die RTC5-DLL und die Programmdateien `RTC5OUT.out`, `RTC5RBF.rbf` und `RTC5DAT.dat`. Sind noch keine Programmdateien geladen, kann die Versionsprüfung zwar nicht explizit durchgeführt werden, gilt aber dennoch als bestanden und verhindert damit die Übernahme nicht. Sind Programmdateien geladen und bei der Versionsprüfung wird ein Fehler festgestellt, dann wird der Zugriff verweigert (**get_last_error**-Returncode `RTC5_ACCESS_DENIED|RTC5_VERSION_MISMATCH`).

6.7.1 Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm

Die Befehle **init_rtc5_dll**, **acquire_rtc**, **free_rtc5_dll**, **release_rtc** und **select_rtc** beeinflussen die Zugriffsberechtigung oder die Aktivierung der installierten RTC5-Karten, lösen aber keinen Reset der RTC5-Karten aus (ein Karten-Reset wird nur durch den Befehl **load_program_file** initiiert). Sie beeinflussen also weder den Listenspeicher, noch die zuvor durch Kontrollbefehle vorgenommenen Einstellungen der Karte. Auch der Programmablauf eines evtl. auf der Karte zuvor gestarteten und noch laufenden Listenprogramms wird durch diese Befehle nicht beeinträchtigt. Auch löschen die Befehle nicht die diesbezüglichen Informationen in der DLL.

Anwenderprogramme, die eine Karte nachträglich mit **acquire_rtc** (bzw. **init_rtc5_dll** oder **select_rtc**) beanspruchen, übernehmen daher die Karte mit unbereinigtem Speicher und Betriebszustand, und sie können die auf der Karte gespeicherten Daten und Einstellungen verwenden und in den Programmablauf eines evtl. durch das vorherige Anwenderprogramm gestarteten Programms eingreifen.

Falls eine Karte von einem Anwenderprogramm mit **release_rtc** freigegeben und anschließend – ohne zwischenzeitliche Beanspruchung durch ein anderes Anwenderprogramm – wieder von demselben Anwenderprogramm beansprucht wird, so kann die RTC5-Karte unverändert weiterbetrieben werden, da hier auch alle Konfigurationsdaten in der DLL unverändert erhalten bleiben. Letzteres gilt aber nicht, wenn die Karte mit **free_rtc5_dll** freigegeben und nach **init_rtc5_dll** wieder beansprucht wird.

Wenn eine RTC5-Karte von einem anderen Anwenderprogramm übernommen wird, werden einige wichtige, vom vorherigen Anwenderprogramm nur in der DLL verwaltete Informationen *nicht* (automatisch) übernommen. Dem übernehmenden Anwenderprogramm fehlen dadurch etwa Informationen bzgl. der Speicherkonfiguration, der Verwaltung des geschützten Bereichs oder dem Betriebsstatus.

Wenn nach einer Kartenübernahme ohnehin mit **load_program_file** ein Karten-Reset ausgeführt wird und alle Einstellungen neu definiert werden, dann spielen solche fehlende Informationen keine Rolle.

Will das übernehmende Anwenderprogramm die RTC5-Karte dagegen im übernommenen Zustand weiterbetreiben, dann muss sie gegebenenfalls die in der DLL fehlenden Informationen explizit abfragen, von dem vorherigen Anwenderprogramm übernehmen und erneut explizit setzen, damit die DLL mit der Karte konsistent bleibt. Beachten Sie dazu die folgende Hinweise:

- Für ein korrektes Verhalten des Input-Pointers an den Listengrenzen muss die in der DLL für das übernehmende Anwenderprogramm eingetragene Speicherkonfiguration mit der aktuellen Speicherkonfiguration der übernommenen Karte konsistent sein. Ist dies nicht der Fall, kann der Befehl **get_config_list** verwendet werden. Dieser Befehl ermittelt die aktuelle Speicherkonfiguration der Karte und stellt die Kartenverwaltung der DLL für das aufrufende Anwenderprogramm entsprechend ein.
- Die Verwaltungstabellen der geschützten Funktionen (indizierte Unterprogramme, Zeichensätze und Zeichenfolgen) befinden sich auf der Karte und werden daher bei einer Kartenübernahme übernommen. 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 Seite 85) wiederhergestellt werden. Eine anderweitige Übertragung ist nicht möglich. Das gemischte Laden von geschützten Funktionen durch unterschiedliche Anwenderprogramme sollte daher vermieden werden.
- Wenn beim Laden einer geschützten Funktion **release_rtc** aufgerufen wird bevor der abschließende **list_return**-Befehl übergeben ist, dann wird die Funktion nicht auf der RTC5-Karte gespeichert.
- Der Input-Pointer wird generell nicht übernommen (es wird die auf der DLL für das übernehmende Anwenderprogramm aktuell gespeicherte Input-Pointer-Position verwendet, evtl. nach **get_config_list** korrigiert). Output-Pointer von Listen können dagegen nach der Übernahme mit **get_status** abgefragt werden.
- Der Listenstatus ist evtl. nach der Übernahme bis zum nächsten `load_`...-Befehl (bzgl. LOAD und READY) falsch. Das ist aber für die weitere Ausführung unwichtig, und der Status wird beim nächsten `load_`...-Befehl neu gesetzt.
- Sonstige Einstellungen wie **start_loop** oder die Lasereinstellungen sind in der DLL irrelevant. Die von dem 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 separat. Beim Austausch von Zugriffsberechtigungen werden diese Daten nicht mitgetauscht.

6.8 Error-Handling

Damit bei der Programmierung der RTC5-Karte Fehler im Programmablauf aufgefangen werden können, führt die RTC5 zum einen eine allgemeine Fehlerbehandlung durch, zum anderen bei einigen Befehlen auch eine individuelle.

Allgemeine Fehler treten z.B. auf, wenn das Anwenderprogramm keine Zugriffsberechtigung für die Karte besitzt (`RTC5_ACCESS_DENIED`), wenn die Karte auf einen Kontrollbefehl nicht reagiert (`RTC5_TIMEOUT`) oder wenn beim Laden ein PCI-Kommunikationsproblem auftritt (`RTC5_SEND_ERROR`).

Individuelle Fehler können z.B. auftreten, wenn ein Befehl mit einem (nicht korrigierbaren) unzulässigen Parameter aufgerufen wird (`RTC5_PARAM_ERROR`, siehe z.B. **get_value** oder **write_da_x**), wenn das Laden eines Listenbefehls verweigert wird (`RTC5_REJECTED`, z.B. bei illegalem Inputpointer) oder ein Kontrollbefehl zum falschen Zeitpunkt gesendet wird (`RTC5_BUSY`, z.B. **goto_xy**, wenn noch eine Liste läuft).

Kontrollbefehle werden in diesen Fällen nicht ausgeführt, für die Listenbefehle wird meist ein **list_nop**-Befehl gesendet (z.B. bei `RTC5_PARAM_ERROR` oder `RTC5_IGNORED`, siehe z.B. **set_end_of_list**).

Die diesen Fehlern zugeordneten Bits werden in der DLL bei jedem Befehl in (kartenspezifischen) Fehlervariablen `LastError` und `AccError` gesetzt bzw. akkumuliert.

- Der Fehlercode `LastError` wird am Anfang eines jeden Befehls automatisch zurückgesetzt und entspricht daher einer Auflistung der beim zuletzt ausgeführten Befehl aufgetretenen Fehler. `LastError` kann mit dem Befehl **get_last_error** ausgelesen werden(1).
- Der akkumulierte Fehlercode `AccError` wird dagegen beim Initialisieren der DLL zurückgesetzt, kann aber auch von der Anwendung selbst durch den Befehl **reset_error** zurückgesetzt werden. `AccError` entspricht einer Auflistung der seit dem letzten Error-Reset (**reset_error**) aufgetretenen Fehler und kann mit **get_error** ausgelesen werden(1).

(1) Der beschriebene Mechanismus wirkt nur bei Befehlen, die eine Kommunikation mit der RTC5 aufbauen. Befehle, die *keine* Kommunikation mit der RTC5 aufbauen (wie z.B. **rtc5_count_cards**, **set_rtc4_mode** oder **get_serial_number**) erzeugen bzw. verändern weder `LastError` noch `AccError` (siehe auch Hinweis bei den entsprechenden Befehlsbeschreibungen).

Die Fehlerbehandlung erfolgt für jede Karte und für jedes Anwenderprogramm separat. Ein **reset_error** löscht nicht den Fehlercode eines anderen Anwenderprogramms, die gerade für die angegebene Karte zugriffsberechtigt ist. Beim Austausch von Zugriffsberechtigungen werden diese Daten nicht mitgetauscht, ebenso wenig alle anderen kartenspezifischen Daten.

Die Fehlerbehandlung geschieht ausschließlich beim Initialisieren und beim Laden auf die RTC5, nicht bei der Ausführung eines Listenprogramms.

Ein Beispielprogramm zur Einbindung der karteneigenen Fehlervariablen finden Sie bei der Befehlsbeschreibung des Befehls **get_error**.

Einige Kontrollbefehle (wie **init_rtc5_dll**, **load_correction_file** oder **load_program_file**) liefern zusätzlich einen speziellen Fehlercode als Returnwert. Dieser wird nicht zwischengespeichert und muss von der Anwendung sofort ausgewertet oder verworfen werden.

6.8.1 Download-Überprüfung

Insbesondere für medizintechnische Anwendungen ist eine Überprüfung der RTC5-Kommunikation erforderlich. Hierfür kann mit **set_verify** für jede Karte separat eine Download-Überprüfung aktiviert werden. Dadurch verlängern sich aber automatisch die Download-Zeiten.

Bei aktivierter Überprüfung wird im Fehlerfall der Fehlercode `RTC5_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 der Befehl **load_program_file** abgebrochen wurde).

Bei aktivierter Download-Überprüfung werden folgende Überprüfungen vorgenommen (beachten Sie auch die Hinweise in der **set_verify**-Befehlsbeschreibung):

Listenbefehl laden

Bei Listenbefehl-Downloads wird jeder Download zurückgelesen und auf Gleichheit zum gesendeten Befehl geprüft. Dabei wird nur der Transfer zur Karte selbst geprüft, etwaige automatische Parameteranpassungen (z. B. Clipping) werden nicht berücksichtigt.

Kontrollbefehl laden

Bei Kontrollbefehlen werden die zugehörigen Parameter zurückgelesen und auf Gleichheit zu den gesendeten Parametern geprüft. Etwaige automatische Parameteranpassungen werden dabei nicht berücksichtigt.

load_program_file

Beim Senden von **load_program_file** erfolgt die folgende Überprüfung:

- Die binäre Hilfsdatei `RTC5DAT.dat` wird über die Check-Summe auf Korrektheit der Datei und auf Korrektheit des PCI-Tansfers geprüft.
- Die Firmware `RTC5RBF.rbf` wird nur per Handshake des bitweisen Transfers kontrolliert, es ist keine zusätzliche Prüfung möglich.
- Die Programmdatei `RTC5OUT.out` wird für jede geladene Sektion unmittelbar durch Zurücklesen überprüft. Bei einem Fehler wird der Ladevorgang abgebrochen. Für diese Aktion ist die Treiberversion 1.0.4.0 oder höher erforderlich (siehe **set_verify**-Befehlsbeschreibung).

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 **set_verify**-Befehlsbeschreibung).

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**-Fehlercode `RTC5_VERIFY_ERROR` wird gegebenenfalls auch der entsprechende Fehler-Rückgabewert des Ladebefehls gesetzt.

6.8.2 Überprüfung auf Überläufe

Mit **get_overrun** können Sie überprüfen, ob Laufzeitüberschreitungen des 10 µs-Taktes stattgefunden haben (siehe auch Seite 142).

6.8.3 Beispiel-Code

Der folgende C-Quellcode zeigt beispielhaft, wie ein Fehler während der Initialisierung aufgefangen werden kann. Er stellt sicher, dass das Programm mit einer Fehlermeldung beendet wird,

- wenn während der Initialisierung mit **init_rtc5_dll** ein Fehler auftritt (z. B. wenn keine RTC5-Karte gefunden wird),
- wenn die gewünschte RTC5-Karte (hier: die Karte mit der Seriennummer 12345) nicht gefunden wird,
- wenn der Zugriff auf die gewünschte RTC5-Karte verweigert wird,
- wenn 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 Seite 71).

```
UINT ErrorCode;

ErrorCode = init_rtc5_dll();

if ( ErrorCode )
{
   // Auslesen der Anzahl der RTC5-Karten, die während der Initialisierung mit init_rtc5_dll gefunden wurden
   const UINT RTC5CountCards = rtc5_count_cards();
   if ( RTC5CountCards )
   {
      // Detaillierte Fehleranalyse für alle gefundenen Karten
      UINT AccError( 0 );
      for ( UINT i = 1; i <= RTC5CountCards; i++ )
      {
         // Fehler, die während der Ausführung von init_rtc5_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( "RTC5 board number %d (serial number %d): Error %d detected\n",
                        i, SerialNumber, Error );
            n_reset_error( i, Error );
         }
      }
      if ( AccError )
      {
         free_rtc5_dll();
         return;
      }
   }
   else
   {
      printf( "Initializing the DLL: Error %d detected\n", ErrorCode );
      free_rtc5_dll();
      return;
   }
}
else
{
```

```
   // Auslesen der internen Kartennummer der gewünschten RTC5-Karte
   const UINT SerialNumberOfDesiredBoard ( 12345 );
   const UINT RTC5CountCards = rtc5_count_cards();
   UINT InternalNumberOfDesiredBoard ( 0 );
   for ( UINT i = 1; i <= RTC5CountCards; i++ )
   {
      if ( n_get_serial_number( i ) == SerialNumberOfDesiredBoard )
      {
         InternalNumberOfDesiredBoard = i;
      }
   }
   if ( InternalNumberOfDesiredBoard == 0 )
   {
      printf( "RTC5 board with serial number %d not detected.\n", SerialNumberOfDesiredBoard);
      free_rtc5_dll();
      return;
   }
   // Auswählen der gewünschten RTC5-Karte als die aktive RTC5-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 )    //  RTC5_VERSION_MISMATCH
      {
         // Hier muss der Multi-Board-Befehl n_load_program_file zur Initialisierung der Karte verwendet werden
         // (der Single-Board-Befehl load_program_file( 0 ) würde hier die Ausführung verweigern).
         if ( ErrorCode = n_load_program_file( InternalNumberOfDesiredBoard, 0 ) )
         {
            printf( "n_load_program_file returned error code %d\n", ErrorCode );
         }
      }
      else
      {
         printf( "No access to RTC5 board with serial number %d\n", SerialNumberOfDesiredBoard );
         free_rtc5_dll();
         return;
      }
      if ( ErrorCode )
      {
         printf( "No access to RTC5 board with serial number %d\n", SerialNumberOfDesiredBoard );
         free_rtc5_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 acht(1) sogenannte freie Variablen zur Verfügung, die von Benutzern mit dem Kontrollbefehl **set_free_variable** oder dem kurzen Listenbefehl **set_free_variable_list**, frei mit Daten belegt werden können.

Die Variablenwerte können über die McBSP/SPI-Schnittstelle ausgegeben werden (siehe Kapitel 9.1.7 "McBSP/SPI-Schnittstelle", Seite 237), und sie können mit **get_free_variable** und **get_value** zurückgelesen und mit **set_trigger**/**set_trigger4** aufgezeichnet werden (siehe Befehlsbeschreibungen).

Mithilfe der freien Variablen können z.B. Steuerbefehle über die McBSP/SPI-Schnittstelle an die Benutzer-Hardware übertragen oder Betriebszustände der Karte (z.B. bei Verzweigungen) dokumentiert werden.

Hinweise

- Durch **set_free_variable** oder **set_free_variable_list** können beliebige 32-Bit-Werte ohne Vorzeichen als Variablenwerte festgelegt werden. Über die McBSP/SPI-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**/**set_trigger4** liefern dagegen die vollen 32 Bit-Werte zurück).

(1) Gilt für ≥DLL 539, ≥OUT 539. Vorher: vier.

7 Grundlegende Funktionen zur Scan-Kopf- und Lasersteuerung

7.1 Markieren von Punkten, Linien und Bögen

7.1.1 Markieren mit Vektor- und Bogenbefehlen

Wie in Kapitel 6.1 "Grundlegendes zum RTC5 Software-Konzept", Seite 65 dargestellt, wird die Positionierung der Achsen des Scan-Systems und somit des Laserstrahls bei Ansteuerung mit der RTC5 durch Aufruf von Jump-, Mark-, Arc- und Ellipse-Befehlen realisiert.

Jeder dieser Befehle beschreibt einen einzelnen Vektor bzw. Bogen. Mit der RTC5 werden daher grundsätzlich ausschließlich Punkte, Linien und Bögen markiert(1).

Auch beim Markieren von Zahlen oder Buchstaben werden die einzelnen Zeichen letztlich über einzelne Linien, Punkte und Bögen definiert (siehe Kapitel 7.5 "Markieren von Uhrzeit, Datum und Seriennummern", Seite 170).

Bei Vektorbefehlen (Jump, Mark) ist als Parameter jeweils der *Endpunkt* des entsprechenden Vektors anzugeben(2). Als Startpunkt eines Vektors wird in der Regel der Endpunkt des zuletzt abgearbeiteten Vektors oder Bogens verwendet.

Bei Bogenbefehlen (Arc, Ellipse) sind als Parameter die Koordinaten des Bogenzentrums und der bzw. die Bogenwinkel anzugeben. Der Arc-Bogen beginnt an der aktuellen Ausgabeposition, der Ellipsenbogen nur bei geeigneten Parametern.

Nach einem Neustart der RTC5 befindet sich die Ausgabeposition in der Bildfeldmitte im Punkt (0|0) (das Koordinatensystem des Bildfelds wird in Kapitel 7.3 "Scan-Kopf-Steuerung", Seite 134 vorgestellt).

Jeder Vektor oder Bogen, der vom Scan-System abgefahren werden soll, wird von der RTC5 zur Laufzeit in sogenannte Mikrovektoren zerlegt (siehe Abschnitt "Mikrovektorisierung", Seite 107)(3).

Prinzipiell dienen Jump-Befehle dazu, die Achsen des Scan-Systems bei ausgeschaltetem Laser möglichst schnell zu einer neuen Anfangsposition zu bewegen(4), während Markierbefehle (Mark, Arc und Ellipse) eine Markierbewegung mit eingeschaltetem Laser veranlassen (siehe auch nachfolgende Beschreibung).

Für das Markieren eines Punktes muss der Laser (d.h. müssen die "Laser active"-Lasersteuersignale) nach einem Jump- oder Mark-Befehl für die gewünschte Zeitdauer eingeschaltet werden (siehe Kapitel 7.1.3 "Markieren von Punkten", Seite 108).

Beim Markieren von Linien und Bögen wird das Anschalten des Lasers (der "Laser active"-Lasersteuersignale) zu Beginn eines Markierbefehls und das Wiederausschalten z. B. bei einem nachfolgenden Jump-Befehl automatisch von der RTC5 erledigt.

Die Synchronisation der Scan-Kopf- und Laser-Steuerung kann dabei vom Kunden durch Einstellen von Delays auf die jeweilige Anwendung abgestimmt werden (siehe Kapitel 7.2 "Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung", Seite 111).

Die Einstellung der Laserparameter wird in Kapitel 7.4 "Lasersteuerung", Seite 144 beschrieben.

Als Beispiel für eine einfache Markieraufgabe ist am Ende dieses Kapitels (Seite 109) ein Beispiel-Code mit ausführlicher Kommentierung aufgeführt.

(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) Bei den i*DRIVE*-Scan-Systemen können Jump-und **goto_xy**-Befehle wahlweise (wie voreingestellt) im Vektormodus (mikrovektorisiert) oder – nach Freigabe und Aktivierung – im Sprungmodus ausgeführt werden (siehe Seite 176).

(4) Außerhalb einer Liste ist so eine Neupositionierung (auch bei eingeschalteten Lasersteuersignalen) mit **goto_xy** oder **goto_xyz** möglich.

Jump-Befehle

Ein Jump-Befehl (**jump_abs** oder **jump_rel**(1)) veranlasst eine (in der Regel) schnelle Bewegung der Scannerspiegel, so dass die für den Laserstrahl vorgesehene Fokusposition vom Anfangs- zum Endpunkt eines Vektors "springt". Während des Sprungs ist der Laser ausgeschaltet (die "Laser active"-Lasersteuersignale werden dazu gegebenenfalls automatisch am Anfang des Vektors ausgeschaltet, siehe auch Abschnitt "Scanner-Delays", Seite 113). Die Sprunggeschwindigkeit wird von Benutzern mit den Befehlen **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 Sprungvektors 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 mit konstanter Geschwindigkeit vom Anfangs- bis zum Endpunkt eines Vektors geführt und produziert eine geradlinige Markierung auf der Oberfläche des Werkstücks. Die "Laser active"-Lasersteuersignale werden durch die RTC5 zu Beginn der Ausführung eines Mark-Befehls automatisch eingeschaltet. Wenn unmittelbar auf einen Mark-Befehl ein weiterer Mark-Befehl (oder ein Bogenbefehl) folgt, dann bleiben die "Laser active"-Lasersteuersignale eingeschaltet. Durch direkte Aneinanderreihung von Mark-Befehlen (und Bogenbefehlen) kann man daher kontinuierlich markieren (Polygonzug). Die "Laser active"-Lasersteuersignale werden am Anfang des dem letzten Mark-Befehl (oder Bogenbefehl) eines Polygonzugs folgenden Jump- (oder allgemein: Nicht-Mark- oder Arc-) Befehls ausgeschaltet (siehe auch "Eckwert", Seite 119).

Die Befehle **set_mark_speed** oder **set_mark_speed_ctrl** definieren die Markiergeschwindigkeit. Sie ist im Allgemeinen kleiner als die Sprunggeschwindigkeit und kann an jeder beliebigen Stelle innerhalb einer Liste geändert werden oder mittels des entsprechenden Kontrollbefehls, wenn gerade keine Liste abgearbeitet wird.

(1) Zur Verwendung von abs- und rel-Befehlen siehe "AbsCalls" Seite 83. Zusätzlich stellt die RTC5 zeitbemessene Vektorbefehle (siehe Seite 223), Para-Vektorbefehle (siehe Seite 166) und – bei freigeschalteter 3D-Option – 3D-Vektorbefehle (siehe Seite 194) zur Verfügung.

Arc-Befehle

Mit den Bogenbefehlen **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 Bogenbefehls wird der Laserfokus mit der angegebenen Geschwindigkeit entlang des spezifizierten Bogens geführt. Die "Laser active"-Lasersteuersignale werden zu Beginn der Ausführung eines Bogenbefehls automatisch eingeschaltet und am Anfang des nachfolgenden Jump- (oder allgemein: Nicht-Mark- oder Nicht-Bogen-) Befehls ausgeschaltet, sofern kein weiterer Bogenbefehl (bzw. eine Serie von Bogen- oder Mark-Befehlen) nachfolgt.

(2) Zur Verwendung von abs- und rel-Befehlen siehe "AbsCalls" Seite 83. Zusätzlich stellt die RTC5 auch zeitbemessene Bogenbefehle zur Verfügung (siehe Seite 223).

Ellipse-Befehle

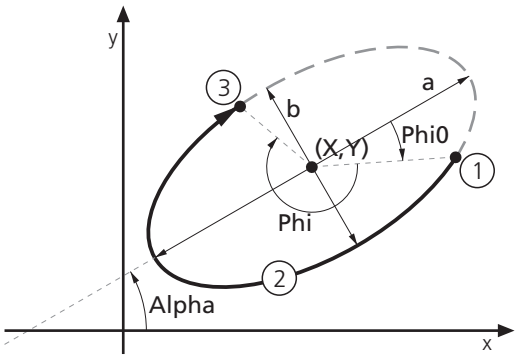
Der RTC5-Treiber stellt auch Befehle zum Markieren ellipsenförmiger Bögen zur Verfügung. Im Gegensatz zum Markieren von Vektoren oder Kreisbögen müssen dabei allerdings in der Regel jeweils zwei Befehle aufgerufen werden: der **set_ellipse**-Befehl sowie der Bogenbefehl **mark_ellipse_abs** oder **mark_ellipse_rel**(1).

Mit dem **set_ellipse**-Befehl wird dabei die Bogenform festgelegt (siehe Abb. 30):

- die Längen `a` und `b` der Ellipsenhalbachsen,
- der Anfangsphasenwinkel `Phi0` (und damit die Lage des Bogenstartpunkts relativ zum Endpunkt der Halbachse `a`) und
- der Bogenwinkel `Phi` (und damit die Länge des zu markierenden Ellipsenausschnitts).

Mit den Bogenbefehlen **mark_ellipse_abs** oder **mark_ellipse_rel** wird die Lage und Orientierung des auszuführenden Bogens festgelegt (siehe Abb. 30):

- die Koordinaten `(X,Y)` des Ellipsenmittelpunkts
- der Winkel `Alpha` zwischen der Ellipsen-Halbachse `a` und der x-Achse.



Legende

1 Startpunkt
2 Markierter Bogen
3 Endpunkt

30

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- und Arc-Befehlen wird bei der Ausführung eines Ellipse-Bogenbefehls der Laserfokus mit der angegebenen Geschwindigkeit entlang des spezifizierten Bogens geführt. Die "Laser active"-Lasersteuersignale werden zu Beginn der Ausführung eines Bogenbefehls automatisch eingeschaltet und am Anfang des nachfolgenden Jump- (oder allgemein: Nicht-Mark- oder Bogen-) Befehls ausgeschaltet, sofern kein weiterer Bogenbefehl (bzw. eine Serie von Bogen- oder Mark-Befehlen) nachfolgt. **set_ellipse** ist ein kurzer Listenbefehl und kann daher zwischen einem Markierbefehl und einem nachfolgenden Ellipse-Bogenbefehl aufgerufen werden, ohne dass dadurch der Polygonzug unterbrochen würde (der Laser bleibt an) (siehe auch Seite 250).
- Im Gegensatz zu Mark- und Arc-Befehlen beginnt die Markierung bei einem Ellipse-Bogenbefehl nicht automatisch an der aktuellen Ausgabeposition, sondern immer an dem mit Hilfe der oben genannten Parameter festgelegten Startpunkt. Stimmt der Startpunkt des Bogens nicht mit der aktuellen Position überein, so wird am Anfang der Markierung ein harter Sprung an den Startpunkt ausgeführt (dabei ist kein Jump-Delay wirksam).

(1) Zur Verwendung von abs- und rel-Befehlen siehe "AbsCalls" Seite 83.

- Ellipsenförmige Bögen könnten auch mit einem Kreisbogenbefehl (wie **arc_abs**) markiert werden, wenn mit **set_matrix** eine geeignete Koordinatentransformation (z. B. eine in x/y-Richtung unterschiedliche Skalierung) eingestellt ist. Hierbei variiert jedoch die Bahngeschwindigkeit entlang des Bogens (siehe auch Hinweis auf Seite 186).

  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 Bahngeschwindigkeit (der aktuell eingestellten 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 Markiergeschwindigkeiten zu numerischen Ungenauigkeiten kommen. Dadurch bleibt dort die Bahngeschwindigkeit u. U. nicht exakt konstant (Beispiel: bei einer Exzentrizität $a/b = 2$ und bei 100 Mikrovektoren pro Umfang beträgt die Geschwindigkeitsabweichung ca. 3,7%).

  Da es keine geschlossene Formel zur Berechnung einer elliptischen Bogenlänge gibt, ist außerdem die Schrittweite des zuletzt markierten Mikrovektors in der Regel kürzer und die Markiergeschwindigkeit entsprechend geringer als die vorgegebene Markiergeschwindigkeit. Die Endposition ist jedoch immer exakt.

  Auch bei Verwendung der Sky-Writing-Option kann es beim Vor- und Rücklauf im Bereich der großen Halbachse zu Unstimmigkeiten kommen. Der Benutzer ist selbst dafür verantwortlich zu prüfen, ob die gewählten Parameter mit der erforderlichen Präzision im Einklang stehen.

Para-Befehle

Bei aktivierter vektorkontrollierter Lasersteuerung können auch die Para-Mark- und Para-Jump-Befehle (**para_jump_abs**, **para_jump_rel**, **para_mark_abs**, **para_mark_rel**) genutzt werden.

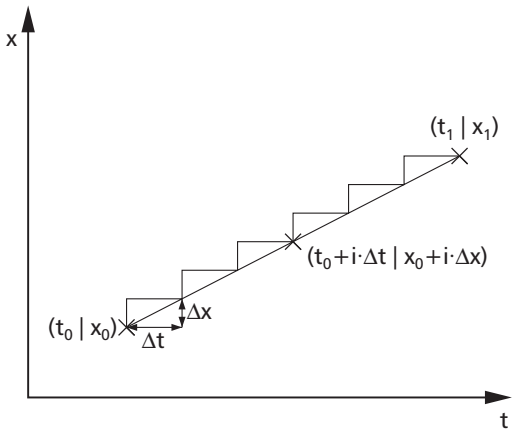
Diese Befehle variieren simultan einen Lasersignalparameter linear entlang des Mark- oder Jump-Vektors (siehe Seite 166)[1].

(1) Zusätzlich stellt die RTC5 zeitbemessene Para-Vektorbefehle zur Verfügung (siehe Seite 166 und Seite 223).

7.1.2 Mikrovektorisierung

Jeder Vektor eines Jump-, Mark- oder Bogenbefehls, der vom Scan-Kopf abgefahren werden soll, wird von der RTC5 in sogenannte Mikrovektoren zerlegt. Diese werden dann bei der Steuerung der Galvanometerscanner über einen analogen Regelkreis umgesetzt.

Abb. 31 zeigt beispielhaft die Zerlegung der x-Komponente eines Vektors in Mikrovektoren. Die Zerlegung der y-Komponente erfolgt analog.



x_0 x-Koordinate der aktuellen Ausgabe-Position vor der Ausgabe des Vektors
x_1 x-Koordinate des Vektor-Endpunkts
Δt Ausgabeintervall (10 µs)

31

Die x-Komponente eines Vektors wird in Mikrovektoren zerlegt.

Die Mikrovektoren werden mit konstantem Ausgabeintervall (Δt) an den Scan-Kopf transferiert.

Die Länge Δs eines Mikrovektors ist

$$\Delta \mathbf{s} = \boldsymbol{v} \times \Delta \mathbf{t}.$$

Dabei ist *v* die momentane Sprung- oder Markiergeschwindigkeit.

Bei 2D- und 3D-Anwendungen wird dasselbe Ausgabeintervall Δt verwendet. Es kann *nicht* von Benutzern verändert werden und ist prinzipiell auf einen Wert von 10 µs festgelegt.

Hinweise

- Eine direkte Ausführung von Vektoren (ohne Mikrovektorisierung) kann mit Hilfe der Mikrovektor-Befehle realisiert werden (siehe Seite 221).
- Bei den i*DRIVE*-Scan-Systemen können Jump-und **goto_xy**-Befehle wahlweise (wie voreingestellt) im Vektormodus (mikrovektorisiert) oder – nach Freigabe und Aktivierung – im Sprungmodus ausgeführt werden (siehe Seite 176).

7.1.3 Markieren von Punkten

Um einen Punkt außerhalb eines Polygonzugs zu markieren, muss der Laser (d.h. müssen die "Laser active"-Lasersteuersignale) nach einem Jump- oder Mark-Befehl für die gewünschte Zeitdauer eingeschaltet werden (siehe **laser_on_list**, **laser_on_pulses_list**, **para_laser_on_pulses_list** und Kapitel 7.4 "Lasersteuerung", Seite 144).

Außerhalb bzw. am Anfang eines Polygonzugs kann ein Punkt (ab RTC5OUT.out Version 527) auch dadurch markiert werden, dass nach einem Jump-Befehl ein Markbefehl (oder Bogenbefehl) der Länge Null verwendet wird (siehe Seite 115).

Innerhalb eines Polygonzugs kann ein Punkt (ab RTC5OUT.out Version 526) auch dadurch markiert werden, dass ein zeitbemessener Vektor- oder Bogenbefehl der Vektor- oder Bogenlänge Null in den Polygonzug eingebaut wird (siehe Seite 223).

7.1.4 Beispiel-Code

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 vorausgesetzt, dass der RTC4-Kompatibilitätsmodus aktiviert ist.

Der Code muss in ein Anwenderprogramm eingebunden werden (siehe Seite 71).

```
// Initialisierung des Scan-Systems:
   // Laden und Zuweisen einer Korrekturdatei
   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_rtc5_dll();
      return;
   }
   select_cor_table( 1, 0 ); // Tabelle #1 wird dem primären Scan-Kopf-Anschluss zugewiesen (Default)

// Initialisierung der Lasersteuerung (siehe Seite 144):
   // Auswählen der CO2-Version als Lasersteuerungsmodus
   set_laser_mode( 0 );

   // Einstellen und Freigeben der "Laser active"-Lasersteuersignale
   set_laser_control( 0x18 ); // All laser signals LOW active (Bits #3 and #4)
      // Dieser Befehl muss mindestens einmal aufgerufen werden, um die Lasersteuersignale zu aktivieren. Danach kann
      // auch enable_laser/disable_laser verwendet werden.

// Die folgenden Befehle sind Listenbefehle
   // Eröffnen der Liste 1
   set_start_list( 1 );

   // Einstellen der Standby-Pulse
   set_standby( 800, 8 );
      // Im RTC4-Kompatibilitätsmodus werden die Standby-Parameter wie bei der RTC4 in Einheiten von 1/8 µs spezi-
      // fiziert und die RTC5 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 Seite 113):
   set_scanner_delays( 25, 10, 5 );
      // Jump-Delay = 250 µs (spezifiziert in [10 µs])
      // Mark-Delay = 100 µs (spezifiziert in [10 µs])
      // Polygon-Delay = 50 µs (spezifiziert in [10 µs])

   // Einstellen der Sprung- und Markiergeschwindigkeit:
   set_jump_speed( 1000.0 );
   set_mark_speed( 250.0 );
      // Im RTC4-Kompatibilitätsmodus werden die Geschwindigkeits-Werte wie bei der RTC4 spezifiziert und die RTC5
      // multipliziert die angegebenen Werte mit 16.
      // Sprunggeschwindigkeit = 1000.0 Bits/ms
      // Markiergeschwindigkeit = 250.0 Bits/ms
```

```
    // Einstellen des Laser-Timings (siehe Seite 144):
    set_laser_pulses( 800, 400 );
        // Im RTC4-Kompatibilitätsmodus werden die Timing-Parameter wie bei der RTC4 in Einheiten von 1/8 µs spezi-
        // fiziert und die RTC5 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 Seite 111):
    set_laser_delays( 100, 100 );
        // Im RTC4-Kompatibilitätsmodus werden die Laser-Delays wie bei der RTC4 in Einheiten von 1 µs spezifiziert und
        // die RTC5 multipliziert die angegebenen Werte mit 2, um sie in ganzzahlige Vielfache von 0,5 µs zu konvertieren.
        // LaserOn-Delay = 100 µs
        // LaserOff-Delay = 100 µs

    // Definieren des Listenendes und des Endes der Befehlsübertragung zur RTC5-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 Listenspeichers zum Schreiben von Listenbefehlen und Setzen des Input-Pointers auf den Anfang
    // von Liste 1
    while ( !load_list( 1, 0 ) );

    // Im Folgenden werden die Listenbefehle zum Markieren von Punkt, Quadrat und Kreis definiert und an die RTC5-Karte
    // übertragen.

    // Markieren des Bildfeldmittelpunkts:
    jump_abs( 0, 0 ); // Sprung zum Bildfeldmittelpunkt
        // Nach dem Sprung wird automatisch ein Jump-Delay eingefügt.
    laser_on_list( 5 ); // Anschalten der Lasersteuersignale für 50 µs

    // Markieren eines Quadrats um den Bildfeldmittelpunkt:
    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 vier Mark-Befehle eingeschaltet. Nach den ersten drei Mark-Befehlen wird jeweils ein
        // Polygon-Delay automatisch eingefügt. Initiiert durch den nachfolgenden Nicht-Markierbefehl (Jump-Befehl,
        // siehe unten), wird nach dem letzten Mark-Befehl automatisch ein Mark-Delay eingefügt und die Lasersteuersignale
        // werden automatisch nach dem letzten Mark-Befehl und einem LaserOff-Delay ausgeschaltet.

    // Markieren eines Kreises rund um den Bildfeldmittelpunkt:
    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-Befehl, siehe unten), wird nach dem
        // Arc-Befehl automatisch ein Mark-Delay eingefügt und die Lasersteuersignale werden automatisch nach dem
        // Arc-Befehl und einem LaserOff-Delay ausgeschaltet.

    // Definieren des Listenendes und des Endes der Befehlsübertragung zur RTC5-Karte
    set_end_of_list();

    // Starten der übertragenen Liste (und damit des Markierprozesses)
    execute_list( 1 );
```

7.2 Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung

Die Synchronisation von Scan-Kopf- und Laser-Steuerung muss auf das dynamische Verhalten der Systemkomponenten abgestimmt sein, d.h. auf das Ansprechverhalten der Galvanometerscanner und des Lasers und auf die Art der Wechselwirkung zwischen Laserstrahlung und Material. Der Benutzer kann dies durch Einstellen der folgenden Delays verwirklichen(1):

- Laser-Delays:
  - LaserOn-Delay
  - LaserOff-Delay
- Scanner-Delays:
  - Jump-Delay (optional variabel)
  - Mark-Delay
  - Polygon-Delay (optional variabel)

Alle Delays werden im Folgenden genauer beschrieben.

7.2.1 Laser-Delays

Es gibt zwei verschiedene Laser-Delays, das LaserOn-Delay und das LaserOff-Delay.

Die Laser-Delays bestimmen, wann die "Laser active"-Lasersteuersignale (d.h. der Laser) vor oder nach einem Mark- oder Bogenbefehl bzw. einer Serie von Mark- und Bogenbefehlen ein- oder ausgeschaltet werden (gegebenenfalls schaltet die RTC5 dabei von "Laser standby"- zu "Laser active"-Signalen oder umgekehrt). Die Laser-Delays haben *keinen* Einfluss auf die Gesamtmarkierungszeit, es sei denn es werden negative Werte gewählt (siehe unten).

LaserOn-Delay und LaserOff-Delay werden durch den Listenbefehl **set_laser_delays** eingestellt. Beide Delays werden in Einheiten von **0,5 µs** angegeben.

Die Längen der Laser-Delays sind auf die eingestellte Sprung- und Markiergeschwindigkeiten abzustimmen (siehe auch "Hinweise zur Optimierung der Delays", Seite 121).

LaserOn-Delay

Das LaserOn-Delay wird von der RTC5 automatisch nach dem Start eines Mark- oder Bogenbefehls eingefügt und verzögert das Einschalten des Lasers, d.h. während der Abarbeitung der ersten Mikrovektoren bleibt der Laser ausgeschaltet. Das LaserOn-Delay kann zu verschiedenen Zwecken eingesetzt werden:

- Um Intensitätsschwankungen entlang der markierten Vektoren im Bildfeld und Einbrenn-Effekte an den Start- und Endpunkten zu vermeiden, sollten die Vektoren in der Bildebene mit möglichst konstanter Geschwindigkeit abgefahren werden.

  Zu Beginn eines Mark- oder Bogenbefehls müssen die Spiegel jedoch zunächst auf die vorgegebene Markiergeschwindigkeit beschleunigt werden (siehe Abb. 34). Um daraus resultierende Einbrenn-Effekte zu vermeiden, sollte für das LaserOn-Delay ein ausreichend hoher (*positiver)* Wert gewählt werden. Dann haben die Spiegel bereits eine gewisse Winkelgeschwindigkeit erreicht, ehe der Laser eingeschaltet wird.

  Andererseits darf das LaserOn-Delay jedoch nicht zu lange gewählt werden, sonst wird der erste Teil des Vektors abgeschnitten.
- Bei einigen Materialien dauert es eine gewisse Zeit, bis sie auf die Bestrahlung reagieren. In diesem Fall kann es sinnvoll sein, den Startpunkt des Vektors vor dem Markieren "vorzuheizen". Dies kann man durch die Wahl eines *negativen* Wertes für das LaserOn-Delay erreichen.
  Dabei sollte man allerdings beachten, dass ein negatives LaserOn-Delay die Gesamtmarkierungszeit verlängert, da es *vor* dem aktuellen Mark- oder Bogenbefehl eingefügt wird. Zudem stellt sich eine automatische Verlängerung des Scanner-Delays ein, wenn ein vorangegangenes LaserOff-Delay noch nicht abgelaufen ist (siehe "Automatische Delay-Anpassung", Seite 121).

(1) Darüber hinaus kann zur Optimierung (für hohe Genauigkeitsanforderungen) eine automatische Nachregelung von "Laser active"-Lasersteuersignalen – und damit der Laserleistung – auch während der Ausführung von Vektor- und Bogenbefehlen mit den Befehlen zur automatischen Lasersteuerung realisiert werden (siehe Seite 159).

LaserOff-Delay

Die Beschleunigungsphase zu Beginn einer Bewegung führt zu einer Differenz zwischen der jeweiligen Soll- und Ist-Position der Spiegel, nämlich dem Schleppverzug (einer Zeitdifferenz zwischen dem geplanten und dem tatsächlichen Erreichen einer bestimmten Position, siehe Abb. 34). Da der Laser nicht *dann* ausgeschaltet werden soll, wenn der *Sollwert*, sondern erst dann, wenn der *Istwert* der Endposition erreicht ist, wird automatisch nach jedem Mark- oder Bogenbefehl ein sogenanntes LaserOff-Delay eingefügt (siehe auch Hinweise Seite 115). Wenn bei kurzen Markvektoren ein vorangegangenes LaserOn-Delay noch nicht abgelaufen ist, wird das LaserOff-Delay vorübergehend automatisch entsprechend verlängert (siehe "Automatische Delay-Anpassung", Seite 121).

7.2.2 Scanner-Delays

Es gibt drei verschiedene Typen von Scanner-Delays: Jump-, Mark- und Polygon-Delays.

Nach jedem Vektor- und Bogenbefehl fügt die RTC5 eines dieser Delays ein, bevor der nächste Befehl gestartet wird.

Der Befehl **set_scanner_delays** definiert die Scanner-Delays. Die Scanner-Delays werden in Einheiten von **10 µs** angegeben.

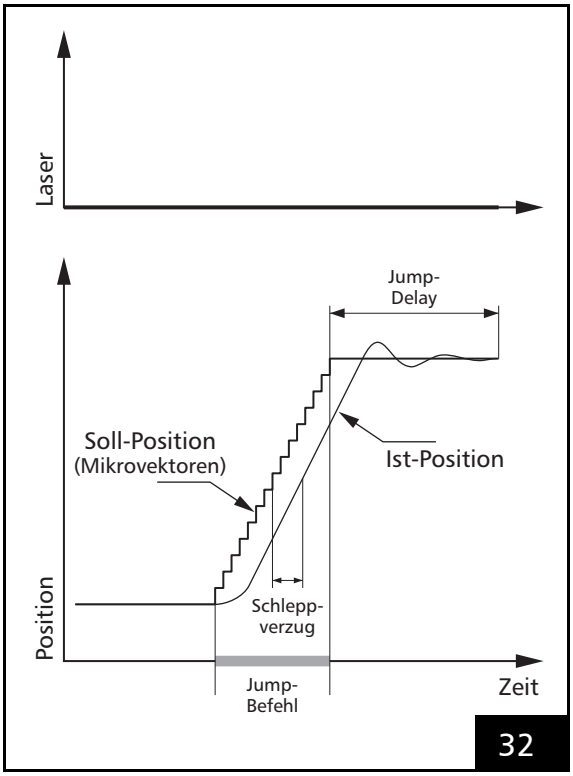
Jump-Delay

Zu Beginn der Ausführung eines Jump-Befehls müssen die Spiegel erst auf die vorgegebene Sprunggeschwindigkeit beschleunigt werden. Dadurch entsteht ein Schleppverzug (siehe Abb. 32).

Am Ende des Sprungs müssen die Spiegel wieder vollständig abgebremst werden. Aufgrund ihrer Trägheit kommt es dabei zu einem gewissen Einschwingen um den Zielpunkt. Um dem Schleppverzug und der notwendigen Einschwingzeit Rechnung zu tragen, wird von der RTC5 automatisch nach jedem Jump-Befehl (dagegen nicht nach **goto_xy** oder **goto_xyz**) ein sogenanntes Jump-Delay eingefügt (siehe Abb. 32).

Die Länge des Jump-Delays muss von Benutzern über den Listenbefehl **set_scanner_delays** definiert werden. Dabei ist zu beachten, dass bei einer größeren Sprunggeschwindigkeit auch ein längeres Jump-Delay notwendig ist.

Die Gesamtzeit für einen Sprung, die sich aus Sprungzeit und Länge des Jump-Delays zusammensetzt, kann durch optimale Abstimmung von Sprunggeschwindigkeit und Länge des Jump-Delays minimiert werden.



Scan-Kopf-Steuerung während eines Jump-Befehls mit Jump-Delay. Der Laser wird nicht eingeschaltet.

Variable Jump-Delays

Während eines Sprungvektors bewegt sich der Laserfokus üblicherweise mit einer konstanten linearen Geschwindigkeit, der Sprunggeschwindigkeit. Da die Sprunggeschwindigkeit für alle Sprungvektoren die gleiche ist, wird ein konstantes Jump-Delay entsprechend der Einschwingzeit der Spiegel benötigt.

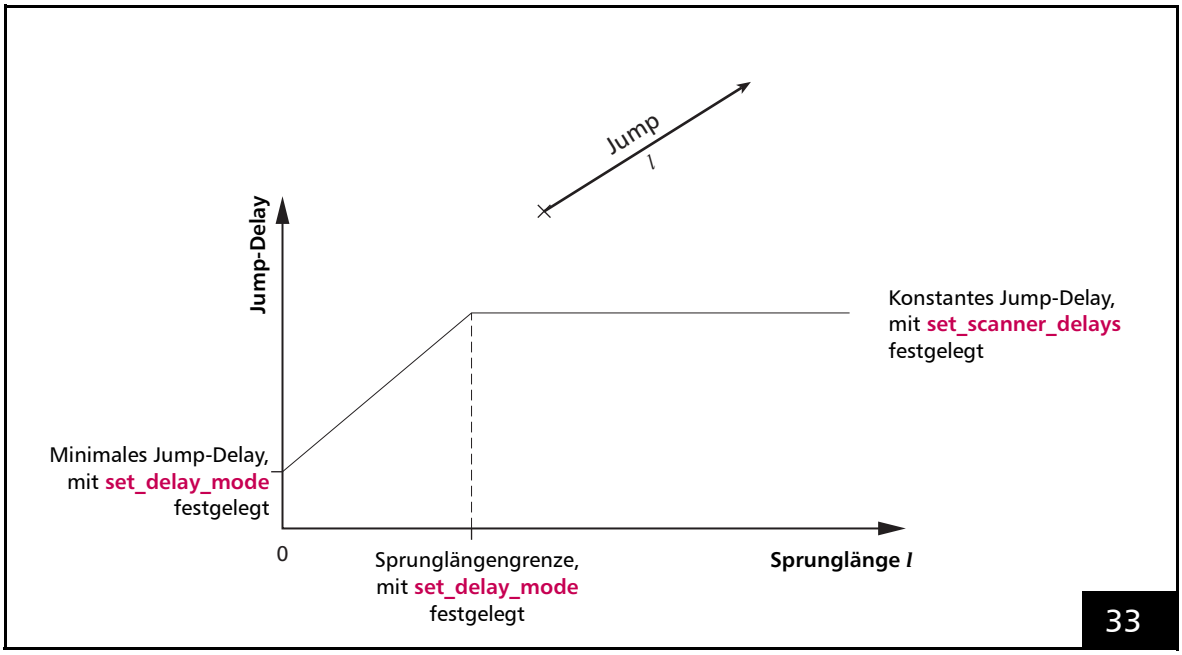
Bei sehr kurzen Sprungvektoren erreicht der Scanner jedoch unter Umständen aufgrund der Massenträgheit der Spiegel während des Sprungs nicht die volle Sprunggeschwindigkeit. In diesem Fall wäre auch ein kürzeres Jump-Delay ausreichend.

Um diese Tatsache auszunützen, gibt es bei der RTC5 einen variablen Jump-Delay-Modus. In diesem Modus wird das Jump-Delay für kurze Sprungvektoren reduziert, wie es in Abb. 33 dargestellt ist. Das minimale Jump-Delay (für einen Sprungvektor der Länge Null) und die Sprunglängengrenze (minimale Sprunglänge, ab der das volle Jump-Delay eingefügt wird) werden durch den Benutzer mit dem Befehl **set_delay_mode** oder **set_delay_mode_list** eingestellt.

Im variablen Jump-Delay-Modus reduziert sich die Gesamtmarkierungszeit insbesondere bei Anwendungen mit vielen kurzen Sprüngen.

Hinweise

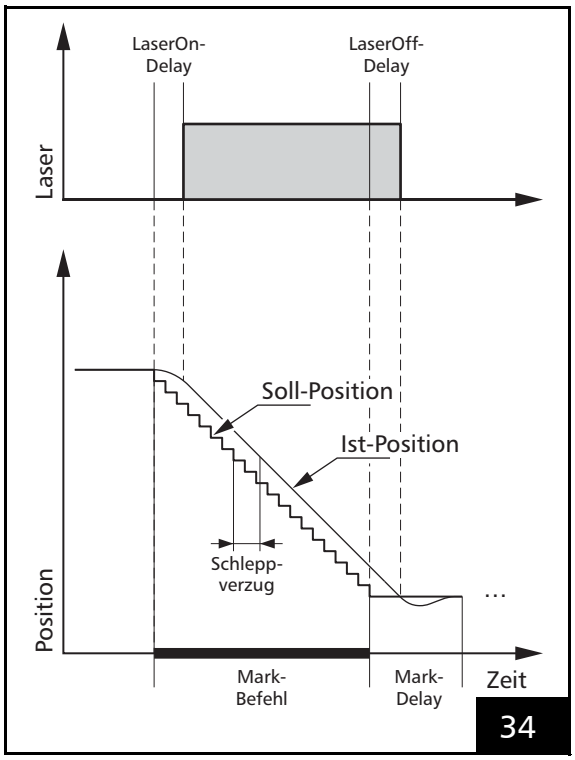
- Um den variablen Jump-Delay-Modus abzuschalten, reicht es, die Sprunglängengrenze mit dem Parameter `JumpLengthLimit` auf Null zu setzen.
- Bei einem Sprungvektor der Länge Null wird das (variable) Jump-Delay nicht ausgeführt.



Variables Jump-Delay
oben: Länge *l* des Sprungvektors
unten: Variation des Jump-Delays

Mark-Delay

Obwohl die Markiergeschwindigkeit üblicherweise niedriger ist als die Sprunggeschwindigkeit, entsteht auch beim Mark- oder Bogenbefehl ein Schleppverzug und gegebenenfalls ein Einschwingen um den Zielpunkt. Um sicherzugehen, dass die Spiegel vor dem folgenden Befehl ihre Sollposition erreicht haben, wird von der RTC5 nach einem einzelnen Mark- oder Bogenbefehl oder nach dem letzten Mark- oder Bogenbefehl eines Polygonzugs automatisch ein Mark-Delay eingefügt (siehe Abb. 34).



Scan-Kopf- und Laser-Steuerung während eines Mark- oder Bogenbefehls mit Mark-Delay. Im grauschattierten Bereich ist der Laser eingeschaltet.

Hinweise

- Wenn auf einen Mark- oder Bogenbefehl *kein* weiterer Mark- oder Bogenbefehl folgt, wird ein Mark-Delay eingefügt und der Laser (nach einem LaserOff-Delay) ausgeschaltet (siehe Abb. 34).
- Wenn unmittelbar auf einen Mark- oder Bogenbefehl ein weiterer Mark- oder Bogenbefehl folgt, dann wird statt eines Mark-Delays ein Polygon-Delay oder ein variables Polygon-Delay eingefügt (siehe Abb. 35 und Abb. 37). Der Laser bleibt dann eingeschaltet, es sei denn mit **set_delay_mode** oder **set_delay_mode_list** wurde ein entsprechend kleiner Eckwert eingestellt (siehe Seite 118).
- Mark- oder Bogenbefehle der Länge Null, die bei *eingeschalteten* "Laser active"-Lasersteuersignalen (also innerhalb eines Polygonzuges) ausgeführt werden:
  - Sind Kurzbefehle (sofern nicht zeitbemessen mit einer Ausführdauer > 5 µs, s.u.). Sie führen zu keiner Unterbrechung des Polygonzugs, auch wenn mehrere unmittelbar aufeinander folgen.
  - Werden (ab `RTC5OUT.out` Version 515) für die Delay-Berechnung ignoriert: Das Mark-Delay bzw. (variable) Polygon-Delay wird entsprechend derjenigen Befehle ausgeführt, die dem (den) Länge-Null-Befehl(en) unmittelbar vorausgehen bzw. nachfolgen. Wird der Befehl einzeln (also nicht innerhalb eines Polygonzugs) ausgeführt, so wird kein Mark-Delay ausgeführt.
  - Lassen (ab `RTC5OUT.out` Version 518) die Lasersteuersignale unverändert, d.h. der Laser geht nicht an, wenn er noch aus ist und bleibt an, wenn er schon an ist.
- Erfolgt ein Mark- oder Bogenbefehl der Länge Null dagegen bei *ausgeschalteten* "Laser active"-Lasersteuersignalen (also als einzelner Befehl oder am Anfang eines Polygonzuges), dann verhält er sich (ab `RTC5OUT.out` Version 527) wie ein zeitbemessener Mark- oder Bogenbefehl mit einer Ausführdauer von 10 µs (siehe nachfolgender Hinweis).

- Zeitbemessene Mark- oder Bogenbefehle der räumlichen Länge Null verhalten sich (ab RTC5OUT.out Version 526, sofern eine Ausführdauer > 5 µs spezifiziert wird) wie Mark- oder Bogenbefehle mit einer endlichen räumlichen Länge:
  – Die Lasersteuersignale werden gegebenenfalls (am Anfang eines Polygonzugs) eingeschaltet bzw. (am Ende eines Polygonzugs) ausgeschaltet.
  – Mark- und Polygon-Delays werden ausgeführt (ein variables Polygon-Delay ist allerdings immer 0).
  – Die Befehle benötigen mindestens einen 10 µs-Takt zur Ausführung.

Hinweise zu früheren Versionen

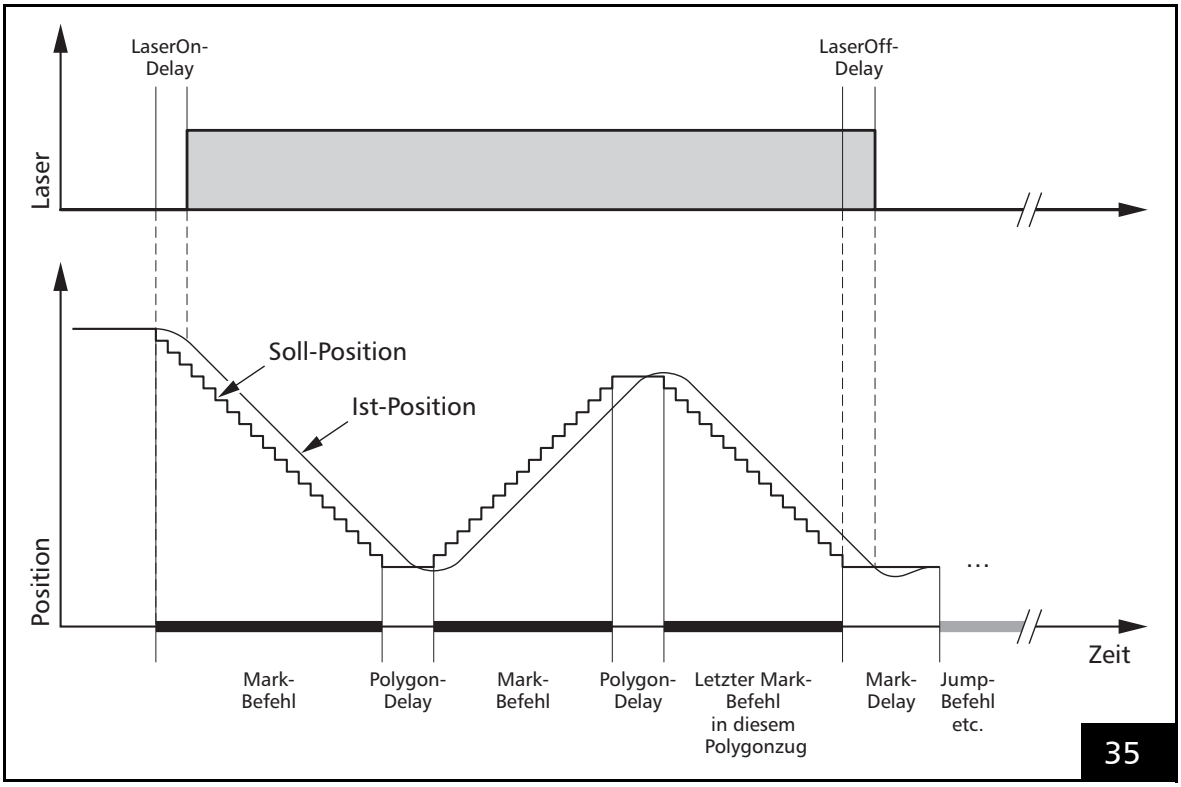
- Bei `RTC5OUT.out` bis Version 514 wird bei einem Mark- oder Bogenbefehl der Länge Null nur das anschließende Mark-Delay bzw. (variable) Polygon-Delay nicht ausgeführt, das vorangehende wird jedoch ausgeführt (ein eventuelles (variables) Polygon-Delay dabei als Mark-Delay).
- Bei `RTC5OUT.out` bis Version 517 wird bei einem Mark- oder Bogenbefehl der Länge Null der Laser wie bei einem Mark- oder Bogenbefehl der Länge größer als Null üblich nach einem LaserOn-Delay eingeschaltet.
- Bei `RTC5OUT.out` bis Version 525 verhalten sich zeitbemessene Mark- oder Bogenbefehle der Länge Null wie nicht-zeitbemessene Mark- oder Bogenbefehle der Länge Null.
- Bei `RTC5OUT.out` bis Version 526 ist das Verhalten bei einem Mark- oder Bogenbefehl der Länge Null unabhängig von den Lasersteuersignalen.

Polygon-Delay

Zwischen zwei *aufeinanderfolgenden* Mark- bzw. Bogenbefehlen ist es nicht immer notwendig, die Bewegung der Spiegel völlig abzubremsen.

Darum wird hier statt eines Mark-Delays ein Polygon-Delay gesetzt, siehe Abb. 35. Der Laser bleibt hier eingeschaltet, es sei denn mit **set_delay_mode** oder **set_delay_mode_list** wurde ein entsprechend kleiner Eckwert eingestellt, siehe Abschnitt "Variables Polygon-Delay", Seite 118.

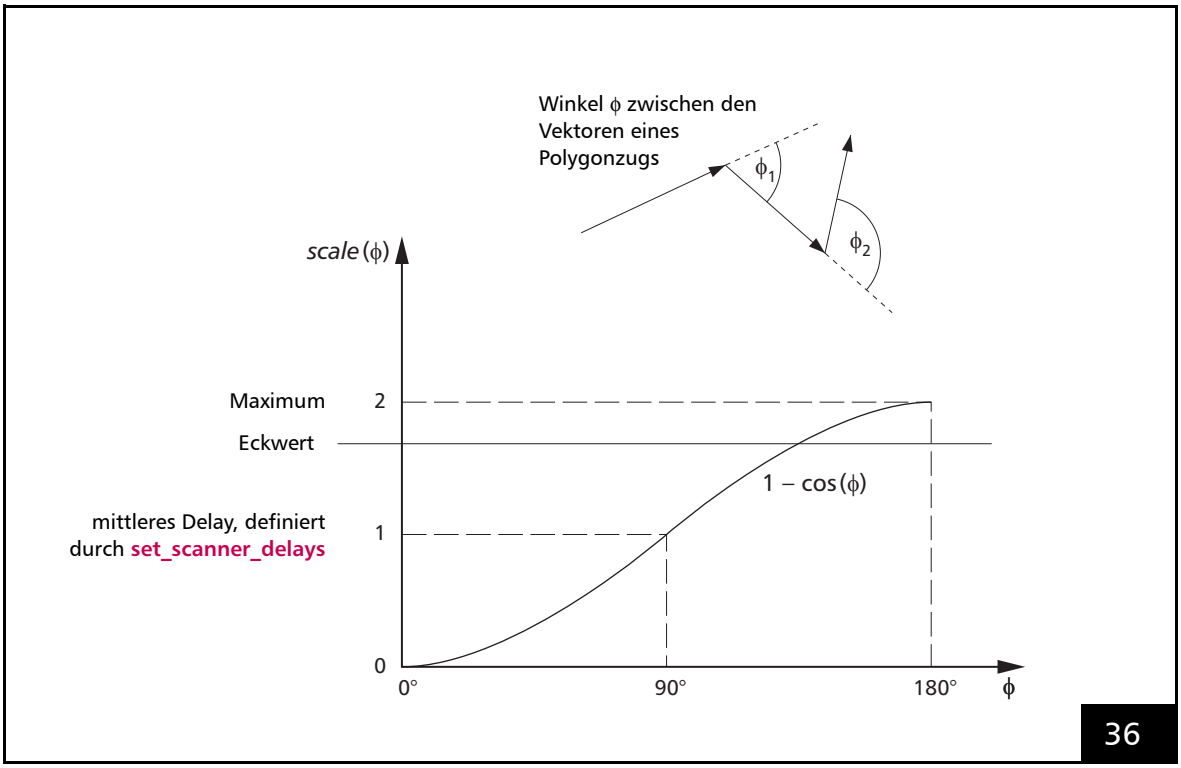
Mark- und Polygon-Delays können unabhängig voneinander definiert werden. Außerdem kann ein variables Polygon-Delay eingestellt werden, das von der RTC5 automatisch entsprechend dem Winkel zwischen zwei Mark-Vektoren bzw. den Tangenten der Kreisbögen angepasst wird, siehe Abschnitt "Variables Polygon-Delay", Seite 118.



Scan-Kopf- und Lasersteuerung während eines Polygonzugs mit konstantem Polygon-Delay

Variables Polygon-Delay

Mit dem Befehl **set_delay_mode** oder **set_delay_mode_list** kann der variable Polygon-Delay-Modus aktiviert werden. Die RTC5 führt dann eine Variation der Länge des Polygon-Delays in Abhängigkeit von der Größe des Winkels ϕ zwischen zwei aufeinanderfolgenden Mark-Vektoren aus (siehe Abb. 36).



Variables Polygon-Delay
oben: Definition des Winkels ϕ
unten: Variation des Polygon-Delays (standardmäßig vorgegebene Kurve)

Für jede Ecke eines Polygonzugs berechnet die RTC5 das variable Polygon-Delay *v_delay* (ϕ) wie folgt:

$$v_delay(\phi) = scale(\phi) \times polygon_delay.$$

Dabei ist *scale* (ϕ) eine Skalierungsfunktion ($0 \leq scale(\phi) \leq 2$). Der Parameter *polygon_delay* wird auch hier über den Befehl **set_scanner_delays** eingestellt. Abb. 36 zeigt die Standard-Skalierungsfunktion. Diese Standardkurve kann durch eine benutzerdefinierte Kurve ersetzt werden (siehe "Benutzerdefinierte variable Polygon-Delays" auf Seite 120).

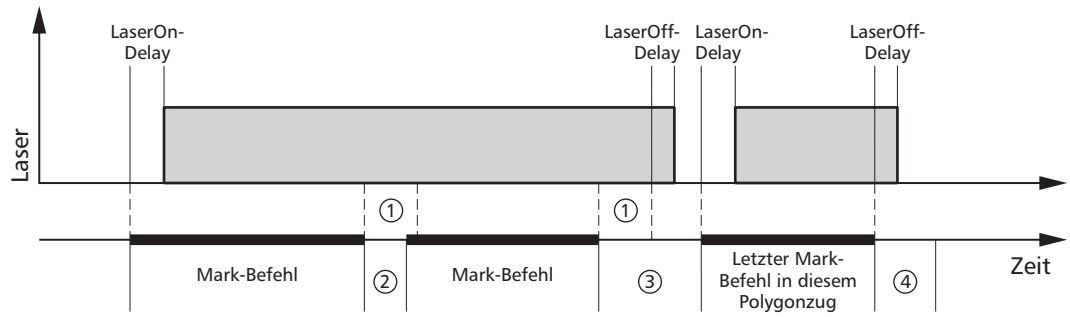
Eckwert

Das variable Polygon-Delay wird bei Winkeln nahe $\phi = 180°$ sehr lang (wie Abb. 36 zeigt).

Das kann unter Umständen zu Einbrenneffekten an scharfen Ecken des Polygonzugs führen.

Um dies zu vermeiden, können Benutzer einen sogenannten *Eckwert (edgelevel)* definieren: Ist das Polygon-Delay zwischen zwei Mark- oder Bogenbefehlen größer als dieser Eckwert, dann schaltet die RTC5 den Laser nach dem ersten Mark- oder Bogenbefehl bei Erreichen des Eckwertes – nach Einfügen eines LaserOff-Delays – aus und beginnt am Anfang des nächsten Mark- oder Bogenbefehls einen neuen Polygonzug (siehe auch Abb. 37). Das Polygon-Delay wird gegebenenfalls verlängert, bis das LaserOff-Delay ganz abgelaufen ist, bevor der Laser wieder eingeschaltet wird.

Weitere Details finden Sie unter **set_delay_mode**.



Legende

1 Eckwert.
2 Variables Polygon-Delay, kürzer als der angegebene Eckwert.
3 Variables Polygon-Delay, länger als der Eckwert.
Der Laser wird zwischen den beiden Mark- oder Bogenbefehlen ausgeschaltet.
4 Mark-Delay.

37

Lasersteuerung während eines Polygonzugs mit variablem Polygon-Delay.
Mit dem Befehl **set_delay_mode** wurde ein Eckwert definiert.

Benutzerdefinierte variable Polygon-Delays

Der Befehl **load_varpolydelay** lädt eine Tabelle für die Skalierungsfunktion *scale*(ϕ) aus einer ASCII-Textdatei. Die Textdatei kann eine oder mehrere Tabellen enthalten.

Jede Tabelle kann bis zu 50 Datenpunkte (ϕ | *scale*(ϕ)) für verschiedene Winkel ϕ enthalten. Aus diesen Daten bestimmt die RTC5 durch lineare Interpolation die Skalierungsfunktion *scale*(ϕ).

Abb. 38 zeigt eine Beispieltabelle mit vier Datenpunkten und die entsprechende Skalierungsfunktion.

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Anweisung (Caption)

  `[VarPolyTable<No>]`

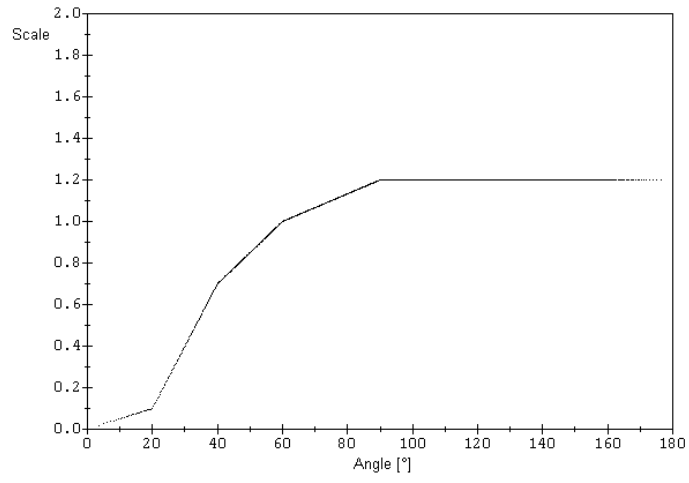
  beginnen. Dabei steht `<No>` für die Tabellennummer (ein nicht-negativer Integer-Wert).
- Enthält die Tabelle mehrere `[VarPolyTable<No>]`-Einträge zur gleichen `<No>`, werden nur die Anweisungen nach dem ersten Eintrag verwendet, diejenigen nach weiteren Einträgen dagegen ignoriert. Es werden nur die Anweisungen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (ϕ | *scale*(ϕ)) wird durch zwei Anweisungen dargestellt:

  ```
  Angle<n> = <Value>
  Scale<n> = <Value>
  ```

  Dabei entspricht `<n>` dem Index des Datenpunktes (Integer-Wert, 1 ≤ `<n>` ≤ 50). Die Werte `<Value>` für den Winkel ϕ (in Grad) und den Skalierungsfaktor können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Verwenden Sie Punkte (.) als Dezimal-Trennzeichen.
- 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 ±0.01° geprüft.
- Für `<Value>` gelten die folgenden Wertebereiche:

  $0.0° \le \phi \le 180.0°$ und $0.0 \le scale(\phi) \le 2.0$
- 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.

```
; sample
[VarPolyTable1]
Angle1 = 20
Scale1 = 0.1
Angle2 = 40
Scale2 = 0.7
Angle3 = 60
Scale3 = 1.0
Angle4 = 90
Scale4 = 1.2
```



38

Beispieltabelle und zugehörige Skalierungsfunktion *scale*(ϕ).

- 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 der Befehl **load_varpolydelay** wirkungslos (Rückgabewert 1 oder 13).
- Der Winkel $\phi = 0°$ bedeutet, dass zwei aufeinanderfolgende Vektoren parallel verlaufen und in die gleiche Richtung markiert werden.
  Enthält die Tabelle keinen Eintrag für $\phi = 0°$ (die Gleichheit wird auf $\pm 0.01°$ geprüft), so wird ein Datenpunkt für $\phi = 0°$ mit dem Skalierungsfaktor *scale*(0°) = 0 hinzugefügt.
- Der Winkel $\phi = 180°$ bedeutet, dass zwei aufeinanderfolgende Vektoren in die entgegengesetzte Richtung markiert werden.
  Enthält die Tabelle keinen Eintrag für $\phi = 180°$ (die Gleichheit wird auf $\pm 0.01°$ geprüft), so wird ein Datenpunkt für $\phi = 180°$ mit dem größten in der Tabelle vorhandenen Skalierungsfaktor eines gültigen Datenpunktes für *scale*(180°) hinzugefügt.

Nach der Initialisierung der RTC5 (mit **load_program_file**) verwendet die RTC5 die interne Standard-Tabelle für das variable Polygon-Delay (1–cos(ϕ), siehe Abb. 36). Dies kann alternativ auch erreicht werden mit `Name` = 0 in **load_varpolydelay**.

7.2.3 Hinweise zur Optimierung der Delays

Die Delays werden mit den Befehlen **set_scanner_delays** und **set_laser_delays** eingestellt. Die Delays sollten auf die gewählte Sprung- und Markiergeschwindigkeit abgestimmt sein. Wenn die Delays nicht optimiert sind, ist unter Umständen die Qualität der Scan-Ergebnisse unzureichend und die Scan-Zeit überhöht. Die Skizzen auf Seite 124 bis 126 zeigen verschiedene Effekte von nichtoptimierten Delays für den Schriftzug “RTC”.

Hinweise

- Beachten Sie, dass die Laser-Delays in Einheiten von 0,5 µs, die Scanner-Delays (Jump-, Mark- und Polygon-Delay) dagegen in Einheiten von 10 µs angegeben werden.

Empfohlene Reihenfolge

Als erstes sollten die Laser-Delays optimiert werden. Dabei ist es empfehlenswert, Jump- und Mark-Delays auf einen hohen Wert zu setzen. Die Längen der Laser-Delays haben keinen Einfluss auf die Gesamt-Scan-Zeit, solange positive Werte gewählt werden (siehe auch folgenden Abschnitt).

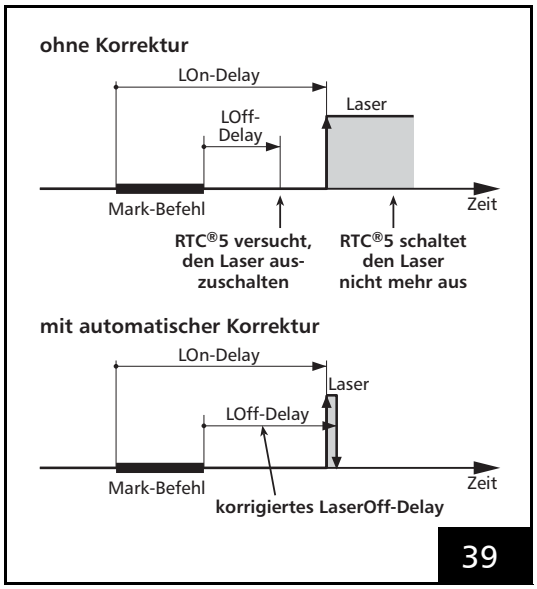
Nachdem die Laser-Delays eingestellt sind, können die Scanner-Delays optimiert werden.

Automatische Delay-Anpassung

Alle Delays können innerhalb ihrer jeweils zulässigen Wertebereiche beliebig gewählt werden. Bei manchen Delays konnte es bei den Vorgängerkarten allerdings (durch “Überkreuzen” von Laser-On und Laser-Off) zu Fehlern bei der Lasersteuerung kommen. Die RTC5 kontrolliert deshalb für jeden Vektor die eingestellten Delays und führt – falls nötig – eine entsprechende Anpassung der Delays für den jeweiligen Vektor durch:

(1) Anpassung des LaserOff-Delays

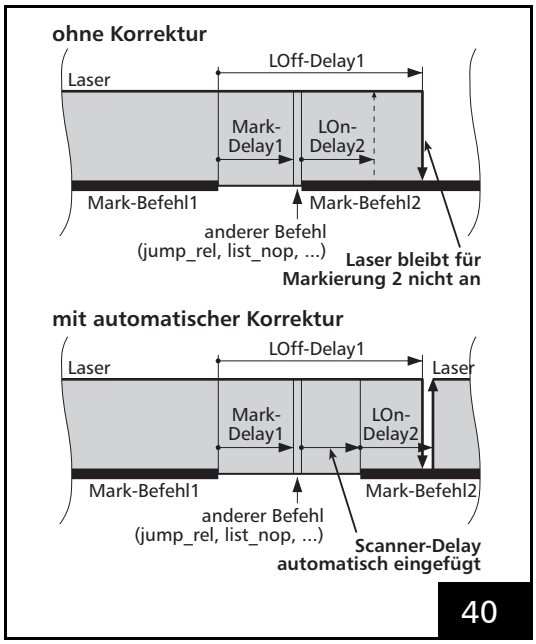
Wird das LaserOff-Delay kürzer als das LaserOn-Delay gewählt (LOffD < LOnD), so konnte es bei den Vorgängerkarten bei sehr kurzen Mark-Vektoren dazu kommen, dass das LaserOff-Delay bereits *vor* dem LaserOn-Delay beendet ist. Diese Karten würden dann versuchen, den Laser auszuschalten, noch ehe sie ihn angeschaltet haben und schalten dann später den Laser ein, ohne ihn anschließend wieder auszuschalten (siehe Abb. 39).



Automatische Anpassung des LaserOff-Delays

Dieses Problem lässt sich einerseits vermeiden, indem das LaserOff-Delay generell *länger* als das LaserOn-Delay gewählt wird (LOffD > LOnD, wie bei den Vorgängerkarten). Andererseits kontrolliert die RTC5 nun für jeden Mark-Vektor die vorgegebenen Laser-Delays und verlängert – falls nötig – automatisch das LaserOff-Delay für den jeweiligen Mark-Vektor so, dass das Laser-Off erst nach dem Laser-On erfolgt. Dadurch sind die Laser-Delays unabhängig voneinander einstellbar. Jeder Mark-Vektor wird dann mindestens mit 0,5 µs markiert.

(2) Anpassung der Scanner-Delays
Auch bei einem zu kurzen Mark-Delay konnte es bei den Vorgängerkarten u.U. zu Fehlern bei der Lasersteuerung kommen, wenn zwischen zwei Mark-Befehlen ein Nicht-Mark-Befehl (z.B. `jump_rel( 0, 0 )` oder **list_nop**) steht. Dann würde der Laser während des zweiten Mark-Befehls ausgeschaltet, aber nicht mehr wie eigentlich vorgesehen wieder eingeschaltet, wenn das LaserOff-Delay länger ist als die Summe aus dem Mark-Delay (und etwaigem Jump-Delay) und dem LaserOn-Delay (siehe Abb. 40).

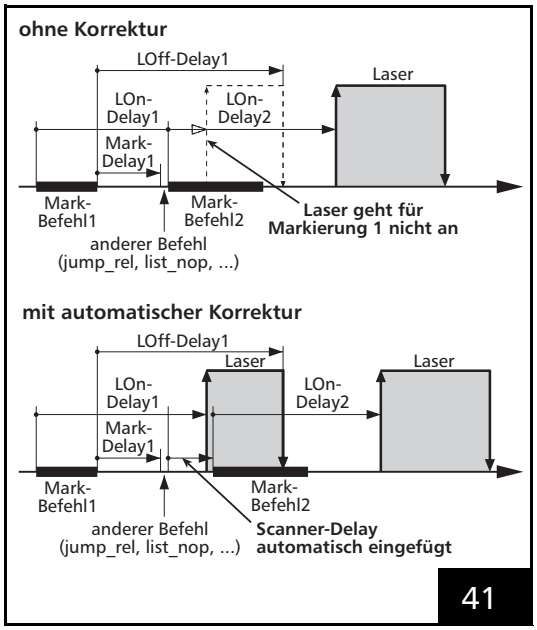


Automatische Anpassung der Scanner-Delays bei kurzen Mark-Delays

Dieses Problem lässt sich einerseits vermeiden, indem das Mark-Delay größer als die Differenz zwischen LaserOff-Delay und LaserOn-Delay gewählt wird (markD>LOffD-LOnD, wie bei den Vorgängerkarten). Dadurch wird aber eine Mindestmaße für das Mark-Delay vorgegeben, die evtl. sonst nicht notwendig wäre, wenn Mark-Befehle unmittelbar aufeinander folgen, und die Gesamtmarkierungszeit unnötig erhöht. Andererseits kontrolliert die RTC5 nun für jeden Mark-Befehl die Laser-Delays und verlängert automatisch das Scanner-Delay so, dass ein vorangegangenes LaserOff-Delay bereits abgelaufen ist, bevor das erneute Laser-On erfolgt. Dadurch wird die Markierungszeit nur im Bedarfsfall erhöht. Die Scanner-Delays können jetzt unabhängig von den Laser-Delays optimiert werden.

Entsprechendes gilt für den Eckwert (edgelevel) des variablen Polygon-Delays und auch für ein zu kurzes Polygon-Delay, wenn eine Polygonkette sich über einen Listenwechsel hinweg fortsetzt, aber kein **auto_change** oder **start_loop** verwendet wird und die zweite Liste sofort nach Abarbeitung der ersten gestartet wird. Im letzteren Fall konnte es bei den Vorgängerkarten vorkommen, dass der Laser für den Rest des Polygonzugs ausgeschaltet blieb.

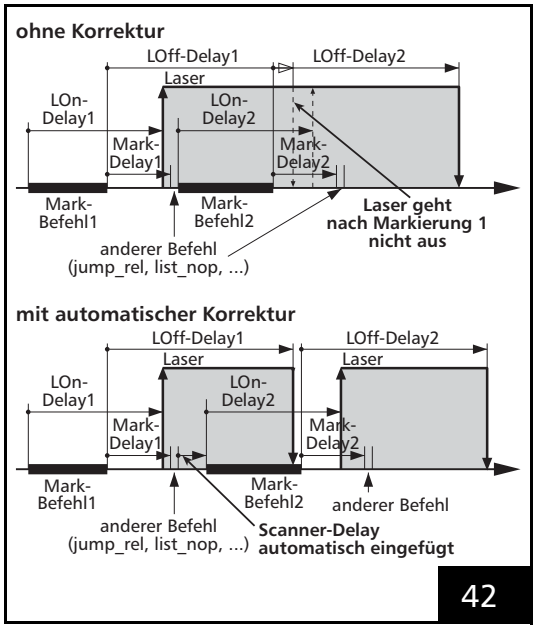
Bei sehr großen LaserOn-Delays, die über einen Markierbefehl und einen nachfolgenden Nicht-Markierbefehl bis zum nächsten Markierbefehl hinausgehen, werden entsprechende Scanner-Delays eingefügt, um ein Überschneiden eines noch nicht abgelaufenen LaserOn-Delays mit dem Beginn des zweiten Markierbefehls zu verhindern. Dadurch würde nämlich das aktuell ablaufende LaserOn-Delay überschrieben und der Laser ginge für den ersten Markierbefehl gar nicht erst an, sondern erst für den zweiten und auch erst nach dem Ablaufen des LaserOff-Delays (siehe Abb. 41).



Automatische Anpassung der Scanner-Delays bei großen LaserOn-Delays

Analog würde bei sehr großen LaserOff-Delays, die am Beginn eines Nicht-Mark-Befehls gesetzt werden und über den nächsten Markbefehl bis zum erneuten Nicht-Markbefehl hinausgehen, das Überschneiden dazu führen, dass der Laser zwischen den beiden Markbefehlen nicht ausgeht, sondern erst nach dem zweiten Nicht-Markbefehl und auch erst nach Ablaufen des LaserOn-Delays (siehe Abb. 42). Um dieses zu verhindern wird am Beginn des Markierbefehls geprüft, ob der Laser vor dessen Ende bereits ausgehen würde und, falls dies nicht der Fall sein sollte, der Markierbefehl durch ein Scanner-Delay entsprechend weit hinausgezögert.

Die Laserabschaltung bei Polygonzügen mit scharfen Ecken (Eckwert edgelevel) erfolgt aber nur, wenn der Laser für diesen Polygonzug bereits eingeschaltet ist (d.h. wenn alle zuvor initiierten LaserOn-Delays bereits abgelaufen sind und kein LaserOff-Delay mehr aktiv ist).



Automatische Anpassung der Scanner-Delays bei großen LaserOff-Delays

Hinweise

- Die obige Beschreibung gilt für Firmwaredateien (`RTC5RBF.rbf`) in einer Version > 509 (siehe **get_rtc_version**). Verwenden Sie keine älteren Firmware-Versionen mehr. Dort würde der Laser im obigen Beispiel für den ersten Mark-Befehl ausgehen und für den zweiten nicht mehr. Ebenso würde er für den ersten Nicht-Mark-Befehl ausgehen und für den zweiten nicht mehr, u.U. auch nicht für **set_end_of_list**.
- Die Bedingungen an die Scanner-Delays (markD>LOffD–LOnD, edgelevel>LOffD–LOnD, polygonD+LOnD>LOffD), die bei der RTC4 noch explizit eingehalten werden mussten, werden infolge der beschriebenen automatischen Anpassung bei der RTC5 zur Laufzeit immer erfüllt.
- Durch die automatische Anpassung der Scanner-Delays ist die Gesamtmarkierungszeit möglicherweise länger als sie sich aus den vorgegebenen Vektorlängen und Delays allein errechnen würde.

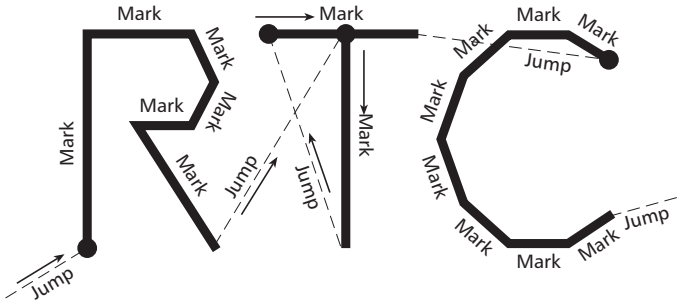
Mögliche Fehler beim Optimieren der Delays

Die folgenden Skizzen zeigen am Beispiel des Schriftzugs “RTC”, wie sich falsch eingestellte Delays auf das Scan-Ergebnis auswirken.

LaserOn-Delay zu kurz

Der Laser wird eingeschaltet, bevor die Spiegel zu Beginn eines Mark-Vektors die notwendige Winkelgeschwindigkeit erreicht haben.

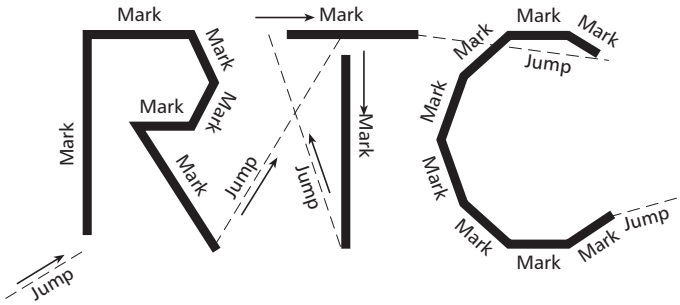
Die Folge sind Einbrenn-Effekte an den Startpunkten der entsprechenden Vektoren.



LaserOn-Delay zu lang

Der Laser wird zu Beginn eines Mark-Vektors zu spät eingeschaltet.

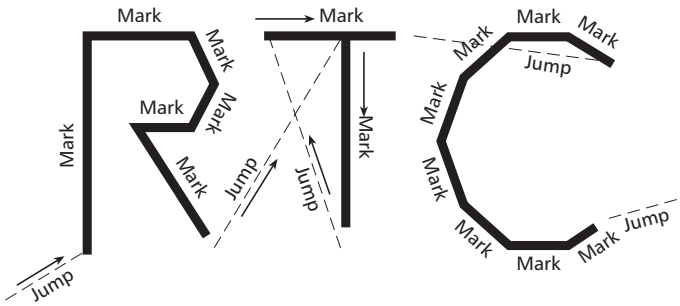
Die entsprechenden Vektoren werden nicht vom Startpunkt an markiert.



LaserOff-Delay zu kurz

Der Laser wird nach einem Mark-Befehl ausgeschaltet, ehe die Spiegel die Vektor-Endposition erreicht haben.

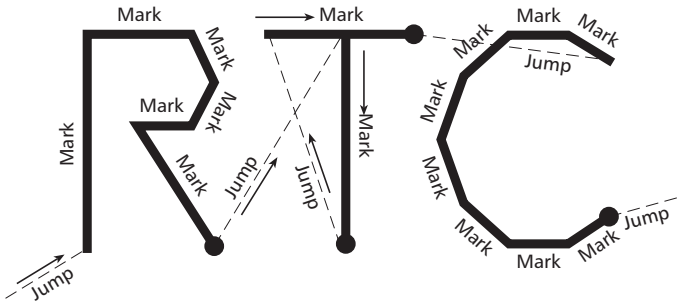
Die entsprechenden Vektoren werden nicht vollständig markiert.



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.

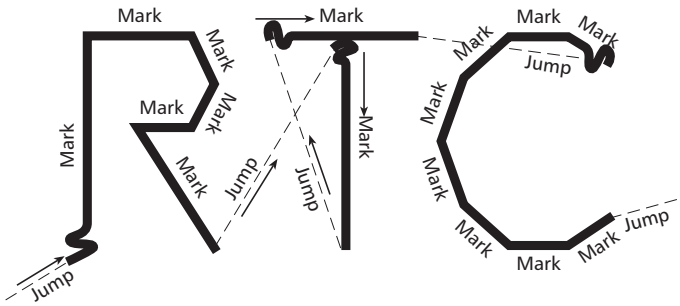
Dies führt zu Einbrenn-Effekten an den jeweiligen Vektorendpunkten.



Jump-Delay zu kurz

Der auf einen Jump-Befehl folgende Vektor wird schon während der notwendigen Einschwingzeit der Spiegel um den Endpunkt des Sprungvektors markiert.

Ein Einlauf oder Überschwinger wird sichtbar.



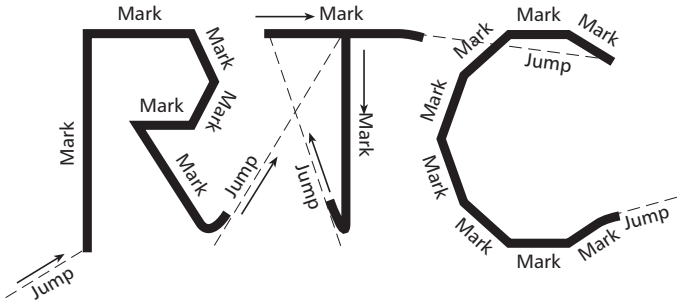
Jump-Delay zu lang

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”.



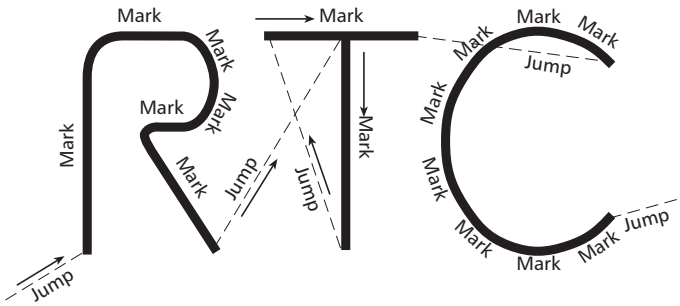
Mark-Delay zu lang

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, ehe die Spiegel die Endposition des vorausgehenden Mark-Vektors erreicht haben.

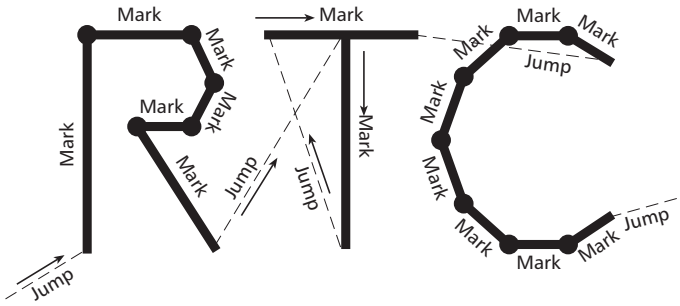
Die Ecken des Polygonzugs werden abgerundet.



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 Vektorendpunkten.



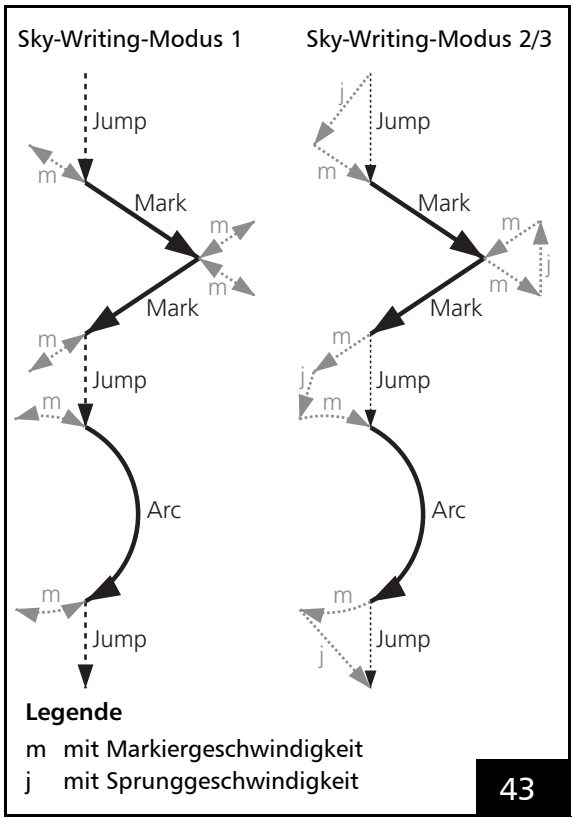
Im variablen Polygon-Delay-Modus kann ein Maximalwert (“Eckwert”) definiert werden (siehe Seite 119).

7.2.4 Sky-Writing

Für Anwendungen mit extremen Genauigkeitsanforderungen kann bei der RTC5 das sogenannte Sky-Writing aktiviert werden. Dadurch wird jeder Mark-Vektor exakt und mit – über die volle Vektorlänge – konstanter Markiergeschwindigkeit ausgeführt.

Bei aktiviertem Sky-Writing führt die RTC5 bei Markierbefehlen vor und nach den auszuführenden Vektoren oder Bögen automatisch nicht-zu-markierende Sky-Writing-Scan-Bewegungen aus (siehe Abb. 43). Über die Parameter der Sky-Writing-Befehle können sowohl die Länge dieser Vor- und Nachlaufbewegungen, als auch die Synchronisation zwischen Scan-Bewegung und Laser-Steuerung eingestellt werden.

Sky-Writing kann im Modus 1, 2 oder 3 aktiviert werden.



Sky-Writing-Bewegungen (grau)
(für Sky-Writing-Modus 3 siehe auch Abb. 44)

Modus 1

Im Sky-Writing-Modus 1 führt die RTC5 bei jedem Markiervektor (oder Bogen) – unabhängig von den vorausgehenden und nachfolgenden Befehlen – die folgenden Sky-Writing-Bewegungen aus:

- In der Vorlaufphase wird dem Vektor (oder Bogen) eine Ausholbewegung der Galvanometerscanner mit Markiergeschwindigkeit vorangestellt (siehe Abb. 43): Die Scanner fahren eine kurze Strecke parallel zum Vektor (bzw. in Verlängerung des Bogens) zunächst in entgegengesetzter Richtung vom Anfangspunkt weg und anschließend wieder bis zum Anfangspunkt zurück.
- Nach Abarbeitung des Vektors (oder Bogens) mit Markiergeschwindigkeit wird dem Vektor (oder Bogen) in der Nachlaufphase eine kurze Abbrems- und Rücklaufbewegung der Scanner (mit Markiergeschwindigkeit) angefügt.

Sky-Writing-Modus 1 kann mit den Befehlen **set_sky_writing_para** oder **set_sky_writing** bzw. mit den entsprechenden Listenbefehlen aktiviert (und auch wieder deaktiviert) werden.

Außer auf Para-Befehle wirkt sich ein aktivierter Sky-Writing-Modus 1 auf alle Mark-, Arc- und Ellipse-Befehle (auch auf zeitbemessene Befehle und 3D-Befehle) aus. Bei der Ausführung von Para-Befehlen (z. B. bei aktivierter vektorkontrollierter Lasersteuerung, siehe Seite 166) und bei der Ausführung von Mikrovektor-Befehlen (siehe Seite 221) wird Sky-Writing-Modus 1 dagegen *nicht* berücksichtigt (aber auch nicht deaktiviert).

Modus 2

Im Sky-Writing-Modus 2 berechnet die RTC5 eine *zeitoptimierte* Sky-Writing-Bewegung. Auch hier wird jedem auszuführenden Vektor (oder Bogen) eine Vor- und Nachlaufbewegung in Verlängerung des Vektors (bzw. Bogens) mit Markiergeschwindigkeit vor- bzw. nachgestellt. Innerhalb einer SkyWriting2-Markiersequenz erfolgt allerdings weder eine Aushol-bewegung der Scanner (in der Vorlaufphase), noch eine Rücklaufbewegung (in der Nachlaufphase). Stattdessen führt die RTC5 direkte Sky-Writing-Sprünge (mit der aktuell definierten Sprunggeschwindigkeit) von Jump-Vektor-Startpunkten zu Vorlauf-Startpunkten, von Nachlauf-Endpunkten zu Vorlauf-Startpunkten und von Nachlauf-Endpunkten zu Jump-Vektor-Endpunkten aus (siehe Abb. 43).

Sky-Writing-Modus 2 kann nur nach erfolgter Aktivierung von Sky-Writing-Modus 1 (s.o.) nachträglich mit **set_sky_writing_mode** oder dem entsprechenden Listenbefehl aktiviert (und deaktiviert) werden.

Ein aktivierter Sky-Writing-Modus 2 wirkt sich auf die gleichen Befehle aus wie Sky-Writing-Modus 1 (außer Ellipse-Befehle, s.o.). *Zeitoptimiertes* Sky-Writing erfolgt allerdings nur innerhalb einer Sequenz von nicht-parametrisierten Mark-, Arc- und Jump-Befehlen (auch zeitbemessene Befehle und 3D-Befehle dürfen enthalten sein). Wird so eine Sequenz durch irgendeinen anderen "nicht-SkyWriting2-fähigen" Listenbefehl unterbrochen (z. B. durch einen Para-Befehl, durch einen Ellipse-Befehl oder durch einen beliebigen Kurz-Befehl), so unterbricht die RTC5 den Sky-Writing-Modus 2 und beendet den vorherigen Befehl im Sky-Writing-Modus 1 (gegebenenfalls mit einer Rücklaufbewegung der Scanner). Sky-Writing-Modus 2 wird durch die Unterbrechung nicht deaktiviert (dies erfolgt nur mit **set_sky_writing_mode** oder **set_sky_writing_mode_list**): Nachfolgende "SkyWriting2-fähige" Befehle werden wieder im Sky-Writing-Modus 2 ausgeführt (der erste Befehl einer solchen nachfolgenden Sequenz aus "SkyWriting2-fähigen" Befehlen erfolgt allerdings zunächst im Sky-Writing-Modus 1, also gegebenenfalls mit einer Ausholbewegung der Scanner).

Ellipse-Befehle werden bei aktiviertem Sky-Writing-Modus 2 immer komplett im Sky-Writing-Modus 1 ausgeführt.

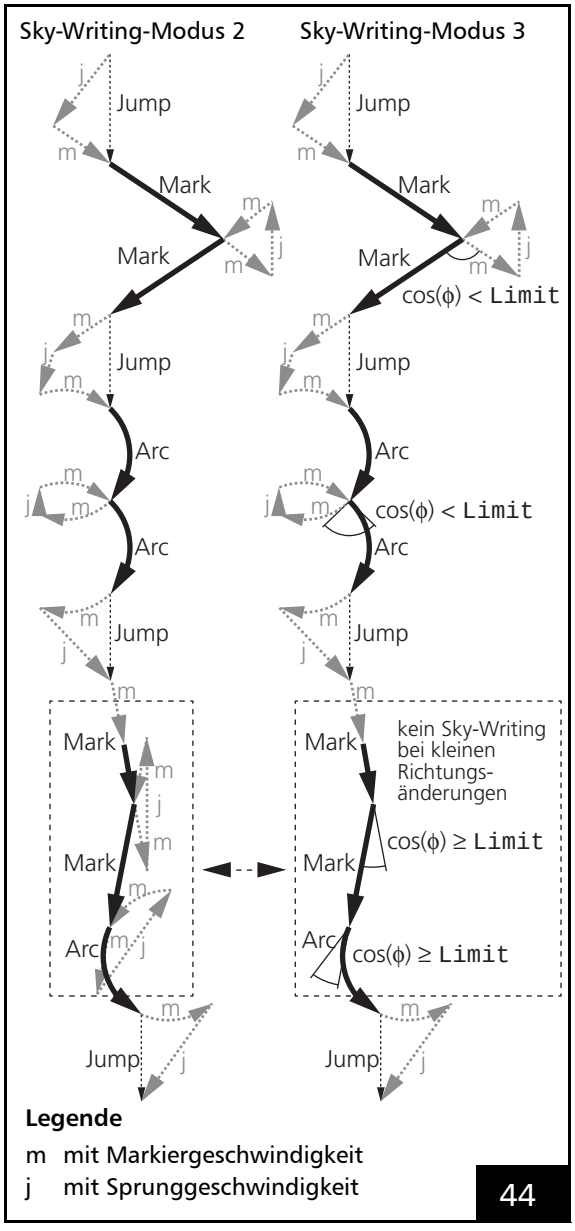
Modus 3

Für Vektoren und Bögen, die innerhalb eines Polygonzugs mit nur geringen Richtungsänderungen aufeinanderfolgen, ist möglicherweise der Zeitverlust durch die Sky-Writing-Bewegung unverhältnismäßig hoch gegenüber dem Genauigkeitsgewinn. Damit das Sky-Writing auch in dieser Beziehung optimiert werden kann (dies reduziert die Gesamtprozesszeit), steht bei der RTC5 zusätzlich der Sky-Writing-Modus 3 zur Verfügung, für den mit **set_sky_writing_limit** oder **set_sky_writing_limit_list** eine Umschaltgrenze `Limit` angegeben werden kann. Nach Aktivieren von Sky-Writing-Modus 3 (mit **set_sky_writing_mode** oder **set_sky_writing_mode_list**) wird (nur) für große Winkelabweichungen zwischen aufeinanderfolgenden Vektoren oder Bögen eines Polygonzugs ($\cos(\phi) < \texttt{Limit}$) eine Sky-Writing-Bewegung wie im Sky-Writing-Modus 2 ausgeführt. Für kleine Winkelabweichungen ($\cos(\phi) \geq \texttt{Limit}$) wird dagegen keine Sky-Writing-Bewegung, sondern ein gegebenenfalls variables Polygon-Delay (siehe Seite 118) ausgeführt. Siehe hierzu auch Abb. 44.

Hinweise:

- Wenn ein Polygonzug mit einem kurzen Mark-Vektor endet (kürzer als Mark-Speed × `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.

Am Anfang und Ende eines Poygonzugs sowie bei Unterbrechung mit "nicht-SkyWriting2-fähigen" Listenbefehlen erfolgt im Sky-Writing-Modus 3 immer eine Sky-Writing-Bewegung (siehe Beschreibung zum Sky-Writing-Modus 2).



Sky-Writing-Bewegungen (grau) für Sky-Writing-Modi 2 und 3

Synchronisation

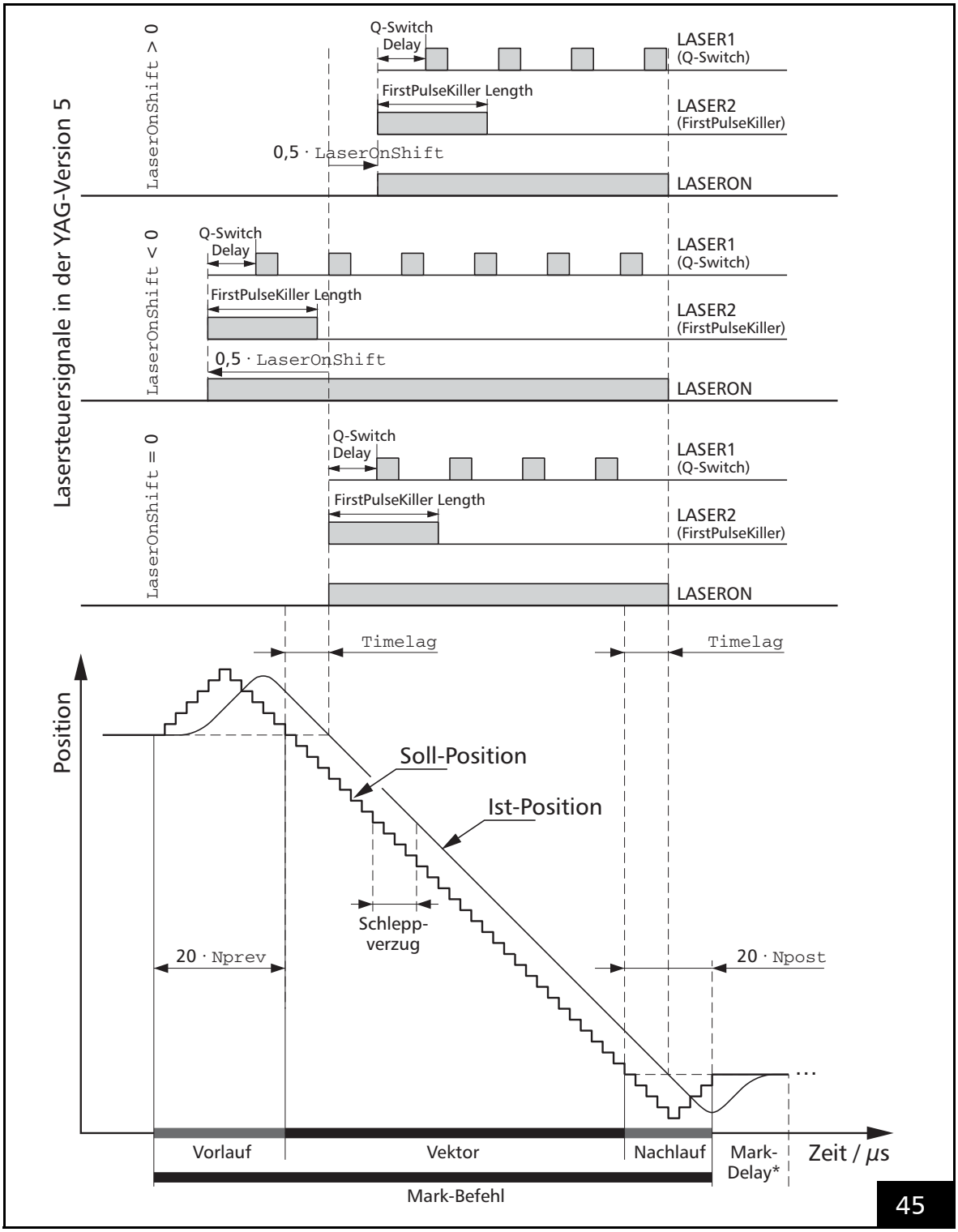
Abb. 45 zeigt das Zeitdiagramm der Scan-Kopf- und Lasersteuerung für Sky-Writing-Modus 1. Das Zeitdiagramm für Sky-Writing-Modi 2 und 3 ist ähnlich, jedoch findet dort im Vorlauf und Nachlauf keine gegenläufige Bewegung der Scanner statt.

Die Parameter `Timelag`, `Nprev`, `Npost` und `LaserOnShift` können mit den Befehlen **set_sky_writing_para**, **set_sky_writing** und den entsprechenden Listenbefehlen eingestellt werden:

- Parameter `Nprev` wird ganzzahlig in 10 µs-Einheiten angegeben und definiert die Länge des Vorlaufs:
  - Vorlaufdauer / µs = 20 × `Nprev` (Modus 1)
  - Vorlaufdauer / µs = 10 × `Nprev` (Modus 2, 3)
- Parameter `Npost` wird ganzzahlig in 10 µs-Einheiten angegeben und definiert die Länge des Nachlaufs:
  - Nachlaufdauer / µs = 20 × `Npost` (Modus 1)
  - Nachlaufdauer / µs = 10 × `Npost` (Modus 2, 3)
- Die Parameter `Timelag` (ganzzahlig in 1 µs-Einheiten) und `LaserOnShift` (ganzzahlig in 0,5 µs-Einheiten) legen die Verzögerung des Einschalt- bzw. Ausschaltzeitpunkts der "Laser active"-Lasersteuersignale relativ zur Soll-Anfangsposition bzw. Soll-Endposition fest:
  - Verzögerung des Einschaltzeitpunkts relativ zur Soll-Anfangsposition / µs
    = `Timelag` + 0,5 × `LaserOnShift`
  - Verzögerung des Ausschaltzeitpunkts relativ zur Soll-Endposition / µs
    = `Timelag`

Für `LaserOnShift` = 0 werden die “Laser active”-Lasersteuersignale mit einer Verzögerung von `Timelag` relativ zur Soll-Anfangsposition bzw. Soll-Endposition an- bzw. 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 “Laser active”-Lasersteuersignale exakt an den Endpunkten des gewünschten Vektors an- bzw. ausgeschaltet werden, dass sich ein konstanter Schleppverzug einstellt, noch ehe der Anfangspunkt des Vektors erreicht ist und dass der Vektor bis zu seinem Endpunkt mit konstanter Markiergeschwindigkeit und daher auch mit konstantem Schleppverzug abgefahren wird.

Der Einschaltzeitpunkt der “Laser active”-Lasersteuersignale kann mit Hilfe des Parameters `LaserOnShift` angepasst werden. Positive Werte verzögern das Einschalten der Lasersteuersignale. Bei negativen Werten werden die Lasersteuersignale früher eingeschaltet, frühestens jedoch zu Beginn des Vorlaufes (ein zusätzliches Delay wie ein negatives LaserOn-Delay – siehe **set_laser_delays** – kann nicht eingestellt werden). Mit negativen Werten kann z. B. in den YAG-Lasersteuerungs-Versionen (mit evtl. voreingestelltem FirstPulseKiller-Signal und Q-Switch-Delay) der Einschaltzeitpunkt der “Laser active”-Lasersteuersignale so weit nach vorne (auch bis in den Bereich der Vorlaufphase) verschoben werden, dass der erste Laserpuls exakt mit der Soll-Anfangsposition erfolgt.



45

Scan-Kopf- und Lasersteuerung im Sky-Writing-Modus 1 (mit Parametern `Timelag`, `Nprev`, `Npost` und `LaserOnShift`)
(Beispielhaft sind die Lasersteuersignale LASER1 und LASER2 für die YAG-Version 5 dargestellt)
(* ein Mark-Delay wird nur u.U. nach der Deaktivierung des Sky-Writing-Modus 1 einmalig eingefügt, siehe Seite 550)

Hinweise

- Mit **set_sky_writing_mode** und **set_sky_writing_mode_list** kann auch zwischen Sky-Writing-Modus 1 und Modus 2 bzw. 3 hin- und hergeschaltet 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: Zuerst sollte der Parameter `Timelag` (mit `LaserOnShift` = 0) so eingestellt 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 dann (bei Bedarf) **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.
- Der Parameter `Timelag` sollte auf den Schleppverzug der Galvanometerscanner eingestellt werden.
- Bei den Befehlen **set_sky_writing** und **set_sky_writing_list** sind `Nprev` und `Npost` fest vorgegeben. Hier gilt:
  - `Nprev` = ca. 0,15 × `Timelag`
  - `Npost` = ca. 0,1 × `Timelag`

  Dies bedeutet für Vor- und Nachlaufdauer:
  - Vorlaufdauer/µs = ca. 3 × `Timelag` (Modus 1)
  - Vorlaufdauer/µs = ca. 1,5 × `Timelag` (Modi 2, 3)
  - Nachlaufdauer/µs = ca. 2 × `Timelag` (Modus 1)
  - Nachlaufdauer/µs = ca. 1 × `Timelag` (Modi 2, 3)

  Bei passendem Parameter `Timelag` sind dies sichere Werte.
- Mit **set_sky_writing_para** oder **set_sky_writing_para_list** können auch kürzere Vor- und Nachlaufphasen vorgegeben werden, um Markierzeit zu sparen. Kürzere Vor- und Nachlaufphasen können insbesondere dann eingestellt werden, wenn unterbrochene Linien in nur einer Richtung markiert und Sprung- und Markiergeschwindigkeit gleich groß eingestellt werden.

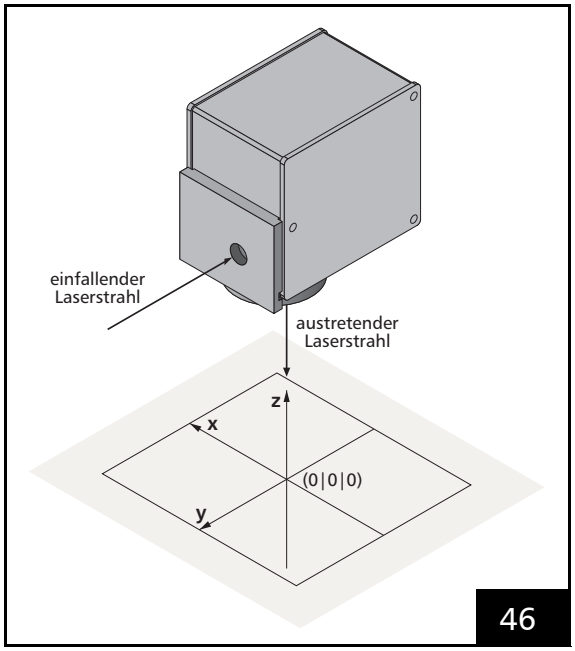
  Dabei sollte man jedoch beachten, dass im Sky-Writing-Modus keine automatische Korrektur der Laser- und Scanner-Delays (wie auf Seite 121 beschrieben) durchgeführt wird:
  - Wenn zu Beginn des nachfolgenden Markierbefehls (Anfang des Vorlaufs) das LaserOn-Delay des vorhergehenden Markierbefehls noch nicht abgelaufen ist, geht der Laser für den vorhergehenden Befehl nicht an (siehe Abb. 41).
  - Wenn zum Ende des nachfolgenden Markierbefehls (Anfang des Nachlaufs) das LaserOff-Delay des vorhergehenden Markierbefehls noch nicht abgelaufen ist, geht der Laser für den vorhergehenden Befehl nicht aus (siehe Abb. 42).
  - 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 und 3 nicht um mehr als (10 × `Npost` – `Timelag`) größer). Andernfalls geht der Laser für diese Markierung nicht an (dies gilt ab Version OUT 535; bei älteren Versionen könnte der Laser bei sehr kurzen Markierbefehlen erst nach dem Ende der Befehlsausführung angehen und dann so lange anbleiben, bis er bei einem der nächsten Befehle wieder explizit ausgeschaltet wird).

- Zuvor für die “Laser active”-Lasersteuersignale definierte Pulslängen und Ausgabeperioden (in den YAG-Versionen 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 u. U. ein vor der Aktivierung definiertes Mark-Delay eingefügt (siehe Seite 131).
- Vor- 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 und 3 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. Im Sky-Writing-Modus 2 und 3 wird das Jump-Delay automatisch intern dynamisch um bis zu `Nprev` reduziert.
- Es können auch zeitbemessene Mark- und Bogenbefehle (siehe Seite 223) im Sky-Writing-Modus ausgeführt werden. Eine kürzere Markierdauer ist hier allerdings auch mit einer höheren Markiergeschwindigkeit und daher auch mit längeren An- und Auslaufstrecken und Beschleunigungsphasen in der Vor- und Nachlaufphase verbunden. `Nprev` und `Npost` können hierfür separat an die vorgegebene Markierdauer angepasst werden (bei **set_sky_writing** muss evtl. `Timelag` an die vorgegebene Markierdauer angepasst werden).
- Bei der Ausführung von Para-Befehlen (z. B. bei aktivierter vektorkontrollierter Lasersteuerung, siehe Seite 166) und bei der Ausführung von Mikrovektor-Befehlen (siehe Seite 221) wird der Sky-Writing-Modus nicht berücksichtigt (aber auch nicht deaktiviert).
- Während Sky-Writing-Modus 2 oder 3 aktiviert ist, werden die beiden folgenden RTC5-Funktionalitäten der 2D-Sprungbefehle **jump_abs** und **jump_rel** deaktiviert:
  - Automatische Tuning-Umschaltung im Sprungmodus (siehe Seite 176)
  - Koordinatentransformationen mit `at_once` = 2 (siehe Seite 185)

7.3 Scan-Kopf-Steuerung

7.3.1 Referenzsystem

Abb. 46 zeigt das Referenzsystem des Bildfelds, das von der RTC5 genutzt wird. 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 RTC5-Koordinaten

7.3.2 Bildfeld-Größe und -Kalibrierung

Die Größe des nutzbaren Bildfelds wird im wesentlichen durch den maximalen Scanwinkel und durch die Brennweite des Objektivs, beziehungsweise durch den Arbeitsabstand (dem Abstand zwischen der Achse des einfallenden Laserstrahls und der Bildfeldebene) bestimmt.

Die x- und y-Koordinaten eines Vektors müssen als 20-Bit-Werte mit Vorzeichen (d.h. als Zahlen zwischen −524.288 und +524.287) an die RTC5 übergeben werden(1), die z-Koordinaten bei einem 3D-System dagegen als 16-Bit-Werte mit Vorzeichen (d.h. als Zahlen zwischen −32.768 und +32.767).

Das Verhältnis zwischen den digitalen Punktkoordinaten [in Bits(2)] und der tatsächlichen Position des Punktes in Millimetern wird durch den Kalibrierungsfaktor *K* definiert.

Bezeichnet man die durch den maximalen Scan-Winkel gegebene Seitenlänge des Bildfelds mit a_0, so errechnet sich ein theoretischer Kalibrierungsfaktor $K_0 = 2^{20}/a_0$ [Bits pro mm(2)]. In der Praxis setzt SCANLAB als Kalibrierungsfaktor *K* einen gerundeten Wert fest, der oberhalb, aber möglichst nahe beim theoretischen Wert liegt. Der von SCANLAB vorgegebene Kalibrierungsfaktor kann mit **get_table_para** oder **get_head_para** aus der verwendeten Korrekturtabelle ausgelesen werden.

Aus dem Kalibrierungsfaktor *K* lässt sich die Seitenlänge *a* des tatsächlich erreichbaren Bildfelds in Millimetern errechnen:

$$a = 2^{20} / K$$

Hinweise

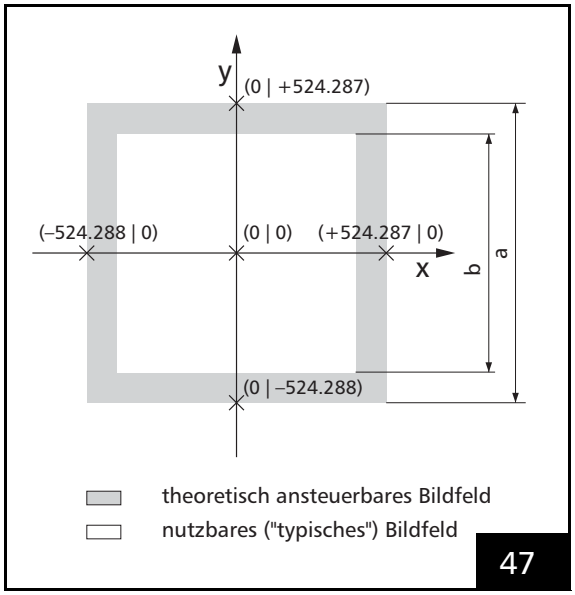
- Der Kalibrierungsfaktor ist für die x- und y-Richtung identisch. Bei 3D-Scan-Systemen gilt $K_z = K_{xy} / 16$.

(1) Eigentlich können x- und y-Koordinaten als 24-Bit-Werte mit Vorzeichen (d.h. als Zahlen zwischen –8.388.608 und 8.388.607) an die RTC5 übergeben werden. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist allerdings nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die jeweils aktuellen Koordinaten werden zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf den Wertebereich [–524.288 … 524.287] (reale Bildfeldgröße) geclippt (siehe auch Seite 135 und Seite 141).

(2) Die Bezeichnung "Bits" ist hier als Synonym für "digitale Ansteuerwerte" zu verstehen (siehe Fußnote auf Seite 103).

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 ansteuerbare Bildfeld.



Bildfeldgröße

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 Abb. 47). 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 Kalibrierungsfaktor *K* und der *halben* Seitenlänge des nutzbaren Bildfelds:

$$M = K \times b/2 \, .$$

RTC4-Kompatibilitätsmodus

Die Bildfeldkoordinaten für die X- und Y-Achse (nicht dagegen für die Z-Achse) und alle damit verbundenen Parameter (z. B. Sprunggeschwindigkeit oder Wobbelamplitude) werden bei der RTC5 (im RTC5-Modus) als 20-Bit-Werte angegeben, im RTC4-Kompatibilitätsmodus dagegen wie bei der RTC4 als 16-Bit-Werte. Im RTC4-Kompatibilitätsmodus werden die angegebenen Werte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Im RTC4-Kompatibilitätsmodus muss entsprechend für x- und y-Koordinatenwerte ein 16-fach kleinerer Kalibrierungsfaktor verwendet werden als im RTC5-Modus. Siehe auch Abschnitt "Erhöhte Parameter-Auflösung", Seite 33.

7.3.3 Virtuelles Bildfeld

Die x- und y-Koordinaten können bei allen Vektor- und Bogenbefehlen und zeitbemessenen Vektor- und Bogenbefehlen (nicht dagegen bei **goto_xy** und **goto_xyz**) als 24-Bit-Werte mit Vorzeichen (d.h. als Zahlen zwischen –8.388.608 und 8.388.607) an die RTC5 übergeben werden. Die jeweils aktuellen Koordinaten werden dann zur Laufzeit – gegebenenfalls nach einer Koordinatentransformation, nach einer "Processing on the fly"-Korrektur und unmittelbar vor dem Anwenden der Korrekturtabelle – auf die reale Bildfeldgröße [–524.288 … 524.287] geclippt (siehe auch Seite 141).

Insbesondere für "Processing on the fly"-Anwendungen steht daher ein virtuelles 24-Bit-Bildfeld zur Verfügung. Es können auch Vektor- und Bogenbefehle für Objekte, die (in "Processing on the fly"-Richtung) bis zu 16 mal größer als das reale Bildfeld sind, geladen werden (für Details siehe Seite 210).

Der erweiterte Wertebereich des virtuellen Bildfelds kann aber auch dazu verwendet werden, um bei eingestellten Koordinatentransformationen (wie Verdrehungen, Verkleinerungen oder Verschiebungen, siehe Seite 183) das volle reale 20-Bit-Bildfeld auszunutzen. Dazu können gegebenenfalls geeignete Koordinatenwerte im virtuellen Bildfeld angegeben werden, die dann durch die Koordinatentransformationen auf Koordinatenwerte im realen Bildfeld transformiert werden (bei den Vorgängerkarten der RTC5 sind bei solchen Koordinatentransformationen gewisse Rand- und Eck-Punkte des realen Bildfelds unerreichbar).

7.3.4 Bildfeldkorrektur und Korrekturtabellen

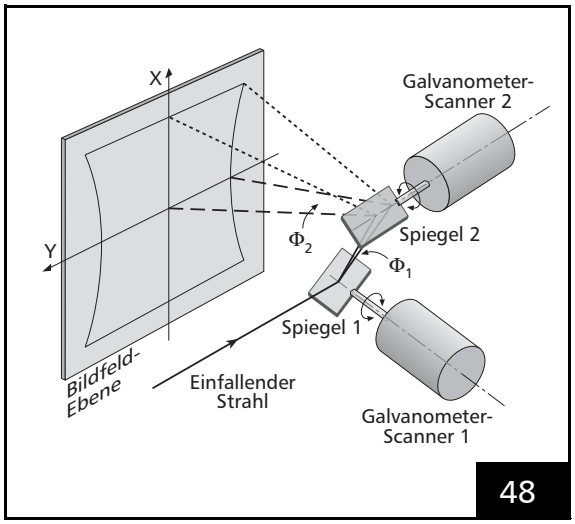
Bildfeldverzerrung

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 Verzerrung des Bildfelds, die in Abb. 48 dargestellt ist. Der Grund dafür ist, dass die Strecke, die der Strahl zwischen Spiegel 1 und Bildfeld zurücklegt, von den Scanwinkeln der beiden Spiegel abhängt.

(2) Die in Folge der Spiegelbewegung auf dem Bildfeld abgefahrene Strecke ist nicht direkt proportional zum Scanwinkel, sondern zum Tangens des Scanwinkels. 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 Bildfeldebene erhält man daher unterschiedliche Spot-Durchmesser.

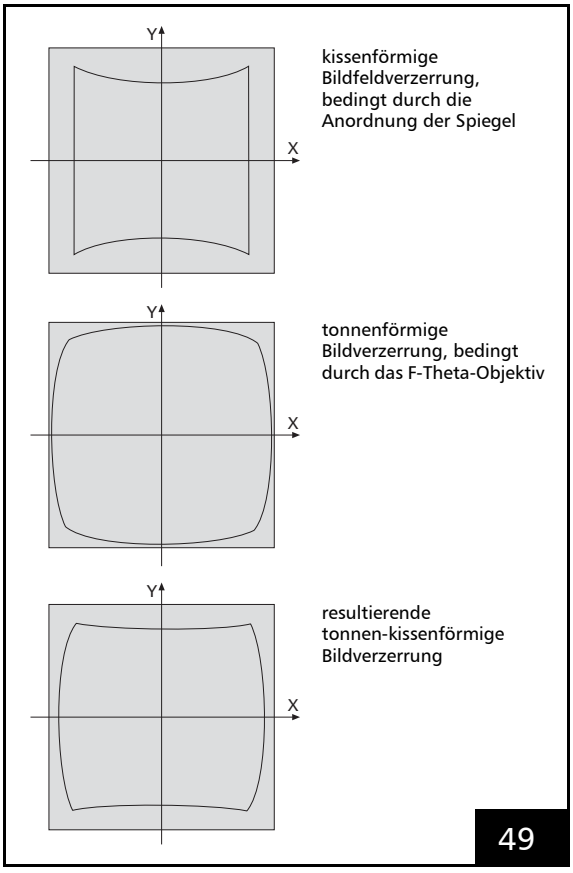


Bildfeldverzerrung bei der Ablenkung eines Strahls in einem Zwei-Spiegel-System

Die beiden letzteren Effekte können vermieden werden, wenn zur Fokussierung des Strahls ein F-Theta-Objektiv eingesetzt wird:

- Man erreicht zum einen eine direkte Proportionalität zwischen dem Scanwinkel und der gescannten Strecke auf dem Bildfeld.
- Zum anderen liegt die Fokusfläche des F-Theta-Objektivs in einer Ebene.

Das F-Theta-Objektiv bewirkt jedoch eine “tonnenförmige” Verzerrung des Bildfelds. Diese Verzerrung wird von der kissenförmigen Verzerrung, bedingt durch die Anordnung der Spiegel, überlagert, was zu einer “tonnen-kissenförmigen” Verzerrung des Bildfelds führt (siehe Abb. 49).



Bildfeldverzerrung, bedingt durch die Anordnung der Spiegel und durch das F-Theta-Objektiv

Feldkorrektur-Algorithmus

Zur Korrektur der oben dargestellten Bildfeldverzerrungen stellt die RTC5 einen Algorithmus zur Verfügung, der auf einer Korrekturtabelle basiert.

Ein quadratisches Gitter von 257 x 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, so berechnet die RTC5 die entsprechenden Punktkoordinaten durch Interpolation aus den angegebenen Gitterpunkten. Die korrigierten Werte werden an den Scan-Kopf ausgegeben. Die Korrektur wird für jeden Mikrovektor einzeln ausgeführt, siehe auch Kapitel 7.1.2 "Mikrovektorisierung", Seite 107.

Standardmäßig berechnet SCANLAB für jedes ausgelieferte System eine individuelle Korrekturtabelle. Sie ist im mitgelieferten Software-Paket in einer Korrekturdatei (`*.CT5`) enthalten. Siehe auch Befehl **load_correction_file**.

Die Korrekturtabelle wird allein aus allgemeinen 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. Dieses generiert RTC-Korrekturdateien aus den Daten einer Testmessung. Nähere Informationen erhalten Sie im zugehörigen Handbuch oder bei SCANLAB.

Aktivieren der Bildfeldkorrektur

Zur Aktivierung der Bildfeldkorrektur müssen die benötigten Korrekturtabellen am Anfang eines Anwenderprogramms (mit **load_correction_file**) aus den jeweiligen Korrekturdateien in den Speicher der RTC5 geladen und anschließend (mit **select_cor_table** oder **select_cor_table_list**) den in dem Anwenderprogramm genutzten Scan-Kopf-Anschlüssen zugewiesen werden (siehe folgende Abschnitte).

Mit 2D-Korrekturtabellen wird eine Bildfeldkorrektur für die x- und y-Koordinaten aktiviert, mit 3D-Korrekturtabellen (sofern die 3D-Option freigeschaltet ist, s.u.) zusätzlich auch für die z-Koordinaten.

2D- und 3D-Korrekturdateien

Die von SCANLAB berechneten 2D- und 3D-Korrekturtabellen werden von SCANLAB in 2D- und 3D-Korrekturdateien ausgeliefert. RTC5-Korrekturdateien 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 beigelegten Readme.txt-Datei und den Parametern im Kopf der Korrekturdatei entnommen werden (siehe auch "Datei-Kopf von Korrekturdateien", Seite 140).

Laden von Korrekturtabellen

Mit **load_correction_file** können bis zu vier Korrekturtabellen aus der jeweiligen Korrekturdatei in den Speicher der RTC5 geladen werden (siehe auch Seite 192).

Wenn die 3D-Option nicht freigeschaltet ist, können nur 2D-Korrekturtabellen geladen werden. Sie können sowohl aus 2D- als auch aus 3D-Korrekturdateien geladen werden (auch 3D-Korrekturdateien enthalten 2D-Korrekturtabellen; der 3D-Anteil der Daten wird gegebenenfalls einfach ignoriert). 3D-Korrekturdateien können so auch für 2D-Scan-Systeme genutzt werden.

Bei freigeschalteter 3D-Option können auch 3D-Korrekturtabellen geladen werden (es können auch 2D- *und* 3D-Korrekturtabellen geladen werden). Eine 3D-Korrekturtabelle kann auch aus einer 2D-Korrekturdatei geladen werden. Die 2D-Korrekturtabelle

wird dabei automatisch um eine lineare z-Korrektur ergänzt. Die tatsächlich passende z-Korrektur kann anschließend mit dem Befehl **load_z_table** geladen werden (siehe Seite 196).

Zuweisung von geladenen Korrekturtabellen

Die Zuweisung von geladenen Korrekturtabellen erfolgt mit **select_cor_table** oder **select_cor_table_list**.

Wenn weder die Option "Zweite-Scan-Kopf-Steuerung" noch die 3D-Option freigeschaltet ist, kann (nur) dem primären Scan-Kopf-Anschluss eine 2D-Korrekurtabelle (eine der bis zu vier geladenen Tabellen) zugewiesen werden.

Ist die Option "Zweite-Scan-Kopf-Steuerung" (aber nicht die 3D-Option) freigeschaltet, kann den beiden Scan-Kopf-Anschlüssen jeweils eine 2D-Korrekturtabelle zugewiesen werden. Es werden dann am sekundären Scan-Kopf-Anschluss die gleichen Signaltypen ausgegeben wie am primären Scan-Kopf-Anschluss. Wahlweise kann beiden Scan-Kopf-Anschlüssen dieselbe Korrekturtabelle oder zwei unterschiedliche Korrekturtabellen zugewiesen werden.

Ist die 3D-Option (aber nicht die Option "Zweite-Scan-Kopf-Steuerung") freigeschaltet, dann kann (nur) dem primären Scan-Kopf-Anschluss eine 2D- oder eine-3D-Korrekturtabelle zugewiesen werden. Bei zugewiesener 2D-Korrekturtabelle gibt der primäre Scan-Kopf-Anschluss Signale für ein XY-Scan-System aus. Bei zugewiesener 3D-Korrekturtabelle werden zusätzlich über beide Kanäle des sekundären Scan-Kopf-Anschlusses Signale für die dritte Achse (Z-Achse) ausgegeben.

Sind *beide* Optionen ("Zweite-Scan-Kopf-Steuerung" *und* 3D) freigeschaltet, können zwei (unterschiedliche) 2D-Korrekturtabellen, aber nicht gleichzeitig zwei 3D-Korrekturtabellen und auch *nicht* gleichzeitig eine 2D- und eine 3D-Korrekturtabelle zugewiesen werden. Zur Ansteuerung eines 3D-Systems muss genau einem der beiden Anschlüsse eine 3D-Korrekturtabelle zugewiesen werden. Über diesen Scan-Kopf-Anschluss werden dann die Signale für ein XY-Scan-System und über beide Kanäle des jeweils anderen Scan-Kopf-Anschlusses die Signale für die dritte Achse (Z-Achse) ausgegeben. Der Benutzer entscheidet bei der Zuweisung der Korrekturtabellen darüber, welche Signale (XY oder Z) an welchem Anschluss ausgegeben werden.

Hinweise

- Wenn keine Korrekturtabelle in den Speicher der RTC5 geladen wurde, dann gibt die RTC5 unvorhergesehene Ansteuerwerte an das Scan-System aus. Falls keine speziell für die jeweilige optische Konfiguration berechnete 2D- oder 3D-Korrekturdatei vorhanden ist, muss deshalb mindestens eine (2D- oder 3D-) 1to1-Korrekturtabelle geladen werden (siehe "1to1-Korrekturtabellen", Seite 139).
  Korrekturdateien sollten zu Beginn eines Anwenderprogramms, vor oder nach dem Laden der Programmdatei (mit **load_program_file**) geladen werden(1). Sie sollten unbedingt geladen und zugewiesen sein, bevor erstmalig eine Liste gestartet wird oder mit einem Kontrollbefehl (wie **goto_xy**) die Galvanometerscanner des Scan-Systems bewegt werden.
- Wird nur einem der beiden Scan-Kopf-Anschlüsse eine 2D-Korrekturtabelle zugewiesen (durch Initialisierung mit **load_program_file** wird beispielsweise nur dem primären Scan-Kopf-Anschluss eine Korrekturtabelle zugewiesen), so gibt der jeweils andere Scan-Kopf-Anschluss keine Signale aus.
- Ab Version DLL 521, OUT 521 ruft **load_correction_file** nach dem Laden einer Korrekturtabelle automatisch **select_cor_table** auf. Dadurch wird auch gleich eine Korrekturtabelle zugewiesen und ein Sprung mit `jump_speed` (kein *harter* Sprung) auf die korrigierte Galvanometerposition ausgelöst.
  – Wird **load_correction_file** aufgerufen, bevor mit **load_program_file** eine Programmdatei geladen wurde, dann bleibt der automatische Aufruf von **select_cor_table** allerdings ohne Wirkung(1). Und **load_program_file** weist zwar – wie mit `select_cor_table(1,0)` – eine Korrekturtabelle zu, führt aber keine Scannerbewegung zur korrigierten Ausgabeposition durch. Unmittelbar nach dem Befehl **load_program_file** steht der Laserfokus daher zunächst – unabhängig von einer eventuell zuvor mit **load_correction_file** geladenen Korrekturdatei #1 – im Zentrum des Bildfeldes (0|0). Die vorgesehene Bildfeldkorrektur wird erst durch einen nachfolgenden **select_cor_table**-, **select_cor_table_list**-, Sprung- oder Markierbefehl wirksam.

(1) Die Korrekturtabellen Nummer 3 und 4 werden durch **load_program_file** gelöscht.

  - Wird **load_correction_file** *nach* **load_program_file** aufgerufen, dann erfolgt der **select_cor_table**-Aufruf mit den Standard-Parameterwerten (`HeadA` = 1, `HeadB` = 0) bzw. – falls zwischenzeitlich **select_cor_table** oder **select_cor_table_list** aufgerufen wurde – mit den dort verwendeten Parametern. Soll eine andere Zuweisung erfolgen, muss **select_cor_table** oder **select_cor_table_list** (erneut) mit den entsprechenden Parameterwerten aufgerufen werden. Ab Version DLL 527, OUT 529 kehrt der **load_correction_file**-Befehl erst zur Anwendung zurück, wenn der durch **select_cor_table** ausgelöste Sprung auf die korrigierte Galvanometerposition abgeschlossen ist.

- RTC5-Korrekturdateien unterscheiden sich von den Korrekturdateien für die früheren RTC-Versionen (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 "Konvertierung von Korrekturdateien", Seite 141.

Hinweise für ältere Versionen als DLL 521, OUT 521

- Hier ruft **load_correction_file** kein **select_cor_table** auf. Hier sollte unmittelbar nach dem Laden einer Korrekturdatei (bzw. nach einem anschließenden **load_program_file**-Befehl) **select_cor_table** oder **select_cor_table_list** aufgerufen werden, damit die aktuelle Position auch sofort gemäß der zuletzt geladenen Korrekturtabelle (mit `jump_speed`) eingestellt wird und dies nicht erst beim nachfolgenden Sprung- oder Markierbefehl durch einen harten Sprung nachgeholt wird. Bei manchen Karten können initialisierungsbedingt ohne expliziten Aufruf von **select_cor_table** bzw. **select_cor_table_list** auch unerwartete Galvanometerscanner-Positionen ausgegeben werden.
- Bei solchen älteren Versionen muss auch, wenn mit **load_correction_file** eine bereits geladene und zugewiesene 2D-Korrekturtabelle durch eine 3D-Korrekturtabelle ausgetauscht wird (bzw. eine 3D- durch eine 2D-Tabelle oder durch eine andere 3D-Tabelle), nach dem Laden in jedem Falle **select_cor_table** oder **select_cor_table_list** aufgerufen werden, damit die jeweils neu geladene Tabelle vollständig wirksam wird. Dies gilt auch dann, wenn die Korrekturtabelle erst nach **load_program_file** geladen wird (die Initialisierung weist zwar eine Korrekturtabelle wie mit `select_cor_table(1,0)` zu, veranlasst aber keine Scannerbewegung zur korrigierten Ausgabeposition). Bei einem Wechsel zwischen zwei 2D-Korrekturtabellen ist der Aufruf von **select_cor_table** oder **select_cor_table_list** nicht zwingend erforderlich, ist aber trotzdem geboten, um harte Sprünge zu vermeiden.

1to1-Korrekturtabellen

Bei der RTC5 müssen 1to1-Korrekturtabellen im Gegensatz zu den früheren RTC-Versionen nicht mehr aus 1to1-Korrekturdateien geladen werden. Sie können stattdessen mit **load_correction_file** und `Name` = 0 direkt erzeugt werden. Die ausgewählte Korrekturtabelle wird dabei für `Dim` = 3 (bei freigeschalteter 3D-Option) durch eine 3D-1to1-Tabelle, sonst durch eine 2D-1to1-Tabelle ersetzt (siehe Befehlsbeschreibung).

Eine 1to1-Korrekturdatei (`Cor_1to1.ct5`) ist dennoch im Lieferumfang der RTC5 enthalten, weil Anwenderprogramme u. U. einen Namen erfordern und 0 als Namen nicht akzeptieren.

Inverse Tabellen

Das Format der RTC5-Korrekturdateien bietet die Möglichkeit, eine Tabelle zu speichern, die das Umrechnen der rückgelesenen Positionsdaten der Scanner-Achsen auf das Eingabe-Koordinatensystem ermöglicht. Diese wird "inverse Tabelle" genannt (zur Verwendung siehe "Überwachung der Positionierung", Seite 174. Die inverse Tabelle wird bei der Berechnung einer ct5-Korrekturdatei durch SCANLAB automatisch erzeugt, dagegen nicht unter allen Umständen auch bei einer Konvertierung älterer ctb-Korrekturdateien (siehe "Konvertierung von Korrekturdateien", Seite 141). Ist keine gültige inverse Tabelle vorhanden, wird stattdessen eine 1to1-Tabelle verwendet. Damit ist keine exakte Rückrechnung auf unkorrigierte Positionsdaten möglich. Die Gültigkeit und die Berechnungsmethode der inversen Tabelle ist im Kopf der Korrekturdatei vermerkt (siehe Parameter 11, Seite 140).

Datei-Kopf von Korrekturdateien

Im Datei-Kopf von RTC5-Korrekturdateien sind Informationen in Form von 16 Parametern gespeichert. Sie geben Auskunft über den Dateinamen, über enthaltene Daten und über optische Daten der zugehörigen optischen Konfiguration. Einige der aufgelisteten Parameter waren bei der RTC4/RTC3/RTC2 nur einer mitgelieferten ReadMe-Datei zu entnehmen und mussten händisch in das Anwenderprogramm übertragen werden. Bei der RTC5 können die in den Korrekturdateien eingetragenen Parameter mit **get_table_para** (aus den aktuell geladenen Korrekturtabellen) bzw. **get_head_para** (aus den zugewiesenen Korrekturtabellen) ausgelesen und auf diese Weise direkt in ein Anwenderprogramm eingebunden werden. Der Datei-Kopf enthält die folgenden Parameter (Nummerierung wie in den Befehlen **get_table_para** und **get_head_para**):

(0) Typ der Korrekturtabelle
- = 0,0: 2D-Korrekturtabelle
- = 1,0: 3D-Korrekturtabelle

(1) Kalibrierungsfaktor K_{xy} [Bit/mm] (siehe Seite 134)
Dieser Parameter ist in der Regel ein ganzzahliges Vielfaches von 16. Aufgrund der 16-fach höheren Auflösung der RTC5 ist er bei einer ct5-Datei 16 mal so groß wie bei einer entsprechenden ctb-Korrekturdatei.

(2) Brennweite / Arbeitsabstand [mm]
- bei einer Konfiguration mit Scan-Objektiv die effektive Brennweite des Objektivs [mm],
- bei einer Konfiguration ohne Scan-Objektiv: den Arbeitsabstand A [mm] (Abstand zwischen der optischen Achse des auf den ersten Ablenkspiegel einfallenden Laserstrahls und der Bildfeldebene).

(3) Streckfaktor in x-Richtung
bewirkt bei 3D-Vektorbefehlen eine z-Koordinaten-abhängige Bildfeldveränderung.

(4) Streckfaktor in y-Richtung
Siehe Parameter 3.

(5) Koeffizient *A* des Polynoms zur Z-Achsen-Ansteuerung (Offset-Anteil, ±26 Bit)

(6) Koeffizient *B* des Polynoms zur Z-Achsen-Ansteuerung (linearer Anteil, ±11 Bit)

(7) Koeffizient *C* des Polynoms zur Z-Achsen-Ansteuerung (quadratischer Anteil, ±4 Bit)

(8) Nummer der Korrekturdatei
Der Parameter entspricht bei von SCANLAB ausgelieferten Korrekturdateien dem Dateinamen (z. B. 145 für `D2_145.ct5` oder `D3_145.ct5`).

(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 = 1000 * P_{Orig} + P_{Ver} mit
- P_{Orig} = 10.000: Originaldatei (original berechnet)
  P_{Orig} = 20.000: aus ctb-Datei konvertiert
  P_{Orig} = 30.000: aus txt-Datei rekonstruiert
  Durch Manipulation einer Korrekturdatei mit Hilfe der correXion5-Software oder eines anderen Korrekturprogrammes 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 + 1000 * P_I mit
- P_T = Schutzglas-Dicke in mm (mit max. 2 Nachkommastellen)
- P_I = Brechungsindex (mit max. 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 Feldgröße in der Korrekturdatei vorgenommen wurde
- = 0,0: keine Begrenzung der Feldgröße
- = 2,0: mit Begrenzung der Feldgröße

Hinweise

- Eine ct5-Datei, die durch Konvertierung aus einer ctb-Datei erzeugt wurde, enthält nicht alle Parameter-Informationen (s. u.). Das gleiche gilt für eine 1to1-Korrekturtabelle.
- Die Parameter 3-7 sind nur für 3D-Vektorbefehle relevant und sind in 2D-Korrekturtabellen 0.
- Wenn die korrekten Kalibrierungsfaktoren verwendet werden (siehe Seite 194), können bei gleichem 3-Achsen-Scan-System für RTC4-Karten (ctb-Dateien) und RTC5-Karten (ct5-Dateien) dieselben ABC-Koeffizienten (Parameter 5-7) verwendet werden.

Konvertierung von Korrekturdateien

Zur Konvertierung von RTC4-Korrekturdateien (ctb) in RTC5-Korrekturdateien (ct5) und umgekehrt ist im Lieferumfang der RTC5 das Programm `CorrectionFileConverter.exe` enthalten. Der Gebrauch dieses Programms ist in der beigelegten Anleitung beschrieben.

Bei der Konvertierung einer ctb-Datei in eine ct5-Datei wird im Datei-Kopf der erzeugten ct5-Datei ein entsprechender Herkunftsindikator eingetragen (Parameter 10, P_{Orig} = 20.000). Eine solche Konvertierung liefert *keine* vollwertige RTC5-Korrekturdatei. Nach der Konvertierung fehlen im Datei-Kopf eine Reihe von Parameter-Informationen. Diese können aber mit dem Programm `CorrectionFileConverter.exe` nachgetragen werden. Zusätzlich kann es vorkommen, dass die bei der Konvertierung numerisch berechnete inverse Korrekturtabelle nicht das gesamte Scan-Feld abdeckt oder nach der Konvertierung überhaupt keine inverse Korrekturtabelle vorhanden ist (siehe auch Parameter 11, Seite 140).

Die Konvertierung einer ct5-Datei in eine ctb-Datei liefert dagegen eine vollwertige ctb-Korrekturdatei. Die Parameter-Informationen aus dem Datei-Kopf der ct5-Datei sind in der ctb-Datei für 2D-Tabellen nicht mehr enthalten, für 3D-Tabellen nur teilweise.

7.3.5 Ausgabewerte an das Scan-System

Berechnung

Die bei Vektor- und Bogenbefehlen angegebenen x- und y-Koordinatenwerte werden von der RTC5 (gegebenenfalls unter Berücksichtigung des z-Koordinatenwertes) in Ausgabewerte für die Galvanometerscanner des Scan-Systems umgerechnet. Ein angegebener z-Koordinatenwert wird (unter Berücksichtigung der x- und y-Koordinatenwerte) zuerst in eine Fokuslänge und anschließend in einen Ausgabewert für die Z-Achse des Scan-Systems umgerechnet.

Bei der Berechnung der Ausgabewerte aus den angegebenen Koordinatenwerten werden nach erfolgter Mikrovektorisierung die folgenden Korrekturen (in der aufgeführten Reihenfolge, sofern zuvor eingestellt) berücksichtigt:

- Falls mit **set_wobbel** oder **set_wobbel_mode** eine Wobbelbewegung aktiviert wurde, werden die x- und y-Koordinaten entsprechend des aktuellen Wobbelvektors transformiert (siehe Seite 189)(1).
- Falls mit **set_fly_x**, **set_fly_y**, **set_fly_rot** oder **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos** eine “Processing on the fly”-Korrektur aktiviert wurde, wird sie auf die x- und y-Koordinaten angewandt (siehe Seite 199)(2).
- Falls eine Koordinatentransformation zur Ausrichtung des Scan-Systems relativ zum Bildfeld definiert wurde (siehe **set_matrix**, **set_offset**, **set_scale**, **set_angle** und entsprechende Listenbefehle), wird sie auf die x- und y-Koordinaten angewandt (siehe Seite 183).
- Mit **set_offset_xyz** oder **set_offset_xyz_list** bzw. **set_defocus** oder **set_defocus_list** kann zusätzlich ein Offset auf die z-Koordinate bzw. die berechnete Fokuslänge festgelegt werden, der sich dann entsprechend auf den z-Ausgabewert auswirkt (siehe Seite 193).
- Koordinatenwerte, die den realen Bildfeldbereich übersteigen (virtuelles Bildfeld, siehe Seite 135), werden auf die Randwerte des realen Bildfelds (20-Bit) geclippt.

(1) nur bei Markierbefehlen (Mark und Arc), nicht bei Jump-Befehlen oder **goto_xy** und **goto_xyz**

(2) nur bei Markierbefehlen (Mark und Arc) und Jump-Befehlen, nicht bei **goto_xy** und **goto_xyz**

- Entsprechend der mit **select_cor_table** oder **select_cor_table_list** zugewiesenen Korrekturtabelle wird eine 2D- bzw. 3D-Bildfeldkorrektur durchgeführt (siehe Seite 136).
- Falls mit dem Befehl **auto_cal** eine automatische Selbstkalibrierung aktiviert wurde (siehe Seite 224), wird bei der Berechnung der Ausgabewerte gegebenenfalls ein kompensierender "Gain" und "Offset" berücksichtigt (eine Gain- und Offset-Korrektur kann auch mit **set_hi** eingestellt werden, siehe Seite 227).

Überlaufende Ausgabewerte werden gegebenenfalls auf die Randwerte des maximal möglichen Wertebereichs geclippt.

Wertebereich

Die Ausgabewerte für das Scan-System sind (anders als bei der RTC4) für alle Achsen des Scan-Systems 20-Bit-Werte mit Vorzeichen (d.h. digitale Werte zwischen −524.288 und +524.287).

Die RTC5 rechnet dazu (durch Multiplizieren mit 16) die bei 3D-Befehlen als 16-Bit-Werte mit Vorzeichen (d.h. als Zahlen zwischen –32.768 und +32.767) spezifizierten Bildfeldkoordinaten für die Z-Achse und die im RTC4-Kompatibilitätsmodus als 16-Bit-Werte mit Vorzeichen spezifizierten Bildfeldkoordinaten für die X- und Y-Achse in 20-Bit-Ausgabewerte um. Vom Scan-System (direkt oder über den XY2-100-Konverter) rückübertragene 20-Bit-Werte werden dagegen von der RTC5 (auch im RTC4-Kompatibilitätsmodus) nicht verändert.

Wenn das Scan-System zusammen mit einem XY2-100-Konverter angesteuert wird, so rechnet der Konverter für alle Achsen des Scan-Systems die 20-Bit-Ausgabewerte der RTC5 (durch Eliminieren der vier niederwertigsten Bits) in 16-Bit-Werte ohne Vorzeichen (d.h. in digitale Werte zwischen 0 und 65.535) um (solche Ansteuerwerte werden von der RTC4 ausgegeben). Vom Scan-System rückübertragene 16-Bit-Werte rechnet der Konverter (durch Auffüllen mit vier 0-Bits) in 20-Bit-Werte um.

Vorausberechnung und Diagnose

Mit Hilfe des Befehls **get_galvo_controls** können die Ausgabewerte für die angegebenen Koordinaten und Einstellparameter in der momentanen Konfiguration (Korrekturdatei, Koordinatentransformationen) vorab berechnet werden, ohne dass sich die Galvanometerscanner dabei bewegen.

Takt-Überlauf

Bei den sehr rechenintensiven Jump-, Mark- 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 simultane Ansteuerung zweier Scan-Systeme, eine Wobbelbewegung, "Processing on the fly" in zwei Achsen (Korrektur über McBSP/SPI-Schnittstelle), automatische Lasersteuerung, Para-Vektoren, Datenaufzeichnung, variables Polygon-Delay (Scanner-Delay evtl. 0), kurze Listenbefehle (z.B. in einem Polygonzug), bestimmte Kontrollbefehle während der Listenausführung.

Diese Überlauf-Situation kann intern erkannt und gezählt werden. Testen Sie Ihre Anwendung 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 stockt also evtl. für 10 µs.

7.3.6 Statusüberwachung und Diagnose

Zur Überwachung des Betriebsstatus und zu Diagnosezwecken können mit den Befehl **get_value** oder **get_values** eine Reihe von Signalen ausgelesen werden:

- ein 20-Bit-Statuswort, das vom Scan-System (gegebenenfalls über den XY2-100-Konverter) zurückgegeben wird (siehe "Statusinformationen vom Scan-System", Seite 143)
- das aktuelle LASERON-Signal
- die aktuellen kartesischen Ansteuerwerte (d.h. die für beide Scan-Kopf-Anschlüsse noch gemeinsamen sogenannten Sample-Werte, die zwar gegebenenfalls eine Wobbel-Korrektur und eine "Processing on the fly"-Korrektur, aber weder eine Bildfeldkorrektur noch eine Koordinatentransformation berücksichtigen)
- die (für den jeweiligen Scan-Kopf-Anschluss spezifischen) Ansteuerwerte gegebenenfalls unter Berücksichtigung
  - der mit **set_matrix**, **set_offset**, **set_scale** oder **set_angle** (bzw. mit den entsprechenden Listenbefehlen) festgelegten Koordinatentransformationen
  - des mit **set_offset_xyz** oder **set_offset_xyz_list** festgelegten Offsets auf die z-Koordinate,
  - des mit **set_defocus** oder **set_defocus_list** festgelegten Offsets auf die Fokuslänge und
  - der geladenen Korrekturtabellen.

Jedes dieser Signale kann zusätzlich mit dem Befehl **set_trigger**/**set_trigger4** über einen längeren Zeitraum – mit auswählbarer Messperiode – aufgenommen werden. Die aufgenommenen Messwerte werden auf der RTC5 gespeichert. Mit dem Befehl **get_waveform** können sie anschließend zur Auswertung an den PC übertragen werden.

Der aktuelle Status einer durch **set_trigger**/**set_trigger4** gestarteten Messung kann mit dem Befehl **measurement_status** abgerufen werden.

Die rückübertragenen Werte sind stets 20-Bit-Werte. Diese müssen bei Bedarf auch im RTC4-Kompatibilitätsmodus von Benutzern selbst in 16-Bit-Werte umgerechnet (durch 16 dividiert) werden. Auch das XY2-100-Statuswort wird von **get_value**, **get_values** oder **set_trigger**/**set_trigger4** als 20-Bit-Wert zurückgegeben (von **get_head_status** dagegen als 16-Bit-Wert, s.u.).

Statusinformationen vom Scan-System

Über das SL2-100-Protokoll werden (auf dem Rückkanal) die folgenden Signale alle 10 µs vom Scan-System an die RTC5 übertragen:

- Ein 20-Bit-Statuswort:
  - Von den meisten Scan-Systemen wird das XY2-100-Statuswort zurückgegeben(1). Bei i*DRIVE*-Scan-Systemen (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario$SCAN_{de}$) können stattdessen auch andere Datensignale für jede Achse separat an die RTC5 rückübertragen und ausgewertet werden (siehe Kapitel 8.1, Seite 172).
  - Scan-Systeme ohne SL2-100-Interface geben ein 16-Bit-Statuswort zurück. Vom XY2-100-Konverter wird dieses mit 16 multipliziert und als 20-Bit-Statuswort an die RTC5 weitergegeben.
  - Das 20-Bit-Statuswort kann mit **get_value**, **get_values** oder **set_trigger**/**set_trigger4** ausgelesen werden (s.o.).
- 6 zusätzliche Statusbits:
  - Scan-Systeme mit SL2-100-Interface übertragen zusätzlich zum 20-Bit-Statuswort die 6 variablen Bits des XY2-100-Statusworts.
  - Bei einem Scan-System ohne SL2-100-Interface überträgt der XY2-100-Konverter zusätzlich zum 20-Bit-Statuswort Bits#6-11 des vom Scan-System ausgegebenen 16-Bit-Statusworts an die RTC5. Falls das Scan-System das XY2-100-Statuswort ausgibt (bei i*DRIVE*-Scan-Systemen sollte dies entsprechend ausgewählt werden), werden also auch bei einem solchen System die 6 variablen Bits des XY2-100-Statusworts übertragen.
  - Die 6 zusätzlichen Statusbits werden von **get_head_status** ausgelesen und als (16-Bit-)XY2-100-Statuswort zurückgegeben (siehe Befehlsbeschreibung).
- ein User-Datenbit
  - i*DRIVE*-Scan-Systeme mit SL2-100-Interface (die ohne XY2-100-Konverter betrieben werden) können über dieses User-Datenbit, parallel zur oben beschriebenen "normalen" Status-

(1) Die Statussignale des XY2-100-Statusworts können auch für eine automatische Lasersignalunterdrückung verwendet werden (siehe Seite 146).

rückgabe weitere, sich langsam verändernde Statusinformationen (z. B. die aktuelle Temperatur der Galvanometerscanner) rückübertragen. Diese Statusinformationen können mit **read_user_data** ausgelesen werden (siehe auch Seite 173).

7.4 Lasersteuerung

Die RTC5 stellt über die LASERON-, LASER1- und LASER2-Ausgänge der 15-poligen Sub-D-LASER-Buchse (siehe Seite 47) bzw. der EXTENSION 2-Stiftleiste (siehe Seite 52) mehrere Lasersteuersignale mit programmierbarer Pulslänge und Periodendauer bereit. Damit lassen sich verschiedene Lasertypen ansteuern.

Mit dem Befehl **set_laser_mode** können Benutzer einen von sieben verschiedenen Lasersteuerungs-Versionen auswählen:

- Die CO_2-Version (Lasermode 0) ist für die Ansteuerung eines CO_2-Lasers konzipiert.
- Die YAG-Versionen (Lasermodi 1, 2, 3 und 5) sind als Steuerungsvarianten für die Familie der Nd:YAG-Laser und verwandte Laser konzipiert.
- Im Lasermode 4 und Lasermode 6 wird ein freilaufendes Steuersignal bereitgestellt.

Abhängig von der gewählten Version sind verschiedene Lasersteuersignale für den “Laser active”-Betrieb und den “Laser standby”-Betrieb verfügbar. Die verschiedenen Laser-Versionen werden in den folgenden Abschnitten ausführlich beschrieben.

Alle Lasersteuersignale sind TTL-Signale und können mit dem Befehl **set_laser_control** wahlweise auf high-aktiv oder auf low-aktiv gesetzt werden (siehe auch Seite 47). Wie sie aktuell gesetzt sind, kann mit dem Befehl **get_startstop_info** (Bit#13) ausgelesen werden.

Alle Lasersteuersignale sind bis zu 20 mA belastbar.

7.4.1 Freigeben, Aktivieren und Schalten der Lasersteuersignale

Nach einem Hardware-Reset und auch nach der Initialisierung mit **load_program_file** sind alle Lasersteuersignale unterdrückt: Alle Lasersignal-Ausgänge (LASERON, LASER1 und LASER2) befinden sich im (hochohmigen) tristate-Zustand. TTL-Zustände (LOW oder HIGH) stellen sich erst dann ein, wenn die Signale mit **set_laser_control**-Befehl freigegeben werden (und dabei der gewünschte TTL-Pegel festgelegt wird).
Nach Aufruf des **set_laser_control**-Befehls

- werden an den Ausgängen Lasersteuersignale für den “Laser active”-Betrieb ausgegeben, sobald diese *aktiviert und eingeschaltet* werden,
- werden an den Ausgängen Standby-Signale ausgegeben, wenn sie aktiviert werden und falls keine Lasersteuersignale für den “Laser active”-Betrieb eingeschaltet sind, oder
- befinden sich die Ausgänge im TTL-Grundzustand (z. B. LOW wenn ein hoher Signal-Pegel ausgewählt wurde), falls keine Lasersteuersignale (weder für den “Laser active”- noch für den “Laser standby”-Betrieb) aktiviert sind.

Nach Aufruf des **set_laser_control**-Befehls stellt sich ein (hochohmiger) tristate-Zustand der Lasersignal-Ausgänge (LASERON, LASER1 und LASER2) erst wieder durch einen erneuten Hardware-Reset ein oder wenn die Lasersteuersignale nachträglich durch einen Befehl unterdrückt werden (siehe **disable_laser**, **pause_list** und **set_laser_control**(Bit#2 = 1)).

Vor einer Aktivierung der Lasersteuersignale sollte mit **set_laser_mode** die Lasersteuerungs-Version ausgewählt werden (sonst wird per Default die CO_2-Version ausgewählt).

Zum erstmaligen Aktivieren der Lasersteuersignale (sowohl für den “Laser active”-Betrieb als auch den “Laser standby”-Betrieb) muss der Befehl **set_laser_control** verwendet werden, um die Lasersignal-Ausgänge (LASERON, LASER1, LASER2) überhaupt freizugeben.

“Laser active”-Signale

Die Signale für den “Laser active”-Betrieb können nach **set_laser_control** mit **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** aktiviert bzw. deaktiviert werden. Zusätzlich können sie mit **set_laser_control** oder **enable_laser** freigegeben und mit **set_laser_control** oder **disable_laser** unterdrückt werden(1). Mit dem Befehl **get_startstop_info** (Bit #9) kann der aktuelle Status der Lasersteuersignale (freigegeben oder unterdrückt) für den “Laser active”-Betrieb ausgelesen werden.

Auch wenn die Lasersteuersignale für den “Laser active”-Betrieb wie oben beschrieben freigegeben und aktiviert wurden, werden sie nur dann an den Ausgängen ausgegeben, wenn sie durch weitere Befehle *eingeschaltet* werden. Ein automatisches Einschalten erfolgt durch Aufruf von Mark- oder Arc-Befehlen (siehe Seite 103). Die Signale können aber auch bei einer fixen Scanner-Position mit dem Befehl **laser_on_list** für eine vorgegebene Zeitdauer(2) oder mit **laser_signal_on** oder **laser_signal_on_list** zeitlich unbegrenzt eingeschaltet und mit **laser_signal_off** oder **laser_signal_off_list** wieder ausgeschaltet werden.

Die Lasersteuersignale für den “Laser active”-Betrieb werden automatisch von der RTC5 ausgeschaltet, wenn auf einen Markierbefehl (Mark oder Arc) ein Nicht-Markierbefehl folgt, wenn eine Liste (mit **set_end_of_list** oder **stop_execution**) beendet wird oder eine Liste (mit **set_wait**, **pause_list** oder **stop_list**) vorübergehend angehalten wird. Im letzteren Fall werden die Signale wieder eingeschaltet, wenn die Liste wieder (mit **release_wait** oder **restart_list**) fortgesetzt wird. Die Lasersteuersignale werden auch durch einen **list_nop**-Befehl ausgeschaltet. Dieser wird unter Umständen von der RTC5 automatisch eingefügt, z. B. als Ersatz für Listenbefehle mit unzulässiger Parameterangabe.

“Laser standby”-Signale

Nach dem erstmaligen Aktivieren der Lasersteuersignale mit **set_laser_control** können Standby-Signale mit **set_standby** oder **set_standby_list** aktiviert bzw. deaktiviert werden.

Standby-Signale werden nach Aktivierung auch ohne weitere Befehle genau dann permanent an den Ausgängen ausgegeben, wenn keine Signale für den “Laser active”-Betrieb eingeschaltet sind. Befehle, die Signale für den “Laser active”-Betrieb ein- oder ausschalten (siehe vorheriger Abschnitt), sorgen gegebenenfalls für ein entsprechendes Umschalten zwischen den Standby- und “Laser active”-Signalen. Sind die “Laser active”-Signale deaktiviert (z. B. durch `PulseLength` = 0) , so erfolgt keine Umschaltung; die Standby-Signale werden dann auch während der Ausführung dieser Befehle permanent ausgegeben.

Sind die Standby-Signale deaktiviert, so werden für den “Laser standby”-Betrieb alle Ausgänge in den TTL-Grundzustand gesetzt (z. B. auf LOW wenn mit **set_laser_control** ein hoher Signal-Pegel ausgewählt wurde).

Wenn aktivierte Standby-Signale beim Anhalten einer Liste mit **pause_list** deaktiviert werden sollen (automatisch werden hier von der RTC5 nur die Lasersteuersignale für den “Laser active”-Betrieb ausgeschaltet), so muss dies durch einen Aufruf des Befehls **set_standby**(`PulseLength` = 0) explizit veranlasst werden. Wenn beim Aufruf von **pause_list** (Kontrollbefehl) nicht bekannt ist, wie die Standby-Signale zu diesem Zeitpunkt definiert sind (die Standby-Parameter könnten innerhalb einer Liste verändert worden sein), können die aktuellen Standby-Parameter mit dem Kontrollbefehl **get_standby** ausgelesen werden.

(1) Bei unterdrückten Signalen sind die Lasersignal-Ausgänge im (hochohmigen) tristate-Zustand (die Signale werden auch automatisch mit **pause_list** unterdrückt und durch **restart_list** wieder freigegeben).

(2) Mit **laser_on_pulses_list** kann das LASERON-Signal auch für eine gewünschte Anzahl von externen Signalpulsen eingeschaltet werden.

Automatische Lasersignalunterdrückung bei Status-Fehlern des Scan-Systems

Mit **set_laser_control** (Bits#16 – 27) kann festgelegt werden, dass die Lasersteuersignale automatisch unterdrückt werden, wenn eines oder mehrere der Statussignale des Scan-Systems einen Fehler meldet. Dabei kann auch ausgewählt werden, welche Statussignale für die automatische Lasersignalunterdrückung verwendet werden sollen: PowerOK, TempOK und/oder PosAck von Kopf A und/oder B bzw. Achse X und/oder Achse Y.

Ist im Fehlerfall mindestens eines der ausgewählten Statussignale gleich 0,

- so wird die Ausgabe der (mit **set_laser_control** oder **enable_laser** freigegebenen) Lasersteuersignale automatisch unterbrochen und nur dann wieder aufgenommen, wenn alle ausgewählten Statussignale gleichzeitig 1 sind (mit **set_laser_control**, **disable_laser** oder **pause_list** unterdrückte Lasersteuerungssignale bleiben unabhängig vom aktuellen Wert der Statussignale unterdrückt),
- so werden interne Fehlerbits (kumulativ) gesetzt, die (ab Version DLL 527, OUT 529) mit **get_marking_info** ausgelesen werden können.
- so wird – sofern mit **set_laser_control**(Bit#28 = 1) so festgelegt – automatisch ein /STOP-Signal erzeugt (Liste hält an, Lasersteuersignale gehen dauerhaft aus).

Automatische Lasersignalunterdrückung bei Bereichsüberschreitung der Galvanometerscanner

Mit **range_checking** kann festgelegt werden, dass die Lasersteuersignale automatisch unterdrückt werden, wenn ein Galvanometerscanner eine vorgegebene Bereichsgrenze überschreitet.

7.4.2 Konfigurieren der Laseranschluss-Buchse (LASER)

Die Lasersignal-Typen, die an den Pins 1, 2 und 9 der Laseranschluss-Buchse (LASER) ausgegeben werden, können mit dem Kontrollbefehl **config_laser_signals** (ab Version DLL 535, OUT 535, RBF 524) und dem Listenbefehl **config_laser_signals_list** (ab Version DLL 537, OUT 537) konfiguriert werden.

Bei Verwendung unterschiedlicher Laser bzw. Laser-Betriebsarten machen diese Befehle u.U. das Konfigurieren per Hardware (unterschiedliche Kabel bzw. Schalter) hinfällig.

Während in der Standard-Einstellung (für normale Vektormarkierungen) 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 beispielsweise auch das FirstPulseKiller-Signal 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 dies nicht möglich, siehe **config_laser_signals**/**config_laser_signals_list**-Befehlsbeschreibung).

Die nachfolgende Beschreibung der verschiedenen Laser-Modi bezieht sich auf die Standard-Einstellung der Lasersignal-Ausgabe.

7.4.3 CO_2-Version

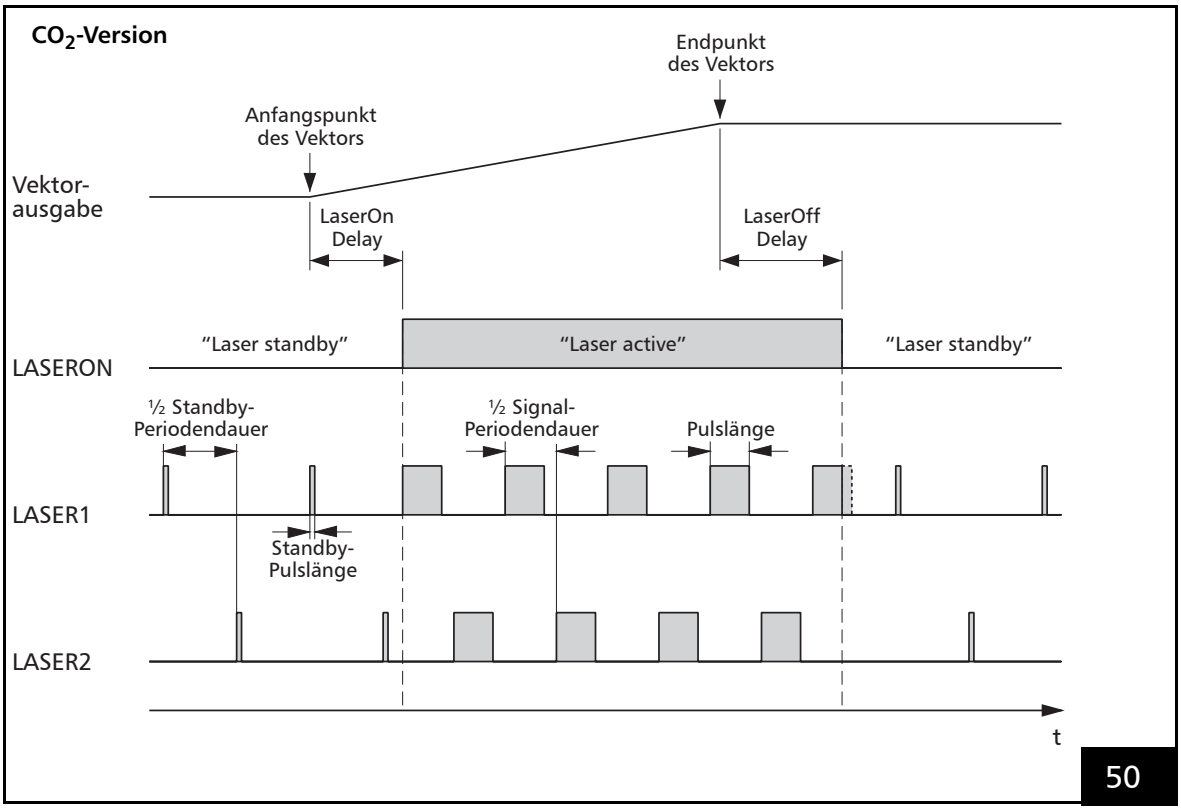
Mit dem Befehl `set_laser_mode(0)` wählt man die CO_2-Laserversion (Lasermode 0) aus. Im Zeitdiagramm Abb. 50 sind die zugehörigen Signale dargestellt.

Für den "Laser active"-Betrieb werden (gegebenenfalls nach einem LaserOn-Delay und sofern mit **set_laser_control** oder **enable_laser** freigegeben)

- das LASERON-Signal eingeschaltet und
- am LASER1- und LASER2-Ausgang zwei alternierende Modulationssignale ausgegeben, deren Pulslänge und Periodendauer mit den Befehlen **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** eingestellt werden können.

Für den "Laser standby"-Betrieb werden (gegebenenfalls nach einem LaserOff-Delay)

- das LASERON-Signal ausgeschaltet und
- am LASER1- und LASER2-Ausgang alternierende Standby-Pulse ausgegeben, deren Pulslänge und Periodendauer mit den Befehlen **set_standby** oder **set_standby_list** eingestellt werden können.



Zeitdiagramm der Steuersignale in der CO_2-Version (mit high-aktiven Lasersteuersignalen)

Hinweise

- Die LASER1- und LASER2-Signale werden durch Angabe einer von Null verschiedenen Pulslänge aktiviert und durch Nullsetzen der Pulslänge (Standardeinstellung) deaktiviert.
- Lasersignal LASER2 ist gegenüber Lasersignal LASER1 um 180° (eine halbe Signal-Periodendauer) phasenverschoben und kann für die Steuerung einer zweiten Laserröhre verwendet werden. Mit **set_laser_control** (Bit#1=1) können die beiden Signale auch miteinander vertauscht werden. Zur Steuerung der Laserleistung kann die Pulslänge von LASER1 und LASER2 variiert werden. Pulslänge und Periodendauer der beiden Signale sind jeweils gleich groß.
- Für die Signale LASER1 und LASER2 muss die *halbe* Periodendauer, d.h. die Verschiebung zwischen den beiden Lasersignalen, angegeben werden.
- Die Befehle **set_laser_pulses** und **set_laser_timing** sind *Listenbefehle* und können in einer Liste eingesetzt werden, um z.B. die Laserleistung zu jeder beliebigen Zeit zwischen zwei Listenbefehlen zu verändern.
- Die tatsächlich ausgegebene Periodendauer und die Pulslänge der Lasersignale LASER1 und LASER2 (nicht dagegen die Laser-Delays) hängen von der Taktfrequenz ab.
  Im RTC5-Modus ist diese fest auf 64 MHz gesetzt (1 Bit entspricht 1/64 µs).
  Im RTC4-Kompatibilitätsmodus kann sie für den "Laser active"-Betrieb mit dem Befehl **set_laser_timing** (nicht dagegen mit **set_laser_pulses_ctrl** oder **set_laser_pulses**) wahlweise entweder auf 1 MHz (1 Bit entspricht 1 µs) oder auf 8 MHz (1 Bit entspricht 1/8 µs) eingestellt werden. Im RTC4-Kompatibilitätsmodus ist es im allgemeinen empfehlenswert, eine Taktfrequenz von 8 MHz auszuwählen: `set_laser_timing(..,..,..,1)`. Eine Taktfrequenz von 1 MHz sollte nur wenn unbedingt nötig eingestellt werden.
  Die Zeitbasis für die Standby-Signale ist immer 1/64 µs (im RTC5-Modus) bzw. 1/8 µs (im RTC4-Kompatibilitätsmodus).
- Ob ein bei eingeschaltetem LASERON-Signal (für den "Laser active"-Betrieb) gestarteter Modulationspuls (LASER1 oder LASER2) noch vollständig ausgeführt oder abgeschnitten wird, wenn er beim Ausschalten des LASERON-Signals noch nicht voll abgearbeitet ist (siehe Abb. 50) kann mit **set_laser_control** (Bit#0) eingestellt werden. Für Standby-Signale kann dies nicht eingestellt werden (sie werden gegebenenfalls abgeschnitten).

7.4.4 YAG-Versionen

Mit **set_laser_mode**([1, 2, 3 oder 5]) wählt man zwischen vier verschiedenen YAG-Lasersteuerungs-Versionen (Lasermodi). Im Zeitdiagramm Abb. 51 sind die zugehörigen Signale dargestellt.

In allen vier YAG-Versionen werden für den "Laser active"-Betrieb (gegebenenfalls nach einem LaserOn-Delay und sofern mit **set_laser_control** oder **enable_laser** freigegeben)

- das LASERON-Signal eingeschaltet,
- am LASER1-Ausgang ein **Q-Switch-Signal** mit variabler Pulslänge und Ausgabeperiode ausgegeben und
- am LASER2-Ausgang ein programmierbares **FirstPulseKiller-Signal** ausgegeben.

Für den Standby-Betrieb werden (gegebenenfalls nach einem LaserOff-Delay)

- das LASERON-Signal ausgeschaltet und
- am LASER1-Ausgang (dagegen nicht am LASER2-Ausgang) Standby-Pulse ausgegeben, deren Pulslänge und Periodendauer mit den Befehlen **set_standby** oder **set_standby_list** eingestellt werden können.

Das LASER1-Signal wird durch Angabe einer von Null verschiedenen Pulslänge aktiviert und durch Nullsetzen der Pulslänge (Standardeinstellung) deaktiviert.

Q-Switch-Signal

Das Q-Switch-Signal dient zur Steuerung des Güteschalters des Lasers. Die Q-Switch-Periodendauer und -Pulsdauer werden mit den Befehlen **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** eingestellt.

Hinweise

- Beachten Sie, dass die *halbe* Q-Switch-Periodendauer angegeben werden muss.
- Die tatsächlich ausgegebene Q-Switch-Periodendauer und die Pulslänge (nicht dagegen die Laser-Delays) hängen von der Taktfrequenz ab.
  Im RTC5-Modus ist diese fest auf 64 MHz gesetzt (1 Bit entspricht 1/64 µs).
  Im RTC4-Kompatibilitätsmodus kann sie mit dem Befehl **set_laser_timing** (nicht mit **set_laser_pulses** oder **set_laser_pulses_ctrl**) wahlweise entweder auf 1 MHz (1 Bit entspricht 1 µs) oder auf 8 MHz (1 Bit entspricht 1/8 µs) eingestellt werden. Im RTC4-Kompatibilitätsmodus ist es im allgemeinen empfehlenswert, eine Taktfrequenz von 8 MHz auszuwählen: `set_laser_timing(..,..,..,1)`. Eine Taktfrequenz von 1 MHz sollte nur wenn unbedingt nötig eingestellt werden.
- Ob ein Q-Switch-Puls (LASER1) noch vollständig ausgeführt oder abgeschnitten wird, wenn er beim Ausschalten des LASERON-Signals noch nicht voll abgearbeitet ist (siehe Abb. 51) kann mit **set_laser_control** (Bit#0) eingestellt werden. Für Standby-Signale kann dies nicht eingestellt werden (sie werden gegebenenfalls abgeschnitten).

FirstPulseKiller-Signal

Das FirstPulseKiller-Signal dient zur Begrenzung der Laserpuls-Leistung zu Beginn einer Pulsfolge. Die ersten Laserpulse einer Pulsfolge können eine höhere Energie haben als die folgenden. Da unterschiedliche Laserleistungen während eines Scan-Vorgangs in der Regel unerwünscht sind, sollte der Laser über eine Schaltung zur Begrenzung der Leistung dieser ersten Pulse verfügen. Zur Steuerung dieser Schaltung wird von der RTC5 das FirstPulseKiller-Signal bereitgestellt (TTL-Pegel).

Das FirstPulseKiller-Signal wird automatisch mit dem LASERON-Signal aktiviert. Während das FirstPulse-Killer-Signal anliegt, sollte die Pulsenergie in geeigneter Weise reduziert werden.

Die Länge des FirstPulseKiller-Signals wird mit den Befehlen **set_firstpulse_killer** oder **set_firstpulse_killer_list** eingestellt.

Unterschiede zwischen den YAG-Versionen

Die vier YAG-Versionen unterscheiden sich nur im relativen Start-Zeitpunkt des ersten Q-Switch-Pulses bzgl. des FirstPulseKiller-Signals (siehe auch die Zeitdiagramme in Abb. 51).

- In der **YAG-Version 1** startet der erste Q-Switch-Puls am *Anfang* des FirstPulseKiller-Signals.
- In der **YAG-Version 2** startet der erste Q-Switch-Puls am *Ende* des FirstPulseKiller-Signals.
- In der **YAG-Version 3** startet der erste Q-Switch-Puls 10 µs nach dem Anfang des FirstPulseKiller-Signals.
- In der **YAG-Version 5** ist der Zeitabstand mit **set_qswitch_delay** oder **set_qswitch_delay_list** variabel einstellbar. Die YAG-Versionen 1-3 sind Spezialfälle der YAG-Version 5 mit
  - Q-Switch-Delay = 0 (YAG-Version 1)
  - Q-Switch-Delay = Länge des FirstPulseKiller-Signals (YAG-Version 2)
  - Q-Switch-Delay = 10 µs (YAG-Version 3)

set_laser_mode bewirkt allerdings lediglich eine *Voreinstellung* des Q-Switch-Delays. Auch in den YAG-Versionen 1-3 kann daher das Q-Switch-Delay auch noch nachträglich über **set_qswitch_delay** oder **set_qswitch_delay_list** variabel eingestellt werden. In der YAG-Version 2 erfolgt auch mit jedem **set_firstpulse_killer**- bzw. **set_firstpulse_killer_list**-Befehl eine entsprechende Anpassung des Q-Switch-Delays.

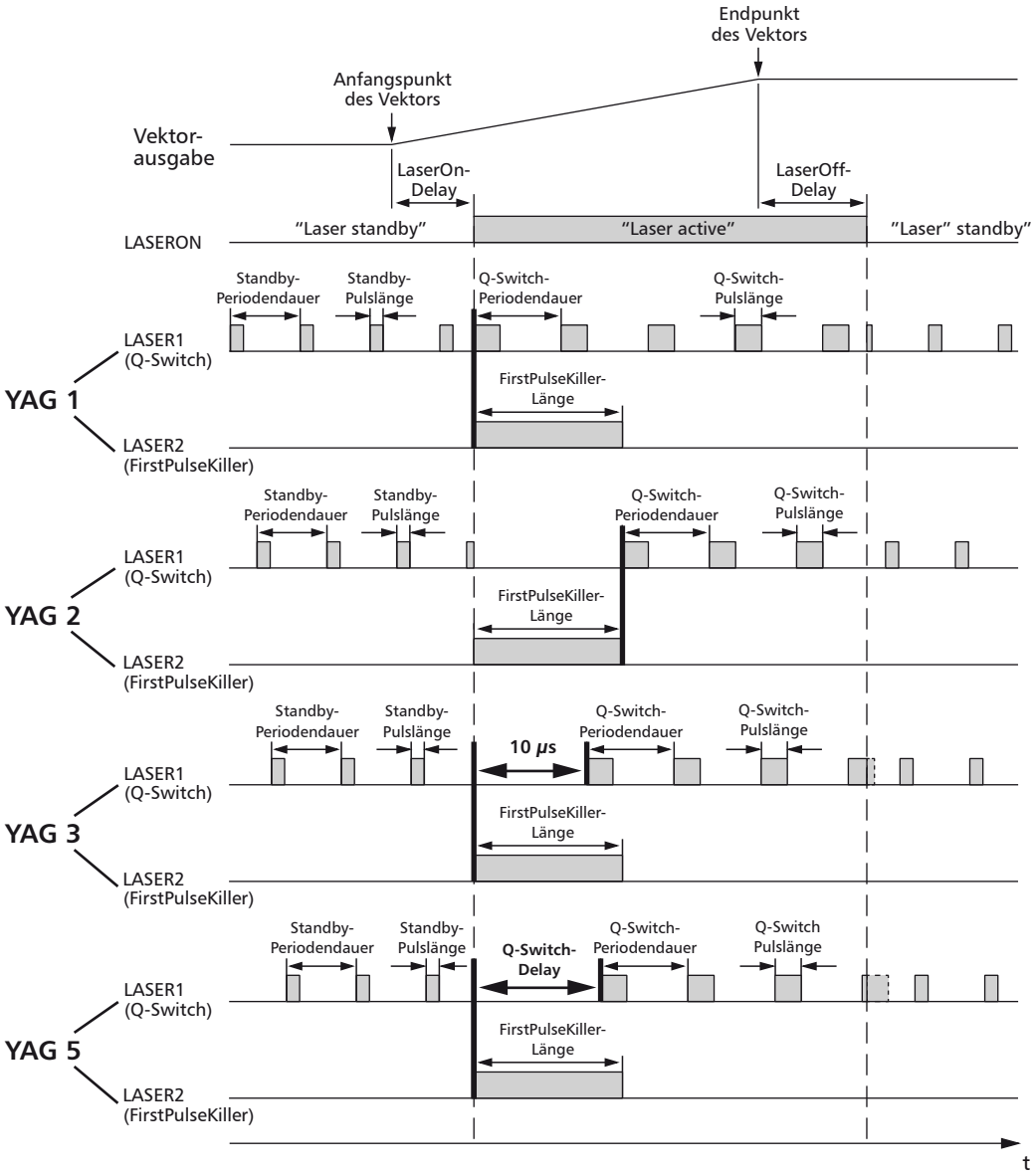
Lampenstrom (Laserleistung)

Zur Steuerung des Lampenstroms eines YAG-Lasers können die 12-Bit-Ausgänge ANALOG OUT1 oder ANALOG OUT2 eingesetzt werden. Sie sind an der 15-poligen Sub-D-LASER-Buchse verfügbar (siehe Abb. 10). Zur Festlegung des analogen Ausgangssignals werden der Kontrollbefehl **write_da_x** und der kurze Listenbefehl **write_da_x_list** bereitgestellt.

Alternativ, kann der Lampenstrom digital über die 8-Bit-wertige digitale EXTENSION 2-Stiftleiste gesteuert werden (siehe Abb. 17). Die Befehle zur Einstellung des 8-Bit-Ausgangs lauten **write_8bit_port** und **write_8bit_port_list** (siehe auch Kapitel 4.4.3 "EXTENSION 2-Stiftleiste", Seite 52).

Wenn der Lampenstrom geändert wird, kann bei Bedarf zusätzlich die weitere Ausführung der Liste mit dem Befehl **long_delay** angehalten werden, bis sich eine konstante Laserleistung eingestellt hat.

YAG-Versionen 1-3 und 5



51

Zeitdiagramm der Steuersignale in den YAG-Versionen 1, 2, 3 und 5 (mit high-aktiven Lasersteuersignalen)
(die Standby-Signale sind nicht mit den "Laser active"-Signalen synchronisiert und können beliebige Phasenlagen dazu haben)

7.4.5 Softstart-Modus

Für manche Anwendungen ist es erforderlich, die Laserintensität zu Beginn eines Markierungsprozesses zu variieren. SCANLAB stellt dazu in allen Laserversionen (für den "Laser active"-Betrieb) den Softstart-Modus zur Verfügung.

Im Softstart-Modus können unterschiedliche Kontrollwerte (`Level`(0) … `Level`(`Number` – 1), `Number` ≤ 1024) für die ersten – bis zu 1024 – Pulse (zu Beginn eines Markierungsprozesses) definiert werden. Wahlweise können Pulslängenwerte für das Lasersignal am LASER1-Ausgang (siehe Abb. 10) oder Analogspannungswerte für variable Laserleistungen definiert werden. Analogspannungswerte werden wahlweise am ANALOG OUT1- oder ANALOG OUT2-Ausgang (siehe Abb. 10) ausgegeben.

Die Initialisierung des Softstart-Modus erfolgt mit **set_softstart_mode** oder **set_softstart_mode_list**. Dabei wird der Typ der Softstartwerte festgelegt und – falls Analogspannungswerte ausgegeben werden sollen – festgelegt, an welchem Analogausgang die Werte ausgegeben werden. Anschließend können die einzelnen Softstart-Kontrollwerte mit **set_softstart_level**- oder **set_softstart_level_list**-Befehlen definiert werden.

Die definierten Softstartwerte (max. 1024) werden nach Einschalten der Lasersteuersignale simultan mit den Laserpulsen des LASER1-Signals – jeweils mit der ersten Flanke eines jeden Laserpulses – am ausgewählten Ausgang ausgegeben. Mit **set_softstart_mode**/**set_softstart_mode_list** kann dabei eine Zeitspanne (`Delay`) vorgegeben werden, für die der Laser ausgeschaltet gewesen sein muss, ehe der Softstart beim nächsten Einschalten aktiviert wird.

Hinweise

- Mit einem ausreichend hohen Wert für das `Delay` verhindert man, dass der Softstart-Modus auch nach sehr kleinen Jump-Befehlen aktiviert wird.
- Sobald das LASERON-Signal (der Laser) länger als `Delay` ausgeschaltet bleibt, wird der Wert `Level`(0) ausgegeben, sofern der Softstart-Modus nicht zwischenzeitlich deaktiviert wurde.
- Wird das LASERON-Signal eingeschaltet, so werden simultan mit den ersten Laserpulsen automatisch die Werte `Level`(0) … `Level`(`Number` – 1) ausgegeben.
- Wenn Softstartwerte als Analogspannungswerte ausgegeben werden sollen, ist es u.U. empfehlenswert die Softstartwerte, die simultan mit den Laserpulsen des LASER1-Signals ausgegeben werden, stattdessen simultan mit den Pulsen eines um 180° phasenverschobenen Lasersignals abzugreifen. In der CO_2-Version steht dafür das LASER2-Signal am LASER2-Ausgang zur Verfügung. In den YAG-Versionen kann dazu – mit **set_laser_control** (Bit#1 = 1) – das Signal am LASER1-Ausgang um 180° phasenverschoben werden. In der CO_2-Version entspricht dies einer Vertauschung von LASER1 mit LASER2.
- Die Laserfrequenz sollte ca. 308 kHz (entspricht **set_laser_pulses**-`HalfPeriod` = 104) nicht überschreiten, weil die sich ändernden Werte nicht schneller ausgegeben werden können. Die Laserfrequenz kann jedoch für diesen Fall nicht automatisch kontrolliert werden.
- Wenn Softstartwerte als Analogspannungswerte ausgegeben werden sollen, sollte zusätzlich beachtet werden, dass die DA-Wandlung für Laserfrequenzen über ca. 100 kHz (d.h. für **set_laser_pulses**-`HalfPeriod` < ca. 320) nicht in jedem Fall vollständig abgeschlossen wird. Bei solchen Laserfrequenzen muss die ausreichende Funktionsfähigkeit von Benutzern sorgfältig überprüft werden.
- Bei inaktivem Softstart-Modus können die Werte an den Analogausgängen mit **write_da_x** oder **write_da_x_list** gesetzt werden.

- Der Softstart-Modus steht auch im Lasermode 4 (siehe Seite 154) und Lasermode 6 (siehe Seite 155) zur Verfügung. Im Lasermode 4 und Lasermode 6 werden am LASER1-Ausgang nur Standby-Pulse ausgegeben, und daher können hier im Softstart-Modus nur Analogspannungswerte variabel eingestellt werden (**set_softstart_mode** `Mode` = 1, 2, 11 oder 12). Die definierten Softstartwerte werden auch hier simultan mit den Pulsen eines intern generierten (aber nicht ausgegebenen) LASER1-Signals ausgegeben. Die Standby-Pulse können im Lasermode 4 und Lasermode 6 nicht phasenverschoben werden und sind nicht mit den internen LASER1-Pulsen synchronisiert. Wenn die Periodendauern der Standby- und LASER1-Pulse (mit **set_laser_pulses** bzw. **set_standby**) gleich eingestellt werden, werden die Analogspannungswerte parallel mit den Standby-Pulsen ausgegeben, können aber eine (zufällige) konstante Phasenverschiebung zu den Standby-Pulsen haben.
- Der Softstart-Modus kann nicht zeitgleich mit dem Wobbelmode “Frei definierbare Wobbelfigur“ verwendet werden, siehe **set_wobbel_vector**.

7.4.6 Lasermode 4

Mit `set_laser_mode(4)` wählt man den Lasermode 4 aus. Im Zeitdiagramm Abb. 52 sind die zugehörigen Signale dargestellt.

Am LASER1-Ausgang wird ein freilaufendes Modulationssignal mit variabler Pulslänge und Periodendauer sowohl für den "Laser active"-Betrieb als auch den "Laser standby"-Betrieb ausgegeben, sofern es mit **set_standby** oder **set_standby_list** aktiviert wurde.

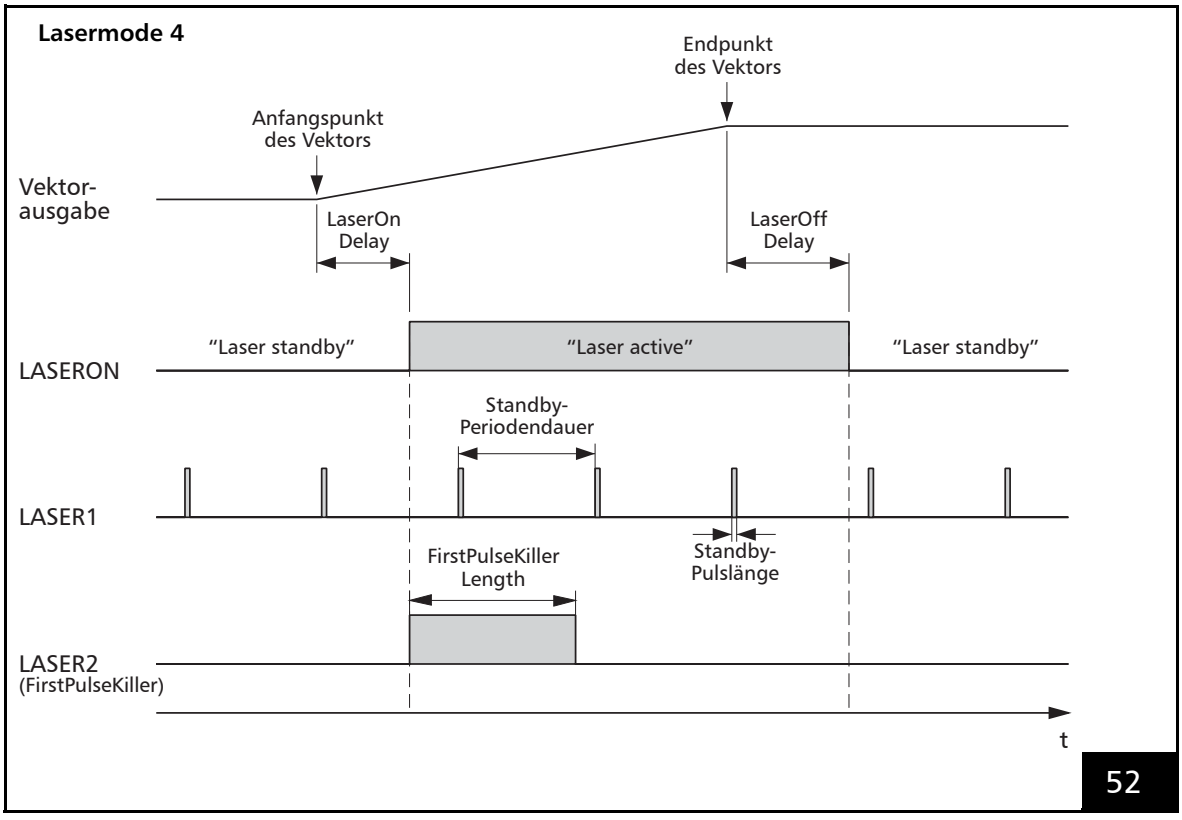
Für den "Laser active"-Betrieb werden zusätzlich (gegebenenfalls nach einem LaserOn-Delay und sofern mit **set_laser_control** oder **enable_laser** freigegeben)

- das LASERON-Signal eingeschaltet und
- am LASER2-Ausgang ein programmierbares FirstPulseKiller-Signal ausgegeben.

Für den Standby-Betrieb werden zusätzlich (gegebenenfalls nach einem LaserOff-Delay) das LASERON-Signal und gegebenenfalls das LASER2-Signal ausgeschaltet.

Hinweise

- Die Pulslänge und Ausgabeperiode des LASER1 Modulationssignals werden mit den Befehlen **set_standby** oder **set_standby_list** eingestellt. Das LASER1-Signal wird durch Angabe einer von Null verschiedenen Pulslänge aktiviert und durch Nullsetzen der Pulslänge (Standardeinstellung) deaktiviert.
- Beachten Sie, dass beim Einstellen der Periodendauer die *halbe* Periodendauer angegeben werden muss.
- Das FirstPulseKiller-Signal wird automatisch mit dem LASERON-Signal aktiviert. Die Länge des FirstPulseKiller-Signals wird mit den Befehlen **set_firstpulse_killer** oder **set_firstpulse_killer_list** eingestellt.
- Im Lasermode 4 ist die Zeitbasis für die Signale LASER1 und LASER2 immer 1/64 µs (im RTC5-Modus) bzw. 1/8 µs (im RTC4-Kompatibilitätsmodus).
- Der Lasermode 4 wird für manche Faserlaser verwendet.



Zeitdiagramm der Steuersignale im Lasermode 4 (mit high-aktiven Lasersteuersignalen)

7.4.7 Lasermode 6

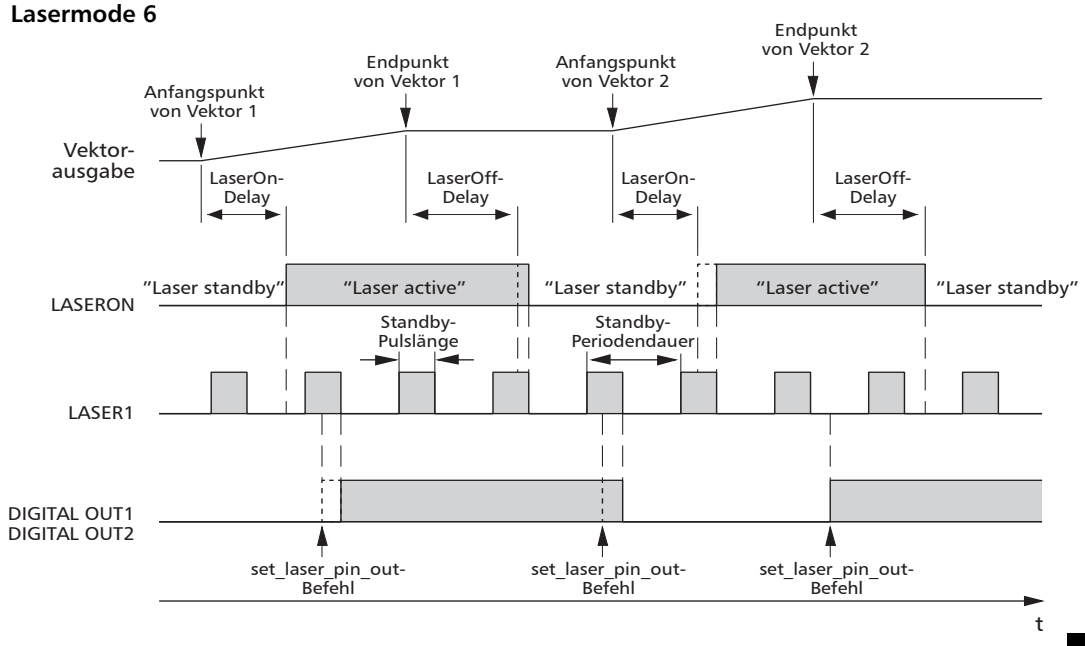
Mit `set_laser_mode(6)` wählt man den Lasermode 6 aus. Diese Lasersteuerungsversion wird für Laser bereitgestellt, bei denen die Laserleistung nicht während eines Pulses geändert werden kann. Im Zeitdiagramm Abb. 53 sind die zugehörigen Signale dargestellt.

Am LASER1-Ausgang wird ein freilaufendes Modulationssignal mit variabler Pulslänge und Periodendauer sowohl für den "Laser active"-Betrieb als auch den "Laser standby"-Betrieb ausgegeben, sofern es mit **set_standby** oder **set_standby_list** aktiviert wurde.

Für den "Laser active"-Betrieb wird zusätzlich das LASERON-Signal (gegebenenfalls nach einem LaserOn-Delay und sofern mit **set_laser_control** oder **enable_laser** freigegeben) synchronisiert mit dem LASER1-Signal (s.u.) eingeschaltet.

Für den Standby-Betrieb wird das LASERON-Signal (gegebenenfalls nach einem LaserOff-Delay) synchronisiert mit dem LASER1-Signal (s.u.) ausgeschaltet.

Lasermode 6



53

Zeitdiagramm der Steuersignale im Lasermode 6 (mit high-aktiven Lasersteuersignalen)

Hinweise

- Ein LASER2-Signal wird weder für den “Laser active”-Betrieb noch für den “Laser standby”-Betrieb ausgegeben.
- Änderungen des LASERON-Signals werden unterbunden während das LASER1-Signal “an“ ist, indem die Änderung erst dann (verzögert) ausgeführt wird, wenn das LASER1-Signal wieder “aus“ ist (siehe Abb. 53). D.h. falls das LASER1-Signal beispielsweise auf high-aktiv eingestellt wurde: Eine Änderung des LASERON-Signals wird nicht ausgeführt während das LASER1-Signal high ist, sondern erst dann (verzögert), wenn es wieder low ist.

  Genauso werden im Lasermode 6 auch Änderungen des Ausgangswertes am 2-Bit-Digital-Ausgang (DIGITAL OUT1 und DIGITAL OUT2) durch **set_laser_pin_out** oder **set_laser_pin_out_list** verzögert ausgeführt, falls das LASER1-Signal sonst zum tatsächlichen Zeitpunkt der Änderung “an” wäre. Dadurch können externe Geräte synchron zu den Laserpulsen gesteuert werden.
- Die Pulslänge und Ausgabeperiode des LASER1 Modulationssignals wird mit den Befehlen **set_standby** oder **set_standby_list** eingestellt. Das LASER1-Signal wird durch Angabe einer von Null verschiedenen Pulslänge aktiviert und durch Nullsetzen der Pulslänge (Standardeinstellung) deaktiviert.
- Beachten Sie, dass beim Einstellen der Periodendauer die *halbe* Periodendauer angegeben werden muss.
- Im Lasermode 6 ist die Zeitbasis für das LASER1-Signal immer 1/64 µs (im RTC5-Modus) bzw. 1/8 µs (im RTC4-Kompatibilitätsmodus).
- Zur Markierung von Punkten kann das LASERON-Signal mit **laser_on_pulses_list** für eine vorgebbare Anzahl von externen Pulsen (bzw. für ein vorgebbares Zeitintervall) eingeschaltet werden. Die externen Pulse müssen dabei am Digital-Eingang DIGITAL IN1 der LASER-Buchse (siehe Seite 48) eingespeist werden. Mit **para_laser_on_pulses_list** wird während der Punktmarkierung ein ausgewählter Signalparameter linear auf den angegebenen Wert variiert.

7.4.8 Pulse-Picking-Lasermodus

Mit **set_pulse_picking** bzw. **set_pulse_picking_list** wählt man den Pulse-Picking-Lasermodus aus. Im Zeitdiagramm Abb. 54 sind die zugehörigen Signale dargestellt.

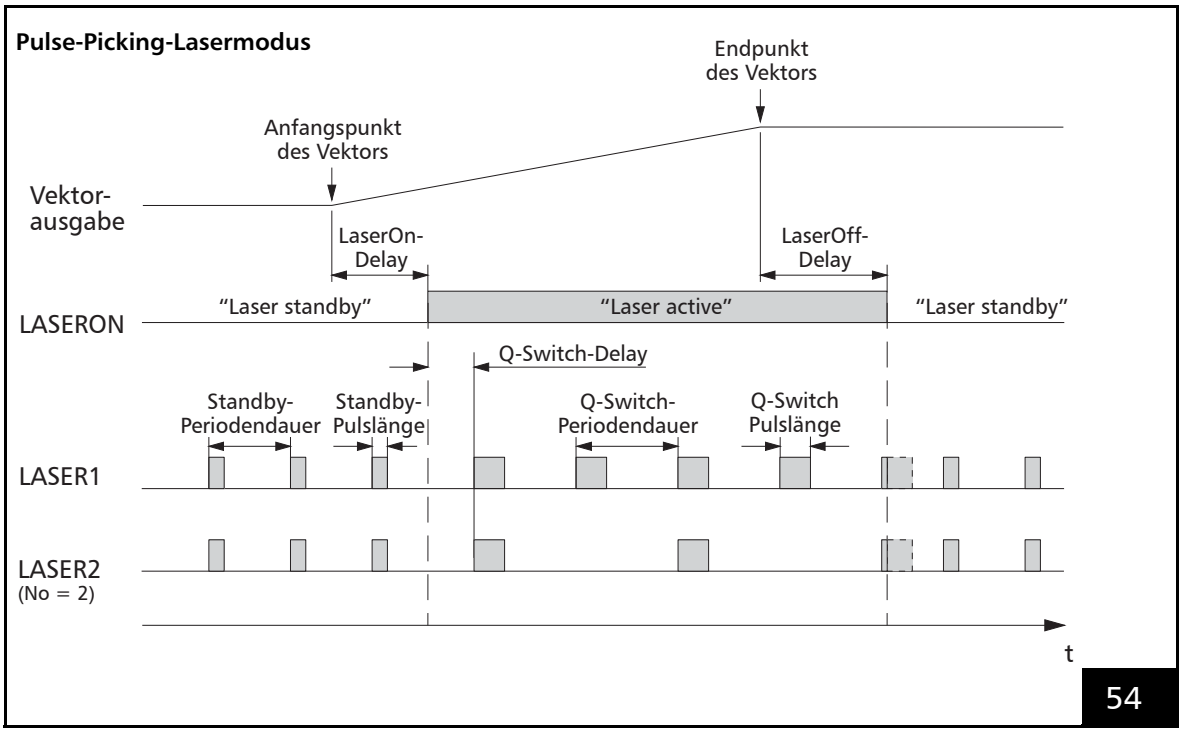
Für den "Laser active"-Betrieb werden (gegebenenfalls nach einem LaserOn-Delay und sofern mit **set_laser_control** oder **enable_laser** freigegeben)

- das LASERON-Signal eingeschaltet,
- am LASER1-Ausgang ein Modulationssignal ausgegeben, dessen Pulslänge und Periodendauer mit den Befehlen **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** eingestellt werden kann.
- am LASER2-Ausgang die gleichen Signale wie am LASER1-Ausgang (phasengleich) ausgegeben, allerdings gemäß Parameter No von **set_pulse_picking** bzw. **set_pulse_picking_list** nur jeder No-te Puls.

Für den Standby-Betrieb werden (gegebenenfalls nach einem LaserOff-Delay)

- das LASERON-Signal ausgeschaltet und
- am LASER1- und LASER2-Ausgang phasengleich Standby-Pulse ausgegeben, deren Pulslänge und Periodendauer mit den Befehlen **set_standby** oder **set_standby_list** eingestellt werden können.

Mit **set_laser_control**(Bit #7 = 1) kann der sogenannte "konstante Pulslänge"-Modus (als Spezialfall des Pulse-Picking-Lasermodus) aktiviert werden, bei dem das "Laser active"-LASER2-Signal mit einer vom LASER1-Signal unabhängigen festen Pulslänge ausgegeben wird (die gewünschte LASER2-Pulslänge wird mit dem Befehl **set_pulse_picking_length** definiert). Dadurch kann das LASER2-Signal auch für kurze LASER1-Pulse als Gate-Signal verwendet werden.



Zeitdiagramm der Steuersignale im Pulse-Picking-Lasermodus (mit high-aktiven Lasersteuersignalen und No = 2)

Hinweise

- Wie in der YAG-Version 5 ist der Zeitabstand zwischen LaserOn und dem ersten Q-Switch-Puls (LASER1/LASER2) mit **set_qswitch_delay** oder **set_qswitch_delay_list** variabel einstellbar. Das LASER2-Signal startet synchron mit dem LASER1-Signal.
- Die LASER1/LASER2-Signale werden durch Angabe einer von Null verschiedenen Pulslänge aktiviert und durch Nullsetzen der Pulslänge (Standardeinstellung) deaktiviert.
- Für die Signale LASER1 und LASER2 muss die *halbe* Periodendauer angegeben werden.
- Die Befehle **set_laser_pulses** und **set_laser_timing** sind *Listenbefehle* und können in einer Liste eingesetzt werden, um z. B. die Laserleistung zu jeder beliebigen Zeit zwischen zwei Listenbefehlen zu verändern.
- Die tatsächlich ausgegebene Periodendauer und die Pulslänge der Lasersignale LASER1 und LASER2 (nicht dagegen die Laser-Delays) hängen von der Taktfrequenz ab.
  Im RTC5-Modus ist diese fest auf 64 MHz gesetzt (1 Bit entspricht 1/64 µs).
  Im RTC4-Kompatibilitätsmodus kann sie für den "Laser active"-Betrieb mit dem Befehl **set_laser_timing** (nicht dagegen mit **set_laser_pulses_ctrl** oder **set_laser_pulses**) wahlweise entweder auf 1 MHz (1 Bit entspricht 1 µs) oder auf 8 MHz (1 Bit entspricht 1/8 µs) eingestellt werden. Im RTC4-Kompatibilitätsmodus ist es im allgemeinen empfehlenswert, eine Taktfrequenz von 8 MHz auszuwählen: `set_laser_timing(..,..,..,1)`. Eine Taktfrequenz von 1 MHz sollte nur wenn unbedingt nötig eingestellt werden.
  Die Zeitbasis für die Standby-Signale ist immer 1/64 µs (im RTC5-Modus) bzw. 1/8 µs (im RTC4-Kompatibilitätsmodus).
- Ob ein bei eingeschaltetem LASERON-Signal (für den "Laser active"-Betrieb) gestarteter Modulationspuls (LASER1 oder LASER2) noch vollständig ausgeführt oder abgeschnitten wird, wenn er beim Ausschalten des LASERON-Signals noch nicht voll abgearbeitet ist (siehe Abb. 54) kann mit **set_laser_control** (Bit#0) eingestellt werden. Für Standby-Signale kann dies nicht eingestellt werden (sie werden gegebenenfalls abgeschnitten).
- Die Einstellungsreihenfolge der LASER1-Signale (mit **set_laser_pulses** etc.) bzw. des Pulse-Picking-Lasermodus ist unerheblich.
- **set_pulse_picking** bzw. **set_pulse_picking_list** überschreibt jede zuvor mit **set_laser_mode** vorgenommene Laser-Modus-Einstellung. Umgekehrt schaltet eine Laser-Modus-Einstellung (mit **set_laser_mode**) den Pulse-Picking-Lasermodus aus.

7.4.9 Automatische Lasersteuerung

Mit den Befehlen zur automatischen Lasersteuerung kann eine automatische Nachregelung von "Laser active"-Lasersteuersignalen – und damit der Laserleistung – auch während der Ausführung von Vektor- und Bogenbefehlen realisiert werden.

Dies ermöglicht es zum einen, Variationen im Lasereintrag (und daher im Bearbeitungsergebnis) zu kompensieren, die bei konstant voreingestellter Laserleistung beispielsweise durch eine sich verändernde Leistungsdichte oder Spot-Größe oder durch eine sich verändernde Bearbeitungsgeschwindigkeit hervorgerufen wird. Bei Beschriftungen lassen sich damit beispielsweise einheitlichere Beschriftungsstärken und bei Schraffuren einheitlichere Linienbreiten erzielen.

Andererseits kann der Leistungseintrag entlang eines Vektors auch ganz bewusst nachgeregelt werden, z. B. um bei Laserschweiß-Anwendungen die gewünschte Schweißnahtform zu erzielen.

Die Befehle zur automatischen Lasersteuerung ergänzen die in den vorherigen Abschnitten dieses Kapitels beschriebenen Befehle wie **set_laser_pulses**, **write_da_x_list** oder **write_8bit_port_list**, mit denen lediglich eine *konstante* Laserleistung – für die jeweils nachfolgenden Mark- und Arc-Befehle – voreingestellt werden kann.

Zur automatischen Lasersteuerung kann eine positions-, geschwindigkeits- oder vektorkontrollierte Korrektur der Lasersteuersignale aktiviert werden (diese drei Korrekturmöglichkeiten können auch miteinander kombiniert werden). Dabei werden die ausgewählten Lasersteuersignale entsprechend der Scan-Bewegung des Scan-Systems gesteuert. Alternativ zur (Galvanometerscanner-) geschwindigkeitsabhängigen Lasersteuerung kann auch eine encodergeschwindigkeitsabhängige Korrektur (kombinierbar mit der positions- und vektorkontrollierten Lasersteuerung) aktiviert werden.

- Mit der *positionsabhängigen* Lasersteuerung (siehe Seite 161) können positionsabhängige (radiale) Variationen des Lasereintrags während der Ausführung von Vektor- und Bogenbefehlen kompensiert werden. Solche Variationen werden z. B. durch Abbildungsfehler im optischen System (z. B. durch einen Randabfall des Objektivs, d.h. durch Leistungsverluste am Objektivrand) hervorgerufen. Sie können aber auch daraus resultieren, dass der Laserstrahl mit zunehmendem Abstand vom Bildfeldmittelpunkt – je nach optischem System – unter einem Winkel, also nicht mehr senkrecht auf das Material trifft und dies zu einer Veränderung der Reflexion und des Laserspots führt. Mit der positionsabhängigen Lasersteuerung kann eine automatische radiale Korrektur (abhängig vom Abstand zum Bildfeldmittelpunkt) eingestellt werden.
- Mit der *geschwindigkeitsabhängigen* Lasersteuerung (siehe Seite 165) können geschwindigkeitsabhängige Variationen des Lasereintrags während der Ausführung von Vektor- und Bogenbefehlen kompensiert werden. Diese resultieren z. B. daraus, dass sich die Geschwindigkeit innerhalb eines Markierbefehls oder von Markierbefehl zu Markierbefehl ändert, etwa
  - infolge einer Beschleunigungs- oder Abbremsbewegung der Scanner am Anfang oder Ende eines Vektors bzw. bei Richtungsänderungen innerhalb eines Polygonzugs
  - durch eine automatische Anpassung der Schrittweite und der effektiven Markiergeschwindigkeit an den 10 µs-Takt (Kurzvektor-Jitter)
  - bei zeitbemessenen Mark-Befehlen durch die automatische Anpassung der Markiergeschwindigkeit entsprechend der vorgegebenen Markierdauer (mit zeitbemessenen Mark-Befehlen kann z. B. der Flatterrand vermieden werden, der bei normalen Mark-Befehlen u. U. am Vektorende entstehen kann)
- Bei der *vektorkontrollierten* Lasersteuerung (siehe Seite 166) kann ein Lasersignalparameter entlang eines Mark- oder Jump-Vektors linear nachgeregelt werden. Dadurch kann z. B. die Variation des Lasereintrags bei unebenen Materialoberflächen (verursacht durch einen über den Vektor variierenden Auftreffwinkel) kompensiert werden.
- Mit der *encodergeschwindigkeitsabhängigen* Lasersteuerung (siehe Seite 168) kann der Lasereintrag entsprechend der aktuell vorliegenden Encodergeschwindigkeit (Zählerpulse des letzten 10 µs-Intervalls) eines Encoder-Zählers, also z. B. entsprechend der aktuellen Fördergeschwindigkeit eines Transportsystems (und damit des Werkstücks) gesteuert werden.

Durch die automatische Lasersteuerung wird wahlweise einer der folgenden Signalparameter angepasst:

- der 12-Bit-Ausgabewert am ANALOG OUT1- oder ANALOG OUT2- Ausgang der 15-poligen Sub-D-LASER-Buchse (siehe auch Seite 48)
- der Ausgabewert am 16-Bit-Digital-Ausgang der EXTENSION 1-Stiftleiste (siehe auch Seite 51)
- der Ausgabewert am 8-Bit-Digital-Ausgang der EXTENSION 2-Stiftleiste (siehe auch Seite 52)
- die Pulslänge (`PulseLength`) oder Ausgabeperiode (`HalfPeriod`) der Lasersignale LASER1 und LASER2
- Nur bei der vektorabhängigen Lasersteuerung: Fokusverschiebung für nachfolgende parametrisierte Vektorausgaben (die 3D-Option muss dazu freigeschaltet und eine 3D-Korrekturdatei geladen und zugewiesen sein)

Für die positionsabhängige und geschwindigkeitsabhängige bzw. encodergeschwindigkeitsabhängige Lasersteuerung kann nur eine gemeinsame Signalparameter-Auswahl getroffen werden. Diese Korrekturen wirken also immer auf den gleichen Signalparameter. Für die vektorabhängige Lasersteuerung kann dagegen ein anderer Signalparameter ausgewählt werden.

Der automatisch angepasste und letztendlich zur Lasersteuerung ausgegebene Wert kann mit **set_trigger**/**set_trigger4** (`Signal1/2`=24 und 31…38) aufgezeichnet werden. Dadurch kann z. B. die Auswirkung der variierenden Geschwindigkeit während der Beschleunigungs- oder Abbremsphase der Scannerbewegung kontrolliert werden.

Hinweise

- Ein variierender Lasereintrag infolge einer expliziten Änderung der Markiergeschwindigkeit zwischen zwei Markierbefehlen durch Aufruf des **set_mark_speed**-Befehls wird *nicht* durch die automatische Lasersteuerung ausgeglichen.
- Wenn durch die automatische Lasersteuerung die 12-Bit-Ausgabewerte am ANALOG OUT1- oder ANALOG OUT2- Ausgang, die Pulslängen (`PulseLength`) oder Ausgabeperioden (`HalfPeriod`) der Lasersignale LASER1 und LASER2 angepasst werden, dann sollte die automatische Lasersteuerung *nicht* gleichzeitig mit dem Pixelausgabe- oder dem Softstart-Modus verwendet werden, da sich diese Funktionen gegenseitig stören. Die automatische Anpassung der Ausgabewerte am 16-Bit-Digital-Ausgang oder am 8-Bit-Digital-Ausgang kann dagegen uneingeschränkt parallel mit dem Pixelausgabe- oder dem Softstart-Modus verwendet werden.
- Die für die automatische Lasersteuerung benötigten Einstellparameter und Tabellen müssen für die jeweilige Anwendung von Benutzern selbst ermittelt werden. Diese Ermittlung kann mit einem beträchtlichen Aufwand verbunden sein, der insbesondere dann empfehlenswert ist, wenn eine besonders hohe Genauigkeit erforderlich ist.
- Die automatische Lasersteuerung kann nicht mit der variablen Lasersteuerung über die McBSP/SPI-Schnittstelle (siehe **set_multi_mcbsp_in**) kombiniert werden.

Positionsabhängige Lasersteuerung

Zur Initialisierung der positionsabhängigen Lasersteuerung muss mit **set_auto_laser_control** (Parameter `Ctrl`) festgelegt werden, welcher Signalparameter angepasst werden soll, und mit **load_position_control** eine Skalierungsfunktion geladen werden.

Bei nachfolgenden Mark- und Arc-Befehlen (auch bei zeitbemessenen Mark-Befehlen) führt die RTC5 dann eine positionsabhängige automatische Korrektur des ausgewählten Signalparameters durch (genauer: eine Korrektur abhängig vom radialen Abstand zwischen aktuellem Ausgabepunkt und Bildfeldmittelpunkt). Der ausgewählte Signalparameter wird mit dem positionsabhängigen (radialen) Korrekturfaktor multipliziert, der durch die geladene Skalierungsfunktion definiert ist. Die Skalierungsfaktoren beziehen sich auf einen 100%-Wert, der mit dem Befehl **set_auto_laser_control** (Parameter `Value`) festgelegt wird.

Da die positionsabhängige Änderung des Leistungseintrags systemabhängig ist, muss die zu ladende Skalierungsfunktion von Benutzern in einer Text-Datei definiert werden (siehe "Hinweise zum Laden der Skalierungsfunktion", Seite 162).

Bei der Berechnung der positionsabhängigen Korrektur werden die aktuellen kartesischen Ansteuerwerte `Sample<X..Y>` (siehe auch **set_trigger**/**set_trigger4**) zu Grunde gelegt. Diese berücksichtigen gegebenenfalls eine Wobbel- und eine "Processing on the fly"-Korrektur, aber weder eine Bildfeldkorrektur noch eine Koordinatentransformation oder eine "Gain"- und "Offset"-Korrektur.

Allgemeine Hinweise

- Mit **set_auto_laser_control** (Parameter `Ctrl`) kann nicht nur die positionsabhängige, sondern auch die geschwindigkeitsabhängige oder encodergeschwindigkeitsabhängige Lasersteuerung initialisiert werden (siehe Seite 165 und Seite 168). Die mit **set_auto_laser_control** festgelegte Signalparameter-Auswahl (Parameter `Ctrl`) und der 100%-Wert (Parameter `Value`) gelten dann gegebenenfalls für beide Korrekturen. Die Korrekturfaktoren der positions- und der geschwindigkeits- bzw. encodergeschwindigkeitsabhängigen Lasersteuerung werden dann multiplikativ zu einem gemeinsamen Faktor zusammengefasst.

  Falls nur eine positionsabhängige Lasersteuerung benötigt wird, kann die geschwindigkeits- und encodergeschwindigkeitsabhängige Lasersteuerung mit **set_auto_laser_control** (Parameter `Mode=0`) deaktiviert werden.

  Wenn umgekehrt allein die geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung aktiviert werden soll, dann kann die positionsabhängige Korrektur effektiv dadurch deaktiviert werden, dass mit **load_position_control** (`Name` = 0) für den vollen Positionsbereich eine Skalierungsfunktion mit konstantem Wert 1.0 geladen wird (*Scale*(*Position*)`=1.0`, entspricht dem Initialisierungszustand nach **load_program_file**, siehe auch Seite 162).
- Wenn zusätzlich eine vektorkontrollierte Lasersteuerung mit gleicher Signalparameter-Auswahl (Parameter `Ctrl`) aktiviert ist, dann wird der 100%-Wert (Parameter `Value`) gegebenenfalls durch die vektorkontrollierte Lasersteuerung verändert (siehe Seite 167).
- Der gesamte Korrekturfaktor (gegebenenfalls aus positions- *und* geschwindigkeits- oder encodergeschwindigkeitsabhängiger Lasersteuerung) kann den Wert 4.0 nicht überschreiten und wird gegebenenfalls auf diesen Maximalwert geclippt.
- Da die Laserleistung nicht streng proportional zu den Ausgangswerten des ausgewählten Signalparameters sein muss, kann man zusätzlich mit **load_auto_laser_control** eine Nichtlinearitätskurve laden, in der dieser Zusammenhang definiert wird. Der gesamte Korrekturfaktor wird dann (gegebenenfalls nach Clippen, s.o.) entsprechend dieser Nichtlinearitätskurve modifiziert, d.h. mit einem entsprechenden Nichtlinearitätsfaktor multipliziert. Die Nichtlinearitätskurve muss vom Benutzer für seine jeweilige Situation ermittelt und in einer Text-Datei definiert werden (siehe "Hinweise zum Laden und zur Ermittlung der Nichtlinearitätskurve", Seite 163).
- Zusätzlich kann der ausgewählte Signalparameter weder den mit **set_auto_laser_control** (Parameter `MinValue` und `MaxValue`) noch den allgemein maximal zulässigen Wertebereich überschreiten. Gegebenenfalls wird der Ausgabewert (gegebenenfalls nach erfolgter Modifizierung mit dem Nichtlinearitätsfaktor) entsprechend geclippt.

- Nach erfolgter Initialisierung der positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängigen Lasersteuerung mit **set_auto_laser_control** wird der ausgewählte Signalparameter jedesmal, wenn der Laser am Ende einer Beschriftung ausgeschaltet wird oder wenn die positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung mit **set_auto_laser_control** auf einen anderen `Ctrl`-Parameter gesetzt oder mit **set_auto_laser_control** (`Ctrl` = 0) deaktiviert wird, auf den folgenden Wert gesetzt:
  - der 12-Bit-Ausgabewert am ANALOG OUT1- oder ANALOG OUT2-Ausgang, der Ausgabewert am 8-Bit-Digital-Ausgang bzw. der Ausgabewert am 16-Bit-Digital-Ausgang auf den zuvor mit **set_port_default** oder **set_laser_off_default** definierten Default-Wert (bzw. wenn kein Default-Wert definiert wurde auf den maximal zulässigen Wert)
  - die Pulslänge (`PulseLength`) oder die Ausgabeperiode (`HalfPeriod`) der Lasersignale LASER1 und LASER2 auf den mit **set_auto_laser_control** (Parameter `Value`) festgelegten 100%-Wert (eine zusätzlich aktivierte vektorkontrollierte Lasersteuerung wirkt sich hier nicht aus).
- Der zu steuernde Signalparameter der positionsabhängigen Lasersteuerung, der 100%-Wert und seine ihm zugeordneten Grenzwerte können außer mit **set_auto_laser_control** auch mit den Befehlen **set_auto_laser_params** oder **set_auto_laser_params_list** festgelegt werden.

Hinweise zum Laden der Skalierungsfunktion

Der Befehl **load_position_control** lädt eine Tabelle für die Skalierungsfunktion *Scale*(*Position*) aus einer ASCII-Textdatei. Die Textdatei kann eine oder mehrere Tabellen enthalten.

Jede Tabelle kann bis zu 50 Datenpunkte (*Position* | *Scale*(*Position*)) für verschiedene Positionen enthalten. Aus diesen Daten bestimmt die RTC5 durch lineare Interpolation die Skalierungsfunktion *Scale*(*Position*).

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Anweisung (Caption)

  `[PositionCtrlTable<No>]`

  beginnen. Dabei steht `<No>` für die Tabellennummer (ein nicht-negativer Integer-Wert).
- Enthält die Tabelle mehrere `[PositionCtrlTable<No>]`-Einträge zur gleichen `<No>`, werden nur die Anweisungen nach dem ersten Eintrag verwendet, diejenigen nach weiteren Einträgen dagegen ignoriert. Es werden nur die Anweisungen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (*Position* | *Scale*(*Position*)) wird durch zwei Anweisungen dargestellt:

  ```
  Position<n> = <Value>
  Scale<n> = <Value>
  ```

  Dabei entspricht `<n>` dem Index des Datenpunktes (Integer-Wert, 1 ≤ `<n>` ≤ 50). Die Werte `<Value>` können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Verwenden Sie Punkte (.) als Dezimal-Trennzeichen.
- 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 Positionswert *Position*, so wird der Datenpunkt mit dem größten Index `<n>` verwendet, die anderen werden ignoriert. Die Gleichheit wird auf ±0.01 geprüft.
- Der Positionswert wird radial als Abstand des zu markierenden Punktes vom Koordinatenmittelpunkt (= $(x^2 + y^2)^{1/2}$) als Prozentwert (in Prozent der halben Bildfeldseitenlänge) angeben.
  Beispiel: (X_{max}|0) entspricht 100%,
  (X_{max}|Y_{max}) entspricht $2^{1/2} \times 100\%$
  (mit X_{max},Y_{max} = 2^{19} bzw. 2^{15} im RTC4-Kompatibilitätsmodus).

- Für <Value> gelten die folgenden Wertebereiche:
  0.0 ≤ *Position* ≤ 150.0 und
  0.0 ≤ *Scale*(*Position*) ≤ 4.0.
  (Die RTC5 verwendet die Skalierungsfunktion nur für den Wertebereich 0.0 ≤ *Position* ≤ 141.42, berücksichtigt bei der Interpolation aber auch die angegebenen Werte im Bereich
  141.42 ≤ *Position* ≤ 150.0)
- 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 der Befehl **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 legale 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, so dass effektiv keine positionsabhängige Korrektur durchgeführt wird. Ein mit **set_auto_laser_control** (Parameter Value) festgelegter 100%-Wert wird mit S multipliziert. Für S = 1.0 ist daher die ortsabhängige Korrektur vollständig ausgeschaltet. Dies kann alternativ auch erreicht werden mit Name = 0 in **load_position_control**.

Bei der Initialisierung der RTC5 (mit **load_program_file**) wird für den vollen Positionsbereich die Skalierungsfunktion *Scale*(*Position*) = 1.0 geladen, so dass effektiv keine positionsabhängige Korrektur durchgeführt wird.

Hinweise zum Laden und zur Ermittlung der Nichtlinearitätskurve

Der Befehl **load_auto_laser_control** lädt eine Tabelle für die Nichtlinearitätskurve *Scale*(*Percent*) aus einer ASCII-Textdatei. Die Textdatei kann eine oder mehrere Tabellen enthalten.

Jede Tabelle kann bis zu 50 Datenpunkte (*Percent* | *Scale*(*Percent*)) für verschiedene Prozentwerte enthalten. Aus diesen Daten bestimmt die RTC5 durch lineare Interpolation die Nichtlinearitätskurve *Scale*(*Percent*).

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Anweisung (Caption)

  [AutoLaserCtrlTable<No>]

  beginnen. Dabei steht <No> für die Tabellennummer (ein nicht-negativer Integer-Wert).
- Enthält die Tabelle mehrere [AutoLaserCtrlTable<No>]-Einträge zur gleichen <No>, werden nur die Anweisungen nach dem ersten Eintrag verwendet, diejenigen nach weiteren Einträgen dagegen ignoriert. Es werden nur die Anweisungen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (*Percent* | *Scale*(*Percent*)) wird durch zwei Anweisungen dargestellt:

  Percent<n> = <Value>
  Scale<n> = <Value>

  Dabei entspricht <n> dem Index des Datenpunktes (Integer-Wert, 1 ≤ <n> ≤ 50). Die Werte <Value> können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Verwenden Sie Punkte (.) als Dezimal-Trennzeichen.
- 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 Prozentwert *Percent*, so wird der Datenpunkt mit dem größten Index <n> verwendet, die anderen werden ignoriert. Die Gleichheit wird auf ±0.01 geprüft.

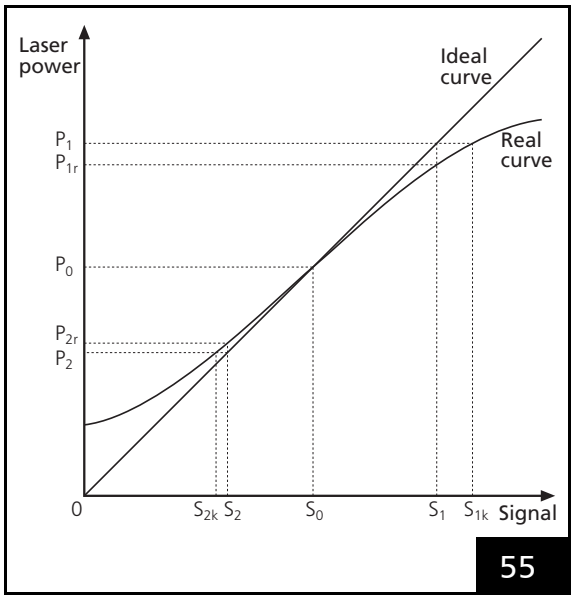
- Der Prozentwert bezieht sich auf den mit **set_auto_laser_control** (Parameter `Value`) festgelegten 100%-Wert (letzterer kann zusätzlich durch eine vektorkontrollierte Lasersteuerung verändert sein, siehe Seite 166).
  Im folgenden Beispiel wird für das 1,5-fache des 100%-Werts ein Nichtlinearitätsfaktor von 1,2 festgelegt:

  ```
  Percent<n> = 150
  Scale<n> = 1.2
  ```

- Für `<Value>` gelten die Wertebereiche $0.0 \leq$ *Percent* ≤ 400.0 und $0.0 \leq$ *Scale*(*Percent*) ≤ 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 der Befehl **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 legale Wert wird nach unten aufgefüllt). Analog bei *Percent* = 400.0 mit Max(`Scale<i>`).

Bei der Initialisierung der RTC5 (mit **load_program_file**) wird für den vollen Prozentbereich die Funktion *Scale*(*Percent*) `= 1.0` geladen (keine Nichtlinearität). Dies kann alternativ auch mit `Name` = 0 in **load_auto_laser_control** erreicht werden.

Die Ermittlung der Nichtlinearitätskurve wird anhand des Beispieldiagramms in Abb. 55 erklärt.



Laserleistungsverlauf – Beispiel zur Ermittlung der Nichtlinearitätskurve

Die gerade Linie im Diagramm beschreibt ein ideal lineares Verhalten für den Zusammenhang zwischen Laserleistung und Lasersteuersignalparameter (der Begriff Laserleistung steht hier auch für die Pulsfrequenz = 0,5/`LaserHalfPeriod`). Als Beispiel für einen realistischen Zusammenhang ist eine gekrümmte Linie eingezeichnet: Die Laserleistung steigt hier langsamer an als der Signalparameter bzw. fällt langsamer ab.

S_0 bezeichnet den Signalparameterwert, der als 100%-Wert definiert wurde und P_0 die zugehörige Laserleistung. Am Punkt (S_0 | P_0) fallen die beiden Kurven zusammen (entspricht dem Datenpunkt `Percent0` = 100, `Scale0` = 1.0).

Eine Erhöhung (Verringerung) des Signalparameters auf S_1 (S_2) führt zur idealen Laserleistung P_1 (P_2) bzw. 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 wären dann in diesem Beispiel 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

Geschwindigkeitsabhängige Lasersteuerung

Zur Initialisierung der geschwindigkeitsabhängigen Lasersteuerung muss mit **set_auto_laser_control** festgelegt werden, welcher Signalparameter angepasst werden soll (Parameter `Ctrl`) und welche Größe v_a bei der Berechnung der geschwindigkeitsabhängigen Korrektur zu Grunde gelegt wird (Parameter `Mode`).

Bei nachfolgenden Mark- und Arc-Befehlen (auch bei zeitbemessenen Mark-Befehlen) führt die RTC5 dann eine geschwindigkeitsabhängige (d.h. eine von der aktuellen Geschwindigkeit abhängige) automatische Korrektur des ausgewählten Signalparameters durch: Der mit **set_auto_laser_control** (Parameter `Value`) festgelegte 100%-Wert wird mit einem Korrekturfaktor von v_a/v_s multipliziert (Ausnahme: für die Ausgabeperiode `HalfPeriod` beträgt der Korrekturfaktor v_s/v_a). Dabei ist v_s die mit **set_mark_speed** definierte Markiergeschwindigkeit und v_a abhängig von der Festlegung mit **set_auto_laser_control**:

- Für `Mode` = 1 ist v_a die tatsächliche Sollgeschwindigkeit, wie sie sich aus der Schrittweite des Mikrovektors in Sollkoordinaten ergibt.
- Für `Mode` = 2 (nur bei i*DRIVE*-Scan-Systemen wählbar) ist v_a die vom Scan-System rückgelesene aktuelle Geschwindigkeit gemäß $(v_x^2 + v_y^2)^{1/2}$

Hinweise

- Zu **set_auto_laser_control** `Mode` = 1:
  Bei Mark- oder Arc-Befehlen kann sich die tatsächliche Sollgeschwindigkeit von der mit **set_mark_speed** vorgegebenen Markiergeschwindigkeit unterscheiden, weil die Markierungszeit stets ein ganzzahliges Vielfaches von 10 µs sein muss (der Befehl wird über eine entsprechende Zahl von Mikroschritten ausgeführt) und die Schrittweite (und damit die effektive Markiergeschwindigkeit = tatsächliche Sollgeschwindigkeit) automatisch daran angepasst wird. Die tatsächliche Sollgeschwindigkeit ist deshalb insbesondere bei kurzen Linien merklich variabel. Solch eine längenabhängige Geschwindigkeitsvariation könnte z.B. – ohne geschwindigkeitsabhängige Lasersteuerung – eine Variation der Linienstärken bei Schraffuren bewirken.

  Bei zeitbemessenen Mark-Befehlen wird die Sollgeschwindigkeit generell entsprechend der jeweils vorgegebenen Vektorlänge und Markierungszeit ermittelt und unterscheidet sich deswegen in aller Regel von der mit **set_mark_speed** definierten Markiergeschwindigkeit.

  Geschwindigkeitsvariationen innerhalb eines Markierbefehls können mit `Mode` = 1 nicht ausgeglichen werden. Die meisten Standard-Scan-Systeme liefern allerdings auch nicht die dafür benötigte Information an die RTC5 zurück.
- Für **set_auto_laser_control** `Mode` = 2 wird die Korrektur immer entsprechend der aktuellen Geschwindigkeit durchgeführt. Auf diese Weise kann die automatische Lasersteuerung auch Geschwindigkeitsänderungen während eines Markierbefehls (z.B. während der Beschleunigungs- oder Abbremsphase) ausgleichen. Allerdings wird die aktuelle Geschwindigkeit aufgrund von Laufzeiten zum Scan-Kopf und zurück um einige wenige Takte verzögert verwertet.
  Diese Funktionalität ist nur dann sinnvoll nutzbar, wenn der Scan-Kopf mit einer speziellen Firmware ausgestattet ist, die diese Laufzeitverzögerung ausgleicht. Falls Sie diese Funktionalität nutzen wollen, aber noch keinen entsprechend ausgestatteten Scan-Kopf besitzen, dann wenden Sie sich an SCANLAB.

- Mit **set_auto_laser_control** `Mode` = 6 kann die aktuelle Galvanometerscanner-Geschwindigkeit (`Mode` = 2) mit der momentanen Encodergeschwindigkeit (`Mode` = 5) kombiniert werden. Dazu muss aber die "Processing on the fly"-Korrektur freigeschaltet und mit **set_fly_x**, **set_fly_y** oder **set_fly_2d** aktiviert sein. Als Referenzgeschwindigkeit dient (wie bei `Mode` = 2) die aktuelle Markiergeschwindigkeit. Der für `Mode` = 5 verwendete Befehl **set_encoder_speed** bleibt im `Mode` = 6 unberücksichtigt (siehe auch Abschnitt "Encodergeschwindigkeitsabhängige Lasersteuerung", Seite 168).
- Ein variierender Lasereintrag infolge einer expliziten Änderung der Markiergeschwindigkeit zwischen zwei Markierbefehlen durch Aufruf des **set_mark_speed**-Befehls wird *nicht* mit der geschwindigkeitsabhängigen Lasersteuerung ausgeglichen.
- Beachten Sie auch alle Hinweise zur positionsabhängigen Lasersteuerung im Abschnitt "Allgemeine Hinweise", Seite 161: Der Korrekturfaktor der geschwindigkeitsabhängigen Lasersteuerung wird gegebenenfalls zusätzlich mit einem Faktor aus der positionsabhängigen Lasersteuerung und einem Nichtlinearitätsfaktor beaufschlagt, bevor er auf den 100%-Wert angewandt wird.
- Wenn ein negatives LaserOn-Delay eingestellt ist, um das Material vor einer Markierung vorzuheizen (siehe Seite 111), ist die geschwindigkeitsabhängige Lasersteuerung u. U. eher kontraproduktiv, da diese die Laserleistung zu Beginn einer Markierung während der Beschleunigungsphase herunterregelt. Dem kann aber auch mit dem **set_auto_laser_control**-Parameter `MinValue` bzw. `MaxValue` entgegengesteuert werden.
- Der zu steuernde Signalparameter der geschwindigkeitsabhängigen Lasersteuerung, der 100%-Wert und seine ihm zugeordneten Grenzwerte können außer mit **set_auto_laser_control** auch mit den Befehlen **set_auto_laser_params** oder **set_auto_laser_params_list** festgelegt werden.

Vektorkontrollierte Lasersteuerung

Die vektorkontrollierte Lasersteuerung ermöglicht es, einen Signalparameter entlang eines Mark- oder Jump-Vektors linear zu variieren.

Zur Initialisierung der vektorkontrollierten Lasersteuerung muss mit **set_vector_control** festgelegt werden, welcher Signalparameter variiert werden soll (Parameter `Ctrl`) und auf welchen Anfangswert dieser Signalparameter bei der Initialisierung gesetzt werden soll (Parameter `Value`).

Bei nachfolgenden Para-Mark- und Para-Jump-Befehlen (parametrisierten Mark- und Jump-Befehlen) führt die RTC5 dann parallel zur Koordinatenvariation eine lineare Variation des ausgewählten Signalparameters durch. Startwert für den ersten Para-Befehl ist der mit **set_vector_control** festgelegte Anfangswert (Parameter `Value`)(1). Der Signalparameter wird wie eine normale Koordinate mikrovektorisiert und linear zum jeweils angegebenen Endwert (Parameter `P`) variiert. Der Endwert ist Startwert für einen nachfolgenden Para-Befehl.

Es stehen die folgenden Para-Mark- und Para-Jump-Befehle zur Verfügung:

- **para_mark_abs**, **timed_para_mark_abs**
- **para_mark_rel**, **timed_para_mark_rel**
- **para_jump_abs**, **timed_para_jump_abs**
- **para_jump_rel**, **timed_para_jump_rel**

Bei freigeschalteter 3D-Option können auch 3D-Para-Befehle ausgeführt werden:

- **para_mark_abs_3d**, **timed_para_mark_abs_3d**
- **para_mark_rel_3d**, **timed_para_mark_rel_3d**
- **para_jump_abs_3d**, **timed_para_jump_abs_3d**
- **para_jump_rel_3d**, **timed_para_jump_rel_3d**

Zur Markierung von Punkten steht der Para-Befehl **para_laser_on_pulses_list** zur Verfügung.

(1) Für `Ctrl` = 7 (Fokusverschiebung) wird gegebenenfalls ein harter Sprung in der z-Koordinate auf den Anfangswert ausgeführt.

Hinweise

- Falls für den Startwert eines Para-Befehls ein anderer Wert erwünscht ist als der Endwert des vorherigen Para-Befehls, kann die entsprechende Signalparameter-Änderung zwischen den beiden Befehlen per **para_jump_rel** mit einem Sprung "auf die gleiche Stelle" ausgeführt werden oder durch erneute Erteilung des Befehls **set_vector_control** mit neuem Startparameter. Wird stattdessen zwischen zwei Para-Befehlen ein Listenbefehl wie **set_laser_pulses**, **write_da_x_list** oder **write_8bit_port_list** aufgerufen, so ändert sich zwar zwischenzeitlich der Signalparameter. Als Startwert für den zweiten Para-Befehl wird aber trotzdem wieder der Endwert des ersten Para-Befehls eingestellt. Kontrollbefehle wie **set_vector_control** oder Befehle zur expliziten Änderung des Einstellwertes zum Parameter `Ctrl` (z. B. **write_da_x** oder **write_8bit_port**) sollten während der Abarbeitung einer Liste von Para-Befehlen vermieden werden.
- Die vektorkontrollierte Lasersteuerung wirkt sich *direkt* nur bei Para-Befehlen aus (normale Mark- oder Jump-Befehle lassen den aktuell eingestellten Wert zum Parameter `Ctrl` unverändert), indirekt u. U. aber auch bei normalen Mark- und Arc-Befehlen oder zeitbemessenen Mark-Befehlen (siehe auch "Allgemeine Hinweise", Seite 161). Wenn nämlich neben der vektorkontrollierten Lasersteuerung (`Ctrl` = 1…6) gleichzeitig eine positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung mit derselben Signalparameter-Auswahl aktiviert ist, dient der aktuelle Signalparameterwert der vektorkontrollierten Lasersteuerung als Basis für den 100%-Wert der positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängigen Lasersteuerung:
  - Bei Initialisierung der vektorkontrollierten Lasersteuerung mit **set_vector_control** wird der 100%-Wert auf den mit **set_vector_control** (Parameter `Value`) festgelegten Anfangswert gesetzt.
  - Bei nachfolgenden Para-Befehlen wird der 100%-Wert entsprechend des aktuellen Signalparameterwerts kontinuierlich variiert.
  - Bei Deaktivierung der vektorkontrollierten Lasersteuerung mit **set_vector_control** (Parameter `Ctrl=0`) wird der 100%-Wert wieder auf den Wert bei der Initialisierung zurückgesetzt (Parameter `Value` aus **set_auto_laser_control**, **set_auto_laser_params** oder **set_auto_laser_params_list**).
- Bei Para-Befehlen wirkt sich eine mit **load_auto_laser_control** geladene Nichtlinearitätskurve nur dann auf den mit **set_vector_control** ausgewählten Signalparameter aus, wenn die positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung mit derselben Signalparameter-Auswahl aktiviert ist (`Ctrl` = 1…6).
- Der Sky-Writing-Modus (siehe Seite 127) wird bei der Ausführung von Para-Befehlen *nicht* berücksichtigt (aber auch nicht deaktiviert).
- Nach Aktivierung der vektorkontrollierten Lasersteuerung mit **set_vector_control**(`Ctrl` = 7) wird die Fokusverschiebung bei 2D-oder 3D-Para-Befehlen verändert, sofern die 3D-Option freigeschaltet und eine 3D-Korrekturdatei geladen und zugewiesen ist.

Encodergeschwindigkeitsabhängige Lasersteuerung

Bei der encodergeschwindigkeitsabhängigen Lasersteuerung kann ein Lasersignalparameter entsprechend der aktuell eingelesenen Encodergeschwindigkeit (Zählerpulsrate) eines Encoder-Zählers (oder beider Encoder-Zähler) gesteuert werden. Dadurch kann z. B. der Lasereintrag bei bewegten Objekten an die aktuelle Fördergeschwindigkeit des Transportsystems angepasst werden.

Zur Initialisierung der encodergeschwindigkeitsabhängigen Lasersteuerung muss mit **set_auto_laser_control**, **set_auto_laser_params** oder **set_auto_laser_params_list** festgelegt werden, welcher Signalparameter (Parameter `Ctrl`) entsprechend der aktuellen Encodergeschwindigkeit (Zählerpulsrate) angepasst werden soll (Parameter `Mode` = 5).

Mit **set_encoder_speed_ctrl** bzw. **set_encoder_speed** kann ausgewählt werden, welcher der beiden Encoder für die Geschwindigkeitsmessung herangezogen werden soll (bzw. ob beide Encoder zur Ermittlung einer vektoriellen Encoder-Geschwindigkeit verwendet werden sollen). Außerdem kann damit eine Soll-Encodergeschwindigkeit und ein Glättungsfaktor für einen 2-Punkte-Tiefpass festgelegt werden (siehe Befehlsbeschreibung).

Bei nachfolgenden Mark- und Arc-Befehlen (auch bei zeitbemessenen Mark-Befehlen) führt die RTC5 dann eine automatische Korrektur des ausgewählten Signalparameters entsprechend der aktuellen Encodergeschwindigkeit durch: Der mit **set_auto_laser_control**, **set_auto_laser_params** oder **set_auto_laser_params_list** (Parameter `Value`) festgelegte 100%-Wert wird mit einem Korrekturfaktor von v_a/v_s multipliziert (Ausnahme: für die Ausgabeperiode HalfPeriod beträgt der Korrekturfaktor v_a/v_s). Dabei ist v_s die mit **set_encoder_speed** definierte Soll-Encodergeschwindigkeit und v_a die aktuelle Encodergeschwindigkeit.

Hinweise

- Die encodergeschwindigkeitsabhängige Lasersteuerung ist mit der positionsabhängigen und vektorkontrollierten Lasersteuerung und ab DLL 540 auch mit der geschwindigkeitsabhängigen Lasersteuerung kombinierbar.
- Beachten Sie auch alle Hinweise zur positionsabhängigen Lasersteuerung im Abschnitt "Allgemeine Hinweise", Seite 161: Der Korrekturfaktor der encodergeschwindigkeitsabhängigen Lasersteuerung wird gegebenenfalls zusätzlich mit einem Faktor aus der positionsabhängigen Lasersteuerung und einem Nichtlinearitätsfaktor beaufschlagt, bevor er auf den 100%-Wert angewandt wird.

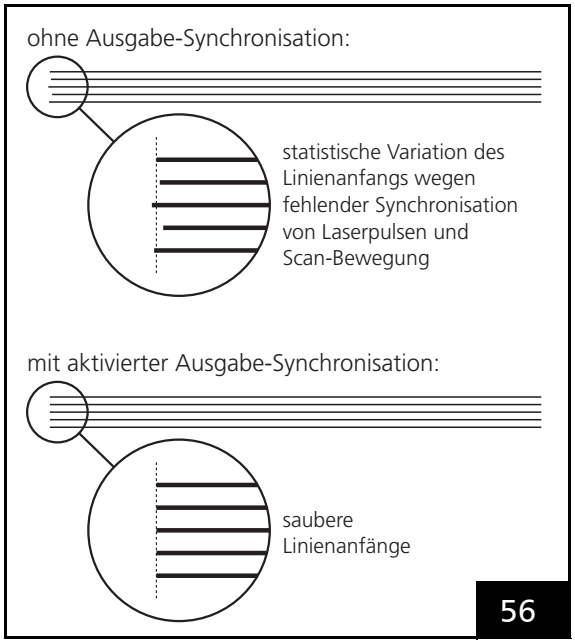
7.4.10 Ausgabe-Synchronisation

Damit die Scan-Bewegung des Scan-Systems (bei allen Markierbefehlen) mit den Laserpulssignalen eines freilaufenden Lasers synchronisiert werden kann, stellt die RTC5 die sogenannte (Galvanometerscanner-)Ausgabe-Synchronisation als Funktionalität bereit.

Das Laserpulssignal (Master-Clock), auf das synchronisiert werden soll, muss am Digital-Eingang DIGITAL IN1 der LASER-Buchse (siehe Seite 48) eingespeist werden.

Die Ausgabe-Synchronisation kann mit **set_laser_control** über das Bit#6 ein- und ausgeschaltet werden.

Eine aktivierte Ausgabe-Synchronisation wirkt sich auf alle Markierbefehle (Mark, Arc und Ellipse) aus. Die RTC5 ermittelt zu Beginn eines jeden Markierbefehls die Zeitverschiebung zwischen dem Startpunkt des Markierbefehls und dem ersten Laserpuls nach dem LaserOn-Delay und korrigiert die Ausgabewerte an das Scan-System entsprechend. Auf diese Weise stellt die RTC5 sicher, dass die Position der Galvanometerscanner, bei der der erste Laserpuls nach dem LaserOn-Zeitpunkt ausgegeben wird, nicht von der zufälligen Phasenlage des Laserpulses abhängt. Flatterhafte Linienbilder werden dadurch vermieden (siehe Abb. 56).



Beispiel für markiertes Linienbild bei freilaufendendem Laser

Hinweise (gültig ab Version RBF 527)

- Das Laserpulssignal, auf das synchronisiert werden soll, muss eine Frequenz zwischen 50 kHz und 6,4 MHz und eine Pulslänge über 0,080 µs aufweisen.
- Die Ausgabeperiode der Lasersteuersignale muss im Anwenderprogramm (vor Aktivieren der Ausgabe-Synchronisation) mit **set_laser_pulses**, **set_laser_pulses_ctrl** oder **set_laser_timing** entsprechend der Laserpulsfrequenz festgelegt werden. Das gilt auch dann, wenn die Standby-Signale von Lasermode 4 (siehe Seite 154) für den externen Input verwendet werden.
- Das LaserOn-Delay muss für Frequenzen zwischen 50 kHz und 100 kHz auf einen Wert über 50 µs festgelegt werden, sonst über 40 µs.
- Kommt innerhalb von 20 µs nach Start eines Markierbefehls kein Laserpuls am Digital-Eingang DIGITAL IN1 an, so wird für die Ausgabe dieses Befehls keine Ausgabe-Synchronisation durchgeführt.
- Die Ausgabe-Synchronisation kann auch zusammen mit Sky-Writing verwendet werden.
- Die Ausgabe-Synchronisation kann nicht in Verbindung mit dem Pixelmode verwendet werden.

7.5 Markieren von Uhrzeit, Datum und Seriennummern

Bei vielen Anwendungen müssen die aktuelle Uhrzeit und das aktuelle Datum oder die Seriennummern von Produkten markiert werden. Dazu verfügt der Befehlssatz der RTC5 über eine Reihe von entsprechenden Befehlen.

Bevor Uhrzeit, Datum oder Seriennummern markiert werden können, müssen die zugehörigen Zeichen und Zeichenfolgen als indizierte Zeichen und Zeichenfolgen definiert werden. Dies wird in "Definition von indizierten Zeichenfolgen für Uhrzeit, Datum und Seriennummer" auf Seite 88 beschrieben. Für die Beschriftung von Uhrzeit/Datum und die Seriennummerbeschriftung können dabei separate Zeichenfolgen definiert werden.

7.5.1 Markieren von Uhrzeit und Datum

Bevor Uhrzeit und Datum markiert werden können, muss die RTC5 (nach jedem Neustart) mit der PC-Zeit abgeglichen und müssen die aktuellen Werte für Uhrzeit und Datum abgefragt werden:

- Die Kalibrierung der 24-Stunden-Zeit der RTC5 und des Kalenders erfolgt durch einen Abgleich mit der PC-Zeit über den Befehl **time_update**. Danach zählt die RTC5 Datum und Uhrzeit intern mit dem quarzgesteuerten 10 µs-Takt weiter.
- Der aktuelle Wert für Uhrzeit und Datum wird mit dem Befehl **time_fix**, **time_fix_f** oder **time_fix_f_off** abgefragt und abgespeichert.

Die Uhrzeit (Stunden, Minuten, Sekunden) kann anschließend mit dem Befehl **mark_time** bzw. **mark_time_abs** und das Datum (Jahr, Monat, Tag, Wochentag) mit dem Befehl **mark_date** bzw. **mark_date_abs** markiert werden. Diese Markierbefehle verwenden die mit **time_fix**, **time_fix_f** bzw. **time_fix_f_off** ausgelesene Uhrzeit und Datum und starten die Ausgabe der zuvor für Uhrzeit und Datum definierten indizierten Zeichenfolgen.

Zur Markierung von Datum und Uhrzeit kann man bei der RTC5 wahlweise das gregorianische oder das julianische Datum, sowie eine 12- oder 24-Stunden-Uhrzeit auswählen.

Zum Markieren eines Haltbarkeitsdatums (o.ä.) kann mit **time_fix_f_off** auch eine Vordatierung zum aktuellen Datum und zur aktuellen Uhrzeit fixiert werden.

7.5.2 Markieren von Seriennummern

Mit dem Befehl **mark_serial** bzw. **mark_serial_abs** können bis zu 12-stellige Seriennummern markiert werden. Die Befehle starten die Ausgabe der Befehlslisten für die Ziffern 0…9, die zuvor als indizierte Zeichen bzw. Zeichenfolgen definiert wurden. Dabei kann zusätzlich festgelegt werden, ob und wie führende Nullen dargestellt werden.

Die RTC5-Karte verwaltet 4 Seriennummernsätze (jeweils Seriennummer und Schrittweite). Bei der Initialisierung mit **load_program_file** wird der Seriennummernsatz 0 ausgewählt. Sollen mehrere Seriennummernsätze verwendet werden, so muss der jeweilige Seriennummernsatz zuvor mit **select_serial_set** bzw. **select_serial_set_list** ausgewählt werden (siehe auch Hinweise unten).

Für jeden Seriennummernsatz kann mit dem Kontrollbefehl **set_serial** bzw. **set_serial_step** oder mit dem Listenbefehl **set_serial_step_list** eine Startseriennummer (max. 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 eines **mark_serial**-Befehls (bzw. **mark_serial_abs**-Befehls) wird die aktuelle Seriennummer des aktuell ausgewählten Seriennummernsatzes BCD-codiert und noch vor der Ausführung der BCD-codierten Seriennummernmarkierung um die angegebene Schrittweite erhöht. Soll eine Seriennummer ausgelassen werden, so kann mit dem **mark_serial**-Befehl 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 (bzw. nach **load_program_file** auf den Seriennummernsatz 0).
- Die Nummer des zuletzt mit **select_serial_set_list** ausgewählten Seriennummernsatzes und die aktuelle Seriennummer dieses Seriennummernsatzes (also in der Regel die zuletzt mit **mark_serial** oder **mark_serial_abs** markierte Seriennummer) 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 (im Bedarfsfall kann die Nummer dann mit **set_serial** oder **set_serial_step** korrigiert werden, allerdings nur im 10-stelligen Bereich).
- 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 (z. B. um parallel zur Markierung eines Seriennummernsatzes die Parameter der anderen Seriennummernsätze festzulegen; eine Parameteranpassung des aktuell zur Markierung verwendeten Seriennummernsatzes ist übrigens nicht unterbunden). Die Kontrollbefehle **set_serial_step** und **set_serial** wirken immer auf den zuletzt mit **select_serial_set** ausgewählten Seriennummernsatz (bzw. nach **load_program_file** auf den Seriennummernsatz 0).
- **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).
- Bei älteren Software-Versionen (DLL 536, OUT 536 und früher) gab es nur einen Seriennummernsatz. Da ab Version DLL 537, OUT 537 nach **load_program_file** der Seriennummernsatz 0 ausgewählt wird, können bereits bestehende Anwenderprogramme unverändert weiter verwendet werden (also auch ohne die neuen Befehle **select_serial_set** und **select_serial_set_list**).
- Mit dem Kontrollbefehl **set_max_counts** lässt sich die maximale Anzahl an externen Listen-Starts und damit die maximale Anzahl an extern gestarteten Beschriftungen einstellen. Die Anzahl von externen Listen-Starts kann mit dem Kontrollbefehl **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 allerdings nicht zwischen den verschiedenen Seriennummernsätzen.

8 Erweiterte Funktionen zur Scan-Kopf- und Lasersteuerung

8.1 i*DRIVE*-Funktionen

In den i*DRIVE*-Scan-Systemen von SCANLAB (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario$SCAN_{de}$) kommt die i*DRIVE*-Technologie von SCANLAB zum Einsatz. Die i*DRIVE*-Technologie realisiert ein erweitertes Regelungs- und Ansteuerkonzept durch eine voll digitale Regelelektronik mit stark erweitertem Funktionsumfang und ein erweiteres Übertragungsprotokoll zwischen Regelelektronik und Ansteuerung (siehe auch folgenden Abschnitt).

Dadurch können eine Reihe von Einstellungen des jeweiligen Scan-Systems angepasst werden. Benutzer können z. B.

- einstellen, welche Daten das Scan-System an die Ansteuerung zurücksenden soll,
- zwischen verschiedenen dynamischen Einstellungen (Tunings) auswählen,
- den PosAcknowledge-Schwellwert einstellen,
- die effektive Kalibrierung des Scan-Kopfs einstellen,
- das Startverhalten des Scan-Kopfs einstellen oder
- eine Fehlerdiagnose oder einen Funktionstest der Datenübertragung durchführen.

Zusammen mit einer RTC5 können die meisten i*DRIVE*-Funktionen mit dem **control_command**-Befehl ausgeführt werden. Während der Ausführung dieses Befehls werden keine Positionsdaten von der RTC5 an das Scan-System übergeben. Die Bewegung der Galvanometerscanner wird dadurch gegebenenfalls kurzzeitig unterbrochen. Für Scan-Systeme mit SL2-100-Interface (die ohne XY2-100-Konverter betrieben werden) können bestimmte i*DRIVE*-Funktionen aber auch parallel zur normalen Kommunikation mit dem Befehl **send_user_data** ausgeführt werden, so dass weder die Übertragung der Positionsdaten noch die Bewegung der Galvanometerscanner unterbrochen wird.

Alle i*DRIVE*-Funktionen werden im Folgenden näher erläutert.

8.1.1 Übertragungsprotokoll

Die Datenübertragung zwischen RTC5 und Scan-System erfolgt nach dem SL2-100-Protokoll. Dieses Protokoll unterstützt den vollen Funktionsumfang der i*DRIVE*-Technologie. Bei i*DRIVE*-Scan-Systemen können damit u. a. die Statussignale der X- und Y-Achse separat und simultan ausgewertet werden. Die Statussignale der Z-Achse eines 3D-Scan-Systems können simultan über den Kanal des Scan-Kopf-Anschlusses, an dem die Z-Achse angeschlossen ist, rückgelesen werden.

Für i*DRIVE*-Scan-Systeme, die mit einem herkömmlichen XY2-100- bzw. XY2-100 Enhanced-Interface ausgestattet sind, kann der XY2-100-Konverter von SCANLAB verwendet werden. Dieser konvertiert die RTC5-Ansteuersignale gemäß SL2-100-Protokoll in Ansteuersignale gemäß XY2-100-Protokoll und Scan-System-Statussignale gemäß XY2-100-Protokoll in Statussignale gemäß SL2-100-Protokoll (siehe Seite 43). Diese Konvertierung unterstützt den vollen Funktionsumfang der i*DRIVE*-Technologie bisheriger Scan-Systeme mit XY2-100-Interface.

Scan-Systeme, die mit einem SL2-100-Interface ausgestattet sind, können nur von einer RTC5 ohne Konverter angesteuert werden.

8.1.2 Anpassen des Rückmeldeverhaltens

Auswahl von Datentypen

Die digitale Regelung von i*DRIVE*-Scan-Systemen erlaubt eine vielfältige Rückgabe von Datensignalen an die RTC5. Jede Achse verfügt über einen eigenen Statuskanal, auf dem alle 10 µs Daten zur Ansteuerkarte übertragen werden: Der STATUS-Kanal ist für die X-Achse (Galvanometerscanner 2), der STATUS1-Kanal 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 dem Befehl **control_command** kann ausgewählt werden, welche Daten das Scan-System der Ansteuerkarte zurücksendet. Die auswählbaren Datentypen werden ausführlich im Handbuch des jeweiligen Scan-Systems und in der Befehlsreferenz des **control_command**-Befehls beschrieben. Die ausgewählte Datenquelle wird so lange übertragen, bis eine andere selektiert wird.

i*DRIVE*-Scan-Systeme mit SL2-100-Interface (die ohne XY2-100-Konverter betrieben werden) können zusätzlich, parallel zur oben beschriebenen "normalen" Statusrückgabe weitere, sich langsam verändernde Statusinformationen (z.B. die aktuelle Temperatur der Galvanometerscanner) rückübertragen. Mit dem Befehl **send_user_data** kann ausgewählt werden, welche Daten das Scan-System zurücksendet. Beim **send_user_data**-Befehl erfolgt die Datenübertragung zum Scan-System über ein sogenanntes User-Datenbit, verteilt auf mehrere Übertragungszyklen, parallel zur normalen Kommunikation. Im Gegensatz zum **control_command** wird hier weder die Übertragung der Positionsdaten noch die Bewegung der Galvanometerscanner unterbrochen (siehe auch Seite 172). Auch die Rückübertragung der zusätzlichen Statusinformationen vom Scan-System erfolgt über ein User-Datenbit. Hierdurch werden weder die oben beschriebene "normale" Statusrückgabe, noch zugehörige Funktionen (wie **set_trigger**/**set_trigger4** oder die geschwindigkeitsabhängige Lasersteuerung) beeinträchtigt.

Nach jedem Neustart oder Reset sendet das Scan-System (ab ca. fünf Sekunden nach dem Neustart oder Reset) über alle Kanäle das XY2-100-Statuswort (dagegen keine Informationen bzw. 0 über das User-Datenbit).

Daten auslesen

Die von der RTC5 empfangenen Daten (mit Ausnahme des User-Datenbits) können jederzeit asynchron mit den Befehlen **get_value**, **get_values** bzw. **get_head_status** oder synchron mit dem **set_trigger**/**set_trigger4**-Befehl gelesen werden (siehe "Statusüberwachung und Diagnose", Seite 143 und die entsprechenden Befehlsreferenzen).

Dabei ist zu berücksichtigen, dass die ersten Daten nach einem Umschalten der Datenquelle aufgrund der seriellen Übertragungszeiten erst nach einer kurzen zeitlichen Verzögerung übertragen werden. Daher kann zwischen dem Umschaltzeitpunkt und dem Einlesen eine Wartezeit von ca. 50 µs entstehen.

Der Wertebereich rückübertragener Signale und die möglichen Statuszustände sind in der Befehlsreferenz des **control_command**-Befehls aufgelistet.

Hinweise

- Der Befehl **get_head_status** liest das XY2-100-Statuswort aus. Dieses wird bei einer Datenübertragung gemäß SL2-100-Protokoll immer und parallel zu anderen Statusinformationen vom Scan-System rückübertragen. Bei einer Ansteuerung über den XY2-100-Konverter dagegen wird das XY2-100-Statuswort nur rückübertragen, wenn auch dieser Datentyp zur Rückübertragung ausgewählt wurde. Andernfalls liefert **get_head_status** unbrauchbare Werte.

Die vom Scan-System über das User-Datenbit (verteilt auf mehrere Übertragungszyklen) rückübertragenen Statusinformationen können von der RTC5 (in Intervallen von ca. 2 ms) mit **read_user_data** ausgelesen werden. Der Wertebereich rückübertragener Signale und die möglichen Statuszustände sind in der Befehlsreferenz des **send_user_data**-Befehls bzw. Handbuch des jeweiligen Scan-Systems aufgelistet. Zwischen **send_user_data** und **read_user_data** sollte wegen der bitweisen seriellen Übertragung und einer möglichen Synchronisierungszeit eine Wartezeit von mindestens ca. 4 - 6 ms eingefügt werden, um sicher neue Daten vom angeforderten Datentyp zu erhalten.

8.1.3 Überwachung der Positionierung

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 (d.h. der Galvanometerscanner und/oder der Z-Achse des Scan-Systems) mit dem **control_command**-Befehl als vom Scan-System rückzuübertragender Datentyp ausgewählt werden. Die zurückgegebenen Ist-Positions-Werte können dann mit **get_values** ausgelesen oder mit **set_trigger**/**set_trigger4** aufgezeichnet werden(1).

Sollen die zurückgegebenen Ist-Positionen der Scan-Achsen mit den kartesischen Koordinatenwerten (x, y, z) verglichen werden, die als Parameter beim Aufruf der RTC5-Befehle angegeben werden, so muss zusätzlich berücksichtigt werden, dass diese Koordinatenwerte in der Regel – abhängig von den gewählten Einstellungen – verschiedene Korrekturen und Transformationen durchlaufen (z. B. eine Bildfeldkorrektur, eine Koordinatentransformation und/oder eine "Processing on the fly"-Korrektur), ehe sie als Ansteuerwerte an das Scan-System ausgegeben werden (siehe auch Seite 141). Damit ein Vergleich der zurückgegebenen Ist-Positions-Werte der Scan-Achsen mit den kartesischen Koordinatenwerten (x, y, z) auch tatsächlich durchgeführt werden kann, stellt der RTC5-Befehlssatz Befehle zur deren Rücktransformation bereit.

Die Rücktransformation kann aus Laufzeitgründen nicht auf der RTC5-Karte selbst geschehen, sondern muss nachträglich vom PC berechnet werden. Der **upload_transform**-Befehl überträgt dazu alle Korrektur- und Transformationseinstellungen, die dem Scan-System aktuell zugeordnet sind, von der RTC5-Karte auf den PC. Mit **transform** kann dann ein einzelnes xy-Wertepaar oder ein einzelner z-Wert zurücktransformiert werden. Mit **get_transform** (siehe auch **get_waveform**) kann eine ganze Serie von zuvor mit **set_trigger**/**set_trigger4** aufgezeichneten xy-Wertepaaren oder z-Werten zurücktransformiert werden.

Hinweise

- Wobbeltransformationen, "Processing on the fly"-Korrekturen und ein etwaiges Clipping bei der Vorwärtstransformation sind grundsätzlich nicht rücktransformierbar (siehe folgende Tabelle).

Vorwärts-transformation	**Rücktrans-formation möglich**
Wobbelbewegung	nein
"Processing on the fly"-Korrektur	nein
Koordinatentransformationen (Gesamtmatrix und Offset) für die x- und y-Koordinaten	ja
Offset auf die z-Koordinate	ja
Clipping auf die Grenzen des ansteuerbaren Bildfelds	nein
Bildfeldkorrektur	ja
Gain- und Offset-Korrektur der automatischen Selbstkalibrierung	ja
Clipping auf maximale Ansteuerwerte	nein

Die Rücktransformation kann auch nicht erfolgen, wenn bei der Vorwärtstransformation eine nicht-invertierbare Transformationsmatrix definiert wurde (Achtung: Clipping auf ±50 für jeden einzelnen Matrixkoeffizienten unter Berücksichtigung eines mit **set_scale** definierten Faktors; siehe Seite 183). Die Nicht-Invertierbarkeit wird bereits vom **upload_transform**-Befehl gemeldet.

Ist bei der Vorwärtstransformation ein Clipping auf die Grenzen des ansteuerbaren Bildfelds bzw. die maximalen Ansteuerwerte aufgetreten, so werden bei der Rücktransformation (idealerweise) die zu diesen Grenzwerten gehörenden kartesischen Koordinatenwerte statt der ursprünglichen Werte berechnet.

(1) Die aktuell zurückgegebenen Ist-Positionen hinken allerdings den aktuell ausgegebenen Ansteuersignalen aufgrund der Kommunikationslaufzeiten um einige Takte hinterher.

- Mit **get_values** können vier beliebige Signale gleichzeitig ausgelesen werden (etwa die Ist-Position von Galvanometerscanner 2 über StatusAX, die Ist-Position von Galvanometerscanner 1 über StatusAY, die Ist-Position der Z-Achse über StatusBX sowie ein beliebiges weiteres Signal, z.B. LaserOn). Der **get_value**-Befehl (ohne das 's' im Befehlsnamen) eignet sich dagegen nicht zur Überwachung der xy-Positionierung, weil mit diesem Befehl nur ein einzelnes Signal ausgelesen werden kann und mehrere Aufrufe zwangsweise zu xyz-Werten aus unterschiedlichen Zeiten führen.
- Mit **set_trigger** können über zwei Messkanäle zwei beliebige Signale gleichzeitig aufgezeichnet werden (mit **set_trigger4** vier Signale über vier Messkanäle).
- Die Zuordnung der über **get_values** oder **set_trigger** ausgelesenen Werte zu x, y bzw. z für die Rücktransformation kann sowohl bei **transform** als auch **get_transform** über den Parameter `Code` festgelegt werden.
- Sowohl bei **transform** als auch **get_transform** kann außerdem ausgewählt werden, welche Teiltransformationen durchgeführt werden sollen.
- Mit **transform** können mit **get_values** ausgelesene oder beliebige synthetische Werte (beispielsweise die Messpunkte eines gescannten Testmusters) tatsächlich zurücktransformiert werden. Bei der Rücktransformation von synthetischen Werten, die außerhalb des durch die Vorwärtstransformation möglichen Bildfeldes liegen (beispielsweise wenn der Vermessung des Testmusters nicht das SCANLAB-Referenzsystem zu Grunde liegt, siehe Seite 134), 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 Rücktransformation mit **transform** (nicht **get_transform**) auch Offline, d.h. ohne weiteren Zugriff auf eine RTC5-Karte ausgeführt werden.
- Wird mit dem **control_command**-Befehl der Positionsfehler an Stelle der Ist-Position als vom Scan-System rückzuübertragender Datentyp ausgewählt, so kann zwar kein direkter Vergleich zwischen dem ursprünglich definierten und dem gescannten Muster durchgeführt werden; es kann aber überprüft werden, ob das Scan-System die Ausgabewerte der RTC5-Karte korrekt abarbeitet. Dies ist insbesondere dann nützlich, wenn eine Rücktransformation von Ist-Positionen nicht (vollständig) möglich ist oder wenn nicht klar ist, ob eine Abweichung zwischen den rücktransformierten 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 Anpassen der dynamischen Einstellung (Tuning)

Entsprechend den unterschiedlichen Anforderungen verschiedener Anwendungen an die Dynamik der Positionierung des Laserstrahls kann die dynamische Einstellung (das Tuning) von Scan-Systemen von SCANLAB unterschiedlich optimiert werden, z. B. für das Ausführen von Vektoren oder Kreisbögen mit konstanter Prozessgeschwindigkeit (Vektortuning) oder das Ausführen von Sprüngen mit minimaler Dauer (Sprungtuning).

i*DRIVE*-Scan-Systeme können optional auch mit mehreren Tunings (d.h. mit mehreren Parametersätzen bzw. Regleralgorithmen) ausgestattet werden, so dass der Benutzer für unterschiedliche Anwendungen das jeweils geeignete Tuning auswählen kann. Die Auswahl des Tunings erfolgt über den **control_command**-Befehl. Bei 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 Seite 176).

Das Startverhalten ist standardmäßig so voreingestellt, dass das Scan-System nach jedem Neustart oder Reset mit dem ersten 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. Das Scan-System muss dazu mit einem Sprungtuning ausgestattet sein (siehe auch Abschnitt "Voraussetzung und Aktivierung", Seite 177).

Funktionsweise

In der Standardeinstellung (nach **load_program_file**) werden sowohl Sprungbefehle wie auch Markierbefehle 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 gegebenenfalls ein über **set_scanner_delays** festgelegtes Jump-Delay ausgeführt.

Nach Freigabe und Aktivierung des Sprungmodus mit den Befehlen **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 sogenannter harter Sprung über einen zeitbemessenen Sprung mit einer Zeitdauer von 10 µs ausgeführt. Die Zielposition wird also instantan und ohne Mikrovektorisierung angesteuert.
- Der Sprung wird mit einem Sprungtuning ausgeführt (das Scan-System muss mit einem solchen ausgestattet sein). Mit **set_jump_mode** kann das zu verwendende Sprungtuning angegeben werden. War vor dem Sprung ein anderes Tuning eingestellt, schaltet die RTC5 am Anfang des Sprungs automatisch auf das mit **set_jump_mode** angegebene um.
- Am Ende des 2D-Sprungs schaltet die RTC5 automatisch auf ein Vektortuning um (sofern das Scan-System mit einem solchen ausgestattet ist und sofern 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 "Sprunglängenabhängige Jump-Delays", Seite 178). 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**).
- Sofern bei einem 2D-Sprung nicht mindestens eine der beiden Achsen die mit **set_jump_mode** spezifizierte 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 bzw. ganz unterdrückt, so werden nachfolgende Sprünge wieder im Vektormodus (mikrovektorisiert) 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 dem **control_command**-Befehl 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.
- Die Software-Version des oder der angeschlossenen Scan-Systeme muss 2078 oder größer sein.
- 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 Tuning-Typ ist als Info in der Firmware des Scan-Systems 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.

Sprunglängenabhängige Jump-Delays

Bei der Ausführung eines harten Sprungs dauert es eine gewisse Zeit, bis der Scanner die vorgegebene Position eingenommen hat. Von der RTC5 wird diese Verzögerung (auch Sprungantwort genannt) dadurch berücksichtigt, dass am Ende des Sprungs ein Jump-Delay eingefügt wird, also die weitere Programmausführung entsprechend verzögert 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 RTC5 gespeichert werden. Bei aktiviertem Sprungmodus ermittelt die RTC5 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 von Benutzern 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` = 0) 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.

Experimentelle Ermittlung von Jump-Delay-Werten

Im Manual des Scan-Systems werden üblicherweise für das jeweilige Sprungtuning Sprungantwortzeiten für ausgewählte Sprunglängen spezifiziert.

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**-Befehlen (und dem gewünschten Sprungtuning) ausgeführt werden. Mit **control_command** sollte zuvor der vom Scan-System rückzuübertragende Datentyp auf Ist-Position umgestellt werden. Letztere kann dann per **set_trigger**/**set_trigger4** 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 bereitgestellt werden.

Hinweise zum Laden der ermittelten Jump-Delay-Werte

Der Befehl **load_jump_table_offset** lädt eine Tabelle mit Sprunglängen- und Jump-Delay-Werten aus einer ASCII-Textdatei. Die Textdatei kann eine oder mehrere Tabellen enthalten.

Jede Tabelle kann bis zu 50 Datenpunkte (*Length* | *Delay*(*Length*)) für verschiedene Sprunglängen enthalten. Aus diesen Daten bestimmt die RTC5 durch lineare Interpolation die vollständige (interne) Jump-Delay-Tabelle *Delay*(*Length*).

Für die Tabellen gelten die folgenden Regeln:

- Jede Tabelle muss mit der Anweisung (Caption)

  `[JumpTable<No>]`

  beginnen. Dabei steht `<No>` für die Tabellennummer (ein nicht-negativer Integer-Wert).
- Enthält die Tabelle mehrere `[JumpTable<No>]`-Einträge zur gleichen `<No>`, werden nur die Anweisungen nach dem ersten Eintrag verwendet, diejenigen nach weiteren Einträgen dagegen ignoriert. Es werden nur die Anweisungen bis zum nächsten '['-Zeichen verwendet (das nicht hinter einem Semikolon steht).
- Jeder Datenpunkt (*Length* | *Delay*(*Length*)) wird durch zwei Anweisungen dargestellt:

  ```
  Length<n> = <LengthValue>
  Delay<n> = <DelayValue>
  ```

  Dabei entspricht `<n>` dem Index des Datenpunktes (Integer-Wert, 1 ≤ `<n>` ≤ 50). Die Werte `<Value>` können als (vorzeichenlose) Gleitkommazahlen angegeben werden. Verwenden Sie Punkte (.) als Dezimal-Trennzeichen.
- 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 Sprunglängenwert *Length*, so wird der Datenpunkt mit dem größten Index `<n>` verwendet, die anderen werden ignoriert. Die Gleichheit wird auf ±0.01 geprüft.
- Für `<Value>` gelten die folgenden Wertebereiche:
  0.0 ≤ *Length* ≤ 1048576.0 und
  0.0 ≤ *Delay*(*Length*) ≤ 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 der Befehl **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 legale 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.

Bei der Initialisierung der RTC5 (mit **load_program_file**) wird die interne Jump-Delay-Tabelle wurzelförmig – mit einem Jump-Delay-Endwert von 1024 (d.h. 10,24 ms) für eine Sprunglänge von 2^{20} Bit – initialisiert.

Automatische Ermittlung der Jump-Delay-Tabelle

Mit **load_jump_table_offset** (Parameter `Name` = 0) kann eine automatische Ermittlung der Jump-Delay-Tabelle gestartet werden, sofern zuvor mit **set_jump_mode** der Sprungmodus erfolgreich freigegeben und aktiviert wurde (`Flag` = 1).

Für die automatische Ermittlung wird die automatische Lasersteuerung gegebenenfalls deaktiviert, der vom Scan-System rückzuübertragende Datentyp auf Ist-Position umgestellt und das Tuning auf das mit **set_jump_mode** definierte Sprungtuning gestellt (die ursprünglichen Einstellungen werden gegebenenfalls am Befehlsende 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**/**set_trigger4** aufgezeichnet und per **get_waveform** ausgelesen. Die Daten werden daraufhin analysiert, zu welchem Zeitpunkt letztmals die spezifizierte 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 Listenspeicher geschrieben und dann abgearbeitet. Der Parameter `ListPos` gibt an, an welcher Stelle im 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 und wird bei den nachfolgenden (kürzeren) Sprunglängen durch das jeweils zuvor ermittelte Jump-Delay bestimmt. `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 vier 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 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 65535 geclippt).
  Durch Angabe eines Offsets können messtechnisch bedingte Laufzeiten (z.B. durch einen XY2-100-Konverter, durch Tuning-Umschaltung oder Laufzeiten des rückübertragenen Signals) bei der Ermittlung der Jump-Delay-Tabelle herausgerechnet werden oder eine gewisse Sicherheitsspanne zum Ausgleich eventueller statistischer Fluktuationen (Rauschen) hinzugefügt werden.
  Der **load_jump_table**-Befehl ist identisch zu **load_jump_table_offset** mit `Offset` = 0.
- Bei einer simultanen 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 Anpassen des PosAcknowledge-Schwellwerts

Der **control_command**-Befehl kann auch dazu verwendet werden, den PosAcknowledge-Schwellwert zu verändern. Das Startverhalten ist standardmäßig so voreingestellt, dass das Scan-System nach jedem Neustart oder Reset einen Schwellwert von 0,28% des vollen Positionsbereichs einstellt. Wenn ein anderer Schwellwert gewünscht wird, dann muss er – für jede Achse separat – angepasst werden.

8.1.7 Anpassen der effektiven Kalibrierung

Mit dem **control_command**-Befehl kann die Regelelektronik des Scan-Systems auch so eingestellt werden, dass sie die von der RTC5 empfangenen Positionswerte um einen bestimmten Faktor (herunter-)skaliert. Der Skalierungsfaktor kann auf die Werte 1, 1/2, 1/4 oder 1/8 eingestellt werden. Die vom Scan-System (optional) an die RTC5 zurückgesandten Positionssignale, sowie die von SCANLAB voreingestellte Kalibrierung des Scan-Systems bleiben davon unberührt. 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 oder Reset mit dem Skalierungsfaktor 1, also mit der von SCANLAB voreingestellten Kalibrierung startet.

Mit dem **control_command**-Befehl ist es auch möglich, die von SCANLAB voreingestellte Kalibrierung und den aktuell eingestellten Skalierungsfaktor auszulesen.

8.1.8 Anpassen des Startverhaltens

In der Standard-Ausführung werden i*DRIVE*-Scan-Systeme von SCANLAB so eingestellt, dass

- das erste Tuning (Nummer 0) ausgewählt ist (siehe auch Abschnitt "Anpassen der dynamischen Einstellung (Tuning)", Seite 176),
- ein PosAcknowledge-Schwellwert von 0,28% des vollen Positionsbereichs (d.h. 0,28% von 2^{16} Bits) eingestellt ist (siehe auch Abschnitt "Anpassen des PosAcknowledge-Schwellwerts", Seite 181),
- ein Skalierungsfaktor 1 eingestellt ist (siehe auch Abschnitt "Anpassen der effektiven Kalibrierung", Seite 181).

Die obigen Einstellungen können mit dem **control_command**-Befehl geändert werden. Die geänderten Einstellungen sind zunächst temporär, können aber auch nachträglich mit `control_command(…, …, 0A00`$_H$`)` als Starteinstellungen für nachfolgende Neustarts oder Resets (Aus- und Einschalten der Spannung) gespeichert werden.

Solange das Startverhalten nicht wie beschrieben geändert wird, startet das Scan-System bei jedem Neustart oder Reset mit den von SCANLAB festgelegten Voreinstellungen.

Das Rückmeldeverhalten des Scan-Systems kann nur temporär geändert werden. Das zugehörige Startverhalten ist dagegen von SCANLAB fest vorgegeben: nach einem Neustart überträgt das Scan-System das XY2-100-Statuswort (siehe auch Abschnitt "Anpassen des Rückmeldeverhaltens", Seite 173).

8.1.9 Fehlerdiagnose und Funktionstest

Zur Diagnose von i*DRIVE*-Scan-Systemen im Fehlerfall können u.a. die vielfältigen Rückmeldefunktionen des Scan-Systems verwendet werden. So kann beispielsweise

- der aktuelle Betriebszustand,
- der Betriebszustand zum Zeitpunkt der zuletzt stattgefundenen Betriebsunterbrechung und
- ein Ereignis-Code, der darüber Auskunft gibt, welches Ereignis für den Wechsel in einen Fehlzustand verantwortlich war

ausgelesen werden.

Damit überprüft werden kann, ob die Datenübertragung zwischen RTC5 und Scan-System funktioniert, kann mit dem **control_command**-Befehl – für jede Achse separat – ein 8-Bit-Wert an das Scan-System übertragen werden. Im Folgenden wird dann am entsprechenden Statuskanal ein 20-Bit-Wert zurückgesendet: Bei fehlerfreier Datenübertragung sind die oberen 8 Bit des rückgesendeten 20-Bit-Wertes identisch mit dem hingesandten 8-Bit-Wert und die nächstniedrigeren 8 Bit identisch mit dem Komplementärwert des hingesandten 8-Bit-Wertes.

Dieser 20-Bit-Wert wird so lange zurückgegeben, bis mit dem **control_command**-Befehl wieder ein anderer Datentyp zur Rückübertragung ausgewählt wird (siehe Abschnitt "Anpassen des Rückmeldeverhaltens", Seite 173).

Um sicherzustellen, dass nach einem Übertragungstest das gleiche Rückmeldeverhalten wie vor dem Test wiederhergestellt werden kann, erlaubt es der **control_command**-Befehl, den aktuell zur Rückübertragung ausgewählten Datentyp zur späteren Wiederauswahl zwischenzuspeichern.

8.2 Koordinatentransformationen

Damit das Scan-System relativ zum Bildfeld exakt ausgerichtet werden kann (bzw. – wenn die Option "Zweite-Scan-Kopf-Steuerung" freigeschaltet ist – *zwei* Scan-Köpfe relativ zum *gemeinsamen* Bildfeld ausgerichtet werden können), kann für alle durch Vektor- oder Bogenbefehle definierten x- und y-Ausgabekoordinaten (x|y) (separat für den primären und sekundären Scan-Kopf-Anschluss) eine lineare Koordinatentransformation festgelegt werden:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$

Die (2 x 2)-Gesamtmatrix M wird dabei von der RTC5 automatisch als Produkt einer Skalierungsmatrix M_S, einer Drehmatrix M_R und einer allgemeinen Transformationsmatrix M_T berechnet:

$$M = M_T \bullet M_R \bullet M_S$$

Die Koeffizienten der drei Matrizen (M_T, M_R und M_S) und die Offsetwerte ($x_0|y_0$) können für den primären und sekundären Scan-Kopf-Anschluss 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:

$$z' = z + z_o$$

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 (mathematische 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: Der Befehl
`set_matrix( 1, 0.5, -0.866, 0.866, 0.5, 1 )`
definiert für den primären Scan-Kopf-Anschluss eine sofortige Rotation um 60° im Gegenuhrzeigersinn um den Mittelpunkt des Bildfeldes. Dasselbe kann auch mit dem Befehl `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.
- Per Initialisierung (**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 RTC5 berechnete Gesamtmatrix M ist unabhängig davon, in welcher Reihenfolge der Offset und 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 geschehen ausschließlich um den Mittelpunkt des Bildfeldes, Spiegelungen relativ zu den Achsen. Passende Offsetverschiebungen müssen von Benutzern berechnet und zur Verfügung gestellt werden.
- 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). Zur Ansammlung über Listenbefehle müssen die Befehle nicht notwendigerweise unmittelbar aufeinander folgen. Wann eine (angesammelte) Transformation wirksam wird, entscheidet immer der zuletzt gesendete `at_once`-Parameterwert.
- Vor Anwendung der (evtl. angesammelten) Gesamttransformation auf die aktuelle Position werden die “Laser active”-Lasersteuersignale für `at_once` = 0 bis 2 ausgeschaltet; für `at_once` = 3 dagegen bleiben die Lasersteuersignale unverändert.

- Mit `at_once` = 1 oder 3 werden alle bis dahin (sowohl über Kontrollbefehle als auch über Listenbefehle) nur angesammelten Einstellungen sofort und gleichzeitig abgearbeitet. Dabei werden – noch bevor ein nachfolgender Befehl ausgeführt wird – die Achsen des Scan-Systems von der aktuellen Position (zur Vermeidung harter Sprünge) mit der (vor-) eingestellten Sprunggeschwindigkeit in die korrigierte Position bewegt. Dies kann daher eine gewisse Anzahl von Taktzyklen benötigen. Etwaige Scanner-Delays werden nicht initiiert. Für die Dauer des Sprungs auf die korrigierte Position ist der INTERNAL-BUSY-Status gesetzt.
- Mit `at_once` = 0 wird zwar eine über einen Kontrollbefehl (aber nicht über einen Listenbefehl) definierte Einstellung (zusammen mit anderen bis dahin angesammelten Einstellungen) auch bereits bei der Ausführung des unmittelbar nächsten Listenbefehls automatisch wirksam (also quasi sofort, wenn gerade eine Liste abgearbeitet wird, `at_once` = 0 wird in diesem Fall exakt wie `at_once` = 1 behandelt).

  Ansonsten bleibt aber jede mit `at_once` = 0 definierte Einstellung (zusammen mit bis dahin angesammelten Einstellungen) so lange unwirksam – auch bei nachfolgenden **goto_xy**- und **goto_xyz**-Befehlen – solange sie nicht durch eine nachfolgende Koordinatentransformation (z. B. mit einem Listenbefehl mit `at_once` = 1 oder 3 oder einem entsprechenden Kontrollbefehl) aktiviert wird.
- Mit `at_once` = 2 werden die (über Kontroll- oder Listenbefehle angesammelten) Einstellungen erst bei der Ausführung des nächsten **jump_abs**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls wirksam (jedoch nicht bei anderen Sprungbefehlen wie **jump_abs_3d** oder **jump_rel_3d**). Die Korrektur der aktuellen Ausgabeposition wird dann – anders als bei `at_once` = 0, 1 oder 3 – nicht separat, sondern gemeinsam mit dem angegebenen Koordinatensprung ausgeführt. Dadurch werden überflüssige Galvanometer-scanner-Bewegungen (inkl. eventueller Delays) und damit Laufzeit eingespart.

  Beispiel: Die folgende Befehlsliste 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);
  ```
- Ist dem jeweiligen Scan-Kopf-Anschluss keine Korrekturtabelle zugewiesen, so werden die neuen Einstellungen für die Koordinatentransformationen (auch für `at_once` = 1 oder 3) lediglich auf der RTC5 gespeichert und erst dann wirksam, wenn eine Korrekturtabelle zugewiesen wird.
- Falls die Option “Zweite-Scan-Kopf-Steuerung” nicht freigeschaltet ist, bleiben Koordinatentransformationen, die für den sekundären Scan-Kopf-Anschluss spezifiziert wurden, wirkungslos.
- Koordinatentransformationen werden auf alle auszugebenden Koordinaten aus allen Vektorbefehlen (Jump- oder Mark-Listenbefehle, aber auch **goto_xy**- oder **goto_xyz**-Befehle) und Bogenbefehlen angewandt. Die Transformation erfolgt erst *nach* der Mikrovektorisierung und *nach* einer Einbindung evtl. vorgenommener Wobbel- oder “Processing on the fly”-Einstellungen, aber *vor* einer Bildfeldkorrektur. Die Matrixtransformation wird dabei *vor* der Offsetverschiebung ausgeführt.

- Die Koordinatentransformation ermöglicht z. B. eine skalierbare Beschriftung. Bei einer Skalierung sollte allerdings berücksichtigt werden, dass sich dadurch die effektive Sprung- oder Markiergeschwindigkeit ändert (bei unterschiedlichen Skalierungsfaktoren k_X und k_Y sogar richtungsabhängig). Unter Umständen müssen daher die Sprung- und Markiergeschwindigkeiten entsprechend angepasst werden.
- Bei 3D-Vektorbefehlen wirkt die Transformationsmatrix in der Regel nur auf die x- und y-Komponenten, nur der Offset wirkt gegebenenfalls auf alle drei Komponenten.
- Um auch bei eingestellten Koordinatentransformationen (wie Verdrehungen, Verkleinerungen oder Verschiebungen) das volle reale Bildfeld auszunutzen, kann der erweiterte Wertebereich des virtuellen 24-Bit-Bildfelds verwendet werden (siehe Seite 135).
- Für zweidimensionale "Processing on the fly"-Anwendungen können zusätzlich Koordinatentransformationen für das komplette virtuelle Bildfeld definiert werden. Diese wirken auch auf die bereits mikrovektorisierten Koordinaten, aber im Gegensatz zu den oben beschriebenen kopfspezifischen Koordinatentransformationen noch *vor* der "Processing on the fly"-Korrektur. Siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208.

Hinweise für RTC4 Benutzer

- Beim Laden einer Korrekturtabelle mit **load_correction_file** können bei der RTC5 *keine* Transformationen der Korrekturtabelle mehr eingestellt werden.
- Koordinatentransformationen können bei der RTC5 für den primären und sekundären Scan-Kopf-Anschluss separat definiert werden und erfüllen denselben Zweck wie diejenigen bei **load_correction_file** bei der RTC4.
- Koordinatentransformationen werden von der RTC5 demzufolge erst *nach* der Mikrovektorisierung, aber unmittelbar *vor* der Bildfeldkorrektur ausgeführt (mit entsprechenden Konsequenzen für die effektive Sprung- oder Markiergeschwindigkeit, s. o.).
- Die bei der RTC4 noch möglichen (für beide Scan-Köpfe gemeinsamen) Koordinatentransformationen vor der Mikrovektorisierung werden mit der RTC5 nicht mehr unterstützt.

8.3 Online-Positionierung

Im vorangehenden Kapitel 8.2 "Koordinatentransformationen", Seite 183 wird ausführlich beschrieben, wie ein Scan-System relativ zum Bildfeld exakt ausgerichtet werden kann. 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 RTC5) zwischen der Positionierung des Werkstücks und der Ausrichtung des Scan-Systems ist allerdings mit den in Kapitel 8.2 "Koordinatentransformationen", Seite 183 beschriebenen (Kontroll-)Befehlen nicht leicht möglich. Für Anwendungen, bei denen solch ein Timing wichtig ist, stellt der RTC5-Befehlssatz auch Befehle zur sogenannten Online-Positionierung zur Verfügung. Hierbei können die Daten für eine Offset- und/oder Rotations-Koordinatentransformation oder einer allgemeinen Matrixoperation (z.B. direkt von der Positionsanalyse des Werkstücks) über die McBSP/SPI-Schnittstelle eingegeben werden.

Das Einlesen der Daten zur Online-Positionierung über die McBSP/SPI-Schnittstelle muss mit den Befehlen **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** oder **set_mcbsp_matrix** (bzw. mit den entsprechenden Listenbefehlen) aktiviert und wie gewünscht konfiguriert werden (s.u.). 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** bzw. den entsprechenden Listenbefehlen) definiert werden. Wie bei den in Kapitel 8.2 "Koordinatentransformationen", Seite 183 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** bzw. **jump_rel** oder auch irgendein anderer Listenbefehl), von der Eingabe eines externen Steuersignals abhängig gemacht werden (zur bedingten Befehlsausführung siehe Seite 244).

Zur Beschreibung der McBSP/SPI-Schnittstelle siehe Kapitel 4.4.6 "SPI / I2C-Stiftleiste", Seite 54.

Konfigurieren der Online-Positionierung

Mit den Befehlen **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** oder **set_mcbsp_matrix** (bzw. mit den entsprechenden Listenbefehlen) wird festgelegt, wie die an der McBSP/SPI-Schnittstelle eingegebenen Werte von der RTC5 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 RTC5 als Offset in X- 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 vier Optionen aktiviert wird, werden alle Werte, die nachfolgend an der McBSP/SPI-Schnittstelle eingegeben werden, an die internen Speicherstellen 1 oder gegebenenfalls 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, y *oder* rot**
  Wird nur *eine* der drei Optionen aktiviert, so können X- oder Y-Offset-Korrekturwerte als 32-Bit-Werte *mit* Vorzeichen bzw. Rotations-Korrekturwerte als 32 Bit-Werte *ohne* Vorzeichen eingegeben werden.
  Die McBSP/SPI-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-Werte 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/SPI-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/SPI-Eingabewerte werden auch alternierend an die internen Speicherstellen 1 und 2 übertragen.
  Die Identifizierung des Datentyps erfolgt über das Codierungsbit #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, 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/SPI-Eingabewerte werden ebenfalls alternierend an die internen Speicherstellen 1 und 2 übertragen.
  Die Identifizierung des Datentyps erfolgt über das Codierungsbit #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 Kodierungsbit #31 unterschieden. Die Reihenfolge der Übertragung ist dadurch zwar unerheblich; es sollten aber stets 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-Kodierungen vorliegen.
- **matrix**
  Wird diese Option aktiviert, können anschließend Matrixkoeffizienten als 32 Bit-Werte mit Vorzeichen übertragen werden. Die Indices werden dem Datenwort kodiert mitgegeben (siehe Befehlsbeschreibung). Die McBSP/SPI-Eingabewerte werden an die interne Speicherstelle 1 übertragen.

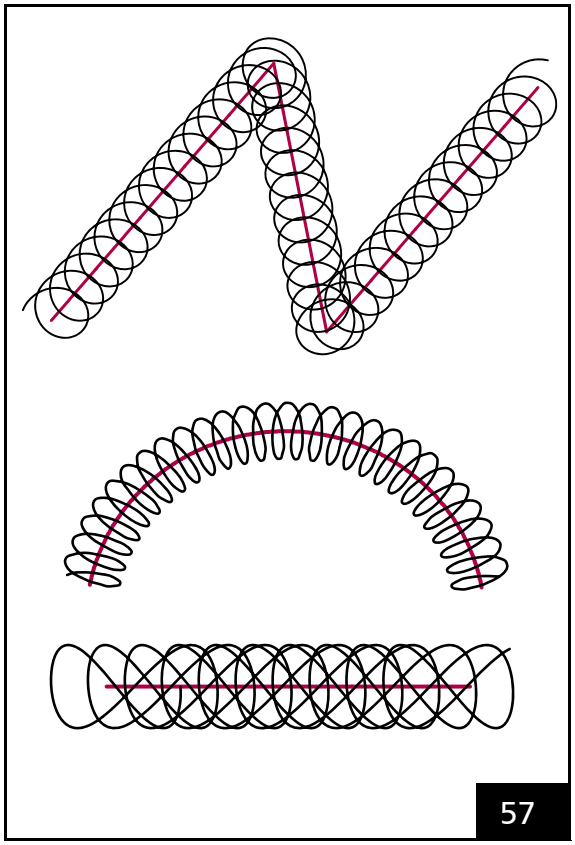
Hinweise

- Eine Online-Positionierung kann in Kombination mit einer über Encoder-Signale gesteuerten "Processing on the fly"-Anwendung genutzt werden, dagegen *nicht* in Kombination mit einer über McBSP/SPI-Signale gesteuerten "Processing on the fly"-Anwendung:
  – Durch die Befehle zur Konfigurierung der 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/SPI-Eingabewerte an die internen Speicherstellen 1 oder gegebenenfalls 2 übertragen (s.o.) und stehen dann für eine Online-Positionierung, aber nicht mehr für eine "Processing on the fly"-Anwendung 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 Online-Positionierung deaktiviert. Nachfolgend werden McBSP/SPI-Eingabewerte gegebenenfalls zusätzlich oder stattdessen an die interne Speicherstelle 0 und/oder 3 übertragen und stehen dann für die "Processing on the fly"-Anwendung, aber nicht mehr für eine Online-Positionierung zur Verfügung.
  – Wird die Online-Positionierung mit **set_mcbsp_x**, **set_mcbsp_y** oder **set_mcbsp_rot** abgeschaltet, werden die Daten zwar weiterhin auf die internen Speicherstellen 1 bzw. 1 und 2 übertragen (sofern weitere Daten gesendet werden), jedoch nicht mehr verwendet (**apply_mcbsp** bleibt wirkungslos).

8.4 Wobbelmode

Der Wobbelmode ermöglicht eine Variation der Linienstrichbreite für eine Lasermarkierung.

Dazu wird die gewöhnliche, lineare Bewegung der Ausgabeposition mit einer ellipsenförmigen Bewegung kombiniert. Der Laserstrahl wird also spiralförmig durch das Bildfeld geführt, siehe Abb. 57. Alternativ kann auch die Kombination mit einer Achterbewegung (mit einer in Bewegungsrichtung stehenden oder liegenden Acht) aktiviert 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.

Ab Version DLL 543, OUT 543, RBF 524 kann die momentane Wobbelamplitude (aus **set_wobbel** oder **set_wobbel_mode**) mit dem Triggersignal 53 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**). Das Format der Daten ist ((Tansversal << 16) + Longitudinal).

Bei einer "Processing on the fly"-Korrektur über die McBSP/SPI-Schnittstelle oder die Encoder-Schnittstelle, bei der der Scan-Kopf lediglich die Differenz der externen Ist-Bewegung zur gewünschten Gesamt-Bewegung auskorrigiert, siehe Kapitel 8.7.2 "Kompensation linearer Bewegungen", Seite 201, 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. Dieser Jitter kann vermieden werden, indem der Wobbelfigur mit dem Listenbefehl **set_wobbel_direction** anstelle der momentanen Bewegungsrichtung eine konstante Bewegungsrichtung aufgeprägt wird.

Für optimale Markierungsergebnisse muss die Wobbelfrequenz auf die Markiergeschwindigkeit abgestimmt sein. Unter Umständen muss auch die Markiergeschwindigkeit neu eingestellt werden.

Nach **set_wobbel** oder **set_wobbel_mode** wird der Wobbel-Startpunkt immer auf denselben Wert relativ zum Vektor/Bogen-Startpunkt bzw. dessen Richtung gesetzt. Die Wobbelphase wird dann sowohl innerhalb eines ununterbrochenen Polygonzugs (inkl. Bögen) als auch nach Unterbrechungen (z. B. durch einen Jump-Befehl) kontinuierlich fortgesetzt, bis **set_wobbel** oder **set_wobbel_mode** erneut aufgerufen wird.

Der Wobbelmode kann nicht kombiniert werden mit:

- Sky-Writing
- Pixelausgabemodus
- Jumps
- **laser_on_list**

Weitere Details finden Sie unter **set_wobbel** und **set_wobbel_mode**.

Für viele Schweißanwendungen führt die vorgegebene Auswahl an klassischen Wobbelfiguren (Kreis, Ellipse, Sinus, Acht) nicht zu einem optimalen (gleichmäßigen) Ergebnis im Bereich der Schweißnaht. Beispielsweise erfolgen bei Ellipsen die Bewegungen parallel zur Translationsbewegung mit erhöhter Geschwindigkeit, und Bewegungen in entgegengesetzter Richtung mit erniedrigter

Geschwindigkeit. Die Folge ist ein ungleichmäßiger Energieeintrag. Mit **set_wobbel_vector** kann das Anwendungsprogramm eine individuell geformte Wobbelfigur definieren, bestehend aus maximal 512 stückweise linearen Abschnitten, und gleichzeitig auch eine Variation der Laserleistung entlang der Figur festlegen (siehe auch **set_wobbel_control**). Jedoch kann **set_wobbel_vector** nicht zusammen mit der Softstart-Funktion verwendet werden, und ist auch nicht mit der automatischen oder der vektorkontrollierten Lasersteuerung kombinierbar.

8.4.1 Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte

"Klassische" Wobbelfiguren

"Klassische" Wobbelfiguren werden mit den Befehlen **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.

Vorsicht!

- 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 evtl. 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 i*DRIVE*-Systemen 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`-Datei zum Korrekturfile 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
  ≈ 47.500 Bit.
  - Im Bildfeld entspricht das einer Wobbelamplitude von A/K ≈ 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 ≈ 17,9 m/s. Beachten Sie, dass dieser Wert nur eine grobe Schätzung und darüber hinaus über das (nicht-lineare) Korrekturfile 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" (Befehl **set_wobbel_vector**) auf die Dynamik des Scan-Kopfs. Wie bei den "klassischen" Wobbelfiguren (s.o.) darf die maximale Positioniergeschwindigkeit nicht wesentlich überschritten werden, d.h. jeder einzelne Mikroschritt der "Frei definierbaren Wobbelfigur" darf eine Dauer von 10 μs mal der maximalen Positioniergeschwindigkeit nicht wesentlich überschreiten. Anderenfalls droht unter Umständen eine Überhitzung der Galvanometerscanner.

8.5 Verwendung mehrerer verschiedener Korrekturtabellen

8.5.1 Konfiguration mit zwei 2D-Scan-Systemen

Der Speicher der RTC5 kann vier verschiedene Korrekturtabellen gleichzeitig enthalten. Bei der Ansteuerung von zwei 2D-Scan-Systemen über *eine* RTC5-Karte kann daher an beiden Scan-Systemen eine separate Bildfeldkorrektur durchgeführt werden.

Durch den Befehl **select_cor_table** oder **select_cor_table_list** werden die zwei benötigten Korrekturtabellen den jeweiligen Scan-Kopf-Steuerungs-Ports (siehe auch Seite 43) zugewiesen.

Wie Sie die beiden Korrekturtabellen in dieser Konfiguration verwenden können, beschreiben die folgenden Schritte:

- Laden Sie die benötigten 2D-Korrekturdateien jeweils mit `load_correction_file(Name, n, 2)` (`n` = 1…4).
  Details finden Sie unter dem Befehl **load_correction_file**.
- Weisen Sie danach mit dem Befehl `select_cor_table(n1, n2)` jeweils eine der geladenen 2D-Korrekturtabellen dem ersten und dem zweiten Scan-System zu (`n1` und `n2` = 1…4).
- Spezifizieren Sie mit **set_matrix**, **set_offset**, **set_scale**, **set_angle** oder den entsprechenden Listenbefehlen Skalierungsfaktor, Rotation und Offset, um die beiden Bildfelder exakt aufeinander auszurichten.

Hinweise

- Die Standardeinstellung für **select_cor_table** bzw. **select_cor_table_list** ist `(1,0)`, d.h. Korrekturtabelle Nr.1 wird für den ersten Scan-Kopf-Anschluss verwendet und die Ausgangssignale am zweiten Scan-Kopf-Anschluss sind abgeschaltet.
- Die RTC5 kehrt nach jedem Reset (durch den Befehl **load_program_file**) zur Standardeinstellung zurück. Wenn eine andere Einstellung verwendet werden soll, muss nach einem Reset der Befehl **select_cor_table** bzw. **select_cor_table_list** erneut aufgerufen werden. Die Korrekturtabellen Nummer 3 und 4 werden durch **load_program_file** gelöscht.
- Den Scan-Kopf-Steuerungs-Ports können *nicht* gleichzeitig zwei 3D-Korrekturtabellen und auch nicht gleichzeitig eine 2D- *und* eine 3D-Korrekturtabelle zugewiesen werden.

8.5.2 Mehrere Korrekturtabellen mit einem Scan-System

Auch bei der Verwendung von nur einem Scan-System kann es nützlich sein, mit mehreren verschiedenen Korrekturtabellen zu arbeiten, etwa wenn ein Positionierlaser und ein Hauptlaser mit unterschiedlichen Wellenlängen eingesetzt werden. Spezielle Anwendungen benötigen u.U. auch Korrekturdateien in schneller Abfolge. Da während der Abarbeitung einer Liste keine Korrekturdateien geladen werden können, würde ein Download zwischen den Bearbeitungen einiges an nicht produktiver Zeit in Anspruch nehmen. In den Speicher einer RTC5 mit freigeschalteter 3D-Option können auch bis zu vier 3D-Korrekturdateien oder gemischt 2D-Korrekturdateien und 3D-Korrekturdateien geladen werden.

- Laden Sie die Korrekturdateien mit **load_correction_file**.
- Wechseln Sie dann mit den Befehlen `select_cor_table(n, 0)` (`n` = 1…4) von einer Korrekturtabelle zur anderen.

8.6 Ansteuerung eines 3-Achsen-Scan-Systems (optional)

Wenn die 3D-Option der RTC5 freigeschaltet ist, kann mit der RTC5 auch ein 3-Achsen-Scan-System, bestehend aus einem XY-Scan-Kopf und einer dynamischen Fokussiereinheit (vario*SCAN*) als Z-Achse, angesteuert werden.

8.6.1 Verwendung

In einem 3-Achsen-Scan-System lenken die Spiegel des XY-Scan-Kopfs den Laserstrahl zur gewünschten XY-Position und stellt die Z-Achse simultan eine geeignete Fokuslänge ein, um den Laserfokus entlang der optischen Achse in die gewünschte Z-Position zu positionieren.

Mit 3-Achsen-Scan-Systemen kann der Laserfokus innerhalb eines ebenen Arbeitsfeldes positioniert werden, ohne dass dafür ein Flachfeld-Objektiv gebraucht wird. Sie kommen daher häufig in Anwendungen zum Einsatz, für die Standard-Flachfeld-Objektive nicht verfügbar sind.

Andererseits können 3-Achsen-Scan-Systeme auch als 3D-Strahlablenksysteme genutzt werden. Hier kann der Laserfokus entlang der Kontur des zu bearbeitenden Werkstücks geführt werden. Dies ermöglicht die Bearbeitung von Werkstücken in drei Dimensionen.

Zur Erweiterung von XY-Scan-Köpfen zu 3-Achsen-Scan-Systemen sind bei SCANLAB vario*SCAN*, vario$SCAN_{de}$, vario*SCAN* FC und vario*SCAN FLEX* dynamische Fokussiereinheiten erhältlich.

8.6.2 Anschluss und Initialisierung

Damit mit einer RTC5 ein 3-Achsen-Scan-System betrieben werden kann,

- muss die 3D-Option der RTC5 freigeschaltet sein (für eine installierte RTC5 kann dies mit dem Befehl **get_rtc_version** überprüft werden),
- muss der XY-Scan-Kopf am primären Scan-Kopf-Anschluss und die Z-Achse am sekundären Scan-Kopf-Anschluss angeschlossen sein(1) und
- muss mit **select_cor_table** oder **select_cor_table_list** dem primären Scan-Kopf-Anschluss eine 3D-Korrekturtabelle (`D3_*.CT5`) zugewiesen werden(1).

Ansonsten werden für den Betrieb des 3-Achsen-Scan-Systems weder zusätzliche Treiber noch weitere Software-Dateien (also auch keine andere DLL und keine andere DSP-Programmdatei) benötigt. Mit Ausnahme der 3D-Korrekturtabelle (s. o.) erfolgen auch Initialisierung und Programmstart unverändert (siehe Seite 66).

Die Standard-DLL (`RTC5DLL.dll`) unterstützt alle Befehle für den Betrieb eines 3-Achsen-Scan-Systems. Die volle Funktionalität der Befehle (Ausgabe der z-Koordinate) steht allerdings nur bei freigeschalteter 3D-Option zur Verfügung.

Mehrere 3-Achsen-Systeme können (mit Multi-Board-Befehlen) simultan von einem PC aus angesteuert werden, wenn entsprechend viele RTC5-Karten mit freigeschalteter 3D-Option in den PC eingebaut sind.

(1) Wenn zusätzlich die Option "Zweite-Scan-Kopf-Steuerung" freigeschaltet ist, kann stattdessen auch dem *sekundären* Scan-Kopf-Anschluss eine 3D-Korrekturtabelle zugewiesen werden und der XY-Scan-Kopf an den *sekundären* Scan-Kopf-Anschluss und die Z-Achse an den *primären* Scan-Kopf-Anschluss angeschlossen werden (siehe auch "Schnittstellen zum Scan-System", Seite 42 und "2D- und 3D-Korrekturdateien", Seite 137).

8.6.3 3D-Markierbefehle

Im Befehlssatz der RTC5 sind die folgenden 3D-Vektorbefehle und zeitbemessenen 3D-Vektorbefehle enthalten:

- **goto_xyz**
- **jump_abs_3d**, **timed_jump_abs_3d**
- **jump_rel_3d**, **timed_jump_rel_3d**
- **mark_abs_3d**, **timed_mark_abs_3d**
- **mark_rel_3d**, **timed_mark_rel_3d**

Zusätzlich stehen die folgenden 3D-Bogenbefehle zur Verfügung, mit denen spiralförmige Markierungen ausgeführt werden können:

- **arc_abs_3d**, **arc_rel_3d**

Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktionieren die Befehle gleich wie die entsprechenden 2D-Befehle:

- Die angegebenen Vektoren oder Bögen werden mikrovektorisiert (mit derselben Ausgabeperiode wie bei 2D-Befehlen, siehe Seite 107).
- Sprung- und Markiergeschwindigkeit können wie bei 2D-Befehlen mit **set_jump_speed** und **set_mark_speed** angegeben werden (siehe Seite 103) bzw. werden bei zeitbemessenen Vektorbefehlen 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-Bildfeldkorrektur durchgeführt (siehe Seite 136).
- Bei Jump- und Mark-Befehlen werden, unter Berücksichtigung der eingestellten Delays, die Lasersteuersignale ein- und ausgeschaltet (siehe Seite 111).

Die RTC5 berechnet simultan zu den Ausgabewerten für die Galvanometerscanner des Scan-Systems die zugehörigen Fokuslängen (bzw. Schnittweiten) und Ausgabewerte für die Z-Achse. Die RTC5 verwendet hierzu die entsprechenden Daten aus der zugewiesenen 3D-Korrekturtabelle. Jede Korrekturtabelle wird von SCANLAB individuell entsprechend der optischen Konfiguration des jeweiligen 3-Achsen-Systems aus allen verfügbaren optischen Systemdaten (wie der Spiegelgeometrie, der Kalibrierung, Objektivdaten und Optik-Daten der Z-Achse) berechnet. Die Korrekturtabellen sind im mitgelieferten Software-Paket in einer Korrekturdatei (`*.CT5`) enthalten (siehe auch Befehl **load_correction_file**).

Im RTC5-Modus müssen die x- und y-Koordinaten eines Vektors als 20-Bit-Werte mit Vorzeichen (d.h. als Zahlen zwischen −524.288 und +524.287) an die RTC5 übergeben werden, die z-Koordinaten dagegen als 16-Bit-Werte mit Vorzeichen (d.h. als Zahlen zwischen −32.768 und +32.767). Der Kalibrierungsfaktor (das Verhältnis zwischen den digitalen Punktkoordinaten in Bits und der tatsächlichen Position des Punktes in Millimetern, siehe Seite 134) ist daher in z-Richtung um den Faktor 16 kleiner als in x- und y-Richtung:

$$K_z = K_{xy} / 16.$$

Der Kalibrierungsfaktor K_{xy} kann mit **get_table_para** aus der verwendeten Korrekturtabelle ausgelesen werden.

Im RTC4-Kompatibilitätsmodus sollte dagegen (wie bei der RTC4): $K_x = K_y = K_z$ ($= K_{xy} / 16$) verwendet werden.

Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bzgl. der Sprung- oder Markiergeschwindigkeit gleich behandelt werden.

Hinweise

- Nach Aktivierung der vektorkontrollierten Lasersteuerung mit **set_vector_control** können auch die Para-Mark- und Para-Jump-Befehle (**para_jump_abs_3d**, **timed_para_jump_abs_3d**, **para_jump_rel_3d**, **timed_para_jump_rel_3d**, **para_mark_abs_3d**, **timed_para_mark_abs_3d**, **para_mark_rel_3d**, **timed_para_mark_rel_3d**) genutzt werden. Diese Befehle variieren simultan einen Lasersignalparameter linear entlang des Mark- oder Jump-Vektors (siehe Seite 166). Für (`Ctrl` = 7) wird beispielsweise die Fokusverschiebung, d.h. der Offset auf die berechnete Fokuslänge (wie mit **set_defocus** oder **set_defocus_list**) variiert.
- Die Größe des nutzbaren Bildfelds und der Z-Hub (Höhe des nutzbaren Arbeitsvolumens) kann der “Readme.txt” Datei, die der 3D-Korrekturdatei beigelegt ist, oder dem Handbuch des jeweiligen 3-Achsen-Scan-Systems oder vario*SCAN*s (Kapitel “Technische Spezifikationen”) entnommen werden.

- Soll eine Z-Achse dazu benutzt werden, den Laserfokus immer nur in einer bestimmten Ebene zu halten, so können auch 2D-Vektorbefehle verwendet werden (nachdem die gewünschte z-Position mit einem 3D-Vektorbefehl eingestellt wurde). 2D-Vektorbefehle lassen die z-Position unverändert, regeln aber – bei freigeschalteter 3D-Option und nach Zuweisung einer 3D-Korrekturtabelle – die Fokuslänge so nach, dass der Laserstrahl immer in die gleiche Ebene fokussiert wird.
- Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist, wirken 3D-Markierbefehle wie 2D-Markierbefehle. Die Mikrovektorisierung wird aber (außer bei den 3D-Bogenbefehlen **arc_abs_3d** und **arc_rel_3d**) wie bei einem 3D-Befehl berechnet, die effektive Sprung- oder Markiergeschwindigkeit in der XY-Ebene wird dadurch beeinflusst.
- Für die z-Koordinate kann mit **set_offset_xyz** oder **set_offset_xyz_list** ein Offset z_0 definiert werden. Zusätzlich kann mit **set_defocus** oder **set_defocus_list** ein Offset auf die berechnete Fokuslänge festgelegt werden, der sich dann auch (in zu z_0 entgegengesetzter Richtung) auf den z-Ausgabewert auswirkt.
- Während eines Markierungsprozesses (während der Abarbeitung eines Mark- oder Arc-Befehls) wird die Fokuslänge des Scan-Systems kontinuierlich nachgestellt, damit der Laserfokus immer auf die aktuelle Bildebene scharf gestellt ist. Während der Ausführung eines Jump-Befehls (oder **goto_xyz**) ist dagegen eine solche Nachregelung nicht nötig. Über den Parameter `DirectMove3D` der Befehle **set_delay_mode** und **set_delay_mode_list** kann daher alternativ eine Sprung-Methode für alle 3D-Jump-Befehle ausgewählt werden, bei der der z-Ausgabewert während des Sprungs direkt (linear) zu seinem Endwert geführt wird. Auf diese Weise lassen sich unnötige Bewegungen der Z-Achse vermeiden.
- Bei 3D-Markierbefehlen wirken sich die folgenden Einstellungen (sofern vorgenommen) *direkt* nur auf die x- und y-Ausgabewerte, dagegen nicht oder höchstens *indirekt* auf den z-Ausgabewert aus (siehe auch Seite 141):
  - eine mit **set_wobbel** oder **set_wobbel_mode** aktivierte Wobbelbewegung (siehe Seite 189)
  - eine mit **set_fly_x**, **set_fly_y**, **set_fly_rot** oder **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos** aktivierte "Processing on the fly"-Korrektur (siehe Seite 199).
  - eine mit **set_matrix**, **set_offset**, **set_offset_xyz**, **set_scale**, **set_angle** (oder den entsprechenden Listenbefehlen) eingestellte Koordinatentransformation für die x-und y-Koordinaten (siehe Seite 183).
  - eine mit **auto_cal** oder **set_hi** aktivierte Gain- und Offset-Korrektur (siehe Seite 224).
- Im Befehlssatz der RTC5 sind zusätzlich auch 3D-Mikrovektorbefehle enthalten (siehe Seite 221).

8.6.4 Justage

3D-Korrekturtabellen werden von SCANLAB so berechnet, dass die Ebene der mittleren Fokuslage (ohne einen mit **set_defocus**, **set_defocus_list**, **set_offset_xyz** oder **set_offset_xyz_list** eingestellten Offset) mit z = 0 angesteuert wird. Insbesondere wenn das 3-Achsen-Scan-System (mit zugehöriger 3D-Korrekturtabelle) dafür vorgesehen ist, den Laserfokus immer in einer bestimmten Ebene zu halten, muss daher der mechanische Abstand zwischen Scan-System und Arbeitsebene so eingestellt werden, dass die Arbeitsebene mit der z = 0 Ebene zusammenfällt. Wird ein anderer mechanischer Abstand eingestellt, so besteht die Gefahr, dass der Laserfokus zumindest in den Randbereichen des Arbeitsfeldes bzw. Arbeitsvolumens nicht scharf gestellt wird.

Beim vario*SCAN* und vario*SCAN*$_{de}$ muss zusätzlich ein bestimmter Abstand zum Scan-System eingehalten werden. Welche mechanischen Abstände einzustellen sind, kann dem Handbuch des jeweiligen 3-Achsen-Scan-Systems oder vario*SCAN*s entnommen werden.

Der für die Berechnung einer Korrekturtabelle verwendete Arbeitsabstand kann in der Regel auch mit **get_table_para** aus der verwendeten Korrekturtabelle ausgelesen werden.

Sind der Arbeitsabstand und gegebenenfalls der Abstand zum Scan-System korrekt eingestellt, sollte eine Feinjustierung der Fokuslage vorgenommen werden. Dazu wird ein Testmuster in der Mitte der z = 0 Arbeitsebene gescannt. Die für das Bearbeitungsergebnis optimale Fokuslage erreicht man

- beim vario*SCAN* und vario*SCAN*$_{de}$ durch manuelles Verdrehen des Fokussierrings,
- beim vario*SCAN FLEX* durch Einstellen der Fokussieroptik-Position mit dem Befehl **move_to**,
- beim vario*SCAN* FC (also auch beim intelli*WELD* FC) durch Anpassen des Koeffizienten *A* der verwendeten 3D-Korrekturtabelle(1).

Überprüfen der Z-Achsen-Kalibrierung

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 daher auch nicht in die Berechnung der 3D-Korrekturtabellen ein.

Unter Umständen ist daher die berechnete Korrekturtabelle noch nicht bestmöglich für das Scan-System optimiert. Um dies zu überprüfen, führen Sie einen Lasermarkierungstest aus, der das ganze Bildfeld überdeckt. Überprüfen Sie, ob der Laserfokus den Anforderungen Ihrer Anwendung entspricht. Stellen Sie eine wesentliche Variation des Spotdurchmessers fest, so ist unter Umständen eine Neukalibrierung der Z-Achsen-Korrektur notwendig.

Der Benutzer kann dazu die Koeffizienten *A*, *B* und *C* der parabolischen Funktion ($z_{out} = A + Bl + Cl^2$) anpassen, die in 3D-Korrekturtabellen den Zusammenhang zwischen dem z-Ausgabewert z_{out} und dem Fokuslängenwert l festlegt. Der (einheitenlose) Fokuslängenwert l entspricht dabei für jeden Punkt `(x|y|z)` im Arbeitsvolumen dem Fokuslängenunterschied zwischen dem angegebenen Punkt `(x|y|z)` und dem Punkt `(0|0|0)`.

Ziel der Z-Achsen-Kalibrierung ist die Bestimmung geeigneter Koeffizienten *A*, *B* und *C*. Diese können mit dem Befehl **load_z_table** auf die RTC5 übertragen werden.

Die ABC-Koeffizientenwerte einer angegebenen Korrekturdatei auf dem PC können direkt mit **read_abc_from_file** ausgelesen und mit **write_abc_to_file** hineingeschrieben werden.

(1) Mit **get_head_para** können die Koeffizienten *A*, *B* und *C* 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.

Vorgehen

(1) Stellen Sie den korrekten mechanischen Abstand zwischen Scan-System und z = 0 Arbeitsebene sowie gegebenenfalls den korrekten Abstand zwischen vario*SCAN* und Scan-System ein(1).

(2) Laden Sie mit **load_correction_file** die gewünschte 3D-Korrekturtabelle und weisen Sie sie mit **select_cor_table** zu.

(3) Lesen Sie mit **get_head_para** die zugewiesenen Koeffizienten *A*, *B* und *C* aus.

Beim vario*SCAN* FC (also auch beim intelli*WELD* FC) müssen die Schritte **(4)** bis **(7)** nicht ausgeführt werden. Machen Sie hier direkt bei Schritt **(8)** weiter.

(4) Richten Sie den Laser-Spot mit dem Befehl **goto_xyz** auf den Referenzpunkt(2).

(5) Platzieren Sie am Referenzpunkt ein Testobjekt.

(6) Setzen Sie die Z-Achse mit **load_z_table**(0, 0, 0) in Neutralstellung (Mittelstellung).

(7) Justieren Sie nun den Laserfokus
- beim vario*SCAN* und vario*SCAN*$_{de}$ durch manuelles Verdrehen des Fokussierrings,
- beim vario*SCAN FLEX* durch Einstellen der Fokussieroptik-Position mit dem Befehl **move_to**.

(8) Richten Sie mit **goto_xyz** den Laserfokus auf einen beliebigen Punkt (x|y|z) im benötigten Arbeitsvolumen(3).

(9) Lesen Sie mit **get_z_distance** den von der RTC5 für diesen Punkt eingestellten Fokuslängenwert *l* (in Bits) aus(4).

(10) Optimieren Sie dann an dieser Stelle den Laserfokus, indem Sie mit **load_z_table**(A=z_{out}, 0, 0) den z-Ausgabewert z_{out} so variieren, dass der Laserfokus die gewünschte Qualität erreicht(5).

(11) Wiederholen sie die Schritte **(8)** bis **(10)** 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 Arbeitsvolumen ab.

(12) Fitten Sie die gewonnen Wertepaare (l|z_{out}) an die Funktion $z_{out} = A + Bl + Cl^2$.

(13) Verwenden Sie die resultierenden Koeffizienten *A*, *B* und *C*, um mit **load_z_table**(*A*, *B*, *C*) die 3D-Korrekturtabelle anzupassen.

Mit **load_z_table** geladene Werte werden durch einen nachfolgenden **load_correction_file**-Befehl überschrieben (**load_correction_file** setzt die drei Koeffizienten *A*, *B* und *C* auf die in der geladenen Korrekturtabelle voreingestellten Werte(7)). Nach jedem **load_correction_file** sollte daher auch **load_z_table**(*A*, *B*, *C*) erneut aufgerufen werden.

Test für 3-Achsen-Scan-Systeme mit F-Theta-Objektiv

- Markieren Sie mit der angepassten 3D-Korrekturtabelle unter Verwendung von 3D-Vektorbefehlen (siehe Seite 194) zwei große identische Quadrate, davon eines mit dem Werkstück in der z-Position z = z_{min} und 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.

(1) Siehe Handbuch des jeweiligen 3-Achsen-Scan-Systems oder vario*SCAN*s.

(2) Die Koordinatenwerte des Referenzpunkts (in mm) sind in der der 3D-Korrekturdatei beigelegten "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). 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).

(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 Arbeitsvolumen ist im jeweiligen Handbuch spezifiziert.

(4) Der Fokuslängenwert *l* kann positiv oder negativ sein.

(5) Der passende z_{out} Ausgabewert kann positiv oder negativ sein. Ein Startwert für die Suche nach dem optimalen Laserfokus kann jeweils gemäß $A + Bl + Cl^2$ aus dem zuvor ausgelesenen Fokuslängenwert *l* und den zuvor ausgelesenen bzw. verwendeten Werten *A*, *B* und *C* berechnet werden.

(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.

(7) Jede von SCANLAB berechnete 3D-Korrekturtabelle enthält Werte für die Koeffizienten A, B und C (siehe auch zugehörige "Readme.txt"-Datei oder **get_table_para**).

8.6.5 Erweiterte 3D-Korrektur

Bei 3D-Scan-Systemen, insbesondere bei denen ohne Objektiv oder mit einem nicht-telezentrischen F-Theta-Objektiv, hängt die Bildfeldgröße vom Abstand zum Scan-Kopf-Objektiv ab (d.h. von der Z-Koordinate). Typischerweise wird die Bildfeldgröße linear mit dem Z-Wert gestreckt oder gestaucht. SCANLAB 3D-Korrekturfiles enthalten daher sogenannte Stretch-Faktoren (siehe Seite 140), 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. 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
```

Bei einem iterativen Vorgehen zur optimalen Einstellung der Korrekturwerte kann die Messung beliebig oft ausgeführt werden. Die jeweils neu ermittelten Korrekturwerte können einfach zu den vorherigen Werten zu einer Gesamtkorrektur addiert werden.

Mit dem Befehl **load_stretch_table** kann dann die erweiterte 3D-Korrektur aus einer ASCII-lesbaren Textdatei auf die RTC5-Karte geladen werden. Die 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** per Default nicht aktiv. Sie ist aktiv, sobald eine gültige Tabelle auf die RTC5-Karte geladen ist und sie kann durch **load_stretch_table** durch Angabe eines Null-Zeigers anstelle des Dateinamens auch wieder deaktiviert werden.

Für die Textdatei gelten folgende Regeln:

- Die Datei kann eine oder mehrere Tabellen enthalten, auch solche anderen Typs.
- Die 2D-Stretchkorrekturtabelle muss mit der Caption `[StretchTable<No>]` eingeleitet werden, wobei `<No>` der beim Befehl **load_stretch_table** anzugebenden Nummer entspricht.
- 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, die anderen werden ignoriert.
- Leere oder unvollständige Zeilen sind ungültig und werden ignoriert.

8.7 “Processing on the fly” (optional)

8.7.1 Verwendung und Initialisierung

Wenn die “Processing on the fly”-Option der RTC5 freigeschaltet ist, können mit der RTC5 auch bewegte Werkstücke (z. B. auf einem Förderband, auf einer rotierenden Scheibe oder auf einem XY-Verschiebetisch), 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 bzw. des Scan-Systems indirekt (über Encoder-Zähler) oder direkt (über die McBSP/SPI-Schnittstelle) an die RTC5 übergeben werden. Bei aktivierter “Processing on the fly”-Korrektur werden dann die Koordinatenwerte aller Vektor- und Bogenbefehle entsprechend der übergebenen Positionswerte transformiert.

Bei Übergabe der Positionswerte über Encoder-Zähler wird die Bewegung über kundeneigene Inkremental-Encoder registriert oder simuliert. Durch die registrierten oder simulierten Encoder-Signale werden RTC5-interne Zähler (Encoder-Zähler) getriggert, dessen Zählwerte dann der aktuellen Position entsprechen und als Basis für die “Processing on the fly”-Korrektur verwendet werden (siehe “Synchronisation durch Encoder-Signale”, Seite 246). Bei Übergabe über die McBSP/SPI-Schnittstelle wird der Eingabewert an der McBSP/SPI-Schnittstelle direkt als Basis für die “Processing on the fly”-Korrektur verwendet (siehe “Synchronisation und Online-Positionierung mit McBSP/SPI-Signalen”, Seite 248).(1)

Die Befehle zur Aktivierung und Deaktivierung der “Processing on the fly”-Korrektur werden bei freigeschalteter “Processing on the fly”-Option von der Standard-DLL (`RTC5DLL.dll`) unterstützt. Es werden weder zusätzliche Treiber noch weitere Software-Dateien (also auch keine andere DLL und keine andere DSP-Programmdatei) benötigt. Auch Initialisierung und Programmstart erfolgen unverändert (siehe Seite 66). Die Aktivierung und Deaktivierung der “Processing on the fly”-Korrektur erfolgt über Listenbefehle. Die bei der Aktivierung einzustellenden Parameter müssen gegebenenfalls zuvor durch einen Kalibrierungsprozess ermittelt werden (s. u.).

Eine aktivierte “Processing on the fly”-Korrektur wirkt sich *direkt* nur auf den x- und y-Ausgabewert, dagegen (bei freigeschalteter 3D-Option) auf den z-Ausgabewert höchstens *indirekt* aus (siehe auch Seite 141). Sind die Option “Zweite-Scan-Kopf-Steuerung” freigeschaltet und beiden Scan-Kopf-Anschlüssen eine 2D-Korrekturtabelle zugewiesen, dann wirkt sich eine “Processing on the fly”-Korrektur an beiden Scan-Kopf-Anschlüssen gleich aus.

Ob die “Processing on the fly”-Option freigeschaltet ist, kann für eine installierte RTC5 mit dem Befehl **get_rtc_version** überprüft werden.

Ab Version DLL 530, OUT 531 kann auch eine “Processing on the fly”-Korrektur für die Z-Achse aktiviert werden (siehe Kapitel 8.7.11 ““Processing on the fly”-Korrektur für die Z-Achse — “FlyZ-Korrektur” (ab Version DLL 530, OUT 531)”, Seite 215).

Hinweise

- Eine aktivierte “Processing on the fly”-Korrektur wirkt sich nicht nur auf Vektor- und Bogenbefehle aus. Auch während der Abarbeitung von normalen Listenbefehlen, die eigentlich keine Scannerbewegung ausführen (z. B. **list_nop**, **set_control_mode_list**), werden (ab Version OUT 515) die Positionen der X- und Y-Achsen entsprechend der aktuellen “Processing on the fly”-Eingangssignale (Position des Werkstücks oder Scan-Systems) kontinuierlich nachgeführt. Bei älteren Versionen erfolgte gegebenenfalls nach Ausführung solcher Befehle ein harter Sprung auf die geänderte Positionsangabe.

(1) Die gleichzeitige Verwendung beider Übergabemöglichkeiten zur “Processing on the fly”-Korrektur zweier unabhängiger Bewegungen ist nicht möglich (siehe Abschnitt “Übersicht”, Seite 200). Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden (siehe Seite 188).

Übersicht

Es sind die folgenden encoderbasierten "Processing on the fly"-Korrekturen von Werkstück-Bewegungen möglich:

- **set_fly_x**, **set_fly_y**: Kompensation linearer Werkstück-Bewegungen (z. B. mit Förderband oder XY-Tisch) – siehe Kapitel 8.7.2, Seite 201.
- **set_fly_rot**: Kompensation von Rotationsbewegungen des Werkstücks (z. B. mit Drehteller) – siehe Kapitel 8.7.3, Seite 204.
- **set_fly_2d**: Kompensation zweidimensionaler Werkstück-Bewegungen (z. B. XY-Tisch) – siehe Kapitel 8.7.4, Seite 206.

Außerdem sind die folgenden McBSP/SPI-basierten "Processing on the fly"-Korrekturen von Scan-System-Bewegungen möglich:

- **set_fly_x_pos**, **set_fly_y_pos**: Kompensation von linearen oder zweidimensionalen Bewegungen des Scan-Systems (z. B. Roboterarm) – siehe Kapitel 8.7.2, Seite 201.
- **set_fly_rot_pos**: Kompensation von Rotationsbewegungen des Scan-Systems (z. B. Roboterarm) – siehe Kapitel 8.7.3, Seite 204.
- **set_mcbsp_in**, **set_mcbsp_in_list**: Kompensation von linearen oder zweidimensionalen Bewegungen oder Rotationsbewegungen des Scan-Systems (z. B. Roboterarm) – siehe Kapitel 8.7.2, Seite 201 und Kapitel 8.7.3, Seite 204.

Folgende "Processing on the fly"-Korrekturen können miteinander kombiniert werden:

- **set_fly_x** kann mit **set_fly_y** kombiniert werden.
- **set_fly_x_pos** kann mit **set_fly_y_pos** kombiniert werden.

Weitere Kombinationen sind *nicht* möglich (außer denjenigen aus Kapitel 8.7.11). Die encoderbasierte **set_fly_x**-Korrektur einer linearen Werkstück-Bewegung kann also beispielsweise weder mit einer **set_fly_rot**-Korrektur von Werkstück-Rotationsbewegungen noch mit einer McBSP/SPI-basierten "Processing on the fly"-Korrektur von Scan-System-Bewegungen kombiniert werden.
Die jeweils letzte Anweisung bestimmt die gesamte Korrektur. **set_fly_rot** beispielsweise deaktiviert jede zuvor aktivierte "Processing on the fly"-Korrektur und aktiviert die Kompensation von Rotationsbewegungen des Werkstücks. Ausnahmen hiervon sind nur die oben genannten Kombinationen (**set_fly_y** beispielsweise deaktiviert **set_fly_x** *nicht*).

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 8.7.7, Seite 211).

Encoderbasierte "Processing on the fly"-Korrekturen sind außerdem mit der Schleppverzugskompensation kombinierbar (siehe Kapitel 8.7.10, Seite 215).

8.7.2 Kompensation linearer Bewegungen

Die “Processing on the fly”-Korrektur linearer Bewegungen kann mit den Listenbefehlen **set_fly_x** und/oder **set_fly_y** oder alternativ mit den Listenbefehlen **set_fly_x_pos** und/oder **set_fly_y_pos** oder aber mit **set_mcbsp_in** oder **set_mcbsp_in_list** (`Mode` = 1…3) aktiviert werden(1). 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.

Mit **fly_return** kann die “Processing on the fly”-Korrektur (für beide Raumrichtungen gleichzeitig) deaktiviert werden (siehe hierzu auch die Hinweise auf Seite 209).

Korrektur über Encoder-Zähler

Erfolgt die Übergabe der Positionswerte für die “Processing on the fly”-Korrektur über die internen Encoder-Zähler, so muss die “Processing on the fly”-Korrektur mit **set_fly_x** und/oder **set_fly_y** aktiviert werden. 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 Encoder-Zählers definiert (s. u.).

Durch **set_fly_x** bzw. **set_fly_y** wird der jeweilige Encoder-Zähler (Encoder0 bzw. Encoder1) auf Null zurückgesetzt(2).

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(3).

(1) Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar (siehe Abschnitt “Übersicht”, Seite 200).

(2) Mit **set_control_mode**, Bit#9 kann zuvor eingestellt werden, ob der Zähler sofort oder erst mit dem nachfolgenden Start-Trigger zurückgesetzt wird (d.h. mit einem nachfolgenden externen Startsignal, **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl, gegebenenfalls verzögert um eine mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** konfigurierte Streckenverzögerung, siehe auch **set_control_mode**, Bit#2). Wird der Zähler erst mit dem nachfolgenden Trigger zurückgesetzt, so wird damit effektiv der 10 µs-Jitter (zufälliger Zeitversatz zwischen Startsignal und Listenstart) ausgeschaltet.

Hinweise

- Die Kombination aus **set_fly_x** und **set_fly_y** kann sowohl für eindimensionale “Processing on the fly”-Anwendungen (z. B. bei einem schräg verlaufenden Förderband) als auch für encoderbasierte zweidimensionale “Processing on the fly”-Anwendungen (z. B. für einen XY-Verschiebetisch) verwendet werden, vornehmlich für separate oder separierbare Markierungen im realen Bildfeld. Für zusammenhängende Markierungen im virtuellen Bildfeld empfiehlt SCANLAB allerdings stattdessen die Verwendung von **set_fly_2d** (siehe Kapitel 8.7.4 “Kompensation zweidimensionaler Bewegungen”, Seite 206 und Kapitel 8.7.11 ““Processing on the fly”-Korrektur für die Z-Achse — “FlyZ-Korrektur” (ab Version DLL 530, OUT 531)”, Seite 215).

Bestimmung des Skalierungsfaktors

Falls die Werkstückposition mit einem (oder zwei) Inkremental-Encoder(n) registriert wird, muss zur Ermittlung des Skalierungsfaktors eine Kalibrierung durchgeführt werden:

Zuerst muss für beide Raumrichtungen mit **get_encoder** das Encoder-Inkrement (i_x bzw. i_y in [Counts/mm]) bestimmt werden:

- Lesen Sie (mit **get_encoder**) den Zähler-Startwert aus und starten Sie dann die Bewegung.
- Stoppen Sie die Bewegung und lesen Sie (mit **get_encoder**) den Zähler-Endwert aus(4).
- Messen Sie die zurückgelegte Distanz in mm.
- Die Encoder-Inkremente i berechnen sich dann wie folgt:

  i = (Zähler-Endwert – Zähler-Startwert) / zurückgelegte Distanz

Aus den ermittelten Encoder-Inkrementen können dann die Skalierungsfaktoren $\mathtt{Scale}_x$ und $\mathtt{Scale}_y$ [in Bits/Count] wie folgt berechnet werden:

$$\mathtt{Scale}_x = K / i_x$$
$$\mathtt{Scale}_y = K / i_y$$

Dabei ist K der Kalibrierungsfaktor [in Bits/mm] (siehe Seite 134).

(3) Die Encoder-Zähler sind 32-Bit-Zähler (für 32-Bit-Werte mit Vorzeichen). Wird der maximale (minimale) Zählerwert erreicht, wird beim minimalen (maximalen) Zählerwert weitergezählt.

(4) Alternativ können Zähler-Start- und Endwert mit dem Listenbefehl **store_encoder** in einen Zwischenspeicher der RTC5 gespeichert und mit **read_encoder** ausgelesen werden.

Falls das Werkstück sich mit konstanter Geschwindigkeit bewegt (v_x bzw. v_y in mm/s) und mit **simulate_encoder** eine Encoder-Simulation aktiviert wurde, dann können die Skalierungsfaktoren wie folgt berechnet werden:

$\texttt{Scale}_x = K \times v_x / (1.000.000\ \text{Counts/s})$

$\texttt{Scale}_y = K \times v_y / (1.000.000\ \text{Counts/s})$

Die Vorzeichen der Skalierungsfaktoren müssen gegebenenfalls entsprechend der Bewegungsrichtung angepasst werden.

Korrektur über McBSP/SPI-Schnittstelle

Erfolgt die Übergabe der Positionswerte für die "Processing on the fly"-Korrektur über die McBSP/SPI-Schnittstelle, so muss die "Processing on the fly"-Korrektur mit **set_fly_x_pos** und/oder **set_fly_y_pos** aktiviert werden. 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/SPI-Schnittstelle [in Bits] (s.u.).

Der Ausgabewert wird aus der aktuellen Ausgabeposition berechnet, indem (für alle Raumrichtungen) das Produkt aus Skalierungsfaktor und aktuellem Eingabewert der McBSP/SPI-Schnittstelle aufaddiert wird. Diese Korrektur erfolgt alle 10 µs.

Für eindimensionale Korrekturen stehen an der McBSP/SPI-Schnittstelle 32 Bit-Werte (mit Vorzeichen) zur Verfügung, für zweidimensionale dagegen nur jeweils 16 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/SPI-Schnittstelle. Zur Beschreibung der Schnittstelle siehe Seite 54.

Bestimmung des Skalierungsfaktors

Falls die Position des Werkstücks oder Scan-Systems über die McBSP/SPI-Schnittstelle übergeben wird, muss zur Ermittlung des Skalierungsfaktors eine Kalibrierung durchgeführt werden:

Zuerst muss für beide Raumrichtungen mit **read_mcbsp** das Positions-Inkrement (i_x bzw. i_y in [Bits/mm]) bestimmt werden:

- Lesen Sie (mit **read_mcbsp**) den Start-Eingabewert aus und starten Sie dann die Bewegung.
- Stoppen Sie die Bewegung und lesen Sie (mit **read_mcbsp**) den End-Eingabewert aus.
- Messen Sie die zurückgelegte Distanz in mm.
- Die Positions-Inkremente i berechnen sich dann wie folgt:

  i = (End-Eingabewert – Start-Eingabewert) / zurückgelegte Distanz

Aus den ermittelten Positions-Inkrementen können dann die Skalierungsfaktoren $\texttt{Scale}_x$ und $\texttt{Scale}_y$ [in Bits/Bits] wie folgt berechnet werden:

$\texttt{Scale}_x = K / i_x$

$\texttt{Scale}_y = K / i_y$

Dabei ist K der Kalibrierungsfaktor [in Bits/mm] (siehe Seite 134).

Hinweise

- Nach **set_fly_x_pos** und **set_fly_y_pos** werden Eingabewerte, die an der McBSP/SPI-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 einem Reset mit **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/SPI-Eingabewerte dagegen gebenenfalls auch an die internen Speicherstellen 1, 2 und/oder 3 übertragen (siehe Seite 203). Bei aktivierter Online-Positionierung werden sie an die interne Speicherstelle 1 oder gegebenenfalls 2 (dagegen nicht an die Speicherstellen 0 oder 3) übertragen (siehe Seite 187).
- 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 bis 3 geschrieben, von wo aus sie mit **read_mcbsp** auch (unsortiert) ausgelesen werden können.

Korrektur über McBSP/SPI-Schnittstelle mit zusätzlicher McBSP/SPI-Eingabe

Eine “Processing on the fly”-Korrektur für lineare Bewegungen mit McBSP/SPI-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/SPI-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. Alle McBSP/SPI-Eingabewerte müssen dazu über Bit#31 kodiert 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 stets alternierend an die internen Speicherstellen 1 und 2 übertragen und anschließend gemäß der Bit#31-Kodierung auch an die Speicherstellen 0 bzw. 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.
- Mit **set_mcbsp_in** oder **set_mcbsp_in_list** kann (wie bei **set_fly_x_pos** und **set_fly_y_pos**) ein Skalierungsfaktor spezifiziert werden. Für zweidimensionale Korrekturen 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 eindimensionaler Korrekturwerte stehen effektiv 31 Bit mit Vorzeichen zur Verfügung (Bit#31 ist als Kodierungsbit reserviert), für die Übertragung zweidimensionaler Korrekturwerte 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/SPI-Schnittstelle. Zur Beschreibung der Schnittstelle siehe Seite 54.

Korrektur über McBSP/SPI-Schnittstelle mit erweiterter McBSP/SPI-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- und Y-Richtung auch eine Korrektur in Z-Richtung sowie eine Anpassung der Laserleistung möglich ist. Außerdem können bis zu vier weitere Signale zur freien Verwendung übertragen werden.

Die asynchron in den McBSP/SPI-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 auch mit **read_multi_mcbsp** ausgelesen werden.

8.7.3 Kompensation von Rotationsbewegungen

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 mit **set_fly_rot** oder alternativ mit **set_fly_rot_pos** oder aber mit **set_mcbsp_in** oder **set_mcbsp_in_list** (`Mode` = 4) aktiviert werden.

Mit **fly_return** kann die "Processing on the fly"-Korrektur deaktiviert werden (siehe hierzu auch die Hinweise auf Seite 209).

Hinweise

- Eine "Processing on the fly"-Rotationskorrektur kann *nicht* mit anderen "Processing on the fly"-Korrekturen kombiniert werden (siehe Abschnitt "Übersicht", Seite 200).
- Eine "Processing on the fly"-Rotationskorrektur simultan für beide Scan-Kopf-Anschlüsse ist nur sinnvoll, wenn die zwei angeschlossenen Scan-Systeme exakt auf dasselbe Rotationszentrum justiert sind.

Korrektur über Encoder-Zähler

Falls die Winkelposition des Werkstücks mit einem Inkremental-Encoder registriert wird, muss der Encoder an den ENCODER X-Eingang angeschlossen werden (siehe auch Seite 247).

Die "Processing on the fly"-Korrektur muss dann mit **set_fly_rot** aktiviert werden[1]. Dabei muss der Parameter `Resolution` [in Counts/Umdrehung] angegeben werden. Das Inverse dieses Parameters entspricht der Rotation [Zahl der Umdrehungen] der aktuellen Ausgabeposition je Zählerschritt (s.u.).

Durch **set_fly_rot** wird der Encoder-Zähler Encoder0 auf Null zurückgesetzt.

Danach wird der Ausgabewert über eine lineare Abbildung aus der aktuellen Ausgabeposition berechnet (Drehung um das angegebene Rotationszentrum mit Drehwinkel / 360° = aktueller Encoder0-Zählerstand / `Resolution`). Diese Korrektur erfolgt alle 10 µs[2].

Bestimmung des Parameters `Resolution`

Falls die Winkelposition des Werkstücks mit einem Inkremental-Encoder registriert wird, muss zur Ermittlung des Parameters `Resolution` eine Kalibrierung durchgeführt werden:

- Lesen Sie (mit **get_encoder**) den Zähler-Startwert aus und starten Sie dann die Drehbewegung (eine beliebige Anzahl komplett ausgeführter Umdrehungen).
- Zählen Sie die Zahl der Umdrehungen.
- Stoppen Sie die Bewegung und lesen Sie (mit **get_encoder**) den Zähler-Endwert aus[3].
- Der Parameter `Resolution` [in Counts/Umdrehung] berechnet sich dann wie folgt:

  `Resolution` = (Zähler-Endwert – Zähler-Startwert) / Zahl der Umdrehungen

Beachten Sie dabei, dass die Anzahl der Zählschritte je Umdrehung gleich der Anzahl der Encoder-Signalflanken, also gleich dem Vierfachen der Anzahl der Encoder-Inkremente je Umdrehung ist (siehe Seite 247).

Falls das Werkstück sich mit konstanter Rotationsgeschwindigkeit ω [in Umdrehungen/s] bewegt und mit **simulate_encoder** eine Encoder-Simulation aktiviert wurde, dann kann der Parameter `Resolution` wie folgt berechnet werden:

`Resolution` = (1.000.000 Counts/s) / ω

Das Vorzeichen von `Resolution` muss gegebenenfalls entsprechend der Drehrichtung angepasst werden.

(1) Siehe Fußnote [2] Seite 201.

(2) Die Encoder-Zähler sind 32-Bit-Zähler (für 32-Bit-Werte mit Vorzeichen). Wird der maximale (minimale) Zählerwert erreicht, wird beim minimalen (maximalen) Zählerwert weitergezählt.

(3) Alternativ können Zähler-Start- und Endwert mit dem Listenbefehl **store_encoder** in einen Zwischenspeicher der RTC5 gespeichert und mit **read_encoder** ausgelesen werden.

Korrektur über McBSP/SPI-Schnittstelle

Erfolgt die Übergabe der Winkelpositionswerte für die “Processing on the fly”-Korrektur von Rotationsbewegungen über die McBSP/SPI-Schnittstelle, so muss die “Processing on the fly”-Korrektur mit **set_fly_rot_pos** aktiviert werden. Der Parameter `Resolution`, der hierbei angegeben werden muss, hat die gleiche Bedeutung wie bei **set_fly_rot** und wird auch gleich ermittelt (s. o.) mit dem Unterschied, dass McBSP/SPI-Eingangswerte mit **read_mcbsp** und Zählerwerte mit **get_encoder** ausgelesen werden.

Hinweise

- Wie bei der Kompensation linearer Bewegungen mit McBSP/SPI-Eingabewerten werden McBSP/SPI-Eingabewerte an die interne Speicherstelle 0 übertragen (siehe Hinweise auf Seite 202).

Korrektur über McBSP/SPI-Schnittstelle mit zusätzlicher McBSP/SPI-Eingabe

Eine “Processing on the fly”-Korrektur für Rotationsbewegungen mit McBSP/SPI-Eingabewerten kann (alternativ zu **set_fly_rot_pos**) auch mit **set_mcbsp_in** oder **set_mcbsp_in_list** (`Mode` = 4) aktiviert werden.

Der Vorteil dieser Befehle liegt darin, dass die McBSP/SPI-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. Alle McBSP/SPI-Eingabewerte müssen dazu über Bit#31 kodiert werden.

Hinweise

- Alle Eingabewerte werden stets alternierend an die internen Speicherstellen 1 und 2 übertragen und anschließend gemäß der Bit#31-Kodierung auch an die Speicherstellen 0 bzw. 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** kann (wie bei **set_fly_rot_pos**) eine Rotationsauflösung spezifiziert werden. Die Kalibrierung zur Ermittlung der Rotationsauflösung erfolgt wie für **set_fly_rot_pos** (s.o.).
- Für die Übertragung von Rotations-Korrekturwerten stehen effektiv 31 Bit mit Vorzeichen zur Verfügung.

8.7.4 Kompensation zweidimensionaler Bewegungen

Zum Aktivieren einer encoderbasierten zweidimensionalen "Processing on the fly"-Korrektur (z. B. für einen XY-Verschiebetisch), insbesondere mit zusammenhängenden Markierungen im virtuellen Bildfeld, kann der Befehl **set_fly_2d** verwendet werden. Gegenüber einer Aktivierung mit **set_fly_x** und **set_fly_y** bringt 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 nichtlinearen Zusammenhängen zwischen den Encoder-Werten und den tatsächlichen XY-Tisch-Bewegungen (siehe Abschnitt "2D-Encoder-Kompensation für XY-Tische", Seite 206).
- **set_fly_2d** erlaubt die Verwendung von Koordinatentransformationen im virtuellen Bildfeld (siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208). Ab Version DLL 541, OUT 541 ist dies auch mit **set_fly_x** und/oder **set_fly_y** möglich.
- 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-Tischs "Processing on the fly"-korrigiert (der Laserfokus bleibt dadurch relativ zum XY-Tisch stehen). Dadurch werden nach XY-Tisch-Bewegungsfahrten unnötige Sprünge vermieden (siehe Kapitel 8.7.6 "Virtuelles Bildfeld", Seite 210).

Hinweise

- Eine **set_fly_2d**-Korrektur kann nicht mit anderen "Processing on the fly"-Korrekturen kombiniert werden (siehe Abschnitt "Übersicht", Seite 200).

2D-Encoder-Kompensation für XY-Tische

Bei erhöhten Anforderungen an die Markiergenauigkeit kann es notwendig sein, auch die mechanischen Abweichungen eines XY-Verschiebetisches zu kompensieren. Zu diesem Zweck kann mit dem Befehl **load_fly_2d_table** eine zweidimensionale Kompensationstabelle auf die RTC5 geladen werden (s.u.). Dann werden die Encoderwerte während einer mit **set_fly_2d** eingeleiteten "Processing on the fly"-Anwendung wie bei einer Feldkorrekturtabelle zweidimensional interpoliert und kompensiert.

Um überflüssige Initialisierungsfahrten einzusparen, kann mit dem Befehl **init_fly_2d** eine beliebige Anfangsposition des Verschiebetisches als Referenzwert für die 2D-Encoder-Kompensation festgelegt (auf der RTC5-Karte gespeichert) werden. Bei jedem nachfolgenden Zurücksetzen der Encoder mit **set_fly_2d** wird die aktuelle Position automatisch als neue Referenzposition übernommen, so dass die Beziehung zwischen den aktuellen Encoder-Werten und der absoluten Position des Tisches für die Kompensation erhalten bleibt. Diese Beziehung bleibt auch beim Beenden mit **fly_return** oder bei einem erneuten Starten mit **set_fly_2d** erhalten, dagegen nicht, wenn zwischendurch andere "Processing on the fly"-Korrekturen aktiviert werden, und auch nicht beim Zurücksetzen der Encoder mit einem externen /START (siehe **set_control_mode** mit Bit # 9 = 1). Die aktuell gültigen Referenzwerte können jederzeit mit **get_fly_2d_offset** ausgelesen werden.

Die Encoder-Kompensation wirkt ausschließlich auf die "Processing on the fly"-Korrektur. Die Befehle **get_encoder**, **store_encoder**, **read_encoder** sowie **wait_for_encoder**, **wait_for_encoder_mode** und **wait_for_encoder_in_range** verwenden die originalen, nicht kompensierten Encoder-Werte, ebenso die Aufzeichnung mit **set_trigger** (`Signal` = 43 oder 44) oder **get_value**.

Für den **load_fly_2d_table**-Befehl muss eine ASCII-Textdatei mit einer Kompensationstabelle bereitgestellt werden. Diese Kompensationstabelle muss Encoder-Kompensationen 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 Encoder-Stützpunkte bilden einen Rahmen, innerhalb dessen die Encoder-Kompensationswerte bilinear interpoliert werden. Encoder-Werte außerhalb dieses Rahmens werden vor der Interpolation auf denselben geclippt. Daher sollten die Stützpunkte zumindest den für die jeweilige Anwendung geforderten Stellbereich des XY-Verschiebetisches überdecken. Wenn die Stützpunkte die Werte ±524288 übersteigen, kann etwas an Genauigkeit verloren gehen.

Die 2D-Encoder-Kompensation ist nach **load_program_file** per Default nicht aktiv. Sie ist aktiv, sobald eine gültige Tabelle aus einer ASCII-lesbaren Textdatei auf die RTC5-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 Textdatei gelten folgende Regeln:

- Die Datei kann eine oder mehrere Tabellen enthalten, auch solche anderen Typs.
- Die 2D-Kompensationstabelle muss mit der Caption `[Fly2DTable<No>]` eingeleitet werden, wobei `<No>` der beim Befehl **load_fly_2d_table** anzugebenden Nummer entspricht.
- 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 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.

Koordinatentransformationen im virtuellen Bildfeld

Für **set_fly_2d** "Processing on the fly"-Anwendungen können auch Koordinatentransformationen im virtuellen Bildfeld definiert werden. Ab Version DLL 541, OUT 541 ist dies auch mit **set_fly_x** und/oder **set_fly_y** möglich.

Solche "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.

Im Gegensatz zu den sonstigen kopfspezifischen Koordinatentransformationen werden die (ggf. zusätzlichen) virtuellen Koordinatentransformationen noch *vor* der "Processing on the fly"-Korrektur auf das gesamte virtuelle Bildfeld angewendet.

Virtuelle Koordinatentransformationen können mit den Kontrollbefehlen **set_matrix** und **set_offset** bzw. **set_offset_xyz** (Parameter `HeadNo` = 4) definiert werden. Damit können Matrixkoeffizienten und Verschiebungen festgelegt werden (bei **set_offset_xyz** wird dabei allerdings `OffsetZ` ignoriert). Die Daten werden auf der Karte zwischengespeichert und fließen mit dem nächsten Aufruf von **set_fly_2d** in die Berechnung ein. Der Parameter `at_once` hat mit `HeadNo` = 4 keine Bedeutung. Solange eine "Processing on the fly"-Anwendung aktiv ist, können die Parameter für die virtuellen Koordinatentransformationen nicht verändert werden.

Der Stellbereich für die Verschiebungen ist ± 23 Bits. Die Matrixkoeffizienten dürfen einen Absolutwert von 1,5 nicht überschreiten. Die Verschiebung wird wie bei den kopfspezifischen Transformationen nach der Matrixoperation ausgeführt.

Für virtuelle Koordinatentransformationen können *nicht* verwendet werden:

- **set_scale**
- **set_angle**
- alle Listenbefehle für Koordinatentransformationen

8.7.5 Deaktivieren der “Processing on the fly”-Korrektur

Mit **fly_return** können alle “Processing on the fly”-Korrekturen (für beide Raumrichtungen gleichzeitig) deaktiviert werden. Dabei kann auch eine neue Ausgabeposition definiert werden.

Hinweise

- Wird die “Processing on the fly”-Korrektur nicht explizit deaktiviert, wirkt sie auch bei der Ausführung nachfolgender Listen, jedoch nicht in der Pause zwischen zwei Listen(1).
- 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 dann mit einem unzulässigen Parameterwert aufgerufen wird. Hierbei findet u.U. ein Sprung auf einen unkorrigierten Ausgabepunkt statt. Siehe auch Befehlsbeschreibungen.
- Eine mit **set_fly_x** aktivierte “Processing on the fly”-Korrektur wird durch **set_fly_y** *nicht* deaktiviert und vice versa. 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 200). Hierdurch findet u.U. ein harter Sprung auf einen neuen Ausgabepunkt statt.
- Eine mit **set_mcbsp_in** oder **set_mcbsp_in_list** aktivierte “Processing on the fly”-Korrektur wird auch durch **set_mcbsp_in** oder **set_mcbsp_in_list** mit `Mode` = 0 deaktiviert. Hierbei findet u.U. ein harter Sprung auf einen unkorrigierten Ausgabepunkt 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 zur Konfigurierung der Online-Positionierung deaktiviert (siehe Seite 188). Hierdurch findet u.U. ein harter Sprung auf einen neuen Ausgabepunkt statt.
- Um harte Sprünge zu vermeiden, sollte eine “Processing on the fly”-Korrektur nur mit **fly_return** deaktiviert werden.
- Mit **set_multi_mcbsp_in** aktivierte “Processing on the fly”-Korrekturen sollten mit **fly_return_z** beendet werden.
- Erfolgt ein Befehl zur Deaktivierung der “Processing on the fly”-Korrektur, so wird die Deaktivierung (ab Version OUT 515) erst nach dem Scanner-Delay ausgeführt (d.h. das Scanner-Delay wird noch mit “Processing on the fly”-Korrektur abgewartet). Bei älteren Versionen hat die Abschaltung schon vor dem Scanner-Delay stattgefunden.
- 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 Listenstopp (für beide Raumrichtungen gleichzeitig) deaktiviert.

(1) **set_end_of_list** deaktiviert die “Processing on the fly”-Korrektur nicht. Die Korrektur wirkt dagegen z.B. nicht bei den Kontrollbefehlen **goto_xy** und **goto_xyz**.

8.7.6 Virtuelles Bildfeld

Unter Ausnutzung des erweiterten Wertebereichs des virtuellen 24-Bit-Bildfelds (siehe Seite 135), können bei "Processing on the fly"-Anwendungen auch Befehlslisten mit Objekten, die um bis zu 16 mal größer als das reale 20-Bit-Bildfeld sind, geladen und ohne zusätzliche Steuerungsmaßnahmen autark und komplett abgearbeitet werden.

Bei eindimensionalen "Processing on the fly"-Anwendungen (z. B. bei Werkstücken auf einem Förderband) dürfen die Objekte nur in einer Dimension (parallel zur "Processing on the fly"-Richtung) größer sein als das reale Bildfeld. Bei zweidimensionalen "Processing on the fly"-Anwendungen (z. B. mit einem XY-Verschiebetisch) können die Beschriftungsobjekte einer Befehlsliste über das komplette virtuelle Bildfeld verteilt werden.

Falls die gesamte Markieraufgabe aus mehreren Teilmarkierungen ("Kacheln") besteht, deren Ausdehnung das reale Bildfeld nicht überschreitet, dann 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 – ggf. auch ohne "Processing on the fly"-Bewegung – abzuarbeiten. Falls die gesamte Markieraufgabe dagegen *nicht* in Kacheln separierbar ist (z. B. wenn ein großer geschlossener Rahmen um andere Objekte markiert werden soll), dann ist eine kontinuierliche "Processing on the fly"-Beschriftung notwendig.

Dass überhaupt Objekte außerhalb des realen Bildfelds markiert werden können, liegt daran, dass bei aktivierter "Processing on the fly"-Korrektur Koordinatenwerte, die außerhalb des realen Bildfelds liegen, während der Abarbeitung einer Liste erst nach der "Processing on the fly"-Korrektur auf die Randwerte des realen Bildfelds (20 Bit) geclippt werden (siehe Kapitel 7.3.5, Seite 141). Wenn die übergebenen Koordinatenwerte auf die Bewegung des Werkstücks bzw. Scan-Systems, auf die eingestellte "Processing on the fly"-Korrektur und gegebenenfalls auf die eingestellte Streckenverzögerung abgestimmt sind (siehe Kapitel 8.7.7, Seite 211), werden daher auch Koordinatenwerte außerhalb des realen Bildfelds zur Laufzeit durch die "Processing on the fly"-Korrektur auf Koordinatenwerte innerhalb des realen Bildfelds transformiert.

Koordinatenwerte, die auch *nach* erfolgter "Processing on the fly"-Korrektur noch außerhalb des realen Bildfelds liegen, werden dabei nach wie vor auf die Randwerte des realen Bildfelds geclippt (dabei werden die Fehlerbits für **get_marking_info** automatisch gesetzt, siehe Seite 213).

Damit die Koordinatenwerte *nach* "Processing on the fly"-Korrektur nie außerhalb des 20-Bit-Bildfeldes liegen, muss die Befehlsliste grundsätzlich immer derart gestaltet sein, dass spätestens immer an denjenigen Stellen gezielt Bewegungsfahrten (des Förderbands, XY-Tischs o.a.) eingebaut werden, an denen bei Listenausführung eine Überschreitung des 20-Bit-Bildfeldes droht. Zu diesem Zweck muss **wait_for_encoder**, **wait_for_encoder_mode**, **wait_for_encoder_in_range** oder **wait_for_mcbsp** in die Liste eingefügt werden. Diese Befehle unterbrechen die Listenausführung so lange, bis bestimmte Encoder- bzw. McBSP/SPI-Werte erreicht sind (siehe hierzu auch Kapitel 8.7.7, Seite 211). Die geeignete Unterteilung der gesamten Markieraufgabe und die Synchronisation mit der ein- oder zweidimensionalen "Processing on the fly"-Bewegung (des Förderbands, XY-Tischs o.a.) obliegt den Benutzern.

Bei **set_fly_2d**-Anwendungen sind auch Koordinatentransformationen im gesamten virtuellen Bildfeld möglich (siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208).

8.7.7 Synchronisation von “Processing on the fly”-Anwendungen

Die Abarbeitung von Befehlslisten kann durch einen Befehl oder ein externes Startsignal (z. B. ausgelöst durch eine Lichtschranke oder einen mechanischen Kontakt) gestartet werden (siehe Seite 78).

Falls die Abarbeitung des Scan-Prozesses nicht unmittelbar mit dem Lichtschranken-Signal gestartet werden soll (z. B. weil das Werkstück erst noch durch das Transportsystem von der Lichtschranke bis zum realen Bildfeld des Scan-Systems bewegt werden muss) kann man eine entsprechende Verzögerung z. B. wie folgt realisieren:

- Bei Positionsübergabe über die McBSP/SPI-Schnittstelle durch einen **wait_for_mcbsp**-Befehl am Beginn der Befehlsliste.
- Bei Positionsübergabe über Encoder-Zähler durch **set_fly_x**/**set_fly_y**, **set_fly_2d** oder **set_fly_rot** (zum Resetten der/des Zähler/s) und einem anschließenden **wait_for_encoder_mode**- oder **wait_for_encoder_in_range**-Befehl am Beginn der Befehlsliste. **wait_for_encoder_in_range** empfiehlt sich für zweidimensionale encoderbasierte “Processing on the fly”-Anwendungen. Diese Funktion wartet, bis beide Encoder gleichzeitig innerhalb des angegebenen Bereichs liegen und ist damit unabhängig von der tatsächlichen Bahnkurve, wie dieser Bereich erreicht wird(1).
- Bei Positionsübergabe über Encoder-Zähler kann alternativ bei externen Listen-Starts eine Streckenverzögerung konfiguriert werden, damit die Ausführung des Listen-Starts relativ zum auslösenden Eingangssignal bzw. Befehl entsprechend verzögert wird (siehe Seite 240). Dadurch kann auch die Ausführung von Objekten im virtuellen 24-Bit-Bildfeldbereich (außerhalb des realen 20-Bit-Bildfelds) so weit verzögert werden, bis die Koordinaten durch die “Processing on the fly”-Korrektur in das reale Bildfeld transformiert werden.

  Damit Totzeiten zwischen der Ausführung mehrerer (“Processing on the fly”-) Listenprogramme vermieden werden können, werden externe Listen-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 Seite 240). Außerdem kann der Listenbefehl **simulate_ext_start** verwendet werden, um mehrere Listen in definierten Abständen auszuführen.

Die Befehle **wait_for_encoder**, **wait_for_encoder_mode**, **wait_for_encoder_in_range** und **wait_for_mcbsp** können auch zur Unterbrechung einer Liste für zwischengeschaltete Bewegungsfahrten (des Förderbands, XY-Tischs o.a., z. B. bei Verwendung des virtuellen Bildfelds – siehe Kapitel 8.7.6, Seite 210) verwendet werden. Wenn die zwischengeschalteten Bewegungsfahrten nicht gezielt für Markierungen verwendet werden sollen, dann sollten die “Laser active”-Lasersteuersignale vor jeder Bewegungsfahrt (d.h. vor den Listenunterbrechungen mit **wait_for_encoder** etc.) ausgeschaltet werden (sonst kann es zu ungewünschtem Markieren oder Einbrennen während solcher Bewegungsfahrten kommen). Bei encoderbasierten “Processing on the fly”-Anwendungen kann der Laserfokus aber auch mit **park_position** vorübergehend (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.

(1) **wait_for_encoder** oder **wait_for_encoder_mode** müssten für jeden Encoder einzeln und nacheinander abgefragt werden. Dadurch wäre die gleichzeitige Erfüllung beider Encoder-Kriterien von der expliziten Bahnkurve der XY-Tisch-Bewegung abhängig und man müsste schon vor dem Laden der Listen eine geeignete Bahnkurve für den XY-Tisch festlegen.

Das Verhalten der Galvanometerscanner während einer Bewegungsfahrt (d.h. während einer Listenunterbrechung nach **wait_for_encoder** etc.) hängt von der verwendeten "Processing on the fly"-Korrektur ab:

- Bei McBSP/SPI-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 stehen. Nach der Unterbrechung muss ggf. entsprechend der gewünschten Anfangsposition der nächsten Markierung ein Sprung dorthin ausgeführt werden.
- Bei der encoderbasierten **set_fly_2d**-"Processing on the fly"-Korrektur werden die Positionen der Galvanometerscanner kontinuierlich entsprechend der aktuellen Encoderwerte (z.B. eines XY-Tischs) "Processing on the fly"-korrigiert (der Laserfokus bleibt dadurch relativ zum XY-Tisch stehen, d.h. während der Bewegungsfahrt immer auf die selbe Stelle des XY-Tischs gerichtet). Dies hat den Vorteil, dass beim Abarbeiten einer kontinuierlichen Linie, die aufgrund ihrer Länge mit einem **wait_for_encoder_in_range**-Befehl gestückelt werden muss, die Galvanometerscanner nach der zwischengeschalteten Bewegungsfahrt bereits an der richtigen Stelle stehen (hier ist nach der Unterbrechung kein Sprung erforderlich).
- 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 (z.B mit **fly_return** – siehe Kapitel 8.7.5, Seite 209) auszuschalten (die Galvanometerscanner bleiben dann stehen). Zum Wiedereinschalten nach der Bewegungsfahrt kann hier der Befehl **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 dann aktuell umgerechneten Koordinatenwerte außerhalb des virtuellen 24-Bit-Bildfeldes zu liegen kommen würden, wird ein Fehler-Bit gesetzt, welches mit **get_marking_info** (Bit # 9 = 1) ausgelesen werden kann. Falls es notwendig ist, bei der Abarbeitung der Liste bereits unmittelbar auf diesen Fehler zu reagieren, kann der Listenbefehl **if_not_activated** aufgerufen werden, um ggf. in eine Fehlerbehandlungsroutine zu verzweigen.

8.7.8 Encoder-Resets

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. nach einer Unterbrechung mit **wait_for_encoder**, vor einer neuen Markierungssequenz, etc.) 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-Tisch) soll dagegen meist im Gegensatz zu einem Endlos-Förderband der Zusammenhang zwischen den Encoder-Werten und der absoluten Position des XY-Verschiebetisches erhalten bleiben. Für diesen Zweck sollte für die "Processing on the fly"-Aktivierung der **set_fly_2d**-Befehl verwendet werden. Vor einer längeren Unterbrechung (mit **wait_for_encoder** etc.) kann dann die "Processing on the fly"-Korrektur mit **fly_return** deaktiviert und nach der Unterbrechung bei Bedarf mit dem Befehl **activate_fly_2d** wieder zugeschaltet werden (siehe hierzu auch Kapitel 8.7.4 "Kompensation zweidimensionaler Bewegungen", Seite 206 und Kapitel 8.7.7 "Synchronisation von "Processing on the fly"-Anwendungen", Seite 211). Mit **activate_fly_xy** 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 Encoder-Werten und der absoluten XY-Tisch-Position notwendig ist (siehe Abschnitt "2D-Encoder-Kompensation für XY-Tische", Seite 206).

8.7.9 Überwachung der "Processing on the fly"-Korrektur

Falls ein "Processing on the fly"-Anwenderprogramm nicht auf die vorgesehene Bewegung des Werkstücks bzw. Scan-Systems optimiert ist oder wenn sich die Geschwindigkeit des Werkstücks bzw. Scan-Systems während einer "Processing on the fly"-Anwendung unvorgesehenerweise stark ändert, wird unter Umständen der Randbereich des Bildfelds erreicht. Da ein Überschreiten des zulässigen Wertebereichs von der RTC5 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 RTC5 bei Überschreitung des Bildfeldrandes (und Clippen der Koordinatenwerte) interne Fehlerbits (#0…3), die mit **get_marking_info** ausgelesen werden können.

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 Fehlerbits #0…3 kann auch festgestellt werden, *welche* Kante des Bildfelds überschritten wurde. Durch jede Bereichsüberschreitung wird das entsprechende Fehlerbit gesetzt.
- Die Fehlerbits #0…3 werden bei der Initialisierung (mit **load_program_file**) und durch den Befehl **get_marking_info** zurückgesetzt. **get_marking_info** liefert also Informationen über Fehler, die seit der letzten Initialisierung bzw. dem letzten Aufruf von **get_marking_info** aufgetreten sind.
- Die RTC5 setzt ein Fehlerbit, wenn bei der Berechnung der Ausgabewerte aus den angegebenen Koordinatenwerten nach Anwendung der "Processing on the fly"-Korrektur eine Bereichsüberschreitung festgestellt wird. Eventuell eingestellte Koordinatentransformationen, Bildfeldkorrekturen und kompensierende Gain- und Offset-Korrekturen, die bei der Berechnung der Ausgabewerte erst nach der "Processing on the fly"-Korrektur angewendet werden (siehe Seite 141), 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 Fehlerbit #9 kann festgestellt werden, ob ggf. während des Wiedereinschaltens einer “Processing on the fly”-Korrektur mit **activate_fly_2d** oder **activate_fly_xy** eine Bereichsüberschreitung bezogen auf das virtuelle 24-Bit-Bildfeld stattgefunden hat (siehe Kapitel 8.7.7, Seite 211).

Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)

Die RTC5 prüft bei “Processing on the fly”-Anwendungen zusätzlich auch auf Überschreitung eines zweiten Wertebereichs, dessen Bereichsgrenzen mit **set_fly_limits** vorgegeben werden können. Auch bei solchen Bereichsüberschreitungen werden interne Fehlerbits gesetzt, die mit **get_marking_info** ausgelesen werden können (Fehlerbits#4…7).

Darüber hinaus kann mit den konditionalen Befehlen **if_fly_x_overflow**, **if_fly_y_overflow**, **if_not_fly_x_overflow** bzw. **if_not_fly_y_overflow** die Ausführung beliebiger Listenbefehle davon abhängig gemacht werden, ob eine Bereichsüberschreitung des kundenspezifischen Überwachungsbereichs stattgefunden hat oder nicht. Die konditionalen Befehle bleiben wirkungslos, wenn die im Befehlsparameter angegebene Bedingung (an die Fehlerbits#4…7) erfüllt bzw. nicht erfüllt ist, ansonsten sorgen sie dafür, dass der unmittelbar nachfolgende Listenbefehl übersprungen wird.

Hinweise

- Eine Überschreitung des kundenspezifischen Überwachungsbereichs führt nicht unbedingt zu einem Clippen der Ausgabe-Koordinatenwerte. Ein Clippen (und Setzen der Fehlerbits #0…3) erfolgt nur dann, wenn das maximale Bildfeld (–524.288 … 524.287 Bit) überschritten wird (der kundenspezifische Überwachungsbereich ist in der Regel kleiner).
- Die Fehlerbits #4…7 werden bei der Initialisierung (mit **load_program_file**), aber *nicht* durch den Befehl **get_marking_info** zurückgesetzt. Einzelne Fehlerbits werden gegebenenfalls implizit durch die konditionalen Befehle zurückgesetzt und können auch explizit mit dem Befehl **clear_fly_overflow** zurückgesetzt werden (siehe Befehlsbeschreibung).
- Auch die Fehlerbits #4…7 berücksichtigen weder Koordinatentransformationen noch Bildfeldkorrekturen oder kompensierende Gain- und Offset-Korrekturen.
- Auch für Rotations-Fly-Anwendungen ist die Überwachung eines rechteckigen Bereichs möglich, aber keine Überwachung eines Drehwinkelbereichs.

8.7.10 Schleppverzugskompensation von Encoderwerten für "Processing on the fly"-Anwendungen

Insbesondere bei "Processing on the fly"-Anwendungen mit variabler Encodergeschwindigkeit führt der Schleppverzug der Galvanometerscanner des Scan-Systems zu gewissen Positionsfehlern.

Für Anwendungen mit höchsten Genauigkeitsanforderungen kann deshalb mit **set_fly_tracking_error** eine Schleppverzugskompensation aktiviert werden.

8.7.11 "Processing on the fly"-Korrektur für die Z-Achse — "FlyZ-Korrektur" (ab Version DLL 530, OUT 531)

Ab Version DLL 530, OUT 531 kann mit **set_fly_z** eine encodergesteuerte "Processing on the fly"-Korrektur für die Z-Achse ("FlyZ-Korrektur") 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- und Y-Achse zugeschaltet werden. Dadurch wird eine dynamische Markierung durch einen schräg zur Werkstücksbewegung stehenden Scan-Kopf (bzw. eine Markierung von Werkstücken, die in einer schräg zur Scan-Kopf-Bearbeitungsebene stehenden Fläche bewegt werden) ermöglicht. Bei aktivierter FlyZ-Korrektur wird die Z-Komponente der Werkstücksbewegung – d.h. die Z-Komponente im Koordinatensystem des Scan-Systems – durch Nachregelung der Z-Achse (vario*SCAN* dynamische Fokussiereinheit) gemäß den aktuellen Encoder-Signalen kompensiert.

Voraussetzungen

- Auf der RTC5 muss sowohl die "Processing on the fly"-Option als auch die 3D-Option freigeschaltet sein.
- Die gewünschte Markierung muss statisch, also bei fest stehendem Werkstück, ordnungsgemäß funktionieren:
  - Die Schräge des Werkstücks muss durch Verwendung passender 3D-Vektor- und Bogenbefehle im Anwenderprogramm abgebildet sein.
  - Der vario*SCAN* muss in der Lage sein, den XY-Galvanometerscanner-Bewegungen des Scan-Kopfs zu folgen.
- Bei bewegten Werkstücken muss der vario*SCAN* 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 bzgl. 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 Kapitel 8.6.2 "Anschluss und Initialisierung", Seite 193.

Hinweise zur Verwendung

- Zur allgemeinen Verwendung der "Processing on the fly"-Korrektur und zur Ermittlung von Skalierungsfaktoren siehe Kapitel 8.7 ""Processing on the fly" (optional)", Seite 199.
- Beim Aktivieren der FlyZ-Korrektur mit **set_fly_z** kann angegeben werden, welcher der beiden Encoder-Zähler hierzu verwendet werden soll. Da für **set_fly_x** bereits Encoder-Zähler Encoder0 und für **set_fly_y** Encoder-Zähler Encoder1 verwendet wird (für **set_fly_rot** Encoder0), muss gegebenenfalls einer der beiden Encoder-Zähler 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 zweidimensionale 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 einen (teilweise) unkorrigierten Ausgabepunkt statt.
- **set_fly_z** deaktiviert eine zuvor 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.
- Umgekehrt wird eine mit **set_fly_z** aktivierte "Processing on the fly"-Korrektur von **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos**, **set_mcbsp_in** und **set_mcbsp_in_list** deaktiviert. Hierdurch findet u.U. ein harter Sprung auf einen neuen Ausgabepunkt statt.
- Um harte Sprünge 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 "Bedingte Befehlsausführung", Seite 244, werden die Befehle **set_fly_limits_z**, **if_fly_z_overflow** und **if_not_fly_z_overflow** bereitgestellt. Fehlerbits (Bit #24…Bit #25 von **get_marking_info**) für die kundenspezifische Überwachung von FlyZ-Anwendungen können mit **clear_fly_overflow** zurückgesetzt werden.

8.8 Pixelausgabemodus – Rasterbilder (Bitmaps) scannen

Die in Kapitel 7.1 beschriebenen Vektorbefehle dienen dem Scannen von vektorbasierten Grafikobjekten. Die RTC5 erlaubt es jedoch auch, Rasterbilder (Bitmaps) zu reproduzieren. Mit geeignet eingestellten Lasern erhält man Schwarzweißbilder oder Graustufenbilder. Natürlich kann man auch raster- und vektorbasierte Bilder beliebig miteinander kombinieren.

8.8.1 Funktionsprinzip

Ein Rasterbild wird Zeile für Zeile erstellt. Jede Zeile besteht aus einer Zahl von äquidistanten Bildelementen (Pixeln) und wird mit einem einzelnen Scan reproduziert. Während dieses Scans bewegt sich der Laserfokus – wie bei einem gewöhnlichen Mark-Befehl – mit annähernd konstanter Geschwindigkeit über die ganze Bildzeile (die Bewegung wird mikrovektorisiert). Die einzelnen Pixel werden im Vorübergehen markiert, indem jedem Pixel an der entsprechenden Stelle ein Laserpuls spendiert wird. Durch Variation der Laserenergie von Pixel zu Pixel erhält man Graustufenbilder (im Spezialfall auch Schwarzweißbilder). Zur Lasersteuerung werden Pulslängen (digital) und Spannungswerte (analog) ausgegeben.

Hinweise

- Der Pixelausgabemodus kann mit "Processing on the fly" (siehe Seite 199) kombiniert werden. Er kann *nicht* mit Sky-Writing (siehe Seite 127) oder Wobbel (siehe Seite 189) kombiniert werden.

8.8.2 Softwarebefehle

Bevor eine Bildzeile geschrieben wird, sollte mit einem Jump-Befehl ein Sprung zum Zeilenanfang ausgeführt werden.

Am Anfang einer jeder Bildzeile werden dann mit dem Befehl **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). Die Pixelausgabeperiode wird über den Parameter `HalfPeriod` (*halbe* Pixelausgabeperiode) definiert. Das ist gleichzeitig auch die halbe Laser-Periode. Der Pixelabstand zwischen zwei benachbarten Bildpunkten in der Zeile und damit auch die Laufrichtung wird bei **set_pixel_line** über einen 2D-Vektor (`dX`,`dY`) definiert und bei **set_pixel_line_3d** (für eine Pixel-Beschriftung auf schräger Fläche) über einen 3D-Vektor (`dX`,`dY`,`dZ`). Mit **set_pixel_line**/**set_pixel_line_3d** wird zusätzlich festgelegt, an welchem der beiden Analogausgänge (ANALOG OUT1 oder ANALOG OUT2) die Analogspannungswerte nachfolgender **set_pixel**- bzw. **set_n_pixel**-Befehle ausgegeben werden sollen.

Unmittelbar auf den **set_pixel_line**/**set_pixel_line_3d**-Befehl folgend muss dann für jedes Bildelement der Zeile separat der Befehl **set_pixel** aufgerufen werden. Damit werden die Laserenergien festgelegt, die an den entsprechenden Pixelpositionen entladen werden sollen. Es kann jeweils die Länge eines Pulses und/oder ein 12-Bit-wertiger Analogspannungswert spezifiziert werden (siehe "Lasersteuerung" auf Seite 218). Pixel-Pulse werden am LASER1-Ausgang ausgegeben, Analogspannungswerte entweder am ANALOG OUT1- oder am ANALOG OUT2-Analogausgang (s. o.).

Soll für mehrere (`n`) unmittelbar aufeinander folgende Bildelemente einer Zeile der gleiche Analogspannungs- bzw. Pulslängenwert spezifiziert werden, so kann an Stelle von `n` **set_pixel**-Befehlen der Befehl **set_n_pixel** verwendet werden. Auf der Karte wird dann nur ein Befehl gespeichert, bei der Ausführung der Liste aber `n` mal ein entsprechender **set_pixel**-Befehl ausgeführt. Dadurch kann die Länge einer Liste insbesondere bei Schwarzweißbildern u. U. drastisch verkleinert werden. Der Befehl **set_n_pixel** sollte nicht mit dem **n_set_pixel**-Befehl (Multi-Board-Version des **set_pixel**-Befehls) verwechselt werden.

Vor dem Ende einer Bildzeile darf kein anderer Befehl außer **set_pixel** oder **set_n_pixel** in die Liste eingetragen 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**-Befehl, der nicht auf einen anderen **set_pixel**/**set_n_pixel**- oder **set_pixel_line**/**set_pixel_line_3d**-Befehl folgt, wird bei der Abarbeitung ignoriert und ist dabei ein kurzer Listenbefehl (siehe Seite 250).

Hinweise

- Die Befehle **set_pixel_line**, **set_pixel_line_3d**, **set_pixel** und **set_n_pixel** sind Listenbefehle und werden daher in eine Liste geschrieben.
- Die Zahl der Bildelemente pro Zeile wird nur durch die Kapazität des Listenspeichers der RTC5 begrenzt (siehe Seite 651). 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 Zeile muss mit dem Befehl **set_pixel_line** oder **set_pixel_line_3d** gestartet werden.
- Der Pixelabstand (`dX`,`dY`) in x- 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 ein beliebiges ganzzahliges Vielfaches von 1/64 µs sein (mindestens jedoch 104/64 µs) und ist unabhängig von der Ausgabeperiode von 10 µs, mit der die mikrovektorisierte Bewegung der Galvanometerscanner des Scan-Systems erfolgt. Bei sehr niedrigen Pixelfrequenzen erfolgen mehrere Scannerschritte je Pixel, bei hohen dagegen gegebenenfalls mehrere Pixelpulse bzw. Spannungswertänderungen je Scannerschritt.
- Ein Pixelausgabemodus im 10 µs-Raster mit variabler Geschwindigkeit und/oder krummliniger Bahn kann mit Hilfe der Mikrovektor-Befehle realisiert werden (siehe Seite 221).

8.8.3 Lasersteuerung

Je nach verwendetem Lasertyp kann die Laserenergie, die an jeder Pixelposition entladen wird, über die Laserpulsdauer oder über die Laserleistung pro Pixel variiert werden.

Variation der Laserpulsdauer

Der Befehl **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 Seite 219). Der Befehl **set_n_pixel** definiert die Pixelpulslänge für `n` unmittelbar aufeinander folgende Pixel einer Zeile.

Variation der Laserleistung

Der Befehl **set_pixel** erlaubt die Spezifizierung eines 12-Bit-wertigen Analogspannungswerts für jedes einzelne Bildelement. Der Wert wird entweder am ANALOG OUT1- oder ANALOG OUT2-Analogausgang der RTC5 ausgegeben (s. o., zur Synchronisation siehe Seite 219). Der Befehl **set_n_pixel** definiert den Analogspannungswert für `n` unmittelbar aufeinander folgende Pixel einer Zeile.

Hinweise

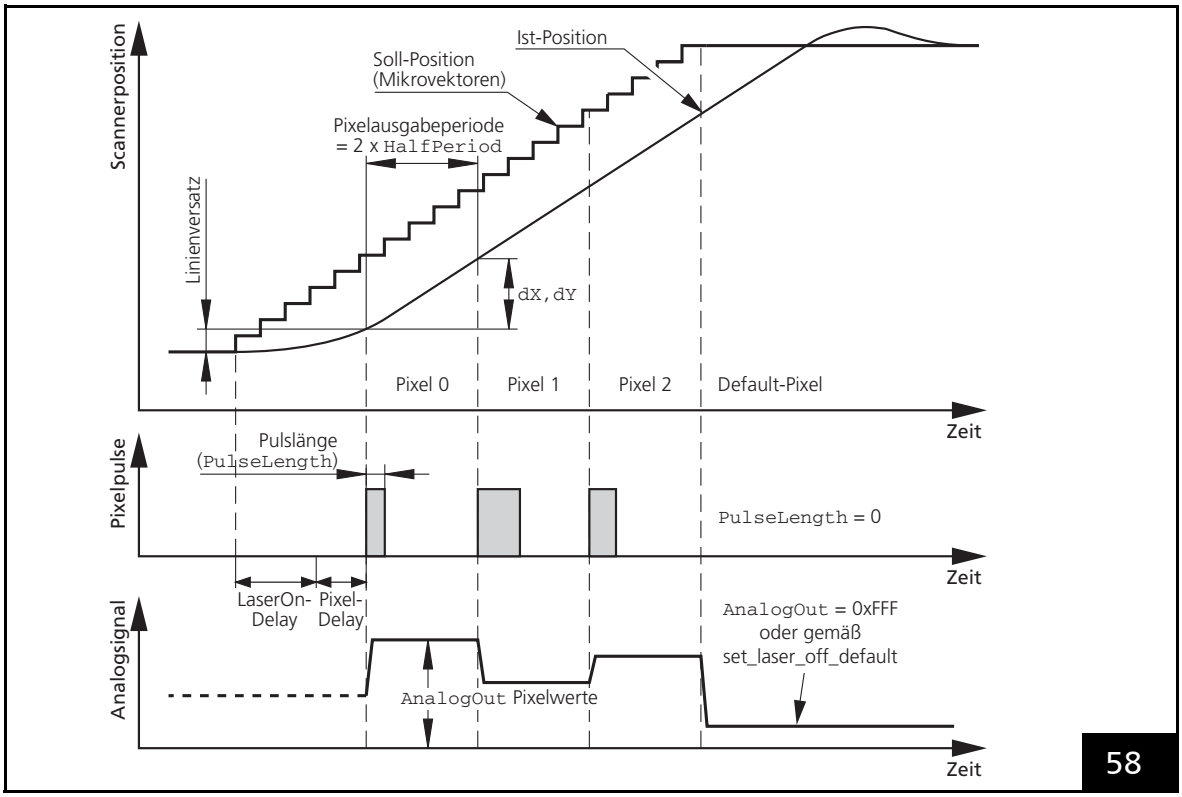
- Es ist empfehlenswert, einige Experimente durchzuführen, um eine passende Gradationskurve für eine gleichmäßige Grauskala zu bestimmen. Die erhaltenen Pixel-"Farben" (Graustufenwerte) hängen stark vom eingesetzten Material und vom Laser ab.
- Bei aktivem Pixelausgabemodus
  - ist das LaserOn-Signal dauernd eingeschaltet,
  - werden in der CO_2-Version die zuvor eingestellten Periodendauer- und Pulslängenwerte überschrieben und die Pixelpulse sowohl am LASER1-Ausgang als auch phasenverschoben am LASER2-Ausgang ausgegeben,
  - werden in einer YAG-Version nur die zuvor eingestellten Werte für das Q-Switch-Delay und/oder die Länge des FirstPulseKiller-Signals berücksichtigt (siehe "Synchronisation", Seite 219); die eingestellten Periodendauer- und Pulsdauerwerte werden überschrieben; am LASER1-Ausgang werden die Pixelpulse und am LASER2-Ausgang das FirstPulseKiller-Signal ausgegeben,

  - werden im Lasermode 4 und Lasermode 6 am LASER1-Ausgang nach wie vor die eingestellten freilaufenden Standby-Pulse (also *keine* Pixelpulse) und im Lasermode 4 am LASER2-Ausgang das FirstPulseKiller-Signal ausgegeben,
  - wird kein Softstart (siehe Seite 152) ausgeführt,
  - ist kein Sky-Writing (siehe Seite 127) möglich.
- Bei deaktiviertem Pixelausgabemodus (also z. B. auch während eines Jumps für einen Zeilenrücklauf zwischen der Bearbeitung zweier Bildzeilen) werden evtl. zuvor aktivierte Standby-Pulse ausgegeben.
- Die Periodendauer- und Pulsdauerwerte für den normalen Makierbetrieb müssen nach dem Pixelausgabemodus wieder neu eingestellt werden.

8.8.4 Synchronisation

Abb. 58 zeigt das Zeitdiagramm der Pixelausgabe für eine Bildzeile mit 3 Pixeln.

Die Bewegung der Galvanometerscanner beginnt mit dem ersten **set_pixel**- oder **set_n_pixel**-Befehl. Die Bildzeile wird wie bei einem gewöhnlichen Mark-Befehl abgefahren (die Bewegung wird mikrovektorisiert). Nach einer anfänglichen Beschleunigungsphase bewegen sich die Scanner mit annähernd konstanter Geschwindigkeit entsprechend der angegebenen halben Pixelausgabeperiode `HalfPeriod` und dem Pixelabstand (`dX`,`dY`) über die ganze Bildzeile. Am Ende der Bildzeile (vor dem auf den letzten **set_pixel**/**set_n_pixel**-Befehl folgenden Nicht-**set_pixel**/**set_n_pixel**-Befehl, der den Pixelausgabemodus beendet) wird ein Default-Pixel ausgegeben. Dadurch bewegen sich die Scanner mit der gleichen Geschwindigkeit über die eigentliche Endposition hinaus. Angefangene 10 µs-Takte werden (insbesondere bei Pixelfrequenzen > 100 kHz) bis zum Ende ausgeführt.



58

Zeitdiagramm der Scannerpositionen und der Lasersteuersignale im Pixelausgabemodus.
Die Pixelausgabeperiode entspricht hier ca. 4,5 Mikrostufen.

Die “Laser active”-Lasersteuersignale werden am Anfang der Bildzeile durch den **set_pixel_line**-Befehl ausgeschaltet (gegebenenfalls – z. B. wenn ein vorausgehender Mark-Befehl noch nicht fertig abgearbeitet ist – wird vor dem Ausschalten ein LaserOff-Delay abgewartet). Die “Laser active”-Lasersteuersignale werden dann mit dem ersten **set_pixel**/**set_n_pixel**-Befehl, allerdings um ein LaserOn-Delay gegenüber dem Start der Scannerbewegung verzögert, eingeschaltet. Die Pixelpulse werden je nach zuvor aktivierter Laser-Version (und den dafür eingestellten Werten) unter Umständen zusätzlich um ein Pixel-Delay verzögert ausgegeben (siehe Abb. 58). Das Pixel-Delay ist

- = 0 in der CO_2-Version,
- = 0 (bzw. Q-Switch-Delay) in der YAG-Version 1,
- = Länge des FirstPulseKiller-Signals (bzw. Q-Switch-Delay) in der YAG-Version 2,
- = 10 µs (bzw. Q-Switch-Delay) in der YAG-Version 3,
- = Q-Switch-Delay in der YAG-Version 5.

Das Analogsignal (am ANALOG OUT1- oder ANALOG OUT2-Ausgang) wechselt synchron mit der vorderen Flanke eines jeden Pixelpulses (siehe Abb. 58).

Hinweis: Der DA-Wandler braucht etwa 1,5 µs … 3 µs, um ein stabiles analoges Ausgabesignal zu produzieren. Für Pixelfrequenzen über ca. 100 kHz (d.h. für `HalfPeriod` < ca. 320) wird die DA-Wandlung nicht in jedem Fall vollständig abgeschlossen. Bei solchen Pixelfrequenzen muss die ausreichende Funktionsfähigkeit von Benutzern sorgfältig überprüft werden.

Die Verzögerung der Pixelpulse bzw. der Analogwertänderung relativ zur Scannerbewegung (d.h. LaserOn-Delay und Pixel-Delay) sollte so eingestellt werden, dass die Beschleunigungsphase der Galvanometerscanner (in der die Scanner auf konstante Geschwindigkeit gebracht werden) ausgeblendet wird. Alternativ können Verzeichnungen während der Beschleunigungsphase vermieden werden, indem zu Beginn einer jeden Zeile einige Leerpixel (z. B. mit Pulslänge Null) eingefügt werden.

Die ganze Bildzeile kann aufgrund des Schleppverzugs und der ausgeblendeten Beschleunigungsphase leicht verschoben sein (siehe “Linienversatz” in Abb. 58). Dies lässt sich durch Anpassen der Anfangspositionen der Bildzeile (Rückwärtsverlängerung der Bildzeile) kompensieren.

Am Ende der Bildzeile (zu Beginn des Default-Pixels) werden die Pixelpulslänge und der Analogwert jeweils auf einen Default-Wert gesetzt. Der Default-Pixelpulslängen-Wert ist 0, wenn nicht zuvor mit **set_default_pixel** oder **set_default_pixel_list** ein anderer Wert festgelegt wurde. Der Default-Analogwert ist 0xFFF (=4095), sofern nicht zuvor mit **set_port_default** oder **set_laser_off_default** ein anderer Wert festgelegt wurde. Die “Laser active”-Lasersteuersignale werden dann im dem Default-Pixel nachfolgenden 10 µs-Takt ohne LaserOff-Delay ausgeschaltet.

Am Ende der Pixelzeile wird *kein* Scanner-Delay eingefügt, weil dessen Größe von der indirekt über `HalfPeriod` und (`dX`,`dY`) gewählten Markiergeschwindigkeit abhängt. Bei Bedarf kann aber nach dem letzten **set_pixel**- oder **set_n_pixel**-Befehl der Pixelzeile ein geeignetes **long_delay** eingefügt werden.

Hinweise

- Die (halbe) Pixelausgabeperiode `HalfPeriod` sollte auf die maximal benötigte Pulslänge abgestimmt sein (für `HalfPeriod` < `PulseLength`/2 ist der Pixelpuls erst im nächsten Pixel zu Ende, d.h. der Laser geht zwischendurch nicht aus). Siehe auch obigen Hinweis zum DA-Wandler.
- Die (halbe) Pixelausgabeperiode `HalfPeriod` und der Pixelabstand (`dX`,`dY`) und damit indirekt die Markiergeschwindigkeit sollten auf die Dynamik des Scan-Systems abgestimmt sein.
- Das LaserOn-Delay sollte entsprechend des Schleppverzugs des Scan-Systems eingestellt werden.
- Im Lasermode 4 und Lasermode 6 werden die Pixelpulse zwar nicht am LASER1-Ausgang ausgegeben (stattdessen freilaufende Standby-Pulse). Sie werden aber intern trotzdem erzeugt und der Analogspannungswert synchron dazu verändert.
- Die Ausgabe-Synchronisation kann nicht zusammen mit der Pixelausgabe verwendet werden. Die Ausgabe-Synchronisation sollte zuvor deaktiviert werden.

8.9 Mikrovektor-Befehle

Durch die Mikrovektor-Befehle **micro_vector_abs** und **micro_vector_rel** (und die entsprechenden 3D-Befehle **micro_vector_abs_3d** und **micro_vector_rel_3d**) werden die Galvanometerscanner per hartem Sprung direkt in die angegebene Position bewegt. Die auszuführenden Vektoren werden hier also (anders als bei normalen Vektor-Befehlen wie **jump_abs** oder **mark_abs**) von der RTC5 *nicht* in Mikrovektoren zerlegt, sondern innerhalb eines einzigen 10 µs-Taktes ausgeführt.

Über die Parameter `LasOn` und `LasOff` der Mikrovektor-Befehle können die "Laser active"-Lasersteuersignale individuell mit den angegebenen Delays ein- und ausgeschaltet werden. Die mit **set_laser_delays** definierten Laser-Delays werden dabei nicht überschrieben. Sie bleiben für normale Jump- und Mark-Befehle weiter gültig.

Unabhängig von der aktuell eingestellten Sprung- oder Markiergeschwindigkeit können also mit den Mikrovektor-Befehlen Linien und Bögen in beliebig feinen oder groben Schritten und auch mit variabler Geschwindigkeit markiert werden. Dadurch kann beispielsweise bei konstanter Laserleistung die Linienstärke innerhalb einer Linie variiert oder auch ein Pixelausgabemodus im 10 µs-Raster mit variabler Geschwindigkeit und/oder krummliniger Bahn realisiert werden.

Hinweise

- Mikrovektor-Befehle warten zwar gegebenenfalls ein vorhergehendes Scanner-Delay ab (z. B. das Scanner-Delay eines vorausgehenden **jump_abs**-Befehls), initiieren selbst aber keine neuen Scanner-Delays.
- Der Benutzer ist selbst dafür verantwortlich, die Mikrovektoren passend zu parametrisieren, d.h. geeignet kleine Mikrovektoren für die zu markierenden Linien und Bögen oder Pixelbilder festzulegen und gegebenenfalls Scanner-Delays zur Kompensation des Schleppverzuges zu integrieren (etwa durch Einfügen mehrerer Mikrovektor-Befehle mit identischem Zielpunkt). Überkreuzungen zwischen LaserOn und LaserOff sollten vermieden werden. Ferner sollten keine Laser-Delays neu definiert werden, solange die vorherigen noch nicht abgelaufen sind (siehe Seite 121, zeitgleich ablaufen und neu definieren ist jedoch zulässig).
- Im Gegensatz zu normalen Vektoren werden bei den Mikrovektor-Befehlen
  - die Vektoren nicht mikrovektorisiert, sondern mit einem einzigen harten Sprung ausgeführt,
  - eine mit **set_wobbel** oder **set_wobbel_mode** festgelegte Wobbelbewegung (siehe Seite 189) *nicht* berücksichtigt (aber auch nicht deaktiviert) und
  - der Sky-Writing-Modus (siehe "Sky-Writing", Seite 127) *nicht* berücksichtigt (aber auch nicht deaktiviert).
- Zur Berechnung der Ausgabewerte aus den angegebenen Koordinatenwerten werden dagegen (wie bei normalen Vektoren) auch bei Mikrovektor-Befehlen die folgenden Korrekturen (in der aufgeführten Reihenfolge, sofern zuvor eingestellt) berücksichtigt:
  - Erfolgt der Befehlsaufruf innerhalb eines mit einem "AbsCall" aufgerufenen Unterprogramms, so werden die angegebenen Koordinatenwerte (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt (siehe auch Seite 83).
  - Falls mit **set_fly_x**, **set_fly_y**, **set_fly_rot** oder **set_fly_x_pos**, **set_fly_y_pos**, **set_fly_rot_pos** eine "Processing on the fly"-Korrektur aktiviert wurde, wird sie auf die x- und y-Koordinaten angewandt (siehe Seite 199).

- Falls eine Koordinatentransformation zur Ausrichtung des Scan-Systems relativ zum Bildfeld definiert wurde (siehe **set_matrix**, **set_offset**, **set_scale**, **set_angle** und entsprechende Listenbefehle), wird sie auf die x- und y-Koordinaten angewandt (siehe Seite 183).
- Die Z-Koordinaten von **set_offset_xyz** oder **set_offset_xyz_list** sowie die Defokussierung von **set_defocus** oder **set_defocus_list** werden berücksichtigt.
- Koordinatenwerte, die den realen Bildfeldbereich übersteigen (virtuelles Bildfeld, siehe Seite 135), werden auf die Randwerte des realen Bildfelds (20-Bit) geclippt.
- Entsprechend der mit **select_cor_table** oder **select_cor_table_list** zugewiesenen Korrekturtabelle wird eine 2D- bzw. 3D-Bildfeldkorrektur durchgeführt (siehe Seite 136).
- Falls mit dem Befehl **auto_cal** eine automatische Selbstkalibrierung aktiviert wurde (siehe Seite 224), wird bei der Berechnung der Ausgabewerte gegebenenfalls ein kompensierender “Gain“ und “Offset” berücksichtigt (eine Gain- und Offset-Korrektur kann auch mit **set_hi** eingestellt werden, siehe Seite 227).

Überlaufende Ausgabewerte werden gegebenenfalls auf die Randwerte des maximal möglichen Wertebereichs geclippt.

8.10 Zeitbemessene Vektor- und Bogenbefehle

Vektorbefehle (Jump- und Mark-Befehle) und Bogenbefehle werden von der RTC5 üblicherweise so ausgeführt, dass sich der Laserfokus mit einer definierten (Sprung- oder Markier-) Geschwindigkeit entlang der Bildfeldoberfläche bewegt. Für die meisten Lasermarkierungs- und Materialbearbeitungs-Anwendungen ist dies ausreichend.

Manche Anwendungen erfordern jedoch, dass jeder Vektor- oder Bogenbefehl exakt dieselbe Zeit in Anspruch nimmt, ungeachtet seiner räumlichen Länge. In diesem Fall ist es praktikabler, eine Sprung- oder Markier-*Dauer* anstelle einer Sprung- oder Markier-*Geschwindigkeit* zu spezifizieren.

Die Befehle

- **timed_jump_abs**
- **timed_jump_rel**
- **timed_mark_abs**
- **timed_mark_rel**
- **timed_arc_abs**
- **timed_arc_rel**

erlauben eine Spezifikation der Dauer eines Vektor- oder Bogenbefehls mit einer Genauigkeit von 10 µs (der Ausgabeperiode der Mikrovektoren) in einem Bereich von 10 µs bis 167.772.160 µs (≈ 2,8 min).

Ein Jump- oder Mark-Vektor oder ein Bogen (aber keine Ellipse) wird in eine entsprechende Anzahl von Mikrostufen aufgeteilt, siehe auch Kapitel 7.1.2 "Mikrovektorisierung", Seite 107. Die Sprung- und Markiergeschwindigkeit (d.h. die Geschwindigkeit des Laserfokus bzw. die Winkelgeschwindigkeit der Spiegelbewegung) hängt dadurch automatisch von der Länge der Vektoren oder Bögen ab.

Hinweise

- Nach einem zeitbemessenen Vektor- oder Bogenbefehl wird wie nach einem gewöhnlichen Vektor- oder Bogenbefehl ein Jump-, Mark- oder Polygon-Delay eingefügt. Die Gesamtzeit für die Ausführung des Befehls ist also die Summe aus der spezifizierten Zeit und dem zugehörigen Delay, siehe auch Kapitel 7.2.2 "Scanner-Delays", Seite 113.
- Bei aktivierter vektorkontrollierter Lasersteuerung können auch zeitbemessene Para-Befehle genutzt werden (zeitbemessene Para-Bogenbefehle werden nicht zur Verfügung gestellt):
  – **timed_para_jump_abs**
  – **timed_para_jump_rel**
  – **timed_para_mark_abs**
  – **timed_para_mark_rel**
- Bei freigeschalteter 3D-Option können auch zeitbemessene 3D-Vektoren ausgeführt werden (zeitbemessene 3D-Bogenbefehle werden nicht zur Verfügung gestellt):
  – **timed_jump_abs_3d**
  – **timed_jump_rel_3d**
  – **timed_mark_abs_3d**
  – **timed_mark_rel_3d**.
  Bei zusätzlich aktivierter vektorkontrollierter Lasersteuerung können auch zeitbemessene 3D-Para-Befehle ausgeführt werden:
  – **timed_para_jump_abs_3d**
  – **timed_para_jump_rel_3d**
  – **timed_para_mark_abs_3d**
  – **timed_para_mark_rel_3d**
- Ab RTC5OUT.out Version 526 dauern auch zeitbemessene Markierbefehle (aber nicht zeitbemessene Jump-Befehle) der räumlichen Länge 0 die angegebene Zeit (sofern eine Ausführdauer > 5 µs spezifiziert wird) und verhalten sich auch bzgl. der Lasersteuersignale und der Mark- und Polygon-Delays wie zeitbemessene Markierbefehle mit einer endlichen räumlichen Länge (siehe auch Seite 116). Dadurch können "Processing on the fly"-Anwendungen, bei denen die "Processing on the fly"-Kompensation z.B. aus einer Roboterarmbewegung bereits in die Markierkoordinaten einberechnet worden ist, problemlos zeitlich synchronisiert ablaufen. Dies kann auch dazu verwendet werden, "Punkte" innerhalb eines Polygonzuges gesondert zu markieren.

8.11 Automatische Selbstkalibrierung

8.11.1 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 bzw. Verkleinerung (Gain-Drift) des Arbeitsfeldes äußern, die tolerierbaren Werte überschreiten.

Bei solchen Anwendungen ist es hilfreich, diese erst nach Erreichen der Scanner-Betriebstemperatur zu starten. Außerdem sollte das Scan-System möglichst geringen Schwankungen der Umgebungsbedingungen ausgesetzt werden und mit möglichst konstanter Belastung betrieben werden.

Für höhere Anforderungen an die Langzeit-Wiederholgenauigkeit können SCANLAB Scan-Systeme (optional für Aperturen ≥ 10 mm) mit einem zusätzlichen, internen Sensorsystem zur automatischen Selbstkalibrierung (ASC-Sensorsystem, Home-In-Sensoren) ausgestattet werden. Zusammen mit den dafür bereitgestellten RTC5-Befehlen können damit die Positionsdetektoren der Galvanometerscanner jederzeit kalibriert und Gain- und Offset-Drifts zuverlässig kompensiert werden. Die Positioniergenauigkeit bleibt so über lange Zeiträume erhalten. Verbleibende Langzeit-Drifteffekte sind von gleicher Größenordnung wie die kurzfristigen Wiederholgenauigkeiten.

8.11.2 Funktionsweise

Mit dem **auto_cal**-Befehl kann eine Messroutine gestartet werden, mit der die exakten Ansteuerwerte für Referenzpositionen (Home-In-Positionen) bestimmt werden, die durch das interne Sensorsystem definiert sind. Die Galvanometerscanner werden dazu von der RTC5 zu den entsprechenden Positionen im Scan-Bereich gesteuert.

Zur *Vermessung* der Drift sollte die Messroutine wie folgt aufgerufen werden:

- beim Einrichten der Maschine, um Referenzwerte für die Home-In-Positionen zu ermitteln und
- während der Anwendung in geeigneten Zeitabständen, um festzustellen, ob und wie sich die Home-In-Positionen verändert haben.

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- und Offset-Werte berechnet und eingestellt werden. Mit diesen werden dann die Ansteuerwerte aller nachfolgenden Vektor- oder Bogenbefehle für beliebige Punkte im Bildfeld korrigiert (transformiert).

Die RTC5 stellt sowohl Befehle für eine automatische Selbstkalibrierung als auch Befehle für eine kundeneigene Kalibrierung bereit. Bei der automatischen Selbstkalibrierung werden die Gain- und Offset-Werte von der RTC5 automatisch ermittelt und eingestellt.

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 und ob das Sensorsystem funktionsfähig ist. Diese ASC-Hardware-Überprüfung wird auch für **auto_cal**(`Command` = 0) und u.U. 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 der Laser ausgeschaltet sein, und es werden keine anderen Befehle an das Scan-System gesendet.
- Zur Gain/Offset-Korrektur:
  Die gemäß verwendeter Bildfeld-Korrekturtabelle (*.CT5) korrigierten Ansteuerwerte werden mit dem aktuell gesetzten Gain-Wert multipliziert und anschließend um den Offset-Wert additiv verändert. Die schlussendlich auszugebenden Positionen der Galvanometerscanner werden automatisch auf die zulässigen Werte geclippt. Die Grenzen können je nach den aktuellen XY-Koordinaten auch schon bei Werten innerhalb der datentechnisch zulässigen Bereiche für die Gain- und Offset-Werte erreicht werden. Die nach einer Initialisierung (mit **load_program_file**) gesetzten Gain- und Offset-Werte (Gain = 1.0 und Offset = 0) lassen die Ansteuerwerte für Vektor- oder Bogenbefehle unverändert.
- Für i*DRIVE*-Scan-Systeme (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*) gilt zusätzlich (ab DLL 513):
  - Zu Beginn der Messroutine wird das XY2-100-Statuswort als rückzuübertragender Datentyp automatisch eingestellt. Am Ende der Messroutine wird der zuvor eingestellte Datentyp wieder restauriert.
  - Wenn **auto_cal** für den primären Scan-Kopf-Anschluss ausgeführt werden soll, aber zuvor die automatische Lasersteuerung im Mode 2 aktiviert wurde (hierbei wird die Ist-Geschwindigkeit automatisch als rückzuübertragender Datentyp eingestellt), so bricht **auto_cal** mit einer Fehlermeldung (Rückgabewert 7) ab. Die automatische Lasersteuerung muss daher gegebenenfalls vor der automatischen Selbstkalibrierung deaktiviert werden.
  - **auto_cal** bricht auch ab (Rückgabewert 1, 10 oder 11), wenn der Skalierungsfaktor mit **control_command**(`Data` = $12xx_H$) zuvor auf einen Wert < 1 eingestellt wurde, weil dann die Sensorpositionen nicht angesteuert werden können (siehe auch **control_command**(`Data` = $053F_H$)). Der Skalierungsfaktor muss daher gegebenenfalls vor der automatischen Selbstkalibrierung mit **control_command**(`Data` = 1283_H) und **control_command**(`Data` = 1200_H) auf den Wert 1 (Standardeinstellung) eingestellt werden.

8.11.3 Ermittlung von Referenzwerten

Die Referenzwertbestimmung erfolgt mit **auto_cal**(`Command` = 0). Dadurch werden eine ASC-Hardware-Prüfung und die Messroutine zur Bestimmung der aktuellen Home-In-Positionen gestartet. Die ermittelten Referenzwerte werden – für die X- und Y-Achse separat – in der DLL gespeichert. Zusätzlich werden die Referenzwerte in einem nichtflüchtigen Speicher (EEPROM) auf der RTC5-Karte gespeichert. Dadurch stehen die Referenzwerte – auch nach einem Reset oder einer Initialisierung mit **load_program_file** – für eine spätere Kalibrierung zur Verfügung (letzteres gilt auch für den ermittelten ASC-Hardware-Typ).

Soll zu einem späteren Zeitpunkt eine automatische Selbstkalibrierung durchgeführt werden, dann sollte zuvor die Referenzwertbestimmung mit `Command` = 0 erfolgen, wodurch die Gain- und Offset-Werte der Galvanometerscanner zugleich automatisch auf ihre Initialisierungswerte (Gain = 1.0, Offset = 0) gesetzt werden.

Hinweise

- Nach **auto_cal**(`Command` = 0) stehen die Galvanometerscanner in der gleichen Position wie vor dem Befehl, jedoch gegebenenfalls um eine geänderte Offset/Gain-Einstellung korrigiert.
- Die Referenzwertbestimmung sollte beim Einrichten der Maschine erfolgen, z. B. wenn das Scan-System installiert bzw. getauscht wird und anschließend eine Bildfeldkalibrierung mittels Vermessen von Testmustern durchgeführt werden soll. Referenzwerte sollten auch dann neu ermittelt werden, wenn starke Änderungen in den Einsatzbedingungen ein neues Ausmessen des Bildfelds erzwingen.
- 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.

8.11.4 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 zum zuvor (mit `Command` = 0) gespeicherten Referenzwert werden von der RTC5 – für die X- und Y-Achse separat – Gain- und Offset-Werte berechnet und eingestellt. Die so eingestellte Drift-Kompensation wirkt sich unmittelbar auf die aktuelle und alle nachfolgenden Positionsausgaben aus. Sie ist so lange wirksam, bis **auto_cal**(`Command` = 1) erneut aufgerufen oder die Drift-Kompensation mit **auto_cal**(`Command` = 2) abgeschaltet wird (indem Gain = 1.0 und Offset = 0 gesetzt werden).

Hinweise

- Nach **auto_cal**(`Command` = 0, 1 und 2) stehen die Galvanometerscanner in derselben Position wie vor dem Befehl, jedoch gegebenenfalls gemäß der neuberechneten bzw. neugesetzten Drift-Kompensation korrigiert.
- Nach einem Reset oder einer Initialisierung (mit **load_program_file**) ist die automatische Drift-Kompensation – wie nach **auto_cal**(`Command` = 2) – abgeschaltet. Zuvor bestimmte Referenzwerte sind jedoch weiterhin verfügbar.
- Die mit **auto_cal**(`Command` = 1) gestartete Kalibrierungsroutine dauert (abhängig von der Stärke der Drift) typischerweise 1-2 Sekunden.
  Falls vor **auto_cal**(`Command` = 1) nicht zuvor **auto_cal**(`Command` = 0 oder 4) ausgeführt wurde, wird allerdings automatisch eine ASC-Hardware-Prüfung durchgeführt; dadurch verlängert sich die Ausführung des Befehls 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 solch eine kundeneigene Kalibrierung kann die mit **auto_cal**(`Command` = 0, 1 oder 3) ermittelte und in der DLL gespeicherte Home-In-Position mit **get_hi_pos** ausgelesen werden. Durch wiederholten Aufruf der Befehle **auto_cal**(`Command` = 3) und **get_hi_pos** kann die Drift ermittelt werden. Daraus können dann kompensierende Gain- und Offset-Werte berechnet und diese mit **set_hi** gesetzt werden. Die so eingestellte Drift-Kompensation wirkt sich unmittelbar auf die aktuelle und alle nachfolgenden Positionsausgaben aus.

Hinweise

- **get_hi_pos** gibt die zuletzt gemessenen Home-In-Positionen zurück. Unmittelbar nach **auto_cal**(`Command` = 0) sind dies auch die zuletzt gemessenen und gespeicherten Home-In-Referenzwerte. Unmittelbar nach Initialisierung (**init_rtc5_dll**) gibt **get_hi_pos** die Referenzwerte aus dem EEPROM zurück (siehe auch **get_hi_pos**-Befehlsbeschreibung).
- Nach **set_hi** wird gegebenenfalls eine Korrektur der aktuellen Position mit Sprunggeschwindigkeit ausgeführt.
- 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.
- Unter bestimmten Bedingungen können mit **write_hi_pos** die Home-In-Referenzpositionen (nicht die Home-In-Positionen) eines Scan-Kopfs von einer RTC5-Karte auf eine andere übertragen werden.

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.
- Tritt während der Messroutine zur Ermittlung der Home-In-Positionen – nach **auto_cal**(`Command` = 0, 1 oder 3) – ein Fehler auf (z.B. aufgrund einer erhöhten Streubreite bei einem Messzyklus, s.u.), werden für die betreffende Achse keine Referenzwerte gespeichert (`Command` = 0) und bleiben die Gain- und Offset-Werte unverändert (`Command` = 1; dagegen initialisiert `Command` = 0 Gain und Offset auf jeden Fall). **get_hi_pos** liefert dann 0 für die fehlerhaften Werte zurück.
- Die Messroutine ermittelt die Home-In-Positionen in einem Zyklus von mehreren Messungen. Wenn die Abweichungen zwischen den individuellen Messungen zu groß sind (Maximum – Minimum > 96 Bits), dann wird die Messroutine gestoppt und ein Fehler zurückgegeben (Rückgabewert 2). Ein Fehler bei einem einzelnen Messzyklus kann darauf hindeuten, dass noch keine stabilen Temperaturverhältnisse erreicht sind, kann aber auch 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- und Offset-Werte so lange weiter verwendet, bis bei einem nachfolgenden (erfolgreichen) Messzyklus neue Korrekturwerte ermittelt werden.

Treten größere Streubreiten über mehrere Messzyklen auf, muss die Anwendung u.U. so lange gestoppt werden, bis wieder einer der nachfolgenden Messzyklen erfolgreich ist.

Treten größere Streubreiten über viele Messzyklen und über längere Zeiten 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 im EEPROM 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.

8.12 Camming

Der **camming**-Befehl (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.

Die Galvanometerscannerbewegung 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 "Auflösung" der Kurve, also der Abstand zwischen zwei Punkten, ist frei wählbar – je feiner desto besser.

Allerdings muss die gesamte Kurve in einen zusammenhängenden Listenbereich passen, siehe **config_list**. Ein Listenwechsel mit Nachladen von weiteren Kurventeilen ist hier nicht möglich.

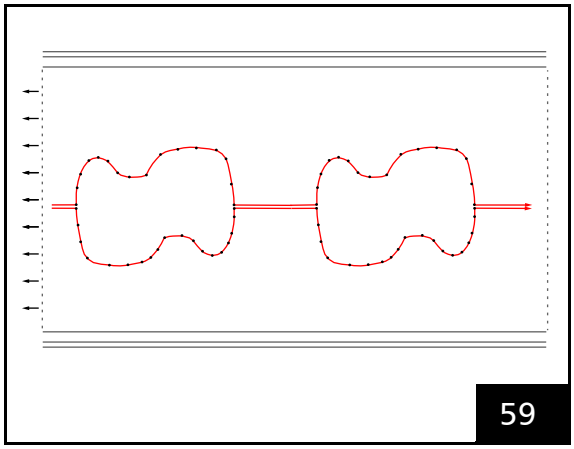
Der "Antrieb" wird durch extern eingespeiste oder intern simulierte Encoder-Pulse bewerkstelligt.

Der jeweils ausgegebene Punkt ergibt sich aus den XY-Koordinaten des **mark_abs**-Befehls, der sich an der Stelle `FirstPos + Index` befindet, mit `Index = Round((EncoderCurrent - EncoderStart) x Scale)`. `FirstPos` bedeutet die Listenposition des ersten **mark_abs**-Befehls. `Scale` ist ein beliebig wählbarer Skalierungsfaktor. `EncoderStart` wird beim Aufruf von **camming** automatisch ermittelt. Es gibt keinen automatischen Encoder-Reset. Der erste ausgegebene Punkt ist immer `Index = 0`.

Die einzelnen Punkte werden im 10 μs-Takt als harter Sprung ohne Scanner-Delay ausgeführt. **mark_abs** wird also nicht mikrovektorisiert, vgl. **micro_vector_abs**. `Scale` bestimmt dabei, wie präzise die Kurve abgetastet wird. Je größer `Scale` ist, desto grober wird die Kurve stückweise linear angenähert.

Die Anzahl der Encoder-Pulse pro 10 μs-Takt bestimmt zusammen mit dem Abstand der Punkte die eigentliche Markiergeschwindigkeit.

"Auflösung", `Scale` und Encoder-Geschwindigkeit sollten an die Eigenschaften des angeschlossenen Scan-Kopfs angepasst werden.



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 verschiedenste Art und Weise gesteuert werden, siehe **camming** Befehlsbeschreibung:

Benutzer können die Lasersteuerung selbst übernehmen (`Ctrl > 0`), z. B. mit **laser_signal_on_list** vor **camming** und **laser_signal_off_list** danach, oder den Laser von der RTC5 automatisch schalten lassen (`Ctrl = 0`) mit Berücksichtigung der Laser-Delays (wie bei einem normalen Polygonzug).

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` aus der Parameterliste des **camming**-Befehls legt die Länge der Punkteliste fest).

Je nach Encoder-Richtung kann die Punkteliste auch rückwärts durchlaufen werden. Dann beendet `Index = 0` die Kurve.

Die Kurve kann unbegrenzt wiederholt werden (`Ctrl = 2` oder `Ctrl = 3`). Um harte Sprünge beim Start der Wiederholung zu vermeiden, sollte die Punkteliste eine geschlossene Kurve darstellen. Zum Beenden der unbegrenzten Wiederholung ist ein Abbruch mit **stop_execution** oder /STOP erforderlich.

Die Kurve kann jederzeit mit einem externen /START bei der mit **set_extstartpos** bzw. **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).

Hinweis:

- Im normalen Betrieb wird ein externer /START während der Ausführung einer Liste unterbunden, beim Camming-Prozess mit unbegrenzter Wiederholung jedoch nicht!

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 zu `Scale` aus dem **camming**-Befehl gewählt wird).

Die Lasersteuerung kann mit der automatischen vektor-kontrollierten Lasersteuerung (**set_vector_control**) 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 automatische Vektor-kontrollierte Lasersteuerung nicht aktiviert ist, erfolgt keine Ausgabe der Laserleistung aus den **para_mark_abs**-Befehlen.

Zusätzlich ist eine Verknüpfung mit der encodergeschwindigkeitsabhängigen automatischen Lasersteuerung möglich.

Alternativ können auch die Befehle **mark_abs_3d** oder **para_mark_abs_3d**, sowie **timed_mark_abs_3d** verwendet werden. Jedoch werden die Z-Koordinate und der T-Parameter 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 75), kann die Form der Kurve auch mit einem normalen Listenstart z.B. durch `execute_at_pointer(FirstPos)` 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 mitgetestet werden soll, kann auch der Befehl **timed_mark_abs_3d** (mit `Time` = 10 μs) verwendet werden, nicht jedoch **timed_para_mark_abs_3d**.

8.13 Zeitmessungen

RTC5-Karten sind mit einem integrierten Timer ausgestattet, im Folgenden als “RTC5-Timer” bezeichnet. Der RTC5-Timer zählt nur Takte von Listenbefehlen. Die Zählung pausiert während durch **set_wait** oder **pause_list** initiierten Unterbrechungen.

Ab DLL 543, OUT 543, RBF 524 steht zusätzlich auch noch ein “Zeitstempel-Zähler” zur Verfügung. Er beginnt mit **load_program_file** bei 0 an zu zählen und zählt unterbrechungsfrei alle 10 μs-Takte.

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 RTC5-Timerwert und setzt ihn auf Null zurück. Die abgelaufene Zeit kann dann über **get_time** ausgelesen werden, der den RTC5-Timerwert zurückgibt, der beim letzten Aufruf von **save_and_restart_timer** gespeichert wurde.

Mit dem Zeitstempel-Zähler (verfügbar ab DLL 543, OUT 543, RBF 524) kann auch bei durch **save_and_restart_timer** zurückgesetztem RTC5-Timer ein absoluter Bezug zu den einzelnen Zeitabschnitten hergestellt werden. Der Zeitstempel-Zähler kann mit dem Triggersignal 52 (siehe **set_trigger** oder **set_trigger4**) aufgezeichnet und mit **get_value**`(52)` ausgelesen werden.

Mit **get_lap_time** kann der momentane RTC5-Timerwert ausgelesen werden. **get_lap_time** gibt die seit dem letzten Aufruf von **save_and_restart_timer** vergangene Zeit zurück, aber ohne den RTC5-Timer zurückzusetzen. Bei länger andauerenden Markierprozessen kann so die zwischenzeitliche Laufzeit überwacht werden.

Hinweise

- **get_time** und **get_lap_time** berücksichtigen nur Listenausführungszeiten (weil der RTC5-Timer nur Listenbefehlstakte zählt und während Unterbrechungen durch **set_wait** oder **pause_list** stehen bleibt).
- Falls die RTC5-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 gegebenenfalls 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).

9 Programmierung von Peripherie-Schnittstellen

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 RTC5 mit einer Reihe von Peripherie-Schnittstellen ausgestattet (siehe Seite 47).

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.

Dies wird in den folgenden Abschnitten näher erläutert.

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 OUT0 … 15) zur Verfügung, dessen Ausgangssignal-Pegel über einen Jumper einstellbar ist (siehe Seite 51).

Mit **write_io_port_list**, **write_io_port**, **write_io_port_mask_list** oder **write_io_port_mask** werden dem Ausgang 16-Bit-Digital-Werte zugewiesen. Vor der ersten Wertezuweisung befindet er sich im (hochohmigen) tristate-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 bei der positions- und/oder geschwindigkeitsabhängigen Lasersteuerung wirksam (siehe **set_port_default**).

Bei aktivierter automatischer Lasersteuerung wird der Wert am 16-Bit-Digital-Ausgang gegebenenfalls automatisch angepasst (siehe Seite 159). Dies kann mit **set_trigger**/**set_trigger4** aufgezeichnet werden.

Bei Änderung des Ausgangswertes 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 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 (DATA0 … DATA7) zur Verfügung (siehe Seite 52).

Seine Ausgangswerte können mit **write_8bit_port** oder **write_8bit_port_list** gesetzt werden. Vor der ersten Wertezuweisung befindet er sich im (hochohmigen) tristate-Zustand. Zusätzlich kann mit **set_port_default** (`Port` = 2) oder **set_laser_off_default** 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 bei der positions- und/oder geschwindigkeitsabhängigen Lasersteuerung wirksam (siehe **set_port_default**).

Bei aktivierter automatischer Lasersteuerung wird der Wert am 8-Bit-Digital-Ausgang gegebenenfalls automatisch angepasst (siehe Seite 159). Dies kann mit **set_trigger**/**set_trigger4** aufgezeichnet werden.

Bei Änderung des Ausgangswertes wird an der EXTENSION 2-Stiftleiste ein LATCH-Signal als Triggersignal zur Synchronisation der Datenübergabe mit ausgegeben, sofern Pin (17) durch entsprechende Jumperung dafür konfiguriert wurde (siehe Seite 52).

9.1.3 2 Bit-Digital-Ausgang

An der LASER-Buchse steht ein gepufferter 2-Bit-Digital-Ausgang (DIGITAL OUT1 und DIGITAL OUT2) zur Verfügung (siehe Seite 48). Mit **set_laser_pin_out** und **set_laser_pin_out_list** werden dem Ausgang Werte zugewiesen. 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-Ausgänge

An der LASER-Buchse stehen die beiden 12-Bit-Analog-Ausgänge ANALOG OUT1 und ANALOG OUT2 bereit (siehe Seite 48). Der Ausgang ANALOG OUT2 steht zusätzlich auch an der MARKING ON THE FLY-Stiftleiste zur Verfügung (siehe Seite 53).

Die Ausgangswerte der Analog-Ausgänge können unabhängig voneinander mit **write_da_x** oder **write_da_x_list** gesetzt werden. Zusätzlich kann mit dem **set_port_default** (`Port` = 0 oder 1) oder **set_laser_off_default** festgelegt werden, auf welche Werte die Ausgänge 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- und/oder geschwindigkeitsabhängigen Lasersteuerung wirksam (siehe **set_port_default**). Der Wert an den Analogausgängen wird – falls entsprechend eingestellt – auch durch einen Softstart (siehe "Softstart-Modus" auf Seite 152) oder im Pixelausgabemodus (siehe Seite 217) geändert.

Bei aktivierter automatischer Lasersteuerung wird der Wert an einem der Analogausgänge gegebenenfalls automatisch angepasst (siehe Seite 159). Dies kann mit **set_trigger**/**set_trigger4** aufgezeichnet werden.

9.1.5 Schrittmotoransteuerung

Ausgangssignale

Die Signale (ENABLE, DIRECTION und CLOCK) zur Ansteuerung von bis zu zwei Schrittmotoren werden an der Stiftleiste "STEPPER MOTOR" ausgegeben(1):

- 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 RTC5 (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 spezifiziert).
- 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 Seite 58.
- Zum Auslesen der Signale siehe Seite 236.
- Die Schrittmotorsignale werden unabhängig von einer etwaigen Listenausführung ausgegeben. Ein **set_end_of_list**, **pause_list**, **set_wait**, **stop_execution** und externe Listen-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.

(1) Schrittmotorsignale stehen nur für RTC5-Karten mit DSP-Versionsnummer 2 und größer (siehe **get_rtc_version** Bits #16-23) zur Verfügung. Bei älteren RTC5-Karten werden die Schrittmotorbefehle nicht ausgeführt.

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.

Bei erfolgreich abgeschlossener Referenzfahrt werden als Referenz für nachfolgende Zielpositionsfahrten sowohl die interne Positionsvariable (für die aktuelle Position) als auch der interne Zielpositionswert auf den über den Parameter `Pos` festgelegten Wert gesetzt. Die angefahrene Referenzposition ist gegenüber der Endschalterposition um ±`Tol` (richtungsabhängig) verschoben.

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 Positionsvariable und Zielpositionswert).

Da **stepper_init** auf jeden Fall eine zuvor gestartete Schrittmotor-Bewegung stoppt, kann der Befehl 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 simultan gestartet werden.

Zielpositionsfahrten

Zielpositionsfahrten werden mit den Befehlen **stepper_abs**, **stepper_rel**, **stepper_abs_no** bzw. **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 simultan.

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 Bewegung der Galvanometerscanner 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 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 dem Befehl **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.

Rücklesen von Signalen und Statuswerten

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** zurückgelesen 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**-Returnwert `RTC5_BUSY` zurückgewiesen und Listenbefehle warten, bis der Init-Status zurückgesetzt wird. Gegebenenfalls muss die Liste selbst oder der Bewegungsvorgang abgebrochen werden.

Anhalten von Endlosfahrten

Abhängig von den gewählten Parametern können mit **stepper_init**, **stepper_abs** etc. gegebenenfalls 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 gegebenenfalls 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 per **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 Listen-Stopp beendet werden. Die Endlosfahrt des Schrittmotors muss auch hier mit **stepper_control**(`Period` = 0) separat gestoppt werden.

9.1.6 RS232-Schnittstelle

An der Stiftleiste “RS232” wird eine RS232-Schnittstelle bereitgestellt (siehe Seite 54). Die Schnittstelle kann mit **rs232_config** konfiguriert werden. Mit **rs232_write_data** können einzelne Daten-Bytes, mit **rs232_write_text** oder **rs232_write_text_list** Texte an die RS232-Schnittstelle gesendet werden.

9.1.7 McBSP/SPI-Schnittstelle

An der “McBSP/SPI-Schnittstelle“, siehe Kapitel 4.4.6 ”SPI / I2C-Stiftleiste”, Seite 54, wird am DX0 Pin (7) ständig ein 32-Bit-Datenwort im 10 µs-Takt ausgegeben.

Mit **set_mcbsp_out** und **set_mcbsp_out_ptr** kann (analog zu **set_trigger**) eingestellt werden, welche Signaltypen dort ausgegeben werden:

- Mit dem Befehl **set_mcbsp_out** können zwei Signaltypen zur gleichzeitigen Ausgabe an der McBSP/SPI-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 dem Befehl **set_mcbsp_out_ptr** kann eine Liste von bis zu 8 Signaltypen definiert werden. Die Signalausgabe erfolgt zyklisch 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/SPI-Schnittstelle ausgegeben. Für eine genaue Beschreibung siehe **set_mcbsp_out_ptr**.

Hinweise

- Zu Signalen und Betriebsbedingungen siehe Kapitel 4.4.6 ”SPI / I2C-Stiftleiste”, Seite 54.
- In der Voreinstellung werden an der McBSP/SPI-Schnittstelle jeweils die Bits #4-19 der kartesischen Ansteuerwerte für die X- und Y-Achse (SampleX und SampleY) im gemeinsamen 32-Bit-Datenwort ausgegeben. Dies entspricht einer Auswahl gemäß `set_mcbsp_out(7, 8)`.
- Die mit **set_mcbsp_out** oder **set_mcbsp_out_ptr** ausgewählten Signale werden so lange (gegebenenfalls zyklisch) ausgegeben, bis einer der beiden Befehle erneut aufgerufen wird.

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 IN0 … 15) zur Verfügung (siehe Seite 51). 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 Seite 244).

9.2.2 2 Bit-Digital-Eingang

Zum Einlesen von 2-Bit-Digital-Werten steht an der LASER-Buchse ein 2-Bit-Digital-Eingang (DIGITAL IN1 und DIGITAL IN2) zur Verfügung (siehe Seite 48). Der Eingang kann mit dem Kontrollbefehl **get_laser_pin_in** ausgelesen werden.

Für eine bedingte Befehlsausführung stehen weitere Befehle zur Verfügung (siehe Seite 244).

9.2.3 RS232-Schnittstelle

An der Stiftleiste “RS232” können Signale über die RS232-Schnittstelle eingelesen werden (siehe Seite 54). Die Schnittstelle kann mit **rs232_config** konfiguriert werden. Mit **rs232_read_data** können Daten eingelesen werden.

9.2.4 McBSP/SPI Schnittstelle

An der McBSP/SPI-Schnittstelle, siehe Kapitel 4.4.6 “SPI / I2C-Stiftleiste”, Seite 54, kann mit dem Befehl **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 obliegt den Benutzern.

Die an der McBSP/SPI-Schnittstelle empfangenen Signale (Positionswerte) können aber auch direkt in die “Processing on the fly”-Korrektur einer Werkstück- oder Scan-System-Bewegung eingebunden (siehe Seite 199 und Seite 248) oder für eine Online-Positionierung verwendet werden (siehe Seite 187).

Hinweise

- Zu Signalen und Betriebsbedingungen siehe Kapitel 4.4.6 “SPI / I2C-Stiftleiste”, Seite 54.

9.3 Steuerung mit externen Signalen

Die oben beschriebene Eingabe und Ausgabe von Peripherie-Signalen kann wie folgt mit der Scan-System- und Lasersteuerung 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 Seite 76) und mit **get_status** der Listenausführungsstatus (siehe Seite 77) abgefragt werden. Zusätzlich wird über das BUSY OUT-Signal an der LASER-Buchse (Seite 47), an der EXTENSION 1-Stiftleiste (Seite 51) und an der MARKING ON THE FLY-Stiftleiste (Seite 53) der BUSY-Listenausführungsstatus bereitgestellt.

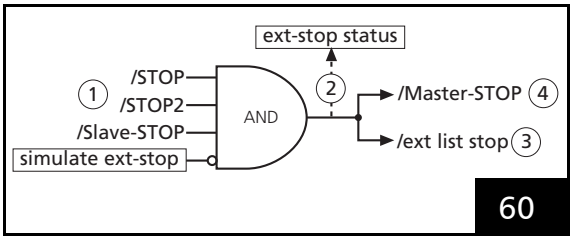
Darüber hinaus stellt die RTC5 die im Folgenden beschriebenen Befehle und Schnittstellen bereit, mit denen die Ausführung von Befehlslisten und die Ausführung einzelner Befehle (also auch die Ausgabe von Peripherie-Signalen) direkt mit externen Steuersignalen (z.B. den Signalen einer Lichtschranke oder eines Encoders) gesteuert und synchronisiert werden kann.

9.3.1 Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation

Externer Listen-Stopp

Über ein Signal an den Eingängen /STOP, /STOP2 oder /Slave-STOP oder über den Befehl **simulate_ext_stop** kann ein externer Listen-Stopp ausgelöst werden (siehe (1) und (3) in Abb. 60). Dadurch werden – wie auch nach Aufruf des Befehls **stop_execution** – die aktuell ausgeführte Liste sofort abgebrochen und die "Laser active"-Lasersteuersignale abgeschaltet (aber nicht deaktiviert).

Zusätzlich werden dann der 16-Bit-Digital-Ausgang an der EXTENSION 1-Stiftleiste, der 8-Bit-Digital-Ausgang (DATA0 … DATA7) an der EXTENSION 2-Stiftleiste und der 2-Bit-Digital-Ausgang sowie die beiden 12-Bit-Analog-Ausgänge (ANALOG OUT1 und ANALOG OUT2) an der LASER-Buchse auf die zuvor – mit **set_port_default** – für den Stopp-Fall festgelegten Werte gesetzt, sofern diese als ungleich "–1" definiert wurden.



Externer Listen-Stopp (Beschreibung siehe Text)

Die Eingänge für externe Stoppsignale (1) sind immer entriegelt, so dass ein externer Stopp zu *jedem* Zeitpunkt erfolgen kann.

Der /STOP-Eingang ist an der 15-poligen Sub-D-LASER-Buchse (siehe Seite 47), der /STOP2-Eingang an der MARKING ON THE FLY-Stiftleiste (siehe Seite 53) 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 (bzw. 0 V oder Massepotential) gesetzt wird.

Wenn die RTC5-Karte über den MASTER- oder SLAVE-Anschluss mit einer weiteren Karte verbunden ist, dann werden externe Listen-Stopps von Master zu Slave durchgereicht: Am MASTER-Ausgang (4) der Master-Karte wird gegebenenfalls ein Stoppsignal ausgegeben, das von der Slave-Karte am SLAVE-Eingang entgegengenommen wird (1), dort gegebenenfalls einen Listen-Stopp auslöst und zusätzlich wiederum am MASTER-Ausgang ausgegeben wird (4). Durch **stop_execution** ausgelöste Listen-Stopps werden *nicht* von Master zu Slave durchgereicht, wohl dagegen durch **simulate_ext_stop** ausgelöste externe Listen-Stopps.

Mit dem Befehl **get_startstop_info** kann der momentane Stopp-Status (d.h. ob einer der Eingänge momentan auf LOW gesetzt ist) ausgelesen werden (2) und ob ein externer Listen-Stopp erfolgt ist.

Ab Version DLL 528, OUT 530 kann mit **set_control_mode**, Bit#1 = 1 eingestellt werden, dass die Warteschlange der externen Start-Eingabe durch einen externen Listen-Stopp explizit gelöscht wird. Bei älteren Versionen und für **set_control_mode**, Bit #1 = 0 wird die Warteschlange nur disabled.

Externer Listen-Start

Über ein Signal an den Eingängen /START, /START2 oder /Slave-START oder über die Befehle **simulate_ext_start** oder **simulate_ext_start_ctrl** kann ein externer Listen-Start ausgelöst werden (siehe (1), (5) und (7) in Abb. 61). 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-Status (wie bei Listenausgabe) noch der INTERNAL-BUSY-Status (wie z. B. bei **goto_xy**) oder der PAUSED-Status (nach **pause_list**, **stop_list** oder **set_wait**) gesetzt ist.

Ehe die Eingänge /START, /START2 oder /Slave-START (1) verwendet werden können, müssen sie mit dem Befehl **set_control_mode** aktiviert werden (3).

Der /START-Eingang ist an der 15-poligen Sub-D-LASER-Buchse (siehe Seite 47), der /START2-Eingang an der MARKING ON THE FLY-Stiftleiste (siehe Seite 53) 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 Listen-Start wird – nach Aktivierung mit **set_control_mode** – ausgelöst, sobald eines der drei Eingangssignale von HIGH auf LOW (bzw. 0 V oder Massepotential) wechselt.

Wenn die RTC5-Karte über den MASTER- oder SLAVE-Anschluss mit einer weiteren Karte verbunden ist, dann werden externe Listen-Starts von Master zu Slave durchgereicht: Am MASTER-Ausgang (8) der Master-Karte wird gegebenenfalls ein Startsignal ausgegeben, das von der Slave-Karte am SLAVE-Eingang (1) entgegengenommen wird, dort gegebenenfalls einen Listen-Start auslöst und zusätzlich wiederum am MASTER-Ausgang ausgegeben wird (8).

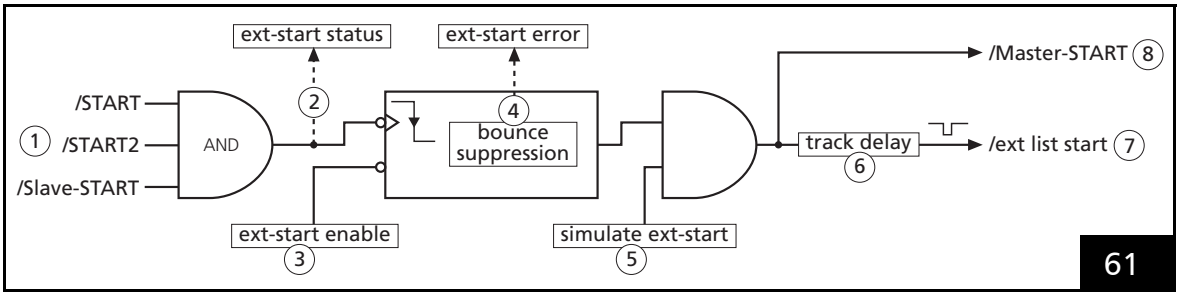
Durch **execute_list** oder **execute_at_pointer** ausgelöste Listen-Starts werden *nicht* von Master zu Slave durchgereicht, wohl dagegen durch **simulate_ext_start** oder **simulate_ext_start_ctrl** ausgelöste externe Listen-Starts (siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93).

Mit dem Befehl **get_startstop_info** kann der momentane Start-Status (d.h. ob einer der Eingänge momentan auf LOW gesetzt ist) ausgelesen werden (2) und ob eine Liste erfolgreich gestartet wurde.

Mit dem Befehl **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 wurde, kann mit dem Befehl **get_marking_info** ausgelesen werden (4).

simulate_ext_start_ctrl kann dazu verwendet werden, einen synchronen Listen-Start Master/Slave-synchronisierter Karten per Kontrollbefehl zu starten. Ab Version DLL 543, OUT 543 kann **simulate_ext_start_ctrl** allerdings auch deaktiviert sein (weil Bit #4 von **set_control_mode** oder **set_control_mode_list** gesetzt ist).

Der Listenbefehl **simulate_ext_start** kann u. a. dazu verwendet werden, nach einem einmalig erfolgten externen Listen-Start weitere Listen-Starts in definierten Abständen auszulösen (s.u.).



Externer Listen-Start (Beschreibung siehe Text)

Externer Listen-Start mit Streckenverzögerung

Bei vielen Anwendungen muss der Listen-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 den Befehlen **set_ext_start_delay**, **set_ext_start_delay_list** oder **simulate_ext_start** eine Streckenverzögerung (track delay) konfiguriert werden (siehe (6) in Abb. 61), so dass die Ausführung des Listen-Starts relativ zum auslösenden Eingangssignal bzw. Befehl entsprechend verzögert wird. Die Streckenverzögerung wird dabei in Zähleinheiten eines internen Zählers (Encoder-Zählers) angegeben, der wiederum durch ein externes oder simuliertes Encoder-Signal getriggert wird (siehe Seite 246).

Externe Listen-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 Fehlerbit 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**).

Hinweise

- Beachten Sie, dass die Eingänge /START, /START2 und /Slave-START *Flanken*-sensitiv sind (auf den *Übergang* von HIGH nach LOW), die Eingänge /STOP, /STOP2 und /Slave-STOP dagegen *Pegel*-sensitiv.
- Der Aufruf des Befehls **stop_execution** 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 von Benutzern mit dem Befehl **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** unterbunden (PAUSED-Status gesetzt). **restart_list**, **stop_execution**, **release_wait** oder ein externer Listen-Stopp heben diese Unterdrückung wieder auf.
- Falls die Listeneingabe noch nicht abgeschlossen ist, sollte vor einem externen Listen-Start z. B. mit `set_input_pointer( get_input_pointer() )` ein Puffer-Flush initiiert werden, damit gegebenenfalls gepufferte Listenbefehle vollständig in den Listenspeicher transferiert werden (siehe Seite 75).
- 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 per externem Startsignal (bzw. per **simulate_ext_start** oder **simulate_ext_start_ctrl**) gestartet werden oder wenn kein Home-Jump aktiviert ist.

Regelmäßige (periodische) externe Listen-Starts

Hinweise

- Für diese Funktionalität ist eine Firmware-Datei in der Version 507 oder größer erforderlich (siehe **get_rtc_version**), ferner eine DSP-Programmdatei in der Version 511 oder größer (siehe **get_hex_version**).

Mit **set_control_mode** bzw. **set_control_mode_list** (Bit #10) können äquidistante externe Listen-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 Listen-Starts – exakt auf den 10 µs-Takt der RTC5 synchronisiert.

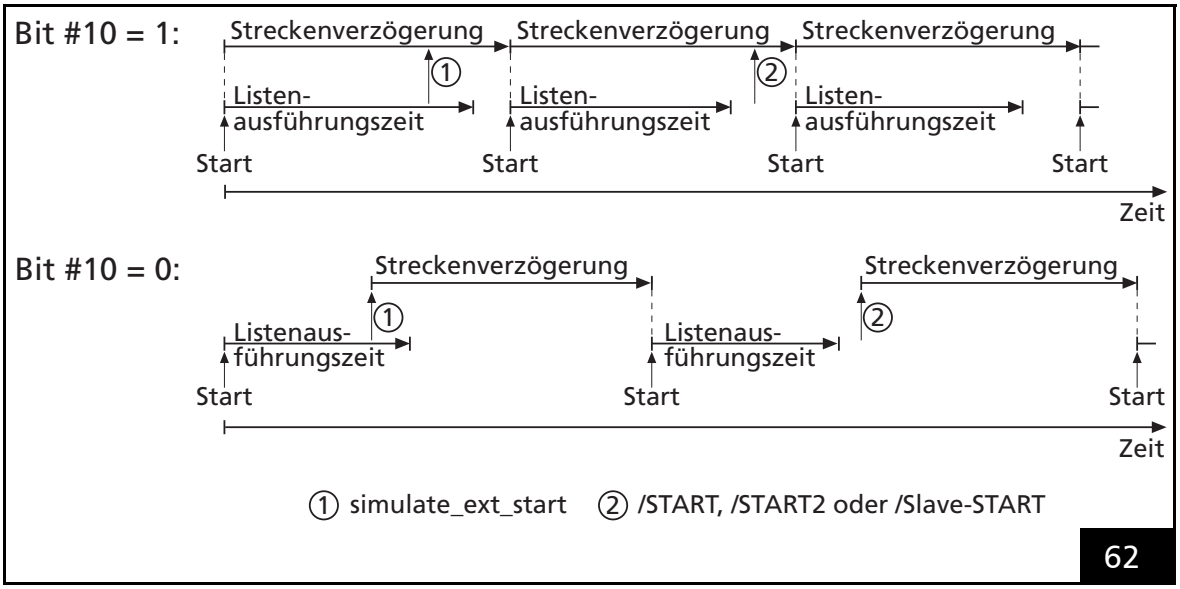
Durch Setzen von Bit #10 = 1 (`Mode|Bit#10`) kann dazu die Verarbeitung der internen Encoder-Zähler so eingestellt werden, dass die Streckenverzögerung eines externen Listen-Starts nicht erst ab dem Zeitpunkt des auslösenden externen Startsignals bzw. **simulate_ext_start**-Befehls gezählt wird (Bit #10 = 0), sondern bereits ab dem *zuletzt* (ebenfalls mit einem externen Startsignal oder mit **simulate_ext_start**) ausgeführten externen *Listen-Start* (siehe Abb. 62). Der Abstand aufeinander folgender externer Listen-Starts (in Encoder-Pulsen) wird dadurch konstant.

Um diesen Modus zu aktivieren, muss ein externer Listen-Start (einmalig) noch im Modus Bit #10 = 0 (`Mode &~Bit#10`) erfolgt sein. Jeder weitere externe Listen-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.



62

Regelmäßige und unregelmäßige externe Listen-Starts (Beschreibung siehe Text)

Ab Version DLL 528, OUT 530 kann mit **set_control_mode**, Bit#1 = 1 eingestellt werden, dass die Warteschlange der externen Start-Eingabe durch einen externen Listen-Stopp explizit gelöscht wird (externe Starts können dadurch auch per externem Listen-Stopp dauerhaft abgebrochen werden).

Bei älteren Versionen und für **set_control_mode**, Bit #1 = 0 (Standard-Einstellung) 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 Listen-Starts vernichtet werden. Andernfalls 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**-Befehle die Streckenverzögerung wieder aktivieren.

Wird ein erneuter externer Start innerhalb der Streckenverzögerung verpasst, muss die Warteschleife gelöscht werden (andernfalls benötigt der Encoder-Zähler 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). Andernfalls ist die erste Streckenverzögerung in der Warteschleife undefiniert.

9.3.2 Bedingte Befehlsausführung

Damit auch das Ausführen einzelner Listenbefehle von externen Steuersignalen abhängig gemacht werden kann, stellt der RTC5-Befehlssatz die sogenannten “konditionalen Befehle” zur Verfügung.

Diese Befehle lesen den aktuellen Wert am 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste (siehe Seite 51) aus und ihre Befehlsausführung ist abhängig vom ausgelesenen Wert:

- Bedingte Sprünge:
  Die Befehle **list_jump_pos_cond** (synonym zu **list_jump_cond**) und **list_jump_rel_cond** führen – abhängig vom ausgelesenen Wert – einen Sprung innerhalb eines Speicherbereichs aus oder nicht aus. Für die Sprungadressen gelten dieselben Bedingungen wie für die Befehle **list_jump_pos** und **list_jump_rel**.
- Sprung mit variabler Sprungweite:
  Der Befehl **switch_ioport** führt einen relativen Listensprung aus, dessen Sprungweite vom ausgelesenen Wert abhängt.
- Bedingte Aufrufe von nichtindizierten Unterprogrammen:
  Die Befehle **list_call_cond** und **list_call_abs_cond** rufen – abhängig vom ausgelesenen Wert – ein nichtindiziertes Unterprogramm am spezifizierten Speicherplatz auf oder nicht auf.
- Bedingte Aufrufe von indizierten Unterprogrammen:
  Die Befehle **sub_call_cond** und **sub_call_abs_cond** rufen – abhängig vom ausgelesenen Wert – ein indiziertes Unterprogramm mit dem spezifizierten Index auf oder nicht auf.
- Bedingte Ausgabe von Peripherie-Signalen:
  Die Befehle **set_io_cond_list** und **clear_io_cond_list** verknüpfen die Ausgabe am 16-Bit-Digital-Ausgang der EXTENSION 1-Stiftleiste (siehe Seite 51) direkt mit den Signalen am Digital-Eingang: Abhängig vom aktuellen Wert am Digital-Eingang werden einzelne Bits des Digital-Ausgangs gesetzt oder rückgesetzt.
- Bedingte Ausführung beliebiger Listenbefehle:
  Die Befehle **if_cond** bzw. **if_not_cond** bleiben wirkungslos, wenn die Bedingung für den ausgelesenen Wert erfüllt bzw. nicht erfüllt ist, ansonsten sorgen sie dafür, dass der unmittelbar nachfolgende Listenbefehl übersprungen wird. Auf diese Weise sind sämtliche Listenbefehle konditionierbar. Sie kommen dann nur bedingt zur Ausführung.
  Beispiel: Die Befehlsfolge

  ```
  if_cond(…)
  list_call(…)
  ```

  ist wirkungsgleich mit dem Befehl

  ```
  list_call_cond(…)
  ```

Die Ausführung beliebiger Listenbefehle kann auch vom aktuellen Wert am 2-Bit-Digital-Eingang der LASER-Buchse abhängig gemacht werden. Hierfür stehen die Befehle **if_pin_cond** und **if_not_pin_cond** zur Verfügung, die analog zu **if_cond** und **if_not_cond** funktionieren. Für ein sicheres Funktionieren dieser konditionalen Befehle sollten die Signale am 2-Bit-Digital-Eingang jeweils für mindestens 10 µs unverändert anliegen.

Ab Version DLL 543, OUT 543 kann mit dem Kontrollbefehl **set_pause_list_cond** eine Bedingung für den 16-Bit-Digital-Eingang der EXTENSION 1-Stiftleiste definiert werden. Ist diese Bedingung erfüllt, wird eine gerade ausgeführte Liste durch ein automatisch abgesetztes **pause_list** angehalten. Die Liste kann dann nur noch mit **restart_list** weiter ausgeführt werden. Die Bedingung wird einmal pro 10 *µs* überprüft. Ein gleichzeitig anliegendes /STOP-Signal hat Vorrang vor dem bedingten **pause_list**.

Programmierbeispiele

Die folgenden Programmierbeispiele sind in PASCAL geschrieben.

(1) Bestätige ein Signal:

```
set_start_list(1);
   ...
   set_io_cond_list(0, 0, 1);                // set bit #0 of the 16-bit digital output port
   list_jump_rel_cond(0, 1, 0);              // loop until the signal is confirmed
                                             // (i.e. bit #0 of the digital input turns HIGH)
   clear_io_cond_list(0, 0, 1);              // clear bit #0 of the 16-bit output
   list_jump_rel_cond(1, 0, 0);              // loop until the signal is confirmed
   ...
   set_end_of_list;
execute_list(1);
```

(2) Wenn die vier 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 RTC5 command uses relative addresses and is relocatable
   list_jump_rel_cond($0006, $0009, 3);      // skip the next two commands if the state
                                             // of the 16-bit input is (xxxx xxxx xxxx 0110)
   clear_io_cond_list(0, 0, 2);              // clear bit #1 of the 16-bit output and ..
   //RTC4 style: set_list_jump(get_input_pointer + 2);
   //this command uses absolute addresses and is not relocatable
   //the following RTC5 command uses relative addresses and is relocatable
   list_jump_rel(2);                         // .. skip the next command
   set_io_cond_list(0, 0, 2);                // set bit #1 of the 16-bit output
   ...                                       // (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
      list_call_cond(i, 15-i, i*100);        // call subroutine at address i*100
                                             // if [bit #3...bit #0] (binary) = i
   ...
```

(4) Wähle unter 15 indizierten Unterprogrammen aus:

```
   ...
   for i := 1 to 15 do
      sub_call_cond(i, 15-i, i);             // call subroutine with index i
                                             // if [bit #3...bit #0] (binary) = i
   ...
```

9.3.3 Synchronisation durch Encoder-Signale

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 RTC5 die Signale von bis zu zwei kundeneigenen Inkremental-Encodern eingelesen werden. Inkremental-Encoder registrieren allerdings in der Regel nicht (direkt) 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 RTC5, die dort zu einer Zählwert-Erhöhung bzw. -Reduktion zweier interner Encoder-Zähler führen. Der Zählerstand der RTC5-Encoder-Zähler entspricht dann direkt der Position des Werkstücks(2).

Falls das Werkstück immer bei einer konstanten Geschwindigkeit bearbeitet wird und daher ein Encoder überflüssig ist, können die Encoder-Signale auch mit **simulate_encoder** simuliert werden, so dass die beiden Encoder-Zähler auch ohne externe Encodersignale mit konstanter Zählrate hochgezählt werden. Die Encoder-Simulation kann natürlich auch für Laser-Scan-Prozesse, die unabhängig von einer Werkstückbewegung verlaufen, zum Einstellen einer konstanten Zählrate des Encoder-Zählers genutzt werden.

Die aktuellen Zählerstände der beiden Encoder-Zähler 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 RTC5 automatisch ausgewertet, wenn die Ausführung von Laser-Scan-Prozessen wie folgt gesteuert wird:

- Bei "Processing on the fly"-Anwendungen (siehe Seite 199) werden die Koordinatenwerte aller Vektor- und Bogenbefehle entsprechend der aktuellen Zählerstände der Encoder-Zähler (d.h. entsprechend der aktuellen Werkstückposition) transformiert.
- Mit dem Listenbefehl **wait_for_encoder_mode** kann die weitere Ausführung einer Liste so lange unterbrochen werden, bis der ausgewählte Encoder-Zähler (d.h. die Werkstückposition) einen vorgegebenen Wert über- oder unterschritten hat.
- Bei externen Listen-Starts kann mit den Befehlen **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** eine Streckenverzögerung zur Verzögerung der Ausführung des Listen-Starts relativ zum auslösenden Eingangssignal bzw. Befehl konfiguriert werden (siehe Seite 240).
- Bei der encodergeschwindigkeitsabhängigen automatischen Lasersteuerung (siehe Seite 168) kann ein Lasersignalparameter entsprechend der aktuell vorliegenden Encodergeschwindigkeit (Zählerpulse des letzten 10 µs-Intervalls) eines Encoder-Zählers gesteuert werden.

(1) Die *tatsächliche* Position des Werkstücks kann auch über die McBSP/SPI-Schnittstelle an die RTC5 übergeben werden (siehe Seite 248).

(2) Die Encoder-Zähler sind 32-Bit-Zähler (für 32-Bit-Werte 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 Encoder-Reset exakt mit externen Startsignalen synchronisiert und damit der 10 µs-Jitter (zufälliger Zeitversatz zwischen Startsignal und Listenstart) ausgeschaltet werden.

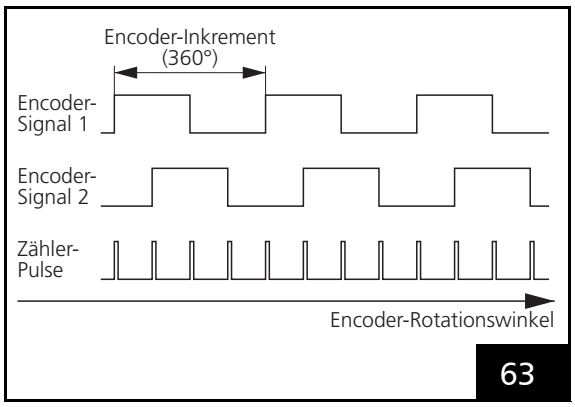
Eingänge für externe Encoder-Signale

Zur Eingabe von Encoder-Signalen stehen an der MARKING ON THE FLY-Stiftleiste (siehe Seite 53) zwei Encoder-Eingänge (ENCODER X und ENCODER Y) zur Verfügung.

Bei linearen Werkstück-Bewegungen kann man an diese Eingänge bis zu zwei kundeneigene Inkremental-Encoder anschließen, die unabhängig voneinander die Werkstück-Bewegung in x- und y-Richtung ermitteln.

Bei Rotationsbewegungen wird nur *ein* Inkremental-Encoder benötigt. Für "Processing on the fly"-Anwendungen muss dieser an den ENCODER X-Eingang angeschlossen werden.

Beide Encoder-Eingänge sind jeweils für ein Paar von standardisierten differentiellen Eingangssignalen (RS422) vorgesehen (HIGH-Pegel ≥2.0 V, LOW-Pegel ≤0.8 V).



Taktdiagram eines typischen Encoder-Signalpaares und der entsprechenden RTC5 Zähler-Pulse

Abb. 63 zeigt das Taktdiagramm eines typischen Encoder-Signalpaares. Das zweite Signal ist üblicherweise um 90° gegenüber dem ersten Signal phasenverschoben. Der RTC5-interne Encoder-Zähler wird durch jede Flanke beider Signale getriggert. *Ein* Encoder-Inkrement führt also zu *vier* Zählerpulsen (Zählschritten). Durch die relative 90°-Phasenverschiebung der beiden Signale kann die RTC5 nicht nur die Geschwindigkeit sondern auch die Bewegungsrichtung des Werkstücks einlesen. Abhängig von der Bewegungsrichtung, wird der jeweilige Zählerwert erhöht oder erniedrigt.

Die Signale am Encoder-Eingang ENCODER X triggern den Encoder-Zähler Encoder0, diejenigen am Encoder-Eingang ENCODER Y den Encoder-Zähler Encoder1.

Die Encoder-Signale sollten die maximal zulässige Frequenz von 4 MHz (bzw. 16 Encoder-Signalflanken je µs) nicht überschreiten.

Encoder-Simulation

Die Encoder-Simulation kann mit **simulate_encoder** aktiviert (und deaktiviert) werden. Die RTC5 stellt ein periodisches 1 MHz-Taktsignal bereit. Nach Aktivierung der Encoder-Simulation wird der ausgewählte Encoder-Zähler (oder werden beide Zähler) mit dieser Frequenz hochgezählt. Signale eines angeschlossenen Inkremental-Encoders werden dann ignoriert.

9.3.4 Synchronisation und Online-Positionierung mit McBSP/SPI-Signalen

Wird bei einem Laser-Scan-Prozess ein Werkstück unter dem (feststehenden) Scan-System oder das Scan-System (z. B. per Roboterarm) über dem (feststehenden) Werkstück bewegt, so muss der Laser-Scan-Prozess auf die momentane Position des Werkstücks relativ zum Scan-System abgestimmt werden ("Processing on the fly"). Bei zufällig bzw. ungenau positionierten und orientierten Werkstücken muss das Scan-System vor dem Bearbeitungsprozess exakt über dem Werkstück ausgerichtet werden (Online-Positionierung).

Die aktuelle Position des Werkstücks bzw. des Scan-Systems kann über die McBSP/SPI-Schnittstelle an die RTC5 übergeben werden. Der zuletzt vollständig übertragene Eingangswert an der McBSP/SPI-Schnittstelle kann mit **read_mcbsp** ausgelesen werden. Er wird von der RTC5 automatisch ausgewertet, wenn die Ausführung von Laser-Scan-Prozessen wie folgt gesteuert wird:

- Bei "Processing on the fly"-Anwendungen (siehe Seite 199) werden die Koordinatenwerte aller Vektor- und Bogenbefehle entsprechend des aktuellen Eingangswerts (d.h. entsprechend der aktuellen Position des Werkstücks bzw. des Scan-Systems) transformiert.
- Mit dem Listenbefehl **wait_for_mcbsp** kann die weitere Ausführung einer Liste so lange unterbrochen werden, bis der Eingangswert (d.h. die Position des Werkstücks bzw. des Scan-Systems) einen vorgegebenen Wert erreicht, über- oder unterschritten hat.
- Bei der Online-Positionierung (siehe Seite 187) wird das Scan-System entsprechend der aktuellen Eingangswerte ausgerichtet.

Hinweise

- Zur Kompensation linearer Bewegungen können kartesische Koordinatenwerte an der McBSP/SPI-Schnittstelle übergeben werden. Zur Kompensation von Rotationsbewegung müssen dagegen Winkelpositionen (d.h. – ähnlich wie bei der Synchronisation mit Encoder-Signalen, siehe Seite 246 – Werte proportional zur jeweiligen Winkelstellung) übergeben werden. Es spielt dabei keine Rolle, ob das zu übergebende Signal bei einem Überlauf (Winkel > 360°) wieder von vorne beginnt oder einfach weiter zählt.
- Die Signale an der McBSP/SPI-Schnittstelle haben keinen Einfluss auf die Streckenverzögerung, die für externe Listen-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 den Befehlen **periodic_toggle** und **periodic_toggle_list** können periodisch wiederholte Signale an einem auswählbaren IO-Port (ANALOG OUT1-Ausgang, siehe Seite 48, ANALOG OUT2-Ausgang, siehe Seite 48, 8-Bit-Digital-Ausgang, siehe Seite 52, 16-Bit-Digital-Ausgang, siehe Seite 51, 2-Bit-Digital-Ausgang, siehe Seite 48) ausgegeben werden.

Dadurch können z.B. externe Peripheriegeräte synchron zur Listenausführung getriggert werden.

Hinweise

- Bei einem **set_end_of_list**, **stop_execution** oder externem /STOP laufen die periodischen Signale weiter, selbst wenn sie über den Listenbefehl **periodic_toggle_list** aktiviert worden sind.
- Ab Version DLL 543, OUT 543 toggeln **periodic_toggle** und **periodic_toggle_list** bei `Count` = 2^{32}–1 endlos.

10 Befehle und Funktionen

10.1 Übersicht

Auf den nachfolgenden Seiten finden Sie eine vollständige Auflistung des RTC5-Befehlssatzes mit allen Kontroll- und Listenbefehlen, geordnet nach ihren vorgesehenen Verwendungszwecken. Die Seitenzahlen beziehen sich auf Kapitel 10.2 ab Seite 261, wo die Befehle (alphabetisch geordnet) im Detail beschrieben werden.

10.1.1 Nomenklatur

Multi-Board-Befehle

Alle Befehle, die in der nachfolgenden Auflistung mit einem (n_) versehen sind, existieren auch in einer Version zur gleichzeitigen Verwendung mehrerer in einem Computer installierten RTC5-Karten (siehe Kapitel 6.6, Seite 92).

Es wird für fast alle Befehle auch ein zugehöriger Multi-Board-Befehl bereitgestellt. Bei Befehlen, bei denen das nicht der Fall ist, wird in der Befehlsauflistung (in Kapitel 10.2) explizit darauf hingewiesen.

Normale, kurze, variable und multiple Listenbefehle

Die Listenbefehle des RTC5-Befehlssatzes unterscheiden sich teilweise in ihrer Befehlsausführungsdauer. Zur Unterscheidung werden die Listenbefehle in der Befehlsauflistung (in Kapitel 10.2) mit “normal”, “kurz”, “variabel” und ”multipel“ bezeichnet.

- Normale Listenbefehle benötigen einen vollen 10 µs-Takt zur Befehlsausführung.
- Kurze Listenbefehle benötigen weniger Zeit zur Befehlsausführung. Sie können daher zusammen mit dem jeweils nachfolgenden Listenbefehl unmittelbar nacheinander innerhalb eines 10 µs-Takts ausgeführt werden, allerdings ohne dass dazwischen ein Kontrollbefehl ausgeführt werden kann. Ein kurzer Listenbefehl, der einem normalen Listenbefehl unmittelbar nachfolgt, wird dagegen 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 Mark-Vektoren 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**-Befehl oder ein **list_continue**-Befehl 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”-Lasersignale werden ausgeschaltet), nicht dagegegen durch **list_continue**.
Derzeit sind bis zu 12 kurze Listenbefehle innerhalb eines 10 µs-Taktes möglich. Die maximale Anzahl wird jedoch in Abhängigkeit von der Belastung der RTC5-Karte und der DSP-Version reduziert(1). Kurze Listenbefehle, die den Output-Pointer verändern (z. B. **sub_call**, **list_return** oder **list_jump_pos**) zählen dabei als zwei Befehle. Bei Überschreiten der maximalen Anzahl wird ein 10 µs-Takt eingefügt (identisch zu einem zusätzlich eingefügten **list_continue**, der Laser bleibt im Polygonzug an)(1). Vor einem normalen Listenbefehl sind innerhalb desselben 10 µs-Taktes maximal zwei kurze Listenbefehle erlaubt. Folgt der normale Listenbefehl auf mehr als zwei kurze Listenbefehle, so werden die kurzen Listenbefehle sofort und der normale Listenbefehl um einen 10 µs-Takt verzögert ausgeführt.

(1) bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) sind nur bis zu maximal 8 kurze Listenbefehle je 10 µs-Takt möglich. Bei RTC5OUT.out Version 517 oder niedriger wird bei Überschreiten der maximalen Anzahl automatisch ein **list_nop** eingefügt (der Polygonzug wird unterbrochen).

Die maximale Anzahl von derzeit bis zu 12 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** oder ein **list_nop** eingefügt werden (**list_nop** unterbricht den Polygonzug).

- Bei variablen Listenbefehlen hängt die Befehlsausführungsdauer von weiteren Parametern oder vom Anwenderprogramm ab. Dies wird im Einzelnen bei der jeweiligen Befehlsbeschreibung erläutert.
- Zu multiplen Listenbefehlen: siehe Seite 252.

Hinweise

- Bei normalen und variablen Listenbefehlen setzt sich die Gesamtausführungsdauer gegebenenfalls 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.

Unverzögerte und verzögerte kurze Listenbefehle (ab Version OUT 515)

Die meisten kurzen Listenbefehle (wie **list_jump_pos**, **list_call**, **sub_call** oder **list_return**) werden (bei allen Software-Versionen) vor dem jeweils nachfolgenden Listenbefehl und noch vor einem eventuellen Scanner-Delay (Jump-, Mark- oder Polygon-Delay) ausgeführt. Diese Befehle werden in der jeweiligen Befehlsbeschreibung (in Kapitel 10.2) als "Unverzögerte kurze Listenbefehle" bezeichnet.

Einige kurze Listenbefehle werden jedoch (ab Version OUT 515) erst nach dem jeweiligen Scanner-Delay, also unmittelbar vor dem nachfolgenden Befehl ausgeführt. Diese Befehle werden in der jeweiligen Befehlsbeschreibung als "Verzögerte kurze Listenbefehle" bezeichnet. Zu diesen Befehlen 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**) oder Befehle, die sich auf die Aufzeichnung oder Zeitmessung auswirken (z. B. **set_trigger** oder **save_and_restart_timer**).

Bei älteren Software-Versionen würden diese (nun verzögerten) Befehle z. B. dazu führen, dass eine veränderte Laserleistung bereits am Ende eines Markierbefehls, also noch während des Mark- oder Polygon-Delays wirksam wäre und nicht erst zum Beginn des nachfolgenden Markierbefehls. **set_trigger** bzw. **save_and_restart_timer** würden bereits das Delay des vorangehenden Befehls mitberücksichtigen anstatt richtigerweise dasjenige des nachfolgenden Befehls.

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 (also wenn er am Ende der Befehlsfolge steht oder wenn ein "unverzögerter kurzer Listenbefehl" unmittelbar nachfolgt)[(1)].

(1) Dies gilt ab RTC5OUT.out Version 518. Bei Version 515-517 oder niedriger wird ein "verzögerter kurzer Listenbefehl" bei mehreren unmittelbar aufeinander folgenden kurzen Listenbefehlen nur dann verzögert ausgeführt, wenn er am Ende dieser Befehlsfolge steht.

Bei mehreren unmittelbar aufeinander folgenden "verzögerten kurzen Listenbefehlen" werden die Befehle am Anfang der Befehlsfolge unverzögert (also gegebenenfalls noch vor dem Scanner-Delay) ausgeführt. Werden also innerhalb eines Polygonzugs mehrere kurze Listenbefehle zwischen zwei Markierbefehle gesetzt, so sollten daher die wichtigsten "verzögerten" Befehle (z. B. Befehle, die die Laserleistung ändern) ans Ende (d.h. unmittelbar vor den nächsten Markierbefehl) oder unmittelbar vor einen "unverzögerten kurzen Listenbefehl" gesetzt werden. 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 zwei kurze Listenbefehle, so werden alle kurzen Listenbefehle (auch die verzögerten) sofort (unverzögert) ausgeführt und der normale Listenbefehl um einen 10 µs-Takt verzögert ausgeführt.

Multiple Listenbefehle

Ein multipler Listenbefehl besteht aus mehreren Teilen und benötigt entsprechend auch mehrere Listenspeicherplätze. Die vorderen Teile sind jeweils unverzögerte kurze Listenbefehle. Der letzte Teil ist jeweils ein kurzer, normaler oder variabler Listenbefehl. Alle Teile werden unmittelbar nacheinander ausgeführt. Eventuell anliegende verzögerte kurze Listenbefehle werden davor ausgeführt. Derzeit sind nur zweiteilige multiple Listenbefehle im Befehlssatz enthalten (z. B. **wait_for_encoder_in_range** oder **set_pixel_line_3d**).

10.1.2 Kompatibilität

Für RTC5-Benutzer, die vorher eine RTC4 verwendet haben, wird bei allen Befehlen in der Zeile "RTC4→ RTC5" der jeweiligen Befehlsbeschreibung (in Kapitel 10.2) darauf hingewiesen

- inwiefern sich ein Befehl gegenüber der RTC4 geändert hat,
- ob der Befehl dem Befehlssatz neu hinzugefügt wurde,
- ob und wie Parameterwerte im RTC4-Kompatibilitätsmodus umgerechnet werden.

Einige RTC3/RTC4-Befehle sowie einige RTC2-Befehle, die von der RTC3/RTC4 noch emuliert wurden, werden von der RTC5 nicht mehr unterstützt. Diese Befehle werden in Kapitel 10.3 ab Seite 642 aufgelistet.

Für weitere Hinweise vom Umstieg von der RTC4 auf die RTC5 und zum RTC4-Kompatibilitätsmodus siehe Kapitel 2.7, Seite 31.

10.1.3 Versionsinformationen

Beginnend mit der Version DLL 515, OUT 514, RBF 512 werden – sofern zutreffend – bei einer Reihe von Befehlen in der jeweiligen Befehlsbeschreibung unter “Versionsinfo” Versionsinformationen eingetragen:

- bei neu eingeführtem Befehl, ab welcher Software- bzw. DSP-Version dieser Befehl zur Verfügung steht, und
- bei älteren Befehlen gegebenenfalls Hinweise auf Befehlsänderungen.

Alle neuen Befehle und Änderungen an alten Befehlen werden auch in der im RTC5-Software-Paket enthaltenen Datei
`RTC5_Software_Aenderungshistorie_<Datum>.pdf`
aufgelistet.

10.1.4 Optionale Funktionen

Die RTC5 kann mit einigen Funktionalitäten ausgestattet werden, die in der Standard-Version nicht enthalten sind (siehe Seite 27). Sind die Funktionalitäten nicht von SCANLAB freigeschaltet bzw. installiert, so sind die zugehörigen Befehle nur eingeschränkt oder gar nicht wirksam. Befehle zur Aktivierung einer “Processing on the fly”-Korrektur bleiben beispielsweise bei nicht-freigeschalteter “Processing on the fly”-Funktionalität ohne Wirkung. 3D-Vektorbefehle werden bei nicht-freigeschalteter 3D-Funktionalität zwar als solche ausgeführt, es werden jedoch keine Z-Achsen-Signale ausgegeben. Ohne freigeschaltete Option “Zweite-Scan-Kopf-Steuerung” wirken sich auch Befehle, bei denen der Scan-Kopf-Ausgang explizit spezifiziert werden kann, nicht auf den sekundären Scan-Kopf-Anschluss aus (z. B. **set_matrix**).

Die Einzelbefehlsbeschreibungen enthalten gegebenenfalls entsprechende Hinweise.

10.1.5 Kontrollbefehle

Initialisierung der DLL

free_rtc5_dll 299
get_rtc_mode 318
init_rtc5_dll 348
set_rtc4_mode 541
set_rtc5_mode 541

Verwendung mehrerer RTC5-Karten in einem Rechner

acquire_rtc 261
release_rtc 448
rtc5_count_cards 453
select_rtc 460

Listenspeicher-Befehle

(n_) config_list 281
(n_) get_config_list 300
(n_) get_list_space 313
(n_) load_disk 379
(n_) save_disk 455

Initialisierung der Karte und Feldkorrektur

(n_) get_head_para (1) 306
(n_) get_sync_status (1) 326
(n_) get_table_para (1) 327
(n_) load_correction_file (1) 375
(n_) load_program_file (1) 389
(n_) load_stretch_table (2) 392
(n_) load_z_table (2) 396
read_abc_from_file 438
(n_) select_cor_table (1) 457
(n_) set_dsp_mode (1) 476
(n_) sync_slaves (1) 603
write_abc_to_file 635

Laser-Version und Laser-Parameter(1)

(n_) config_laser_signals 280
(n_) get_standby 320
(n_) load_auto_laser_control 373
(n_) load_position_control 387
(n_) set_auto_laser_control 464
(n_) set_auto_laser_params 467
(n_) set_encoder_speed_ctrl 479
(n_) set_firstpulse_killer 483
(n_) set_laser_control 504
(n_) set_laser_mode 508
(n_) set_laser_pulses_ctrl 511
(n_) set_pulse_picking 538
(n_) set_pulse_picking_length 538
(n_) set_qswitch_delay 539
(n_) set_softstart_level 553
(n_) set_softstart_mode 555
(n_) set_standby 557

Festlegen der Scannerparameter(1)

(n_) load_varpolydelay 395
(n_) set_delay_mode 475
(n_) set_jump_speed_ctrl 502
(n_) set_mark_speed_ctrl 514
(n_) set_sky_writing 546
(n_) set_sky_writing_limit 547
(n_) set_sky_writing_mode 548
(n_) set_sky_writing_para 550

Koordinatentransformationen(1)

(n_) set_angle 462
(n_) set_defocus (2) 473
(n_) set_matrix 515
(n_) set_offset 530
(n_) set_offset_xyz (2) 531
(n_) set_scale 542

Online-Positionierung(1)

(n_) apply_mcbsp 264
(n_) set_mcbsp_matrix 520
(n_) set_mcbsp_rot 523
(n_) set_mcbsp_x 524
(n_) set_mcbsp_y 525

Statusüberwachung und Diagnose(1)

(n_) get_head_status 307
(n_) get_overrun 318
(n_) get_value 332
(n_) get_values 334
(n_) get_waveform 336
(n_) measurement_status 417
(n_) stop_trigger 597

(1) siehe auch entsprechende Listenbefehle

(2) nur bei freigeschalteter 3D-Option

i*DRIVE*-Befehle

(n_) control_command ... 283
(n_) get_transform ... 329
(n_) read_user_data ... 447
(n_) send_user_data ... 461
transform ... 627
(n_) upload_transform ... 630

Pixelausgabemodus – Rasterbilder (Bitmaps) scannen

(n_) set_default_pixel ... 472

I/O-Befehle(1)

(n_) get_free_variable ... 304
(n_) get_io_status ... 310
(n_) get_mcbsp ... 317
(n_) mcbsp_init ... 415
(n_) mcbsp_init_spi ... 416
(n_) periodic_toggle ... 434
(n_) read_analog_in (2) ... 439
(n_) read_io_port ... 440
(n_) read_io_port_buffer ... 441
(n_) read_mcbsp ... 443
(n_) read_multi_mcbsp ... 444
(n_) rs232_config ... 450
(n_) rs232_read_data ... 451
(n_) rs232_write_data ... 452
(n_) rs232_write_text ... 452
(n_) set_free_variable ... 495
(n_) set_laser_off_default ... 508
(n_) set_mcbsp_freq ... 517
(n_) set_mcbsp_out_ptr ... 522
(n_) set_port_default ... 537
(n_) write_8bit_port ... 634
(n_) write_da_1 ... 636
(n_) write_da_2 ... 636
(n_) write_da_x ... 637
(n_) write_io_port ... 640
(n_) write_io_port_mask ... 641

Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation(1)

(n_) get_counts ... 301
(n_) get_master_slave ... 316
(n_) get_startstop_info ... 321
(n_) set_control_mode ... 470
(n_) set_extstartpos ... 481
(n_) set_max_counts ... 517
(n_) simulate_ext_start_ctrl ... 584
(n_) sync_slaves ... 603

Listen-Handling und -Status(1)

(n_) auto_change ... 273
(n_) auto_change_pos ... 274
(n_) get_lap_time ... 311
(n_) get_status ... 323
(n_) get_wait_status ... 335
(n_) pause_list ... 434
(n_) quit_loop ... 436
(n_) read_status ... 445
(n_) release_wait ... 449
(n_) restart_list ... 450
(n_) start_loop ... 585
(n_) stop_execution ... 596
(n_) stop_list ... 596

Input-Pointer-Befehl

(n_) get_input_pointer ... 310
(n_) get_list_pointer ... 312
(n_) load_list ... 385
(n_) set_input_pointer ... 497
(n_) set_start_list ... 558
(n_) set_start_list_1 ... 558
(n_) set_start_list_2 ... 558
(n_) set_start_list_pos ... 559

Output-Pointer-Befehl

(n_) execute_at_pointer ... 295
(n_) execute_list ... 295
(n_) execute_list_1 ... 295
(n_) execute_list_2 ... 295
(n_) execute_list_pos ... 296
(n_) get_out_pointer ... 318

(1) siehe auch entsprechende Listenbefehle

(2) nur bei RTC5 PCI-Express-Karte oder bei RTC5 PCI-Karte mit aufgesteckter ADC-Aufsteckplatine

Unterprogramm-Befehle(1)
(n_) copy_dst_src ... 293
(n_) get_char_pointer ... 300
(n_) get_sub_pointer ... 325
(n_) get_text_table_pointer ... 328
(n_) load_char ... 374
(n_) load_sub ... 393
(n_) load_text_table ... 394
(n_) set_char_pointer ... 468
(n_) set_char_table ... 469
(n_) set_sub_pointer ... 560
(n_) set_text_table_pointer ... 561

Direkte Laser- und Scan-Kopf-Steuerung(1)
(n_) disable_laser ... 294
(n_) enable_laser ... 294
(n_) get_laser_pin_in ... 311
(n_) get_z_distance (2) ... 337
(n_) goto_xy ... 338
(n_) goto_xyz (2) ... 339
(n_) laser_signal_off ... 356
(n_) laser_signal_on ... 357
(n_) set_laser_pin_out ... 509

Versionsbefehle
get_dll_version ... 301
(n_) get_hex_version ... 308
(n_) get_rtc_version ... 319
(n_) get_serial_number ... 320

Error-Befehle
(n_) get_error ... 302
(n_) get_last_error ... 312
(n_) reset_error ... 449
(n_) set_verify ... 571
verify_checksum ... 631

Datum, Uhrzeit, Seriennummern(1)
(n_) get_list_serial ... 313
(n_) get_serial ... 319
(n_) select_serial_set ... 461
(n_) set_serial ... 544
(n_) set_serial_step ... 545
(n_) time_update ... 606

"Processing on the fly"-Befehle(1)
(n_) get_encoder ... 301
(n_) get_fly_2d_offset ... 304
(n_) get_marking_info ... 314
(n_) init_fly_2d ... 347
(n_) load_fly_2d_table ... 381
(n_) read_encoder ... 440
(n_) set_ext_start_delay ... 482
(n_) set_fly_tracking_error ... 489
(n_) set_mcbsp_in (3) ... 518
(n_) set_multi_mcbsp_in (3) ... 526
(n_) set_rot_center ... 540
(n_) simulate_encoder ... 582
(n_) simulate_ext_stop ... 584

Schrittmotoransteuerung(1)
(n_) get_stepper_status ... 324
(n_) stepper_abs ... 586
(n_) stepper_abs_no ... 587
(n_) stepper_control ... 588
(n_) stepper_disable_switch ... 589
(n_) stepper_enable ... 590
(n_) stepper_init ... 591
(n_) stepper_rel ... 593
(n_) stepper_rel_no ... 594

Sprungmodus(1)
(n_) get_jump_table ... 311
(n_) load_jump_table ... 382
(n_) load_jump_table_offset ... 383
(n_) set_jump_mode ... 498
(n_) set_jump_table ... 503

Sonstige Kontrollbefehle
(n_) auto_cal ... 270
(n_) bounce_supp ... 275
(n_) get_auto_cal ... 299
(n_) get_galvo_controls ... 305
(n_) get_hi_data ... 309
(n_) get_hi_pos ... 309
(n_) get_time ... 328
(n_) home_position ... 340
(n_) home_position_xyz ... 341
(n_) move_to ... 422
(n_) set_hi ... 496
(n_) set_pause_list_cond ... 533
(n_) write_hi_pos ... 639

(1) siehe auch entsprechende Listenbefehle

(2) nur bei freigeschalteter 3D-Option

(3) ohne "Processing on the fly"-Option nur eingeschränkte Funktionalität

10.1.6 Listenbefehle

Initialisierung der Karte und Feldkorrektur

(n_) select_cor_table_list var 459

Jump-Befehle

(n_) jump_abs nor 350
(n_) jump_rel nor 352
(n_) para_jump_abs nor 422
(n_) para_jump_rel nor 424
(n_) timed_jump_abs nor 609
(n_) timed_jump_rel nor 611
(n_) timed_para_jump_abs nor 617
(n_) timed_para_jump_rel nor 620

3D-Jump-Befehle

(n_) jump_abs_3d nor (1) 351
(n_) jump_rel_3d nor (1) 353
(n_) para_jump_abs_3d nor (1) 423
(n_) para_jump_rel_3d nor (1) 425
(n_) timed_jump_abs_3d nor (1) 610
(n_) timed_jump_rel_3d nor (1) 612
(n_) timed_para_jump_abs_3d mul (1) 619
(n_) timed_para_jump_rel_3d mul (1) 621

Mikrovektor-Befehle

(n_) micro_vector_abs nor 418
(n_) micro_vector_abs_3d nor 419
(n_) micro_vector_rel nor 420
(n_) micro_vector_rel_3d nor 421

(1) nur bei freigeschalteter 3D-Option

nor normaler Listenbefehl

var variabler Listenbefehl

uk unverzögerter kurzer Listenbefehl

vk verzögerter kurzer Listenbefehl

mul multipler Listenbefehl

Markierbefehle

(n_) arc_abs nor 266
(n_) arc_rel nor 268
(n_) mark_abs nor 398
(n_) mark_ellipse_abs nor 405
(n_) mark_ellipse_rel nor 406
(n_) mark_rel nor 407
(n_) para_mark_abs nor 427
(n_) para_mark_rel nor 429
(n_) set_ellipse uk 477
(n_) timed_arc_abs nor 607
(n_) timed_arc_rel nor 608
(n_) timed_mark_abs nor 613
(n_) timed_mark_rel nor 615
(n_) timed_para_mark_abs nor 622
(n_) timed_para_mark_rel nor 625

3D-Markierbefehle

(n_) arc_abs_3d nor (1) 267
(n_) arc_rel_3d nor (1) 269
(n_) mark_abs_3d nor (1) 399
(n_) mark_rel_3d nor (1) 408
(n_) para_mark_abs_3d nor (1) 428
(n_) para_mark_rel_3d nor (1) 430
(n_) timed_mark_abs_3d nor (1) 614
(n_) timed_mark_rel_3d nor (1) 616
(n_) timed_para_mark_abs_3d mul (1) 624
(n_) timed_para_mark_rel_3d mul (1) 626

Textbefehle

(n_) mark_char uk 400
(n_) mark_char_abs uk 401
(n_) mark_text var 412
(n_) mark_text_abs var 413
(n_) select_char_set uk 456

Datum, Uhrzeit, Seriennummern

(n_) mark_date nor 402
(n_) mark_date_abs nor 404
(n_) mark_serial nor 409
(n_) mark_serial_abs nor 411
(n_) mark_time nor 413
(n_) mark_time_abs nor 415
(n_) select_serial_set_list uk 461
(n_) set_serial_step_list nor 546
(n_) time_fix nor 604
(n_) time_fix_f nor 604
(n_) time_fix_f_off nor 605

Statusüberwachung und Diagnose

(n_) set_trigger vk 562
(n_) set_trigger4 vk 568

Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation

(n_) set_control_mode_list nor 472
(n_) set_extstartpos_list uk 481

Listen-Handling und Strukturierte Programmierung

(n_) list_continue nor 363
(n_) list_jump_pos uk 365
(n_) list_jump_rel uk 367
(n_) list_next uk 369
(n_) list_nop nor 369
(n_) list_repeat uk 370
(n_) list_until uk 372
(n_) long_delay nor 397
(n_) set_end_of_list nor 480
(n_) set_list_jump uk 513
(n_) set_wait nor 572

Unterprogramm-Befehle

(n_) list_call uk 358
(n_) list_call_abs uk 359
(n_) list_call_abs_repeat uk 360
(n_) list_call_repeat uk 362
(n_) list_return uk 371
(n_) sub_call uk 598
(n_) sub_call_abs uk 599
(n_) sub_call_abs_repeat uk 600
(n_) sub_call_repeat uk 601

Festlegen der Laserparameter

(n_) config_laser_signals_list vk 280
(n_) set_auto_laser_params_list vk 467
(n_) set_encoder_speed vk 478
(n_) set_firstpulse_killer_list uk 483
(n_) set_laser_delays uk 507
(n_) set_laser_pin_out_list uk 509
(n_) set_laser_pulses vk 510
(n_) set_laser_timing vk 512
(n_) set_pulse_picking_list uk 539
(n_) set_qswitch_delay_list uk 539
(n_) set_softstart_level_list nor 554
(n_) set_softstart_mode_list var 556
(n_) set_standby_list vk 558
(n_) set_vector_control uk 569

Festlegen der Scannerparameter

(n_) set_delay_mode_list mul 502
(n_) set_jump_speed uk 502
(n_) set_mark_speed vk 514
(n_) set_scanner_delays vk 544
(n_) set_sky_writing_limit_list uk 547
(n_) set_sky_writing_list nor 547
(n_) set_sky_writing_mode_list nor 549
(n_) set_sky_writing_para_list nor 552

Koordinatentransformationen

(n_) set_angle_list var 463
(n_) set_defocus_list var (1) 474
(n_) set_matrix_list var 516
(n_) set_offset_list var 530
(n_) set_offset_xyz_list var (1) 532
(n_) set_scale_list var 543

Online-Positionierung

(n_) apply_mcbsp_list nor 265
(n_) set_mcbsp_matrix_list uk 521
(n_) set_mcbsp_rot_list uk 523
(n_) set_mcbsp_x_list uk 524
(n_) set_mcbsp_y_list uk 525

Direkte Laser- und Scan-Kopf-Steuerung

(n_) laser_on_list var 354
(n_) laser_on_pulses_list var 355
(n_) laser_signal_off_list nor 356
(n_) laser_signal_on_list nor 357
(n_) para_laser_on_pulses_list var 426

Pixelausgabemodus – Rasterbilder (Bitmaps) scannen

(n_) set_default_pixel_list uk 472
(n_) set_n_pixel var 529
(n_) set_pixel var 534
(n_) set_pixel_line nor 535
(n_) set_pixel_line_3d mul (1) 536

(1) nur bei freigeschalteter 3D-Option
nor normaler Listenbefehl
var variabler Listenbefehl
uk unverzögerter kurzer Listenbefehl
vk verzögerter kurzer Listenbefehl
mul multipler Listenbefehl

I/O-Befehle

(n_) clear_io_cond_list uk 279
(n_) periodic_toggle_list uk 435
(n_) read_io_port_list uk 442
(n_) rs232_write_text_list var 453
(n_) set_free_variable_list uk 495
(n_) set_io_cond_list uk 497
(n_) set_mcbsp_out uk 521
(n_) write_8bit_port_list vk 635
(n_) write_da_1_list vk 636
(n_) write_da_2_list vk 637
(n_) write_da_x_list vk 638
(n_) write_io_port_list vk 640
(n_) write_io_port_mask_list vk 641

Konditionierte Befehle

(n_) if_cond uk 342
(n_) if_not_cond uk 344
(n_) if_not_pin_cond uk 346
(n_) if_pin_cond uk 347
(n_) list_call_abs_cond uk 360
(n_) list_call_cond uk 361
(n_) list_jump_cond uk 364
(n_) list_jump_pos_cond uk 366
(n_) list_jump_rel_cond uk 368
(n_) sub_call_abs_cond uk 599
(n_) sub_call_cond uk 600
(n_) switch_ioport uk 602

"Processing on the fly"-Befehle

(n_) activate_fly_2d var (1) 262
(n_) activate_fly_xy var (1) 263
(n_) clear_fly_overflow uk 278
(n_) fly_return nor 297
(n_) fly_return_z nor 298
(n_) if_fly_x_overflow uk 342
(n_) if_fly_y_overflow uk 343
(n_) if_fly_z_overflow uk 343
(n_) if_not_activated uk 344
(n_) if_not_fly_x_overflow uk 345
(n_) if_not_fly_y_overflow uk 345
(n_) if_not_fly_z_overflow uk 346
(n_) park_position var (1) 431
(n_) park_return var (1) 432
(n_) set_ext_start_delay_list nor 483
(n_) set_fly_2d nor (1) 484
(n_) set_fly_limits uk 485
(n_) set_fly_limits_z uk 486
(n_) set_fly_rot nor (1) 487
(n_) set_fly_rot_pos nor (1) 488
(n_) set_fly_x nor (1) 490
(n_) set_fly_x_pos nor (1) 491
(n_) set_fly_y nor (1) 492
(n_) set_fly_y_pos nor (1) 493
(n_) set_fly_z nor (1) 494
(n_) set_mcbsp_in_list uk (2) 519
(n_) set_multi_mcbsp_in_list nor (2) 528
(n_) set_rot_center_list vk 540
(n_) simulate_ext_start nor 583
(n_) store_encoder uk 597
(n_) wait_for_encoder nor 631
(n_) wait_for_encoder_in_range mul 632
(n_) wait_for_encoder_mode nor 633
(n_) wait_for_mcbsp nor 634

Schrittmotoransteuerung

(n_) stepper_abs_list uk 587
(n_) stepper_abs_no_list uk 588
(n_) stepper_control_list uk 589
(n_) stepper_enable_list uk 590
(n_) stepper_rel_list uk 593
(n_) stepper_rel_no_list uk 594
(n_) stepper_wait nor 595

Sprungmodus

(n_) set_jump_mode_list nor 501

Sonstige Listenbefehle

(n_) camming nor 276
(n_) range_checking uk 436
(n_) save_and_restart_timer vk 454
(n_) set_wobbel vk 573
(n_) set_wobbel_control uk 575
(n_) set_wobbel_direction uk 576
(n_) set_wobbel_mode vk 577
(n_) set_wobbel_offset vk 579
(n_) set_wobbel_vector uk 580

(1) nur bei freigeschalteter "Processing on the fly"-Option

(2) ohne "Processing on the fly"-Option nur eingeschränkte Funktionalität

nor normaler Listenbefehl

var variabler Listenbefehl

uk unverzögerter kurzer Listenbefehl

vk verzögerter kurzer Listenbefehl

mul multipler Listenbefehl

10.1.7 Datentypen

Die folgende Tabelle definiert die Formate und die Wertebereiche der verschiedenen Datentypen, die in den RTC5-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
64-Bit IEEE Gleitkommaformat		double	double	double
Pointer (Zeiger) auf einen nullterminierten ANSI-String, 1 Byte pro Zeichen	4 Byte (für Win32-basierte Anwender-programme) 8 Byte (für Win64-basierte Anwender-programme)	pchar	char*	string

Pointer auf Arbeitsspeicherbereiche

Einige Befehle (wie **get_transform**, **get_values**, **get_waveform**, **transform** oder **upload_transform**) haben Pointer auf Arbeitsspeicherbereiche als Parameter. In C# und Pascal werden hierfür passende Zeigertypen verwendet (siehe Importdeklarationen). In C und C++ wird hierfür 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-basierte Anwender-programme, ULONG_PTR = 64-Bit-Wert ohne Vorzeichen für Win64-basierte Anwenderpro-gramme):

```
#if !defined(ULONG_PTR)
    #if defined(WIN32)
        #define ULONG_PTR UINT
    #else
        #define ULONG_PTR UINT64
    #endif
#endif
```

Üblicherweise ist der Datentyp ULONG_PTR auch entsprechend in der Windows-Kopfdatei <BaseTsd.h> definiert.

10.2 RTC5 Befehlssatz

Die Befehle sind alphabetisch geordnet.

Kontrollbefehl	**acquire_rtc**
Funktion	Reserviert die gewünschte RTC5-Karte für ein Anwenderprogramm.
Aufruf	`NoOfAcquiredCard = acquire_rtc( CardNo )`
Parameter	`CardNo` (DLL-interne) Nummer der gewünschten Karte als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Der Rückgabewert (32-Bit-Wert ohne Vorzeichen) ist `CardNo`, sofern die Reservierung erfolgreich war und 0, wenn die Karte bereits für ein anderes Anwenderprogramm reserviert ist oder die Versionsprüfung einen Fehler feststellt.
Hinweise	• Der Befehl **acquire_rtc** ist auch in einem Single-Board-System hilfreich, wenn die Verwendung der Karte durch unterschiedliche Anwenderprogramme koordiniert werden muss. • Der Befehl **acquire_rtc** bleibt wirkungslos, wenn `CardNo` die Zahl der bei der Initialisierung gefundenen RTC5-Karten (siehe **rtc5_count_cards**) übersteigt oder wenn `CardNo` = 0 ist (reale Karten zählen ab 1) (Rückgabewert 0, **get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Zugriffsberechtigungen auf die vorhandenen Karten werden exklusiv (gleichzeitig jeweils immer nur an *ein* Anwenderprogramm) vergeben. Der Befehl **acquire_rtc** bleibt deshalb auch wirkungslos, wenn die angeforderte Karte bereits durch eine anderes Anwenderprogramm reserviert ist (Rückgabewert 0, **get_last_error**-Returncode `RTC5_ACCESS_DENIED`). Bevor eine bereits reservierte Karte mit **acquire_rtc** beansprucht werden kann, muss die Karte vom zugriffsberechtigten Anwenderprogramm mit **release_rtc** oder **free_rtc5_dll** explizit freigegeben werden. Wenn die Karte andererseits bereits vor der Initialisierung eines Anwenderprogramms freigegeben wurde, so wird das Anwenderprogramm bei der Initialisierung mit **init_rtc5_dll** als für die Karte zugriffsberechtigt in die Kartenverwaltung aufgenommen. Ein explizites **acquire_rtc** ist in diesem Fall nicht erforderlich und bleibt dann auch wirkungslos. Rückgabewert ist dann aber trotzdem `CardNo`. • Die RTC5-DLL und die Programmdateien `RTC5OUT.out`, `RTC5RBF.rbf` und `RTC5DAT.dat` können nicht in beliebigen Programmversionen miteinander kombiniert werden. Der Befehl **acquire_rtc** führt eine entsprechende Versionsprüfung durch. Sind noch keine Programmdateien geladen, kann die Versionsprüfung zwar nicht explizit durchgeführt werden (**get_last_error**-Returncode `RTC5_TIMEOUT`), gilt aber dennoch als bestanden und verhindert damit die Übernahme nicht. Sind Programmdateien geladen und bei der Versionsprüfung wird ein Fehler festgestellt, dann wird der Zugriff verweigert (Rückgabewert 0, **get_last_error**-Returncode `RTC5_ACCESS_DENIED`|`RTC5_VERSION_MISMATCH`). • Eine mit **acquire_rtc** erfolgreich angeforderte Karte wird nicht automatisch zur aktiven Karte. Die Aktivierung geschieht nur mit den Befehlen **select_rtc** oder **init_rtc5_dll**. • Laufende Karten werden durch **acquire_rtc** nicht angehalten und insbesondere auch nicht initialisiert. • Der Befehl **acquire_rtc** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **acquire_rtc** ist nicht als Multi-Board-Befehl verfügbar. • Siehe auch Kapitel 6.7.1 "Hinweise zur Übernahme einer Karte durch ein Anwenderprogramm", Seite 96.

Kontrollbefehl	acquire_rtc
RTC4→ RTC5	Neuer Befehl.
Verweise	init_rtc5_dll, select_rtc, free_rtc5_dll, release_rtc, rtc5_count_cards

Variabler Listenbefehl	activate_fly_2d
Funktion	Aktiviert eine set_fly_2d-“Processing on the fly”-Anwendung ohne Encoder-Reset.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_2d( ScaleX, ScaleY )`
Parameter	`ScaleX, ScaleY` Skalierungsfaktoren wie bei set_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 8.7.4, Seite 206). Im Gegensatz zu set_fly_2d setzt dabei **activate_fly_2d** die Encoder nicht zurück, sondern rechnet die letzten nicht-“Processing on the fly”-korrigierten Koordinatenwerte so um, dass die “Processing on the fly”-korrigierte Ausgabe mit der aktuellen Ausgabe übereinstimmt. Falls die dann aktuell umgerechneten Koordinatenwerte außerhalb des virtuellen 24-Bit-Bildfeldes zu liegen kommen würden, wird “Processing on the fly” nicht aktiviert. Dann wird ebenfalls ein Fehlerbit gesetzt, welches mit get_marking_info (Bit #9 = 1) ausgelesen werden kann. • Bei *aktiver* “Processing on the fly”-Korrektur ist **activate_fly_2d** ein kurzer Listenbefehl ohne weitere Auswirkungen und setzt lediglich ein Fehlerbit, welches mit get_marking_info (Bit #9 = 1) 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 Fehlerbit kann auch mit dem kurzen Listenbefehl if_not_activated abgefragt werden, um ggf. in eine geeignete Fehlerbehandlungsroutine zu verzweigen. • Eine erfolgreiche Aktivierung mit **activate_fly_2d** löscht ein evtl. bereits zuvor gesetztes Fehlerbit *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 `RTC5_PARAM_ERROR`). • **activate_fly_2d** lässt die Lasersignale unangetastet (“Laser active”-Lasersteuersignale bleiben an/aus, wenn sie an/aus sind).
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe set_fly_2d.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	set_fly_2d

Variabler Listenbefehl	**activate_fly_xy**
Funktion	Aktiviert eine **set_fly_x**/**set_fly_y**-"Processing on the fly"-Anwendung ohne Encoder-Reset.
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, schaltet der Befehl den "Processing on the fly"-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`activate_fly_xy( ScaleX, ScaleY )`
Parameter	`ScaleX, ScaleY` Skalierungsfaktoren wie bei **set_fly_x**/**set_fly_y**.
Hinweise	• Wenn keine "Processing on the fly"-Korrektur aktiv ist, aktiviert **activate_fly_xy** eine **set_fly_x**/**set_fly_y**-"Processing on the fly"-Korrektur. Im Gegensatz zu **set_fly_x** und **set_fly_y** setzt dabei **activate_fly_xy** die Encoder nicht zurück, sondern rechnet die letzten nicht-"Processing on the fly"-korrigierten Koordinatenwerte so um, dass die "Processing on the fly"-korrigierte Ausgabe mit der aktuellen Ausgabe übereinstimmt. Falls die dann aktuell umgerechneten Koordinatenwerte außerhalb des virtuellen 24-Bit-Bildfeldes zu liegen kommen würden, wird "Processing on the fly" nicht aktiviert. Dann wird ebenfalls ein Fehlerbit gesetzt, welches mit **get_marking_info** (Bit #9 = 1) ausgelesen werden kann. • Bei *aktiver* "Processing on the fly"-Korrektur ist **activate_fly_xy** ein kurzer Listenbefehl ohne weitere Auswirkungen und setzt lediglich ein Fehlerbit, welches mit **get_marking_info** (Bit #9 = 1) ausgelesen werden kann. **activate_fly_xy** kann damit auch nicht dafür verwendet werden, den "Processing on the fly"-Modus oder die Skalierungsfaktoren beim selben Modus zu ändern. • Das Fehlerbit kann auch mit dem kurzen Listenbefehl **if_not_activated** abgefragt werden, um ggf. in eine geeignete Fehlerbehandlungsroutine zu verzweigen. • Eine erfolgreiche Aktivierung mit **activate_fly_xy** löscht ein evtl. bereits zuvor gesetztes Fehlerbit *nicht*. Es bleibt für **get_marking_info** stehen, bis **get_marking_info** aufgerufen wurde. • Für unzulässige Parameterwerte (z.B. `ScaleX` = 0) wird **activate_fly_xy** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • **activate_fly_xy** lässt die Lasersignale unangetastet ("Laser active"-Lasersteuersignale bleiben an/aus, wenn sie an/aus sind).
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_fly_x**/**set_fly_y**.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**set_fly_x**, **set_fly_y**

Kontrollbefehl	apply_mcbsp
Funktion	Ruft die zuletzt für die Online-Positionierung vollständig über die McBSP/SPI-Schnittstelle übertragenen Werte ab und definiert gegebenenfalls 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 *primären* Scan-Kopf-Anschluss. = 2: Die Definition wirkt nur auf den *sekundären* Scan-Kopf-Anschluss (Freischaltung erforderlich). = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`at_once` Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird. = 0: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten Listenbefehls berechnet und auf die dann aktuelle Position angewandt. = 1: Die neue Gesamttransformation wird sofort berechnet und auf die aktuelle Position angewandt (oder vor dem nächsten Listenbefehl, sofern gerade eine Liste BUSY oder die Karte INTERNAL-BUSY ist). = 2: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten **jump_abs**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls berechnet und auf die dann aktuelle Position angewandt. > 2: Wie `at_once` = 2.
Hinweise	• Das Einlesen der Daten zur Online-Positionierung über die McBSP/SPI-Schnittstelle muss zuvor mit den Befehlen **set_mcbsp_x**, **set_mcbsp_y** und/oder **set_mcbsp_rot** oder **set_mcbsp_matrix** (bzw. mit den entsprechenden Listenbefehlen) aktiviert werden. Je nach Konfigurierung definiert **apply_mcbsp** nur (wie mit **set_offset**) einen X- und/oder Y-Offset oder auch (wie mit **set_angle**) eine Drehmatrix oder (wie mit **set_matrix**) eine allgemeine Matrixoperation (siehe Kapitel 8.3 "Online-Positionierung", Seite 187). Wie bei den in Kapitel 8.2 "Koordinatentransformationen", Seite 183 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 gegebenenfalls durch **apply_mcbsp** überschrieben. Zuvor mit **set_scale** bzw. **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/SPI-Schnittstelle definierten Koordinatentransformationen. • Das jeweils erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP/SPI-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen (siehe Seite 56).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
Verweise	**apply_mcbsp_list**, **set_mcbsp_x**, **set_mcbsp_y**, **set_mcbsp_rot**, **set_mcbsp_matrix**

Normaler Listenbefehl	**apply_mcbsp_list**
Funktion	Identisch mit **apply_mcbsp** (s. o.), aber ein Listenbefehl.
Aufruf	`apply_mcbsp_list( HeadNo, at_once )`
Parameter	`HeadNo` Siehe **apply_mcbsp**.
	`at_once` Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird. = 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**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewandt. > 2: Wie `at_once` = 2.
Hinweise	• Siehe **apply_mcbsp**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
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 zweidimensionalen Bildfeld.
Aufruf	`arc_abs( X, Y, Angle )`
Parameter	`X, Y` Absolute Koordinaten des Bogenzentrums in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. `Angle` Bogenwinkel in ° im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet "im Uhrzeigersinn". Zulässiger Wertebereich: [–3600.0° … +3600.0°] (±10 Vollkreise). Überlaufende Werte werden auf den Randbereich geclippt.
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 pro ms* ausgeführt. • Die "Laser active"-Lasersteuersignale werden automatisch am Anfang der Markierung eingeschaltet (bzw. bleiben nach einem unmittelbar vorhergehenden Mark- oder Bogenbefehl eingeschaltet). Die eingestellten Scanner- und Laser-Delays werden dabei berücksichtigt (siehe Kapitel 7.2 "Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung", Seite 111). Beachten Sie, dass im Sky-Writing-Modus (siehe Seite 127) andere Delays ausgeführt werden. Ausnahme: Bogenbefehl der Bogenlänge 0 (siehe Hinweise Seite 115). • Falls eine Koordinatentransformation definiert wurde (siehe **set_matrix**, **set_offset**, **set_scale**, **set_angle**), wird sie nach der Mikrovektorisierung auf die aktuellen Koordinaten angewandt. • Entsprechend der mit **select_cor_table** oder **select_cor_table_list** zugewiesenen Korrekturtabelle wird eine Bildfeldkorrektur durchgeführt. • Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (Bildfeldgröße) geclippt.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die für das Bogenzentrum angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–524.288 … 524.287]).
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 spiralförmig 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 Spiralachse und der z-Endwert werden als absolute Koordinatenwerte angegeben.	
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **arc_abs**.	
Aufruf	`arc_abs_3d( X, Y, Z, Angle )`	
Parameter	`X, Y`	Lage der Spiralachse (parallel zur z-Achse) als absolute Koordinaten in *Bits* (32-Bit-Werte mit Vorzeichen). Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`Z`	Absolute z-Endkoordinate in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–32.768 … 32.767]. Ein überlaufender Wert wird auf den Randbereich geclippt.
	`Angle`	Bogenwinkel in ° im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet “im Uhrzeigersinn”. Zulässiger Wertebereich: [–3600.0° … +3600.0°] (±10 Vollkreise). Ein überlaufender Wert wird auf den Randbereich geclippt.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **arc_abs**-Befehl (siehe Hinweise dort). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bzgl. des angegebenen Wertes für die Markiergeschwindigkeit gleich behandelt werden. • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Seite 141). • Die z-Bewegung wird für die Berechnung der Anzahl der Mikroschritte nicht berücksichtigt.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die für `X` und `Y` angegebenen Werte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für `Z` ist bei RTC5 und RTC4 identisch.	
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.	
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 zweidimensionalen Bildfeld.
Aufruf	`arc_rel( dX, dY, Angle )`
Parameter	`dX, dY` *Relativ*koordinaten des Bogenzentrums in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt.
	`Angle` Bogenwinkel in ° im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet "im Uhrzeigersinn". Zulässiger Wertebereich: [–3600.0° … +3600.0°] (±10 Vollkreise). Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Die Koordinaten des Bogenzentrums werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **arc_abs** (siehe Hinweise dort). • Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (Bildfeldgröße) geclippt.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die für das Bogenzentrum angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–524.288 … 524.287]).
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 spiralförmig 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 Spiralachse und der z-Endwert werden als relative Koordinatenwerte angegeben.	
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **arc_rel**.	
Aufruf	`arc_rel_3d( dX, dY, dZ, Angle )`	
Parameter	`dX, dY`	Lage der Spiralachse (parallel zur z-Achse) als Relativkoordinaten in *Bits* (32-Bit-Werte mit Vorzeichen). Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`dZ`	Relative z-Endkoordinate in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–32.768 … 32.767]. Ein überlaufender Wert wird auf den Randbereich geclippt.
	`Angle`	Bogenwinkel in ° im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet “im Uhrzeigersinn”. Zulässiger Wertebereich: [–3600.0° … +3600.0°] (±10 Vollkreise). Ein überlaufender Wert wird auf den Randbereich geclippt.
Hinweise	• Die Lage der Spiralachse (`dX, dY`) und der z-Endwert (`dZ`) werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **arc_abs_3d** (siehe Hinweise dort).	
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **arc_abs_3d**.	
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.	
Verweise	**arc_abs_3d**, **arc_rel**	

Kontrollbefehl	auto_cal	
Funktion	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: Primärer Scan-Kopf-Anschluss. = 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich).
	`Command`	Kontrollparameter (als 32-Bit-Wert ohne Vorzeichen, zulässiger Wertebereich: [0 … 4]): = 0: Die RTC5 ermittelt die aktuellen Home-In-Positionen, speichert sie als Home-In-Referenzwerte in der DLL und im RTC5-EEPROM und initialisiert die Gain- und Offset-Werte (Gain = 1.0, Offset = 0). = 1: Die RTC5 ermittelt die aktuellen Home-In-Positionen, berechnet und setzt neue Gain- und Offset-Werte und aktiviert dadurch eine Drift-Kompensation. = 2: Die RTC5 deaktiviert die Drift-Kompensation durch Initialisierung der Gain- und Offset-Werte (Gain = 1.0, Offset = 0). = 3: Die RTC5 ermittelt die aktuellen Home-In-Positionen (lässt aber im Gegensatz zu `Command` = 1 die Gain- und Offset-Werte unverändert, aktiviert also auch keine Drift-Kompensation). = 4: Die RTC5 überprüft die ASC-Hardware (d.h. 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 bzw. den Status des Sensorsystems zurück. Der ermittelte Typ wird auch im RTC5-EEPROM gespeichert.
Rückgabe	Fehlercode bzw. Typ des Sensorsystems als 32-Bit-Wert ohne Vorzeichen: 3 Der Befehl kann nicht ausgeführt werden, weil die Karte gerade BUSY oder INTERNAL-BUSY 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 Achse X (Galvanometerscanner 2) 10: bei Achse Y (Galvanometerscanner 1) 11: bei beiden Achsen) 2, 20, 22 Streuung der Messwerte in einem Messzyklus zu hoch. (2: bei Achse X / 20: bei Achse Y / 22: bei beiden Achsen) 4, 40, 44 Referenzwerte nicht gefunden (nur bei `Command` = 1 und 3) (4: bei Achse X / 40: bei Achse Y / 44: bei beiden Achsen)	

Kontrollbefehl	auto_cal	
Rückgabe (Forts.)	5, 50, 55	Kalibrierungsfehler (Fehler während der Kalibrierung oder Fehler bei den Referenzwerten) (5: bei Achse X / 50: bei Achse Y / 55: bei beiden Achsen); diese Rückgabewerte können (bei Version DLL 519 oder geringer) auch bedeuten, dass kein(e) Home-In-Sensor(en) im Scan-System eingebaut sind.
	7	Fehler: automatische Lasersteuerung im Mode 2 aktiviert.
	9, 90	Sensor für Achse X oder Y 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	EEPROM-Schreibfehler (bei diesem Fehler wird in jedem Fall auch der **get_last_error**-Returncode `RTC5_EEPROM_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	Achse X (Galvanometerscanner 2): Sensorsystem vom Typ1 eingebaut, funktioniert. Achse Y (Galvanometerscanner 1): Sensorsystem defekt.
	29	Achse X (Galvanometerscanner 2): Sensorsystem vom Typ2 eingebaut, funktioniert. Achse Y (Galvanometerscanner 1): Sensorsystem defekt.
	91	Achse X (Galvanometerscanner 2): Sensorsystem defekt. Achse Y (Galvanometerscanner 1): Sensorsystem vom Typ1 eingebaut, funktioniert.
	92	Achse X (Galvanometerscanner 2): Sensorsystem defekt. Achse Y (Galvanometerscanner 1): Sensorsystem vom Typ2 eingebaut, funktioniert.
	99	Bei beiden Achsen: Sensorsystem defekt.
	255	Bei beiden Achsen: kein Sensorsystem eingebaut.
Hinweise	• Zur Verwendung des Befehls siehe Kapitel 8.11 "Automatische Selbstkalibrierung", Seite 224). • Die RTC5 fährt die Galvanometerscanner am Ende der Befehlsausführung immer wieder in die gleiche Stellung wie vor dem Befehlsaufruf, gegebenenfalls in eine um geänderte Gain- 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 Scanner. • Nach dem erstmaligen oder erneuten Anschließen eines Scan-Systems sollte eine Referenzwertbestimmung mit **auto_cal**(`Command` = 0) durchgeführt werden. Die RTC5 verwendet sonst gegebenenfalls bei einer späteren automatischen Selbstkalibrierung die gespeicherten Referenzwerte für ein zuvor betriebenes (evtl. anderes) Scan-System. • Nach der Initialisierung der RTC5 oder nach einem Reset ist die Drift-Kompensation abgeschaltet (Gain = 1.0, Offset = 0). Zuvor bestimmte Referenzwerte sind jedoch weiterhin verfügbar.		

Kontrollbefehl	auto_cal
Hinweise (Forts.)	• 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. • Für einen unzulässigen Wert von `HeadNo` oder `Command` wird **auto_cal** nicht ausgeführt (Rückgabewert 6, **get_last_error**-Returncode: `RTC5_PARAM_ERROR`). Dies gilt auch für `HeadNo` = 2, falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist (Rückgabewert 6). • Der Befehl **auto_cal** wird auch nicht ausgeführt (Rückgabewert 3, **get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). Bei einem `RTC5_PARAM_ERROR` wird der BUSY-Status nicht überprüft, d. h. Returncode `RTC5_BUSY` und `RTC5_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. • Ab Version DLL 520: 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 RTC5-EEPROM 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 RTC5-EEPROM 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 RTC5-EEPROM gelöscht werden. Der Fehler wird *nicht* durch den Aufruf von `auto_cal(command = 1)` aus dem EEPROM gelöscht.
RTC4→ RTC5	Die Funktionen zur automatischen Selbstkalibrierung (`Command` = 0 …2) sind (größtenteils) unverändert (siehe Versionsinfo). Neu: `Command` = 3 und `Command` = 4.
Versionsinfo	Änderung mit Version DLL 520, OUT 519 (siehe Seite 693). Letzte Änderung mit Version DLL 535: Rückgabewert 8 (s.o.).
Verweise	**set_hi**, **get_hi_pos**, **get_auto_cal**, **write_hi_pos**

Kontrollbefehl	auto_change
Funktion	Aktiviert einen einmaligen automatischen Listenwechsel.
Aufruf	`auto_change()`
Hinweise	• Der Befehl **auto_change** ist synonym zu `auto_change_pos( 0 )`. Die nachfolgende Liste wird damit an deren Listenanfang gestartet. • Siehe auch Hinweise unter **auto_change_pos**.
RTC4→ RTC5	Unveränderte Funktionalität.
Versionsinfo	Letzte Änderung Version OUT 515.
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 (Listenspeicheradresse) als Offset bzgl. des Listenanfangs der durch den automatischen Listenwechsel zu startenden Liste (als 32-Bit-Wert ohne Vorzeichen).
Hinweise	• Der **auto_change_pos**-Befehl löst nur einen *einmaligen* nachfolgenden Listenwechsel bzw. Listenneustart aus. Für weitere Listenwechsel oder Listenneustarts muss der Befehl erneut aufgerufen werden. • Der Befehl kann zu jedem beliebigen Zeitpunkt erteilt 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 "Automatischer Listenwechsel", Seite 79). • Damit der **auto_change_pos**-Befehl überhaupt wirkt, muss die evtl. gerade aktive Liste unbedingt mit **set_end_of_list** abgeschlossen worden sein, die neue Liste sollte bereits geladen sein bzw. der Input-Pointer einen gewissen Vorsprung vor dem Output-Pointer haben (andernfalls 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 Listenspeicheradresse außerhalb des jeweiligen Listenbereichs 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* der **start_loop**-Befehl 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. • Die aktuellen Listenstatuswerte und Listenausführungsstatuswerte können mit den Befehlen **read_status** und **get_status** gelesen werden. • Der **auto_change_pos**-Befehl löst einen Flush der gepufferten Listeneingabe aus (siehe Seite 75).
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Die Listenspeicheradresse (`Pos`) wird bei der RTC5 als relative Speicheradresse bzgl. des Listenanfangs der jeweiligen Liste, bei der RTC4 dagegen als absolute Speicheradresse (0…7999) angegeben. Wird mit **config_list** der "Liste 2" kein Speicherbereich zugewiesen (`Mem2` = 0), wirkt der RTC5-Befehl wie bei der RTC4.
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.
Hinweise	• Mit dem Befehl **bounce_supp** kann eine Entprellung für die über die Eingänge /START, /START2 oder /Slave-START eingespeisten Startsignale eingestellt werden (siehe Abschnitt "Externer Listen-Start", Seite 240). 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.
RTC4→ RTC5	Neuer Befehl.
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 Listenspeicher) als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [0 … (2^{20}–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^{20}–`FirstPos`)].
	`EncoderNo`	Nummer des Encoder-Zählers, dessen Pulse zur Steuerung des Camming-Prozesses ausgewertet werden, als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0. = 1: Encoder-Zähler Encoder1. Höherwertige Bits werden ignoriert.
	`Ctrl`	Camming-Steuerungsmodus als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: RTC5 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, der Ausgabe-Index wird auf [0, `NPos`–1] geclippt. = 3: Benutzer steuert Laser, **camming** endet *nicht* automatisch, die Camming-Liste wird kontinuierlich umlaufend (Ausgabe-Index modulo `NPos`) abgearbeitet.
	`Scale`	Übersetzung (Umrechnungsfaktor) zwischen Encoder-Zählwert und Befehls-Index im 64-Bit-IEEE-Gleitkommaformat. Zulässige Werte: $2^{-60} < \lvert$`Scale`$\rvert < 2^{60}$.
	`Code`	32-Bit-Wert ohne Vorzeichen. Wird nicht verwendet.
Hinweise	• Siehe auch Kapitel 8.12 "Camming", Seite 229. • Bei unzulässigen Parameterangaben wird **camming** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Bei jedem Aufruf von **camming** wird der aktuelle Encoder-Zählwert als neuer Referenzwert ermittelt. Zu einem späteren Zeitpunkt wird für den dann aktuellen Encoder-Zählwert (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 obliegt dann vollständig dem Benutzer.	

Normaler Listenbefehl	camming
	• Für `Ctrl` = 0 oder 1 endet der Befehl **camming** automatisch (im nächsten Zyklus), sobald der Index erstmalig 0 unter- bzw. `NPos`–1 überschritten hat (als letzter Camming-Befehl wird dann der Befehl mit Index 0 bzw. `NPos`–1 ausgeführt). In diesen beiden Modi sind (wie immer, wenn eine Liste BUSY ist) keine externen Listen-Starts erlaubt, solange der Befehl **camming** noch nicht beendet ist. • Die Steuerungsmodi `Ctrl` = 2 und 3 sind dagegen endlos. Der Camming-Prozess kann hier nur durch **stop_execution** bzw. durch einen externen Listen-Stopp beendet werden. In diesen Modi ist aber (ausnahmsweise) ein externer Listen-Start auch dann erlaubt, wenn die Liste (bzw. der Befehl **camming**) noch aktiv ist. Für `Ctrl` = 2 wird der Index auf die Grenzen 0 bzw. `NPos` – 1 geclippt, für `Ctrl` = 3 modulo `NPos` gesetzt (dabei wird vorausgesetzt, dass die Encoder-Geschwindigkeit nicht so groß ist, dass mehr als ein kompletter Umlauf übersprungen wird). `Ctrl` = 3 wirkt also wie ein Ringpuffer. • Der Befehl **camming** ist ein normaler Listenbefehl, aber variabel in der Ausführungsdauer. • **camming** funktioniert auch dann, wenn die “Processing on the fly”-Option nicht freigeschaltet ist. • Während der Ausführung von **camming** liefert **get_out_pointer** stets die Position des Befehls **camming**, nicht diejenige des aktuellen Indexes.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_encoder_speed**, **set_auto_laser_control**

Unverzögerter kurzer Listenbefehl	**clear_fly_overflow**
Funktion	Setzt die Fehlerbits (#4…7 von **get_marking_info**) für die kundenspezifische Überwachung von “Processing on the fly”-Anwendungen zurück.
Aufruf	`clear_fly_overflow( Mode )`
Parameter	`Mode` Dieser Parameter (32-Bit-Wert ohne Vorzeichen) gibt an, welche Fehlerbits zurückgesetzt werden sollen: • Bit#0 = 1: Fehlerbit#4 (Underflow X) wird zurückgesetzt. • Bit#1 = 1: Fehlerbit#5 (Overflow X) wird zurückgesetzt. • Bit#2 = 1: Fehlerbit#6 (Underflow Y) wird zurückgesetzt. • Bit#3 = 1: Fehlerbit#7 (Overflow Y) wird zurückgesetzt. • Bit#4 = 1: Fehlerbit#24 (Underflow Z) wird zurückgesetzt. • Bit#5 = 1: Fehlerbit#25 (Overflow Z) wird zurückgesetzt. Bits #0...5 können beliebig kombiniert werden. Höherwertige Bits werden ignoriert.
Hinweise	• Zur Verwendung des Befehls siehe Abschnitt “Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)”, Seite 214. • Für `Mode` = 0 werden wie bei `Mode` = 15 alle sechs Fehlerbits zurückgesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525, OUT 527. Letzte Änderung (Version DLL531, OUT532): Bits #4, 5.
Verweise	**get_marking_info**

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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask_0` gesetzten Bits von `IOvalue` gleich `0` sind).
Aufruf	`clear_io_cond_list( Mask1, Mask0, MaskClear )`
Parameter	`Mask1`, `Mask0`, `MaskClear` — 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
Hinweise	• Der Befehl 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 -Ausgang", Seite 51 und Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.

Beispiele (Pascal)	• Lösche Bit #4 des Digital-Ausgangs (DIGITAL OUT4), falls Bit #0 des Digital-Eingangs (DIGITAL IN0) gesetzt und die Bits #1 bis #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→ RTC5	Unveränderte Funktionalität.
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 Lasersignal-Typen, die an Pin 1 (LASER1), Pin 2 (LASERON) und Pin 9 (LASER2) der Laseranschluss-Buchse (LASER) ausgegeben werden.
Aufruf	`config_laser_signals( Config )`
Parameter	`Config` Gewünschte Signalkonfiguration als 32 Bit-Wert ohne Vorzeichen. Dabei konfigurieren: Bits # 0-1: Pin 2 (LASERON-Kanal) Bits # 2-3: Pin 1 (LASER1-Kanal) Bits # 4-5: Pin 9 (LASER2-Kanal) Bits # 6-31 werden ignoriert Dabei bedeutet: = 0 = 00_b: LASERON-Signal = 1 = 01_b: LASER1-Signal = 2 = 10_b: LASER2-Signal = 3 = 11_b: FirstPulseKiller-Signal Die Standard-Einstellung nach **load_program_file** ist: `Config` = 100100_b = 0x24 = 36
Hinweise	• Die spezifizierte Konfiguration wird wirksam, sobald der Laser das nächste Mal eingeschaltet wird. Der Befehl sollte also nicht während einer aktiven Markierung erteilt werden.
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 FirstPilseKiller-Signal verzögert eingeschaltet werden. • `Config` = 100100_b (Standard-Einstellung): 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 RTC5 nur dann erzeugt werden, wenn das LaserON-Signal an ist, werden hier LASER1-Signale erst frühestens mit dem Laser-Start ausgegeben).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 533, OUT 534, RBF 524.

Verzögerter kurzer Listenbefehl	config_laser_signals_list
Funktion	Identisch mit **config_laser_signals**, aber ein Listenbefehl.
Aufruf	`config_laser_signals_list( Config )`
Parameter	`Config` Siehe **config_laser_signals**.
Hinweise	• Siehe **config_laser_signals**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**config_laser_signals**

Kontrollbefehl	config_list	
Funktion	Konfiguriert den Listen-Speicher, d. h. weist den Listen Speicher-Bereiche zu.	
Aufruf	`config_list( Mem1, Mem2 )`	
Parameter	`Mem1`	Gewünschte Speichergröße für “Liste 1” (als 32-Bit-Wert ohne Vorzeichen).
	`Mem2`	Gewünschte Speichergröße für “Liste 2” (als 32-Bit-Wert ohne Vorzeichen).
Hinweise	• Der Listenspeicher der RTC5 umfasst 2^{20} Speicherplätze, die durch eine Konfiguration mit **config_list** auf drei Bereiche (Listen) aufgeteilt werden können. Die Größen von “Liste 1” und “Liste 2” werden mit den Parametern `Mem1` und `Mem2` vorgegeben. Dem geschützten Speicherbereich “Liste 3” werden durch **config_list** automatisch die restlichen, nicht an “Liste 1” oder “Liste 2” vergebenen Speicherplätze zugeteilt. • 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 in `Mem1` = 1 korrigiert. – `Mem1` ≤ 2^{20} (“Liste 1” kann max. 2^{20} Speicherplätze umfassen) `Mem1` > 2^{20} wird in `Mem1` = 2^{20} korrigiert. – `Mem1` = “–1” wird als `Mem1` = (2^{32}–1) interpretiert und nach `Mem1` = 2^{20} korrigiert. Beispiel: Mit `config_list( -1, x )` und `x` beliebig (auch für `x` = –1) wird “Liste 1” automatisch der komplette Listenspeicher zugeordnet (`Mem1` = 2^{20}, `Mem2` = 0, keine Speicherplätze für “Liste 3”). – `Mem2` ≤ 2^{20} – `Mem1` (“Liste 2” darf maximal den “Rest” einnehmen) `Mem2` = 0 ist erlaubt. `Mem2` > 2^{20} – `Mem1` wird in `Mem2` = 2^{20} – `Mem1` korrigiert. – `Mem2` = “–1” wird als `Mem2` = (2^{32}–1) interpretiert und nach `Mem2` = 2^{20} – `Mem1` korrigiert. Beispiel: Mit `config_list( Mem1, -1 )`, wird “Liste 2” automatisch der “Rest” des Listenspeichers zugeordnet (`Mem2` = 2^{20} – `Mem1`, keine Speicherplätze für “Liste 3”). – Speicherplätze für “Liste 3”: 2^{20} – `Mem1` – `Mem2` • Der RTC5-Listenspeicher ist standardmäßig so vorkonfiguriert, dass die Speicherbereiche “Liste 1” und “Liste 2” jeweils 4.000 Listenbefehle aufnehmen können (`Mem1` = `Mem2` = 4.000). Der geschützte Speicherbereich “Liste 3” umfasst dann die restlichen 1.040.576 der 2^{20} Speicherplätze. • Der Befehl **config_list** wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status oder PAUSED-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** oder **set_wait** angehalten). • 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 Input-Pointer (liegt dieser vor der Konfiguration auf einer Speicherposition, die bei der Konfiguration der “Liste 3” zugewiesen wird, so wird der Input-Pointer 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 “Konfiguration des Listenspeichers”, Seite 74.	

Kontrollbefehl	config_list
Hinweise (Forts.)	• Sind die aktuellen Konfigurationsdaten des 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).
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_config_list**

<table>
<tr><td>Kontrollbefehl</td><td colspan="4">control_command</td></tr>
<tr><td>Funktion</td><td colspan="4">Sendet einen Kontrollbefehl an ein iDRIVE-Scan-System (intelliSCAN, intelliSCAN_{de}, intelliDRILL, intellicube, intelliWELD, varioSCAN_{de}).</td></tr>
<tr><td>Aufruf</td><td colspan="4"><code>control_command(Head, Axis, Data)</code></td></tr>
<tr><td>Parameter</td><td><code>Head</code></td><td colspan="3">Nummer des Scan-Kopf-Anschlusses als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte:
= 1: Primärer Scan-Kopf-Anschluss.
= 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich).</td></tr>
<tr><td></td><td><code>Axis</code></td><td colspan="3">Nummer der Achse als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte:
= 1: Achse X (STATUS-Kanal, Galvanometerscanner 2).
= 2: Achse Y (STATUS1-Kanal, Galvanometerscanner 1).</td></tr>
<tr><td></td><td><code>Data</code></td><td colspan="3">Befehlscode mit optionalem Parameter als 32-Bit-Wert ohne Vorzeichen.
Die obere Hälfte (Bits#16-31) wird nicht ausgewertet, sondern nur die untere (Bits#0-15). Das höherwertigere Daten-Byte Code_H (Bits#8-15) dieser unteren Hälfte gibt dabei einen Befehlscode an und das niederwertigere Daten-Byte Code_L (Bits#0-7) einen optionalen Parameter. In dieser Tabelle wird für jeden Befehlscode der entsprechende Befehl mit den erlaubten Parameterwerten beschrieben.
Beispiel: Mit <code>Data</code> = 0501_H, d.h. Code_H = 05_H (Befehl SetMode) und Code_L = 01_H, wird die Ist-Position als vom Scan-System rückzuübertragender Datentyp ausgewählt.</td></tr>
<tr><td></td><td></td><td>Code_H</td><td colspan="2">Befehl und Parameterwerte (Code_L)</td></tr>
<tr><td></td><td></td><td>05_H</td><td colspan="2">SetMode: Mit diesem Befehl kann ausgewählt werden, welcher Datentyp für die ausgewählte Achse über den entsprechenden Statuskanal vom Scan-System rückübertragen wird. Siehe auch Kapitel 8.1.2 "Anpassen des Rückmeldeverhaltens", Seite 173. Dabei entspricht jeder Parameterwert Code_L einem bestimmten Datentyp.
– Standardeinstellung: Code_L = 00_H (XY2-100-Statuswort).
– Die grau markierten Datentypen sind nur für den intelliWELD relevant.
iDRIVE-Scan-Systeme mit SL2-100-Interface geben generell einen 20-Bit-Wert mit Vorzeichen zurück, iDRIVE-Scan-Systeme mit XY2-100 Enhanced-Interface einen 16-Bit-Wert mit Vorzeichen. Bei letzteren werden die Werte vom XY2-100-Konverter durch Multiplizieren mit 16 in 20-Bit-Werte mit Vorzeichen konvertiert.</td></tr>
<tr><td></td><td></td><td></td><td>Code_L</td><td>Vom Scan-System rückübertragener Datentyp</td></tr>
<tr><td></td><td></td><td></td><td>00_H</td><td>XY2-100-Statuswort
Bit #19 (MSB), 11 = 1: Interne Spannungen in Ordnung (Power OK).
Bit #18, 10 = 1: Galvanometerscanner-Temperatur im erlaubten Bereich (TempOK).
Bit #16, 15, 8, 7 = 1: Positionsfehler der X- und Y-Achse im erlaubten Bereich (PosAck-Signal).
Bit #17, 12, 9, 4 = 1.
Bit #14, 6 = 1 (Beim intelliWELD und bei Scan-Systemen mit Sensoren zur automatischen Selbstkalibrierung: Reserviert)
Bit #13, 5 = 0.
Bits #0…3 = 0.</td></tr>
<tr><td></td><td></td><td></td><td>01_H</td><td>Ist-Position (Winkelstellung) / Bit [-2^{19} … $2^{19}-1$].</td></tr>
<tr><td></td><td></td><td></td><td>02_H</td><td>Sollposition (Winkelstellung) / Bit [-2^{19} … $2^{19}-1$].</td></tr>
<tr><td></td><td></td><td></td><td>03_H</td><td>Positionsfehler (= Soll-Position – Ist-Position) / Bit [-2^{19} … $2^{19}-1$].</td></tr>
<tr><td></td><td></td><td></td><td>04_H</td><td>Ist-Strom (Endstufenstrom) / 16^{-1} mA [-2^{19} … $2^{19}-1$].</td></tr>
<tr><td></td><td></td><td></td><td>05_H</td><td>Relative Galvanometerscanner-Ansteuerung [–16000 … 16000] (nicht beim intellicube) (der Rückgabewert 16 entspricht 1‰).</td></tr>
<tr><td></td><td></td><td></td><td>06_H</td><td>Ist-Geschwindigkeit (Winkelgeschwindigkeit) / (Bit/ms) [-2^{19} … $2^{19}-1$].</td></tr>
</table>

Kontrollbefehl	control_command			
		(05_H)	***Code$_L$***	***Rückübertragener Datentyp***
			12_H	Ist-Position der Z-Achse / Bit [-2^{19} … $2^{19}-1$] (nur beim intelli*WELD* mit vario*SCAN* FC; nicht mit vario*SCAN* FC i). Dieser Datentyp kann nur über Kanal des Scan-Kopf-Anschlusses, an dem die Z-Achse angeschlossen ist, ausgelesen werden.
			14_H	Temperatur des Galvanometerscanners / 160^{-1} °C [-2^{19} … $2^{19}-1$].
			15_H	Temperatur der Servokarte / 160^{-1} °C [-2^{19} … $2^{19}-1$].
			16_H	AGC-Spannung (PD-Versorgungsspannung) / 1600^{-1} V [-2^{19} … $2^{19}-1$].
			17_H	DSP Core-Versorgungsspannung (1,8 V) / 1600^{-1} V [-2^{19} … $2^{19}-1$].
			18_H	DSP IO-Spannung (3,3 V) / 1600^{-1} [-2^{19} … $2^{19}-1$].
			19_H	Spannung Analogteil (9 V) / 1600^{-1} V [-2^{19} … $2^{19}-1$].
			$1A_H$	Versorgungsspannung des AD-Konverters (5 V) / 1600^{-1} V [-2^{19} … $2^{19}-1$].
			$1B_H$	AGC-Strom (PD-Versorgungsstrom) / 16^{-1} mA [-2^{19} … $2^{19}-1$].
			$1D_H$	Relative Heizleistung der Heizung des jeweiligen Galvanometerscanners (nicht relevant beim intelli*cube*, dyn*AXIS*® XS und bei Scan-Systemen mit wassergekühlten Scannern). Bits #4…19 Relative Heizleistung / ‰ [0 … 1000]. Bits #0…3 = 0.
			$1E_H$	Seriennummer (untere 16 Bits) Bits #4…19 Seriennummer (untere 16 Bits) [0 … $2^{16}-1$]. Bits #0…3 = 0.
			$1F_H$	Seriennummer (obere 16 Bits) Bits #4…19 Seriennummer (obere 16 Bits) [0 … $2^{16}-1$]. Bits #0…3 = 0.
			20_H	Artikelnummer (untere 16 Bits) Bits #4…19 Artikelnummer (untere 16 Bits) [0 … $2^{16}-1$]. Bits #0…3 = 0.
			21_H	Artikelnummer (obere 16 Bits) Bits #4…19 Artikelnummer (obere 16 Bits) [0 … $2^{16}-1$]. Bits #0…3 = 0.
			22_H	Firmware-Versionsnummer Bits #4…19 Versionsnummer [0 … $2^{16}-1$]. Bits #0…3 = 0.
			23_H	Kalibrierwinkel (mechanischer Scan-Winkel (±) bei 96% des maximalen bzw. minimalen Ansteuerwertes, d.h. bei ±503316 Bit) Bits #4…19 Kalibrierwinkel / 10^{-3} ° [0 … $2^{16}-1$]. Bits #0…3 = 0.
			24_H	Apertur Bits #4…19 Apertur / mm [0 … $2^{16}-1$]. Bits #0…3 = 0.

Kontrollbefehl	control_command			
		(05_H)	***Code$_L$***	***Rückübertragener Datentyp***
			25_H	Wellenlänge
				Bits #4…19 Wellenlänge / nm [0 … 2^{16}–1].
				Bits #0…3 = 0.
			26_H	Tuningnummer (ab Firmware-Versionsnummer 2078)
				Bits #12…19 Starteinstellung.
				Bits #4…11 Aktuelle Einstellung.
				Bits #0…3 = 0.
			27_H	Nur nach **control_command**(`Data` = $17FF_H$): Nummer des intern (z.B. durch *Code* = $17FF_H$ zwecks Wiederherstellung mit *Code* = 1700_H) zwischengespeicherten, zur Rückübertragung ausgewählten Datentyps (ab Firmware-Versionsnummer 2078)
				Bits #12…19 = 0.
				Bits #4…11 Zwischengespeicherte Einstellung.
				Bits #0…3 = 0.
			28_H	Aktueller Betriebszustand (Flags B15 … B0):
				Bit #19 (MSB) (B15) = 1: Die Endstufe des Galvanometerscanners ist eingeschaltet.
				Bit #18 (B14) = 1: Die Endstufe der Galvanometerscanner-Heizung ist eingeschaltet (beim intelli*cube*, dyn*AXIS*® XS und bei Scan-Systemen mit wassergekühlten Scannern immer 0).
				Bit #17 (B13) = 1: Interne Spannungen in Ordnung.
				Bit #16 (B12) = 1: Positionsfehler im erlaubten Bereich.
				Bit #15 (B11) = 1: Temperatur von Galvanometerscanner und Servokarte im erlaubten Bereich.
				Bit #14 (B10) = 1: Anfahrprozess beendet.
				Bit #13 (B9) = 0: Es liegt ein kritischer Fehler vor. Das Scan-System wurde automatisch in einen permanenten Fehlzustand geschaltet.
				Bit #12 (B8) = 0: Es lag mindestens kurzzeitig ein Spannungseinbruch der externen Versorgungsspannung vor.
				Bit #11 (B7) = 0: Die Temperatur im Scan-System liegt über dem max. zulässigen Wert. Das Scan-System wurde automatisch in einen temporären Fehlzustand geschaltet.
				Bit #10 (B6) = 1: Der AD-Wandler wurde erfolgreich initialisiert.
				Bit #9 (B5) = 0: Der Galvanometerscanner hat eine kritische Position erreicht.
				Bit #8 (B4) = 1: Alle Regelparameter gültig.
				Bit #7 (B3) Reserviert (beim intelli*WELD*: 1 = aktuell kein INTERLOCK-Fehler).
				Bit #6 (B2) Reserviert (beim intelli*WELD*: 1 = die Sensor-Platine wurde erkannt).
				Bit #5 (B1) Reserviert.
				Bit #4 (B0) = 1: Die Regelung wird angefahren, sobald alle erforderlichen Flags gesetzt sind.
				Bits #0…3 = 0.

Kontrollbefehl	control_command			
		(05_H)	***Code$_L$***	***Rückübertragener Datentyp***
			29_H	Aktueller Betriebszustand (Flags B31 … B16): Bit #19 (MSB) (B31) = 1: Versorgungsspannung des AD-Konverters (5 V) OK. Bit #18 (B30) = 1: Spannung Analogteil (9 V) OK. Bit #17 (B29) = 1: DSP IO-Spannung (3,3 V) OK. Bit #16 (B28) = 1: DSP Core-Versorgungsspannung (1,8 V) OK. Bit #15 (B27) = 1: AGC-Spannung (PD-Versorgungsspannung) OK. Bit #14(B26) = 1: Betriebstemperatur der Servokarte im erlaubten Bereich. Bit #13 (B25) = 1: Betriebstemperatur des Galvanometerscanners im erlaubten Bereich. Bits #4…12 (B16-24) Reserviert. Bits #0…3 = 0.
			$2A_H$	Stopp-Ereigniscode = 00010_H: Der Galvanometerscanner hat eine kritische Position erreicht. = 00020_H: Fehler beim AD-Wandler. = 00030_H: Temperatur im Scan-System über dem max. zulässigen Wert. = 00040_H: Spannungseinbruch der externen Versorgungsspannung. = 00050_H: Die Flags erfüllen nicht die Betriebsvoraussetzung. = 00060_H - $000C0_H$: Reserviert. = $000D0_H$: Watchdog 10 µs Time-out (Schleifenzeit überschritten). = $000E0_H$: Position Acknowledge Time-out (Sollposition lange nicht erreicht). = $000F0_H$: Reserviert.
			$2B_H$	Betriebszustand zum Zeitpunkt der zuletzt stattgefundenen Betriebsunterbrechung (Flags B15 … B0, siehe `Data` = 0528_H)
			$2C_H$	Betriebszustand zum Zeitpunkt der zuletzt stattgefundenen Betriebsunterbrechung (Flags B31 … B16, siehe `Data` = 0529_H)
			$2F_H$	Betriebszeit (Sekunden) (ab Firmware-Versionsnummer 2061) Bits #4…19 Betriebszeit (Sekunden) / s [0 … 59]. Bits #0…3 = 0.
			30_H	Betriebszeit (Minuten) (ab Firmware-Versionsnummer 2061) Bits #4…19 Betriebszeit (Minuten) / min [0 … 59]. Bits #0…3 = 0.
			31_H	Betriebszeit (Stunden) (ab Firmware-Versionsnummer 2061) Bits #4…19 Betriebszeit (Stunden) / h [0 … 23]. Bits #0…3 = 0.
			32_H	Betriebszeit (Tage) (ab Firmware-Versionsnummer 2061) Bits #4…19 Betriebszeit (Tage) / d [0 … 2^{16}–1]. Bits #0…3 = 0.
			Die grau markierten Datentypen sind nur für den intelli*WELD* relevant.	
			33_H	3.3 V Sensorplatinen-Betriebsspannung Bits #4…19 Betriebsspannung / 10^{-3} V [0 … 7000]. Bits #0…3 = 0.
			34_H	Sensorplatinen-Betriebstemperatur Bits #4…19 Betriebstemperatur / 0,1°C [0 … 65535]. Bits #0…3 = 0.

Kontrollbefehl	control_command			
		(05_H)	$Code_L$	***Rückübertragener Datentyp***
			35_H	Temperatur am Strahlaustritt Bits #4…19 Temperatur / 0,05°C [0 … 2250]. Bits #0…3 = 0.
			36_H	Temperatur von Spiegel 2 Bits #4…19 (Temperatur + 26,6°C) / 0,05°C [0 … 1992]. Bits #0…3 = 0.
			37_H	Temperatur von Spiegel 1 Bits #4…19 (Temperatur + 26,6°C) / 0,05°C [0 … 1992]. Bits #0…3 = 0.
			38_H	Temperatur des Schutzglasfensters Bits #4…19 Temperatur / 0,044°C [0 … 2250]. Bits #0…3 = 0.
			39_H	Kollimatortemperatur Bits #4…19 Temperatur / 0,05°C [0 … 2250]. Bits #0…3 = 0.
			$3A_H$	Temperatur des Galvanometerscanner-Halters Bits #4…19 Temperatur / 0,05°C [0 … 2250]. Bits #0…3 = 0.
			$3B_H$	Kühlflussrate Bits #4…19 (Kühlflussrate + 0,93 l × min^{-1}) / (0,0052 l × min^{-1}) [0 … 2250]. Bits #0…3 = 0.
			$3C_H$	Schutzglasfenster-Streulichtwert Bits #4…19 Streulichtwert / 0,00444 V [0 … 2250]. Bits #0…3 = 0.
			$3D_H$	Sensor-Platinen-Flags (*) falls "1" dann wird ein INTERLOCK-Fehler ausgelöst. Bit #19 (MSB) = 1: Temperaturwert des Schutzglasfensters außerhalb [0 … 1882]. Bit #18 = 1: Temperaturwert von Spiegel 1 außerhalb [0 … 2250]. Bit #17 = 1: Temperaturwert von Spiegel 2 außerhalb [0 … 2250]. Bit #14 (*) = 1: Temperaturwert am Strahlaustritt außerhalb [0 … 1200]. Bit #7 = 1: Schutzglasfenster-Streulichtwert außerhalb [0 … 2250]. Bit #6 = 1: Kühlflussrate außerhalb [750 … 2250]. Bit #5(*) = 1: Temperaturwert des Galvanometerscanner-Halters außerhalb [0 … 1600]. Bit #4 (*) = 1: Kollimatortemperaturwert außerhalb [0 … 2000]. Bits #0…3 = 0.
			$3F_H$	Eingestellter Skalierungsfaktor für Positionswerte (ab Firmware-Versionsnummer 2072) Bits #6…19 Reserviert. Bits #4, 5 Skalierungsfaktor = $1/2^n$ mit n = 2 × Bit #5 + Bit #4. Bits #0…3 = 0.

<table>
<tr><td>Kontrollbefehl</td><td colspan="4">control_command</td></tr>
<tr><td></td><td></td><td>Code_H</td><td colspan="2">Befehl und Parameterwerte (Code_L)</td></tr>
<tr><td></td><td></td><td>0A_H</td><td colspan="2">UpdatePermanentMemory (ab Firmware-Versionsnummer 2078):
Dieser Befehl (mit dem einzig zulässigen Parameterwert Code_L = 00_H) stellt das aktuell eingestellte Regelverhalten als Startverhalten für nachfolgende Neustarts oder Resets ein. Siehe auch Kapitel 8.1.8 “Anpassen des Startverhaltens”, Seite 182.</td></tr>
<tr><td></td><td></td><td>0E_H</td><td colspan="2">SetControlDefinitionMode
Mit diesem Befehl kann eingestellt werden, dass Informationen über ein bestimmtes Tuning (bzw. den entsprechenden Regelalgorithmus) über den ausgewählten Statuskanal vom Scan-System rückübertragen wird. Code_L [0…3] gibt dabei die Tuningnummer an. Rückübertragen werden die Tuninginformationen – wie beim SetMode-Befehl – generell als 20-Bit-Wert mit Vorzeichen.
<table>
<tr><td>Code_L</td><td colspan="2">vom Scan-Kopf rückübertragener Wert:</td></tr>
<tr><td rowspan="4">00_H
…
03_H</td><td>Bit #19 (MSB)</td><td>= 0: Tuning vorhanden.
= 1: Tuning nicht vorhanden.</td></tr>
<tr><td>Bits #8…18</td><td>Reserviert.</td></tr>
<tr><td>Bits #4…7</td><td>Tuningtyp:
= 0: Vektortuning (Mikrovektorisierung).
= 1: Sprungtuning.
= 2,3: Reserviert.</td></tr>
<tr><td>Bits #0…3</td><td>= 0.</td></tr>
</table></td></tr>
<tr><td></td><td></td><td>11_H</td><td colspan="2">SelectControlDefinition (ab Firmware-Versionsnummer 2078):
Wenn das Scan-System mit mehreren Tunings (bzw. entsprechenden Regelalgorithmen) ausgestattet ist, dann kann mit diesem Befehl das gewünschte Tuning ausgewählt werden. Siehe auch Kapitel 8.1.4 “Anpassen der dynamischen Einstellung (Tuning)”, Seite 176. Code_L gibt dabei die Tuningnummer an.
Standardeinstellung: Code_L = 00_H</td></tr>
<tr><td></td><td></td><td>12_H</td><td colspan="2">SetPositionScale (ab Firmware-Versionsnummer 2072):
Mit diesem Befehl kann ein Skalierungsfaktor festgelegt werden, mit dem die Regelelektronik des Scan-Systems die von der RTC5 empfangenen Positionswerte multipliziert. Siehe auch Kapitel 8.1.7 “Anpassen der effektiven Kalibrierung”, Seite 181. Die Änderung des Skalierungsfaktors erfolgt in zwei Schritten. Zuerst muss der Befehl mit dem Befehlsparameter Code_L = 83_H und anschließend ein zweites Mal mit dem gewünschten Parameterwert (siehe folgende Auflistung) ausgeführt werden.
In der Standardeinstellung (Code_L = 00_H) ist der Skalierungsfaktor 1 (keine Skalierung).
<table>
<tr><td>Code_L</td><td>Effekt auf den Skalierungsfaktor</td></tr>
<tr><td>00_H</td><td>Der Skalierungsfaktor wird auf den Wert 1 eingestellt
(vorher den Befehlscode <code>Data</code> = 1283_H ausführen!)</td></tr>
<tr><td>01_H</td><td>Der Skalierungsfaktor wird auf den Wert 1/2 eingestellt
(vorher den Befehlscode <code>Data</code> = 1283_H ausführen!)</td></tr>
<tr><td>02_H</td><td>Der Skalierungsfaktor wird auf den Wert 1/4 eingestellt
(vorher den Befehlscode <code>Data</code> = 1283_H ausführen!)</td></tr>
<tr><td>03_H</td><td>Der Skalierungsfaktor wird auf den Wert 1/8 eingestellt
(vorher den Befehlscode <code>Data</code> = 1283_H ausführen!)</td></tr>
</table></td></tr>
</table>

Kontrollbefehl	**control_command**		
		Code$_H$	***Befehl und Parameterwerte (Code$_L$)***
		15_H	***SetPosAcknowledgeLevel*** (ab Firmware-Versionsnummer 2066): Mit diesem Befehl kann der PosAcknowledge-Schwellwert eingestellt werden. Siehe auch Kapitel 8.1.6 "Anpassen des PosAcknowledge-Schwellwerts", Seite 181. Der Parameterwert *Code$_L$* ist der gewünschte PosAcknowledge-Schwellwert in Bit [00_H … FF_H]. Siehe auch Hinweise Seite 292.
		17_H	***Store/RestoreTransmissionMode*** (ab Firmware-Versionsnummer 2078): Mit diesem Befehl kann der aktuell zur Rückübertragung ausgewählte Datentyp zwischengespeichert (*Code$_L$* = FF_H) und zu einem späteren Zeitpunkt wieder ausgewählt werden (*Code$_L$* = 00_H). Siehe auch *Code* = 0527_H.
		21_H	***SetEchoMode*** (ab Firmware-Versionsnummer 2078): Dieser Befehl dient zur Überprüfung der Datenübertragung. Siehe auch Kapitel 8.1.9 "Fehlerdiagnose und Funktionstest", Seite 182. Er überträgt als Befehlsparameter den gewünschten 8-Bit-Wert (*Code$_L$*) an das Scan-System und veranlasst, dass im Folgenden am entsprechenden Statuskanal ein 20-Bit-Wert zurückgesendet wird: Bei fehlerfreier Datenübertragung sind die oberen 8 Bit des rückgesendeten 20-Bit-Wertes identisch mit Code$_L$ und die nächstniedrigeren 8 Bit identisch mit dem Komplementärwert von *Code$_L$* (NOT *Code$_L$*). Beispiel: Für `control_command(1, 1, 0x210A)` liefert `(get_value(1) AND 0xFFFF0)` bei fehlerfreier Datenübertragung den Wert 0x0AF50 zurück.
Hinweise	*Allgemeine Hinweise:* • Der **control_command**-Befehl kann nur zusammen mit i*DRIVE*-Scan-Systemen (intelli*SCAN*, intelli*SCAN$_{de}$*, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario*SCAN$_{de}$*) verwendet werden. Von herkömmlichen Scan-Systemen (ohne i*DRIVE*-Technologie) wird der Befehl nicht ausgewertet. • Für unzulässige Werte von `Head` und/oder `Axis` wird der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`). Dies gilt auch für `Head` = 2, falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist. • Der Befehlscode `Data` wird anstelle der sonst für `Head` und `Axis` üblichen Positionsdaten an das Scan-System ausgegeben. Der entsprechende Galvanometerscanner-Mikroschritt fällt damit aus, wenn der Befehl während der Abarbeitung einer Liste erteilt wird. • Einige Befehlsparameter des **control_command**-Befehls sind nicht zusammen mit älteren Firmware-Versionen von "intelli"-Scan-Systemen nutzbar. Sofern keine entsprechende Angabe vermerkt ist, ist der jeweilige Befehlsparameter jedoch bei allen Firmware-Versionen nutzbar. • Der **control_command**-Befehl wird u. U. durch Aktivierung einer geschwindigkeitsabhängigen Lasersteuerung mit dem Befehl **set_auto_laser_control** (`Mode` = 2) für den primären Scan-Kopf-Anschluss gesperrt.		

Kontrollbefehl	control_command
Hinweise (Forts.)	*Hinweise zu den Befehlen SetMode ($Code_H = 05_H$), SetControlDefinitionMode ($Code_H = 0E_H$) SetEchoMode ($Code_H = 21_H$) und RestoreTransmissionMode (`Data` = 1700_H):* • Der mit dem **control_command**-Befehl ($Code_H = 05_H$, $0E_H$ oder 21_H bzw. `Data` = 1700_H) ausgewählte Datentyp wird so lange übertragen, bis ein anderer selektiert wird. • An die RTC5 rückübertragene Daten können mit den Befehlen **get_value**, **get_values**, **set_trigger**/**set_trigger4** bzw. **get_waveform** gelesen werden. Da die ersten Daten nach einem Umschalten der Datenquelle aufgrund der seriellen Übertragungszeiten erst nach einer kurzen zeitlichen Verzögerung übertragen werden, kann zwischen dem Umschaltzeitpunkt und dem Einlesen eine Verzögerungszeit von bis zu 60 µs entstehen. Ab DLL-Version 520 fügt **control_command** nach einem Umschalten der Datenquelle automatisch immer eine Wartezeit von 60 µs ein (die oben genannten Befehle lesen jetzt also in jedem Fall korrekte Rückgabewerte aus). • Alle vom Scan-System rückübertragenen Daten werden als 20-Bit-Werte mit Vorzeichen an die RTC5 übergeben. Dies gilt auch dann, wenn die DLL auf den RTC4-Kompatibilitätsmodus eingestellt ist und auch für Scan-Systeme ohne SL2-100-Interface, die über einen XY2-100-Konverter angesteuert werden. Beim Auslesen der zurückgegebenen Daten mit den Befehlen **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-Interface liest **get_head_status** unabhängig von einer mit **control_command** ($Code_H = 05_H$, $0E_H$ oder 21_H bzw. `Data` = 1700_H) getroffenen Einstellung das XY2-100-Statuswort aus, das hier zusätzlich zu den ausgewählten Daten rückübertragen wird. Bei i*DRIVE*-Scan-Systemen (ohne integriertes SL2-100-Interface), die über einen XY2-100-Konverter mit der RTC5 angesteuert werden, liefert **get_head_status** dagegen das XY2-100-Statuswort nur dann zurück, wenn dieses Signal auch zuvor mit **control_command** zur Rückübertragung ausgewählt wurde (siehe auch Seite 173). • Nach einem Neustart oder Reset des Scan-Systems kann es ca. 5 Sekunden dauern, bis Daten vom Scan-System rückübertragen werden (siehe auch **get_value**). Nach einem Neustart oder Reset wird immer das XY2-100-Statuswort rückübertragen. • **control_command** (`Data` = 1700_H) bleibt wirkungslos, wenn nach dem letztmaligen Aufruf des StoreTransmissionMode-Befehls (`Data` = $17FF_H$) ein Neustart oder Reset durchgeführt wurde.

Kontrollbefehl	control_command
Hinweise (Forts.)	*Hinweise zum Befehl SetMode (Code$_H$ = 05$_H$):* • Die vom Scan-System rückübertragenen Sollpositionswerte (Winkelstellung) entsprechen den tatsächlichen Ausgabewerten `Sample<AX..BY>_Out` bei **set_trigger**/**set_trigger4** und berücksichtigen gegebenenfalls eine Wobbel-Korrektur, eine "Processing on the fly"-Korrektur, eine Koordinatentransformation, eine Bildfeldkorrektur, sowie "Offset" und "Gain" zur automatischen Selbstkalibrierung des Scan-Systems. Die zurückgegebenen 20-Bit-Sollpositionswerte der Z-Achse (und im RTC4-Kompatibilitätsmodus auch die der X- und Y-Achsen) sind zusätzlich um einen Faktor 16 gegenüber den hierfür angegebenen 16-Bit-Koordinatenwerten skaliert. • Im temporären Fehlzustand nach Überschreiten der maximal zulässigen Temperatur (Flag Bit#7 = 0) sind die Endstufen der jeweiligen Achse (vorübergehend) deaktiviert. Der Normalbetrieb wird wieder aufgenommen, sobald die Temperatur wieder unter die maximal zulässige Temperatur fällt. • Beim Auftreten eines kritischen Fehlers (Flag Bit#9 = 0) schaltet das Scan-System automatisch in einen permanenten Fehlzustand, in dem die Endstufen der betroffenen Achse permanent deaktiviert bleiben – auch dann, wenn der kritische Fehler nur kurzzeitig aufgetreten ist. Der Normalbetrieb wird *nicht* wieder aufgenommen. Das Flag Bit#9 wird erst nach einem Hardware-Reset wieder gesetzt. Kritische Fehler sind z. B. fehlerhafte interne Spannungen (Flag Bit#13 = 0), ein Einbruch der externen Versorgungsspannung (Flag Bit#8 = 0) oder das Erreichen einer kritischen Randposition (Flag Bit#5 = 0). • Sowohl im temporären als auch im permanenten Fehlzustand werden weiterhin Daten vom Scan-System an die Ansteuerkarte zurückgesendet. Alle Datentypen können in diesen Zuständen noch nachträglich zu Diagnosezwecken ausgewählt bzw. ausgelesen werden. • Die Flags geben Auskunft über den Betriebszustand des Scan-Systems. Es kann wahlweise einer von zwei Blöcken von Flags zur Rückübertragung ausgewählt werden, die den aktuellen Betriebszustand beschreiben (`Data` = 0528$_H$, 0529$_H$) oder alternativ einer von zwei weiteren Blöcken, die den Betriebszustand zum Zeitpunkt der zuletzt stattgefundenen Betriebsunterbrechung angeben (`Data` = 052B$_H$, 052C$_H$). Nach jedem erfolgreichen Neustart – und solange kein Fehler auftritt – sind alle Statusinformationen der zwei letzteren Blöcke (`Data` = 052B$_H$, 052C$_H$) bedeutungslos. Erst beim Auftreten eines Fehlers, der zum Wechsel in einen temporären oder permananten Fehlzustand führt, werden hier die zu diesem Zeitpunkt aktuellen Statuswerte gespeichert. Gleichzeitig wird dann auch ein Ereignis-Code gesetzt, der darüber Auskunft gibt, welches Ereignis für den Wechsel in den Fehlzustand verantwortlich war. Dieser Ereignis-Code kann separat abgerufen werden (`Data` = 052A$_H$). • Die an die RTC5 zurückgegebenen Winkelgrößen in Bit (Ist-Position, Soll-Position, Positionsfehler) bzw. Bit/ms (Ist-Geschwindigkeit) können mit Hilfe des Kalibrierwinkels des Scan-Systems in die Maßeinheiten ° bzw. °/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 RTC5-Handbuch sind die Befehlscodes für alle theoretisch möglichen Sensoren aufgelistet.

Kontrollbefehl	control_command
Hinweise (Forts.)	*Hinweise zum Befehl SetPositionScale ($Code_H = 12_H$):* • Wenn der Skalierungsfaktor für die Positionswerte verändert wird, dann muss ein im Anwenderprogramm verwendeter Kalibrierungsfaktor zur Berechnung der RTC5-Ansteuerwerte (in Bits) aus den gewünschten Positionswerten im Bildfeld (in mm) (siehe Kapitel 7.3.2 "Bildfeld-Größe und -Kalibrierung", Seite 134) umgekehrt proportional skaliert werden. Außerdem sollten die Sprung- und Markiergeschwindigkeit angepasst werden. • Mit `Data` = $053F_H$ kann der aktuell eingestellte Skalierungsfaktor ausgelesen werden. *Hinweise zum Befehl SetPosAcknowledgelevel ($Code_H = 15_H$):* • Der Schwellwert muss bezogen auf einen 16-Bit-Ansteuerbereich angegeben werden. Er wird von der RTC5 durch Multiplizieren mit 16 in einen auf einen 20-Bit-Ansteuerbereich bezogenen Wert umgerechnet (und gegebenenfalls von einem XY2-100-Konverter wieder zurückkonvertiert). • Das Startverhalten ist standardmäßig so voreingestellt, dass das Scan-System nach jedem Neustart oder Reset einen PosAcknowledge-Schwellwert von 0,28% des vollen Positionsbereichs einstellt (bei i*DRIVE*-Scan-Systemen mit SL2-100-Interface sind dies 0,28% von 2^{20} Bits, bei i*DRIVE*-Scan-Systemen mit XY2-100 Enhanced-Interface 0,28% von 2^{16} Bits oder 183 Bits; in beiden Fällen entspricht dies $Code_L$ = 183 = $B7_H$). Wenn ein anderer Schwellwert gewünscht wird, dann muss er – für jede Achse separat – neu eingestellt werden ($Code_H = 15_H$). SCANLAB empfiehlt nur Schwellwerte ($Code_L$) über 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.
RTC4→ RTC5	Unveränderte Funktionalität. Beachten Sie, dass alle mit **control_command** zur Rückgabe ausgewählten Daten immer im 20-Bit-Wertebereich liegen, auch im RTC4-Kompatibilitätsmodus (siehe auch obigen Hinweis und Hinweise bei **get_value**).
Versionsinfo	Letzte Änderung Version DLL 520 (siehe Seite 693).
Verweise	**get_value**, **get_values**, **get_head_status**, **set_trigger**, **set_trigger4**, **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 (als 32-Bit-Wert ohne Vorzeichen) des indizierten Zeichens, der indizierten Zeichenfolge oder des indizierten Unterprogramms, dessen Einträge aus `Src` kopiert werden sollen. Zulässiger Wertebereich: [0 … 1023] für indizierte Zeichen oder Unterprogramme, [1024+0 … 1024+41] für indizierte Zeichenfolgen.
	`Src`	Index (als 32-Bit-Wert ohne Vorzeichen) des indizierten Zeichens, der indizierten Zeichenfolge oder des indizierten Unterprogramms, dessen Einträge nach `Dst` kopiert werden sollen. Zulässiger Wertebereich: [0 … 1023] für indizierte Zeichen oder Unterprogramme, [1024+0 … 1024+41] für indizierte Zeichenfolgen.
	`Mode`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, welche Verwaltungstabellen geändert werden sollen: Bit #0 = 0:`Dst` ist der Index eines indizierten Zeichens oder einer indizierten Zeichenfolge. = 1:`Dst` ist der Index eines indizierten Unterprogramms. Bit #1 = 0:`Src` ist der Index eines indizierten Zeichens oder einer indizierten Zeichenfolge. = 1:`Src` ist der Index eines indizierten Unterprogramms. Bit #2-31:Werden nicht berücksichtigt.
Hinweise	• Bei ungültigen Indexwerten (`Dst` und/oder `Src`) wird der Befehl ignoriert (**get_last_error**-Returncode: `RTC5_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). Der Befehl ändert nur die entsprechenden Einträge in der internen Verwaltungstabelle und lässt den Speicherinhalt des 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 Speicherbereich des 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 Listenspeicher geschrieben. Man sollte darauf achten, dass es dabei zu keinem Speicherüberlauf im geschützten Speicherbereich kommt. • Siehe auch Abschnitt "Verwalten von indizierten Zeichen und Zeichenfolgen", Seite 89.	
RTC4→ RTC5	Neuer Befehl.	
Verweise	**save_disk**, **load_disk**	

Kontrollbefehl	**disable_laser**
Funktion	Unterdrückt die Lasersteuersignale für den “Laser active”-Betrieb.
Aufruf	`disable_laser()`
Hinweise	• Der Befehl **disable_laser** unterdrückt die Lasersteuersignale für den “Laser active”-Betrieb an den Ausgängen LASER1, LASER2 und LASERON (die Ausgänge sind dann im hochohmigen tristate-Zustand). Pin (2) der 15-poligen 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 (1) und Pin (9) der 15-poligen LASER-Buchse und Pin (19) und Pin (22) der EXTENSION 2-Stiftleiste (on-board) statisch. • Die Lasersteuersignale 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 RTC5 mit **load_program_file** ist die Lasersteuerung *deaktiviert* und benötigt unbedingt **set_laser_control** zur Aktivierung. • Wenn der Befehl einen `RTC5_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 dem Befehl **get_startstop_info** (Bit #9) kann der aktuelle Status der Lasersteuersignale ausgelesen werden.
RTC4→ RTC5	Unverändert.
Verweise	**enable_laser**, **set_laser_control**, **get_startstop_info**

Kontrollbefehl	**enable_laser**
Funktion	Gibt die Lasersteuersignale für den “Laser active”-Betrieb frei.
Aufruf	`enable_laser()`
Hinweise	• Nach einem Hardware-Reset (und nach dem Befehl **load_program_file**) muss zur Aktivierung der Ausgänge LASER1, LASER2 und LASERON und zum Festlegen der Signalpegel der Befehl **set_laser_control** aufgerufen werden (siehe Kapitel 7.4 “Lasersteuerung”, Seite 144). Sonst bleibt der Befehl **enable_laser** wirkungslos. • Nach einer Initialisierung der RTC5 mit **load_program_file** sind die Lasersteuersignale *deaktiviert*. Zur erstmaligen Aktivierung muss der Befehl **set_laser_control** aufgerufen werden (s. o.). Nach einer anschließenden Unterdrückung mit **disable_laser** (oder **set_laser_control**) kann der Befehl **enable_laser** (oder **set_laser_control**) zur erneuten Freigabe verwendet werden. • Mit dem Befehl **get_startstop_info** (Bit #9) kann der aktuelle Status der Lasersteuersignale ausgelesen 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 “Lasersteuerung”, Seite 144).
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität. U. U. muss vor **enable_laser** der Befehl **set_laser_control** aufgerufen werden (s. o.).
Verweise	**disable_laser**, **set_laser_control**, **get_startstop_info**, **set_laser_mode**

Kontrollbefehl	execute_at_pointer
Funktion	Startet die Listenausführung (“Liste 1” oder “Liste 2”) an der gewünschten Adresse des RTC5-Listenspeichers.
Aufruf	`execute_at_pointer( Pos )`
Parameter	`Pos` Absolute Speicheradresse des ersten auszugebenden Listenbefehls als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … (2^{20}–1)].
Hinweise	• Der Befehl **execute_at_pointer** wirkt prinzipiell wie der Befehl **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 spezifiziert. • Für `Pos` ≥ `Mem1` + `Mem2` (siehe **config_list**) wird `Pos` auf 0 gesetzt.
RTC4→ RTC5	Unveränderte Funktionalität.
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”].
Hinweise	• Der Befehl **execute_list** ist synonym zu **execute_list_pos** mit `Pos` = 0. • Alternativ können die Befehle **execute_list_1** und **execute_list_2** (ohne Parameter) verwendet werden.
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**execute_list_pos**, **execute_at_pointer**, **set_start_list**

Kontrollbefehl	execute_list_1
Funktion	Siehe **execute_list**.
Aufruf	`execute_list_1`
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**execute_list**

Kontrollbefehl	execute_list_2
Funktion	Siehe **execute_list**.
Aufruf	`execute_list_2`
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**execute_list**

Kontrollbefehl	execute_list_pos
Funktion	Startet die Ausführung der gewünschten Liste (“Liste 1” oder “Liste 2”) an der gewünschten 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 bzgl. des Listenanfangs der jeweiligen Liste) als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … (2^{20}–1)].
Hinweise	• Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status oder PAUSED-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** oder **set_wait** angehalten). Wenn der INTERNAL-BUSY-Status der Karte gesetzt ist, wird der Befehl gegebenenfalls erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Status 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 “Listen-Handling”, Seite 75). • Programme im geschützten 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”, andernfalls “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 stets “Liste 1” ausgeführt. • Wird `Pos` größer als der Speicherbereich der jeweiligen Liste gewählt (`Pos` ≥ `Mem1` bzw. `Pos` ≥ `Mem2`), so wird `Pos` auf 0 gesetzt. • Der BUSY-Listenstatus der gewählten Liste wird gesetzt und der BUSY-Listenstatus der jeweils anderen Liste rückgesetzt (siehe **read_status**). Der BUSY-Listenausführungsstatus (siehe **get_status**) wird gesetzt. • Die Ausgabe stoppt, wenn ein **set_end_of_list**-Befehl angetroffen wird. Wenn das Ende des jeweiligen Listenbereichs erreicht ist, ohne dass ein **set_end_of_list**-Befehl angetroffen wurde, wird am Anfang desselben Listenbereichs fortgefahren, nicht mit der nächsten Liste. Der Output-Pointer bleibt stets im aktiven Listen-Bereich, es sei denn, ein **set_end_of_list**-Befehl wird angetroffen und es wurde zuvor ein **auto_change_pos**- oder **start_loop**-Befehl erteilt. Sollen beide Listen wie eine einzige behandelt werden, muss die Konfiguration entsprechend angepasst werden: z. B. `ConfigList( Mem1+Mem2, 0 )`. • 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-Status gesetzt). • Der **execute_list_pos**-Befehl löst einen Flush der gepufferten Listeneingabe aus (siehe Seite 75), auch dann, wenn der Start nicht erfolgreich war. • **execute_list_pos** deckt auch die Spezialisierungen **execute_list_1**, **execute_list_2**, **execute_list** bzw. **execute_at_pointer** ab.

Kontrollbefehl	execute_list_pos
RTC4→ RTC5	Neuer Befehl.
Verweise	execute_list, execute_at_pointer, set_start_list_pos

Normaler Listenbefehl	fly_return
Funktion	Deaktiviert eine zuvor eingestellte “Processing on the fly”-Korrektur (gegebenfalls simultan für mehrere Raumrichtungen) und führt anschließend einen Sprung auf die angegebene neue Ausgabeposition aus.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, führt der Befehl nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`fly_return( X, Y )`
Parameter	`X, Y` Absolute Koordinaten der neuen Ausgabeposition in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–524.288 … 524.287]. Überlaufende Werte werden auf den Randbereich geclippt.
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 83). • Siehe auch Kapitel 8.7 “”Processing on the fly” (optional)”, Seite 199.
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich und “AbsCall”, s.o.). Die Koordinaten der neuen Ausgabeposition werden als 20-Bit-Werte angegeben. Im RTC4-Kompatibilitätsmodus werden die wie bei der RTC4 als 16-Bit-Werte angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert. Der zulässige Wertebereich verringert sich entsprechend auf [–32.768 … 32.767].
Verweise	set_fly_x, set_fly_y, set_fly_rot, set_fly_x_pos, set_fly_y_pos, set_fly_rot_pos

Normaler Listenbefehl	**fly_return_z**
Funktion	Deaktiviert eine zuvor eingestellte “Processing on the fly”-Korrektur (gegebenfalls simultan für mehrere Raumrichtungen) und führt anschließend einen Sprung auf die angegebene neue Ausgabeposition aus.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, führt der Befehl nur den Sprung auf die angegebene neue Ausgabeposition aus.
Aufruf	`fly_return_z( X, Y, Z )`
Parameter	`X`, `Y`, `Z` Absolute Koordinaten der neuen Ausgabeposition in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–524.288 … 524.287]. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf die Randbereiche geclippt.
Hinweise	• Der Sprung auf die neue Ausgabeposition wird wie mit **jump_abs_3d** ausgeführt (siehe Hinweise dort). • Wird eine “Processing on the fly”-Korrektur innerhalb eines mit einem “AbsCall” aufgerufenen Unterprogramms aktiviert und mit **fly_return_z** deaktiviert, so werden die mit **fly_return_z** angegebenen Koordinatenwerte (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt (siehe auch Kapitel 6.5.1 “Unterprogramme”, Abschnitt ““AbsCalls””, Seite 83). • Siehe auch Kapitel 8.7 ““Processing on the fly” (optional)”, Seite 199.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–32.768 … 32.767]). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	**set_fly_z**

Kontrollbefehl	free_rtc5_dll
Funktion	Gibt alle von der DLL für ein Anwenderprogramm belegten Ressourcen frei.
Aufruf	`free_rtc5_dll()`
Hinweise	• Zu den von der DLL belegten Ressourcen zählt vor allem der in der DLL belegte Speicherbereich für die Kartenverwaltung, die mit dem Befehl **init_rtc5_dll** aufgebaut wird. Mit **free_rtc5_dll** wird die Kartenverwaltung der DLL gelöscht. Danach ist vom Anwenderprogramm keine Karte mehr ansprechbar (**get_last_error**-Returncode: `RTC5_ACCESS_DENIED`). • Der Befehl **free_rtc5_dll** löst keinen Reset der RTC5-Karten aus. Ein Karten-Reset wird nur durch den Befehl **load_program_file** initiiert. • Der Aufruf von **free_rtc5_dll** ist nicht zwingend erforderlich, da die von der DLL belegten Ressourcen auch automatisch freigegeben werden, wenn das Anwenderprogramm beendet wird und Microsoft Windows dabei die DLL entlädt. Einige Entwicklungsumgebungen melden jedoch im Debugmodus “Memory leaks detected!“ noch ehe die DLL entladen wird. Der Aufruf von **free_rtc5_dll** beseitigt diese Irritation.
RTC4→ RTC5	Neuer Befehl.
Verweise	**init_rtc5_dll**, **release_rtc**

Kontrollbefehl	get_auto_cal
Funktion	Liefert den zuvor mit **auto_cal** ermittelten Typ der im angeschlossenen Scan-System integrierten ASC-Hardware zurück.
Aufruf	`ASCtype = get_auto_cal( HeadNo )`
Parameter	`HeadNo` Nummer des Scan-Kopf-Anschlusses als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 1: Primärer Scan-Kopf-Anschluss. = 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich).
Rückgabe	ASC-Hardware-Typ als 32-Bit-Wert ohne Vorzeichen.
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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 520, OUT 519.
Verweise	**auto_cal**

Kontrollbefehl	get_char_pointer
Funktion	Liefert 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.
Hinweise	• Der Befehl **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 Speicherbereich geladen wurde oder ein ungeschütztes Unterprogramm nur nachträglich referenziert wurde, liegt die rückgelesene Speicheradresse im geschützten oder ungeschützten Listenbereich. • Für `Index` > 1023 oder wenn kein Zeichen für den ausgewählten Index referenziert ist, gibt der Befehl einen Wert von "–1" (d.h. 2^{32}–1) zurück. • Der Befehl 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 konditionale 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 der Befehl **list_call_cond** ignoriert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_sub_pointer**, **get_text_table_pointer**

Kontrollbefehl	get_config_list
Funktion	Gibt die Parameter der aktuellen Listenspeicherkonfiguration (`Mem1, Mem2`) an die Kartenverwaltung der DLL zurück und initialisiert diese so, als wäre der Befehl **config_list** erteilt worden.
Aufruf	`get_config_list()`
Hinweise	• Der Befehl **get_config_list** ist nützlich, wenn eine Karte den "Besitzer" wechselt und die neue Kartenverwaltung die Speicherkonfiguration nicht kennt (Karte und Verwaltung initialisieren sich unabhängig voneinander mit jeweils `Mem1 = 4000` und `Mem2 = 4000`, 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 96. • Der Befehl **get_config_list** liefert keinen Rückgabewert an das Anwenderprogramm. Die Konfigurationsdaten des Listenspeichers (`Mem1` und `Mem2`) können aber von dem Anwenderprogramm nach `load_list( ListNo, 0 )` oder `set_start_list_pos( ListNo, 0 )` mit **get_list_space** ermittelt werden. • Der Befehl **get_config_list** wird unabhängig vom BUSY-Status der Karte ausgeführt.
RTC4→ RTC5	Neuer Befehl.
Verweise	**config_list**

Kontrollbefehl	get_counts
Funktion	Liefert die aktuelle Anzahl erfolgter externer Listen-Starts zurück.
Aufruf	`Counts = get_counts()`
Rückgabe	Anzahl erfolgter externer Listen-Starts als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Die Anzahl wird aus einem internen Zähler ausgelesen, der bei jedem erfolgten Listen-Start inkrementiert wird. Der Zähler kann mit **set_control_mode** auf Null zurückgesetzt werden.
RTC4→ RTC5	Unverändert.
Verweise	**set_max_counts**, **set_control_mode**, **get_startstop_info**

Kontrollbefehl	get_dll_version
Funktion	Liefert die Versionsnummer der RTC5-DLL zurück.
Aufruf	`DLLVersion = get_dll_version()`
Rückgabe	DLL-Versionsnummer als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Die RTC5 DLL-Versionsnummern liegen im Bereich 500-599. • Der Befehl **get_dll_version** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **get_dll_version** ist nicht als Multi-Board-Befehl verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe "Error-Handling", Seite 98) werden von **get_dll_version** weder erzeugt noch verändert.
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**get_hex_version**, **get_rtc_version**

Kontrollbefehl	get_encoder
Funktion	Liefert die aktuellen Zählerstände der beiden internen Encoder-Zähler zurück.
Aufruf	`get_encoder( &Encoder0, &Encoder1 )`
Parameter-rückgabe	`Encoder0, Encoder1` Aktuelle Zählerstände als Pointer auf 32-Bit-Werte mit Vorzeichen.
Hinweise	• Zur Verwendung des Befehls siehe ""Processing on the fly" (optional)", Seite 199 und "Synchronisation durch Encoder-Signale", Seite 246. • Wird die Werkstück-Bewegung mit einem Inkremental-Encoder registriert, so wird Encoder-Zähler Encoder0 durch die Signale am Encoder-Eingang ENCODER X und Encoder-Zähler Encoder1 durch die Signale am Encoder-Eingang ENCODER Y getriggert. Wurde dagegen mit **simulate_encoder** eine Encoder-Simulation gestartet, so werden beide Encoder-Zähler durch ein internes periodisches 1 MHz-Taktsignal getriggert.
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich).
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 (entspricht einer Auflistung der seit dem letzten Reset bzw. Error-Reset aufgetretenen Fehlertypen) zurück.			
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 auch Fehlerkonstanten vordefiniert.			
	Bit	Fehlertyp		Fehlerkonstante
		Kein Fehler.		`RTC5_NO_ERROR = 0`
	Bit #0 (LSB)	= 1:	Keine Karte gefunden (der Fehler kann nur bei **init_rtc5_dll** auftreten).	`RTC5_NO_CARD = 1`
	Bit #1	= 1:	Zugriff verweigert (der Fehler kann bei **init_rtc5_dll**, **select_rtc**, **acquire_rtc** oder bei allen Multi-Board-Befehlen auftreten).	`RTC5_ACCESS_DENIED = 2`
	Bit #2	= 1:	Befehl nicht gesendet (der Fehler deutet auf einen internen, PCI- oder Treiber-Fehler z. B. durch eine defekte Hardware oder einen fehlerhaften Anschluss).	`RTC5_SEND_ERROR = 4`
	Bit #3	= 1:	Karte reagiert nicht (vermutlich ist kein Programm auf die RTC5 geladen. Der Fehler kann bei allen Kontrollbefehlen auftreten, die eine Antwort erwarten, z. B. bei **get_hex_version**).	`RTC5_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).	`RTC5_PARAM_ERROR = 16`
	Bit #5	= 1:	Listenausführung ist (nicht) aktiv (z. B. bei **execute_list**, wenn noch eine Liste läuft z. B. bei **stop_execution**, wenn gerade keine Liste läuft z. B. bei **restart_list**, wenn nicht zuvor **pause_list** aufgerufen wurde).	`RTC5_BUSY = 32`

Kontrollbefehl	get_error			
	Bit #6	= 1:	Listenbefehl verweigert, illegaler Inputpointer (z. B. bei beliebigem Listenbefehl direkt nach **load_char** + **list_return**: der Listenbefehl wird dann nicht gespeichert).	`RTC5_REJECTED = 64`
	Bit #7	= 1:	Listenbefehl wurde in **list_nop** konvertiert (z. B. **set_end_of_list** in einem geschützten Unterprogramm).	`RTC5_IGNORED = 128`
	Bit #8	= 1:	Versions-Fehler: DLL-Version (der RTC5-DLL), RTC-Version (der Firmwaredatei) und HEX-Version (der DSP-Programmdatei) passen nicht zusammen (siehe auch **load_program_file**).	`RTC5_VERSION_MISMATCH = 256`
	Bit #9	= 1:	Verify-Fehler: bei der Download-Überprüfung (siehe Seite 99) wurde ein fehlerhafter Download festgestellt.	`RTC5_VERIFY_ERROR = 512`
	Bit #10	= 1:	DSP-Versions-Fehler: Es liegt eine veraltete DSP-Version vor (der Fehler kann nur bei älteren RTC5-Karten – siehe **get_rtc_version** Bits #16-23 – und auch nur bei einigen Befehlen wie **mark_ellipse_abs** auftreten; in den entsprechenden Befehlsbeschreibungen ist unter "Versionsinfo" ein entsprechender Hinweis eingetragen; Befehle, bei denen der Fehler auftritt, werden weder ausgeführt noch durch **list_nop** ersetzt).	`RTC5_TYPE_REJECTED = 1024`
	Bit #11	= 1:	Eine DLL-interne Windows-Speicher-Anforderung schlug fehl.	`RTC5_OUT_OF_MEMORY = 2048`
	Bit #12	= 1:	EEPROM-Lese- oder Schreibfehler (kann bei der Initialisierung und **auto_cal** auftreten).	`RTC5_EEPROM_ERROR = 4096`
	Bits #13… #15	Reserviert.		
	Bit #16	= 1:	Lesefehler beim PCI-Konfigurationsregister (kann nur bei **init_rtc5_dll** auftreten).	`RTC5_CONFIG_ERROR = 65536`
	Bits #17… #31	Reserviert.		
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Error-Handling", Seite 98. • Die Befehle **get_error** und **n_get_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **get_error** weder erzeugt noch verändert.			

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 sei von Benutzern als Konstante definiert`
	`UINT Error = init_rtc5_dll(); // Suche alle installierten RTC5-Karten`
	`if (Error & RTC5_ACCESS_DENIED) { // mindestens eine Karte ist gesperrt`
	`UINT Anzahl = rtc5_count_cards(); // Anzahl Karten gefunden`
	`for ( UINT Nr = 1; Nr <= Anzahl; Nr++ ) {`
	`NoAccess[Nr] = n_get_last_error(Nr) & RTC5_ACCESS_DENIED;`
	`n_reset_error(Nr, RTC5_ACCESS_DENIED);`
	`}`
	`}`

RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Änderung mit Version DLL 518: Bit #10. Letzte Änderung Version DLL 535: Bits #11, 12, 16.
Verweise	**get_last_error**, **reset_error**, **set_verify**

Kontrollbefehl	get_fly_2d_offset
Funktion	Liefert die aktuellen Referenzwerte (Offsetwerte) für die 2D-Encoder-Kompensation zurück.
Aufruf	`get_fly_2d_offset( &OffsetX, &OffsetY )`
Parameter-rückgabe	`OffsetX, OffsetY` Referenzwerte als Pointer auf 32-Bit-Werte mit Vorzeichen.
Hinweise	• Zur 2D-Encoder-Kompensation siehe Abschnitt "2D-Encoder-Kompensation für XY-Tische", Seite 206.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**init_fly_2d**, **set_fly_2d**

Kontrollbefehl	get_free_variable
Funktion	Liefert 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 Bereich: [0 … 7]. Es werden nur die beiden niederwertigsten Bits ausgewertet.
Rückgabe	Der aktuell in der freien Variablen `No` gespeicherte Wert (32-Bit-Wert ohne Vorzeichen).
Hinweise	• Siehe auch Kapitel 6.9.1 "Freie Variablen", Seite 102.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532. Letzte Änderung (Version DLL 539, OUT 539): Der Wertebereich des Parameters `No` wurde auf [0…7] erhöht.
Verweise	**set_free_variable**, **set_free_variable_list**

Kontrollbefehl	get_galvo_controls
Funktion	Gibt die zu den Eingabewerten gehörigen Ansteuerwerte zurück.
Einschränkung	Der Befehl kann nur ausgeführt werden, wenn gerade keine Liste läuft (**get_last_error**-Returncode: `RTC5_BUSY`).
Aufruf	`get_galvo_controls( InPtr, OutPtr )`
Parameter	`InPtr` Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. als 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf ein Array von fünf 32-Bit-Werten ohne Vorzeichen, in dem die gewünschten Einstellungen angegeben sind: X, Y, Z, Defocus, Zoom. Überlaufende Werte werden auf die jeweiligen Randbereiche geclippt. `OutPtr` Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. als 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf ein Array von vier 32-Bit-Werten ohne Vorzeichen, in dem die zugehörigen Ansteuerwerte abgelegt werden sollen: XA, YA, XB, YB. Wertebereiche jeweils [–524.288 … 524.287].
Parameter-rückgabe	XA, YA, XB, YB bezeichnen die Ansteuerwerte der XY-Achsen für Scan-Köpfe A und B.
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 im Befehl **get_galvo_controls** angegeben sind (z. B. Matrix, Offset, Angle, etc.; auch Stretch-Table) werden so berücksichtigt, wie sie gerade eingestellt sind. Der Benutzer ist selbst dafür verantwortlich, diese vor Aufruf des Befehls **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 `RTC5_BUSY` erzeugt wurde.

Kontrollbefehl	get_galvo_controls
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen X- und Y-Werte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Alle zurückgegebenen Werte sind auch im RTC4-Kompatibilitätsmodus im RTC5-20-Bit-Bereich.
Versionsinfo	Verfügbar ab Version DLL 539, OUT 539.
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: Primärer Scan-Kopf-Anschluss. = 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich).
	`ParaNo`	Nummer des Parameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: 0-15 (Zuordnung: siehe Seite 140).
Rückgabe	Parameterwert im 64-Bit-IEEE-Gleitkommaformat (siehe Seite 140).	
Hinweise	• Die Parameterwerte können mit **get_table_para** aus einer aktuell geladenen und mit **get_head_para** aus einer zugewiesenen RTC5-Korrekturtabelle ausgelesen und auf diese Weise direkt in ein Anwenderprogramm eingebunden werden (siehe Seite 140). • Für unzulässige Parameterwerte für `HeadNo` und `ParaNo` ist der Rückgabewert 0 (**get_last_error**-Returncode: `RTC5_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-Tabelle zugewiesen ist. • Ist einem Kopf eine 3D-Tabelle zugewiesen, werden unabhängig von `HeadNo` die Parameter dieser 3D-Tabelle zurückgegeben (zwei 3D-Tabellen können nicht gleichzeitig zugewiesen sein). `HeadNo` muss dennoch 1 oder 2 sein (siehe vorheriger Hinweis).	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Letzte Änderung Version DLL 516, OUT 514.	

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 des primären Scan-Kopf-Anschlusses zurück (Byte #1 = Byte #0).
		= 2:	Gibt den Status des sekundären Scan-Kopf-Anschlusses (Freischaltung erforderlich) zurück (Byte #1 = Byte #0).
		Sonst:	Gibt den Status des primären Scan-Kopf-Anschlusses (Byte #0) *und* des sekundären Scan-Kopf-Anschlusses (Byte #1) zurück.
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.
		Bit #7	Power-Status, 1 = OK.
	Byte #1	Bits #8… #15	Bit-Zuordnung wie bei Byte #0.
	Bytes #2… #3	Bits #16… #31	0.
Hinweise	• Der Befehl **get_head_status** liest nicht das "normale" 20-Bit-Statuswort aus, sondern die vom SL2-100-Protokoll (auch über den XY2-100-Konverter) permanent übertragenen 6 zusätzlichen Statusbits (siehe Seite 143). Die Statusbits werden von **get_head_status** über Bit#2-7 und/oder Bit#10-15 zurückgegeben. Bit#0 und 8 werden von **get_head_status** unabhängig vom aktuellen Zustand des Scan-Systems als 1, sowie Bit#1 und 9 als 0 zurückgegeben. • Ist aktuell kein Scan-System angeschlossen oder nicht eingeschaltet, wird der Status des zuletzt angeschlossenen Systems zurückgeliefert. Zur Unterscheidung kann **get_startstop_info** (Bit #17 und/oder 25) verwendet werden. War nach einem Reset (Einschalten oder **load_program_file**) noch kein Scan-System angeschlossen (oder für `Head` = 2, falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist), liefert **get_head_status** den Wert 0. • Ist ein XY2-100-Konverter der Version 1.x allein angesteckt oder ein angeschlossenes Scan-System ist nicht eingeschaltet, liefert **get_head_status** ein 0xFD-Byte ("alles OK") zurück. Das PowerOK-Signal ist ein elektronisch generiertes Signal. Es kann daher nicht dafür eingesetzt werden, um zu prüfen, ob ein angeschlossener Scan-Kopf überhaupt eingeschaltet ist. • Mit einem XY2-100-Konverter der Version 2.0 wird der Wert 0 zurückgegeben, wenn kein Scan-Kopf angeschlossen oder dieser nicht eingeschaltet ist. • 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.		

Kontrollbefehl	get_head_status
Hinweise (Forts.)	• Bei i*DRIVE*-Scan-Systemen (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario$SCAN_{de}$) – ohne SL2-100-Interface liefert **get_head_status** nur dann brauchbare Rückgabewerte, wenn das XY2-100-Statuswort zur Rückübertragung ausgewählt wurde. – werden die Positionsrückmeldesignale der x- und y-Achse UND-verknüpft und als gemeinsames Signal (z.B. Bit#3,4) zurückgegeben. Die Positionsrückmeldesignale der x- und y-Achse können hier allerdings auch separat mit dem **control_command**-Befehl mit data = 0528_H an Flag Bit#12 ausgelesen werden. – kann es nach einem Neustart oder Reset des Scan-Systems ca. 5 Sekunden dauern, bis Daten vom Scan-System rückübertragen werden. • Statussignale können auch mit **get_value**, **get_values**, **set_trigger** und **set_trigger4** ausgelesen werden. • Siehe auch Kapitel 8.5 "Verwendung mehrerer verschiedener Korrekturtabellen", Seite 192 zur Verwendung zweier Scan-Köpfe.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: • Bei der RTC5 können (anders als bei der RTC4) gleichzeitig die Statusworte beider Scan-Kopf-Anschlüsse ausgelesen werden und auch (bei i*DRIVE*-Scan-Systemen) unabhängig von einem mit dem **control_command**-Befehl ausgewählten Rückübertragungssignal. • Bei i*DRIVE*-Scan-Systemen liefern Bit#0 und 1 (anders als bei der RTC4) keine Informationen über die Betriebsbereitschaft des Scan-Systems (siehe **get_value**).
Verweise	**get_value**, **get_values**, **set_trigger**, **set_trigger4**, **get_waveform**

Kontrollbefehl	get_hex_version
Funktion	Gibt die Versionsnummer der aktuell auf die RTC5 geladenen RTC5-*Software* (DSP-Programmdatei `RTC5OUT.out`) zurück.
Aufruf	`HexVersion = get_hex_version()`
Rückgabe	RTC5 Software-Versionsnummer als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Die Versionsnummern der Programmdateien liegen im Bereich 500 … 599. **get_hex_version** gibt die folgenden Werte zurück: – ohne freigeschaltete 3D-Option Werte im Bereich 2.500 … 2.599 (Versionsnummer + 2.000) – bei freigeschalteter 3D-Option Werte im Bereich 3.500 … 3.599 (Versionsnummer + 3.000) • Die Dateinamenerweiterung für RTC5-Programmdateien ist nicht mehr `*.hex` (wie bei der RTC4, RTC3 und RTC2), sondern `*.out` (siehe auch **load_program_file**). • Für ≥ DLL 535: Die Software-Versionsnummer kann auch nach einem `RTC5_VERSION_MISMATCH` und `RTC5_ACCESS_DENIED`-Fehler zurückgegeben werden. Der Rückgabewert ist 0, wenn noch kein Programm geladen ist. Dabei wird kein `RTC5_TIMEOUT`-Fehler mehr erzeugt.
RTC4→ RTC5	Unveränderte Funktionalität.
Versionsinfo	Letzte Änderung mit Version DLL 535 (siehe Hinweis).
Verweise	**get_dll_version**, **get_rtc_version**

Kontrollbefehl	get_hi_data
Funktion	Gibt die jeweils zuletzt gemessenen Home-In-Positionen zurück, die zuvor mit **auto_cal** für das am primären Scan-Kopf-Anschluss angeschlossene Scan-System ermittelt wurden.
Aufruf	`get_hi_data( &X1, &X2, &Y1, &Y2)`
Parameter-rückgabe	`X1, X2, Y1, Y2` Koordinaten der aktuell gespeicherten (zuletzt gemessenen) Home-In-Positionen in *Bits* als Pointer auf 32-Bit-Werte mit Vorzeichen.
Hinweise	• Der Befehl ist synonym zum Befehl **get_hi_pos** mit `HeadNo` = 1 (siehe Hinweise dort).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Die zurückgegebenen Werte liegen im 20-Bit-Wertebereich.
Verweise	**get_hi_pos**, **write_hi_pos**

Kontrollbefehl	get_hi_pos
Funktion	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: Primärer Scan-Kopf-Anschluss. = 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich).
Parameter-rückgabe	`X1, X2, Y1, Y2` Koordinaten der aktuell gespeicherten (zuletzt gemessenen) Home-In-Positionen in *Bits* als Pointer auf 32-Bit-Werte mit Vorzeichen.
Hinweise	• Zur Verwendung des Befehls siehe Abschnitt "Kundeneigene Kalibrierung", Seite 227. • Es obliegt den Benutzern, sicherzustellen, 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 oder bei der Ermittlung der Werte für ein solches System ein Fehler aufgetreten ist. Für `HeadNo` = 2 wird auch dann 0 zurückgegeben, falls die Option "Zweite-Scan-Kopf-Steuerung" nicht freigeschaltet ist. • Unmittelbar nach der Initialisierung (**init_rtc5_dll**), insbesondere bevor **auto_cal**(`Command` = 0, 1 oder 3) erstmals aufgerufen wurde, bedeuten die zurückgegebenen Werte die auf der RTC5-Karte (im EEPROM) gespeicherten Referenzwerte für die Home-In-Positionen. Sind diese noch nicht mindestens einmal mit **auto_cal**(`Command` = 0) erfolgreich bestimmt worden, wird 0 zurückgegeben. • Der Befehl **get_hi_pos** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Für unzulässige Parameterwerte wird für alle Koordinatenwerte 0 zurückgeliefert (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`).
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_hi_data**, **auto_cal**, **set_hi**, **write_hi_pos**

Kontrollbefehl	get_input_pointer
Funktion	Gibt die aktuelle (absolute) Position des Input-Pointers zurück.
Aufruf	`InputPointer = get_input_pointer()`
Rückgabe	Position des Input-Pointers [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Die Position des Input-Pointers entspricht der Position im RTC5-Listenspeicher (auch im geschützten Listenbereich "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 Listenspeicheradresse (Offset bzgl. Listenanfang von "Liste 1") zurück. Die relative Position bzgl. des Listenanfangs des jeweiligen Listenbereichs 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 Output-Pointers kann mit **get_status** oder **get_out_pointer** ausgelesen werden. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe "Error-Handling", Seite 98) werden von **get_input_pointer** weder erzeugt noch verändert.
RTC4→ RTC5	Unveränderte Funktionalität.
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.
Hinweise	• Dieser Befehl ist zur Verwendung in Kombination mit den Befehlen **set_io_cond_list** und **clear_io_cond_list** vorgesehen (siehe auch "Programmierbeispiele", Seite 245). • Siehe auch Abschnitt "16-Bit-Digital-Eingang und -Ausgang", Seite 51.
RTC4→ RTC5	Unveränderte Funktionalität.
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 und kopiert 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 Treiber-Fehler.
Hinweise	• Der Befehl 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 × 1024 mit 0 ≤ N < 1024 bedeuten. Die Werte können auch von Benutzern direkt so erzeugt oder manipuliert werden.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525.
Verweise	set_jump_table, load_jump_table_offset

Kontrollbefehl	get_lap_time
Funktion	Gibt den *aktuellen* Wert des RTC5-Timers zurück (ohne ihn auf Null zurückzusetzen).
Aufruf	`TimerValue = get_lap_time()`
Rückgabe	Wert des RTC5-Timers in Sekunden seit dem letzten Aufruf von save_and_restart_timer. Im 64-Bit-IEEE-Gleitkommaformat.
Hinweise	• Der Befehl **get_lap_time** dient dazu, die bereits verstrichene Zeit einer länger dauernden Markierung noch während der Laufzeit abzufragen.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 541, OUT 541.
Verweise	get_time, save_and_restart_timer

Kontrollbefehl	get_laser_pin_in
Funktion	Gibt den aktuellen Zustand der zwei digitalen Eingänge an der LASER-Buchse zurück.
Aufruf	`LaserPinIn = get_laser_pin_in()`
Rückgabe	32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) DIGITAL IN1. Bit #1 DIGITAL IN2. Bits #2-31 Reserviert.
Hinweise	• Die beiden Eingänge können von Benutzern frei verwendet werden.
RTC4→ RTC5	Neuer Befehl.
Verweise	set_laser_pin_out

Kontrollbefehl	get_last_error
Funktion	Gibt einen Fehlercode als Auflistung der beim zuletzt ausgeführten Befehl evtl. 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 wie beim Befehl **get_error**.
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Error-Handling", Seite 98. • Die Befehle **get_last_error** und **n_get_last_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe "Error-Handling", Seite 98) werden von **get_last_error** weder erzeugt noch verändert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_error**, **reset_error**, **set_verify**

Kontrollbefehl	get_list_pointer	
Funktion	Gibt die aktuelle (relative) Position des Input-Pointers sowie die Listennummer zurück.	
Aufruf	`get_list_pointer( &ListNo, &Pos )`	
Parameter-rückgabe	`ListNo`	Listennummer der Liste, in dem sich der Input-Pointer aktuell befindet, als Pointer auf einen 32-Bit-Wert ohne Vorzeichen [1…3].
	`Pos`	Aktuelle Position des Input-Pointers (Offset bzgl. des Listenanfangs der jeweiligen Liste) als Pointer auf einen 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Die absolute Listenspeicheradresse (Offset bzgl. Listenanfang von "Liste 1") des Input-Pointers kann mit **get_input_pointer** ausgelesen werden (siehe auch Hinweise dort). • Die Zahl der Listenplätze bis zum Ende der jeweiligen Liste (ab dem Input-Pointer) kann mit **get_list_space** ausgelesen werden. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **get_list_pointer** weder erzeugt noch verändert.	
RTC4→ RTC5	Neuer Befehl.	
Verweise	**get_input_pointer**, **get_list_space**	

Kontrollbefehl	get_list_serial
Funktion	Liefert die Nummer des zuletzt mit **select_serial_set_list** ausgewählten Seriennummernsatzes (bzw. nach **load_program_file** des Seriennummernsatzes 0) und die aktuelle Seriennummer dieses Seriennummernsatzes zurück.
Aufruf	`LastMarkedSerialNo = get_list_serial( &Set )`
Rückgabe	Seriennummer im 64-Bit-IEEE-Gleitkommaformat.
Parameter-rückgabe	`Set` Nummer des ausgewählten Seriennummernsatzes als Pointer auf einen 32-Bit-Wert ohne Vorzeichen.
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 des Befehls siehe Kapitel 7.5.2 "Markieren von Seriennummern", Seite 170.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**select_serial_set_list**, **get_serial**

Kontrollbefehl	get_list_space
Funktion	Liefert den freien Listen-Speicher, d. h. die Anzahl der noch ladbaren Listenbefehle von der aktuellen Position des Input-Pointers ab bis zur letzten Listenposition der jeweiligen Liste.
Aufruf	`ListSpace = get_list_space()`
Rückgabe	Zahl der freien Listenplätze als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Wenn gerade ein indiziertes Unterprogramm oder ein indizierter Zeichensatz in den geschützten Bereich ("Liste 3") geladen wird, liefert **get_list_space** den noch verfügbaren, geschützten Speicherplatz zurück (andernfalls befindet sich der Input-Pointer nicht im geschützten Bereich).
RTC4→ RTC5	Der Befehl **get_list_space** wurde bei der RTC3/RTC4 für den Ringpuffer-Modus zur Verfügung gestellt und lieferte dort den Abstand zwischen Input- und Output-Pointer. Der Ringpuffer-Modus wird nicht mehr von der RTC5 unterstützt. Die Position des Input-Pointers kann mit **get_input_pointer** oder **get_list_pointer** und die Position des Output-Pointers mit **get_status** oder **get_out_pointer** ausgelesen werden.
Verweise	**get_input_pointer**, **get_status**, **get_out_pointer**, **get_list_pointer**

Kontrollbefehl	get_marking_info
Funktion	Liefert Informationen über eine mögliche Bereichsüberschreitung während einer "Processing on the fly"-Korrektur und über fehlerhafte Encoder-Signale zurück. Die Funktion liefert außerdem die Fehlerbits der automatischen Lasersignalunterdrückung zurück.
Aufruf	`MarkingInfo = get_marking_info()`
Rückgabe	Fehlercode als 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) = 1: "Processing on the fly"-Unterschreitung in x-Richtung (x < –524.288). 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: ActivateFlyError: beim Aktivieren einer "Processing on the fly"-Korrektur mit **activate_fly_2d** oder **activate_fly_xy** ist ein Fehler aufgetreten (siehe auch Kapitel 8.7.7 "Synchronisation von "Processing on the fly"-Anwendungen", Seite 211). Bit #10 PosAck-Fehlerbit von Kopf A und Achse X. Bit #11 TempOK-Fehlerbit von Kopf A und Achse X. Bit #12 PowerOK-Fehlerbit von Kopf A und Achse X. Bit #13 PosAck-Fehlerbit von Kopf A und Achse Y. Bit #14 TempOK-Fehlerbit von Kopf A und Achse Y. Bit #15 PowerOK-Fehlerbit von Kopf A und Achse Y. Bit #16 = 1: Signal 1 des Encoder-Eingangs ENCODER X hat zu kurzen Abstand. Bit #17 = 1: Signal 2 des Encoder-Eingangs ENCODER X hat zu kurzen Abstand. Bit #18 = 1: Signal 1 des Encoder-Eingangs ENCODER Y hat zu kurzen Abstand. Bit #19 = 1: Signal 2 des Encoder-Eingangs ENCODER Y hat zu kurzen Abstand. Bit #20 = 1: Falsche Signalsequenz am Encoder-Eingang ENCODER X. Bit #21 = 1: Falsche Signalsequenz am Encoder-Eingang ENCODER Y. Bit #22 = 1: "Processing on the fly"-Unterschreitung in z-Richtung (z < –32.768). Bit #23 = 1: "Processing on the fly"-Überschreitung in z-Richtung (z > +32.767). 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).

Kontrollbefehl	get_marking_info
Rückgabe (Forts.)	Bit #26 PosAck-Fehlerbit von Kopf B und Achse X. Bit #27 TempOK-Fehlerbit von Kopf B und Achse X. Bit #28 PowerOK-Fehlerbit von Kopf B und Achse X. Bit #29 PosAck-Fehlerbit von Kopf B und Achse Y. Bit #30 TempOK-Fehlerbit von Kopf B und Achse Y. Bit #31 PowerOK-Fehlerbit von Kopf B und Achse Y. Die anderen Bits sind reserviert.
Hinweise	• Zur Verwendung des Befehls und der Fehlerbits#0…7 siehe Kapitel 8.7.9 "Überwachung der "Processing on the fly"-Korrektur", Seite 213. • Die Bereichsgrenzen Xmin, Xmax, Ymin, Ymax für den kundenspezifischen Überwachungsbereich (Bits#4…7) werden durch den Befehl **set_fly_limits** festgelegt. • Der Abstand der Encoder-Signale kann zu kurz sein, wenn ein Störsignal auftritt, ein schneller Richtungswechsel erfolgt oder die Frequenz insgesamt zu hoch ist. • Eine falsche Encoder-Signalsequenz liegt vor, wenn sich die beiden Signale 1 und 2 gleichzeitig ändern. Dann lässt sich die Zählrichtung nicht mehr detektieren. • Wenn kein externer Encoder verwendet wird, haben die entsprechenden Eingänge auf der MARKING ON THE FLY-Stiftleiste undefinierte Pegel. Es werden dann möglicherweise ständig Signalfehler für diese Eingänge detektiert (obere 16 Bit des Returnwertes), auch wenn simulierte Encoder verwendet werden. Wenn dies die Anwendung stört, sollten die Pins über Pullup-Widerstände auf einen definierten Signalpegel gezogen werden. • Die Fehlerbits#10…15 und 26…31 werden nur bei aktivierter automatischer Lasersignalunterdrückung im Fehlerfall gesetzt (siehe Abschnitt "Automatische Lasersignalunterdrückung bei Status-Fehlern des Scan-Systems", Seite 146). Bei auftretenden Fehlern werden immer diejenigen Fehlerbits (kumulativ) gesetzt, für die das zugehörige Statussignal einen Fehler meldet. Gegebenenfalls werden auch Fehlerbits gesetzt, deren Statussignale nicht mit **set_laser_control** für die automatische Lasersignalunterdrückung ausgewählt wurden. Alle Fehlerbits werden beim Auslesen mit **get_marking_info** zurückgesetzt. • Die Bereichsgrenzen Zmin, Zmax für den kundenspezifischen Überwachungsbereich (Bits#24…25) werden durch den Befehl **set_fly_limits_z** festgelegt. • Alle Fehlerbits werden bei der Initialisierung (mit **load_program_file**) zurückgesetzt. • Die Fehlerbits #22…23 werden auch durch **get_marking_info** zurückgesetzt, nicht dagegen die Fehlerbits #24…25. Letztere werden gegebenenfalls implizit durch die konditionalen Befehle zurückgesetzt und können auch explizit mit dem Befehl **clear_fly_overflow** zurückgesetzt werden.
RTC4→ RTC5	Unveränderte Funktionalität für bereits bei der RTC4 verwendete Info-Bits.
Versionsinfo	Letzte Änderungen: • Version DLL 525, OUT 527 (Bits#4…7). • Version DLL 527, OUT 529, RBF 519 (Bits#10…15 und 26…31). • Version DLL 531, OUT 532 (Bits#22…25). • Version DLL 536, OUT 536 (Bit #9).
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**, **if_not_activated**

Kontrollbefehl	get_master_slave
Funktion	Liefert den Master/Slave-Status der angesprochenen RTC5-Karte zurück.
Aufruf	`MasterSlaveStatus = get_master_slave()`
Rückgabe	Master/Slave-Status als 32-Bit-Wert ohne Vorzeichen, d.h. Information, ob am MASTER- oder SLAVE-Anschluss der angesprochenen RTC5-Karte eine weitere RTC5-Karte angeschlossen ist: Bit #0 (LSB) = 1: Am SLAVE-Anschluss ist eine Karte angeschlossen. Bit #1 = 1: Am MASTER-Anschluss ist eine Karte angeschlossen. Bit #2-31 = 0. und Information, ob die angesprochene Karte als Master-, Slave- oder Einzelkarte betrieben wird: = 0 Die Karte wird als Einzelkarte betrieben. = 1 Die Karte wird als Slave (ohne nachgeschaltete Slave-Karte) betrieben. = 2 Die Karte wird als Master betrieben. = 3 Die Karte wird als Slave (mit nachgeschalteter Slave-Karte) betrieben.
Hinweise	• Zur Verwendung Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93. • Der Befehl liefert den Status nur dann zurück, wenn die angesprochene RTC5-Karte zuvor mit **load_program_file** initialisiert wurde. Andernfalls wird 0 zurückgegeben.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 515, OUT 514, RBF 512.
Verweise	**sync_slaves**

Kontrollbefehl	get_mcbsp
Funktion	Liest den Eingabewert, der zuletzt vollständig über die McBSP/SPI-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.
Hinweise	• **get_mcbsp** ist synonym zu **read_mcbsp**(0) (siehe Hinweise dort).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Ab OUT 526 kommt es an der McBSP/SPI-Schnittstelle nicht mehr zu einem Stagnieren der Datenübertragung. **get_mcbsp** muss also nicht mehr aufgerufen werden, um so ein Stagnieren aufzulösen. Bis OUT 525 können je 10 µs-Takt (auch bei 16 MHz) nur maximal zwei vollständige Übertragungen bearbeitet werden. Das jeweils dritte Datenwort wird gegebenenfalls mit weiteren Datenworten überschrieben (die Übertragung stagniert). Bis OUT 525 muss deshalb gegebenenfalls ein Stagnieren der Datenübertragung erst durch einen zusätzlich ausgeführten **get_mcbsp**-Befehl aufgelöst werden, bevor mit einem erneuten **get_mcbsp** tatsächlich aktuelle Werte ausgelesen werden können. **get_mcbsp_list** erlaubt dies auch innerhalb einer Listenausführung, damit eine nachfolgende (mit McBSP/SPI-Signalen gesteuerte) “Processing on the fly”-Anwendung bereits von Beginn an mit aktuellen Daten versorgt wird. • Ab OUT 526 werden die nach einem **load_program_file** an der McBSP/SPI-Schnittstelle bereitgestellten Daten bereits direkt und permanent abgeholt (siehe auch Seite 56), bis OUT 525 dagegen nur wenn eine “Processing on the fly”-Korrektur mit Verwendung der McBSP/SPI-Schnittstelle aktiviert worden ist und aktuell eine Liste läuft.
Verweise	**read_mcbsp**

Unverzögerter kurzer Listenbefehl	get_mcbsp_list
Funktion	Bis OUT 525: liest die im Puffer der McBSP/SPI-Schnittstelle (Multi channel Buffered Serial Port, siehe auch Seite 54) anstehenden Daten (Eingabewert) aus und speichert sie intern. Ab OUT 526: keine Funktion, siehe Versions-Info.
Aufruf	`get_mcbsp_list()`
Hinweise	• Der Befehl hat keine Auswirkung auf das Anwenderprogramm.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Ab OUT 526 wird **get_mcbsp_list** nicht mehr benötigt. Aus Kompatibilitätsgründen für bestehende Anwendungen wird der Befehl weiterhin zur Verfügung gestellt und darf auch weiterhin ausgeführt werden. • Bis OUT 525 können die von der McBSP/SPI-Schnittstelle ausgelesenen und gespeicherten Daten intern für eine “Processing on the fly”-Anwendung mit Verwendung der McBSP/SPI-Schnittstelle verwendet werden (es erfolgt keine Rückgabe an das Anwenderprogramm). Siehe auch Versionsinfo unter **get_mcbsp**.
Verweise	**get_mcbsp**

Kontrollbefehl	get_out_pointer	
Funktion	Gibt die aktuelle (bzw. die letzte) Position des Output-Pointers als Offset bzgl. des Listen-Starts der jeweiligen Liste sowie die Nummer der jeweiligen Liste zurück.	
Aufruf	`get_out_pointer( &ListNo, &Pos )`	
Parameter-rückgabe	`ListNo`	Nummer der Liste (“Liste 1” oder “Liste 2”), in der sich der Output-Pointer aktuell befindet (oder zuletzt befand) als Pointer auf einen 32-Bit-Wert ohne Vorzeichen.
	`Pos`	Aktuelle (bzw. letzte) Position des Output-Pointers (relative Speicheradresse) als Pointer auf einen 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Der Befehl ruft den Befehl **get_status** auf (siehe Hinweise dort, insbesondere bzgl. “Liste 3”), und ermittelt aus der von diesem Befehl rückgegebenen absoluten Output-Pointer-Position die Listennummer und die relative Position in der Liste. Mit der von **get_out_pointer** rückgegebenen relativen Position und Listennummer kann beim alternierenden Wechsel die Output-Pointer-Position leichter mit der aktuellen Input-Pointer-Position verglichen werden, insbesondere bzgl. Listengleichheit.	
RTC4→ RTC5	Neuer Befehl.	
Verweise	**get_status**, **get_input_pointer**, **get_list_pointer**	

Kontrollbefehl	get_overrun
Funktion	Liefert 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.
Hinweise	• Siehe Abschnitt “Takt-Überlauf”, Seite 142.
RTC4→ RTC5	Neuer Befehl.

Kontrollbefehl	get_rtc_mode
Funktion	Gibt den aktuell eingestellten Betriebsmodus der RTC5-DLL zurück.
Aufruf	`DLLMode = get_rtc_mode()`
Rückgabe	DLL-Betriebsmodus als 32-Bit-Wert ohne Vorzeichen. = 4: RTC4-Kompatibilitätsmodus. = 5: RTC5-Modus (Standardeinstellung).
Hinweise	• Der DLL-Betriebsmodus kann mit **set_rtc4_mode** oder **set_rtc5_mode** eingestellt werden. Standardmäßig ist der RTC5-Modus voreingestellt. • Der Befehl **get_rtc_mode** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **get_rtc_mode** ist nicht als Multi-Board-Befehl verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe “Error-Handling”, Seite 98) werden von **get_rtc_mode** weder erzeugt noch verändert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_rtc4_mode**, **set_rtc5_mode**

Kontrollbefehl	get_rtc_version
Funktion	Gibt die Versionsnummer der RTC5 Firmware (`RTC5RBF.rbf`) und zusätzliche Informationen über freigegebene Optionen der RTC5-Karte zurück.
Aufruf	`RTCVersion = get_rtc_version()`
Rückgabe	RTC5-Versionsnummer als 32-Bit-Wert ohne Vorzeichen: Bit #0 (LSB) … Bit #7: Firmware-Version der RTC5-Karte. Bit #8 = 1: "Processing on the fly"-Option ist freigeschaltet (siehe Seite 199). Bit #9 = 1: Die Option "Zweite-Scan-Kopf-Steuerung" ist freigeschaltet (siehe Seite 43). Bit #10 = 1: Die 3D-Option ist freigeschaltet (siehe Seite 193). Bits #11…#15: Reserviert. Bits #16…#23: DSP-Versionsnummer. Bits #24…#31: Firmware-Subversion der RTC5-Karte.
Hinweise	• Die RTC5 Firmware-Versionsnummern liegen im Bereich 500 … 599. **get_rtc_version** (Bit#0…7) gibt Werte im Bereich 0 … 99 (Versionsnummer - 500) zurück. • Die aktuelle DSP-Versionsnummer ist 2 (0 und 1 sind ältere Versionen). • Die aktuelle Firmware-Subversion ist 0. • Für ≥ DLL 535: Die Firmware-Version kann auch nach einem `RTC5_VERSION_MISMATCH` und `RTC5_ACCESS_DENIED`-Fehler noch zurückgegeben werden. Der Rückgabewert ist 0, wenn noch kein Programm geladen ist. Dabei wird kein `RTC5_TIMEOUT`-Fehler mehr erzeugt.
RTC4→ RTC5	Unveränderte Funktionalität.
Versionsinfo	Änderung mit Version DLL 518, OUT 517: Bits #16…23. Letzte Änderung mit Version DLL 535: Bits #24…31 (siehe auch Hinweis).
Verweise	**get_hex_version**, **get_dll_version**

Kontrollbefehl	get_serial
Funktion	Liefert die aktuelle Seriennummer des mit **select_serial_set** ausgewählten Seriennummernsatzes (bzw. nach **load_program_file** des Seriennummernsatzes 0) zurück.
Aufruf	`CurrentSerialNo = get_serial()`
Rückgabe	Seriennummer im 64-Bit-IEEE-Gleitkommaformat.
Hinweise	• Zur Verwendung des Befehls siehe Kapitel 7.5.2 "Markieren von Seriennummern", Seite 170. • Der Befehl sollte nicht verwechselt werden mit **get_serial_number**, welcher die Seriennummer der RTC5-Karte zurückliefert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**select_serial_set**

Kontrollbefehl	get_serial_number
Funktion	Gibt die Seriennummer der aktiven RTC5-Karte zurück.
Aufruf	`RTCSerialNumber = get_serial_number()`
Rückgabe	RTC5-Seriennummer als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Die Seriennummern der eingebauten Karten werden durch **init_rtc5_dll** ermittelt und in der DLL zwischengespeichert. Dort werden sie dann von **get_serial_number** ausgelesen. • Der Befehl **get_serial_number** ist hilfreich, wenn mehrere RTC5-Karten in *einem* Computer benutzt werden (siehe Kapitel 6.6, Seite 92). Mit dem zugehörigen Multi-Board-Befehl **n_get_serial_number** kann die Zuordnung zwischen den bei der Initialisierung für die installierten RTC5-Karten vergebenen DLL-internen Nummern und den installierten Karten ermittelt werden. Die DLL-internen Nummern werden bei jeder Initialisierung eines Anwenderprogramms (siehe **init_rtc5_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 **rtc5_count_cards** ausgelesen werden. • Die Befehle **get_serial_number** und **n_get_serial_number** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **get_serial_number** weder erzeugt noch verändert.
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**rtc5_count_cards**

Kontrollbefehl	get_standby	
Funktion	Liefert die aktuell eingestellten Standby-Parameter zurück.	
Aufruf	`get_standby( &HalfPeriod, &PulseLength )`	
Parameter-rückgabe	`HalfPeriod`	Eingestellte *halbe* Standby-Ausgabeperiode der Standby-Pulse als Pointer auf einen 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*.
	`PulseLength`	Eingestellte Pulslänge der Standby-Pulse als Pointer auf einen 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*.
Hinweise	• Zur Verwendung siehe ""Laser standby"-Signale", Seite 145.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmode werden die Parameterwerte durch 8 dividiert. *1 Bit entspricht 1/8 µs*.	
Versionsinfo	Verfügbar ab Version DLL 515, OUT 514, RBF 512.	
Verweise	**set_standby**	

Kontrollbefehl	**get_startstop_info**
Funktion	Liefert Informationen über interne und externe Listen-Starts und -Stopps seit dem letzten Aufruf des Befehls, die aktuellen externen Start- und Stopp-Pegel, den Status und den Signal-Pegel der Lasersteuersignale, sowie evtl. Übertragungsfehler von und zum angeschlossenen Scan-System.
Aufruf	`StartStopInfo = get_startstop_info()`
Rückgabe	Infosignal als 32-Bit-Wert ohne Vorzeichen: Bit #0 (LSB) = 1: Seit dem letzten Aufruf von **get_startstop_info** wurde (mit **execute_list** o.ä.) ein *interner* Listen-Start ausgeführt. Bit #1 = 1: Seit dem letzten Aufruf von **get_startstop_info** wurde (über /START, /START2, /Slave-START, mit **simulate_ext_start** oder **simulate_ext_start_ctrl**) ein *externer* Listen-Start ausgeführt. Bit #2 = 1: Seit dem letzten Aufruf von **get_startstop_info** wurde (mit **stop_execution**) ein *interner* Listen-Stopp ausgeführt. Bit #3 = 1: Seit dem letzten Aufruf von **get_startstop_info** wurde (über /STOP, /STOP2, /Slave-STOP oder mit **simulate_ext_stop**) ein *externer* Listen-Stopp ausgeführt. Bit #4 Ext-Stop-Pegel (logische UND-Verknüpfung der Signale /STOP, /STOP2, /Slave-STOP und **simulate_ext_stop**): = 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. Bits #5…8 Reserviert. Bit #9 = 1: Die Lasersteuersignale für den "Laser active"-Betrieb sind freigegeben (siehe auch **set_laser_control**, **enable_laser** und **disable_laser**). Bit #10 = 1: Die TTL-Lasersteuersignale an den LASER1- und LASER2-Ausgängen sind *low-aktiv* (der Signal-Pegel kann mit **set_laser_control** eingestellt werden). Bit #11 = 1: Seit dem letzten Aufruf von **get_startstop_info** ist mindestens ein externer Listen-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-Pegel (logische UND-Verknüpfung der Signale /START, /START2 und /Slave-START): = 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 Signal-Pegel kann mit **set_laser_control** eingestellt werden). Bits #14…15 Reserviert.

Kontrollbefehl	get_startstop_info	
	Bits #16 …31	Fehlerbits, die gesetzt werden, wenn bei der Datenübertragung vom Scan-System ein Fehler auftritt (Bits #16-23 für den primären Scan-Kopf-Anschluss, Bits #24-31 für den sekundären Scan-Kopf-Anschluss): Bits #16,24: Anzahl Frames innerhalb eines Datenblockes fehlerhaft. Bits #17,25: Pulslänge des vom Scan-System empfangenen Signals unzulässig, möglicherweise kein Scan-System angeschlossen. Bits #18,26: Preamble-Folge unzulässig. Bits #19,27: Bitanzahl innerhalb eines Subframes fehlerhaft. Bits #20,28: Parityfehler beim Lesen der vom Scan-System empfangenen Daten. Bits #21,29: Die anliegenden Daten sind ungültig (veraltet). Bits #22,30: Reserviert. Bits #23,31: Reserviert.
Hinweise	• Die Info-Bits #0 … #3 sowie die Fehlerbits #16 - #31 werden nach der Ausführung des Befehls zurückgesetzt. • Siehe auch "Externer Listen-Stopp", Seite 239 und "Externer Listen-Start", Seite 240.	
RTC4→ RTC5	Unveränderte Funktionalität für bereits bei der RTC4 verwendete Info-Bits.	
Verweise	**get_counts**, **get_status**, **set_control_mode**	

Kontrollbefehl	get_status	
Funktion	Gibt die aktuellen Listenausführungsstatuswerte (BUSY, INTERNAL-BUSY und PAUSED) und die aktuelle (bzw. die letzte) Position des Output-Pointers zurück.	
Aufruf	`get_status( &Status, &Pos )`	
Parameter-rückgabe	`Status`	Statuswert als Pointer auf einen 32-Bit-Wert ohne Vorzeichen. Bit #0 (LSB) = 1: BUSY-Status gesetzt. Bit #1 … Bit #6 Reserviert. Bit #7 = 1: INTERNAL-BUSY-Status gesetzt. Bit #8 … Bit #14 Reserviert. Bit #15 = 1: PAUSED-Status gesetzt. Bit #16 … Bit #31 0.
	`Pos`	Aktuelle (bzw. letzte) Position des Output-Pointers (absolute Speicheradresse) als Pointer auf einen 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Wann die BUSY, INTERNAL-BUSY- oder PAUSED-Statuswerte gesetzt oder nicht gesetzt sind, wird unter Kapitel 6.4.3 "Listenausführungsstatus", Seite 77 beschrieben. • Für (BUSY und PAUSED gesetzt) muss mit **restart_list** fortgefahren werden, für (BUSY nicht gesetzt und PAUSED gesetzt) mit **release_wait** und für (BUSY und PAUSED nicht gesetzt) mit **execute_list_pos**. Für (BUSY gesetzt und PAUSED nicht gesetzt) läuft aktuell eine Liste, ein "Fortfahren" ist nicht erlaubt. Ein falsches Fortfahren erzeugt den **get_last_error**-Returncode `RTC5_BUSY`. Für (INTERNAL-BUSY gesetzt) werden **release_wait** und **execute_list_pos** gegebenenfalls erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Status wieder zurückgesetzt wurde). • Der Output-Pointer zeigt auf den Listenbefehl (in "Liste 1" oder "Liste 2"), der momentan ausgeführt wird bzw. zuletzt ausgeführt wurde. Wenn die Output-Pointer-Position `Pos` während der Abarbeitung eines Unterprogramms im geschützten Speicherbereich ("Liste 3") abgefragt wird, so wird die Position *des* Listenbefehls aus dem Listen-Bereich ("Liste 1" oder "Liste 2") zurückgegeben, in der sich der Output-Pointer zuletzt befunden hat (in der Regel von wo aus das Unterprogramm z.B. mit **list_call** aufgerufen wurde). Die Befehle **pause_list** und **set_wait** lassen `Pos` unverändert. • **get_status** liefert die Output-Pointer-Position als absolute Speicheradresse (Offset bzgl. Listenanfang von "Liste 1") zurück. Die relative Position bzgl. des Listenanfangs des jeweiligen Listenbereichs kann mit **get_out_pointer** ausgelesen werden. • Die aktuelle Input-Pointer-Position kann mit **get_input_pointer** oder **get_list_pointer** ausgelesen werden. • Die Listenstatuswerte für die einzelnen Listen (siehe Kapitel 6.4.2 "Listenstatus", Seite 76) können mit **read_status** ausgelesen werden. • Die Befehle **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**-Befehl). • Wenn noch kein Programm (mit **load_program_file**) geladen ist, liefert **get_status** undefinierte Werte zurück. • Zum Auslesen der Statussignale der *Scan-Köpfe* dient der Befehl **get_head_status**.	

Kontrollbefehl	get_status
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Bei der RTC5 liefert der Parameter `Status` den BUSY-, INTERNAL-BUSY- *und* PAUSED-Status, bei der RTC4 nur den BUSY-Status.
Verweise	**read_status**, **get_input_pointer**, **get_list_pointer**, **get_out_pointer**

Kontrollbefehl	get_stepper_status	
Funktion	Liefert für beide Schrittmotor-Ausgänge den jeweils aktuellen Status der Schrittmotorsignale, die aktuell eingestellte CLOCK-Pulsperiode, den Busy- und Init-Status sowie den aktuellen Wert der internen Positionsvariablen zurück.	
Aufruf	`get_stepper_status( &Status1, &Pos1, &Status2, &Pos2 )`	
Parameter-rückgabe	`Status1`, `Status2`	Aktueller Status des Schrittmotor-Ausgangs 1 bzw. 2 als Pointer auf 32-Bit-Werte ohne Vorzeichen. Bit #0 (LSB): ENABLE-Signal. Bit #1: DIRECTION-Signal. Bit #2: CLOCK-Signal. Bit #3: SWITCH-Signal. Bit #4: Busy-Status. Bit #5: Init-Status. Bits #6…7: Reserviert. Bits #8…31: CLOCK-Pulsperiode (24-Bit-Wert).
	`Pos1`, `Pos2`	Aktueller Wert der internen Positionsvariablen von Schrittmotor-Ausgang 1 bzw. 2 als Pointer auf 32-Bit-Werte mit Vorzeichen.
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoransteuerung", Seite 234. • Wenn das SWITCH-Signal (Endschalter-Signal) gesetzt ist (LOW), werden keine CLOCK-Pulse mehr erzeugt. • Der Busy-Status zeigt an, dass eine zuvor (mit **stepper_abs**, **stepper_rel**, etc.) initiierte Zielpositionsfahrt noch nicht abgeschlossen ist. • Der Init-Status zeigt an, dass eine zuvor (mit **stepper_init**) initiierte Referenzfahrt noch nicht abgeschlossen ist.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).	

Kontrollbefehl	**get_sub_pointer**
Funktion	Liefert 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.
Hinweise	• Der Befehl **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 Speicherbereich geladen wurde oder ein ungeschütztes Unterprogramm nur nachträglich referenziert wurde, liegt die rückgelesene Speicheradresse im geschützten oder ungeschützten Listenbereich. • 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. • Der Befehl 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→ RTC5	Neuer Befehl.
Verweise	**get_char_pointer**, **get_text_table_pointer**

Kontrollbefehl	get_sync_status
Funktion	Liefert den Master/Slave-Synchronisierungszustand der angesprochenen RTC5-Karte zurück.
Aufruf	`MasterSlaveSyncStatus = get_sync_status()`
Rückgabe	Master/Slave-Synchronisierungszustand [0 … 640] als 32-Bit-Wert ohne Vorzeichen. < 11: Die Karte ist synchron zur Master-Karte (bzw. ihrer Vorgängerkarte in der Master/Slave-Kette). = 640: Die angesprochene Karte wird als Master-Karte betrieben.
Hinweise	• Zur Verwendung siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93. • Falls die angesprochene RTC5-Karte nicht zuvor mit **load_program_file** initialisiert wurde, liefert **get_sync_status** 0 zurück. • Die Karten der Master/Slave-Kette sollten vor **get_sync_status** mit dem Befehl **sync_slaves** synchronisiert worden sein. • Damit **get_sync_status** tatsächlich den aktuellen Master/Slave-Synchronisierungszustand zurückliefert, muss zuvor auf der Master-Karte (per externem Start-Signal oder per Befehl) ein externer Listen-Start ausgelöst worden sein (siehe Abschnitt "Externer Listen-Start", Seite 240). Dieser Listen-Start löst dann bei allen Slave-Karten eine Messung des Zeitunterschieds zwischen dem jeweiligen /SLAVE-Startpuls und dem jeweiligen 10 µs-Takt aus. Dieser Zeitunterschied wird als Master/Slave-Synchronisierungszustand (in Einheiten von 15 ns) auf der jeweiligen Karte gespeichert, bleibt dort gespeichert, bis an der Master-Karte ein neuer externer Listen-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 Seite 93): Master/Slave-Synchronisierungszustand < 11. Typischerweise beträgt der Master/Slave-Synchronisierungszustand 7. • Im nicht explizit synchronisierten Zustand (vor **sync_slaves**) kann der Synchronisierungszustand dennoch zufällig < 11 sein. Die Karten verhalten sich dann wie synchronisierte Karten.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version RBF 523, DLL 531, OUT 532.
Verweise	**sync_slaves**, **get_master_slave**

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…4].
	`ParaNo`	Nummer des Parameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: 0-15 (Zuordnung: siehe Abschnitt "Datei-Kopf von Korrekturdateien", Seite 140).
Rückgabe	Parameterwert im 64-Bit-IEEE-Gleitkommaformat (siehe Abschnitt "Datei-Kopf von Korrekturdateien", Seite 140).	
Hinweise	• Die Parameterwerte können mit **get_table_para** aus einer aktuell geladenen und mit **get_head_para** aus einer zugewiesenen RTC5-Korrekturtabelle ausgelesen und auf diese Weise direkt in ein Anwenderprogramm eingebunden werden (siehe Abschnitt "Datei-Kopf von Korrekturdateien", Seite 140). • Liegen die Parameter `TableNo` und `ParaNo` im unzulässigen Bereich, ist der Rückgabewert 0 (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`). • Wenn noch keine Korrekturtabelle mit der angegebenen Nummer geladen wurde, sind die Parameterwerte undefiniert.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Letzte Änderung (Version DLL 533, OUT 534): Der zulässige Wertebereich für Parameter `TableNo` wurde auf [0…4] erhöht (mit **load_correction_file** können bis zu 4 Korrekturdateien geladen werden).	

Kontrollbefehl	get_text_table_pointer
Funktion	Liefert 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.
Hinweise	• Der Befehl **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 Speicherbereich geladen wurde oder ein ungeschütztes Unterprogramm nur nachträglich referenziert wurde, liegt die rückgelesene Speicheradresse im geschützten oder ungeschützten Listenbereich. • 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. • Der Befehl 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 konditionale 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 der Befehl **list_call_cond** ignoriert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_char_pointer**, **get_sub_pointer**

Kontrollbefehl	get_time
Funktion	Gibt den Wert des RTC5-Timers zurück (ohne ihn auf Null zurückzusetzen). Er wurde beim letzten Aufruf des Befehls **save_and_restart_timer** gespeichert.
Aufruf	`TimerValue = get_time()`
Rückgabe	Wert des RTC5-Timers in Sekunden. Im 64-Bit-IEEE-Gleitkommaformat.
Hinweise	• Siehe **save_and_restart_timer**. • Wieviele Listenbefehlstakte seit dem Zurücksetzen auf Null vergangen sind, kann mit **get_lap_time** abgefragt werden.
RTC4→ RTC5	Unverändert.
Verweise	**get_lap_time**, **save_and_restart_timer**

Kontrollbefehl	**get_transform**		
Funktion	Überträgt Positionswerte, die mit **set_trigger** aufgenommen und auf der RTC5 gespeichert wurden, auf den PC und führt eine Rücktransformation dieser Positionswerte durch.		
Aufruf	`get_transform( Number, Ptr1, Ptr2, Ptr, Code )`		
Parameter	`Number`	Anzahl [1…2^{20}] der rückzutransformierenden Positionswerte als 32-Bit-Wert ohne Vorzeichen; es werden die Messwerte mit Index 0 bis (`Number`–1) rücktransformiert.	
	`Ptr1`, `Ptr2`	Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. 32-Bit- bzw. 64-Bit-Werte ohne Vorzeichen) auf die beiden Arbeitsspeicherbereiche des PCs, an die die rücktransformierten Werte übertragen werden sollen (siehe auch `Code`).	
	`Ptr`	Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, in den die Korrektur- und Transformationseinstellungen, die zur Rücktransformation verwendet werden sollen, zuvor mit **upload_transform** übertragen wurden.	
	`Code`	Dieser Parameter (32-Bit-Wert ohne Vorzeichen) steuert Details der Rücktransformation, insbesondere welche Teiltransformationen durchgeführt werden: Soll eine Teiltransformationen *nicht* durchgeführt werden, so muss das entsprechende Bit (#2…#5) auf 1 gesetzt werden. Der Parameter hat die gleiche Bedeutung wie bei **transform** (`Ptr1` entspricht `Sig1` und `Ptr2` `Sig2`). Für Bit #0 = 0 werden die mit **set_trigger** aufgenommenen Messwertpaare als xy-Koordinaten rücktransformiert:	
		Bit #1	= 0: Die über Speicherkanal 1 (`Signal1`) aufgenommenen Werte werden unter `Ptr1` auf den PC übertragen und als x-Koordinaten rücktransformiert. Die über Speicherkanal 2 (`Signal2`) aufgenommenen Werte werden unter `Ptr2` auf den PC übertragen und als y-Koordinaten rücktransformiert.
			= 1: Die über Speicherkanal 1 (`Signal1`) aufgenommenen Werte werden unter `Ptr1` auf den PC übertragen und als y-Koordinaten rücktransformiert. Die über Speicherkanal 2 (`Signal2`) aufgenommenen Werte werden unter `Ptr2` auf den PC übertragen und als x-Koordinaten rücktransformiert.
		Bit #2	= 0: Die Gain/Offset-Korrektur der automatischen Selbstkalibrierung wird rücktransformiert.
		Bit #3	= 0: Die Bildfeldkorrektur wird rücktransformiert.
		Bit #4	= 0: Der Offset der definierten Koordinatentransformation wird rücktransformiert.
		Bit #5	= 0: Die Gesamtmatrix der definierten Koordinatentransformation wird rücktransformiert.
		Bits #6…#31	Reserviert.

Kontrollbefehl	get_transform		
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** aufgenommen wurden, als z-Koordinaten rücktransformiert:	
		Bit #1 = 0:	Die über Speicherkanal 1 (`Signal1`) aufgenommenen Werte werden unter `Ptr1` auf den PC übertragen und als z-Koordinaten rücktransformiert. Die über Speicherkanal 2 (`Signal2`) aufgenommenen Werte werden untransformiert unter `Ptr2` auf den PC übertragen.
		= 1:	Die über Speicherkanal 2 (`Signal2`) aufgenommenen Werte werden unter `Ptr1` auf den PC übertragen und als z-Koordinaten rücktransformiert. Die über Speicherkanal 1 (`Signal1`) aufgenommenen 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 rücktransformiert.
		Bit #3 = 0:	Die ABC-Korrektur wird rücktransformiert.
		Bit #4 = 0:	Der mit **set_offset_xyz** oder **set_offset_xyz_list** definierte Offset auf die z-Koordinate wird rücktransformiert.
		Bits #5…#31	Reserviert.
Hinweise	• Zur Rücktransformation von Positionswerten siehe Kapitel 8.1.3 "Überwachung der Positionierung", Seite 174. • **get_transform**(`Number,Ptr1,Ptr2,Ptr,Code`) überträgt die mit **set_trigger** aufgezeichneten Datenpaare mit **get_waveform**(1,`Number,Ptr1`) und **get_waveform**(2,`Number,Ptr2`) auf den PC und überschreibt die Daten paarweise mit **transform**(`Sig1,Sig2,Ptr,Code`). • Vor der Ausführung von **get_transform** muss **upload_transform** ausgeführt worden sein. Außerdem sollten mit **set_trigger** Positionswerte aufgezeichnet worden sein. • Vor dem Aufruf von **get_transform** kann (wie bei **get_waveform**) mit **measurement_status** überprüft werden, ob noch eine mit **set_trigger** gestartete Messung läuft. Während einer noch aktiven Datenaufzeichnung sollten möglichst keine Daten ausgelesen werden. Zusätzlich kann mit **measurement_status** die Anzahl `Pos` der bei der letzten (bzw. aktuellen) Messung aufgezeichneten Datenpaare ausgelesen werden. Es sollten maximal `Pos`+1 Datenelemente ausgelesen werden. Alle weiteren Datenelemente stammen aus früheren Aufzeichnungen oder der Initialisierung. • Die Arbeitsspeicherbereiche unter `Ptr1` und `Ptr2` müssen von Benutzern in ausreichender Größe (`Number` × 4 Byte je Kanal) zur Verfügung gestellt werden. • Wenn nur z-Positionswerte rücktransformiert werden sollen (`Code` Bit #0 = 1), so kann für den nicht benötigten Rückkanal der Pointer (`Ptr1` oder `Ptr2`) auf NULL gesetzt werden. Für diesen Kanal werden dann keine Daten auf den PC übertragen und rücktransformiert. Achten Sie dabei auf ein passend gesetztes `Code` Bit #1.		

Kontrollbefehl	get_transform
Hinweise (Forts.)	• Erfolgt der Befehlsaufruf mit `Ptr1` = NULL ***und*** `Ptr2` = NULL, so werden für beide Kanäle keine Daten an den PC übertragen und auch keine Rücktransformation durchgeführt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). Das gleiche gilt für `Number` = 0 oder `Number` > 2^{20}. • Für `Ptr` = NULL wird keine Rücktransformation 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 `Ptr` ≠ NULL, falls bei der Ausführung von **get_transform** ein Fehler auftritt (beispielsweise wenn unter `Ptr` ungültige oder fehlerhafte Daten stehen oder wenn eine Z-Achsen-Invertierung nicht möglich ist). Hier wird dann allerdings zusätzlich ein **get_last_error**-Returncode `RTC5_PARAM_ERROR` erzeugt. • Mit **set_trigger** erfasste und mit **get_transform** rücktransformiert auf dem PC gespeicherte Werte können bei Bedarf zusätzlich mit **get_waveform** untransformiert auf den PC übertragen werden. • Wird eine Rücktransformation 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 bzw. die z-Koordinate mit 0 und die Werte A, B, C mit 0, 1, 0 initialisiert (1zu1-Rücktransformation). • Bei der Rücktransformation 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 `RTC5_SEND_ERROR`.
RTC4→ RTC5	Neuer Befehl. Alle auf den PC übertragenen Werte (auch die z-Werte) sind auch im RTC4-Kompatibilitätsmodus im RTC5-20-Bit-Bereich und müssen gegebenenfalls von Benutzern selbst umgerechnet (durch 16 dividiert) werden.
Versionsinfo	Verfügbar ab Version DLL 516, OUT 515, RBF 512. Änderung Version DLL 517: Parameter `Code` geändert.
Verweise	**upload_transform**, **transform**, **set_trigger**, **get_waveform**

Kontrollbefehl	get_value
Funktion	Liefert 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.
Hinweise	• Die auswählbaren Signaltypen sind identisch wie beim Befehl **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 `RTC5_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 i*DRIVE*-Scan-Systemen (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario$SCAN_{de}$) kann es nach einem Neustart oder Reset des Scan-Systems ca. 5 Sekunden dauern, bis korrekte Daten vom Scan-System rückübertragen werden. Wenn das XY2-100-Statuswort zur Rückübertragung ausgewählt wurde, kann eine Überprüfung der Betriebsbereitschaft über Bit#4 und Bit#5 dieses 20-Bit-Wertes durchgeführt werden: (erst) bei korrekter Rückübertragung ist Bit#4 = 1 und Bit#5 = 0. • 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 RTC5 zurückgegeben werden (`Status<AX…BY>`), dann achten Sie beim Auswerten auf den Wertebereich des zurückgegebenen Datentyps (siehe hierzu die Befehlsbeschreibung zum **control_command**-Befehl): – 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 RTC5 rückgemeldet werden (Bits #0…3 = 0) werden die relevanten Informationen über die Bits #4…19 übertragen. Da **get_value** bei diesen Datentypen für Bits #20…31 nicht nur 0, sondern – abhängig vom Bit #19 des eigentlichen 16-Bit-Statuswerts – gegebenenfalls auch den Wert 1 an den PC zurückgibt, sollten hier ausschließlich die Bits #4…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 RTC5 rückgemeldet werden, können dagegen bedenkenlos durch Übernahme des kompletten von **get_value** zurückgegebenen 32-Bit-Wertes mit Vorzeichen ausgewertet werden (siehe auch Beispiel-Code unten). – Obwohl Z-Werte in den 3D-Befehlsparametern immer als 16-Bit-Werte spezifiziert werden müssen, liefern SCANLAB-Z-Achsen und 3-Achsen-Scan-Systeme (und entsprechend auch der **get_value**-Befehl) Z-Werte immer als 20-Bit-Werte mit Vorzeichen zurück. Diese zurückgegebenen Z-Werte müssen bei Bedarf vom Anwenderprogramm konvertiert (durch 16 geteilt) werden.

Kontrollbefehl	get_value

Beispiel (C/C++)	Auslesen von unterschiedlichen Datentypen (jeweils prim. Scan-Kopf-Anschluss, Achse X):
	a) XY2-100-Statuswort, PowerOK-Status
	`UINT statusword, powerOK;`
	`control_command (1, 1, 0x0500); // nur bei iDRIVE-Systemen sinnvoll`
	`statusword = (get_value(1) & 0x000FFFF0) >> 4;`
	`powerOK = (statusword & 0x00000080);`
	b) Seriennummer (nur bei i*DRIVE*-Systemen möglich)
	`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) Ist-Position (nur bei i*DRIVE*-Systemen möglich)
	`long real_position;`
	`control_command (1, 1, 0x0501);`
	`real_position = get_value(1);`

RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Alle zurückgegebenen Werte sind auch im RTC4-Kompatibilitätsmodus im RTC5-20-Bit-Bereich, werden jedoch als 32-Bit-Werte an den PC übertragen und müssen von Benutzern geeignet ausgewertet werden (siehe oben).
Versionsinfo	Letzte Änderung (DLL 536, OUT 536): Weitere auswählbare Datensignale (`Signal` = 43 und 44, siehe **set_trigger**).
Verweise	**get_values**, **set_trigger**, **get_waveform**, **get_head_status**

Kontrollbefehl	get_values
Funktion	Liefert die aktuellen Werte von bis zu vier ausgewählten Signalen zurück.
Aufruf	`get_values( SignalPtr, ResultPtr )`
Parameter	`SignalPtr` Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf ein Array von vier 32-Bit-Werten ohne Vorzeichen, in dem die gewünschten Signaltypen angegeben sind.
	`ResultPtr` Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf ein Array von vier 32-Bit-Werten mit Vorzeichen, in dem die aktuellen Werte der bis zu vier ausgewählten Signale abgelegt werden sollen.
Hinweise	• Es können bis zu vier beliebige Signale gleichzeitig ausgelesen werden. Die auswählbaren Signaltypen sind identisch wie beim Befehl **set_trigger** (für den zulässigen Wertebereich, die Signaltypen und entsprechende Hinweise siehe dort). Die gewünschten 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 der **get_value**-Befehl (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 `RTC5_PARAM_ERROR` wird allerdings nur dann erzeugt, wenn alle vier angegebenen Signaltypwerte ungültig sind. • Ist mindestens einer der Zeigerparameter gleich NULL, wird der Befehl nicht ausgeführt (alle Rückgabewerte sind 0) und ein **get_last_error**-Returncode `RTC5_PARAM_ERROR` erzeugt.
RTC4→ RTC5	Neuer Befehl. Die vier zurückgegebenen Werte sind auch im RTC4-Kompatibilitätsmodus immer im RTC5-20-Bit-Bereich, werden jedoch als 32-Bit-Werte an den PC übertragen und müssen von Benutzern geeignet ausgewertet werden (siehe Hinweise bei **get_value**).
Versionsinfo	• Verfügbar ab Version DLL 516, OUT 515, RBF 512. • Letzte Änderung (DLL 536, OUT 536): Weitere auswählbare Datensignale (siehe **set_trigger**).
Verweise	**get_value**, **set_trigger**, **transform**

Kontrollbefehl	get_wait_status
Funktion	Gibt den Unterbrechungs-Status der RTC5-Karte zurück.
Aufruf	`WaitStatus = get_wait_status()`
Rückgabe	Unterbrechungs-Status als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Wenn die Ausführung an einem Unterbrechungspunkt (wait marker) gestoppt wurde, dann liefert **get_wait_status** die zugehörige Nummer zurück (siehe **set_wait**). • Falls kein Unterbrechungspunkt angetroffen wurde, dann gibt **get_wait_status** den Wert Null zurück. • Die Ausführung der Liste wird durch den Aufruf des Befehls **release_wait** fortgesetzt.
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**set_wait**, **release_wait**

Kontrollbefehl	get_waveform	
Funktion	Überträgt die Messwerte, die mit **set_trigger** oder **set_trigger4** aufgenommen und auf der RTC5 gespeichert wurden, an den PC.	
Aufruf	`get_waveform( Channel, Number, Ptr )`	
Parameter	`Channel`	Messkanal [1 oder 2, bei mit **set_trigger4** gestarteten Aufzeichnungen auch 3 oder 4] als 32-Bit-Wert ohne Vorzeichen.
	`Number`	Anzahl [1…2^{20} bei **set_trigger**, 1…2^{19} bei **set_trigger4**] der zu übertragenden Werte als 32-Bit-Wert ohne Vorzeichen Es werden die Messwerte von der Position 0 bis (`Number`–1) übertragen.
	`Ptr`	Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. als 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, an den die Messwerte übertragen werden sollen.
Hinweise	• In den folgenden Fällen werden keine Daten übertragen (**get_last_error**-Returncode `RTC5_PARAM_ERROR`): – Für `Number` = 0. – Für `Number` > 2^{20}. – Für `Number` > 2^{19}, falls `Channel` = 3 oder 4 ausgewählt wurde oder falls mit **load_correction_file** eine Korrekturdatei als Korrekturtabelle `No`=3 oder `No`=4 geladen wurde. – Für `Ptr` = NULL. • Im Falle einer fehlerhaften PCI-Übertragung folgt ein **get_last_error**-Returncode `RTC5_SEND_ERROR`. • Vor dem Aufruf von **get_waveform** kann mit **measurement_status** überprüft werden, ob noch eine mit **set_trigger** oder **set_trigger4** gestartete Messung läuft. Während einer noch aktiven Datenaufzeichnung sollten möglichst keine Daten ausgelesen werden. Zusätzlich kann mit **measurement_status** die Anzahl `Pos` der bei der letzten (bzw. aktuellen) Messung aufgezeichneten Datenpaare ausgelesen werden. Es sollten maximal `Pos`+1 Datenelemente ausgelesen werden. Alle weiteren Datenelemente stammen aus früheren Aufzeichnungen oder der Initialisierung.	
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Alle Werte sind auch im RTC4-Kompatibilitätsmodus immer im RTC5-20-Bit-Bereich, werden jedoch als 32-Bit-Werte an den PC übertragen und müssen von Benutzern geeignet ausgewertet werden (siehe Hinweise bei **get_value**).	
Versionsinfo	Letzte Änderung (DLL 536, OUT 536): `Channel` = 3 und 4.	
Verweise	**set_trigger**, **set_trigger4**, **get_value**, **get_values**	

Kontrollbefehl	get_z_distance
Funktion	Liefert den Fokuslängenwert *l* für den angegebenen Punkt im Arbeitsvolumen zurück.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), liefert der Befehl den Wert 0 zurück und hat sonst keine Wirkung.
Aufruf	`ZDistance = get_z_distance( X, Y, Z )`
Parameter	`X, Y, Z` Absolute Koordinaten des Punkts `(x|y|z)` im Arbeitsvolumen in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–524.288 … 524.287]. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf die jeweiligen Randbereiche geclippt.
Rückgabe	Fokuslängenwert als 32-Bit-Wert mit Vorzeichen [–32.768 … 32.767].
Hinweise	• Der Befehl wird nur zur Neu-Kalibrierung der Z-Achse in einem 3-Achsen-Scan-System benötigt (siehe Seite 196). Der (einheitenlose) Fokuslängenwert *l* entspricht dem Fokuslängenunterschied zwischen dem angegebenen Punkt `(x|y|z)` und dem Punkt `(0|0|0)`. Der Fokuslängenwert kann positiv oder negativ sein. • Der Befehl führt zunächst einen (virtuellen) Sprung zum Punkt `(x|y|z)` aus und liefert dann den Fokuslängenwert zurück. • Der Befehl bleibt unwirksam und liefert den Wert 0 zurück (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen normal ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Zur 3D-Bildfeldkalibrierung siehe Kapitel 8.6.3 "3D-Markierbefehle", Seite 194.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich für `X` und `Y`). Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
Verweise	**load_z_table**

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 zweidimensionalen Bildfeld.
Aufruf	`goto_xy( X, Y )`
Parameter	`X`, `Y` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–524.288 … 524.287]. Überlaufende Werte werden auf den Randbereich geclippt.
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 pro 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. • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141). Ab Version DLL 520, OUT 519 werden zuvor angesammelte Koordinatentransformationen gegebenenfalls erst durch **goto_xy** wirksam (siehe Seite 185, `at_once` = 2). • Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Der Befehl wird nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn aktuell ein externes Stoppsignal (/STOP, /STOP2 oder /Slave-STOP) anliegt. • Der Befehl **goto_xy** setzt den INTERNAL-BUSY-Status für die Dauer seiner Ausführung. • Ab Version DLL 527, OUT 529 kehrt der Befehl erst zur Anwendung zurück, wenn die Mikrovektorisierung abgearbeitet ist. Während der Ausführung von Sprüngen mit **goto_xy** ist daher die Ausführung weiterer Kontrollbefehle unterbunden.
RTC4→ RTC5	• Erhöhter Wertebereich • Befehlsrückkehr erst nach Abarbeitung der Mikrovektorisierung (siehe Hinweis oben) • Die Bildfeldkoordinaten für die X- und Y-Achse werden als 20-Bit-Werte angegeben. Im RTC4-Kompatibilitätsmodus werden die wie bei der RTC4 als 16-Bit-Werte angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–32.768 … 32.767]).
Versionsinfo	Letzte Änderungen: • Version DLL 520, OUT 519: Verhalten bei Koordinatentransformationen mit `at_once` = 2 (siehe Seite 694). • Version DLL 527, OUT 529: Befehlsrückkehr erst nach Ende der Bewegung.
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **goto_xy**. Die Mikrovektorisierung 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`, `Y`, `Z` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–524.288 … 524.287]. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf die Randbereiche geclippt.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **goto_xy**-Befehl (siehe Hinweise dort). • Die x- und y-Koordinatenwerte können mit einer Auflösung von 20 Bit, der z-Koordinatenwert mit einer Auflösung von 16 Bit angegeben werden. Letzterer wird von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bzgl. des angegebenen Wertes für die Sprunggeschwindigkeit gleich behandelt werden. • Der Parameter `DirectMove3D` des Befehls **set_delay_mode** bestimmt die Art der Z-Achsen-Bewegung (linear oder schrittweise korrigiert). • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Kapitel 7.3.5 "Ausgabewerte an das Scan-System", Seite 141).
RTC4→ RTC5	• Erhöhter Wertebereich. • Befehlsrückkehr erst nach Ende der Bewegung (siehe Hinweis bei **goto_xy**). • RTC4-Kompatibilitätsmodus: siehe **goto_xy**, der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
Versionsinfo	Letzte Änderungen: • Version DLL 520, OUT 519: Verhalten bei Koordinatentransformationen mit `at_once` = 2 (siehe Seite 694). • Version DLL 527, OUT 529: Befehlsrückkehr erst nach Ende der Bewegung.
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`, `YHome` Absolute Koordinaten der Home-Position in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Bereich: [–524.288…524.287]. Zu große Werte werden geclippt.
Hinweise	• Der Befehl **home_position** definiert die Koordinaten für einen Home-Jump, der z. B. beim Beenden einer Liste ausgeführt wird. Entsprechend wird dann beim Listen-Start wieder ein Home-Return zur zuletzt gültigen Position ausgeführt. Home-Jump und Home-Return werden mit Sprunggeschwindigkeit ausgeführt. • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Kapitel 7.3.5 "Ausgabewerte an das Scan-System", Seite 141). • Der Befehl **home_position** ist für Lasersysteme vorgesehen, die kein schnelles Schalten des Lasers erlauben. Nach Aufruf des Befehls bewegt sich der Laserfokus immer dann zur angegebenen Home-Position, wenn gerade keine Liste ausgeführt wird bzw. 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-Status gesetzt. • An die Home-Position sollte eine *Strahlenfalle (beam dump)* angebracht werden. • Der Home-Jump-Modus wird mit `home_position(0, 0)` deaktiviert, auch wenn er mit **home_position_xyz** aktiviert wurde.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Die Bildfeldkoordinaten für die X- und Y-Achse werden als 20-Bit-Werte angegeben. Im RTC4-Kompatibilitätsmodus werden die wie bei der RTC4 als 16-Bit-Werte angegebenen Koordinatenwerte der Home-Position von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–32.768 … 32.767]).
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 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **home_position**. Die Mikrovektorisierung 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`, `YHome`, `ZHome`: Absolute Koordinaten der Home-Position in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • für `XHome` und `YHome`: [–524.288 … 524.287]. • für `ZHome`: [–32.768 … 32.767]. Überlaufende Werte werden auf die Randbereiche geclippt.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **home_position**-Befehl (siehe Hinweise dort). • Mit `home_position_xyz( 0, 0, 0)` oder `home_position( 0, 0 )` wird der Home-Jump-Modus deaktiviert. • Die x- und y-Koordinatenwerte können mit einer Auflösung von 20 Bit, der z-Koordinatenwert mit einer Auflösung von 16 Bit angegeben werden. Letzterer wird von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bzgl. des angegebenen Wertes für die Sprunggeschwindigkeit gleich behandelt werden. • Der Parameter `DirectMove3D` des Befehls **set_delay_mode** bestimmt die Art der Z-Achsen-Bewegung (linear oder schrittweise korrigiert). • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Kapitel 7.3.5 "Ausgabewerte an das Scan-System", Seite 141).
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **home_position**, der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
Verweise	**home_position**

Unverzögerter kurzer Listenbefehl	**if_cond**
Funktion	*Bedingte Befehlsausführung*: Dieser Befehl 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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird der nachfolgende Listenbefehl übersprungen.
Aufruf	`if_cond( Mask1, Mask0 )`
Parameter	`Mask1`, `Mask0` 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
Hinweise	• Siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Neuer Befehl.
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*: Dieser Befehl führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die X-Achse erfüllt ist. Andernfalls wird der nachfolgende 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 (Fehlerbit#4 = 1 oder Fehlerbit#5 = 1). > 0: Es hat ein Overflow stattgefunden (Fehlerbit#5 = 1). < 0: Es hat ein Underflow stattgefunden (Fehlerbit#4 = 1).
Hinweise	• Zur Verwendung des Befehls siehe Abschnitt "Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)", Seite 214. • Die gemäß `Mode` abgefragten Fehlerbits (Fehlerbit#4 und/oder Fehlerbit#5) werden zurückgesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525, OUT 527.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **if_not_fly_x_overflow**

Unverzögerter kurzer Listenbefehl	**if_fly_y_overflow**
Funktion	*Bedingte Befehlsausführung für* "Processing on the fly"*-Anwendungen*: Dieser Befehl führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die Y-Achse erfüllt ist. Andernfalls wird der nachfolgende 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 (Fehlerbit#6 = 1 oder Fehlerbit#7 = 1). > 0: Es hat ein Overflow stattgefunden (Fehlerbit#7 = 1). < 0: Es hat ein Underflow stattgefunden (Fehlerbit#6 = 1).
Hinweise	• Zur Verwendung des Befehls siehe Abschnitt "Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)", Seite 214. • Die gemäß `Mode` abgefragten Fehlerbits (Fehlerbit#6 und/oder Fehlerbit#7) werden zurückgesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525, OUT 527.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **if_not_fly_y_overflow**

Unverzögerter kurzer Listenbefehl	**if_fly_z_overflow**
Funktion	*Bedingte Befehlsausführung für* "Processing on the fly"*-Anwendungen*: Dieser Befehl führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die Z-Achse erfüllt ist. Andernfalls wird der nachfolgende 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 (Fehlerbit#24 = 1 oder Fehlerbit#25 = 1). > 0: Es hat ein Overflow stattgefunden (Fehlerbit#25 = 1). < 0: Es hat ein Underflow stattgefunden (Fehlerbit#24 = 1).
Hinweise	• Zur Verwendung des Befehls siehe auch Kapitel 8.7.9 "Überwachung der "Processing on the fly"-Korrektur", Seite 213. • Die gemäß `Mode` abgefragten Fehlerbits (Fehlerbit#24 und/oder Fehlerbit#25) werden zurückgesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	**get_marking_info**, **set_fly_limits_z**, **clear_fly_overflow**, **if_not_fly_z_overflow**

Unverzögerter kurzer Listenbefehl	**if_not_activated**
Funktion	Bedingte Befehlsausführung aufgrund eines Fehlerbits aus **activate_fly_xy** oder **activate_fly_2d**: Wenn das Fehlerbit gesetzt ist, wird der unmittelbar nächste Listenbefehl ausgeführt, ansonsten übersprungen.
Aufruf	`if_not_activated()`
Hinweise	• Siehe auch Hinweise zu **activate_fly_2d** und **activate_fly_xy**. • 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 Fehlerbit aus **activate_fly_xy** oder **activate_fly_2d**. Wenn es zur späteren Abfrage mittels **get_marking_info** (Bit #9 = 1) noch benötigt wird, kann in der Fehlerbehandlungssequenz **activate_fly_2d** oder **activate_fly_xy** erneut aufgerufen werden, um das Fehlerbit wieder zu setzen (vorausgesetzt, die fehlerhafte Situation besteht noch). • Eine erfolgreiche Aktivierung mit **activate_fly_2d** oder **activate_fly_xy** löscht ein evtl. bereits zuvor gesetztes Fehlerbit *nicht*. Es bleibt für **get_marking_info** stehen, bis **get_marking_info** aufgerufen wurde.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**activate_fly_2d**, **activate_fly_xy**, **get_marking_info**

Unverzögerter kurzer Listenbefehl	**if_not_cond**
Funktion	*Bedingte Befehlsausführung*: Dieser Befehl 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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` ***nicht*** gleich `1` ***oder*** die in `Mask0` gesetzten Bits von `IOvalue` ***nicht*** gleich `0` sind). Andernfalls wird der nachfolgende Listenbefehl übersprungen.
Aufruf	`if_not_cond( Mask1, Mask0 )`
Parameter	`Mask1`, `Mask0` 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
Hinweise	• Siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Neuer Befehl.
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*: Dieser Befehl führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die X-Achse **nicht** erfüllt ist. Andernfalls wird der nachfolgende 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 (Fehlerbit#4 = 1 oder Fehlerbit#5 = 1). > 0: Es hat ein Overflow stattgefunden (Fehlerbit#5 = 1). < 0: Es hat ein Underflow stattgefunden (Fehlerbit#4 = 1).
Hinweise	• Zur Verwendung des Befehls siehe Abschnitt "Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)", Seite 214. • Die gemäß `Mode` abgefragten Fehlerbits (Fehlerbit#4 und/oder Fehlerbit#5) werden zurückgesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525, OUT 527.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **if_fly_x_overflow**

Unverzögerter kurzer Listenbefehl	**if_not_fly_y_overflow**
Funktion	*Bedingte Befehlsausführung für* "Processing on the fly"*-Anwendungen*: Dieser Befehl führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die Y-Achse **nicht** erfüllt ist. Andernfalls wird der nachfolgende 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 (Fehlerbit#6 = 1 oder Fehlerbit#7 = 1). > 0: Es hat ein Overflow stattgefunden (Fehlerbit#7 = 1). < 0: Es hat ein Underflow stattgefunden (Fehlerbit#6 = 1).
Hinweise	• Zur Verwendung des Befehls siehe Abschnitt "Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)", Seite 214. • Die gemäß `Mode` abgefragten Fehlerbits (Fehlerbit#6 und/oder Fehlerbit#7) werden zurückgesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525, OUT 527.
Verweise	**get_marking_info**, **set_fly_limits**, **clear_fly_overflow**, **if_fly_y_overflow**

Unverzögerter kurzer Listenbefehl	**if_not_fly_z_overflow**
Funktion	*Bedingte Befehlsausführung für* "Processing on the fly"*-Anwendungen:* Dieser Befehl führt sofort den unmittelbar folgenden Listenbefehl aus, wenn die Bedingung gemäß `Mode` für die Z-Achse ***nicht*** erfüllt ist. Andernfalls wird der nachfolgende 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 (Fehlerbit#24 = 1 oder Fehlerbit#25 = 1). > 0: Es hat ein Overflow stattgefunden (Fehlerbit#25 = 1). < 0: Es hat ein Underflow stattgefunden (Fehlerbit#24 = 1).
Hinweise	• Zur Verwendung des Befehls siehe auch Kapitel 8.7.9 "Überwachung der "Processing on the fly"-Korrektur", Seite 213. • Die gemäß `Mode` abgefragten Fehlerbits (Fehlerbit#24 und/oder Fehlerbit#25) werden zurückgesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	**get_marking_info**, **set_fly_limits_z**, **clear_fly_overflow**, **if_fly_z_overflow**

Unverzögerter kurzer Listenbefehl	**if_not_pin_cond**
Funktion	*Bedingte Befehlsausführung*: Dieser Befehl 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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` ***nicht*** gleich `1` ***oder*** die in `Mask0` gesetzten Bits von `IOvalue` ***nicht*** gleich `0` sind). Andernfalls wird der nachfolgende Listenbefehl übersprungen.
Aufruf	`if_not_pin_cond( Mask1, Mask0 )`
Parameter	`Mask1`, `Mask0` 2-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 2 Bits ausgewertet.
Hinweise	• Siehe auch "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Neuer Befehl.
Verweise	**if_pin_cond**, **if_cond**, **if_not_cond**

Unverzögerter kurzer Listenbefehl	**if_pin_cond**
Funktion	*Bedingte Befehlsausführung*: Dieser Befehl 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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird der nachfolgende Listenbefehl übersprungen.
Aufruf	`if_pin_cond( Mask1, Mask0 )`
Parameter	`Mask1`, `Mask0` — 2-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 2 Bits ausgewertet.
Hinweise	• Siehe auch "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Neuer Befehl.
Verweise	**if_not_pin_cond**, **if_cond**, **if_not_cond**

Kontrollbefehl	**init_fly_2d**
Funktion	Initialisiert die Encoder-Referenzwerte für die 2D-Encoder-Kompensation einer nachfolgend mit **set_fly_2d** eingestellten "Processing on the fly"-Anwendung und setzt beide Encoderwerte zurück.
Aufruf	`init_fly_2d( OffsetX, OffsetY )`
Parameter	`OffsetX`, `OffsetY` — Referenzwert des X- bzw. Y-Achsen-Encoders (32 Bit-Werte mit Vorzeichen). Zulässiger Wertebereich: abhängig von der mit **load_fly_2d_table** geladenen Kompensationstabelle. Defaultwerte (nach **load_program_file**): = 0.
Hinweise	• Der endgültig für die 2D-Korrektur verwendete Encoderwert (also aktueller Encoderwert + Referenzwert) darf den zulässigen Wertebereich nicht überschreiten. Andernfalls erfolgt ein Clipping (siehe auch **load_fly_2d_table**).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**set_fly_2d**, **load_fly_2d_table**, **get_fly_2d_offset**

Kontrollbefehl	**init_rtc5_dll**
Funktion	Initialisiert die Ansteuerung der installierten RTC5-Karten für ein Anwenderprogramm.
Aufruf	`InitErrorNo = init_rtc5_dll()`
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 wie beim Befehl **get_error**.
Hinweise	• *Der Befehl* **init_rtc5_dll** *muss von jedem Anwenderprogramm zu Beginn aufgerufen werden*, damit von dem Anwenderprogramm überhaupt eine RTC5-Karte ansprechbar ist. **init_rtc5_dll** sucht nach sämtlichen vorhandenen RTC5-Karten und baut die entsprechende Verwaltung auf. Wird keine RTC5-Karte gefunden, wird der Fehlercode `RTC5_NO_CARD` zurückgegeben. • Wenn beim Lesen des PCI-Konfigurationsregisters ein Fehler aufgetreten ist (**get_last_error**-Returncode `RTC5_CONFIG_ERROR`), wird auch der Fehler `RTC5_ACCESS_DENIED` erzeugt. Auf die Karte kann dann so lange nicht mehr zugegriffen werden, bis nach einem PC-Neustart der Fehler beseitigt ist. • Der Befehl **init_rtc5_dll** vergibt dem Anwenderprogramm automatisch wie mit **acquire_rtc** die Zugriffsberechtigung für die gefundenen Karten, sofern die Zugriffsberechtigung nicht bereits an ein anderes Anwenderprogramm vergeben wurde (es können zwar beliebig viele RTC5-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 der Befehl **init_rtc5_dll** dann den Fehlercode `RTC5_ACCESS_DENIED` zurück. Die Karten sind dann von diesen Anwenderprogrammen nur mit ausschließlich 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_rtc5_dll** oder **acquire_rtc** – von einem anderen Anwenderprogramm beansprucht werden, nachdem die Karte mit **release_rtc** oder **free_rtc5_dll** explizit vom zugriffsberechtigten Anwenderprogramm freigegeben wurde. Der Fehlercode `RTC5_ACCESS_DENIED` wird zurückgegeben, wenn mindestens eine der gefundenen Karten den Zugriff verweigert hat. Welche das ist oder sind, kann mit `get_error(CardNo)` (`CardNo` von 1 bis zur Zahl der gefundenen Karten) oder unmittelbar nach **init_rtc5_dll** (noch vor dem ersten anderen Befehl) mit `get_last_error(CardNo)` abgefragt werden. • Beim Akquirieren einer Karte durch **init_rtc5_dll** wird (wie mit **acquire_rtc**) eine Versionsprüfung durchgeführt. Liegt ein Versionsfehler vor, wird der Zugriff auf die Karte verweigert (Rückgabewert `RTC5_ACCESS_DENIED`|`RTC5_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. • Wird der Befehl **init_rtc5_dll** von demselben Anwenderprogramm mehrmalig aufgerufen, so wird die beim vormaligen Aufruf des Befehls aufgebaute Kartenverwaltung für dieses Anwenderprogramm gelöscht und die dabei an das Anwenderprogramm vergebenen Zugriffsberechtigungen aufgehoben, ehe die neue Kartenverwaltung aufgebaut wird und wieder neue Zugriffsberechtigungen erteilt werden.

Kontrollbefehl	init_rtc5_dll
Hinweise (Forts.)	• Bei jeder Initialisierung eines Anwenderprogramms mit **init_rtc5_dll** werden DLL-interne Nummern für alle gefundenen RTC5-Karten neu vergeben. Die Zuordnung zwischen den DLL-internen Nummern und den installierten Karten kann über **get_serial_number** ermittelt werden. Die Karte mit der kleinsten 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 DLL-internen Nummer (siehe **rtc5_count_cards**) zur aktiven Karte. Im letzteren Fall können dann aber ausschließlich rein 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 96. • Der Befehl **init_rtc5_dll** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **init_rtc5_dll** ist nicht als Multi-Board-Befehl verfügbar. • Nach der Initialisierung der DLL mit **init_rtc5_dll** kann zusätzlich zwischen zwei DLL-Betriebsmodi ausgewählt werden (siehe **set_rtc4_mode** und **set_rtc5_mode**). Standardmäßig ist der RTC5-Modus voreingestellt. • Der Befehl **init_rtc5_dll** löst keinen Reset der RTC5-Karte aus. Ein Karten-Reset wird nur durch den Befehl **load_program_file** initiiert.
RTC4→ RTC5	Neuer Befehl.

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 zweidimensionalen Bildfeld.
Aufruf	`jump_abs( X, Y )`
Parameter	`X, Y` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
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 pro ms* ausgeführt. • Die "Laser active"-Lasersteuersignale sind während des Sprungs ausgeschaltet (werden gegebenenfalls vor dem Sprung ausgeschaltet). • Nach einem Jump-Befehl wird ein (variables) Jump-Delay eingefügt. Ausnahme: Das einem Sprungvektor der Länge 0 folgende (variable) Jump-Delay wird nicht ausgeführt. Der Befehl selber benötigt dafür aber dennoch einen 10 µs-Takt zur Ausführung. • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141). Ab Version DLL 520, OUT 519 werden zuvor angesammelte Koordinatentransformationen gegebenenfalls erst durch **jump_abs** wirksam (siehe Seite 185, `at_once` = 2).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	Letzte Änderung Version DLL 520, OUT 519: Verhalten bei Koordinatentransformationen mit `at_once` = 2 (siehe Seite 694).
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **jump_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **jump_abs**-Befehl (siehe Hinweise dort). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bzgl. des angegebenen Wertes für die Sprunggeschwindigkeit gleich behandelt werden. • Der Parameter `DirectMove3D` des Befehls **set_delay_mode** bestimmt die Art der Z-Achsen-Bewegung (linear oder schrittweise korrigiert). • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Kapitel 7.3.5 "Ausgabewerte an das Scan-System", Seite 141).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
Verweise	**jump_abs**, **jump_rel_3d**, **timed_jump_abs_3d**

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 zweidimensionalen Bildfeld.
Aufruf	`jump_rel( dX, dY )`
Parameter	`dX, dY` *Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **jump_abs** (siehe Hinweise dort).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	Letzte Änderung Version DLL 520, OUT 519: Verhalten bei Koordinatentransformationen mit `at_once` = 2 (siehe Seite 694).
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **jump_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **jump_abs_3d** (siehe Hinweise dort).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
Verweise	**jump_abs_3d**, **jump_abs**, **jump_rel**, **timed_jump_rel_3d**

Variabler Listenbefehl	**laser_on_list**
Funktion	Schaltet die "Laser active"-Lasersteuersignale 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 Bereich: $0 \leq$ `Period` $\leq (2^{31}-1)$.
Hinweise	• Ab Version OUT 531 wird Bit #31 von `Period` ignoriert (siehe auch Hinweise und Versionsinfo zu **laser_on_pulses_list**). • Die Lasersteuersignale für den "Laser active"-Betrieb müssen erst mit **set_laser_mode** ausgewählt, deren Parameter mit weiteren Befehlen spezifiziert und mit **set_laser_control** oder **enable_laser** freigegeben werden, ehe sie mit **laser_on_list** eingeschaltet werden können (siehe Kapitel 7.4 "Lasersteuerung", Seite 144). • 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. • Der Befehl **laser_on_list** ist nützlich für das Markieren separater Punkte (siehe Kapitel 7.1.3 "Markieren von Punkten", Seite 108). • Der Wobbelmode (siehe Kapitel 8.4 "Wobbelmode", Seite 189) bleibt erhalten, wird aber nicht berücksichtigt. • Für `Period` = 0 gehen die Lasersteuersignale nicht an, der Befehl ist dann ein kurzer Listenbefehl. • Der Befehl **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→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich).
Versionsinfo	Letzte Änderung (Version DLL 530, OUT 531, RBF 522): Veränderte Funktionalität für $2^{31} \leq$ `Period` $\leq (2^{32}-1)$ (siehe Hinweise zu **laser_on_pulses_list**).
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. Der Befehl wirkt für `Pulses` > 65535 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 Bereich: 0 ≤ `Period` ≤ (2^{32}–1).
	`Pulses` Anzahl externer Signalpulse als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: 0 ≤ `Pulses` ≤ 65535 oder größer (siehe Hinweis unten).
Hinweise	• Der Befehl ist nützlich für das Markieren separater Punkte im Lasermode 6 (siehe Kapitel 7.4.7 "Lasermode 6", Seite 155), ist aber auch in allen anderen Laser-Versionen wirksam. • Die externen Signalpulse müssen als TTL-Pulse am Digital-Eingang DIGITAL IN1 der LASER-Buchse (siehe Abschnitt "Digitaler Eingang und Ausgang", Seite 48) 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 bleibt der Befehl ohne Wirkung. Der Befehl 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 (d.h. (`Period` – 2^{31}) × 10 µs). Hier wird der Befehl jedoch vorzeitig abgebrochen, sobald die angegebene Anzahl von externen Signalpulsen detektiert wurde. • Für `Pulses` > 65535 wirkt **laser_on_pulses_list** identisch zu **laser_on_list** (hier werden keine externen Signalpulse berücksichtigt; siehe auch Hinweise dort). Ansonsten (0 ≤ `Pulses` ≤ 65535) 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 (also nicht die LASER1- und LASER2-Signale) und berücksichtigt keine Laser-Delays. Auch in vorherigen Befehlen eventuell aktivierte, aber noch nicht abgelaufene Laser-Delays sind während dieses Befehls nicht wirksam (jedoch gegebenenfalls sofort dann wieder, wenn das LASERON-Signal nach dem letzten Puls oder nach `Period` wieder ausgeschaltet wird). • Für 1 ≤ `Pulses` ≤ 65535 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.

Variabler Listenbefehl	laser_on_pulses_list
Hinweise (Forts.)	• 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** spezifiziert und freigegeben werden, ehe es mit **laser_on_pulses_list** eingeschaltet werden kann (siehe Kapitel 7.4 "Lasersteuerung", Seite 144). • Während der Befehlsausführung bleibt die Position des Scanners unverändert. Der nachfolgende Listenbefehl wird erst ausgeführt, wenn das angegebene Zeitintervall `Period` verstrichen ist. • Der Wobbelmode (siehe Seite 189) bleibt erhalten, wird aber nicht berücksichtigt. • Ein eventuell eingestellter Softstart wird nicht ausgeführt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 515, OUT 514, RBF 512. • Letzte Änderung (Version DLL 530, OUT 531, RBF 522): Veränderte Funktionalität für $2^{31} \leq$ `Period` $\leq (2^{32}-1)$.
Verweise	**laser_on_list**, **para_laser_on_pulses_list**

Kontrollbefehl	laser_signal_off
Funktion	Schaltet die "Laser active"-Lasersteuersignale ohne Verzug aus.
Aufruf	`laser_signal_off()`
Hinweise	• Dieser Befehl ist für eine direkte Lasersteuerung in Kombination mit dem Befehl **laser_signal_on** vorgesehen. • Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist, d.h. wenn gerade eine Liste ausgegeben wird oder wenn eine Liste mit **pause_list** angehalten wurde. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist.
RTC4→ RTC5	Unverändert.
Verweise	**laser_signal_on**

Normaler Listenbefehl	laser_signal_off_list
Funktion	Identisch mit **laser_signal_off**, aber ein Listenbefehl.
Aufruf	`laser_signal_off_list()`
RTC4→ RTC5	Unverändert.

Kontrollbefehl	laser_signal_on
Funktion	Schaltet die “Laser active”-Lasersteuersignale ohne Verzug ein.
Aufruf	`laser_signal_on()`
Hinweise	• Die Lasersteuersignale für den “Laser active”-Betrieb müssen erst mit **set_laser_mode** ausgewählt, deren Parameter mit weiteren Befehlen spezifiziert und mit **set_laser_control** oder **enable_laser** freigegeben werden, ehe sie mit **laser_signal_on** eingeschaltet werden können (siehe Kapitel 7.4 “Lasersteuerung”, Seite 144). • Der Befehl ist dafür vorgesehen, den Laser etwa zur Justage direkt einzuschalten. • Die “Laser active”-Lasersteuersignale müssen mit dem Befehl **laser_signal_off** ausgeschaltet werden. • Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • *Vorsicht! Überprüfen Sie den Strahlengang, ehe Sie den Laser einschalten!*
RTC4→ RTC5	Unverändert.
Verweise	**laser_signal_off**

Normaler Listenbefehl	laser_signal_on_list
Funktion	Identisch mit **laser_signal_on**, aber ein Listenbefehl und nie ignoriert.
Aufruf	`laser_signal_on_list()`
RTC4→ RTC5	Unverändert.

Unverzögerter kurzer Listenbefehl	**list_call**
Funktion	Bewirkt einen unbedingten Sprung zu einem Unterprogramm, das an der angegebenen absoluten Adresse (in einem beliebigen Speicherbereich des Listenspeichers) beginnt.
Aufruf	`list_call( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Der erste Befehl des mit **list_call** aufgerufenen Unterprogramms wird (evtl. nach einem **list_nop**-Befehl bzw. ab Version DLL 525 nach einem **list_continue**-Befehl) sofort und ohne Verzögerung ausgeführt. Es sind auch geschachtelte bzw. rekursive Unterprogrammaufrufe zulässig mit einer Verschachtelungstiefe von bis zu 63 Aufrufen (siehe auch Kapitel 6.5.1 "Unterprogramme", Seite 81). • Jedes Unterprogramm muss mit dem Befehl **list_return** abgeschlossen sein, damit nach Abarbeiten eines Unterprogramms einschließlich des abschließenden **list_return**-Befehls mit dem der aufrufenden Stelle nachfolgenden Befehl fortgefahren wird. Dieser wird dann auch (evtl. nach einem **list_nop**- bzw. **list_continue**-Befehl) sofort und ohne Verzögerung ausgeführt. Wird statt eines erwarteten **list_return**-Befehls ein **set_end_of_list**-Befehl angetroffen, wird die Listenausführung beendet oder – sofern 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 Unterprogrammaufrufen. Ein evtl. noch unfertiger **mark_text**-Befehl wird nicht mehr zu Ende ausgeführt. Wird das Ende eines Listenbereichs ("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 Speicherbereich wird in einem solchen Fall zwangsweise ein **list_return**-Befehl eingefügt und ausgeführt. • Der Befehl **list_call** wird durch **list_nop** ersetzt, wenn `Pos` >(2^{20}–1) oder wenn `Pos` gleich der aktuellen Adresse ist (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`). • Erfolgt in einem aufgerufenen Unterprogramm ein weiterer **list_call**-Befehl auf die Adresse des aufrufenden **list_call**-Befehls (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). Ab Version OUT 540 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. • Der mit **read_status** auslesbare BUSY-Listenstatus wird durch **list_call** gegebenenfalls geändert, wenn sich die aufgerufene Adresse im Listen-Bereich ("Liste 1" oder "Liste 2") befindet. Liegt diese dagegen im geschützten Speicherbereich ("Liste 3"), wird der Listenstatus der aufrufenden Stelle beibehalten, da der geschützte Bereich keinen eigenen Listenstatus besitzt. Beim Ausführen eines Unterprogramms im geschützten Speicherbereich gibt auch **get_status** den Listenausführungsstatus der aufrufenden Stelle und **get_status** und **get_out_pointer** als Position des Output-Pointers die Position des aufrufenden **list_call**-Befehls zurück.

Unverzögerter kurzer Listenbefehl	list_call
Hinweise (Forts.)	• Absolute Vektor- und Bogenbefehle 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-, Bogen- oder Jump-Befehle enthalten oder es ist der Befehl **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 aufgerufenden Unterprogramms am Anfang derselben Liste mit der Ausführung fortgefahren. • `list_call( Pos )` ist synonym zu `list_call_repeat( Pos, 1 )`.
RTC4→ RTC5	Weitgehend unveränderte Funktionalität.
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 (in einem beliebigen Speicherbereich des Listenspeichers) beginnt. Im aufgerufenen Unterprogramm werden absolute Vektor- und Bogenbefehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt.
Aufruf	`list_call_abs( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Der Befehl **list_call_abs** ist weitgehend identisch mit dem **list_call**-Befehl (siehe Hinweise dort). Der Befehl **list_call_abs** bewirkt allerdings eine andere Ausführung von absoluten Vektor- und Bogenbefehlen 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- und Bogenbefehlen 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- und Bogenbefehle 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 synonym zu `list_call_abs_repeat( Pos, 1 )`.
RTC4→ RTC5	Neuer Befehl.
Verweise	**list_call**, **list_call_abs_cond**, **list_call_abs_repeat**

Unverzögerter kurzer Listenbefehl	**list_call_abs_cond**
Funktion	*Bedingter Aufruf (AbsCall) eines Unterprogramms*: Dieser Befehl 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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_call_abs_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1`, `Mask0` — 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Pos` — Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Siehe **list_call_abs**. • Siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Neuer Befehl.
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 (in einem beliebigen Speicherbereich des Listenspeichers) beginnt und führt dessen Rumpf mehrmals aus.
Aufruf	`list_call_abs_repeat( Pos, Number )`
Parameter	`Pos` — Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen (wie bei **list_call_abs**). `Number` — Anzahl der Wiederholungen als 32-Bit-Wert ohne Vorzeichen. `Number` = 0 wird wie `Number` = 1 behandelt.
Hinweise	• `list_call_abs( Pos )` ist synonym zu `list_call_abs_repeat( Pos, 1 )`. • Der Befehl **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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 540, OUT 540.
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*: Dieser Befehl führt den Befehl **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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_call_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1`, `Mask0` — 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Pos` — Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Siehe **list_call**. • Siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Weitgehend unveränderte Funktionalität (siehe **list_call**).
Verweise	**list_call**

Unverzögerter kurzer Listenbefehl	**list_call_repeat**
Funktion	Bewirkt einen unbedingten Sprung zu einem Unterprogramm, das an der angegebenen absoluten Adresse (in einem beliebigen Speicherbereich des Listenspeichers) beginnt und führt dessen Rumpf mehrmals aus.
Aufruf	`list_call_repeat( Pos, Number )`
Parameter	`Pos` Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen (wie bei **list_call**). `Number` Anzahl der Wiederholungen als 32-Bit-Wert ohne Vorzeichen. `Number` = 0 wird wie `Number` = 1 behandelt.
Hinweise	• `list_call_abs( Pos )` ist synonym zu `list_call_abs_repeat( Pos, 1 )`. • Der Befehl **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 dem Befehl **list_call_repeat** können z. B. Trajektorien für Anlauf- und Auslaufkurven aus Mikrovektor-Befehlen mit Figuren aus Unterprogrammen nahtlos aneinander gefügt werden. • Wenn bei **list_call_repeat** unmittelbar ein weiterer Unterprogrammaufruf folgt (verschachtelte Aufrufe), muss allerdings ein Leertakt eingefügt werden. Das kann vermieden werden, wenn zwischen zwei Unterprogrammaufrufen z. B. mindestens ein Mikrovektor-Befehl steht. • Der Beginn des Unterprogrammes `Pos` kann an einer beliebigen Stelle im Listenspeicherbereich liegen. Das gilt insbesondere auch unmittelbar nach dem Befehl **list_call_repeat**. Dadurch kann der gesamte 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 Unterprogrammes unmittelbar auf **list_call_repeat** folgt, wird mit dem Befehl unmittelbar nach **list_return** fortgefahren.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 540, OUT 540.
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 Listenspeicher ein.
Aufruf	`list_continue()`
Hinweise	• Der Befehl **list_continue** benötigt wie **list_nop** 10 µs zur Ausführung. • Folgt der Befehl 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 Lasersignale 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. • Seit Version OUT 519 wird bei einer Überschreitung der maximal zulässigen Anzahl von unmittelbar aufeinander folgenden kurzen Listenbefehlen automatisch ein Leertakt eingefügt, der genau dem Befehl **list_continue** entspricht (bei älteren Versionen wird hier stattdessen zur Laufzeit automatisch ein **list_nop** eingefügt, was einen Polygonzug unterbricht). • Der Befehl **list_continue** kann auch dazu verwendet werden, mehrfache Ausgaben auf ein und denselben Ausgabeport voneinander zu trennen. Ohne diesen Befehl würden beispielsweise mehrere unmittelbar aufeinanderfolgende **write_da_x_list**-Befehle (kurze Listenbefehle) den analogen Ausgabeport in einem 10 µs-Takt mehrmals beschreiben, wodurch einige Werte nie zur Ausgabe gelangen könnten. • Die RTC5 setzt **list_continue** nie als Platzhalter für andere (zurückgewiesene) Listenbefehle ein, sondern immer nur **list_nop**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525, OUT 527.
Verweise	**list_nop**

Unverzögerter kurzer Listenbefehl	**list_jump_cond**
Funktion	*Bedingter (absoluter) Listensprung*: Dieser Befehl führt den Befehl **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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_jump_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1`, `Mask0` 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Pos` Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Der Befehl ist synonym zum Befehl **list_jump_pos_cond** (siehe Hinweise dort).
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: • Erhöhter Wertebereich für `Pos`. • Sprünge in den geschützten Speicherbereich “Liste 3” und Sprünge aus dem geschützten Speicherbereich heraus sind verboten (fehlerhafte Sprungbefehle werden beim Abarbeiten ignoriert, siehe auch **list_jump_pos**).
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 Listenspeichers. Der dortige nächste Befehl wird sofort und ohne Verzögerung ausgeführt.
Aufruf	`list_jump_pos( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Für `Pos` kann eine absolute Speicheradresse innerhalb des konfigurierten Listenbereichs (“Liste 1” und “Liste 2”) angegeben werden, nicht jedoch innerhalb des geschützten Bereichs “Liste 3” oder ganz außerhalb des Listenspeicherbereichs. • Verboten sind neben Sprüngen in den geschützten Speicherbereich “Liste 3” auch solche aus dem geschützten Speicherbereich heraus. • Fehlerhafte Sprungbefehle werden zwar unverändert zur RTC5 übertragen, beim Abarbeiten jedoch ignoriert. Stattdessen wird der unmittelbar nachfolgende Befehl ausgeführt. Das Programm funktioniert vermutlich nicht mehr wie gewünscht. Bei der Programmierung mit Sprungbefehlen muss daher mit einer entsprechenden Sorgfalt vorgegangen werden (siehe auch Kapitel 6.5.3 “Sprünge”, Seite 89). • Sprungbefehle, die einen Sprung auf sich selbst veranlassen (`Pos` = Listenposition des Sprungbefehls) werden zur Laufzeit ebenso ignoriert, sonst würden sie bei der Ausführung eine Endlosschleife ohne weitere Aktivitäten ergeben, die sich nur mit **stop_execution** oder per externem Listen-Stopp beenden ließe. • Entscheidend sind die Bedingungen zur Laufzeit. Beim Umkonfigurieren des Listenspeichers oder bei der Konvertierung eines Unterprogramms kann es passieren, dass eine ursprünglich legale Sprungadresse aufgrund neuer Listengrenzen oder einer veränderten Speicherposition des Unterprogramms zu einer illegalen Sprungadresse wird. • Der BUSY-Listenstatus der Listen wird durch **list_jump_pos** wechselnd gesetzt, der USED-Listenstatus der Listen bleibt unverändert (siehe **read_status**). • Der Befehl **list_jump_pos** ist synonym zum Befehl **set_list_jump**.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_list_jump**, **list_jump_rel**, **list_jump_pos_cond**

Unverzögerter kurzer Listenbefehl	**list_jump_pos_cond**
Funktion	*Bedingter (absoluter) Listensprung*: Dieser Befehl führt den Befehl **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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_jump_pos_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1`, `Mask0` — 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Pos` — Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
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 Sprungbefehls) erlaubt, z.B. zum Warten auf die Bestätigung eines Signals. • Der Befehl ist synonym zum Befehl **list_jump_cond**. • Siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.

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 "Programmierbeispiele", Seite 245.

RTC4→ RTC5	Neuer Befehl.
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 [$(-2^{20}+1)$ … $(2^{20}-1)$] als 32-Bit-Wert mit Vorzeichen.
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 Speicherbereichen, also auch im geschützten Speicherbereich ("Liste 3"), verwendet werden. • Im Bereich der "Liste 1" oder "Liste 2" bzw. 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 Speicherbereichs 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 Sprungbefehle werden zwar unverändert zur RTC5 übertragen, beim Abarbeiten jedoch ignoriert. Stattdessen wird der unmittelbar nachfolgende Befehl ausgeführt. Das Programm funktioniert vermutlich nicht mehr wie gewünscht. Bei der Programmierung mit Sprungbefehlen muss daher mit einer entsprechenden Sorgfalt vorgegangen werden (siehe auch "Sprünge", Seite 89). • Sprungbefehle, 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 per externem Listen-Stopp beenden ließe. • Entscheidend sind die Bedingungen zur Laufzeit. Beim Umkonfigurieren des Listenspeichers oder bei der Konvertierung eines Unterprogramms kann es passieren, dass eine ursprünglich legale Sprungadresse aufgrund neuer Listengrenzen oder einer veränderten Speicherposition des Unterprogramms zu einer illegalen Sprungadresse wird.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version OUT 517.
Verweise	**list_jump_pos**, **list_jump_rel_cond**

Unverzögerter kurzer Listenbefehl	**list_jump_rel_cond**
Funktion	*Bedingter (relativer) Listensprung*: Dieser Befehl führt den Befehl **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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`list_jump_rel_cond( Mask1, Mask0, Pos )`
Parameter	`Mask1`, `Mask0` — 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Pos` — Sprungweite [$(-2^{20}+1)$ … $(2^{20}-1)$] als 32-Bit-Wert mit Vorzeichen.
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 81.

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 "Programmierbeispiele", Seite 245.

RTC4→ RTC5	Neuer Befehl.
Verweise	**list_jump_rel**

Unverzögerter kurzer Listenbefehl	**list_next**
Funktion	Führt den nächsten Listenbefehl aus.
Aufruf	`list_next()`
Hinweise	• Der Befehl **list_next** liest den nächsten Listenbefehl und führt ihn sofort aus, sofern nicht die maximale Anzahl von zulässigen Kurzbefehlen überschritten ist. Andernfalls wird er erst im nächsten 10 *µ*s-Takt ausgeführt. • Der Befehl eignet sich daher als Platzhalter für einen anderen Befehl in einer Liste, weil er einen zusätzlichen Takt vermeidet, der sonst bei Platzhaltern wie **list_nop** oder **list_continue** unvermeidlich ist.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 541, OUT 541.
Verweise	**list_nop**, **list_continue**

Normaler Listenbefehl	**list_nop**
Funktion	Fügt eine Null-Operation (no operation) in den Listenspeicher ein.
Aufruf	`list_nop()`
Hinweise	• Der Befehl **list_nop** wirkt wie **long_delay**(1), schaltet also die “Laser active”-Lasersignale gegebenenfalls nach einem LaserOff-Delay aus und wartet ein etwaiges Scanner-Delay ab. Auch wenn keine Delays abzuwarten sind, benötigt der Befehl dennoch 10 µs zur Ausführung. • Der Befehl dient als Platzhalter für zurückgewiesene Listenbefehle (**get_last_error**-Returncode: `RTC5_IGNORED`). • Folgt der Befehl 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→ RTC5	Prinzipiell unveränderte Funktionalität, schaltet aber die “Laser active”-Lasersignale aus.
Verweise	**long_delay**, **list_continue**

Unverzögerter kurzer Listenbefehl	**list_repeat**
Funktion	Leitet eine Befehlswiederholung für eine Gruppe von Listenbefehlen ein.
Aufruf	`list_repeat()`
Hinweise	• Siehe Kapitel 6.5.5 "Schleifen", Seite 91. • Ein evtl. noch anliegender verzögerter kurzer Listenbefehl wird noch davor ausgeführt. • Alle Listenbefehle zwischen diesem Befehl und dem nächsten **list_until**-Befehl werden ggf. mehrfach wiederholt. • Es können bis zu 8 verschachtelte **list_repeat**/**list_until**-Befehlsschleifen programmiert werden. Jeder weitere **list_repeat**-Befehl darüber hinaus wird ignoriert, solange kein **list_until**-Befehl 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 geschieht 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 Unterprogrammes 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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
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 nächsten Listenbefehl (evtl. nach einem **list_nop**) sofort und ohne Verzögerung aus.
Aufruf	`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 Input-Pointer explizit auf eine andere Position gesetzt wird, so werden sie nicht gespeichert und sind im Folgenden nicht verfügbar. • Der **list_return**-Befehl löst beim Laden einen Flush der gepufferten Listeneingabe aus (siehe Seite 75). • Folgt **list_return** unmittelbar nach **load_sub**, **load_char** oder **load_text_table**, dann ist der Inputpointer 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 Unterprogrammstruktur zerstört.
RTC4→ RTC5	In der alten Funktionalität (Beenden eines Unterprogramms) unverändert. Neu: Bedeutung beim Laden in den geschützten Speicherbereich.
Verweise	**list_call**, **sub_call**, **load_char**, **load_text_table**, **load_sub**

Unverzögerter kurzer Listenbefehl	**list_until**
Funktion	Beendet ggf. 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.
Hinweise	• Siehe Kapitel 6.5.5 "Schleifen", Seite 91. • Ein evtl. noch anliegender verzögerter kurzer Listenbefehl wird noch davor ausgeführt. • Alle Listenbefehle zwischen diesem Befehl und dem zuletzt vorausgegangenen **list_repeat**-Befehl werden `Number` mal wiederholt. • Es können bis zu 8 verschachtelte **list_repeat**/**list_until**-Befehlsschleifen programmiert werden. Jeder weitere **list_until**-Befehl, für den kein vorausgegangener **list_repeat**-Befehl aufgerufen wurde, wird ignoriert. • Eine leere Schleife, bestehend aus einem **list_repeat**-Befehl, unmittelbar gefolgt von einem **list_until**-Befehl, 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 geschieht 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 Unterprogrammes 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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**list_repeat**

Kontrollbefehl	load_auto_laser_control	
Funktion	Lädt eine Tabelle mit Datenpunkten aus einer ASCII-Textdatei und bestimmt durch lineare Interpolation die Nichtlinearitätskurve für die positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung (siehe Seite 163).	
Aufruf	`NoOfDataPoints = load_auto_laser_control( Name, No )`	
Parameter	`Name`	Name der Textdatei als Pointer auf einen nullterminierten ANSI-String. Die Textdatei kann eine oder mehrere Tabellen enthalten.
	`No`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird spezifiziert, welche Tabelle aus der Textdatei geladen werden soll (der Parameter entspricht der Erweiterung `<No>` der Anweisung `[AutoLaserCtrlTable<No>]` am Anfang der gewünschten Tabelle).
Rückgabe	32-Bit-Wert mit Vorzeichen (im Fehlerfall ein positiver Fehlercode, im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte).	
	Wert	Beschreibung
	–1…– 50	Erfolg. Der Absolutwert des Rückgabewertes ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert (siehe auch Seite 163).
	–1024	Für `Name` = 0 (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 oder INTERNAL-BUSY, kein Download (**get_last_error**-Returncode `RTC5_BUSY`).
	8	Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC5_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC5_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC5_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer konnte nicht in der Datei gefunden werden.
Hinweise	• Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für die Nichtlinearitätskurve in die Text-Datei sind im Abschnitt "Hinweise zum Laden und zur Ermittlung der Nichtlinearitätskurve", Seite 163 beschrieben. Beim Laden der Tabelle bestimmt die RTC5 durch lineare Interpolation geeignete Werte für den ganzen Prozentbereich. • Der Befehl **load_auto_laser_control** überschreibt jede zuvor geladene Nichtlinearitätskurve. • Für `Name` = 0 wird (wie bei Initialisierung mit **load_program_file**) für den vollen Prozentbereich die Funktion *Scale*(*Percent*)`=1.0` geladen (keine Nichtlinearität). • Der Befehl **load_auto_laser_control** wird *nicht* ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Während der Laufzeit des Befehls 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.	

Kontrollbefehl	load_auto_laser_control
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_auto_laser_control**

Kontrollbefehl	load_char
Funktion	Versieht ein durch nachfolgende Listenbefehle definiertes Zeichen mit dem gewünschten Index und lädt es in den geschützten Speicherbereich (“Liste 3”).
Aufruf	`load_char( Char )`
Parameter	`Char` Index des indizierten Zeichens als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich [0 … 1023].
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 der **load_char**-Befehl ignoriert (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Die Adressen im geschützten Speicherbereich, 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**-Befehl abgeschlossen werden. Erst damit werden die Befehle endgültig gespeichert, die Startadresse und evtl. andere Informationen wie die Anzahl der Befehle in die interne Verwaltungstabelle eingetragen und auch ein Flush der gepufferten Listeneingabe ausgelöst (siehe Seite 75). Andernfalls (der Input-Pointer wird ohne vorhergehenden **list_return**-Befehl verändert) ist die Zeichendefinition mit diesem Index nicht verfügbar. • Eine indizierte Zeichendefinition wird nicht gespeichert, falls nicht zuvor der geschützte Speicherbereich (“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 87.
RTC4→ RTC5	Neuer Befehl.
Verweise	**list_return**, **load_sub**, **load_text_table**

Kontrollbefehl	load_correction_file	
Funktion	Überträgt die angegebene Bildfeld-Korrekturdatei in den Speicher der RTC5 (als Tabelle #1, 2,3 oder 4) und ruft automatisch **select_cor_table** mit den zuletzt verwendeten Parameterwerten (bzw. den Default-Parameterwerten) auf.	
Aufruf	`ErrorNo = load_correction_file( Name, No, Dim )`	
Parameter	`Name`	Name der Korrekturdatei als Pointer auf einen nullterminierten ANSI-String.
	`No`	Diese Nummer (32-Bit-Wert ohne Vorzeichen) bestimmt, ob die Datei als Korrekturtabelle #1, 2, 3 oder 4 gespeichert wird. Zulässige Werte: [1…4].
	`Dim`	Dieser Parameter (32-Bit-Wert ohne Vorzeichen) bestimmt, ob die Datei als 2D-Korrekturtabelle oder (sofern möglich) als 3D-Korrekturtabelle gespeichert wird (siehe auch Hinweise). = 2: Sowohl 2D- als auch 3D-Dateien werden als 2D-Tabelle gespeichert (gegebenenfalls Downgrade). = 3: Bei freigeschalteter 3D-Option werden sowohl 2D- als auch 3D-Dateien als 3D-Tabelle gespeichert (gegebenenfalls Upgrade). Bei nicht freigeschalteter 3D-Option wirkt der Befehl wie mit `Dim` = 2. Sonst: Parameter hat keine Auswirkung. 2D-Dateien werden als 2D-Tabelle gespeichert. 3D-Dateien werden bei freigeschalteter 3D-Option als 3D-Tabelle gespeichert, bei nicht freigeschalteter 3D-Option als 2D-Tabelle (Downgrade).
Rückgabe	Fehlercode als 32-Bit-Wert ohne Vorzeichen:	
	Wert	Beschreibung
	0	Erfolg.
	1	Datei Fehler (Datei korrupt oder unvollständig).
	2	Memory Fehler (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 (Treiber-Fehler).
	8	Systemtreiber nicht gefunden (**get_last_error**-Returncode: `RTC5_ACCESS_DENIED`).
	10	Parameter-Fehler (falsche `No`).
	11	Access-Fehler: Karte für anderes Anwenderprogramm reserviert (**get_last_error**-Returncode: `RTC5_ACCESS_DENIED`) oder die Versionsprüfung hat einen Fehler festgestellt (OUT- oder RTC-Version nicht kompatibel mit der aktuellen DLL-Version, **get_last_error**-Returncode: `RTC5_ACCESS_DENIED` | `RTC5_VERSION_MISMATCH`).
	12	Warnung: 3D-Tabelle oder Dim = 3 ausgewählt, aber 3D-Option 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 oder INTERNAL-BUSY (**get_last_error**-Returncode: `RTC5_BUSY`).
	14	PCI-Upload-Fehler (Treiber-Fehler, nur bei Download-Überprüfung).
	15	Verify-Fehler (nur bei Download-Überprüfung)

Kontrollbefehl	load_correction_file
Hinweise	Hinweise zum Laden von Korrekturtabellen: • Die RTC5 kann vier verschiedene Korrekturtabellen zur selben Zeit speichern, z. B. zur Verwendung in einer Konfiguration mit mehreren Scan-Köpfen. • Korrekturtabellen mit den Nummern 3 und 4 müssen unbedingt erst nach **load_program_file** geladen werden. Diese Tabellen belegen die obere Hälfte des für die Datenaufzeichnung mit **set_trigger** oder **set_trigger4** reservierten Speicherbereichs (siehe Hinweise bei diesen Befehlen). Bei **load_program_file** wird dieser Speicherbereich automatisch initialisiert und evtl. vorher geladene Korrekturtabellen mit den Nummern 3 und 4 gehen verloren. • 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). • Der Name der zu ladenden Korrekturdatei muss bei **load_correction_file** als Pointer auf einen nullterminierten ANSI-String übergeben werden. Wird für `Name` ein Null-Pointer (0) angegeben, wird die entsprechende Tabelle durch eine 1to1-Tabelle ersetzt (für `Dim` = 3 bei freigeschalteter 3D-Option durch eine 3D-1to1-Tabelle, sonst durch eine 2D-1to1-Tabelle). Die im vorigen Hinweis erwähnten Parameterwerte werden dabei größtenteils auf 0 gesetzt. Ein leerer String (“”) für `Name` erzeugt einen Fehlercode 3. • Ist die 3D-Option *nicht* freigeschaltet (Standard), werden – unabhängig vom angegebenen Wert für `Dim` – sowohl 2D- als auch 3D-Korrekturdateien als 2D-Korrekturtabellen gespeichert (3D-Korrekturdateien enthalten auch 2D-Tabellen). Der 3D-Anteil der 3D-Korrekturdateien wird hier ignoriert. • Ist die 3D-Option 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 stets nur eine 2D-Korrektur berechnet, die Z-Achse bleibt dann unverändert, so als ob keine 3D-Option freigeschaltet wäre. – Für `Dim` = 3 werden bei freigeschalteter 3D-Option 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 dem Befehl **load_z_table** geladen werden (siehe Seite 196). – Alle anderen Werte von `Dim` ändern am Korrekturdatei-Typ nichts.

Kontrollbefehl	load_correction_file
Hinweise (Forts.)	Hinweise zum Zuweisen von Korrekturtabellen mit **select_cor_table**: • Eine Zuweisung einer (oder zweier) auf der RTC5 gespeicherten Korrekturtabelle(n) an den Scan-Kopf (bzw. an die beiden Scan-Kopf-Anschlüsse) erfolgt mit dem Befehl **select_cor_table** oder **select_cor_table_list**. • Ab Version DLL 521, OUT 521 ruft **load_correction_file** 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 bleibt der automatische Aufruf von **select_cor_table** allerdings ohne Wirkung. Wird **load_correction_file** *nach* **load_program_file** aufgerufen, dann erfolgt der **select_cor_table**-Aufruf mit den Parameterwerten (`HeadA` = 1, `HeadB` = 0) bzw. mit den Parametern des zuletzt (nach **load_program_file**) erfolgten **select_cor_table**-Aufrufs. Ab Version DLL 527, OUT 529 kehrt der Befehl erst zur Anwendung zurück, wenn der durch **select_cor_table** ausgelöste Sprung auf die korrigierte Galvanometerposition abgeschlossen ist. • Bei älteren Versionen (vor DLL 521, OUT 521) muss nach **load_correction_file** bzw. **load_program_file** in jedem Fall **select_cor_table** oder **select_cor_table_list** aufgerufen werden. • Für weitere Informationen hierzu, siehe Hinweise auf Seite 138. Sonstige Hinweise: • In RTC5-Korrekturtabellen sind Parameter eingetragen, die bei der RTC4/RTC3/RTC2 noch einer mitgelieferten ReadMe-Datei zu entnehmen waren und händisch in das Anwenderprogramm übertragen werden mussten. Diese Parameter können bei der RTC5 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. • Für die Zeitdauer des Befehls **load_correction_file** können keine weiteren Kontrollbefehle ausgeführt werden, externe Starts werden unterdrückt. • Der Befehl **load_correction_file** wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). Bei älteren Versionen (vor DLL 521, OUT 521) sollte aber dann unbedingt vor dem erneuten Starten einer Liste **select_cor_table** aufgerufen werden. Dann ist ein fließender Übergang mit `jump_speed` von den alten Galvanometerpositionen in die neuen garantiert. Andernfalls können harte Sprünge auftreten (siehe auch Hinweise auf Seite 138). • Bei einem `RTC5_VERSION_MISMATCH`-Fehler (Rückgabewert 11) muss die DLL-Version passend zu den Programmdateien gewählt und die Karte einmal stromlos gemacht werden (um die Programmsoftware zu entladen) oder die zur DLL-Version passende Programmdateien mit dem Multi-Board-Befehl **n_load_program_file** neu geladen werden (der Single-Board-Befehl **load_program_file** erhält nach `RTC5_ACCESS_DENIED` keinen Zugriff auf die Karte). Anschließend (und nachdem die Karte wieder durch **acquire_rtc** bzw. **select_rtc** in Besitz genommen und gegebenenfalls durch **select_rtc** als Default-Karte spezifiziert wurde) kann **load_correction_file** wieder normal ausgeführt werden.

Kontrollbefehl	load_correction_file
RTC4→ RTC5	Bis auf die Grundfunktion des Befehls und seiner ersten beiden Parameter (Laden von Korrekturdateien) hat sich die Funktionalität des Befehls stark geändert: • Der Befehl 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 RTC5 (mit freigeschalteter 3D-Option) können jetzt auch *zwei* 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. • Ab Version DLL 521, OUT 521 wird automatisch **select_cor_table** aufgerufen (siehe Hinweise oben bzw. Seite 138). • Ab Version DLL 527, OUT 529 Befehlsrückkehr erst nach Ende der Korrekturbewegung (siehe Hinweise oben) • Ab Version DLL 533, OUT 534 können auch bis zu vier Korrekturdateien geladen werden (siehe Hinweise oben)
Versionsinfo	Letzte Änderung Version DLL 533, OUT 534 (siehe oben).

Kontrollbefehl	load_disk	
Funktion	Lädt die zuvor mit **save_disk** in einer binären Datei gespeicherten indizierten Zeichen, Zeichenfolgen und Unterprogramme in den geschützten Speicherbereich (“Liste 3”) und liefert die Anzahl der tatsächlich geladenen Listenbefehle zurück.	
Aufruf	`NoOfLoadedCommands = load_disk( Name, Mode )`	
Parameter	`Name`	Dateiname als Pointer auf einen nullterminierten ANSI-String.
	`Mode`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wie geladen werden soll. =0: Die internen Verwaltungstabellen für indizierte Zeichen, Zeichenfolgen und Unterprogramme werden initialisiert (alle alten Referenzierungen gehen dabei verloren), und der Input-Pointer 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 Input-Pointer wird hinter das zuletzt (mit **load_char**) gespeicherte Zeichen, die zuletzt (mit **load_text_table**) gespeicherte Zeichenfolge bzw. 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.	
Hinweise	• Für `Name` = 0 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 der Befehl **load_disk** nur zum Einlesen von Dateien verwendet werden, die mit dem Befehl **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 entsprechenden Listenbefehle in den geschützten Speicherbereich geschrieben und gemäß der in der Datei gespeicherten Indexzuordnung entsprechende 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. • Zusammen mit **save_disk** kann **load_disk** u.a. zur Defragmentierung des geschützten Speicherbereichs und zum nachträglichen Schützen von Unterprogrammen verwendet werden (siehe Seite 84 und Seite 85). 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, sofern keine Indices in verschiedenen Dateien gleichzeitig vorkommen. Speicherlücken im geschützten Speicherbereich, die beim Entreferenzieren von indizierten Zeichen, Zeichenfolgen oder Unterprogrammen entstehen, werden dadurch geschlossen.	

Kontrollbefehl	load_disk
Hinweise (Forts.)	• Wird beim Laden das Ende des geschützten Speicherbereichs 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**-Befehlen) nicht gespeichert. Vor einem **load_disk**-Befehl sollte daher sichergestellt werden, dass der geschützte Speicherbereich in ausreichender Größe jenseits des Listenbereichs (“Liste 1” und “Liste 2”) konfiguriert ist. Der **save_disk**-Befehl gibt die Anzahl der in der Datei gespeicherten Listenbefehle zurück. Beim Besitzwechsel einer Karte ist es Aufgabe des Anwenders sicherzustellen, dass die Informationen über die Speicherkonfiguration konsistent sind (`Mem1` und `Mem2` werden aus der DLL abgefragt, nicht aus der Karte – siehe auch **get_config_list** und Seite 96). • 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. • Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`), wenn “Liste 3” kein Speicherbereich zugewiesen wurde (`Mem1` + `Mem2` = 2^{20}). Der Befehl wird ebenfalls ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Wenn dagegen eine Liste nur mit **set_wait** angehalten wurde (PAUSED-Status gesetzt), kann **load_disk** ausgeführt werden. Der Benutzer ist 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 **load_disk**-Befehl abgearbeitet wird, sind keine weiteren Befehle ausführbar. Externe Starts werden für die Laufzeit des Befehls unterdrückt.
RTC4→ RTC5	Neuer Befehl.
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-Encoder-Kompensation für XY-Tische, siehe Abschnitt "2D-Encoder-Kompensation für XY-Tische", Seite 206.	
Aufruf	`NoOfDataPoints = load_fly_2d_table( Name, No )`	
Parameter	`Name`	Name der Textdatei als Pointer auf einen nullterminierten ANSI-String (die Textdatei kann eine oder mehrere Tabellen enthalten).
	`No`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird spezifiziert, welche Tabelle aus der Textdatei geladen werden soll (der Parameter entspricht der Erweiterung `<No>` der Anweisung `[Fly2DTable<No>]` am Anfang der gewünschten Tabelle).
Rückgabe	32-Bit-Wert mit Vorzeichen (im Fehlerfall ein positiver Fehlercode, im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte).	
	Wert	Beschreibung
	< 0	Erfolg. Der Absolutwert des Rückgabewertes ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert (siehe auch Seite 207).
	0	Für `Name` = 0 (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 oder INTERNAL-BUSY, kein Download (**get_last_error**-Returncode `RTC5_BUSY`).
	8	Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC5_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC5_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC5_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer konnte nicht in der Datei gefunden werden.
Hinweise	• Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für die 2D-Encoder-Kompensation in die Text-Datei sind auf Seite 207 beschrieben. Die größten vorkommenden Stützstellen sollten den Wertebereich -524.288 bis 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. Andernfalls erfolgt ein Clipping. • Der Befehl **load_fly_2d_table** überschreibt jede zuvor für die 2D-Encoder-Kompensation geladene Tabelle. • Für `Name` = 0 wird eine 0-Korrektur-Tabelle für die 2D-Encoder-Kompensation geladen. • Der Befehl **load_fly_2d_table** wird nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Während der Laufzeit des Befehls 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.	

Kontrollbefehl	load_fly_2d_table
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
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.	
Hinweise	• Der Befehl ist identisch zu load_jump_table_offset mit `Offset` = 0.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	• Verfügbar ab Version DLL 525.	
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 als Pointer auf einen nullterminierten ANSI-String (die Textdatei kann eine oder mehrere Tabellen enthalten) oder NULL für eine automatische Ermittlung.
	`No`	Dieser Parameter (32-Bit-Wert ohne Vorzeichen) spezifiziert: • Für `Name` = Dateiname, welche Tabelle aus der Textdatei geladen werden soll (der Parameter entspricht der Erweiterung `<No>` der Anweisung `[JumpTable<No>]` am Anfang der gewünschten Tabelle). • Für `Name` = 0, welches Scan-System (bzw. welcher Scan-Kopf-Anschluss) für die automatische Ermittlung verwendet werden soll. Zulässige Werte: [1, 2].
	`PosAck`	Positionstoleranzwert in [Bits] als 32-Bit-Wert ohne Vorzeichen (nur bei automatischer Ermittlung relevant).
	`Offset`	Offset in [10 µs] als 32-Bit-Wert mit Vorzeichen, der zu allen automatisch ermittelten Delays aufaddiert wird (nur bei automatischer Ermittlung relevant).
	`MinDelay`	Minimales Jump-Delay in [10 µs] als 32-Bit-Wert ohne Vorzeichen (nur bei automatischer Ermittlung relevant).
	`MaxDelay`	Maximales Jump-Delay in [10 µs] als 32-Bit-Wert ohne Vorzeichen (nur bei automatischer Ermittlung relevant).
	`ListPos`	Listenposition (im Bereich von Liste 1 oder 2) für sechs Listenbefehle, die überschrieben werden dürfen (nur bei automatischer Ermittlung relevant).
Rückgabe	Fehlercode als 32-Bit-Wert mit Vorzeichen.	
	–1…– 50	Erfolg für `Name` = Dateiname. Der Absolutwert des Rückgabewertes ist gleich der Zahl der gültigen Datenpunkte, die in der Tabelle gefunden wurden. Ungültige Eingabewerte werden ignoriert (siehe auch Seite 179).
	–1024	Erfolg für `Name` = 0: 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 oder INTERNAL-BUSY, kein Download (**get_last_error**-Returncode `RTC5_BUSY`).
	8	Access-Fehler: Karte durch anderes Anwenderprogramm reserviert (**get_last_error**-Returncode: RTC5_ACCESS_DENIED).
	10	nur für `Name` = 0: Param-Error: HeadNo oder ListPos unzulässig.
	11	PCI-Fehler beim Download (**get_last_error**-Returncode `RTC5_SEND_ERROR`).
	13	Die angegebene Tabellennummer konnte nicht in der Datei gefunden werden.

Kontrollbefehl	load_jump_table_offset
Hinweise	• Zur Verwendung des Befehls siehe "Sprunglängenabhängige Jump-Delays", Seite 178. • Zum Sprungmodus siehe Seite 176. • Die Formatvorschriften zum Eintragen der Tabelle mit den Jump-Delay-Datenpunkten in die Text-Datei sind im Abschnitt "Hinweise zum Laden der ermittelten Jump-Delay-Werte" auf Seite 179 beschrieben. Beim Laden der Tabelle bestimmt die RTC5 durch lineare Interpolation geeignete Werte für den ganzen Sprunglängenbereich. • Der Befehl **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` = 0 wird die Jump-Delay-Tabelle für Kopf `No` automatisch ermittelt und geladen (hierfür 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 "Automatische Ermittlung der Jump-Delay-Tabelle", Seite 180). • 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. • Der Befehl **load_jump_table_offset** wird *nicht* ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Während der Laufzeit des Befehls 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.
RTC4→ RTC5	Neuer Befehl. Für diesen Befehl gibt es keinen RTC4-Kompatibilitätsmodus.
Versionsinfo	• Verfügbar ab Version DLL 526.
Verweise	**load_jump_table**, **set_jump_mode**, **set_jump_mode_list**, **get_jump_table**, **set_jump_table**

Kontrollbefehl	load_list
Funktion	Öffnet den Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Input-Pointer an die angegebene (relative) Position in der gewünschten Liste (“Liste 1” oder “Liste 2”), aber nur, wenn die Liste nicht gerade abgearbeitet wird (BUSY-Status nicht gesetzt) bzw. wenn sie bereits abgearbeitet ist (USED-Status gesetzt). Im Erfolgsfall wird der nächste Listenbefehl an dieser Adresse gespeichert und alle nachfolgenden Listenbefehle an den darauffolgenden Adressen in der gewählten Liste.
Aufruf	`NoOfOpenedList = load_list( ListNo, Pos )`
Parameter	`ListNo` Listennummer der Liste, in die der Input-Pointer gesetzt werden soll, als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [0 … 3]; es werden nur die beiden niederwertigsten Bits berücksichtigt. =1: “Liste 1” wird geöffnet, falls sie nicht BUSY ist (BUSY1-Status nicht gesetzt). =2: “Liste 2” wird geöffnet, falls sie nicht BUSY ist (BUSY2-Status nicht gesetzt). =0: Diejenige Liste wird geöffnet, die momentan nicht BUSY ist. Sind beide Listen nicht BUSY, wird “Liste 1” geöffnet. =3: Diejenige Liste wird geöffnet, die momentan nicht BUSY aber USED ist (USED-Status gesetzt). Sind beide Listen nicht BUSY und beide USED, wird diejenige Liste geöffnet, die beim nächsten automatischen Listenwechsel ausgeführt werden würde. Für Mem2 = 0 (siehe **config_list**) wird “Liste 1” geöffnet, falls sie nicht BUSY ist (ListNo = 0…2) bzw. falls sie nicht BUSY aber USED ist (ListNo = 3). `Pos` Position des Input-Pointers (Offset bzgl. des Listenanfangs der jeweiligen Liste) als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … (2^{20}–1)].
Rückgabe	Nummer der geöffneten Liste [1 oder 2] im Erfolgsfall, sonst 0 (als 32-Bit-Wert ohne Vorzeichen).
Hinweise	• Der **load_list**-Befehl (`ListNo` = 3) hilft u.a. beim alternierenden Listenwechsel, um gezielt auf das Abarbeiten einer Liste zu warten (siehe ”Alternierender Listenwechsel”, Seite 80), 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 *unbedingt*es Laden ausgeführt werden, kann z.B. der Befehl **set_start_list_pos** verwendet werden. • Nach einem erfolgreichen Check der Listennummer (BUSY-Status nicht gesetzt und gegebenenfalls USED-Status gesetzt) wirkt der **load_list**-Befehl wie **set_start_list_pos** (siehe Hinweise dort). • Falls der **load_list**-Befehl nicht erfolgreich war (Rückgabewert 0), wird keine Liste geöffnet und der Input-Pointer auf eine illegale Position gesetzt. Es ist dann so lange keine Eingabe von weiteren Listenbefehlen möglich, bis der Input-Pointer (z.B. mit Hilfe eines erneuten **load_list**-Befehls mit positivem Resultat, mit **set_start_list_pos** etc.) wieder gültig gesetzt wird. Es obliegt den Benutzern, auf den Rückgabewert 0 geeignet zu reagieren. Der **load_list**-Befehl führt *keine* Warteschleife aus und blockiert *nicht* die Ausführung nachfolgender Kontrollbefehle.

Kontrollbefehl	load_list
Hinweise (Forts.)	• Für `ListNo` = 0…2 sollte bei der Verwendung des **load_list**-Befehls berücksichtigt werden, dass sich der BUSY-Status 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 Input-Pointer nicht vom Output-Pointer 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 evtl. erfolgten automatischen Listenwechsel) zum Laden geöffnet wird. Insbesondere wenn keine Liste BUSY ist und beide Listen USED sind (z. B. nach der Initialisierung, einem **stop_execution** bzw. einem externen Listen-Stopp) ist die geöffnete Listennummer mit dem nächsten automatischen Listenwechsel synchronisiert.
RTC4→ RTC5	Neuer Befehl.
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 positionsabhängige Lasersteuerung (radiale Korrektur, siehe Abschnitt "Positionsabhängige Lasersteuerung", Seite 161).	
Aufruf	`NoOfDataPoints = load_position_control( Name, No )`	
Parameter	`Name`	Name der Textdatei als Pointer auf einen nullterminierten ANSI-String (die Textdatei kann eine oder mehrere Tabellen enthalten).
	`No`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird spezifiziert, welche Tabelle aus der Textdatei geladen werden soll (der Parameter entspricht der Erweiterung `<No>` der Anweisung `[PositionCtrlTable<No>]` am Anfang der gewünschten Tabelle).
Rückgabe	32-Bit-Wert mit Vorzeichen (im Fehlerfall ein positiver Fehlercode, im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte).	
	Wert	Beschreibung
	–1…– 50	Erfolg. Der Absolutwert des Rückgabewertes 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 162).
	–256	Für `Name` = 0 (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 oder INTERNAL-BUSY, kein Download (**get_last_error**-Returncode `RTC5_BUSY`).
	8	Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC5_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC5_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC5_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer konnte nicht in der Datei gefunden werden.
Hinweise	• Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für die positionsabhängige Lasersteuerung in die Text-Datei sind im Abschnitt "Hinweise zum Laden der Skalierungsfunktion", Seite 162 beschrieben. Beim Laden der Tabelle bestimmt die RTC5 durch lineare Interpolation geeignete Werte für den ganzen Ansteuerbereich. • Der Befehl **load_position_control** überschreibt jede zuvor für die positionsabhängige Lasersteuerung geladene Skalierungsfunktion. • Für `Name` = 0 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. • Wirksam wird die positionsabhängige Lasersteuerung bei nachfolgenden Mark- oder Arc-Befehlen erst dann, wenn sie mit **set_auto_laser_control** initialisiert wurde. Deaktiviert wird die positionsabhängige Lasersteuerung mit **set_auto_laser_control** (`Ctrl` = 0) oder durch Laden von *Scale*(*Position*)=1.0. Siehe auch "Positionsabhängige Lasersteuerung", Seite 161.	

Kontrollbefehl	load_position_control
Hinweise (Forts.)	• Der Befehl **load_position_control** wird *nicht* ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Während der Laufzeit des Befehls 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.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_auto_laser_control**

Kontrollbefehl	load_program_file	
Funktion	Führt einen Reset der Karte durch, führt einen DSP-Speicher-Check aus, lädt die Programmdatei `RTC5OUT.out` zusammen mit den Dateien `RTC5RBF.rbf` und `RTC5DAT.dat` aus dem angegebenen Verzeichnis in den Speicher der RTC5, führt eine Versionsprüfung durch und startet den Signalprozessor (DSP).	
Aufruf	`ErrorNo = load_program_file( pPath )`	
Parameter	`pPath`	Pfadname des Verzeichnisses, in dem sich die Dateien `RTC5OUT.out`, `RTC5RBF.rbf` und `RTC5DAT.dat` befinden (als Pointer auf einen nullterminierten String).
Rückgabe	Fehlercode als 32-Bit-Wert ohne Vorzeichen.	
	Wert	Beschreibung
	0	Erfolg.
	1	Reset-Fehler: die Karte konnte nicht resettet werden.
	2	Unreset-Fehler: die Karte läuft nicht wieder an.
	3	Datei-Fehler: Datei nicht gefunden oder nicht geöffnet.
	4	Format-Fehler: `RTC5OUT.out` hat falsches Format. PCI-Fehler: der Treiber meldet einen Fehler beim PCI-Transfer, weil der Treiber und die DLL nicht zusammen passen. Das kommt insbesondere dann vor, wenn DLL 533 oder kleiner zusammen mit dem Treiber 6.1.7600.16385 verwendet wird; oder wenn überhaupt die PCI-Kommunikation gestört ist. Bis zur DLL 536 bedeutet der Fehlercode 4 auch einen PCI Fehler. Ab DLL 537 gibt es für PCI Fehler den Fehlercode 16, siehe dort.
	5	System-Fehler: nicht genügend Windows-Speicher.
	6	Access-Fehler: Karte für anderes Anwenderprogramm reserviert.
	7	Versions-Fehler: DLL-Version (der RTC5-DLL), RTC-Version (der Firmwaredatei) und HEX-Version (der DSP-Programmdatei) passen nicht zusammen.
	8	Systemtreiber nicht gefunden (**get_last_error**-Returncode: `RTC5_ACCESS_DENIED`).
	9	DriverCall-Fehler: das Laden der `RTC5OUT.out`-Datei schlug fehl.
	10	Konfiguration-Fehler: die DSP-Register-Initialisierung schlug fehl.
	11	Firmware-Fehler: das Laden der `RTC5RBF.rbf`-Datei schlug fehl.
	12	PCI-Download-Fehler: das Laden der `RTC5DAT.dat`-Datei schlug fehl.
	13	Busy-Fehler: Download gesperrt, Karte BUSY oder INTERNAL-BUSY (**get_last_error**-Returncode: RTC5_BUSY).
	14	DSP-Speicher-Fehler.
	15	Verify-Fehler (nur bei Download-Überprüfung).
	16	PCI-Fehler: ab DLL 537. Oder möglicherweise ein Treiber - DLL-Konflikt, siehe Hinweis auf Seite 26: “Ältere RTC5-DLL-Dateien (Version DLL 533 und früher) sind nicht mit den neuen WDF-Treibern kompatibel.”.

Kontrollbefehl	load_program_file
Hinweise	• Für `pPath` = 0 wird der Pfad des aktuellen Arbeitsverzeichnisses des Anwenderprogramms verwendet. Achtung: Das aktuelle Arbeitsverzeichnisses des Anwenderprogramms 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). • Der Befehl **load_program_file** muss nach jedem Hardware-Reset (Einschalten) bei der Initialisierung der RTC5-Karte am Beginn des jeweils ersten Anwenderprogramms ausgeführt werden (siehe "Initialisierung der Karte", Seite 69). Der Befehl 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 RTC5-Karten als Master und Slave miteinander verbunden sind, dann muss auf allen Karten der Befehl **load_program_file** erteilt worden sein, bevor einzelne Karten mit weiteren Befehlen initialisiert und betrieben werden (siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93). • Der Befehl **load_program_file** setzt die RTC5 zurück (dies entspricht einem Hardware-Reset), initialisiert die Speicherkonfiguration (in der Standard-Konfiguration), führt einen DSP-Speicher-Check aus, lädt die Firmware (`RTC5RBF.rbf`), die Programmdatei `RTC5OUT.out` und eine binäre Hilfsdatei (`RTC5DAT.dat`) und startet den Signalprozessor (DSP). Nach Ausführung des Befehls **load_program_file** ist der Laserfokus im Zentrum des Bildfeldes (0|0) und die Lasersteuerung ist ***deaktiviert***. • Der Befehl **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. • Beim Reset der RTC5 wird der Listenspeicher gelöscht und es werden alle Parameter (wie die Speicherkonfiguration, interne Variablen, Matrizen, Offsets und Zuordnungen), die zuvor mit Befehlen wie **config_list** oder **select_cor_table** eingestellt wurden, gelöscht bzw. auf ihre Standardwerte zurückgesetzt. Beim Reset wird zwar eine Korrekturtabelle gemäß `select_cor_table(1,0)` zugewiesen aber keine Scannerbewegung zur korrigierten Ausgabeposition durchgeführt. • Die Funktion kehrt erst dann wieder zum aufrufenden Programm zurück, wenn die Initialisierung des DSP komplett abgeschlossen ist. • Der Befehl **load_program_file** wird nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste zuvor ordnungsgemäß beendet oder mit **set_wait** (PAUSED-Status gesetzt), **stop_execution** oder mit einem externen Stopp angehalten wurde.

Kontrollbefehl	load_program_file
Hinweise (Forts.)	• Die Dateien `RTC5OUT.out`, `RTC5RBF.rbf` und `RTC5DAT.dat` sind im RTC5-Software-Paket enthalten. Zur Kennzeichnung und Verwaltung unterschiedlicher Softwareversionen werden die Dateien zusätzlich gezippt ausgeliefert (die `zip`-Dateinamen `RTC5<…>_<Version>.zip` enthalten die jeweiligen Versionsnummern). Kopieren oder entpacken Sie die drei Dateien auf die Festplatte Ihres PCs. • Die RTC5-DLL und die Dateien `RTC5OUT.out`, `RTC5RBF.rbf` und `RTC5DAT.dat` können nicht in beliebigen Programmversionen miteinander kombiniert werden (die `zip`-Dateien enthalten jeweils eine Text-Datei mit entsprechenden Versions-Informationen). Der Befehl **load_program_file** führt eine entsprechende Versionsprüfung durch. Liegt ein Versionsfehler vor, bleiben die geladenen Dateien zwar im Speicher der RTC5, 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 `RTC5_ACCESS_DENIED` | `RTC5_VERSION_MISMATCH`). Lediglich der Multi-Board-Befehl **n_load_program_file** kann dann noch aufgerufen werden, um eine korrekte Programmversion zu laden. Dazu wird temporär die Zugriffsberechtigung angefordert, nach dem Download (sofern die Karte nicht von einem anderen Anwenderprogramm beansprucht ist) aber wieder zurückgegeben (**n_load_program_file** führt kein **acquire_rtc** aus). Der Single-Board-Befehl **load_program_file** erhält in diesem Falle keine Zugriffsberechtigung. Alternativ kann die Karte auch stromlos gemacht werden, um die Programmsoftware zu entladen (danach führt auch **load_program_file** wieder zum Ziel). • Die Dateinamenerweiterung für RTC5-Programmdateien ist nicht mehr `*.hex` (wie bei der RTC4, RTC3 und RTC2), sondern `*.out`. Die Versionsnummer wird aber nach wie vor mit dem Befehl **get_hex_version** ausgelesen. • Den Befehl **dsp_start** gibt es bei der RTC5 nicht mehr (siehe Seite 642). Der DSP wird automatisch durch **load_program_file** gestartet.
RTC4→ RTC5	• Der Befehlsparameter gibt bei der RTC5 einen Verzeichnisnamen, bei der RTC4 einen Dateinamen an. • Der Befehl lädt jetzt drei Dateien, die mit festen Formaten und Namensbezeichnungen (`RTC5OUT.out`, `RTC5RBF.rbf`, `RTC5DAT.dat`) bereitgestellt werden müssen (s. o.). • Nach Ausführung des Befehls ist die Lasersteuerung ***deaktiviert***.
Versionsinfo	Letzte Änderung Version DLL 537.

Kontrollbefehl	load_stretch_table	
Funktion	Lädt eine Tabelle aus einer ASCII-Textdatei für eine erweiterte 3D-Korrektur, siehe Kapitel 8.6.5 "Erweiterte 3D-Korrektur", Seite 198.	
Aufruf	`NoOfDataPairs = load_stretch_table( Name, No )`	
Parameter	`Name`	Name der Textdatei als Pointer auf einen nullterminierten ANSI-String (die Textdatei kann eine oder mehrere Tabellen enthalten) oder NULL.
	`No`	32-Bit-Wert mit Vorzeichen. • Für `No` ≥ 0 wird mit diesem Parameter spezifiziert, welche Tabelle aus der Textdatei geladen werden soll (der Parameter entspricht der Erweiterung `<No>` der Anweisung `[StretchTable<No>]` am Anfang der gewünschten Tabelle). • `No` < 0: Reserviert.
Rückgabe	32-Bit-Wert mit Vorzeichen (im Fehlerfall ein positiver Fehlercode, im Erfolgsfall die negative Anzahl der gefundenen Datenpaare).	
	Wert	Beschreibung
	< 0	Erfolg. Der Absolutwert des Rückgabewertes 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 oder INTERNAL-BUSY, kein Download (**get_last_error**-Returncode `RTC5_BUSY`).
	6	Daten-Fehler: es fehlen Datenpaare.
	11	PCI-Fehler (**get_last_error**-Returncode `RTC5_SEND_ERROR`).
	13	Die angegebene Tabellennummer konnte nicht in der Datei gefunden werden.
	15	Verify-Fehler (**get_last_error**-Returncode `RTC5_VERIFY_ERROR`, nur mit aktivierter **set_verify**-Option möglich).
Hinweise	• Details zur erweiterten 3D-Korrektur (z. B. die Formatvorschriften zum Eintragen der Tabelle mit den Datenpaaren) sind auf Seite 198 beschrieben. • Eine erfolgreich geladene Tabelle aktiviert die neue erweiterte 3D-Korrektur. Dabei überschreibt **load_stretch_table** jede zuvor für die erweiterte 3D-Korrektur geladene Tabelle. • Falls `Name` *nicht* NULL ist aber keine Tabelle erfolgreich gelesen wird, so gibt **load_stretch_table** einen Fehlercode zurück (z. B. Fehlercode 13, falls ein `No`-Wert spezifiziert wurde, für den kein `[StretchTable<No>]`-Eintrag in der Textdatei existiert), 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. • Der Befehl **load_stretch_table** wird nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Während der Laufzeit des Befehls 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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.

Kontrollbefehl	load_sub
Funktion	Versieht ein durch nachfolgende Listenbefehle definiertes Unterprogramm mit dem gewünschten Index und lädt es in den geschützten Speicherbereich (“Liste 3”).
Aufruf	`load_sub( Index )`
Parameter	`Index` Index des indizierten Unterprogramms als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich [0 … 1023].
Hinweise	• Es können 1024 indizierte Unterprogramm gespeichert werden. Für `Index` > 1023 wird der **load_sub**-Befehl ignoriert (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Die Adressen im geschützten Speicherbereich, wo die jeweiligen Unterprogramme gespeichert werden, werden automatisch ermittelt und intern verwaltet. • Indizierte Unterprogramme müssen jeweils mit einem **list_return**-Befehl abgeschlossen werden. Erst damit werden die Befehle endgültig gespeichert, die Startadresse und evtl. andere Informationen wie die Anzahl der Befehle in die interne Verwaltungstabelle eingetragen und auch ein Flush der gepufferten Listeneingabe ausgelöst (siehe Seite 75). Andernfalls (der Input-Pointer wird ohne vorhergehenden **list_return**-Befehl verändert) ist das Unterprogramm mit diesem Index nicht verfügbar. • Ein indiziertes Unterprogramm wird nicht gespeichert, falls nicht zuvor der geschützte Speicherbereich (“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 “Aufruf von Unterprogrammen”, Seite 83). • Beachten Sie auch alle Hinweise in Abschnitt “Indizierte Unterprogramme”, Seite 82.
RTC4→ RTC5	Neuer Befehl.
Verweise	**list_return**, **sub_call**, **load_char**, **load_text_table**

Kontrollbefehl	load_text_table
Funktion	Versieht eine durch nachfolgende Listenbefehle definierte Zeichenfolge mit dem gewünschten Index und lädt sie in den geschützten Speicherbereich (“Liste 3”).
Aufruf	`load_text_table( Index )`
Parameter	`Index` Index der indizierten Zeichenfolge als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich [0 … 41].
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 der **load_text_table**-Befehl ignoriert (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Wenn auch die Ziffern statt über einzelne Zeichen mit **mark_char** über **mark_text** definiert werden, kann auch hierfür der Zeichensatz nachträglich noch umgeschaltet werden (siehe **select_char_set**). • Die Adressen im geschützten Speicherbereich, 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**-Befehl abgeschlossen werden. Erst damit werden die Befehle endgültig gespeichert, die Startadresse und evtl. andere Informationen wie die Anzahl der Befehle in die interne Verwaltungstabelle eingetragen und auch ein Flush der gepufferten Listeneingabe ausgelöst (siehe Seite 75). Andernfalls (der Input-Pointer wird ohne vorhergehenden **list_return**-Befehl verändert) ist die Zeichenfolgedefinition mit diesem Index nicht verfügbar. • Eine indizierte Zeichenfolgedefinition wird nicht gespeichert, falls nicht zuvor der geschützte Speicherbereich (“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 87.
RTC4→ RTC5	Neuer Befehl.
Verweise	**list_return**, **load_sub**, **load_char**

Kontrollbefehl	load_varpolydelay	
Funktion	Lädt eine Tabelle mit Datenpunkten für die Skalierungsfunktion des benutzerdefinierten variablen Polygon-Delays (siehe Seite 120) aus einer ASCII-Textdatei.	
Aufruf	`NoOfDataPoints = load_varpolydelay( Name, No )`	
Parameter	`Name`	Name der Textdatei als Pointer auf einen nullterminierten ANSI-String (die Textdatei kann eine oder mehrere Tabellen enthalten).
	`No`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird spezifiziert, welche Tabelle aus der Textdatei geladen werden soll (der Parameter entspricht der Erweiterung `<No>` der Anweisung `[VarPolyTable<No>]` am Anfang der gewünschten Tabelle).
Rückgabe	32-Bit-Wert mit Vorzeichen (im Fehlerfall ein positiver Fehlercode, im Erfolgsfall die negative Anzahl der gefundenen Datenpunkte).	
	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 Seite 120).
	–1024	für `Name` = 0: es wurde die gemäß 1–cos(ϕ) initialisierte Tabelle intern (wie bei Programmstart) geladen (siehe Abb. 36).
	1	Keine gültigen Datenpunkte gefunden (wohl aber Tabelle `No`).
	3	Datei nicht gefunden.
	4	DSP-Speicher-Fehler.
	5	Busy-Fehler, Karte ist BUSY oder INTERNAL-BUSY, kein Download (**get_last_error**-Returncode `RTC5_BUSY`).
	8	Karte durch anderes Anwenderprogramm gesperrt (**get_last_error**-Returncode `RTC5_ACCESS_DENIED`).
	11	PCI-Fehler (**get_last_error**-Returncode `RTC5_SEND_ERROR`), Verify-Fehler (**get_last_error**-Returncode `RTC5_VERIFY_ERROR`).
	13	Die angegebene Tabellennummer konnte nicht in der Datei gefunden werden.
Hinweise	• Die Formatvorschriften zum Eintragen der Tabelle mit den Datenpunkten für das benutzerdefinierte variable Polygon-Delay in die Text-Datei sind im Abschnitt "Benutzerdefinierte variable Polygon-Delays" auf Seite 120 beschrieben. Beim Laden der Tabelle bestimmt die RTC5 durch lineare Interpolation geeignete Werte für den ganzen Winkelbereich. • Der Befehl **load_varpolydelay** überschreibt jede zuvor für das variable Polygon-Delay geladene Tabelle. • Für `Name` = 0 wird (wie bei Initialisierung mit **load_program_file**) die interne Standard-Tabelle für das variable Polygon-Delay (1–cos(ϕ), siehe Abb. 36) geladen. • Der Befehl **load_varpolydelay** wird *nicht* ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Während der Laufzeit des Befehls werden externe Starts unterdrückt. • Vor dem Laden einer Tabelle führt **load_varpolydelay** einen DSP-Speicher-Check aus, der im Fehlerfall zum Fehlercode 4 führt.	

Kontrollbefehl	load_varpolydelay
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: • Um zur internen Standard-Polygon-Delay-Tabelle zurückzukehren ist (anders als bei der RTC4) *kein* Reset bzw. erneutes Programmladen mit **load_program_file** mehr erforderlich (s. o. `Name` = 0). • Für die ASCII-Textdatei sind beliebige Dateinamenerweiterungen (also nicht nur `*.STB` wie bei der RTC4) zulässig.
Verweise	**load_program_file**, **set_delay_mode**

Kontrollbefehl	load_z_table
Funktion	Lädt die Koeffizienten *A*, *B* und *C* in die aktuell zugewiesene 3D-Korrekturtabelle.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), liefert der Befehl Fehlercode 12 oder 13 zurück und hat sonst keine Wirkung.
Aufruf	`ErrorNo = load_z_table( A, B, C )`
Parameter	`A, B, C` Koeffizienten (im 64-Bit-IEEE-Gleitkommaformat) der parabolischen Funktion $z_{out} = A + Bl + Cl^2$, die zur Berechnung der z-Ausgabewerte verwendet wird. Zulässiger Wertebereich: • Für `A`: [–67108864,0 … 67108864,0] • Für `B`: [–2048,0 … 2048,0] • Für `C`: [–16,0 … 16,0] Überlaufende Werte werden auf den jeweiligen Randbereich geclippt.
Rückgabe	Fehlercode als 32-Bit-Wert ohne Vorzeichen: Die Fehlerbits mit den Werten 1 bis 64 können auch kombiniert vorkommen, aber nicht zusammen mit den Fehlercodes 11 bis 14, letztere auch nur separat. Die Warnungen 12 und 13 werden nur dann zurückgegeben, wenn kein anderer Fehler vorliegt. Wert – Beschreibung 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- oder INTERNAL-BUSY-Fehler, für eine genaue Ursache siehe **get_last_error**). 11 – Zugriff verweigert. 12 – 3D-Option nicht freigeschaltet. 13 – Es ist aktuell keine 3D-Tabelle zugeordnet. 14 – Treiber nicht gefunden.

Kontrollbefehl	load_z_table
Hinweise	• Der Befehl wird nur zur Neu-Kalibrierung der Z-Achse in einem 3-Achsen-Scan-System benötigt (zur Bestimmung geeigneter Koeffizienten siehe Seite 196). Es können positive oder negative Koeffizienten angegeben werden. Die Koeffizienten sollten möglichst so gewählt werden, dass alle z-Ausgabewerte im Bereich [−32.768 … +32.767] liegen. • Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Der Befehl sollte stets *nach* dem Befehl **load_correction_file** aufgerufen werden, da **load_correction_file** die drei Koeffizienten auf die in der geladenen Korrekturtabelle festgelegten Werte setzt. • Vor dem **load_z_table** unmittelbar nachfolgenden Listenbefehl wird ein fließender Übergang der letzten z-Position in die geänderte Position mit `jump_speed` 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.
RTC4→ RTC5	Unveränderte Funktionalität (bis auf veränderte Wertebereiche und Fehlercodes). Wenn die korrekten Kalibrierungsfaktoren verwendet werden (siehe Seite 194), können bei gleichem 3-Achsen-Scan-System für RTC4- und RTC5-Karten dieselben ABC-Koeffizienten verwendet werden.
Versionsinfo	Letzte Änderung Version DLL 516.
Verweise	**get_z_distance**, **read_abc_from_file**, **write_abc_to_file**

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 Bereich: $0 \leq$ delay $\leq (2^{32}-1)$.
Hinweise	• Der Befehl schaltet die "Laser active"-Lasersignale gegebenenfalls nach einem LaserOff-Delay aus, wartet ein etwaiges Scanner-Delay ab und wartet mit der weiteren Ausführung der Liste für die angegebene Zeit. • Dieser Befehl sollte immer nach Änderung des Lampenstroms eines YAG-Lasers angewandt werden, damit sich eine konstante Laserleistung einstellen kann. • Der Befehl **list_nop** entspricht `long_delay( 1 )`.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich).

Normaler Listenbefehl	**mark_abs**
Funktion	Führt den Laserfokus mit Markiergeschwindigkeit entlang eines 2D-Vektors von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im zweidimensionalen Bildfeld.
Aufruf	`mark_abs( X, Y )`
Parameter	`X, Y` Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
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 pro ms* ausgeführt. • Die "Laser active"-Lasersteuersignale werden automatisch am Anfang der Markierung eingeschaltet (bzw. bleiben nach einem unmittelbar vorhergehenden Mark- oder Bogenbefehl eingeschaltet). Die eingestellten Scanner- und Laser-Delays werden dabei berücksichtigt (siehe Kapitel 7.2 "Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung", Seite 111). Beachten Sie, dass im Sky-Writing-Modus (siehe Seite 127) andere Delays ausgeführt werden. Ausnahme: Mark-Befehl der Länge 0 (siehe Hinweise Seite 115). • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **mark_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **mark_abs**-Befehl (siehe Hinweise dort). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bzgl. des angegebenen Wertes für die Markiergeschwindigkeit gleich behandelt werden. • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Seite 141).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
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 bis 3).
Hinweise	• Der Befehl **mark_char** liest die Startadresse des indizierten Zeichens entsprechend dem ausgewählten Index aus der internen Verwaltungstabelle und ruft dann den Befehl **list_call** auf (siehe auch Hinweise dort), der dann die entsprechende Befehlsliste startet. • Der Befehl **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 Listenbereich (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_nop** ausgeführt (siehe Seite 250). Ob für einen bestimmten Index ein Zeichen referenziert ist, kann mit **get_char_pointer** (`Char`) abgefragt werden. Falls *kein* Zeichen referenziert ist, gibt dieser Befehl 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 `RTC5_PARAM_ERROR`). • Absolute Vektor- und Bogenbefehle 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-, Bogen- oder Jump-Befehle enthalten oder es ist der Befehl **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 87.
RTC4→ RTC5	Neuer Befehl.
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- und Bogenbefehle (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 bis 3).
Hinweise	• Der Befehl **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→ RTC5	Neuer Befehl.
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`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen, zulässiger Wertebereich: [0 … 7]) wird ausgewählt, 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, …).
	`Mode`	Auswahl des Formats als 32-Bit-Wert ohne Vorzeichen (zulässiger Wertebereich: [0 … 3]). a) Anzahl der führenden Nullstellen für die Tagesnummer (betrifft nur `Part` = 2, 4, 6 oder 7). 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) Auswahl des Schriftsatzes für die Ziffernmarkierung (betrifft nur `Part` = 0, 2 oder 4 …7). 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 bzw. 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).

Normaler Listenbefehl	mark_date
Hinweise	• Vor einer Datumsmarkierung sollte (nach jedem Neustart) ein Abgleich zwischen RTC5 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 "Markieren von Uhrzeit, Datum und Seriennummern", Seite 170). • Das vollständige Datum wird durch mehrfachen Aufruf von **mark_date** markiert. • Der Befehl **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 Seite 88). 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 "Zeichensätze und Zeichenfolgen", Seite 87). • Für `Part` > 7 und/oder `Mode` > 3 wird **mark_date** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Absolute Vektor- und Bogenbefehle werden nach Aufruf mit **mark_date** auch absolut ausgeführt. Wenn Datumsmarkierungen an unterschiedlichen Stellen im Bildfeld ausgeführt werden sollen, dürfen die entsprechenden Zeichenfolgen (bzw. Zeichen) nur relative Mark-, Bogen- oder Jump-Befehle enthalten oder es ist der Befehl **mark_date_abs** zu verwenden. • Bei der Beschriftung sowohl des gregorianischen als auch des julianischen Datums erfolgt die Umschaltung zwischen zwei Tagen um 24:00 Uhr. Bei beiden Datumsversionen werden die Schaltjahre berücksichtigt.
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich). Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
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 bzw. Zeichen (s. u.) werden absolute Vektor- und Bogenbefehle (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**.
Hinweise	• Der Befehl **mark_date_abs** wirkt wie **mark_date**; der interne Aufruf der indizierten Zeichenfolgen (bzw. Zeichen) erfolgt allerdings mit **list_call_abs** statt mit **list_call**. Enthalten die Befehlslisten der aufgerufenen indizierten Zeichenfolgen (bzw. Zeichen) keine absoluten Befehle, besteht zwischen **mark_date_abs** und **mark_date** kein Unterschied.
RTC4→ RTC5	Neuer Befehl.
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 zweidimensionalen Bildfeld.	
Aufruf	`mark_ellipse_abs( X, Y, Alpha )`	
Parameter	`X, Y`	Absolute Koordinaten des Ellipsenmittelpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt.
	`Alpha`	Winkel zwischen der (mit **set_ellipse** definierten) Ellipsen-Halbachse a und der x-Achse in ° im 64-Bit-IEEE-Gleitkommaformat (der Winkel bezieht sich auf die positive x-Richtung, ein positives Vorzeichen bedeutet im Gegenuhrzeigersinn). `Alpha` wird auf den Wertebereich [0…<360°] normiert.
Hinweise	• Die Parameter von **mark_ellipse_abs** legen nur die Lage und Orientierung des auszuführenden Bogens fest. Seine Form muss vor der Ausführung von **mark_ellipse_abs** mit **set_ellipse** definiert worden sein. Zur Bedeutung der einzelnen Parameter siehe auch Seite 105. • Stimmt der mit **mark_ellipse_abs** und **set_ellipse** festgelegte Startpunkt des Bogens nicht mit der aktuellen Position überein, so wird vor der Markierung ein harter Sprung an den Startpunkt ausgeführt (siehe auch Hinweise auf Seite 105). • Siehe auch alle Hinweise unter **arc_abs**.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die für den Ellipsenmittelpunkt angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–524.288 … 524.287]).	
Versionsinfo	• Verfügbar ab Version DLL 518, OUT 517, RBF 515. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Return-code: `RTC5_TYPE_REJECTED`).	
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 zweidimensionalen Bildfeld.	
Aufruf	`mark_ellipse_rel( dX, dY, Alpha )`	
Parameter	`dX, dY`	*Relativ*koordinaten des Ellipsenmittelpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt.
	`Alpha`	Winkel zwischen der (mit **set_ellipse** definierten) Ellipsen-Halbachse a und der x-Achse in ° im 64-Bit-IEEE-Gleitkommaformat (siehe **mark_ellipse_abs**).
Hinweise	• Die Koordinaten des Ellipsenmittelpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **mark_ellipse_abs** (siehe Hinweise dort). • Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (Bildfeldgröße) geclippt.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die für den Ellipsenmittelpunkt angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–524.288 … 524.287]).	
Versionsinfo	• Verfügbar ab Version DLL 518, OUT 517, RBF 515. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Return-code: `RTC5_TYPE_REJECTED`).	
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 zweidimensionalen Bildfeld.
Aufruf	`mark_rel( dX, dY )`
Parameter	`dX, dY` *Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **mark_abs** (siehe Hinweise dort).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **mark_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **mark_abs_3d** (siehe Hinweise dort).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
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 (bzw. 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.
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**-Befehls 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 dem Befehl **get_list_serial** ausgelesen werden, z.B. um nach einem Listenabbruch festzustellen, ob die aktuelle Nummer bereits hochgezählt wurde oder noch nicht. • 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 Listen-Starts und damit die maximale Anzahl an Beschriftungen einstellen. Dabei werden alle Markierungen auch aus unterschiedlichen Seriennummernsätzen stets zusammen gezählt.

Normaler Listenbefehl	mark_serial
Hinweise (Forts.)	• Mit `digits` = 0 wird eine Leermarkierung ausgeführt. Für M_2 = 0 findet dabei eine Seriennummernerhöhung um 1 statt (nicht um die evtl. mit **set_serial_step** oder **set_serial_step_list** anders eingestellte Schrittweite!). Dies kann nützlich sein, wenn eine einzelne Seriennummer ausgelassen werden soll, kann aber auch (wie bei der RTC4) dazu verwendet werden, um indirekt die Schrittweite (oder bei der RTC5 eine *zusätzliche* Schrittweite) für die Seriennummernerhöhung zu definieren. • Der Befehl **mark_serial** liest für jede zu markierende Seriennummernziffer die Startadresse der zugehörigen indizierten Zeichenfolge (bzw. – für M_1 = 1 – des indizierten Zeichens) 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 enthalten (siehe Seite 88). 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 "Zeichensätze und Zeichenfolgen", Seite 87). • Für unzulässige Werte für `Mode` wird **mark_serial** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Absolute Vektor- und Bogenbefehle werden nach Aufruf mit **mark_serial** auch absolut ausgeführt. Wenn Seriennummernmarkierungen an unterschiedlichen Stellen im Bildfeld ausgeführt werden sollen, dürfen die entsprechenden Zeichenfolgen (bzw. Zeichen) nur relative Mark-, Bogen- oder Jump-Befehle enthalten oder es ist der Befehl **mark_serial_abs** zu verwenden.
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich). Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
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 (bzw. nach **load_program_file** des Seriennummernsatzes 0) im ausgewählten Format an der aktuellen Position. In den dabei ausgeführten indizierten Zeichenfolgen bzw. Zeichen (s.u.) werden absolute Vektor- und Bogenbefehle (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**.
Hinweise	• Der Befehl **mark_serial_abs** wirkt wie **mark_serial**; der interne Aufruf der indizierten Zeichenfolgen (bzw. Zeichen) erfolgt allerdings mit **list_call_abs** statt mit **list_call**. Enthalten die Befehlslisten der aufgerufenen indizierten Zeichenfolgen (bzw. Zeichen) keine absoluten Befehle, besteht zwischen **mark_serial_abs** und **mark_serial** kein Unterschied.
RTC4→ RTC5	Neuer Befehl.
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 Pointer auf einen nullterminierten String.
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**-Befehls wird der zu markierende Text, sofern 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 Listenspeicher geschrieben (man sollte darauf achten, dass es dabei zu keinem unerwünschten Speicherüberlauf des jeweiligen Speicherbereichs 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 dem **mark_text**-Befehl mit **select_char_set** ausgewählt werden. In der Voreinstellung wird der Zeichensatz 0 verwendet. Wenn innerhalb eines aufgerufenen indizierten Zeichens der Befehl **select_char_set** verwendet wird, werden alle nachfolgend aufgerufenen indizierten Zeichen mit diesem Zeichensatz markiert. • Wird beim Laden eines **mark_text**-Befehls 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 `RTC5_REJECTED`) und das indizierte Unterprogramm nicht gespeichert. • Absolute Vektor- und Bogenbefehle 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-, Bogen- oder Jump-Befehle enthalten oder es ist der Befehl **mark_text_abs** zu verwenden. • Der **mark_text**-Befehl sollte nicht innerhalb einer indizierten Zeichendefinition verwendet werden. Der entsprechende Text würde *nicht* markiert und das indizierte Zeichen daher nicht vollständig ausgeführt.
RTC4→ RTC5	Neuer Befehl.
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- und Bogenbefehle (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 Pointer auf einen nullterminierten String.
Hinweise	• Beim Abarbeiten der einzelnen **mark_text_abs**-Befehle werden entsprechend des gewählten Zeichensatzes die zugehörigen **mark_char_abs**-Befehle ausgeführt. • Enthalten die Befehlslisten der verwendeten indizierten Zeichen keine absoluten Befehle, besteht zwischen **mark_text_abs** und **mark_text** kein Unterschied.
RTC4→ RTC5	Neuer Befehl.
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` Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen, zulässiger Wertebereich: [0 … 4]) wird ausgewählt, 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)
	`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** ausgewählt 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.

Normaler Listenbefehl	mark_time
Hinweise	• Vor einer Uhrzeitmarkierung sollte (nach jedem Neustart) ein Abgleich zwischen RTC5 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 "Markieren von Uhrzeit, Datum und Seriennummern", Seite 170). • Die vollständige Uhrzeit wird durch mehrfachen Aufruf von **mark_time** markiert. • Der Befehl **mark_time** liest entsprechend der gespeicherten Uhrzeit und des ausgewählten Uhrzeitteils die Startadressen der zugehörigen indizierten Zeichenfolgen (bzw. – 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 Seite 88). 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 "Zeichensätze und Zeichenfolgen", Seite 87). • Für `Part` > 4 und/oder `Mode` > 3 wird **mark_time** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Absolute Vektor- und Bogenbefehle werden nach Aufruf mit **mark_time** auch absolut ausgeführt. Wenn Uhrzeitmarkierungen an unterschiedlichen Stellen im Bildfeld ausgeführt werden sollen, dürfen die entsprechenden Zeichenfolgen (bzw. Zeichen) nur relative Mark-, Bogen- oder Jump-Befehle enthalten oder es ist der Befehl **mark_time_abs** zu verwenden.
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich). Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
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 bzw. Zeichen (s.u.) werden absolute Vektor- und Bogenbefehle (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**.
Hinweise	• Der Befehl **mark_time_abs** wirkt wie **mark_time**; der interne Aufruf der indizierten Zeichenfolgen (bzw. Zeichen) erfolgt allerdings mit **list_call_abs** statt mit **list_call**. Enthalten die Befehlslisten der aufgerufenen indizierten Zeichenfolgen (bzw. Zeichen) keine absoluten Befehle, besteht zwischen **mark_time_abs** und **mark_time** kein Unterschied.
RTC4→ RTC5	Neuer Befehl.
Verweise	**mark_time**

Kontrollbefehl	mcbsp_init
Funktion	Definiert die Daten-Delays für Senden und Empfangen von Daten über die McBSP/SPI-Schnittstelle (Multi channel Buffered Serial Port, siehe auch Seite 54).
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].
Hinweise	• Bei unzulässigen Parameterangaben wird der Befehl nicht ausgeführt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Die Initialisierungswerte (nach Programmstart) sind `XDelay` = `RDelay` = 1. • Die Signale und Betriebsbedingungen der McBSP/SPI-Schnittstelle sind in Kapitel 4.4.6 "SPI / I2C-Stiftleiste", Seite 54 dargestellt.
RTC4→ RTC5	Neuer Befehl.
Verweise	**read_mcbsp**, **set_mcbsp_out**, **set_mcbsp_out_ptr**, **set_mcbsp_freq**, **mcbsp_init_spi**

Kontrollbefehl	mcbsp_init_spi
Funktion	Initialisiert die SPI-Funktionalität (Serial Peripheral Interface) an der “SPI /I²C”-Stiftleiste (statt McBSP-Funktionalität).
Aufruf	`mcbsp_init_spi ( ClockLevel, ClockDelay )`
Parameter	`ClockLevel` =0: inaktiv low. >0: inaktiv high. 32-Bit-Wert ohne Vorzeichen. `ClockDelay` =0: Taktsignal zeitgleich zu den Datenbits. >0: Taktsignal um eine halbe Periode verzögert. 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Siehe Kapitel 4.4.6 “SPI / I2C-Stiftleiste”, Seite 54. • Beim SPI-Protokoll erfolgen Dateneingabe und Datenausgabe zeitgleich mit einem gemeinsamen Taktsignal. Die RTC5 ist dabei der SPI-Master. • **mcbsp_init_spi** kann jederzeit aufgerufen werden. Angefangene und noch nicht beendete Übertragungen werden abgebrochen. • Die Taktfrequenz kann zuvor mit **set_mcbsp_freq** eingestellt werden. • Mit **mcbsp_init** kann jederzeit die McBSP-Funktionalität an der “SPI /I²C”-Stiftleiste wieder hergestellt werden. • Bei der SPI-Funktionalität ist unabhängig von der Taktfrequenz nur eine Übertragung pro 10 µs möglich (siehe z. B. **set_mcbsp_in**).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 538, OUT 538.
Verweise	**mcbsp_init**, **set_mcbsp_freq**

Kontrollbefehl	measurement_status	
Funktion	Gibt den Status einer mit **set_trigger** oder **set_trigger4** gestarteten Messung und die aktuelle Position des Messwertzählers zurück.	
Aufruf	`measurement_status( &Busy, &Pos )`	
Parameter-rückgabe	`Busy`	Status der Messung als Pointer 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 RTC5) als Pointer auf einen 32-Bit-Wert ohne Vorzeichen ($0 \leq$ `Pos` $\leq (2^{20}-1)$ oder $0 \leq$ `Pos` $\leq (2^{19}-1)$ oder `Pos` $= 2^{32}-1$, s.u.).
Hinweise	• Wenn die mit **set_trigger** oder **set_trigger4** gestartete Messung nicht mehr aktiv ist, zeigt `Pos`+1 die Anzahl der bis zum Abbruch aufgezeichneten Datenpaare an (bei **set_trigger** maximal 2^{20}, bei **set_trigger4** maximal 2^{19}). • `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 Messwertspeicherung wird mit **set_trigger** (`Period` = 0), **set_trigger4**(`Period` = 0), **stop_trigger** oder **stop_execution** zurückgesetzt (siehe Hinweis bei **set_trigger**).	
RTC4→ RTC5	Unveränderte Funktionalität.	
Versionsinfo	Letzte Änderung Version DLL 516, OUT 515.	
Verweise	**set_trigger**, **set_trigger4**	

Normaler Listenbefehl	**micro_vector_abs**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus per hartem Sprung (ohne Mikrovektorisierung) direkt von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im zweidimensionalen Bildfeld.
Aufruf	`micro_vector_abs( X, Y, LasOn, LasOff )`
Parameter	`X, Y` Absolute Koordinaten des Mikrovektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`LasOn, LasOff` LaserOn-Delay und LaserOff-Delay als 32-Bit-Werte mit Vorzeichen. *1 Bit entspricht 0,5 µs*. Zulässiger Bereich (wie bei **set_laser_delays**): $[-2^{31} \ldots +(2^{15}-1)]$. ≥ 0: Delay wird neu gesetzt (Werte über $(2^{15}-1)$ werden geclippt). < 0: Das zuvor eingestellte Delay läuft unverändert ab.
Hinweise	• Zur Verwendung von Mikrovektor-Befehlen siehe auch Seite 221. • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 221). • Der Mikrovektor wird in jedem Fall als harter Sprung ausgeführt. • Mit `LasOn` ≥ 0 bzw. `LasOff` ≥ 0 kann für jeden einzelnen Mikrovektor ein neues LaserOn- bzw. 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 gegebenenfalls durch ein vorausgehendes Scanner-Delay verzögert werden). • Bei negativen Werten (`LasOn` < 0 bzw. `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 Bogenbefehlen (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**. • Der optionale XY2-100-Konverter (siehe Seite 43) 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.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC5 die angegebenen Koordinatenwerte mit 16 und die angegebenen Delay-Werte mit 2. Die zulässigen Wertebereiche verringern sich entsprechend.
Versionsinfo	Letzte Änderung mit Version DLL 536, OUT 536: `LasOn, LasOff` < 2^{15}.
Verweise	**micro_vector_rel**, **micro_vector_abs_3d**

Normaler Listenbefehl	**micro_vector_abs_3d**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus per hartem Sprung (ohne Mikrovektorisierung) direkt von der aktuellen zur angegebenen Position (absolute Koordinatenwerte) im dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **micro_vector_abs**.
Aufruf	`micro_vector_abs_3d( X, Y, Z, LasOn, LasOff )`
Parameter	`X, Y, Z` Absolute Koordinaten des Mikrovektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`LasOn, LasOff` LaserOn-Delay und LaserOff-Delay als 32-Bit-Werte mit Vorzeichen. *1 Bit entspricht 0,5 µs*. Zulässiger Bereich (wie bei **set_laser_delays**): $[-2^{31} \ldots +(2^{15}-1)]$. ≥ 0: Delay wird neu gesetzt (Werte über $(2^{15}-1)$ werden geclippt). < 0: Das zuvor eingestellte Delay läuft unverändert ab.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **micro_vector_abs**-Befehl (siehe Hinweise dort). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert, so dass die drei Raumrichtungen bzgl. des angegebenen Wertes für die Markiergeschwindigkeit gleich behandelt werden. • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Seite 141).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC5 die angegebenen x- und y-Koordinatenwerte mit 16 und die angegebenen Delay-Werte mit 2 (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**micro_vector_abs**, **micro_vector_rel_3d**

Normaler Listenbefehl	**micro_vector_rel**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus per hartem Sprung (ohne Mikrovektorisierung) direkt von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im zweidimensionalen Bildfeld.
Aufruf	`micro_vector_rel( dX, dY, LasOn, LasOff )`
Parameter	`dX, dY` *Relativ*koordinaten des Mikrovektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`LasOn, LasOff` LaserOn-Delay und LaserOff-Delay (siehe **micro_vector_abs**).
Hinweise	• Die Koordinaten des Vektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **micro_vector_abs** (siehe Hinweise dort). • Der Mikrovektor wird in jedem Fall als harter Sprung ausgeführt.
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **micro_vector_abs**.
Versionsinfo	Letzte Änderung mit Version DLL 536, OUT 536: `LasOn, LasOff` < 2^{15}.
Verweise	**micro_vector_abs**, **micro_vector_rel_3d**

Normaler Listenbefehl	**micro_vector_rel_3d**
Funktion	Bewegt den Ausgabepunkt für den Laserfokus per hartem Sprung (ohne Mikrovektorisierung) direkt von der aktuellen zur angegebenen Position (relative Koordinatenwerte) im dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **micro_vector_rel**.
Aufruf	`micro_vector_rel_3d( dX, dY, dZ, LasOn, LasOff )`
Parameter	`dX,dY,dZ` *Relativ*koordinaten des Mikrovektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`LasOn, LasOff` LaserOn-Delay und LaserOff-Delay (siehe **micro_vector_abs_3d**).
Hinweise	• Die Koordinaten des Vektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **micro_vector_abs_3d** (siehe Hinweise dort). • Der Mikrovektor wird in jedem Fall als harter Sprung ausgeführt.
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **micro_vector_abs_3d**.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**micro_vector_abs_3d**, **micro_vector_rel**

Kontrollbefehl	move_to
Hinweise	• Der Befehl **move_to** ist im Handbuch “Installation and Operation RTC Step Motor Extension for the RTC4 and RTC5 PC interface boards” (nur in Englisch verfügbar) beschrieben. • Der Befehl **move_to** wurde für die Erweiterungskarte “RTC4 STEP MOTOR EXTENSION” (Artikelnummer 112097) eingeführt. Er wird von der Erweiterungskarte RTC5 vario*SCAN* 40 FLEX Extension (Artikelnummer 128683) *nicht* unterstützt.

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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.	
Aufruf	`para_jump_abs( X, Y, P )`	
Parameter	`X, Y`	Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`P`	Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Wenn nicht zuvor mit **set_vector_control** die vektorkontrollierte Lasersteuerung aktiviert wurde, wirkt **para_jump_abs** wie **jump_abs** (siehe Hinweise dort). Der Parameter `P` wird dann ignoriert. • Bei aktivierter vektorkontrollierter Lasersteuerung 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 ”Vektorkontrollierte Lasersteuerung”, Seite 166). • Zu `P` gibt es keinen abs-Mechanismus. `P` wird gegebenenfalls auf den zulässigen Maximalwert geclippt. • Wird **para_jump_abs** gleichzeitig mit einer positions- oder geschwindigkeitsabhängigen Lasersteuerung mit demselben Kontrollparameter verwendet, so wird der aktuelle Wert von `P` als Basis für den 100%-Wert der Lasersteuerung verwendet (siehe ”Vektorkontrollierte Lasersteuerung”, Seite 166).	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.	
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **para_jump_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **para_jump_abs**-Befehl (siehe Hinweise dort). • Weitere Hinweise siehe **jump_abs_3d**.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch. Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.
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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_jump_rel( dX, dY, P )`
Parameter	`dX, dY` *Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **para_jump_abs** (siehe Hinweise dort). • `P` wird nicht relativ behandelt.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **para_jump_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **para_jump_abs_3d** (siehe Hinweise dort). • `P` wird nicht relativ behandelt.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch. Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.
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-Lasersteuersignal für die angegebene Anzahl von externen Signalpulsen, jedoch maximal für das angegebene Zeitintervall ein und variiert simultan 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 Bereich: 0 ≤ `Period` ≤ (2^{32}–1). `Pulses` Anzahl externer Signalpulse als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: 0 ≤ `Pulses` ≤ 65535 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 wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Der Befehl ist nützlich für das Markieren separater Punkte (siehe Seite 108). • Wenn nicht zuvor mit **set_vector_control** die vektorkontrollierte Lasersteuerung aktiviert wurde, wirkt **para_laser_on_pulses_list** wie **laser_on_pulses_list**, d.h. für `Pulses` > 65535 wie **laser_on_list** (siehe Hinweise dort). Der Parameter `P` wird dann ignoriert. • Bei aktivierter vektorkontrollierter Lasersteuerung wird der mit **set_vector_control** ausgewählte Signalparameter innerhalb der Ausführdauer des Befehls (`Period` × 10 µs) linear vom zuletzt gültigen Wert zum Wert `P` variiert (siehe "Vektorkontrollierte Lasersteuerung", Seite 166). • Zu `P` gibt es keinen abs-Mechanismus. `P` wird gegebenenfalls auf den zulässigen Maximalwert geclippt. Dieser Maximalwert ist (2^{31}–1) bzw. abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`) auch ein niedrigerer Wert. • Wird **para_jump_abs** gleichzeitig mit einer positions- oder geschwindigkeitsabhängigen Lasersteuerung mit demselben Kontrollparameter verwendet, so wird der aktuelle Wert von `P` als Basis für den 100%-Wert der Lasersteuerung verwendet (siehe "Vektorkontrollierte Lasersteuerung", Seite 166). • Für `Period` = 0 bleibt der Befehl ohne Wirkung, der Befehl 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 (d.h. (`Period` – 2^{31}) × 10 µs). Hier wird der Befehl jedoch vorzeitig abgebrochen, sobald die angegebene Anzahl von externen Signalpulsen detektiert wurde.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version OUT 526, DLL 524.
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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_mark_abs( X, Y, P )`
Parameter	`X, Y` Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. `P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Wenn nicht zuvor mit **set_vector_control** die vektorkontrollierte Lasersteuerung aktiviert wurde, wirkt **para_mark_abs** wie **mark_abs** (siehe Hinweise dort). Der Parameter `P` wird dann ignoriert. • Bei aktivierter vektorkontrollierter Lasersteuerung wird parallel zur Bewegung des Laserfokus (Sky-Writing wird hier nicht berücksichtigt) der mit **set_vector_control** ausgewählte Signalparameter linear vom zuletzt gültigen Wert zum Wert `P` variiert (siehe "Vektorkontrollierte Lasersteuerung", Seite 166). • Zu `P` gibt es keinen abs-Mechanismus. `P` wird gegebenenfalls auf den zulässigen Maximalwert geclippt. • Wird **para_mark_abs** gleichzeitig mit einer positions- oder geschwindigkeitsabhängigen Lasersteuerung mit demselben Kontrollparameter verwendet, so wird der aktuelle Wert von `P` als Basis für den 100%-Wert der Lasersteuerung verwendet (siehe "Vektorkontrollierte Lasersteuerung", Seite 166).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **para_mark_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **para_mark_abs**-Befehl (siehe Hinweise dort). • Weitere Hinweise siehe **mark_abs_3d**.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch. Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.
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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`para_mark_rel( dX, dY, P )`
Parameter	`dX, dY` *Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **para_mark_abs** (siehe Hinweise dort). • `P` wird nicht relativ behandelt.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **para_mark_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **para_mark_abs_3d** (siehe Hinweise dort). • `P` wird nicht relativ behandelt.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist bei RTC5 und RTC4 identisch. Für den Parameter `P` gibt es keinen RTC4-Kompatibilitätsmodus. Es müssen die originalen RTC5-Einheiten verwendet werden.
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 zweidimensionalen Bildfeld.
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, wirkt der Befehl wie ein normaler **jump_abs**-Befehl.
Aufruf	`park_position( Mode, X, Y )`
Parameter	`Mode` 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 [–8.388.608 … 8.388.607]. 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 ggf. 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), ggf. geclippt auf den Randbereich des realen Bildfelds.
	`X, Y` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: s.o. (überlaufende Werte werden jeweils auf den Randbereich geclippt)
Hinweise	• Der Befehl 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 8.7.7 "Synchronisation von "Processing on the fly"-Anwendungen", Seite 211). Für den Rücksprung (zum Verlassen der Parkposition) sollte der Befehl **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 Sprungbefehl 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 der Befehl ein kurzer Listenbefehl und bewirkt nichts weiter. • **park_position** schaltet die "Laser active"-Lasersteuersignale nach einem LaserOff-Delay ab, aktiviert danach aber kein Scanner-Delay.

Variabler Listenbefehl	park_position
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	park_return

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 zweidimensionalen Bildfeld.	
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, ist der Befehl ein normaler Sprungbefehl. Falls sich der Laserfokus aktuell nicht in einer Parkposition befindet, ist der Befehl ein kurzer Listenbefehl und bewirkt nichts weiter (s.u.).	
Aufruf	`park_return( Mode, X, Y )`	
Parameter	`Mode`	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, Y`	Absolute Koordinaten des Sprungvektorendpunkts im virtuellen Bildfeld in Bits als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Der Befehl 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 8.7.7 "Synchronisation von "Processing on the fly"-Anwendungen", Seite 211, 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 dem Befehl angegebene Stelle. Die RTC5 berücksichtigt bei der Berechnung der neuen Position ggf. 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 (ggf. geclippt auf den Randbereich des realen Bildfelds).	

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 Seite 208). • Nach dem Befehl **park_return** wird ein Scanner-Jump-Delay aktiviert. • **park_return** schaltet die "Laser active"-Lasersteuersignale nach einem LaserOff-Delay ab.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**park_position**

Kontrollbefehl	pause_list
Funktion	Unterbricht die Ausführung einer Liste und unterdrückt die “Laser active”-Lasersteuersignale.
Aufruf	`pause_list()`
Hinweise	• **pause_list** ist synonym zum bisherigen Befehl **stop_list**. Dieser Befehl ist aber leicht zu verwechseln mit **stop_execution**. Bevorzugt sollte daher **pause_list** verwendet werden. • Wird der **pause_list**-Befehl während der Ausführung einer Liste aufgerufen, so werden die “Laser active”-Lasersteuersignale unterdrückt (die Lasersignal-Ausgänge sind dann im hochohmigen tristate-Zustand) und das Scan-System bleibt im zuletzt vorgegebenen Zustand stehen – gegebenenfalls auch mitten in einer Mikrovektorisierung. Der (mit **get_status** abrufbare) PAUSED-Status wird gesetzt, der BUSY-Status dagegen unverändert gelassen. Ein Fortfahren mit **execute_list_pos**, **release_wait** oder mit einem externen Start ist in der Folge unterbunden. Ein **stop_execution** oder externer Listen-Stopp ist dagegen möglich. **restart_list**, **stop_execution** oder ein externer Listen-Stopp heben die Unterdrückung des Listen-Starts wieder auf. • Wenn die Ausführung der Liste fortgesetzt werden soll, dann *muss* der Befehl **restart_list** verwendet werden. Nach dem erfolgten **restart_list**-Befehl setzt das Scan-System die vorgesehenen Bewegungen (des aktuellen Befehls) fort und die “Laser active”-Lasersteuersignale werden gegebenenfalls wieder freigegeben (der Laser wird mit den Standard-Laser-Parametern gestartet, also ohne eventuell vor **pause_list** festgelegte Softstart-Parameter o. ä.). Der PAUSED-Status wird dann rückgesetzt (BUSY bleibt auch hier unverändert). • Wenn beim Aufrufen des **pause_list**-Befehls aktuell keine Liste ausgeführt wird (BUSY-Status nicht gesetzt) oder eine Liste bereits mit **pause_list** oder **set_wait** angehalten wurde (PAUSED-Status gesetzt), dann wird der **pause_list**-Befehl ignoriert (**get_last_error**-Returncode `RTC5_BUSY`; der Laser ist dann bereits aus).
RTC4→ RTC5	Neuer Befehl.
Verweise	**restart_list**, **set_wait**

Kontrollbefehl	periodic_toggle
Funktion	Identisch mit **periodic_toggle_list**, aber ein Kontrollbefehl.
Aufruf	`periodic_toggle ( Port, Mask, P1, P2, Count, Start )`
Parameter	Siehe **periodic_toggle_list**.
Hinweise	• Siehe **periodic_toggle_list**. • Bei einem unzulässigen Wert für `Port` oder 0 für `P1` oder `P2` wird der Befehl nicht gesendet und ein **get_last_error**-Returncode `RTC5_PARAM_ERROR` erzeugt. • Siehe auch Kapitel 9.4 “Periodische I/O-Signale”, Seite 249.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung mit DLL 543, OUT 543: 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 Ausgabe-Port.
Aufruf	`periodic_toggle_list ( Port, Mask, P1, P2, Count, Start )`
Parameter	`Port` Ausgabeport (0–4, wie bei **set_port_default**). `Mask` Definiert die zu toggelnden Bits (der Höchstwert ist Port-abhängig). 1 = Bit wird getoggelt. 0 = Bit bleibt unverändert. `P1`, `P2` Dauer des getoggelten Signals in [10 µs] (>0, höchstens 16 Bit). `Count` Anzahl der Wiederholungen einer P1–P2-Periode. `Start` Dauer bis zum erstmaligen Toggeln in [10 µs]. Alle Parameter sind 32-Bit-Werte ohne Vorzeichen.
Hinweise	• Bei einem unzulässigen Wert für `Port` oder 0 für `P1` oder `P2` wird der Befehl durch ein **list_nop** ersetzt und ein **get_last_error**-Returncode `RTC5_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 verzögerte kurze Listenbefehle – werden noch vor dem Befehl **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 gegebenenfalls 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. • Die Parameter `P1` und `P2` werden auf den Wertebereich [0…65535] geclippt. `P1` und `P2` dürfen beide nicht 0 sein. • Der ausgewählte `Port` sollte nicht gleichzeitig für andere Ausgaben, etwa eine automatische Lasersteuerung, verwendet werden. • Bei einem **set_end_of_list**, **stop_execution** oder externem /STOP laufen die periodischen Signale weiter. • Ab Version DLL 543, OUT 543 toggelt **periodic_toggle_list** bei `Count` = 2^{32}–1 endlos. • Siehe auch Kapitel 9.4 “Periodische I/O-Signale”, Seite 249.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung mit DLL 543, OUT 543: 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()`
Hinweise	• Die aktuelle Liste wird noch komplett bis zum Erreichen des nächsten **set_end_of_list**-Befehls abgearbeitet, ehe die Ausführung beendet wird. • Wenn vorher kein **start_loop**-Befehl erteilt worden ist, hat **quit_loop** keinerlei Auswirkungen.
RTC4→ RTC5	Unverändert.
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 überwachender Scan-Kopf als 32-Bit-Wert ohne Vorzeichen. 0: Die Überwachung ist ausgeschaltet (default nach **load_program_file**). 1: Primärer Scan-Kopf wird überwacht. 2: Sekundärer Scan-Kopf wird überwacht. 3: Beide Scan-Köpfe werden überwacht. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`Mode`	Zu treffende Maßnahme als 32-Bit-Wert ohne Vorzeichen. 0: Laser wird sofort ausgeschaltet und die Liste läuft weiter. 1: Laser wird sofort ausgeschaltet und die Liste wird sofort beendet (wie **stop_execution** oder `/STOP`).
	`Data`	Zur Überwachung ausgewählter Datentyp als 32-Bit-Wert ohne Vorzeichen. 0: Sample-Werte (“Processing on the fly”- und Wobbel-korrigiert, wie **set_trigger**-Signaltypen 7–9). 1: Transformierte Ansteuerwerte (mit Scan-Kopf-spezifischen Transformationen, wie **set_trigger**-Signaltypen 25–30). 2: Korrigierte Ansteuerwerte (mit Korrekturdatei, wie **set_trigger**-Signaltypen10–15). 3: tatsächliche Ausgabewerte (mit “Offset” und “Gain”, wie **set_trigger**-Signaltypen 20–23). 4: Galvanometerscanner-Ist-Position (nur mit i*DRIVE*-Systemen, vario*SCAN* ist intern an einer X-Achse angeschlossen, wie **set_trigger**-Signaltypen 1, 2 und 3). 5: Galvanometerscanner-Ist-Position (nur mit i*DRIVE*-Systemen, vario*SCAN* ist intern an einer Y-Achse angeschlossen, wie **set_trigger**-Signaltypen 1, 2 und 4).

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_<Achse>_overflow** und **if_not_fly_<Achse>_overflow**-Befehlen 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. • Die **range_checking**-Überwachung ist auch ohne "Processing on the fly"-Anwendung wirksam. • `Data` > 5 führt zu einem **get_last_error**-Fehlercode `RTC5_PARAM_ERROR`. Der Befehl wird in diesem Fall als **list_nop** an die RTC5 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 rückzuübertragender Datentyp von Benutzern selbst eingestellt werden. Eine gleichzeitige Verwendung mit einer geschwindigkeitsabhängigen Lasersteuerung mit `Mode` = 2 (siehe **set_auto_laser_control**) ist nicht möglich. • `Data` = 4 und `Data` = 5 unterscheiden sich nur um die Achse, an die ggf. 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 ggf. unsinnige Abschaltungen. • Wurde der Laser mit `Mode` = 0 abgeschaltet und die Fehlerbedingung liegt nicht mehr vor, dann wird er erst mit dem nächsten Markierbefehl wieder eingeschaltet. Er wird *nicht* mitten in einem aktuell ablaufenden Markierbefehl eingeschaltet.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_fly_limits**, **set_fly_limits_z**

Kontrollbefehl	**read_abc_from_file**	
Funktion	Liest die ABC-Werte direkt aus der angegebenen Korrekturdatei im PC.	
Aufruf	`ErrorNo = read_abc_from_file( Name, &A, &B, &C )`	
Rückgabe	`ErrorNo`	Fehlercode als 32-Bit-Wert ohne Vorzeichen:
	`0`	Kein Fehler.
	`1`	Dateifehler (korrupt oder unvollständig).
	`2`	Memory Fehler (DLL-intern, Windows-Arbeitsspeicher).
	`3`	Datei-Open Fehler (leerer String, Datei nicht gefunden etc.).
Parameter	`Name`	Name der Korrekturdatei als Pointer auf einen nullterminierten ANSI-String.
Parameter-rückgabe	`A`, `B`, `C`	Koeffizienten (im 64-Bit-IEEE-Gleitkommaformat) der parabolischen Funktion $z_{out} = A + B/ + C/^2$, die zur Berechnung der z-Ausgabewerte verwendet wird.
Hinweise	• Der Befehl **read_abc_from_file** ist auch ohne explizite Zugriffsberechtigung auf irgendeine RTC5-Karte verfügbar. • Der Befehl **read_abc_from_file** ist nicht als Multi-Board-Befehl verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **read_abc_from_file** weder erzeugt noch verändert.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Verfügbar ab Version DLL 538, OUT 538.	
Verweise	**write_abc_to_file**	

Kontrollbefehl	read_analog_in
Funktion	Liest die analogen Eingabewerte (ANALOG IN0 und ANALOG IN1) an der optionalen ADC-Aufsteckplatine der RTC5 PCI-Karte oder an der Stiftleiste "SPI / I2C" der RTC5 PCI-Express-Karte ein und liefert sie als 12-Bit-Digitalwerte zurück.
Aufruf	`AnalogValue = read_analog_in()`
Rückgabe	32-Bit-Wert ohne Vorzeichen: Bits#0…11 ANALOG IN0-Eingabewert als 12-Bit-Digitalwert. Bit#12 = 0 (Kanalnummer). Bits#13…14 = 0. Bit#15 Old-Bit für ANALOG IN0. Bits#16…27 ANALOG IN1-Eingabewert als 12-Bit-Digitalwert. Bit#28 = 1 (Kanalnummer). Bits#29…30 = 0. Bit#31 Old-Bit für ANALOG IN1.
Hinweise	• **read_analog_in** ist nur bei RTC5 PCI-Express-Karten und RTC5 PCI-Karten mit aufgesteckter ADC-Aufsteckplatine verwendbar (siehe Kapitel 4.4.8 "Analog-Eingänge (Sonderzubehör)", Seite 59 und Kapitel 16.2.3 "McBSP/SPI-Schnittstelle und Analog-Eingänge", Seite 676). • **read_analog_in** ist nicht mit RTC5 PC/104-*Plus*-Karten und RTC5 PCIe/104-Karten verwendbar. • Die Old-Bits (Bits #15 und #31) werden durch **read_analog_in** nach dem Einlesen auf 1 gesetzt. Sie werden automatisch wieder zurückgesetzt, wenn am jeweiligen Analog-Eingang neue Daten vorliegen. • Ab Version DLL 543, OUT 543, RBF 524 können die analogen Eingabewerte (ANALOG IN0 und ANALOG IN1) mit dem Triggersignal 54 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**). Die Old-Bits werden jedoch nicht aufgezeichnet. Das Format der Daten ist ((ANALOG IN1 << 16) + ANALOG IN0).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 527, OUT 529, RBF 519.
Verweise	**set_trigger**, **set_trigger4**

Kontrollbefehl	read_encoder
Funktion	Liefert die mit **store_encoder** zwischengespeicherten Zählerstände der beiden RTC5-Encoder-Zähler zurück.
Aufruf	`read_encoder( &Encoder0_0, &Encoder1_0, &Encoder0_1, &Encoder1_1 )`
Parameter-rückgabe	`Encoder0_0`, `Encoder1_0`, `Encoder0_1`, `Encoder1_1` Zählerstände als Pointer auf 32-Bit-Werte mit Vorzeichen. Für Parameter 'Encoder*n*_*m*' bedeutet: • *n*: Nummer der Encoder-Zählers (Encoder0, Encoder1). • *m*: Nummer der Speicherposition (Parameter `Pos` von **store_encoder**).
Hinweise	• Siehe auch ""Processing on the fly" (optional)", Seite 199 und "Synchronisation durch Encoder-Signale", Seite 246. • Wird die Werkstück-Bewegung mit einem Inkremental-Encoder registriert, so wird Encoder-Zähler Encoder0 durch die Signale am Encoder-Eingang ENCODER X und Encoder-Zähler Encoder1 durch die Signale am Encoder-Eingang ENCODER Y getriggert. Wurde dagegen mit **simulate_encoder** eine Encoder-Simulation gestartet, so werden beide Encoder-Zähler 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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 520, OUT 519.
Verweise	**store_encoder**, **get_encoder**, **set_fly_x**, **set_fly_y**, **set_fly_rot**, **wait_for_encoder**

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 IN0 … DIGITAL IN15) als 32-Bit-Werte ohne Vorzeichen.
Hinweise	• Siehe Abschnitt "16 Bit-Digital-Eingang", Seite 238.
RTC4→ RTC5	Unveränderte Funktionalität.
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-Sollposition 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 4 niederwertigsten Bits ausgewertet (der IOPort-Buffer umfasst nur sechzehn Einträge). Dadurch kann in der Anwendung ein durchlaufender Zähler für den Index verwendet werden.
Parameter-rückgabe	`Value`	Im IOPort-Buffer unter `Index` abgespeicherter 16-Bit-Wert als Pointer auf einen 32-Bit-Wert ohne Vorzeichen.
	`XPos, YPos`	Im IOPort-Buffer unter `Index` abgespeicherte Galvanometerscanner-Sollposition (X- und Y-Werte) als Pointer auf 32-Bit-Werte mit Vorzeichen.
	`Time`	Im IOPort-Buffer unter `Index` abgespeicherte Zeit in Sekunden (als Pointer 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-Werte ohne Vorzeichen.	
Hinweise	• Siehe auch Hinweise unter **read_io_port_list**. • Wenn vor **read_io_port_buffer** noch kein **read_io_port_list**-Befehl aufgerufen wurde, werden die Initialisierungswerte (default: 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→ RTC5	Neuer Befehl.	
Verweise	**read_io_port_list**	

Unverzögerter kurzer Listenbefehl	**read_io_port_list**
Funktion	Schreibt den aktuellen Zustand (DIGITAL IN0 … DIGITAL IN15) des 16-Bit-Digital-Eingangs der EXTENSION 1-Stiftleiste in den internen IOPort-Buffer. Gleichzeitig werden die aktuellen XY-Sollpositionen sowie die (relative) Zeit in Sekunden abgespeichert.
Aufruf	`read_io_port_list()`
Hinweise	• Der Befehl **read_io_port_list** gibt keine Werte zurück. Diese können aber anschließend mit dem Befehl **read_io_port_buffer** aus dem IOPort-Buffer ausgelesen werden. • Der IOPort-Buffer umfasst sechzehn Einträge und ist zyklisch organisiert. Er speichert stets die jeweils letzten 16 Einträge ab und überschreibt die vorherigen ohne Warnung. Der fortlaufende Index startet nach einem Reset (Programmstart) mit Index 0 und beginnt nach 15 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 238.
RTC4→ RTC5	Neuer Befehl.
Verweise	**read_io_port_buffer**, **read_io_port**

Kontrollbefehl	read_mcbsp
Funktion	Liest den Eingabewert, der zuletzt vollständig über die McBSP/SPI-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, 2: Speicherstellen für Online-Positionierung und die **set_mcbsp_in**-Eingabe. = 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 beiden niederwertigsten Bits ausgewertet.
Rückgabe	Eingabewert als 32-Bit-Wert mit Vorzeichen.
Hinweise	• Zur McBSP/SPI-Schnittstelle (Multi channel Buffered Serial Port) und den zugehörigen Speicherstellen siehe auch Seite 54. • Die Interpretation als ein 32 Bit-Datenwort mit oder ohne Vorzeichen, als zwei 16 Bit-Datenworte mit Vorzeichen oder als zwei 15-Bit-Datenworte mit Vorzeichen obliegt den Benutzern: – Zu "Processing on the fly"-Anwendungen (`No` = 0) siehe Seite 199 und Seite 248. – Zur Online-Positionierung (`No` = 1…2) siehe Seite 187. – Zur **set_mcbsp_in**-Eingabe (`No` = 0…3) siehe Seite 203 und Seite 205. – Zur **set_multi_mcbsp_in**-Eingabe siehe Seite 526. • Nach einem Reset mit **load_program_file** werden Eingabewerte an der McBSP/SPI-Schnittstelle an die Speicherstelle 0 übertragen (default), auch dann, wenn noch keine "Processing on the fly"-Korrektur aktiviert wurde. • Das jeweils erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP/SPI-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen (siehe Seite 56).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version OUT 526, DLL 524. • Letzte Änderung (DLL 531, OUT 532): `No` = 3 und **set_mcbsp_in**-Eingabe.
Verweise	**get_mcbsp**, **get_mcbsp_list**, **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**

Kontrollbefehl	read_multi_mcbsp
Funktion	Liest den Eingabewert zurück, der zuletzt über die Mehrfach-McBSP/SPI-Übertragung typsortiert gespeichert wurde.
Aufruf	`mcbsp_value = read_multi_mcbsp( No )`
Parameter	`No` Nummer der typsortierten internen Speicherstelle als 32-Bit-Wert ohne Vorzeichen (`No` entspricht den Codierungs-Bits #0 - 2 der Mehrfach-McBSP/SPI-Eingabe). 0 = Speicherstelle für die X-Koordinate der "Processing on the fly"-Anwendung. 1 = Speicherstelle für die Y-Koordinate der "Processing on the fly"-Anwendung. 2 = Speicherstelle für die Z-Koordinate der "Processing on the fly"-Anwendung. 3 = Speicherstelle für die Laserleistung P. 4 = Speicherstelle für den Extraparameter E1. 5 = Speicherstelle für den Extraparameter E2. 6 = Speicherstelle für den Extraparameter E3. 7 = Speicherstelle für den Extraparameter E4. Es werden nur die drei niederwertigsten Bits ausgewertet.
Rückgabe	Speicherwert als 32-Bit-Wert mit Vorzeichen.
Hinweise	• Die Mehrfach-McBSP/SPI-Übertragung sollte zuvor mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktiviert und verwendet worden sein. Andernfalls werden Default-Werte oder veraltete Werte ausgelesen.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_multi_mcbsp_in**, **set_multi_mcbsp_in_list**

Kontrollbefehl	read_status		
Funktion	Liefert den Listenstatus der RTC5 zurück, siehe Kapitel 6.4.2 "Listenstatus", Seite 76.		
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 Input-Pointer 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 Input-Pointer in die "Liste 1" gesetzt wird. LOAD1 wird rückgesetzt, wenn ein **set_end_of_list**-Befehl 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 Input-Pointer 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 Input-Pointer in die "Liste 2" gesetzt wird. LOAD2 wird rückgesetzt, wenn ein **set_end_of_list**-Befehl 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**-Befehl in die "Liste 1" geschrieben wurde. READY1 wird rückgesetzt, 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**-Befehl in die "Liste 2" geschrieben wurde. READY2 wird rückgesetzt, 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 Output-Pointer 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). Rückgesetzt wird BUSY1 durch Ausführen des **set_end_of_list**-Befehls 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 Output-Pointer 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). Rückgesetzt wird BUSY2 durch Ausführen des **set_end_of_list**-Befehls 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**-Befehl erreicht wurde. USED1 wird rückgesetzt, 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**-Befehl erreicht wurde. USED2 wird rückgesetzt, wenn LOAD2 (z.B. mit **set_start_list_pos**) gesetzt wird.
	Bits #8…31		1	
Hinweise	• Bei der Interpretation der mit **read_status** rückgelesenen Statuswerte sollte immer der jeweils programmierte Lade- bzw. 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-Status einer Liste bereits rückgesetzt und ihr READY-Status gesetzt sein, wenn nach dem Laden eines **set_end_of_list**-Befehls weitere Listenbefehle in denselben Listenbereich geladen werden. – Die Statuswerte ändern sich nicht, wenn ein **set_end_of_list**-Befehl durch einen anderen Befehl überschrieben wird (READY wird nicht rückgesetzt). – Wenn während eines laufenden Ladevorgangs in derselben Liste ein **set_end_of_list**-Befehl abgearbeitet wird, gilt die Liste als bereits abgearbeitet (USED gesetzt), obwohl sie noch neu geladen wird (USED war eigentlich rückgesetzt). – Auch wenn in einem Listenbereich eine vollständig geladene Befehlsliste (inkl. **set_end_of_list**) gespeichert ist, kann der READY-Status einer Liste rückgesetzt sein, wenn der Input-Pointer – z.B. mit **set_input_pointer** – neu in die Liste gesetzt wurde. Dann kann auch der USED-Status einer Liste rückgesetzt 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-Statuswerte beider Listen (im Gegensatz zu den BUSY-Statuswerten) unverändert und sind dann nicht mehr aussagekräftig.			

Kontrollbefehl	read_status
Hinweise (Forts.)	• Wenn der Listenstatus während der Abarbeitung eines Unterprogramms im geschützten Speicherbereich (“Liste 3”) abgefragt wird, so wird der Status derjenigen Liste (“Liste 1” oder “Liste 2”) zurückgegeben, in der sich der Output-Pointer 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 und PAUSED (siehe Kapitel 6.4.3 “Listenausführungsstatus”, Seite 77) können mit **get_status** ausgelesen werden. • Zum Auslesen der Statussignale der *Scan-Köpfe* dient der Befehl **get_head_status**.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Bei der RTC5 gibt es auch einen USED-Status.
Verweise	**get_status**, **get_head_status**

Kontrollbefehl	read_user_data
Funktion	Liest die Statusinformationen aus, die aktuell vom Scan-System über das User-Datenbit rückübertragen werden.
Hinweise	• Dieser Befehl hat zur Zeit keinerlei Wirkung, ist aber für eine zukünftige Verwendung gemäß Kapitel 7.3.6 “Statusüberwachung und Diagnose”, Seite 143 und Kapitel 8.1.2 “Anpassen des Rückmeldeverhaltens”, Seite 173 reserviert.
Verweise	**send_user_data**

Kontrollbefehl	release_rtc
Funktion	Gibt die ausgewählte RTC5-Karte zur Verwendung durch andere Anwenderprogramme frei.
Aufruf	`NoOfReleasedCard = release_rtc( CardNo )`
Parameter	`CardNo` DLL-interne Nummer der RTC5-Karte als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Der Rückgabewert (32-Bit-Wert ohne Vorzeichen) ist `CardNo`, sofern das Anwenderprogramm zuvor im Besitz der Zugriffsberechtigung für diese Karte war. Sonst wird 0 zurückgegeben.
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 derselben oder von einem anderen Anwenderprogramm mit **acquire_rtc** oder **init_rtc5_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 dem Befehl **select_rtc** ausgeführt werden. • Der Befehl **release_rtc** löst keinen Reset der RTC5-Karte aus. Ein Karten-Reset wird nur durch den Befehl **load_program_file** initiiert. • Der Befehl **release_rtc** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar, bleibt dann aber immer wirkungslos (mit Rückgabewert 0). • Der Befehl **release_rtc** bleibt auch wirkungslos (mit Rückgabewert 0), wenn `CardNo` die Zahl der bei der Initialisierung gefundenen RTC5-Karten (siehe **rtc5_count_cards**) übersteigt oder wenn `CardNo` = 0 ist (reale Karten zählen ab 1). • Der Befehl **release_rtc** ist nicht als Multi-Board-Befehl verfügbar.
RTC4→ RTC5	Neuer Befehl.
Verweise	**acquire_rtc**, **init_rtc5_dll**

Kontrollbefehl	release_wait
Funktion	Setzt die Ausführung einer mit **set_wait** angehaltenen Liste fort.
Aufruf	`release_wait()`
Hinweise	• Der Befehl **release_wait** wird nur ausgeführt, wenn die RTC5 tatsächlich im Unterbrechungs-Status ist (d.h. wenn zuvor ein Unterbrechungspunkt erreicht und die Ausführung unterbrochen wurde; der PAUSED-Status ist dann gesetzt, der BUSY-Status *nicht* gesetzt). Der Befehl wird sonst ignoriert (**get_last_error**-Returncode `RTC5_BUSY`). • Durch **release_wait** wird der (mit **get_status** abrufbare) PAUSED-Status rückgesetzt und der BUSY-Status erneut gesetzt (siehe auch "Listenausführungsstatus", Seite 77). • Der Befehl **release_wait** setzt das wait_word, 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, so führt **release_wait** einen entsprechenden Home-Return aus (für die Dauer des Home-Returns ist der INTERNAL-BUSY-Status gesetzt). • Der Wait-Status kann mit **get_wait_status** abgefragt werden.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Bei der RTC5 gibt es auch einen PAUSED-Status. Dieser wird durch **set_wait** gesetzt und durch **release_wait** rückgesetzt.
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).
Hinweise	• Zur Fehlerbehandlung siehe Kapitel 6.8 "Error-Handling", Seite 98. • Der akkumulierte Fehlercode kann bitweise (für jeden Fehlertyp einzeln, z.B. Bit#5 und Bit#6 mit Code = $2^5 \mid 2^6$ oder mit `Code = RTC5_BUSY | RTC5_REJECTED`) oder komplett (für `Code` = "–1", d.h. für `Code` = 2^{32}–1) zurückgesetzt werden. Die Zuordnung zwischen den Bitnummern, Fehlertypen und Fehlerkonstanten ist beim Befehl **get_error** aufgelistet. • **reset_error** löscht nicht den Fehlercode eines anderen Anwenderprogramms, die gerade für die angegebene Karte zugriffsberechtigt ist. • Die Befehle **reset_error** und **n_reset_error** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Die kartenspezifische Fehlervariable `LastError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) wird von **reset_error** weder erzeugt noch verändert. `AccError` wird dagegen wie mit `Code` spezifiziert verändert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_error**, **get_last_error**

Kontrollbefehl	restart_list
Funktion	Gibt die "Laser active"-Lasersteuersignale wieder frei und setzt eine mit **pause_list** oder **stop_list** unterbrochene Listenausgabe fort.
Aufruf	`restart_list()`
Hinweise	• Der Befehl **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-Status als auch der PAUSED-Status gesetzt sind. Andernfalls (z.B. wenn eine Liste mit **set_wait** angehalten wurde) wird der Befehl ignoriert (**get_last_error**-Returncode `RTC5_BUSY`). • Durch **restart_list** wird der PAUSED-Status rückgesetzt. Der BUSY-Status wird unverändert gelassen.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Bei der RTC5 gibt es auch einen PAUSED-Status. Dieser wird durch **pause_list** und **stop_list** gesetzt und durch **restart_list** rückgesetzt.
Verweise	**pause_list**, **stop_list**, **release_wait**

Kontrollbefehl	rs232_config
Funktion	Konfiguriert die RS232-Schnittstelle mit der angegebenen Baud-Rate.
Aufruf	`rs232_config( BaudRate )`
Parameter	`BaudRate` Baud-Rate als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [300 … +115.200].
Hinweise	• Die Default-Einstellung ist 9600 Baud. • Die anderen Einstellungen der RS232-Schnittstelle können nicht verändert werden (Data-Bits: 8, Start-Bits: 1, Stopp-Bits: 1, Parity: None). • Siehe auch Kapitel 4.4.5 "RS232-Stiftleiste", Seite 54.
RTC4→ RTC5	Neuer Befehl.
Verweise	**rs232_write_data**, **rs232_read_data**

Kontrollbefehl	rs232_read_data
Funktion	Liest einen Wert aus dem Eingabepuffer der RS232-Schnittstelle (siehe Seite 238).
Aufruf	`RS232Data = rs232_read_data()`
Rückgabe	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). Bits #9… #15: 0. Bits #16… #23: Anzahl weiterer (noch nicht gelesener) Zeichen. Bits #24… #31: Anzahl der Pufferüberläufe.
Hinweise	• Die RS232-Schnittstelle wird intern alle 10 µs eingelesen (je ein Zeichen). Dieses Zeichen wird, sofern es neu ist, in einem 256 Zeichen großen Ringpuffer abgelegt, woraus es mit **rs232_read_data** (asynchron zum Einlesen) zum Anwenderprogramm übertragen werden kann. • Byte #0 (Bits #0…#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 (Bits #16…#23) gibt an, wieviele 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 (Bits #24…#31) gibt an, wie oft der interne Puffer bereits übergelaufen ist (entsprechend viele Zeichen sind überschrieben worden und damit endgültig verloren). Um den Überlaufzähler zurückzusetzen sind entsprechend viele **rs232_read_data**-Funktionsaufrufe auszuführen. • 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).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version RBF 513.
Verweise	**rs232_write_data**

Kontrollbefehl	rs232_write_data
Funktion	Sendet ein Datenwort (Byte) an die RS232-Schnittstelle.
Aufruf	`rs232_write_data( Data )`
Parameter	`Data` Datenwort als 32-Bit-Wert ohne Vorzeichen. Nur das niederwertigste Byte wird an die RS232-Schnittstelle übergeben.
Hinweise	• Das vollständige Senden eines etwaigen vorherigen Datenwortes wird stets abgewartet. Ein Überrennen der Schnittstelle ist nicht möglich. • Siehe auch Kapitel 9.1.6 "RS232-Schnittstelle", Seite 237.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 516.
Verweise	**rs232_write_text**, **rs232_read_data**

Kontrollbefehl	rs232_write_text
Funktion	Sendet einen Text (zeichenweise) an die RS232-Schnittstelle.
Aufruf	`rs232_write_text( pData )`
Parameter	`pData` PC-Speicheradresse des ersten Zeichens (Byte) des auszugebenden Textes als Pointer auf einen nullterminierten String.
Hinweise	• Der Befehl wird in eine entsprechende Folge von **rs232_write_data**-Befehlen zerlegt. • Während der Ausführung dieses Befehls 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**-Befehle enthält, sollte **rs232_write_text** möglichst nicht verwendet werden, um Konflikte zu vermeiden (race condition). • Siehe auch Kapitel 9.1.6 "RS232-Schnittstelle", Seite 237.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 516, OUT 515.
Verweise	**rs232_write_data**, **rs232_write_text_list**

Variabler Listenbefehl	rs232_write_text_list
Funktion	Sendet einen Text (zeichenweise) an die RS232-Schnittstelle.
Aufruf	`rs232_write_text_list( pData )`
Parameter	`pData` PC-Speicheradresse des ersten Zeichens (Byte) des auszugebenden Textes als Pointer auf einen nullterminierten String.
Hinweise	• Beim Laden eines **rs232_write_text_list**-Befehls wird der auszugebende Text, sofern 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**-Befehl in den Listenspeicher geschrieben (man sollte darauf achten, dass es dabei zu keinem unerwünschten Speicherüberlauf des jeweiligen Speicherbereichs kommt). Das Abarbeiten der einzelnen **rs232_write_text_list**-Befehle erfolgt dann ähnlich wie das Abarbeiten von **rs232_write_text**-Befehlen (der 12er-Block wird in einzelne Zeichen zerlegt und die einzelnen Zeichen der Reihe nach an die RS232-Schnittstelle gesendet). • Der Befehl nimmt je nach eingestellter Baud-Rate und Länge des Textes eine gewisse Zeit in Anspruch. • Siehe auch Kapitel 9.1.6 "RS232-Schnittstelle", Seite 237.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 516, OUT 515.
Verweise	**rs232_write_text**

Kontrollbefehl	rtc5_count_cards
Funktion	Gibt die Zahl der bei der Initialisierung gefundenen RTC5-Karten zurück.
Aufruf	`NoOfCards = rtc5_count_cards()`
Rückgabe	Zahl der RTC5-Karten als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Die Initialisierung der installierten RTC5-Karten wird für jedes Anwenderprogramm separat mit **init_rtc5_dll** aufgerufen. • Der Befehl **rtc5_count_cards** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **rtc5_count_cards** ist nicht als Multi-Board-Befehl verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **rtc5_count_cards** weder erzeugt noch verändert.
RTC4→ RTC5	Neuer Befehl. In der Funktionalität analog zur Vorgängerversion **rtc4_count_cards**.

Verzögerter kurzer Listenbefehl	**save_and_restart_timer**
Funktion	Speichert den aktuellen Wert des RTC5-Timers und setzt ihn auf Null zurück.
Aufruf	`save_and_restart_timer()`
Hinweise	• Der gespeicherte RTC5-Timer-Wert kann mit dem Befehl **get_time** ausgelesen werden. • Der Befehl ist nützlich bei der Messung der Markierungszeit eines Markierungsprozesses, siehe Kapitel 8.13 "Zeitmessungen", Seite 231. • Der RTC5-Timer zählt nur Listenbefehlstakte. Während Unterbrechungen durch **set_wait** oder **pause_list** bleibt die Zählung stehen. • Mit **get_lap_time** kann abgefragt werden, wieviele Listenbefehlstakte seit dem Zurücksetzen auf Null vergangen sind. • Falls die RTC5-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 gegebenenfalls 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). • Ab DLL 543, OUT 543, RBF 524 steht zusätzlich zum RTC5-Timer auch noch ein Zeitstempel-Zähler zur Verfügung. Er beginnt mit **load_program_file** bei 0 an zu zählen und zählt unterbrechungsfrei alle 10 μs-Takte. Siehe auch Kapitel 8.13 "Zeitmessungen", Seite 231.
RTC4→ RTC5	Unverändert.
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 Pointer auf einen nullterminierten ANSI-String.
	`Mode`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird ausgewählt, was gespeichert werden soll. Bit #0 = 1: Es werden alle indizierten Zeichen und Zeichenfolgen gespeichert. Bit #1 = 1: Es werden alle indizierten Unterprogramme gespeichert. Bit #2…31:Werden nicht berücksichtigt
Rückgabe	Anzahl der mit **save_disk** gespeicherten Listenbefehle als 32-Bit-Wert ohne Vorzeichen	
Hinweise	• Der Befehl **save_disk** kann zusammen mit dem Befehl **load_disk** u.a. zur Defragmentierung und zum nachträglichen Schützen von Unterprogrammen verwendet werden (siehe Seite 84 und Seite 85). • Durch **save_disk** werden stets 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 oft dupliziert, also auch mehrfach gespeichert. Nicht (nachträglich mit **set_char_pointer**, …) referenzierte nichtindizierte Zeichen, Zeichenfolgen und Unterprogramme werden von **save_disk** nicht berücksichtigt. • **save_disk** speichert außer den Listenbefehlen der Zeichen, Zeichenfolgen und/oder Unterprogramme auch deren Indices und Befehlsanzahlen in der binären Datei. • Die Anzahl der mit **save_disk** gespeicherten Listenbefehle kann wegen einer möglichen Mehrfachreferenzierung eines indizierten Zeichens, einer indizierten Zeichenfolge oder eines indizierten Unterprogramms oder einer erfolgten Referenzierung eines Zeichens, einer Zeichenfolge oder eines indizierten Unterprogramms im nichtgeschützten Listenbereich (“Liste 1” oder “Liste 2”) durchaus von der Anzahl der im geschützten Speicherbereich (“Liste 3”) gespeicherten Listenbefehle (= 2^{20} – `Mem1` – `Mem2` – **get_list_space**) abweichen. Die rückgegebene Anzahl sollte unbedingt vor einem anschließenden **load_disk**-Befehl mit der Größe des geschützten Speicherbereichs (= 2^{20} – `Mem1` – `Mem2`) verglichen werden. • Nicht mehr benötigte Zeichen (bzw. Zeichenfolgen oder Unterprogramme) sollten vor **save_disk** mit **load_char** (bzw. **load_text_table** oder **load_sub**) und unmittelbar nachfolgendem **list_return** entreferenziert werden (siehe Seite 85). • **save_disk** speichert alle Zeichen, Zeichenfolgen und/oder Unterprogramme jeweils ab der referenzierten Adresse bis zum nächstmöglichen **list_return**. Sprungbefehle (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.	

Kontrollbefehl	save_disk
Hinweise (Forts.)	• Für `Mode` = 0 oder `Name` = 0 wird der Befehl **save_disk** ignoriert (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`). Der Befehl wird ebenfalls ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Wenn dagegen eine Liste nur mit **set_wait** angehalten wurde (PAUSED-Status gesetzt), kann **save_disk** ausgeführt werden. • Während der **save_disk**-Befehl abgearbeitet wird, sind keine weiteren Befehle ausführbar. Externe Starts werden für die Laufzeit des Befehls unterdrückt.
RTC4→ RTC5	Neuer Befehl.
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**-Befehle aus.
Aufruf	`select_char_set( No )`
Parameter	`No` Nummer des gewünschten Zeichensatzes als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … 3].
Hinweise	• Die Auswahl bleibt so lange gültig, bis eine andere getroffen wird. • Default-Wert (nach Initialisierung) ist `No` = 0. • Für `No` > 3 wird der Befehl durch **list_nop** ersetzt. Der bisherige Zeichensatz bleibt dann gültig (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Ein Wechsel des Zeichensatzes *innerhalb* eines **mark_text**- oder **mark_text_abs**-Befehls 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→ RTC5	Neuer Befehl.
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 Bildfeldkorrektur.
Aufruf	`select_cor_table( HeadA, HeadB )`
Parameter	`HeadA` = 0: Schaltet die Signale für Scan-Kopf A (prim. Scan-Kopf-Anschluss) ab. = 1…4: Weist Korrekturtabelle Nummer `HeadA` Scan-Kopf A zu. (als 32-Bit-Wert ohne Vorzeichen)
	`HeadB` = 0: Schaltet die Signale an Scan-Kopf B (sek. Scan-Kopf-Anschluss) ab. = 1…4: Weist Korrekturtabelle Nummer `HeadB` Scan-Kopf B zu (Freischaltung erforderlich). (als 32-Bit-Wert ohne Vorzeichen)
Hinweise	• Der Befehl **select_cor_table** bzw. der Befehl **select_cor_table_list** sollte unmittelbar nach dem Laden der gewünschten Korrekturtabelle(n) mit **load_correction_file** und/oder **load_z_table** (bzw. nach einem anschließenden **load_program_file**-Befehl – siehe unten) aufgerufen werden, um die Korrekturtabelle(n) den jeweiligen Steuerungs-Ports zuzuweisen (ab Version DLL 521, OUT 521 wird **select_cor_table** automatisch von **load_correction_file** aufgerufen; siehe Hinweise Seite 138). Damit sich die Bildfeldkorrektur auch unmittelbar auf die aktuell eingestellte Galvanometerpositionen auswirkt, lösen **select_cor_table** und **select_cor_table_list** zusätzlich einen Sprung mit `jump_speed` (keinen *harten* Sprung) auf die korrigierte Galvanometerposition 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 Galvanometerposition 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 die Befehle **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** bzw. **select_cor_table_list** voll wirksam (bei älteren Versionen als DLL 521, OUT 521 möglicherweise auch Korrekturtabellen, die nach **load_program_file** geladen werden; siehe Hinweise Seite 138). Die Korrekturtabellen Nummer 3 und 4 werden durch **load_program_file** gelöscht. • Der Befehl **select_cor_table** wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten). Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). Der Listenbefehl **select_cor_table_list** kann uneingeschränkt verwendet werden. • Wenn der INTERNAL-BUSY-Status der Karte gesetzt ist, wird **select_cor_table** gegebenenfalls erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Status wieder zurückgesetzt wurde). • Für die Dauer des Sprungs auf die korrigierte Galvanometerposition setzt der Befehl **select_cor_table** den INTERNAL-BUSY-Status.

Kontrollbefehl	select_cor_table
Hinweise (Forts.)	• Bei ungültigen Werten für die Parameter `HeadA` oder `HeadB` (alle Werte > 4) und beim Wert 0 schaltet **select_cor_table** die 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 sekundären Scan-Kopf-Anschluss keine Korrekturtabelle zu. • Die Standardeinstellung, die mit dem Befehl **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 freigegeben ist, bleiben die Signale des zweiten Scan-Kopfs permanent ausgeschaltet, es können jedoch trotzdem mehrere Korrekturtabelle geladen und jederzeit auch für den primären Scan-Kopf verwendet werden. Dadurch kann man auch bei einem Scan-Kopf schnell zwischen mehreren Korrekturtabellen, etwa zwischen einer Datei für einen Positionierlaser und einer zweiten für den Hauptlaser mit unterschiedlicher Wellenlänge, hin- und herwechseln (verwenden Sie dazu `select_cor_table(1,0)` und `select_cor_table(2,0)`, siehe auch Kapitel 8.5 "Verwendung mehrerer verschiedener Korrekturtabellen", Seite 192). • 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 3D-Option freigeschaltet sein. Es können dann mehrere (2D- oder 3D-) Korrekturtabellen geladen werden. Ohne freigeschaltete Option "Zweite-Scan-Kopf-Steuerung" müssen die XY-Achsen an den primären Scan-Kopf-Anschluss und die Z-Achse an den X- oder Y-Kanal des sekundären Scan-Kopf-Anschlusses angeschlossen werden. Wenn dem primären Scan-Kopf-Anschluss eine 3D-Korrekturtabelle zugewiesen wird, werden am primären Scan-Kopf-Anschluss die korrigierten Signale für einen XY-Scan-Kopf und über beide Kanäle des sekundären Scan-Kopf-Anschlusses die korrigierten Signale für die Z-Achse ausgegeben. Wenn mehrere RTC5-Karten mit freigeschalteter 3D-Option in einen PC eingebaut werden, dann können auch entsprechend viele 3-Achsen-Systeme gleichzeitig angesteuert werden. • Bei freigeschalteter 3D-Option 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"Scan-Kopf-Anschlüsse und Übertragungsprotokoll", Seite 42 und "2D- und 3D-Korrekturdateien", Seite 137). 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-Tabelle 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**).

Kontrollbefehl	select_cor_table
RTC4→ RTC5	Unveränderte Funktionalität. Ausnahme: Im Speicher der RTC5 können jetzt auch bis zu vier Korrekturtabellen gespeichert werden, mit freigeschalteter 3D-Option auch bis zu vier 3D-Korrekturtabellen. Es können aber *nicht* zwei *3D*-Korrekturtabellen gleichzeitig den Scan-Kopf-Steuerungs-Ports zugewiesen werden.
Versionsinfo	Letzte Änderung (Version DLL 533, OUT 534): Der Wertebereich der Parameter wurde auf [0…4] erhöht (mit **load_correction_file** können bis zu 4 Korrekturdateien geladen werden).
Verweise	**select_cor_table_list**, **load_correction_file**, **load_program_file**

Variabler Listenbefehl	select_cor_table_list
Funktion	Ähnlich wie **select_cor_table**, aber ein Listenbefehl.
Aufruf	`select_cor_table_list( HeadA, HeadB )`
Parameter	`HeadA` = 0: Schaltet die Signale für Scan-Kopf A (prim. Scan-Kopf-Anschluss) ab. = 1…4: Weist Korrekturtabelle Nummer `HeadA` Scan-Kopf A zu. (als 32-Bit-Wert ohne Vorzeichen)
	`HeadB` = 0: schaltet die Signale an Scan-Kopf B (sek. Scan-Kopf-Anschluss) ab. = 1…4: weist Korrekturtabelle Nummer `HeadB` Scan-Kopf B zu (Freischaltung erforderlich). (als 32-Bit-Wert ohne Vorzeichen)
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 Galvanometerposition um einige Takte verzögert ausgeführt, abhängig vom Inhalt der zugewiesenen Korrekturtabelle an der momentanen Galvanometerposition (z. B. (0|0) nach **load_program_file**), mindestens jedoch um 10 µs, auch wenn die Korrektur-tabelle keine Korrektur vorsieht.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung (Version DLL 533, OUT 534): Der Wertebereich der Parameter wurde auf [0…4] erhöht (mit **load_correction_file** können bis zu 4 Korrekturdateien geladen werden).

Kontrollbefehl	select_rtc
Funktion	Definiert die für ein Anwenderprogramm aktive RTC5-Karte in einem Multi-Board-System (siehe Kapitel 6.6, Seite 92).
Aufruf	`NoOfSelectedCard = select_rtc( CardNo )`
Parameter	`CardNo` (DLL-interne) Nummer der RTC5-Karte als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	Der Rückgabewert (32-Bit-Wert ohne Vorzeichen) 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 RTC5-Karten übersteigt 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 `RTC5_ACCESS_DENIED` und gegebenenfalls `RTC5_VERSION_MISMATCH`).
Hinweise	• Die Aktivierung einer Karte für ein Anwenderprogramm wird auch bereits bei der Initialisierung der DLL für dieses Anwenderprogramm ausgeführt (siehe **init_rtc5_dll**). Der Befehl **select_rtc** ermöglicht aber zusätzlich jederzeit einen Wechsel der aktiven Karte. Alle dem Befehl **select_rtc** nachfolgenden Befehle (außer Multi-Board-Befehle) des Anwenderprogramms werden an die jeweils aktive Karte weitergeleitet. • Falls die angegebene Karte durch eine anderes Anwenderprogramm reserviert ist, bleibt **select_rtc** wirkungslos (Rückgabewert 0, **get_last_error**-Returncode `RTC5_ACCESS_DENIED`). • Falls für die angegebene Karte keine Zugriffsberechtigung vorliegt, diese aber auch nicht durch ein anderes Anwenderprogramm belegt ist, so wird die Karte nicht nur aktiviert, sondern auch automatisch (wie mit **acquire_rtc**) – bei positiver Versionsprüfung (s. u.) – für dieses Anwenderprogramm reserviert und damit für andere Anwenderprogramme gesperrt (Rückgabewert: `CardNo`). 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 `RTC5_ACCESS_DENIED`|`RTC5_VERSION_MISMATCH`). • Falls die angegebene Karte bereits die aktive Karte für dieses Anwenderprogramm ist, bleibt **select_rtc** wirkungslos (Rückgabewert: `CardNo`). • Der Befehl **select_rtc** bleibt auch wirkungslos, wenn `CardNo` die Zahl der bei der Initialisierung gefundenen RTC5-Karten (siehe **rtc5_count_cards**) übersteigt 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. • Der Befehl **select_rtc** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **select_rtc** ist nicht als Multi-Board-Befehl verfügbar.
RTC4→ RTC5	Bei der RTC4 führt dieser Befehl nur die Aktivierung der angegebenen Karte (unbedingt) aus. Bei der RTC5 liefert der Befehl zusätzlich einen Rückgabewert und erteilt bei fehlender Zugriffsberechtigung für die angegebene Karte u. U. zusätzlich (wie mit dem Befehl **acquire_rtc**) die Zugriffsberechtigung oder bleibt wirkungslos.

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.
Hinweise	• Nach Initialisierung mit **load_program_file** ist Seriennummernsatz 0 ausgewählt. • **select_serial_set** legt *nicht* den Seriennummernsatz für die Seriennummernmarkierung fest. Dafür ist der Listenbefehl **select_serial_set_list** vorgesehen. • Zur Verwendung des Befehls siehe Kapitel 7.5.2 "Markieren von Seriennummern", Seite 170.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
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.
Hinweise	• Nach Initialisierung mit **load_program_file** ist Seriennummernsatz 0 ausgewählt. • Zur Verwendung des Befehls siehe Kapitel 7.5.2 "Markieren von Seriennummern", Seite 170.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**select_serial_set**, **set_serial_step_list**, **get_list_serial**, **mark_serial**, **mark_serial_abs**

Kontrollbefehl	send_user_data
Funktion	Sendet Datenwerte über das User-Datenbit zum ausgewählten Scan-System.
Hinweise	• Dieser Befehl hat zur Zeit keinerlei Wirkung, ist aber für eine zukünftige Verwendung gemäß Kapitel 7.3.6 "Statusüberwachung und Diagnose", Seite 143 und Kapitel 8.1.2 "Anpassen des Rückmeldeverhaltens", Seite 173 reserviert.
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 183).	
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 *primären* Scan-Kopf-Anschluss. = 2: Die Definition wirkt nur auf den *sekundären* Scan-Kopf-Anschluss (Freischaltung erforderlich). = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`Angle`	Drehwinkel in ° im 64-Bit-IEEE-Gleitkommaformat; positive Werte entsprechen einer Drehung um den Mittelpunkt des Bildfeldes im Gegenuhrzeigersinn.
	`at_once`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird. = 0: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten Listenbefehls berechnet und auf die dann aktuelle Position angewandt. = 1: Die neue Gesamttransformation wird sofort berechnet und auf die aktuelle Position angewandt (oder vor dem nächsten Listenbefehl, sofern gerade eine Liste BUSY oder die Karte INTERNAL-BUSY ist). Die "Laser active"-Lasersteuersignale werden gegebenenfalls zuvor ausgeschaltet. = 2: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten **jump_abs**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls berechnet und auf die dann aktuelle Position angewandt. = 3: Wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: Wie `at_once` = 2
Hinweise	• Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 183.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Letzte Änderung Version DLL 525, OUT 527: Verhalten für `at_once` = 3.	
Verweise	**set_angle_list**, **set_offset**, **set_matrix**, **set_scale**	

Variabler Listenbefehl	**set_angle_list**
Funktion	Identisch mit **set_angle**, aber ein Listenbefehl.
Aufruf	`set_angle_list( HeadNo, Angle, at_once )`
Parameter	`HeadNo` Siehe **set_angle**.
	`Angle` Siehe **set_angle**.
	`at_once` Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird. = 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 “Laser active”-Lasersteuersignale 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**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewandt. = 3: Wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: Wie `at_once` = 2.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 525, OUT 527: Verhalten für `at_once` = 3.
Verweise	**set_angle**

Kontrollbefehl	set_auto_laser_control	
Funktion	Initialisiert bzw. deaktiviert die positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung.	
Aufruf	`ErrorCode = set_auto_laser_control( Ctrl, Value, Mode, MinValue, MaxValue )`	
Parameter	`Ctrl`	Kontrollparameter (32-Bit-Wert ohne Vorzeichen) zur Initialisierung bzw. Deaktivierung der positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängigen Lasersteuerung. = 1…6: Definiert, welcher Signalparameter durch die positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung automatisch 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 Lasersignale LASER1 und LASER2. = 5: Ausgabeperiode (`HalfPeriod`) der Lasersignale LASER1 und LASER2. = 6: Ausgabewert am 16-Bit-Digital-Ausgang. = 0 oder > 6: deaktiviert die positions-, geschwindigkeits- und encodergeschwindigkeitsabhängige Lasersteuerung (für `Ctrl` > 6: **get_last_error**-Returncode `RTC5_PARAM_ERROR`).
	`Value`	Legt den 100%-Wert für den über `Ctrl` ausgewählten Parameter fest (32-Bit-Wert ohne Vorzeichen). Zulässige Werte: Für `Ctrl` = 1/2: 12-Bit-Werte [0 … 4095], 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 … (2^{16}–1)], höherwertige Bits werden ignoriert.
	`Mode`	Kontrollparameter (32-Bit-Wert ohne Vorzeichen) zur Initialisierung bzw. Deaktivierung der geschwindigkeits- oder encodergeschwindigkeitsabhängigen (nicht aber der positionsabhängigen) Lasersteuerung. Für die geschwindigkeitsabhängige Lasersteuerung (Mode =1…5) kann ausgewählt werden, welche Größe bei der Berechnung der geschwindigkeitsabhängigen Korrektur zu Grunde gelegt wird. =1: Sollgeschwindigkeit (bei allen Scan-Systemen). =2: Rückgelesene aktuelle Geschwindigkeit (nur bei i*DRIVE*-Scan-Systemen: intelli*SCAN*, intelli*SCAN*$_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario*SCAN*$_{de}$). =3: Reserviert. =4: Reserviert. =5: Encodergeschwindigkeit (Zählerpulsrate). =6: Kombinierte Geschwindigkeit aus 2 und 5. = 0 oder > 6: Deaktiviert die geschwindigkeitsabhängige und encodergeschwindigkeitsabhängige Lasersteuerung (für `Mode` > 6: **get_last_error**-Returncode `RTC5_PARAM_ERROR`).

Kontrollbefehl	set_auto_laser_control	
	`MinValue`	Legt die *untere* Bereichsgrenze für den über `Ctrl` ausgewählten Parameter fest, der durch die automatische Korrektur nicht unterschritten werden kann (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 (32-Bit-Wert ohne Vorzeichen). Zulässige Werte: siehe `Value`.
Rückgabe	`ErrorCode` als 32-Bit-Wert ohne Vorzeichen: 0 Kein Fehler. 1 Kein primärer Scan-Kopf aktiv. 2 Kein i*DRIVE*-Scan-System (intelli*SCAN*, intelli*SCAN*$_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario*SCAN*$_{de}$) aktiv. 3 `Ctrl` unzulässig. 4 `Mode` unzulässig. 5 Zugriff verweigert (Karte gesperrt, **get_last_error**-Returncode `RTC5_ACCESS_DENIED`).	
Hinweise	• Zur Verwendung des Befehls siehe Kapitel 7.4.9 "Automatische Lasersteuerung", Seite 159. • `MinValue` und `MaxValue` werden automatisch vertauscht, sofern `MinValue` > `MaxValue` ist und gegebenenfalls `MinValue` auf ≤ `Value` sowie `MaxValue` auf ≥ `Value` geclippt. • Die Steuerungsdaten der positions- und/oder geschwindigkeitsabhängigen Lasersteuerung werden ausschließlich aus den Daten des Scan-Systems am primären Scan-Kopf-Anschluss abgeleitet und dann auch für den sekundären Scan-Kopf-Anschluss verwendet (sofern dieser freigeschaltet ist und dort ein Scan-System angeschlossen ist). Dem primären Scan-Kopf-Anschluss muss bereits zum Zeitpunkt der Befehlserteilung eine Korrekturtabelle zugewiesen sein, sonst wird Fehlercode 1 zurückgegeben und `Ctrl` = 0 gesetzt. Wenn nur der sekundäre Scan-Kopf-Anschluss verwendet wird und/oder das Scan-System am primären Scan-Kopf-Anschluss abgeschaltet ist, funktioniert die geschwindigkeitsabhängige Lasersteuerung nicht. • Für `Mode` = 2 muss am primären Scan-Kopf-Anschluss ein funktionsfähiges i*DRIVE*-Scan-System angeschlossen sein. Ist dies nicht der Fall, wird Fehlercode 2 zurückgegeben und `Ctrl` = 0 gesetzt. Wenn das i*DRIVE*-Scan-System reagiert hat, wird der rückzulesende Datentyp für den primären Scan-Kopf-Anschluss und beide Achsen per **control_command** (`Data` = 0506_H) auf Ist-Geschwindigkeit gestellt. Erst danach wird der Befehl zur RTC5 geschickt, wobei die anderen Parameter übernommen werden. In der Folge ist der **control_command**-Befehl für den primären Scan-Kopf-Anschluss gesperrt, solange diese Mode-Auswahl besteht. Für `Mode` = 0 oder 1 ist der **control_command**-Befehl uneingeschränkt verfügbar. • Die encodergeschwindigkeitsabhängige Lasersteuerung (`Mode` = 5) funktioniert auch dann, wenn zur Laufzeit kein Scan-Kopf an den Scan-Kopf-Anschlüssen angeschlossen ist. Auch die "Processing on the fly"-Option muss hierfür nicht freigeschaltet sein.	

Kontrollbefehl	set_auto_laser_control
Hinweise (Forts.)	• `Mode` = 6 ist grundsätzlich wie `Mode` = 2. Jedoch werden die Encoder-Geschwindigkeiten mit den Skalierungsfaktoren aus **set_fly_x** und **set_fly_y** bzw. **set_fly_2d** in Galvanometerscanner-Einheiten (Bits/ms) umgerechnet und zu den aktuellen Galvanometerscanner-Geschwindigkeiten hinzuaddiert. Die Option "Processing on the fly" muss dazu freigeschaltet sein (i. Ggs. zu `Mode` = 5). Als Referenzgeschwindigkeit wird wie bei `Mode` = 2 die aktuelle Markiergeschwindigkeit verwendet. Der für `Mode` = 5 verwendete Befehl **set_encoder_speed** bleibt im `Mode` = 6 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 `Mode` = 6 wie `Mode` = 2. **set_fly_rot**, **set_fly_rot_pos**, **set_fly_x_pos** und **set_fly_y_pos** können nicht mit der Galvanometerscanner-Geschwindigkeit kombiniert werden. • Für einen unzulässigen Wert von `Ctrl` oder `Mode` wird **set_auto_laser_control** nicht ausgeführt (Rückgabewert 3 bzw. 4, **get_last_error**-Returncode: `RTC5_PARAM_ERROR`). • Die automatische Lasersteuerung sollte nicht zusammen mit der variablen Laserleistung aus **set_multi_mcbsp_in** verwendet werden.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden für `Ctrl` = 1/2 die Werte für `Value`, `MinValue` und `MaxValue` mit 4 multipliziert. Für `Ctrl` = 4/5 müssen im RTC4-Kompatibilitätsmodus die Werte für `Value`, `MinValue` und `MaxValue` in 1/8 µs angegeben werden. Sie werden dann intern auf 1/64 µs-Einheiten umgerechnet (d.h. mit 8 multipliziert). Die zulässigen Wertebereiche verringern sich entsprechend.
Versionsinfo	Letzte Änderung Version DLL 540, OUT 540: `Mode` = 6.
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**, **set_fly_rot**, **set_fly_rot_pos**, **set_fly_x_pos**, **set_fly_y_pos**, **set_multi_mcbsp_in**

Kontrollbefehl	set_auto_laser_params	
Funktion	Legt den zu steuernden Signalparameter der automatischen Lasersteuerung, den 100%-Wert und seine ihm zugeordneten Grenzwerte fest.	
Aufruf	`set_auto_laser_params( Ctrl, Value, MinValue, MaxValue)`	
Parameter	`Ctrl`	Auswahl des zu steuernden Signalparameters (siehe **set_auto_laser_control**). Zulässiger Wertebereich: [1 … 6].
	`Value`	100%-Wert (siehe **set_auto_laser_control**).
	`MinValue`	*Untere* Bereichsgrenze (siehe **set_auto_laser_control**).
	`MaxValue`	*Obere* Bereichsgrenze (siehe **set_auto_laser_control**).
Hinweise	• Zur Verwendung des Befehls siehe "Automatische Lasersteuerung", Seite 159. • Die Bedeutung der Parameter ist identisch zum Befehl **set_auto_laser_control** (siehe auch Hinweise dort). Jedoch kann die automatische Lasersteuerung mit **set_auto_laser_params** weder aus- noch eingeschaltet werden. • Für einen unzulässigen Wert von `Ctrl` (0 oder >6) wird **set_auto_laser_params** nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`).	
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_auto_laser_control**.	
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.	
Verweise	**set_auto_laser_control**	

Verzögerter kurzer Listenbefehl	set_auto_laser_params_list	
Funktion	Identisch mit **set_auto_laser_params**, aber ein Listenbefehl.	
Aufruf	`set_auto_laser_params_list( Ctrl, Value, MinValue, MaxValue)`	
Parameter	`Ctrl`	Auswahl des zu steuernden Signalparameters (siehe **set_auto_laser_control**). Zulässiger Wertebereich: [1 … 6].
	`Value`	100%-Wert (siehe **set_auto_laser_control**).
	`MinValue`	*untere* Bereichsgrenze (siehe **set_auto_laser_control**).
	`MaxValue`	*obere* Bereichsgrenze (siehe **set_auto_laser_control**).
Hinweise	• Für einen unzulässigen Wert von `Ctrl` (0 oder >6) wird **set_auto_laser_params_list** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).	
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_auto_laser_control**.	
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.	
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 beim Befehl **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^{20}–1)].
Hinweise	• Für `Char` > 1023 und/oder `Pos` > (2^{20}–1) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode `RTC5_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 werden und/oder über den Index aufgerufen werden. • Mit **set_char_pointer** kann aber auch ein indiziertes Unterprogramm oder Zeichen (bzw. Zeichenfolge) erneut referenziert werden, so dass es auch über einen zweiten Index aufgerufen werden kann. Zur Indexverwaltung sollte aber bevorzugt der Befehl **copy_dst_src** verwendet werden. • Die Anfangsadressen von Befehlslisten, die mit **set_char_pointer** referenziert werden sollen, können – *vor* dem Abspeichern der Befehlslisten – mit dem Befehl **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 stets die Anfangsadresse eines tatsächlichen Unterprogramms sein, welches mit **list_return** abgeschlossen worden ist und kein **set_end_of_list** enthält.	
RTC4→ RTC5	Neuer Befehl.	
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^{20}–1)].
Hinweise	• Der Befehl ist synonym zum Befehl **set_text_table_pointer**.
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich). Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
Verweise	**set_text_table_pointer**

Kontrollbefehl	set_control_mode		
Funktion	Aktiviert bzw. deaktiviert die externe Start-Eingabe (für externe Listen-Starts) und setzt den Zähler für externe Listen-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 externen Listen-Stopp (/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 Auswirkungen
	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 Listen-Starts relativ zum auslösenden Eingangssignal bzw. **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl (siehe "Externer Listen-Start", Seite 240) wird deaktiviert.
		= 0:	kein Effekt. Um eine Streckenverzögerung (z.B. für "Processing on the fly"-Anwendungen) zu konfigurieren und aktivieren, kann man die Befehle **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** verwenden.
	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**.
	Bits 5…8		Reserviert.
	Bit #9	= 1:	Encoder-Resets der beiden internen Encoder-Zähler (die durch die "Processing on the fly"-Befehle **set_fly_x**, **set_fly_y** oder **set_fly_rot** initiiert werden) erfolgen mit dem nachfolgenden Start-Trigger (d.h. mit einem nachfolgenden externen Startsignal oder **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl, gegebenenfalls verzögert um eine mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** konfigurierte Streckenverzögerung, siehe auch Bit#2).
		= 0:	Encoder-Resets erfolgen unmittelbar mit dem jeweiligen "Processing on the fly"-Befehl.

Kontrollbefehl	set_control_mode		
	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 Listen-Start gezählt. Der Abstand aufeinander folgender externer Listen-Starts (in Encoder-Pulsen) wird dadurch konstant (siehe auch Seite 242). Bei **stop_execution** oder einem externen Stoppsignal wird Bit #10 auf "0" zurückgesetzt. Mit Firmware-Versionen 506 oder kleiner (siehe **get_rtc_version**) ist dieses Bit wirkungslos.
		= 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 Listen-Start ab dem Anforderungs-Zeitpunkt (**simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl oder externes Startsignal) gezählt. Der Abstand aufeinander folgender externer Listen-Starts (in Encoder-Pulsen) kann dadurch variieren. Standard für Firmware-Versionen 506 oder kleiner (siehe **get_rtc_version**).
	Bits #11-31		Reserviert.
Hinweise	• Wenn die Ausführung mit dem Befehl **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 ggf. 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 Listenstart. Mit Bit #9 = 1 wird dieser 10 µs-Jitter ausgeschaltet, weil dann der Encoder-Reset exakt mit dem Startsignal erfolgt. • Siehe auch "Externer Listen-Stopp", Seite 239 und "Externer Listen-Start", Seite 240.		
RTC4→ RTC5	Anders als bei der RTC4 kann Bit#8 nicht dafür verwendet werden, eine Sperrung oder Entsperrung für die oberen 8 Bits des 16 Bit-Digital-Ausgangs (für I/O-Befehle) vorzunehmen. Die RTC5 reserviert (sperrt) diese Bits für die vario*SCAN FLEX*-Ansteuerung automatisch durch **move_to**. Eine explizite Entsperrung kann nicht vorgenommen werden (dies geschieht nur automatisch durch **load_program_file**). Ansonsten unveränderte Funktionalität für bereits bei der RTC4 verwendete Bits.		
Versionsinfo	Letzte Änderung (DLL 543, OUT 543): Bit #4.		
Verweise	**set_extstartpos**, **get_counts**, **set_max_counts**, **get_startstop_info**, **move_to**		

Normaler Listenbefehl	set_control_mode_list
Funktion	Ähnlich wie **set_control_mode**, aber ein Listenbefehl.
Aufruf	`set_control_mode_list( Mode )`
Parameter	`Mode` (als 32-Bit-Wert ohne Vorzeichen).
Hinweise	• Der Zähler für externe Listen-Starts wird durch diesen Befehl *nicht* zurückgesetzt.
RTC4→ RTC5	Unveränderte Funktionalität für bereits bei der RTC4 verwendete Bits.
Versionsinfo	Letzte Änderung (DLL 543, OUT 543): Bit #4.
Verweise	**set_control_mode**

Kontrollbefehl	set_default_pixel
Funktion	Legt den Pulslängen-Default-Wert für das den Pixelausgabemodus abschließende Default-Pixel fest.
Aufruf	`set_default_pixel( PulseLength )`
Parameter	`PulseLength` Länge des Default-Pixels (als 32-Bit-Wert ohne Vorzeichen). *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)]
Hinweise	• Im Pixelausgabemodus wird am Ende der Bildzeile (zu Beginn des Default-Pixels) die Pixelpulslänge auf den spezifizierten Default-Wert gesetzt (siehe Seite 219). • Bei der Initialisierung der RTC5 (mit **load_program_file**) wird der Pulslängen-Default-Wert auf 0 gesetzt.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_port_default**

Unverzögerter kurzer Listenbefehl	set_default_pixel_list
Funktion	Legt den Pulslängen-Default-Wert für das den Pixelausgabemodus abschließende Default-Pixel fest.
Aufruf	`set_default_pixel_list( PulseLength )`
Parameter	`PulseLength` Länge des Default-Pixels (als 32-Bit-Wert ohne Vorzeichen). *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
Hinweise	• Identisch mit **set_default_pixel**, aber ein Listenbefehl.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_default_pixel**, **set_port_default**

Kontrollbefehl	**set_defocus**
Funktion	Legt eine Fokusverschiebung für alle nachfolgenden 3D-Vektorausgaben fest.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), hat der Befehl 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: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randwert geclippt.
Hinweise	• Eine Fokusverschiebung bewirkt eine Defokussierung des Laserfokus relativ zur Arbeitsebene. Es können positive und negative Werte angegeben werden. Ein positiver Wert bewirkt eine Vergrößerung der Fokuslänge der Z-Achse (vario*SCAN*) und verschiebt beispielsweise die Fokuslage für den Ansteuerwert `(0|0|0)` um ca. d = `Shift` / K_z in die Ebene z = –d (die Fokusverschiebung läuft der z-Koordinatenachse und auch einer mit **set_offset_xyz** festgelegten Verschiebung der Arbeitsebene entgegen). Dabei ist K_z der Kalibrierungsfaktor für die z-Richtung (siehe auch "3D-Markierbefehle", Seite 194). • 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. Der Benutzer sollte dafür Sorge tragen, dass dies vermieden wird. • Wird der Befehl während einer Vektorausgabe aufgerufen, wird er erst vor dem nächsten Listenbefehl ausgeführt. Um harte Sprünge zu vermeiden, wird ein Sprung mit Sprunggeschwindigkeit auf die geänderte z-Koordinate ausgeführt. Dies kann je nach eingestellter Sprunggeschwindigkeit und Fokusverschiebung einige Takte dauern. Wenn gerade *keine* Liste BUSY ist, wird der Sprung sofort ausgeführt, wodurch beim nächsten Listen-Start keine Totzeiten entstehen. • Wenn der INTERNAL-BUSY-Status der Karte gesetzt ist, wird **set_defocus** gegebenenfalls erst verzögert ausgeführt (nachdem der INTERNAL-BUSY-Status wieder zurückgesetzt wurde). • Für die Dauer des Sprungs auf die geänderte z-Koordinate setzt der Befehl **set_defocus** den INTERNAL-BUSY-Status. • Nach Aktivierung der vektorkontrollierten Lasersteuerung mit **set_vector_control**(`Ctrl` = 7) wird die Fokusverschiebung gegebenenfalls auch bei parametrisierten Mark- oder Jump-Befehlen verändert.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, außer Vermeidung harter Sprünge.
Versionsinfo	Letzte Änderung Version OUT 515.
Verweise	**set_defocus_list**, **set_offset_xyz**

Variabler Listenbefehl	**set_defocus_list**
Funktion	Ähnlich wie **set_defocus**, aber ein Listenbefehl.
Einschränkung	Siehe **set_defocus**.
Aufruf	`set_defocus_list( Shift )`
Parameter	`Shift` Siehe **set_defocus**.
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-Position um einige Takte verzögert ausgeführt, abhängig von der Größe der angegebenen Fokusverschiebung, mindestens jedoch um 10 µs, auch für `Shift`= 0.
RTC4→ RTC5	Siehe **set_defocus**.

Kontrollbefehl	set_delay_mode		
Funktion	Schaltet den variablen Polygon-Delay-Modus und den variablen Jump-Delay-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	Alle Parameter werden als 32-Bit-Werte ohne Vorzeichen übergeben.		
	Parameter	Zulässige Werte	Beschreibung
	`VarPoly`	> 0	Aktiviert den variablen Polygon-Delay-Modus (siehe Seite 118).
		= 0	Deaktiviert den variablen Polygon-Delay-Modus (Standardeinstellung).
	`DirectMove3D`		Dieser Parameter beeinflusst nur 3D-Anwendungen.
		> 0	Der z-Wert wird während eines Sprungs direkt (linear) zu seinem Endwert geführt.
		= 0	Der z-Wert wird schrittweise so variiert, dass der Fokus während des ganzen Sprungs in einer Ebene bleibt (wie bei einem Mark-Befehl).
	`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 Eckwert muss kleiner sein als das Zweifache des eingestellten Wertes für das Polygon-Delay, sonst bleibt er ohne Wirkung (siehe auch Abb. 36). Hinweis: Um diese Anwendung zu deaktivieren, kann man den Eckwert auf (2^{32}–1) (Standardwert) setzen.
	`MinJumpDelay`	0 … (2^{32}–1). *1 Bit entspricht 10 µs.*	Minimales Jump-Delay, das für Sprünge kleiner als JumpLengthLimit auch für einen Sprungvektor der Länge Null nicht unterschritten werden kann (siehe Abb. 33). 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.
	`JumpLengthLimit`	0 … (2^{32}–1)	Sprunglängengrenze in *Bits*. Wenn der Sprungvektor *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 berechnet und eingefügt (siehe Abb. 33). Will man den variablen Jump-Delay-Modus deaktivieren, so setzt man die Sprunglängengrenze `JumpLengthLimit` auf 0.

Kontrollbefehl	set_delay_mode
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhte Wertebereiche). Im RTC4-Kompatibilitätsmodus wird der angegebene Wert für `JumpLengthLimit` von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [0 … (2^{28}–1)]).
Verweise	**set_scanner_delays**, **load_varpolydelay**, **set_delay_mode_list**

Multipler Listenbefehl	set_delay_mode_list
Funktion	Identisch mit **set_delay_mode**, aber ein Listenbefehl.
Aufruf	`set_delay_mode_list( VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit )`
Parameter	Siehe **set_delay_mode**.
Hinweise	• Siehe **set_delay_mode**. • Der Befehl benötigt zwei Listenspeicherplätze. Sie werden als zwei unverzögerte kurze Listenbefehle ausgeführt. • Ein evtl. noch anliegender verzögerter kurzer Listenbefehl wird noch davor ausgeführt.
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_delay_mode**
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**set_delay_mode**

Kontrollbefehl	set_dsp_mode
Funktion	Setzt einen DSP-Mode für die Kurzbefehlszählung.
Aufruf	`initial_dsp_mode = set_dsp_mode( Mode )`
Parameter	`Mode` Gewünschter DSP-Mode als 32-Bit-Wert ohne Vorzeichen.
Rückgabe	DSP-Mode, der beim Initialisieren der Karte eingestellt wurde (als 32-Bit-Wert ohne Vorzeichen).
Hinweise	• Dieser Befehl wird nur benötigt, wenn mehrere RTC5-Karten unterschiedlicher DSP-Version in einer Master/Slave-Kette synchron betrieben werden sollen (siehe Seite 94) und auch nur dann, wenn die Anpassung der Kurzbefehlszählung nicht mit **sync_slaves** ausgeführt wird. • Auch wenn zuvor mit **set_dsp_mode** ein anderer Mode gesetzt wurde, gibt **set_dsp_mode** immer denjenigen DSP-Mode zurück, der beim Initialisieren der Karte eingestellt wurde. Dieser Rückgabewert (`initial_dsp_mode`) ist identisch zu Byte #2 von **get_rtc_version** (DSP-Versionsnummer). • Es kann kein DSP-Mode größer `initial_dsp_mode` gesetzt werden (`Mode` wird gegebenenfalls auf `initial_dsp_mode` geclippt). Es können also nur schnellere Karten an langsamere Karten angepasst werden, aber nicht umgekehrt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 523, OUT 524.
Verweise	**sync_slaves**, **get_rtc_version**

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, b`	Länge der Ellipsenhalbachsen in *Bits* als 32-Bit-Werte ohne Vorzeichen. Zulässiger Wertebereich: [1 … 8.388.607]. Überlaufende positive Werte werden auf den Randbereich geclippt.
	`Phi0`	Anfangsphasenwinkel in ° (Lage des Bogenstartpunkts relativ zum Endpunkt der Halbachse a) im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet “im Uhrzeigersinn”. `Phi0` wird auf den Wertebereich [0…<360°] normiert.
	`Phi`	Bogenwinkel in ° (zu markierender Ellipsenausschnitt) im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet “im Uhrzeigersinn”. Zulässiger Wertebereich: [–3600.0° … +3600.0°] (±10 volle Ellipsen). Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Die Position und Orientierung des zu markierenden ellipsenförmigen Bogens im zweidimensionalen Bildfeld werden mit **mark_ellipse_abs** oder **mark_ellipse_rel** angegeben. Zur Bedeutung der einzelnen Parameter siehe auch Seite 105. • Für `a` < 1 oder/und `b` < 1 wird **set_ellipse** durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`).	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die für die Ellipsenhalbachsen angegebenen Längenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [1 … 524.287].	
Versionsinfo	• Verfügbar ab Version DLL 518, OUT 517, RBF 515. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).	
Verweise	**mark_ellipse_abs**, **mark_ellipse_rel**	

Verzögerter kurzer Listenbefehl	**set_encoder_speed**
Funktion	Definiert die Soll-Encodergeschwindigkeit und weitere Parameter für die encodergeschwindigkeitsabhängige automatische Lasersteuerung.
Aufruf	`set_encoder_speed( EncoderNo, Speed, Smooth )`
Parameter	`EncoderNo` Nummer des Encoders, der für die Geschwindigkeitsmessung herangezogen werden soll als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0. = 1: Encoder-Zähler Encoder1. = 2 und 3: vektorielle Encodergeschwindigkeit: aus beiden Encodergeschwindigkeiten (aus den Zählerpulsraten beider Encoder-Zähler wird eine vektorielle Geschwindigkeit berechnet und dann bei der encodergeschwindigkeitsabhängigen automatischen Lasersteuerung mit `Mode`= 5 verwendet. Höherwertige Bits werden ignoriert.
	`Speed` Soll-Encodergeschwindigkeit (Zählerpulsrate) in [Zählerpulse/ms] im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: [100.0 … 16000.0]; für `Speed` > 16000.0 wird `Speed` auf 16000.0 geclippt, für 0.0 < `Speed` < 100.0 auf 100.0.
	`Smooth` Glättungsfaktor eines 2-stufigen Tiefpasses im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: [0.0 … 1.0]. Zu große Werte werden geclippt.
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 der Befehl schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Ein Encoder-Inkrement liefert 4 Zählerpulse. • 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 encodergeschwindigkeitsabhängigen Lasersteuerung bei kleinen Geschwindigkeitswerten! Die encodergeschwindigkeitsabhängige Korrektur ist nur dann empfehlenswert, wenn deutlich mehr als ein Zählerpuls pro Ausgabeperiode (10 µs) ankommt. • Siehe auch "Encodergeschwindigkeitsabhängige Lasersteuerung", Seite 168.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung mit Version DLL 536, OUT 536: `EncoderNo` = 2 und 3.
Verweise	**set_encoder_speed_ctrl**, **set_auto_laser_control**

Kontrollbefehl	set_encoder_speed_ctrl
Funktion	Identisch mit **set_encoder_speed**, aber ein Kontrollbefehl.
Aufruf	`set_encoder_speed_ctrl( EncoderNo, Speed, Smooth )`
Parameter	`EncoderNo` Nummer des Encoders, der für die Geschwindigkeitsmessung herangezogen werden soll als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0. = 1: Encoder-Zähler Encoder1. = 2 und 3: vektorielle Encodergeschwindigkeit: aus beiden Encodergeschwindigkeiten (aus den Zählerpulsraten beider Encoder-Zähler wird eine vektorielle Geschwindigkeit berechnet und dann bei der encodergeschwindigkeitsabhängigen automatischen Lasersteuerung mit `Mode`= 5 verwendet. Höherwertige Bits werden ignoriert.
	`Speed` Soll-Encodergeschwindigkeit (Zählerpulsrate) in [Zählerpulse/ms] im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: [100.0 … 16000.0]; für `Speed` > 16000.0 wird `Speed` auf 16000.0 geclippt, für 0.0 < `Speed` < 100.0 auf 100.0.
	`Smooth` Glättungsfaktor eines 2-stufigen Tiefpasses im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: [0.0 … 1.0]. Zu große Werte werden geclippt.
Hinweise	• Für `Speed` ≤ 0.0 oder `Smooth` < 0.0 wird der Befehl ignoriert (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Der Befehl wird auch ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten). Er wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung mit Version DLL 536, OUT 536: `EncoderNo` = 2 und 3.
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()`
Hinweise	• Falls beim Abarbeiten einer Liste ein **set_end_of_list**-Befehl erreicht wird und zuvor *kein* automatischer Listenwechsel aktiviert wurde (siehe "Automatischer Listenwechsel", Seite 79), so endet die Listenausführung. Die "Laser active"-Lasersteuersignale werden dann ausgeschaltet und ein evtl. mit **home_position** oder **home_position_xyz** definierter Home-Jump ausgeführt (für die Dauer des Home-Jumps ist der INTERNAL-BUSY-Status gesetzt). • Die Ausführung fährt dagegen beim Erreichen eines **set_end_of_list**-Befehls mit der jeweils anderen Liste fort, wenn vorher ein automatischer Listenwechsel *aktiviert* wurde. Die jeweils andere Liste kann auch stets "Liste 1" sein, falls "Liste 2" nicht konfiguriert wurde (`Mem2` = 0, siehe **config_list**). • Beim Abarbeiten des **set_end_of_list**-Befehls wird in jedem Fall der USED-Status der jeweiligen Liste (USED1 bzw. USED2) gesetzt und der BUSY-Status (BUSY1 bzw. BUSY2) der Liste rückgesetzt (siehe auch "Listenstatus", Seite 76). Der BUSY-Listenausführungsstatus wird dagegen nur rückgesetzt, wenn zuvor *kein* automatischer Listenwechsel aktiviert wurde. • Beim *Laden* des **set_end_of_list**-Befehls erfolgt niemals ein automatischer Listenwechsel des *Input*-Pointers (im Gegensatz zum automatischen Listenwechsel des *Output*-Pointers bei der *Ausführung* des **set_end_of_list**-Befehls, sofern zuvor mit **auto_change_pos** oder **start_loop** aktiviert). • Beim Laden des **set_end_of_list**-Befehls wird der READY-Status der Liste (READY1 bzw. READY2) gesetzt und der LOAD-Status (LOAD1 bzw. LOAD2) der Liste rückgesetzt. Zusätzlich wird ein Flush der gepufferten Listeneingabe ausgelöst (siehe Seite 75). • Der Befehl **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 bzw. ausgeführt wird (**get_last_error**-Returncode `RTC5_IGNORED`).
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Bei der RTC5 gibt es auch einen USED-Status. Dieser wird beim Laden des **set_end_of_list**-Befehls rückgesetzt.
Versionsinfo	Letzte Änderung Version OUT 517.
Verweise	**set_start_list**

Kontrollbefehl	set_extstartpos
Funktion	Definiert die Adresse (im Listenbereich “Liste 1” oder “Liste 2”), bei der die Ausführung bei nachfolgenden externen Listen-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^{20}–1)].
Hinweise	• In der Standardeinstellung wird die Ausführung durch einen externen Listen-Start am Anfang von “Liste 1” (`Pos` = 0) gestartet bzw. fortgesetzt. • `Pos` muss sich im Bereich von “Liste 1” oder “Liste 2” befinden. Andernfalls wird `Pos` = 0 gesetzt. • Die angegebene Startadresse wird so lange für alle nachfolgenden externen Listen-Starts verwendet, bis eine neue Adresse mit **set_extstartpos** oder **set_extstartpos_list** spezifiziert wird. Die Bereichsprüfung für `Pos` findet vor jedem externen Listen-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 ”Externer Listen-Start”, Seite 240.
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich). Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
Verweise	**set_extstartpos_list**, **set_control_mode**

Unverzögerter kurzer Listenbefehl	set_extstartpos_list
Funktion	Identisch mit **set_extstartpos**, aber ein Listenbefehl.
Aufruf	`set_extstartpos_list( Pos )`
Parameter	`Pos` Absolute Speicheradresse des ersten auszugebenden Listenbefehls als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … (2^{20}–1)].
Hinweise	• Dieser Listenbefehl kann innerhalb einer Liste verwendet werden, um sie mit der nachfolgenden Liste zu verknüpfen. • Siehe **set_extstartpos**.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich).
Verweise	**set_extstartpos**

Kontrollbefehl	set_ext_start_delay
Funktion	Stellt eine Streckenverzögerung für nachfolgende externe Listen-Starts ein, so dass die Ausführung der Listen-Starts relativ zum jeweils auslösenden Eingangssignal bzw. **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 Encoder-Zählers) als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31} \ldots +(2^{31}-1)]$ `EncoderNo` Nummer des zu verwendenden Encoder-Zählers als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0. = 1: Encoder-Zähler Encoder1.
Hinweise	• Zu externen Listen-Starts siehe Seite 240. • Die Streckenverzögerung wird in (relativen) Zähleinheiten des ausgewählten Encoder-Zählers angegeben (die RTC5-Encoder-Zähler werden durch ein externes oder simuliertes Encoder-Signal getriggert, siehe Seite 246). • Das Vorzeichen der Streckenverzögerung (Parameter `Delay`) sollte passend zur jeweiligen Encoderzählrichtung ausgewählt werden (diese richtet sich bei externer Triggerung nach der Bewegungsrichtung des Werkstücks). • Mit `Delay` = 0 wird die Streckenverzögerung deaktiviert. • Wird die Streckenverzögerung so angegeben, dass ein nachfolgender Start-Trigger (initiiert durch **simulate_ext_start** oder ein externes Startsignal) erfolgt, während der BUSY-Status (wie bei Listenausgabe) oder der INTERNAL-BUSY-Status (wie z.B. bei **goto_xy**) gesetzt ist, so wird durch diesen Start-Trigger kein Listen-Start ausgelöst (in diesem Fall wird Bit#11 des **get_startstop_info**-Rückgabewertes gesetzt). • Streckenverzögerungen werden auch mit **simulate_ext_start** eingestellt. Bei der Initialisierung (mit **load_program_file**), durch einen externen Listen-Stopp und durch **stop_execution** wird die Steckenverzögerung deaktiviert. Sie kann auch mit **set_control_mode** (Bit#2) deaktiviert werden. • Zuvor angetriggerte externe Listen-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 `RTC5_PARAM_ERROR`).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Der Befehl kann bei der RTC5 nicht nur (wie bei RTC4) zusammen mit einem realen Encoder (Hardware-Encoder), sondern auch mit einer mit **simulate_encoder** gestarteten Encoder-Simulation verwendet werden. Im RTC4-Kompatibilitätsmodus wird der angegebene `Delay`-Wert von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Verweise	**simulate_ext_start**, **set_ext_start_delay_list** **set_extstartpos**, **set_extstartpos_list**, **set_control_mode**

Normaler Listenbefehl	set_ext_start_delay_list
Funktion	Identisch mit **set_ext_start_delay**, aber ein Listenbefehl.
Aufruf	`set_ext_start_delay_list( Delay, EncoderNo )`
Parameter	`Delay` Siehe **set_ext_start_delay**. `EncoderNo` Siehe **set_ext_start_delay**.
Hinweise	• Für `EncoderNo` > 1 wird **set_ext_start_delay_list** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).
RTC4→ RTC5	Siehe **set_ext_start_delay**.
Verweise	**set_ext_start_delay**

Kontrollbefehl	set_firstpulse_killer
Funktion	Definiert die Länge des FirstPulseKiller-Signals für einen YAG-Laser.
Aufruf	`set_firstpulse_killer( Length )`
Parameter	`Length` Länge des FirstPulseKiller-Signals als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässige Werte: [0 … +(2^{26}–1)].
Hinweise	• Werte über (2^{26}–1) werden geclippt. • Die mit dem FirstPulseKiller-Signal verbundene Taktfrequenz ist immer 64 MHz. *1 Bit entspricht 1/64 µs*. • Der Signalpegel wird mit dem Befehl **set_laser_control** eingestellt. • In der YAG-Version 2 wird auch das Q-Switch-Delay entsprechend verändert (siehe Seite 150). • In der CO_2-Version bleibt der Befehl **set_firstpulse_killer** ohne Auswirkung. • Die Lasersteuerungsversion wird über den Befehl **set_laser_mode** ausgewählt (siehe Kapitel 7.4 "Lasersteuerung", Seite 144). • Q-Switch-Pulslänge und -Periodendauer können mit den Befehlen **set_laser_pulses_ctrl**, **set_laser_pulses** oder **set_laser_timing** eingestellt werden.
RTC4→ RTC5	Prinzipiell gleiche Funktionalität (außer erhöhter Wertebereich), allerdings: `Length` wird bei der RTC5 in Einheiten von 1/64 µs angegeben, im RTC4-Kompatibilitätsmodus dagegen wie bei der RTC4 in Einheiten von 1/8 µs. Im RTC4-Kompatibilitätsmodus wird der angegebene Wert von der RTC5 mit 8 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Verweise	**set_laser_mode**, **set_laser_timing**

Unverzögerter kurzer Listenbefehl	set_firstpulse_killer_list
Funktion	Identisch mit **set_firstpulse_killer**, aber ein Listenbefehl.
Aufruf	`set_firstpulse_killer_list( Length )`
Parameter	`Length` Länge des FirstPulseKiller-Signals als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässige Werte: [0 … +(2^{26}–1)].
RTC4→ RTC5	wie **set_firstpulse_killer**

Normaler Listenbefehl	**set_fly_2d**
Funktion	Aktiviert die "Processing on the fly"-Korrektur zur Kompensation einer linearen Werkstück-Bewegung in zwei Dimensionen (gemäß den Encoderwerten, die über die Encoder-Zähler Encoder0 und Encoder1 an die RTC5 übergeben und ggf. über eine spezielle 2D-Tabelle korrigiert werden) und setzt die zugehörigen Skalierungsfaktoren. Die aktuellen Encoderwerte werden zu den bisherigen Referenzwerten der 2D-Encoder-Kompensation hinzuaddiert und die Summen als neue Referenzwerte gespeichert (siehe Hinweise). Die Encoder-Zähler werden auf Null zurückgesetzt.
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, schaltet der Befehl 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 (Encoder-Zähler Encoder0) in *Bits*/*Count* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: 1/256 ≤ |`ScaleX`| ≤ 16.000,0 `ScaleY` Skalierungsfaktor für die y-Richtung (Encoder-Zähler Encoder1) in *Bits*/*Count* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: 1/256 ≤ |`ScaleY`| ≤ 16.000,0
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 8.7 ""Processing on the fly" (optional)", Seite 199. Zur Verwendung des **set_fly_2d**-Befehls siehe Kapitel 8.7.4 "Kompensation zweidimensionaler Bewegungen", Seite 206. • Für unzulässige Parameterwerte (z. B. `ScaleX` = 0) aktiviert **set_fly_2d** keine "Processing on the fly"-Korrektur bzw. 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- oder Bogenbefehls (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 200). • Mit **set_control_mode**, Bit#9 sollte **nicht** eingestellt werden, dass die Encoder-Zähler erst mit dem nachfolgenden externen Start-Trigger zurückgesetzt werden. Andernfalls gehen die Referenzwerte der 2D-Encoder-Kompensation verloren. • 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-Encoder-Kompensation für einen XY-Tisch verwenden wollen, da hierbei ebenfalls die Referenzwerte verloren gehen. • Wenn noch keine Korrekturtabelle für die 2D-Encoder-Kompensation auf die Karte geladen ist (siehe **load_fly_2d_table**), werden die Encoderwerte unkorrigiert verwendet.

Normaler Listenbefehl	**set_fly_2d**
Hinweise (Forts.)	• Falls eine Koordinatentransformation im virtuellen Bildfeld aktiv ist, wird diese noch *vor* der "Processing on the fly"-Korrektur auf das gesamte virtuelle Bildfeld angewendet (siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208). • Zum Einschalten einer **set_fly_2d**-"Processing on the fly"-Korrektur kann auch der Befehl **activate_fly_2d** verwendet werden.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Skalierungsfaktoren von der RTC5 mit 16 multipliziert (der zulässige Wertebereich ändert sich entsprechend).
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**init_fly_2d**, **load_fly_2d_table**, **get_fly_2d_offset**, **activate_fly_2d**, **activate_fly_xy**

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`, `Xmax`, `Ymin`, `Ymax` Bereichsgrenzen als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–524.288 … 524.287]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Zur Verwendung des Befehls siehe "Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)", Seite 214. • Bei der Initialisierung (mit **load_program_file**) werden die Bereichsgrenzen auf die folgenden Standardwerte gesetzt: `Xmin` = `Ymin` = –524.288 und `Xmax` = `Ymax` = +524.287. • Auch Bereichsgrenzen, für die der Parameterwert 0 übergeben wird, werden auf die oben genannten Standardwerte gesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525, OUT 527.
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`, `Zmax` Bereichsgrenzen als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Zur Verwendung des Befehls siehe auch Kapitel 8.7.9 “Überwachung der “Processing on the fly”-Korrektur”, Seite 213. • Bei der Initialisierung (mit **load_program_file**) werden die Bereichsgrenzen auf die folgenden Standardwerte gesetzt: `Zmin` = –32.768 und `Zmax` = +32.767. • Auch Bereichsgrenzen, für die der Parameterwert 0 übergeben wird, werden auf die oben genannten Standardwerte gesetzt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
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 Encoder-Zähler Encoder0 an die RTC5 übergeben werden) und setzt den zugehörigen Parameter `Resolution`. Der Encoder-Zähler Encoder0 wird gegebenenfalls auf Null zurückgesetzt (siehe Hinweis).
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_rot( Resolution )`
Parameter	`Resolution` Zählschritte (Counts) per Umdrehung (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: |`Resolution`| > 100,0.
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 8.7 ““Processing on the fly” (optional)”, Seite 199. • 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. Andernfalls wird der Default-Wert (0|0) verwendet. • Für unzulässige Parameterwerte (z.B. `Resolution` = 0) aktiviert **set_fly_rot** keine “Processing on the fly”-Korrektur bzw. 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- oder Bogenbefehls (ohne “**set_fly_rot**”-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar (siehe Seite 200). • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Seite 209. • Mit **set_control_mode**, Bit#9 kann zuvor eingestellt werden, ob der Encoder-Zähler Encoder0 durch **set_fly_rot** sofort oder erst mit dem nachfolgenden externen Start-Trigger (d.h. mit einem nachfolgenden externen Startsignal oder **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl, gegebenenfalls verzögert um eine mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** eingestellte Streckenverzögerung, siehe auch **set_control_mode**, Bit#2) zurückgesetzt wird.
RTC4→ RTC5	Unveränderte Funktionalität.
Versionsinfo	Letzte Änderung Version OUT 515.
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/SPI-Schnittstelle an die RTC5 übergeben werden) und setzt dabei den zugehörigen Parameter `Resolution`.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_fly_rot_pos( Resolution )`
Parameter	`Resolution` Zählschritte (Counts) per Umdrehung (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: |`Resolution`| > 100,0.
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 8.7 ““Processing on the fly” (optional)”, Seite 199. • 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. Andernfalls 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 bzw. 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- oder Bogenbefehls (ohne “**set_fly_rot_pos**“-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar (siehe Seite 200). • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Seite 209. • Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden (siehe Seite 188). • Das jeweils erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP/SPI-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen (siehe Seite 56).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version OUT 515.
Verweise	**set_rot_center**, **set_rot_center_list**, **fly_return**, **read_mcbsp**, **set_fly_rot**

Kontrollbefehl	set_fly_tracking_error
Funktion	Aktiviert oder deaktiviert eine Schleppverzugskompensation von Encoderwerten für “Processing on the fly”-Anwendungen.
Aufruf	`set_fly_tracking_error( TrackingErrorX, TrackingErrorY )`
Parameter	`TrackingErrorX`, `TrackingErrorY` Schleppverzug in Einheiten von [10 µs] für die jeweilige Achse (32 Bit-Werte mit Vorzeichen). Zulässiger Wertebereich: [0 … 65535]. Überzählige Bits werden ignoriert.
Hinweise	• Die Schleppverzugskompensation ist eine Kompensation in erster Näherung: EC = E + (E – EP) * `TrackingError`, wobei E der aktuelle Encoderwert bedeutet, EP derjenige aus dem vorherigen Takt und EC der kompensierte (für die “Processing on the fly”-Korrektur verwendete) Encoderwert. • Um eine Stufigkeit bei der Galvanometerbewegung zu vermeiden, sollten die verwendeten Encoder eine ausreichend hohe Auflösung (Counts pro 10 µs-Takt) besitzen. • Mit `TrackingErrorX,Y` = 0 (Initialisierungswert) ist diese Kompensation ausgeschaltet. • Die Befehle **store_encoder**, **read_encoder** und **get_encoder** verwenden weiterhin die originalen, unkorrigierten Encoderwerte. • Die Kompensation ist nur für lineare Bewegungen sinnvoll (Kapitel 8.7.2 “Kompensation linearer Bewegungen”, Seite 201), nicht für Drehbewegungen (Kapitel 8.7.3 “Kompensation von Rotationsbewegungen”, Seite 204). Für letztere sollte `TrackingErrorX,Y` auf 0 gesetzt sein. • Wenn die Encoder mit dem Start-Trigger (**set_control_mode**(Bit #9 = 1)) zurückgesetzt werden, kann der Ausgabewert im ersten 10 µs-Takt danach in unvorhersehbarer Weise fehlerhaft sein. Beim Zurücksetzen mit den Befehlen **set_fly_x** bzw. **set_fly_y** wird intern automatisch ein Leertakt abgewartet. • Die Kompensation wirkt nur bei “Processing on the fly”-Anwendungen mit Encodern, dagegen nicht mit Positionssignalen über die McBSP/SPI-Schnittstelle.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 533, OUT 534, RBF 524.

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 Encoder-Zähler Encoder0 an die RTC5 übergeben werden) und setzt den zugehörigen Skalierungsfaktor. Der Encoder-Zähler Encoder0 wird gegebenenfalls auf Null zurückgesetzt (siehe Hinweis).
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, schaltet der Befehl 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 (Encoder-Zähler Encoder0) in *Bits*/*Count* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: 1/256 ≤ |`ScaleX`| ≤ 16.000,0
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 8.7 ""Processing on the fly" (optional)", Seite 199. • Für unzulässige Parameterwerte (z.B. `ScaleX` = 0) aktiviert **set_fly_x** keine "Processing on the fly"-Korrektur bzw. 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- oder Bogenbefehls (ohne "**set_fly_x**"-"Processing on the fly"-Korrektur) ausgeführt. • Unterschiedliche "Processing on the fly"-Korrekturen sind nicht beliebig kombinierbar (siehe Seite 200). • Zur Deaktivierung der "Processing on the fly"-Korrektur siehe Seite 209. • Mit **set_control_mode**, Bit#9 kann zuvor eingestellt werden, ob der Encoder-Zähler Encoder0 durch **set_fly_x** sofort oder erst mit dem nachfolgenden externen Start-Trigger (d.h. mit einem nachfolgenden externen Startsignal oder **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl, gegebenenfalls verzögert um eine mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** eingestellte Streckenverzögerung, siehe auch **set_control_mode**, Bit#2) zurückgesetzt wird. • Falls eine Koordinatentransformation im virtuellen Bildfeld aktiv ist, wird diese noch *vor* der "Processing on the fly"-Korrektur auf das gesamte virtuelle Bildfeld angewendet (siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208). • Zum Einschalten einer **set_fly_x**/**set_fly_y**-"Processing on the fly"-Korrektur kann auch der Befehl **activate_fly_xy** verwendet werden.
RTC4→ RTC5	Unveränderte Funktionalität (außer veränderter Wertebereich) Im RTC4-Kompatibilitätsmodus wird der angegebene Skalierungsfaktor von der RTC5 mit 16 multipliziert (der zulässige Wertebereich ändert sich entsprechend).
Versionsinfo	Letzte Änderung Version OUT 515.
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/SPI-Schnittstelle an die RTC5 übergeben werden) und setzt dabei den zugehörigen Skalierungsfaktor.
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, schaltet der Befehl 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 *(RTC5-)Bits/(McBSP-)Bit* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: 1/256 ≤ |`ScaleX`| ≤ 16.000,0.
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 8.7 ""Processing on the fly" (optional)", Seite 199. • Für unzulässige Parameterwerte (z.B. `ScaleX` = 0) aktiviert **set_fly_x_pos** keine "Processing on the fly"-Korrektur bzw. 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- oder Bogenbefehls (ohne "**set_fly_x_pos**"-"Processing on the fly"-Korrektur) ausgeführt. • Unterschiedliche "Processing on the fly"-Korrekturen sind nicht beliebig kombinierbar (siehe Seite 200). • Zur Deaktivierung der "Processing on the fly"-Korrektur siehe Seite 209. • Für eindimensionale Korrekturen (wenn nur **set_fly_x_pos** verwendet wird) stehen an der McBSP/SPI-Schnittstelle 32 Bit-Werte (mit Vorzeichen) zur Verfügung, für zweidimensionale (**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/SPI-Schnittstelle. • Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für "Processing on the fly"-Anwendungen und für eine Online-Positionierung verwendet werden (siehe Seite 188). • Das jeweils erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP/SPI-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen (siehe Seite 56).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus wird der angegebene Skalierungsfaktor von der RTC5 mit 16 multipliziert (der zulässige Wertebereich ändert sich entsprechend).
Versionsinfo	Letzte Änderung Version OUT 515.
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 Encoder-Zähler Encoder1 an die RTC5 übergeben werden) und setzt den zugehörigen Skalierungsfaktor. Der Encoder-Zähler Encoder1 wird gegebenenfalls auf Null zurückgesetzt (siehe Hinweis).
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, schaltet der Befehl 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 (Encoder-Zähler Encoder1) in *Bits*/*Count* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: 1/256 ≤ |`ScaleY`| ≤ 16.000,0.
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 8.7 ""Processing on the fly" (optional)", Seite 199. • Für unzulässige Parameterwerte (z. B. `ScaleY` = 0) aktiviert **set_fly_y** keine "Processing on the fly"-Korrektur bzw. 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- oder Bogenbefehls (ohne "**set_fly_y**"-"Processing on the fly"-Korrektur) ausgeführt. • Unterschiedliche "Processing on the fly"-Korrekturen sind nicht beliebig kombinierbar (siehe Seite 200). • Zur Deaktivierung der "Processing on the fly"-Korrektur siehe Seite 209. • Mit **set_control_mode**, Bit#9 kann zuvor eingestellt werden, ob der Encoder-Zähler Encoder1 durch **set_fly_y** sofort oder erst mit dem nachfolgenden externen Start-Trigger (d.h. mit einem nachfolgenden externen Startsignal oder **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl, gegebenenfalls verzögert um eine mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** eingestellte Streckenverzögerung, siehe auch **set_control_mode**, Bit#2) zurückgesetzt wird. • Falls eine Koordinatentransformation im virtuellen Bildfeld aktiv ist, wird diese noch *vor* der "Processing on the fly"-Korrektur auf das gesamte virtuelle Bildfeld angewendet (siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208). • Zum Einschalten einer **set_fly_x**/**set_fly_y**-"Processing on the fly"-Korrektur kann auch der Befehl **activate_fly_xy** verwendet werden.
RTC4→ RTC5	Unveränderte Funktionalität (außer veränderter Wertebereich) Im RTC4-Kompatibilitätsmodus wird der angegebene Skalierungsfaktor von der RTC5 mit 16 multipliziert (der zulässige Wertebereich ändert sich entsprechend).
Versionsinfo	Letzte Änderung Version OUT 515.
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/SPI-Schnittstelle an die RTC5 übergeben werden) und setzt dabei den zugehörigen Skalierungsfaktor.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl 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 *(RTC5-)Bits/(McBSP-)Bit* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: 1/256 ≤ |`ScaleY`| ≤ 16.000,0.
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 8.7 ““Processing on the fly” (optional)”, Seite 199. • Für unzulässige Parameterwerte (z.B. `ScaleY` = 0) aktiviert **set_fly_y_pos** keine “Processing on the fly”-Korrektur bzw. 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- oder Bogenbefehls (ohne “**set_fly_y_pos**”-“Processing on the fly”-Korrektur) ausgeführt. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar (siehe Seite 200). • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Kapitel 8.7.5 “Deaktivieren der “Processing on the fly”-Korrektur”, Seite 209. • Für eindimensionale Korrekturen (wenn nur **set_fly_y_pos** verwendet wird) stehen an der McBSP/SPI-Schnittstelle 32 Bit-Werte (mit Vorzeichen) zur Verfügung, für zweidimensionale (**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/SPI-Schnittstelle. • Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden (siehe Seite 188). • Das jeweils erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP/SPI-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen (siehe Seite 56).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus wird der angegebene Skalierungsfaktor von der RTC5 mit 16 multipliziert (der zulässige Wertebereich ändert sich entsprechend).
Versionsinfo	Letzte Änderung Version OUT 515.
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 Encoder-Zähler an die RTC5 übergeben werden) und setzt den zugehörigen Skalierungsfaktor. Der angegebene Encoder-Zähler wird gegebenenfalls auf Null zurückgesetzt (siehe Hinweise).
Einschränkung	Falls die "Processing on the fly"-Option nicht freigeschaltet ist, schaltet der Befehl 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*. Im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: 1/256 ≤ |`ScaleZ`| ≤ 16.000,0.
	`EncoderNo` Nummer des zu verwendenden Encoder-Zählers. Als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0. = 1: Encoder-Zähler Encoder1.
Hinweise	• `ScaleZ` kann je nach Bewegungsrichtung des Werkstücks auch negativ sein. Lediglich der Absolutbetrag ist im Wertebereich eingeschränkt. • Beim Skalierungsfaktor `ScaleZ` wird (entgegen der sonst bei Markier- oder Sprung-Befehlen üblichen Angabe von z-Koordinaten in 16 Bits Fullscale) die 20-Bit-Basis zugrundegelegt, da die z-Koordinate RTC5-intern wie x und y mit 20 Bit Fullscale behandelt wird. Entsprechend ist für `ScaleZ` auch ein aktivierter RTC4-Kompatibilitätsmodus wirksam. Zur "Processing on the fly"-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.7 ""Processing on the fly" (optional)", Seite 199. • Für unzulässige Parameterwerte von `ScaleZ` (z.B. `ScaleZ` = 0) aktiviert **set_fly_z** keine "Processing on the fly"-Korrektur bzw. 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- oder Bogenbefehls (ohne "**set_fly_z**"-"Processing on the fly"-Korrektur) ausgeführt. • Zur Deaktivierung der "Processing on the fly"-Korrektur siehe Abschnitt "Hinweise zur Verwendung", Seite 216. • Mit **set_control_mode**, Bit #9 kann zuvor eingestellt werden, ob der angegebene Encoder-Zähler durch **set_fly_z** sofort oder erst mit dem nachfolgenden externen Start-Trigger (d.h. mit einem nachfolgenden externen Startsignal oder **simulate_ext_start**- oder **simulate_ext_start_ctrl**-Befehl, gegebenenfalls verzögert um eine mit **simulate_ext_start**, **set_ext_start_delay** oder **set_ext_start_delay_list** eingestellte Streckenverzögerung, siehe auch **set_control_mode**, Bit #2) zurückgesetzt wird. • Für `EncoderNo` > 1 wird **set_fly_z** durch ein **list_nop** ersetzt (**get_last_error**-Return-code `RTC5_PARAM_ERROR`).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus wird der angegebene Skalierungsfaktor von der RTC5 mit 16 multipliziert (der zulässige Wertebereich ändert sich entsprechend).
Versionsinfo	Verfügbar ab Version DLL 530, OUT 531.
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 Bereich: [0 … 7]. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`Value`	Gewünschter Variablenwert als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [0 … (2^{32}–1)].
Hinweise	• Siehe auch Kapitel 6.9.1 "Freie Variablen", Seite 102.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532. Letzte Änderung (Version DLL 539, OUT 539): Der Wertebereich des Parameters `No` wurde auf [0…7] erhöht.	
Verweise	**set_free_variable_list**, **get_free_variable**, **set_trigger**	

Unverzögerter kurzer Listenbefehl	set_free_variable_list	
Funktion	Identisch mit **set_free_variable**, aber ein Listenbefehl.	
Aufruf	`set_free_variable_list( No, Value )`	
Parameter	`No`	Siehe **set_free_variable**.
	`Value`	Siehe **set_free_variable**.
Hinweise	• Siehe **set_free_variable**.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.	
Verweise	**set_free_variable**	

Kontrollbefehl	set_hi
Funktion	Definiert Gain- 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: Primärer Scan-Kopf-Anschluss. = 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich).
	`GalvoGainX`, `GalvoGainY` Gain-Werte (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Wertebereich: [0,01…100].
	`GalvoOffsetX`, `GalvoOffsetY` Offset-Werte in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–524.288 … 524.287].
Hinweise	• Zur Verwendung des Befehls siehe Abschnitt “Kundeneigene Kalibrierung”, Seite 227. • Die angegebenen Gain- und Offset-Werte überschreiben die mit **auto_cal** eingestellten Werte bzw. können bei einem nachfolgenden **auto_cal**-Befehl selbst wieder überschrieben werden. • Änderungen der Gain- und Offset-Werte können harte Sprünge verursachen. Um diese zu vermeiden, wird der Übergang automatisch mit der voreingestellten Sprunggeschwindigkeit ausgeführt (siehe **set_jump_speed**). • Für einen unzulässigen Parameterwert wird der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`). Der Befehl wird auch nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). Bei einem `RTC5_PARAM_ERROR` wird der BUSY-Status nicht überprüft, d. h. Returncode `RTC5_BUSY` und `RTC5_PARAM_ERROR` kommen nicht gleichzeitig vor. • Falls die Option “Zweite-Scan-Kopf-Steuerung” nicht freigeschaltet ist, bleiben Werte, die für den sekundären Scan-Kopf-Anschluss spezifiziert wurden, wirkungslos.
RTC4→ RTC5	Neuer Befehl.
Verweise	**auto_cal**, **get_hi_pos**, **write_hi_pos**

Kontrollbefehl	set_input_pointer
Funktion	Öffnet den Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Input-Pointer an die angegebene (absolute) Adresse im Listenbereich (“Liste 1” oder “Liste 2”). Der nächste Listenbefehl wird an dieser Adresse gespeichert und alle nachfolgenden Listenbefehle an den darauffolgenden Adressen in der gewählten Liste.
Aufruf	`set_input_pointer( Pos )`
Parameter	`Pos` Position (absolute Speicheradresse) des Input-Pointers [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Der Befehl **set_input_pointer** wirkt prinzipiell wie der Befehl **set_start_list_pos** (siehe Hinweise dort). Der Input-Pointer 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→ RTC5	Unveränderte Funktionalität.
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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask_0` gesetzten Bits von `IOvalue` gleich `0` sind).
Aufruf	`set_io_cond_list( Mask1, Mask0, MaskSet )`
Parameter	`Mask1`, `Mask0`, `MaskSet` 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
Hinweise	• Der Befehl setzt nur diejenigen Bits des Digital-Ausgangs, die im Parameter `MaskSet` gesetzt sind und lässt die anderen Bits unverändert. • Siehe auch Abschnitt “16-Bit-Digital-Eingang und -Ausgang”, Seite 51 und Kapitel 9.3.2 “Bedingte Befehlsausführung”, Seite 244.

Beispiele (Pascal)	• Setze Bit #4 des Digital-Ausgangs (DIGITAL OUT4), falls Bit #0 des Digital-Eingangs (DIGITAL IN0) gesetzt und die Bits #1 bis #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→ RTC5	Unveränderte Funktionalität.
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 bzw. unterdrückt und deaktiviert den Sprungmodus für 2D-Sprünge und legt die zugehörigen Parameter fest (vor einer Aktivierung wird gegebenenfalls eine Überprüfung ausgeführt, siehe auch Seite 177).
Voraussetzung	Eine Freigabe ist nur möglich, wenn mindestens an einem der beiden Scan-Kopf-Anschlüsse ein intelli*SCAN*, intelli*cube*, intelli*WELD* oder intelli*DRILL* Scan-System (mit Software-Version 2078 oder größer) angeschlossen ist, das mit mindestens einem Sprungtuning ausgestattet ist. Ansonsten bleibt der Befehl 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 (gegebenenfalls nur nach erfolgreich bestandener Prüfung, s.u.) freigegeben, aber deaktiviert. Anschließend ist auch mit **set_jump_mode_list** eine Umschaltung des Flags möglich. = 1: Der Sprungmodus wird (gegebenenfalls nur nach erfolgreich bestandener Prüfung, s.u.) 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.
	`VA1, VA2, VB1, VB2, JA1, JA2, JB1, JB2` Nummern (als 32-Bit-Wert ohne Vorzeichen) 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 Primärer Scan-Kopf-Anschluss. B Sekundärer Scan-Kopf-Anschluss. 1: Achse X (STATUS-Kanal, Galvanometerscanner 2). 2: Achse Y (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 wievielen Tunings das angeschlossene Scan-System ausgestattet ist).

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 (wie gewünscht) auf –1 umgeschaltet (Sprungmodus deaktiviert und unterdrückt). –2 Busy-Fehler: Karte BUSY oder INTERNAL-BUSY (**get_last_error**-Returncode: `RTC5_BUSY`). –3 Karte reagiert nicht: evtl. kein Programm geladen oder PCI-Fehler (**get_last_error**-Returncode: `RTC5_TIMEOUT`). –4 Access-Fehler: Karte für anderes Anwenderprogramm reserviert (**get_last_error**-Returncode: `RTC5_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 = Errorcode für primären Scan-Kopf-Anschluss. Byte #2 = Errorcode für sekundären Scan-Kopf-Anschluss. Byte #3 = 0. Mit Errorcode: =1: Achse X (Galvanometerscanner 2) reagiert nicht oder kein intelli*SCAN*, intelli*cube*, intelli*WELD* oder intelli*DRILL* Scan-System (mit Software-Version 2078 oder größer) angeschlossen. =2: Achse Y (Galvanometerscanner 1) reagiert nicht oder kein intelli*SCAN*, intelli*cube*, intelli*WELD* oder intelli*DRILL* Scan-System (mit Software-Version 2078 oder größer) angeschlossen. =4: Keine Korrekturtabelle zugewiesen. =8: Falsche Tuningnummer(n): falscher Typ oder zum schnellen Umschalten ungeeignet.
Hinweise	• Zur Verwendung des Befehls siehe Kapitel 8.1.5 "Sprungmodus", Seite 176. • Eine Überprüfung (siehe auch "Voraussetzung und Aktivierung", Seite 177) wird nur durchgeführt, wenn `Flag` vor Aufruf des Befehls auf –1 stand (Initialisierungszustand) und/oder wenn die angegebenen Tuningnummern nicht mit den auf der Karte gespeicherten übereinstimmen. Andernfalls wird nur das `Flag` umgeschaltet. Für die Prüfung darf die Karte weder BUSY noch INTERNAL-BUSY sein, da zwischenzeitlich der rückzuübertragende Datentyp geändert und dazu die automatische Lasersteuerung deaktiviert werden muss (beides wird am Befehlsende wieder hergestellt). Je nach Ausgang der Prüfung werden verschiedene Fehlercodes zurückgeliefert (s.o.). Im Fehlerfalle wird `Flag` stets 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 hergeschaltet werden, ohne dass eine Überprüfung erneut durchgeführt werden muss.

Kontrollbefehl	set_jump_mode
Hinweise (Forts.)	• 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 sekundären Scan-Kopf-Anschluss (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 Seite 176). Falls das dann aktuell eingestellte Tuning nicht mit dem unter **set_jump_mode** angegebenen Sprung- 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, Errorcode = 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.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 525. • Letzte Änderung (Version DLL 530, OUT 531, RBF 522): Veränderter Errorcode für Rückgabe >1 (Byte#1 und Byte#2).
Verweise	**set_jump_mode_list**, **load_jump_table_offset**, **set_jump_table**

Normaler Listenbefehl	**set_jump_mode_list**
Funktion	Aktiviert und deaktiviert bzw. unterdrückt den Sprungmodus für 2D-Sprünge.
Voraussetzung	Siehe **set_jump_mode**.
Aufruf	`set_jump_mode_list( Flag )`
Parameter	`Flag` Siehe **set_jump_mode**.
Hinweise	• Zur Verwendung des Befehls siehe Kapitel 8.1.5 "Sprungmodus", Seite 176. • **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 Sprunglängengrenzen 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 bzw. 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 der Befehl wirkungslos.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 525.
Verweise	**set_jump_mode**

Unverzögerter kurzer Listenbefehl	**set_jump_speed**
Funktion	Definiert die Sprunggeschwindigkeit für die nachfolgenden Vektorbefehle.
Aufruf	`set_jump_speed( Speed )`
Parameter	`Speed` Sprunggeschwindigkeit in *Bits pro ms* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [1,6 … 800.000,0].
Hinweise	• Standardmäßig ist eine Sprunggeschwindigkeit von 10.000 *Bits pro ms* voreingestellt. • Die angegebene Sprunggeschwindigkeit (bzw. die voreingestellte Sprunggeschwindigkeit, falls kein anderer Wert spezifiziert wurde) wird für alle nachfolgenden Jump-Befehle verwendet, bis ein neuer Wert spezifiziert wird. • Um die tatsächliche Sprunggeschwindigkeit *v* in der Bildebene (in *Meter pro Sekunde*), zu erhalten, muss der angegebene Geschwindigkeitswert (in *Bits pro ms*) durch den Kalibrierungsfaktor *K* (in *Bits pro mm*) aus der Korrekturtabelle geteilt werden: v_{jump} = `Speed` / *K* Der Kalibrierungsfaktor *K* kann mit **get_table_para** oder **get_head_para** aus der Korrekturtabelle ausgelesen werden. • Der Befehl **set_jump_speed** existiert auch als Kontrollbefehl **set_jump_speed_ctrl**.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Die Bildfeldkoordinaten für die X- und Y-Achse (nicht dagegen für die Z-Achse) und alle damit verbundenen Parameter (z.B. Sprunggeschwindigkeit oder Wobbelamplitude) werden bei der RTC5 als 20-Bit-Werte angegeben, im RTC4-Kompatibilitätsmodus dagegen wie bei der RTC4 als 16-Bit-Werte. Im RTC4-Kompatibilitätsmodus wird der angegebene Wert für `Speed` von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [0,1 … 50.000,0]).
Verweise	**jump_abs**, **jump_rel**, **set_mark_speed**, **set_jump_speed_ctrl**

Kontrollbefehl	**set_jump_speed_ctrl**
Funktion	Identisch mit **set_jump_speed**, aber ein Kontrollbefehl.
Aufruf	`set_jump_speed_ctrl( Speed )`
Parameter	`Speed` Sprunggeschwindigkeit in *Bits pro ms* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [1,6 … 800.000,0].
Hinweise	• Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten). Er wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus: wie **set_jump_speed**.
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 oder INTERNAL-BUSY (**get_last_error**-Returncode: RTC5_BUSY). 4 Verify-Fehler: DSP-Speicher-Fehler. 11 Treiber-Fehler.
Hinweise	• Der Befehl **set_jump_table** wird *nicht* ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 525.
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` (32-Bit-Wert ohne Vorzeichen): Bit # Beschreibung Bit #0 (LSB) Pulse Switch Setting (betrifft weder Lasermode 4 noch Lasermode 6): Die Einstellung betrifft nur diejenigen Lasersteuersignale (genauer: diejenigen LASER1- oder LASER2-"Laser active"-Modulationspulse in der CO_2-Version bzw. LASER1-Q-Switch-Pulse in den YAG-Versionen), die am Ende des LASERON-Signals noch nicht vollständig abgearbeitet sind, siehe Abb. 50 und Abb. 51. = 0: Die Signale werden am Ende des LASERON-Signals abgeschnitten. = 1: Der letzte Puls wird trotz beendeten LASERON-Signals noch vollständig ausgeführt. Bit #1 Phasenverschiebung der Lasersteuersignale (betrifft weder Lasermode 4 noch Lasermode 6) = 0: Keine Phasenverschiebung. = 1: CO_2-Version: Das LASER1-Signal wird mit dem LASER2-Signal vertauscht. YAG-Versionen: Das LASER1-Signal wird um 180° (eine halbe Signal-Periodendauer) nach hinten phasenverschoben. Bit #2 Freigabe oder Unterdrückung der Lasersteuersignale für den "Laser active"-Betrieb. = 0: Die "Laser active"-Lasersteuersignale werden freigegeben. = 1: Die "Laser active"-Lasersteuersignale werden unterdrückt (die Lasersignal-Ausgänge sind dann im hochohmigen tristate-Zustand). Bit #3 LASERON-Signal-Pegel. = 0: Das Signal am LASERON-Ausgang wird high-aktiv gesetzt. = 1: Das Signal am LASERON-Ausgang wird low-aktiv gesetzt. Bit #4 LASER1/LASER2-Signal-Pegel. = 0: Die Signale an den LASER1- und LASER2-Ausgängen werden high-aktiv gesetzt. = 1: Die Signale an den LASER1- und LASER2-Ausgängen werden low-aktiv gesetzt. 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. Bit #6 = 0: Die Ausgabe-Synchronisation wird ausgeschaltet (Standard-Einstellung). = 1: Die Ausgabe-Synchronisation wird eingeschaltet (siehe Seite 169).

Kontrollbefehl	set_laser_control		
Parameter (Forts.)	Bit #7	= 0:	Der “konstante Pulslänge”-Modus wird ausgeschaltet (Standard-Einstellung).
		= 1:	Der “konstante Pulslänge”-Modus wird eingeschaltet (siehe Seite 157 und **set_pulse_picking_length**).
	Bits #8-15	Reserviert.	
	Bit #16	PowerOK von Kopf A und Achse X wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #17	TempOK von Kopf A und Achse X wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #18	PosAck von Kopf A und Achse X wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #19	PowerOK von Kopf A und Achse Y wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #20	TempOK von Kopf A und Achse Y wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #21	PosAck von Kopf A und Achse Y wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #22	PowerOK von Kopf B und Achse X wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #23	TempOK von Kopf B und Achse X wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #24	PosAck von Kopf B und Achse X wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #25	PowerOK von Kopf B und Achse Y wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #26	TempOK von Kopf B und Achse Y wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #27	PosAck von Kopf B und Achse Y wird für die automatische Lasersignalunterdrückung verwendet.	
	Bit #28:	= 1:	Im Fehlerfall wird bei einer automatischen Überwachung (bei der automatischen Lasersignalunterdrückung) auch automatisch ein /STOP-Signal erzeugt (Liste hält an, Lasersteuersignale gehen dauerhaft aus).
	Bits #29…31	Reserviert.	

Kontrollbefehl	set_laser_control
Hinweise	• In der Standardeinstellung (nach einem Reset) sind alle Bits auf 0 gesetzt. Nach einem Hardware-Reset werden die Einstellungen allerdings erst durch einen mindestens einmaligen Aufruf von **set_laser_control** wirksam. Vorher befinden sich alle Lasersignal-Ausgänge (LASERON, LASER1 und LASER2) im (hochohmigen) tristate-Zustand. TTL-Zustände (LOW oder HIGH) stellen sich erst dann ein, wenn **set_laser_control** aufgerufen wird und dabei der gewünschte TTL-Pegel festgelegt wird – siehe auch Seite 144. Auch nach dem Befehl **load_program_file**, der die Lasersteuersignale deaktiviert (den Signalpegel an den Lasersignal-Ausgängen aber ansonsten unverändert lässt) muss **set_laser_control** zur erstmaligen Aktivierung aufgerufen werden. Der Befehl **set_laser_control** sollte daher bei der Inbetriebnahme der RTC5 am Anfang eines jeden RTC5 Anwenderprogramms nach **load_program_file** aufgerufen werden. • Bei den Vorgängerversionen der RTC5 wurden die Lasersignal-Pegel über (lötbare) Jumper definiert. Bei der RTC5 werden sie von Benutzern selbst komplett softwaregesteuert. Mit dem Befehl **get_startstop_info** (Bit #9) kann der aktuelle Status der Lasersteuersignale (Bit #9) ausgelesen werden und ob die Signale high- oder low-aktiv eingestellt sind (Bit#13). • Das Freigeben und Unterdrücken der Lasersteuersignale kann auch mit den Befehlen **enable_laser** bzw. **disable_laser** durchgeführt 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 Seite 144). • Die Phasenverschiebung der Lasersteuersignale (Bit #1 = 1) kann z. B. im Softstart-Modus (siehe Seite 152) oder im Pixelausgabemodus (siehe Seite 217) eingesetzt werden. • Zur Verwendung der Bits #16 – 28 für die automatische Lasersignalunterdrückung siehe Seite 146.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Ergänzung Version DLL 515, OUT 514, RBF 512: Bit#5 für **laser_on_pulses_list**. Ergänzung Version DLL 530, OUT 531, RBF 522: Bit#6 für die Ausgabe-Synchronisation. Ergänzung Version DLL 533, OUT 534, RBF 524: Bit#7 für den "konstante Pulslänge"-Modus. Ergänzung Version DLL 536, OUT 536: Bit#28 für /STOP im Fehlerfall.
Verweise	**set_laser_mode**

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 0,5 µs*. Zulässiger Bereich: $[-2^{31} \ldots +(2^{21}-1)]$.
	`LaserOffDelay` LaserOff-Delay als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 0,5 µs*. Zulässiger Bereich: $[0 \ldots +(2^{21}-1)]$.
Hinweise	• Werte über $(2^{21}-1)$ werden geclippt. • 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 "Automatische Delay-Anpassung" auf Seite 121. Die bei den Vorgängerkarten der RTC5 notwendige Bedingung `LaserOffDelay` > `LaserOnDelay` entfällt dadurch. • Beachten Sie die Hinweise in Kapitel 7.2.1 "Laser-Delays", Seite 111. • Durch ein negatives LaserOn-Delay wird die Gesamtmarkierungszeit erhöht. • Der optionale XY2-100-Konverter (siehe Seite 43) 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( 20, 20 )`.
RTC4→ RTC5	Prinzipiell gleiche Funktionalität (außer erhöhter Wertebereich), allerdings: `LaserOnDelay` und `LaserOffDelay` werden bei der RTC5 in Einheiten von 0,5 µs angegeben, im RTC4-Kompatibilitätsmodus dagegen wie bei der RTC4 in Einheiten von 1 µs. Im RTC4-Kompatibilitätsmodus werden die angegebenen Delay-Werte von der RTC5 mit 2 multipliziert (die zulässigen Wertebereiche verringern sich entsprechend).
Verweise	**set_scanner_delays**

Kontrollbefehl	set_laser_mode
Funktion	Bestimmt die Lasersteuerungsversion der RTC5.
Aufruf	`set_laser_mode( Mode )`
Parameter	`Mode` Als 32-Bit-Wert ohne Vorzeichen. = 0: CO_2-Version. = 1: YAG-Version 1. = 2: YAG-Version 2. = 3: YAG-Version 3. = 4: Lasermode 4. = 5 YAG-Version 5. = 6: Lasermode 6.
Hinweise	• Welche Lasersignale verfügbar sind, hängt von der gewählten Lasersteuerungsversion ab, siehe auch Kapitel 7.4 "Lasersteuerung", Seite 144. • Der Befehl sollte nur einmal – am Anfang des Programms – verwendet werden, siehe auch "Inbetriebnahme der RTC5-Karte", Seite 69.
RTC4→ RTC5	Die YAG-Version 5 und den Lasermode 6 gibt es erst ab der RTC5. Sonst unveränderte Funktionalität.
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	Definiert die Default-Ausgabewerte für die Ausgänge ANALOG OUT1 und ANALOG OUT2, sowie der 8-Bit-Digital-Ausgang der RTC5-Karte.	
Aufruf	`set_laser_off_default( AnalogOut1, AnalogOut2, DigitalOut )`	
Parameter	`AnalogOut1, AnalogOut2`	12-Bit-Ausgabewerte für die analogen Ausgänge ANALOG OUT1 bzw. ANALOG OUT2 (siehe auch Seite 48) als 32-Bit-Werte ohne Vorzeichen. Höherwertige Bits werden ignoriert (Ausnahme: `AnalogOut1/2` = "–1", siehe Hinweise bei **set_port_default**).
	`DigitalOut`	8-Bit-Ausgabewert für den 8-Bit-Digital-Ausgang (siehe auch Seite 52) als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert (Ausnahme: `DigitalOut`= "–1", siehe Hinweise bei **set_port_default**).
Hinweise	• Die Defaultwerte können auch mit dem Befehl **set_port_default** festgelegt werden. Siehe auch alle Hinweise dort.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden `AnalogOut1` und `AnalogOut2` jeweils mit 4 multipliziert (bei der RTC4 waren die Analogausgänge nur für 10 Bit-Werte ausgelegt).	
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. Bits # 2…31:Reserviert.
Hinweise	• Siehe auch Kapitel 9.1.3 “2 Bit-Digital-Ausgang”, Seite 233.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_laser_pin_out_list**, **get_laser_pin_in**

Unverzögerter kurzer Listenbefehl	set_laser_pin_out_list
Funktion	Identisch mit **set_laser_pin_out**, aber ein Listenbefehl.
Aufruf	`set_laser_pin_out_list( Pins )`
Parameter	`Pins` Ausgabewert (DIGITAL OUT1 und DIGITAL OUT2) als 32-Bit-Wert ohne Vorzeichen. Bit # 0: DIGITAL OUT1. Bit # 1: DIGITAL OUT2. Bits # 2…31:Reserviert.
Hinweise	• Siehe **set_laser_pin_out**.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_laser_pin_out**

Verzögerter kurzer Listenbefehl	**set_laser_pulses**	
Funktion	Definiert die Ausgabeperiode und die Pulslänge der Lasersignale LASER1 und LASER2 für den “Laser active”-Betrieb.	
Aufruf	`set_laser_pulses( HalfPeriod, PulseLength )`	
Parameter	`HalfPeriod`	*Halbe* (Q-Switch-) Ausgabeperiode in *Bits* als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
	`PulseLength`	Pulslänge der Lasersignale LASER1 und LASER2 in *Bits* als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
Hinweise	• Beachten Sie, dass unter `HalfPeriod` die *halbe* Periodendauer angegeben werden muss (siehe Abb. 50 und Abb. 51). • Wenn `HalfPeriod` und/oder `PulseLength` den Wert 0 haben, werden keine Lasersignale ausgegeben. • `HalfPeriod` sollte bei Verwendung des Softstart-Modus (siehe Seite 152) nicht kleiner als 104 sein (dies wird aber nicht automatisch kontrolliert). Dies entspricht einer Laserpulsfrequenz von ca. 308 kHz. • Wenn `PulseLength` größer gleich der Ausgabeperiode (2 × `HalfPeriod`) ist, bleibt der Laser dauernd an. • Der Signalpegel wird mit dem Befehl **set_laser_control** eingestellt. • Der Befehl **set_laser_pulses** existiert auch als Kontrollbefehl **set_laser_pulses_ctrl**. • Der Befehl **set_laser_pulses** ist weitgehend synonym zum bisherigen RTC4-Befehl **set_laser_timing**, hat aber weniger Parameter.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus müssen die Parameter-Werte in 1/8 µs angegeben werden (die 1 µs-Zeitbasis – siehe **set_laser_timing** – wird nicht mehr unterstützt). Sie werden dann intern auf 1/64 µs-Einheiten umgerechnet (d.h. mit 8 multipliziert). Die zulässigen Wertebereiche verringern sich entsprechend (der kleinste sinnvolle Wert für `HalfPeriod` mit Softstart-Modus ist dann 13).	
Verweise	**set_laser_pulses_ctrl**, **set_laser_timing**	

Kontrollbefehl	**set_laser_pulses_ctrl**	
Funktion	Identisch mit **set_laser_pulses**, aber ein Kontrollbefehl.	
Aufruf	`set_laser_pulses_ctrl( HalfPeriod, PulseLength )`	
Parameter	`HalfPeriod`	*Halbe* (Q-Switch-) Ausgabeperiode in *Bits* als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
	`PulseLength`	Pulslänge der Lasersignale LASER1 und LASER2 in *Bits* als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
Hinweise	• siehe **set_laser_pulses**	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus: wie **set_laser_pulses**.	
Verweise	**set_laser_pulses**, **set_laser_timing**	

Verzögerter kurzer Listenbefehl	**set_laser_timing**	
Funktion	Definiert die Ausgabeperiode und die Pulslänge der Lasersignale 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 RTC5-Modus: *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [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 Lasersignale LASER1 und LASER2 in *Bits* als 32-Bit-Wert ohne Vorzeichen. • Im RTC5-Modus: *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [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)`	(Als 32-Bit-Wert ohne Vorzeichen.) Der Wert von `PulseLength2` wird ignoriert und nicht mehr verwendet. Die Pulslänge der Lasersignale LASER1 und LASER2 ist bei der RTC5 immer identisch und wird durch `PulseLength1` festgelegt.
	`TimeBase`	Als 32-Bit-Wert ohne Vorzeichen. • Im RTC5-Modus 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*.

Verzögerter kurzer Listenbefehl	set_laser_timing
Hinweise	• Beachten Sie, dass unter `HalfPeriod` die *halbe* Periodendauer angegeben werden muss (siehe Abb. 50 und Abb. 51). • Wenn `HalfPeriod` und/oder `PulseLength1` den Wert 0 haben, werden keine Lasersignale ausgegeben. • `HalfPeriod` sollte bei Verwendung des Softstart-Modus (siehe Seite 152) nicht kleiner als 104 sein (dies wird aber nicht automatisch kontrolliert). Dies entspricht einer Laserpulsfrequenz von ca. 308 kHz. • Wenn `PulseLength1` größer gleich der Ausgabeperiode (2 × `HalfPeriod`) ist, bleibt der Laser dauernd an. • Die Einstellungen der Taktfrequenz gelten nur für die Parameter *dieses* Befehls. • Im RTC4-Kompatibilitätsmodus ist es im Allgemeinen empfehlenswert eine Taktfrequenz von 8 MHz zu wählen. Eine Taktfrequenz von 1 MHz sollte nur wenn unbedingt nötig eingestellt werden. • Beachten Sie auch die Hinweise in Kapitel 7.4 "Lasersteuerung", Seite 144. • Im RTC5-Modus ist der Befehl **set_laser_timing** synonym zu **set_laser_pulses**. • Der Signalpegel wird mit dem Befehl **set_laser_control** eingestellt. • Im Lasermode 4 und Lasermode 6 ist die Zeitbasis für die Signale LASER1 und LASER2 unabhängig von `TimeBase` immer 1/64 µs (im RTC5-Modus) bzw. immer 1/8 µs (im RTC4-Kompatibilitätsmodus).
RTC4→ RTC5	• Die Pulslänge der Lasersignale LASER1 und LASER2 ist bei der RTC5 immer identisch. • Taktfrequenz: – Im RTC5-Modus: feste Taktfrequenz von 64 MHz. – Bei der RTC4 und im RTC4-Kompatibilitätsmodus: 1 MHz oder 8 MHz.
Verweise	**set_laser_pulses_ctrl**, **set_laser_pulses**, **set_laser_mode**, **set_firstpulse_killer**, **set_firstpulse_killer_list**, **set_standby**, **set_standby_list**

Unverzögerter kurzer Listenbefehl	set_list_jump
Funktion	Bewirkt bei der Ausführung einen unbedingten Sprung auf die angegebene Adresse des Listenspeichers. Der dortige nächste Befehl wird sofort und ohne Verzögerung ausgeführt.
Aufruf	`set_list_jump( Pos )`
Parameter	`Pos` Absolute Sprungadresse [0 … (2^{20}–1)] als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Der Befehl **set_list_jump** ist synonym zum Befehl **list_jump_pos** – siehe Hinweise dort.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich).
Verweise	**list_jump_pos**

Verzögerter kurzer Listenbefehl	**set_mark_speed**
Funktion	Definiert die Markiergeschwindigkeit für die nachfolgenden Vektor- und Bogenbefehle.
Aufruf	`set_mark_speed( Speed )`
Parameter	`Speed` Markiergeschwindigkeit in *Bits pro ms* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [1,6 … 800.000,0].
Hinweise	• Standardmäßig ist eine Markiergeschwindigkeit von 1000 *Bits pro ms* voreingestellt. • Die angegebene Markiergeschwindigkeit (bzw. die voreingestellte Markiergeschwindigkeit, falls kein anderer Wert spezifiziert wurde) wird für alle nachfolgenden Mark-und Bogenbefehle verwendet, bis ein neuer Wert spezifiziert wird. • Um die tatsächliche Markiergeschwindigkeit *v* in der Bildebene (in *Meter pro Sekunde*), zu erhalten, muss der angegebene Geschwindigkeitswert (in *Bits pro ms*) durch den Kalibrierungsfaktor *K* (in *Bits pro mm*) aus der Korrekturtabelle geteilt werden: v_{mark} = `Speed` / *K* Der Kalibrierungsfaktor *K* kann mit **get_table_para** oder **get_head_para** aus der Korrekturtabelle ausgelesen werden. • Der Befehl **set_mark_speed** existiert auch als Kontrollbefehl **set_mark_speed_ctrl**.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Die Bildfeldkoordinaten für die X- und Y-Achse (nicht dagegen für die Z-Achse) und alle damit verbundenen Parameter (z. B. Markiergeschwindigkeit oder Wobbelamplitude) werden bei der RTC5 als 20-Bit-Werte angegeben, im RTC4-Kompatibilitätsmodus dagegen wie bei der RTC4 als 16-Bit-Werte. Im RTC4-Kompatibilitätsmodus wird der angegebene Wert für `Speed` von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [0,1 … 50.000,0]).
Verweise	**mark_abs**, **mark_rel**, **set_jump_speed**, **set_mark_speed_ctrl**

Kontrollbefehl	**set_mark_speed_ctrl**
Funktion	Identisch mit **set_mark_speed**, aber ein Kontrollbefehl.
Aufruf	`set_mark_speed_ctrl( Speed )`
Parameter	`Speed` Markiergeschwindigkeit in *Bits pro ms* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [1,6 … 800.000,0].
Hinweise	• Der Befehl wird ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`), wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten). Er wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus: wie **set_mark_speed**
Verweise	**set_mark_speed**, **set_jump_speed**, **set_jump_speed_ctrl**

Kontrollbefehl	set_matrix
Funktion	Definiert die Koeffizienten der allgemeinen Transformationsmatrix M_T für nachfolgende Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 183.
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 *primären* Scan-Kopf-Anschluss. = 2: Die Definition wirkt nur auf den *sekundären* Scan-Kopf-Anschluss (Freischaltung erforderlich). = 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, M12, M21, M22` Matrixkoeffizienten der allgemeinen Transformationsmatrix M_T im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: [–50 … +50]. Bei unzulässigen Werten wird der Befehl ignoriert.
	`at_once` Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird: = 0: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten Listenbefehls berechnet und auf die dann aktuelle Position angewandt. = 1: Die neue Gesamttransformation wird sofort berechnet und auf die aktuelle Position angewandt (oder vor dem nächsten Listenbefehl, sofern gerade eine Liste BUSY oder die Karte INTERNAL-BUSY ist). Die "Laser active"-Lasersteuersignale werden gegebenenfalls zuvor ausgeschaltet. = 2: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten **jump_abs**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls berechnet und auf die dann aktuelle Position angewandt. = 3: Wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: Wie `at_once` = 2.
Hinweise	• Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 183. • Die mit `HeadNo` = 4 definierte Koordinatentransformation ist nur während einer mit **set_fly_2d** eingeleiteten "Processing on the fly"-Anwendung wirksam, siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208. Der Parameter `at_once` wird hier ignoriert.
RTC4→ RTC5	Unveränderte Grundfunktion (Definition der Transformationsmatrix), allerdings: • Die Parameter `HeadNo` und `at_once` sind neu. • Verringerter Wertebereich für die Koeffizienten. • Siehe auch "Hinweise für RTC4 Benutzer", Seite 186.
Versionsinfo	Letzte Änderungen: • Version DLL 525, OUT 527: Verhalten für `at_once` = 3. • Version DLL 536, OUT 536: `HeadNo` = 4.
Verweise	**set_matrix_list**, **set_angle**, **set_offset**, **set_scale**

Variabler Listenbefehl	**set_matrix_list**	
Funktion	Ändert *einen* der vier Koeffizienten der allgemeinen Transformationsmatrix M_T während der Ausführung einer Liste, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 183.	
Aufruf	`set_matrix_list( HeadNo, Ind1, Ind2, Mij, at_once )`	
Parameter	`HeadNo`	Siehe **set_matrix**.
	`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 im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: [–50 … +50]. Bei einem unzulässigen Wert wird der Befehl durch **list_nop** ersetzt.
	`at_once`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird: = 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 "Laser active"-Lasersteuersignale 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**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewandt. = 3: Wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: Wie `at_once` = 2.
Hinweise	• Der Befehl **set_matrix_list** erlaubt zu einem bestimmten Zeitpunkt nur die Änderung *eines* der vier Koeffizienten. Um mit dem Befehl während der Ausführung einer Liste mehrere Koeffizienten zu ändern, muss er 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 183.	
RTC4→ RTC5	Siehe **set_matrix**.	
Versionsinfo	Letzte Änderung Version DLL 525, OUT 527: Verhalten für `at_once` = 3.	
Verweise	**set_matrix**	

Kontrollbefehl	set_max_counts
Funktion	Definiert die maximale Zahl an externen Listen-Starts.
Aufruf	`set_max_counts( Counts )`
Parameter	`Counts` maximale Zahl an externen Listen-Starts als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [0 … (2^{32}–1)].
Hinweise	• Sobald die angegebene maximale Zahl an externen Listen-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 Listen-Starts unbegrenzt. • Bei der Initialisierung der RTC5 (**load_program_file**) wird `Counts` auf 0 gesetzt. • Die aktuelle Anzahl erfolgter externer Listen-Starts kann mit **get_counts** aus dem entsprechenden internen Zähler ausgelesen werden. Der Zähler kann mit **set_control_mode** rückgesetzt werden.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich).
Verweise	**get_counts**, **set_control_mode**

Kontrollbefehl	set_mcbsp_freq
Funktion	Legt die Sendefrequenz der McBSP/SPI-Schnittstelle fest.
Aufruf	`mcbsp_freq = set_mcbsp_freq( Freq )`
Parameter	`Freq` Gewünschte Sendefrequenz der McBSP/SPI-Schnittstelle in Hz als 32-Bit-Werte ohne Vorzeichen. Zulässiger Bereich: [4.000.000 … 16.000.000] (4 … 16 MHz). Überlaufende Werte werden auf den Randbereich geclippt. Bei überlaufenden Werten wird 0 zurückgegeben und ein **get_last_error**-Returncode `RTC5_PARAM_ERROR` erzeugt.
Rückgabe	Die tatsächlich eingestellte Frequenz in Hz als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Default-Wert der Sendefrequenz (nach Initialisierung) ist 8 MHz (`Freq` = 8.000.000). • Da nicht alle Frequenzen auch realisierbar sind, gibt der Befehl 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/SPI-Schnittstelle mit dem Befehl **mcbsp_init** neu initialisiert wird. • Die Signale und Betriebsbedingungen der McBSP/SPI-Schnittstelle sind in Kapitel 4.4.6 "SPI / I2C-Stiftleiste", Seite 54 dargestellt. • Hinweis: Die Empfangsfrequenz wird ausschließlich aus den ankommenden Clock-Pulsen bestimmt und darf maximal 16 MHz betragen.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	**mcbsp_init**, **set_mcbsp_out**, **set_mcbsp_out_ptr**

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/SPI-Schnittstelle an die RTC5 übergeben werden. Die McBSP/SPI-Schnittstelle kann zusätzlich zur Eingabe von beliebigen anderen Signalen verwendet werden.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_mcbsp_in( Mode, Scale )`
Parameter	`Mode` 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- und y-Richtung. = 4: Kompensation von Rotationsbewegungen. = 5: Die “Processing on the fly”-Korrektur wird abgeschaltet. • `Mode` = 0 … 5: Alle McBSP/SPI-Eingabewerte werden alternierend an die internen Speicherstellen 1 und 2 übertragen. • `Mode` = 1 … 5 (dagegen nicht für `Mode` = 0): Mit “Bit#31 = 0” kodierte McBSP/SPI-Eingabewerte werden zusätzlich an die interne Speicherstelle 0 übertragen und mit “Bit#31 = 1” kodierte McBSP/SPI-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 bzw. Rotationsauflösung im 64-Bit-IEEE-Gleitkommaformat: • `Mode` = 1 … 3: Skalierungsfaktor in *(RTC5-)Bits/(McBSP-)Bit.* *Zulässig*er Wertebereich: 1/256 ≤ |`Scale`| ≤ 16.000,0 (für `Mode` = 3 gilt `Scale` für beide Achsen). • `Mode` = 4: Zählschritte (Counts) per Umdrehung. Zulässiger Wertebereich: |`Scale`| > 100,0.
Hinweise	• Die internen Speicherstellen können jederzeit mit **read_mcbsp** ausgelesen werden. • Zur “Processing on the fly”-Korrektur und zur Ermittlung des Skalierungsfaktors siehe Kapitel 8.7 ““Processing on the fly” (optional)”, Seite 199. • Unterschiedliche “Processing on the fly”-Korrekturen sind nicht beliebig kombinierbar (siehe Seite 200). • Zur Deaktivierung der “Processing on the fly”-Korrektur siehe Seite 209. • Für zweidimensionale 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” kodierten McBSP/SPI-Eingabewertes. • Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für “Processing on the fly”-Anwendungen und für eine Online-Positionierung verwendet werden (siehe Seite 188). • Das jeweils erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP/SPI-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen (siehe Seite 56). • Für `Mode` > 5 wird **set_mcbsp_in** nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`). • Für unzulässige `Scale`-Werte (z. B. `Scale` = 0) wirkt **set_mcbsp_in** wie für `Mode` = 0.

Kontrollbefehl	set_mcbsp_in
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	set_mcbsp_in_list

Unverzögerter kurzer Listenbefehl	set_mcbsp_in_list
Funktion	Identisch mit set_mcbsp_in, aber ein Listenbefehl.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_mcbsp_in_list( Mode, Scale )`
Parameter	`Mode` Siehe set_mcbsp_in.
	`Scale` Siehe set_mcbsp_in.
Hinweise	• Für `Mode` > 5 wird **set_mcbsp_in_list** durch ein list_nop ersetzt (get_last_error-Return-code `RTC5_PARAM_ERROR`). • Siehe set_mcbsp_in.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	set_mcbsp_in

Kontrollbefehl	set_mcbsp_matrix
Funktion	Aktiviert eine Matrix-Korrektur für die Online-Positionierung über die McBSP/SPI-Schnittstelle.
Aufruf	`set_mcbsp_matrix()`
Hinweise	• Zur Online-Positionierung siehe Kapitel 8.3 "Online-Positionierung", Seite 187. • 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-Option. • Für die über die McBSP/SPI-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/SPI-Eingabewert M_{in} an die McBSP/SPI-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 RTC5 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** bzw. **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/SPI-Schnittstelle kann nicht gleichzeitig für die Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden (siehe Seite 188).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 533, OUT 534, RBF 524.
Verweise	**set_mcbsp_matrix_list**

Unverzögerter kurzer Listenbefehl	set_mcbsp_matrix_list
Funktion	Identisch mit **set_mcbsp_matrix**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_matrix_list()`
Hinweise	• Siehe **set_mcbsp_matrix**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 533, OUT 534, RBF 524.
Verweise	**set_mcbsp_matrix**

Unverzögerter kurzer Listenbefehl	set_mcbsp_out
Funktion	Definiert zwei Signaltypen für die Ausgabe an der McBSP/SPI-Schnittstelle (Multi channel Buffered Serial Port, siehe Kapitel 4.4.6 "SPI / I2C-Stiftleiste", Seite 54).
Aufruf	`set_mcbsp_out( Signal1, Signal2 )`
Parameter	`Signal1, Signal2` Gewünschte Signaltypen als 32-Bit-Werte ohne Vorzeichen.
Hinweise	• Die auswählbaren Signaltypen sind identisch zum Befehl **set_trigger** (für den zulässigen Wertebereich, die Signaltypen und entsprechende Hinweise siehe dort). Bei unzulässigen Angaben für `Signal1`/`Signal2` wird der Befehl durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Die beiden ausgewählten Datensignale werden permanent (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-Datenwortes. – Bei Sample-Werten und bei vom Scan-System rückübertragenen Statuswerten (`Signal1, Signal2` = 1-23, 25-30) werden nur die RTC4-kompatiblen Bits #4-19 ausgegeben. Die unteren 4 Bits sowie die aus dem virtuellen Bildfeld überlaufenden Bits werden ignoriert. – Bei den anderen Datentypen (`Signal1, Signal2` = 0, 24, 31-42) werden jeweils die unteren 16 Bit ausgegeben, die oberen Bits werden ignoriert (relevant nur für (`Signal1, Signal2` = 37-42 und gegebenfalls 24 und 31). • 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/SPI-Schnittstelle sind in Kapitel 4.4.6 "SPI / I2C-Stiftleiste", Seite 54 dargestellt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung (DLL 531, OUT 532): • Weitere auswählbare Datensignale (`Signal1, Signal2` = 25…42, siehe **set_trigger**). • Bei einigen der neuen Datensignale werden die unteren 16 Bit ausgegeben (s.o.).
Verweise	**read_mcbsp**, **mcbsp_init**, **set_mcbsp_freq**, **set_mcbsp_out_ptr**, **set_trigger**

Kontrollbefehl	**set_mcbsp_out_ptr**
Funktion	Definiert eine Liste von bis zu 8 Signaltypen für die Ausgabe an der McBSP/SPI-Schnittstelle (Multi channel Buffered Serial Port, siehe Kapitel 4.4.6 "SPI / I2C-Stiftleiste", Seite 54).
Aufruf	`set_mcbsp_out_ptr( Number, SignalPtr )`
Parameter	`Number` 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [0 … 8]. = 1…8: Anzahl der auszugebenden Signaltypen. = 0 Die Ausgabe an der McBSP/SPI-Schnittstelle erfolgt gemäß Voreinstellung bzw. gemäß Auswahl mit einem vorherigen **set_mcbsp_out**-Befehl.
	`SignalPtr` Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf ein Array von `Number` 32-Bit-Werten ohne Vorzeichen, in dem die gewünschten `Number` Signaltypnummern gespeichert sind.
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: `RTC5_PARAM_ERROR`). • Die auswählbaren Signaltypen sind identisch zum Befehl **set_trigger** (für den zulässigen Wertebereich, die Signaltypen und entsprechende Hinweise siehe dort). • Die bis zu 8 ausgewählten Datentypen werden zyklisch (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. • 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 37-42 und gegebenfalls 24 und 31): 32-Bit-Ausgabewert = (Signalwert << 8) | (Signaltypnummer & 0xFF). • Die Signale und Betriebsbedingungen der McBSP/SPI-Schnittstelle sind in Kapitel 4.4.6 "SPI / I2C-Stiftleiste", Seite 54 dargestellt.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	**set_mcbsp_out**, **set_trigger**

Kontrollbefehl	set_mcbsp_rot
Funktion	Aktiviert oder deaktiviert eine Rotations-Korrektur für die Online-Positionierung über die McBSP/SPI-Schnittstelle.
Aufruf	`set_mcbsp_rot( Resolution )`
Parameter	`Resolution` Angabe im 64-Bit IEEE Gleitkommaformat: 2^{-26} < `Resolution` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert), sonst: Korrektur wird deaktiviert. `Resolution` = McBSP/SPI-Bits pro Vollkreis.
Hinweise	• Zur Online-Positionierung siehe Seite 187. • Bei einem McBSP/SPI-Eingabewert Rot_{in} für die Rotations-Korrektur wirkt **apply_mcbsp** wie `set_angle( …, Angle × 360°, … )` mit (VK = Vollkreis) `Angle` (in VK) = Rot_{in} / `Resolution` Es sind nur `Angle`-Werte im Bereich [0,0…+20,0 VK] zulässig. • Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für die Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden (siehe Seite 188).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
Verweise	**set_mcbsp_rot_list**, **set_mcbsp_x**, **set_mcbsp_y**, **apply_mcbsp**

Unverzögerter kurzer Listenbefehl	set_mcbsp_rot_list
Funktion	Identisch mit **set_mcbsp_rot**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_rot_list( Resolution )`
Parameter	`Resolution` Siehe **set_mcbsp_rot**.
Hinweise	• Siehe **set_mcbsp_rot**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
Verweise	**set_mcbsp_rot**

Kontrollbefehl	set_mcbsp_x
Funktion	Aktiviert oder deaktiviert eine X-Offset-Korrektur für die Online-Positionierung über die McBSP/SPI-Schnittstelle.
Aufruf	`set_mcbsp_x( Scale )`
Parameter	`Scale` Angabe im 64-Bit IEEE Gleitkommaformat: 2^{-26} < `Scale` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert), sonst: Korrektur wird deaktiviert.
Hinweise	• Zur Online-Positionierung siehe Seite 187. • Bei einem McBSP/SPI-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, sofern `Scale` × X_{in} den Wertebereich $\pm 2^{31}$ nicht überschreitet. • Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für die Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden (siehe Seite 188).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
Verweise	**set_mcbsp_x_list**, **set_mcbsp_y**, **set_mcbsp_rot**, **apply_mcbsp**

Unverzögerter kurzer Listenbefehl	set_mcbsp_x_list
Funktion	Identisch mit **set_mcbsp_x**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_x_list( Scale )`
Parameter	`Scale` Siehe **set_mcbsp_x**.
Hinweise	• Siehe **set_mcbsp_x**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
Verweise	**set_mcbsp_x**

Kontrollbefehl	set_mcbsp_y
Funktion	Aktiviert oder deaktiviert eine Y-Offset-Korrektur für die Online-Positionierung über die McBSP/SPI-Schnittstelle.
Aufruf	`set_mcbsp_y( Scale )`
Parameter	`Scale` Angabe im 64-Bit IEEE Gleitkommaformat: 2^{-26} < `Scale` < 2^{26}: Skalierungsfaktor (Korrektur wird aktiviert), sonst: Korrektur wird deaktiviert.
Hinweise	• Zur Online-Positionierung siehe Kapitel 8.3 "Online-Positionierung", Seite 187. • Bei einem McBSP/SPI-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, sofern `Scale` × Y_{in} den Wertebereich $\pm 2^{31}$ nicht überschreitet. • Die McBSP/SPI-Schnittstelle kann nicht gleichzeitig für die Online-Positionierung und für "Processing on the fly"-Anwendungen verwendet werden (siehe Hinweise Seite 188).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
Verweise	**set_mcbsp_y_list**, **set_mcbsp_x**, **set_mcbsp_rot**, **apply_mcbsp**

Unverzögerter kurzer Listenbefehl	set_mcbsp_y_list
Funktion	Identisch mit **set_mcbsp_y**, aber ein Listenbefehl.
Aufruf	`set_mcbsp_y_list( Scale )`
Parameter	`Scale` Siehe **set_mcbsp_y**.
Hinweise	• Siehe **set_mcbsp_y**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.
Verweise	**set_mcbsp_y**

Kontrollbefehl	set_multi_mcbsp_in
Funktion	Aktiviert eine “Processing on the fly”-Anwendung und eine Mehrfach-Übertragung über die McBSP/SPI-Schnittstelle mit bis zu acht verschiedenen Datentypen.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl 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 `RTC5_PARAM_ERROR`). `P` Initialisierungswert für die Laserleistung als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: siehe **set_auto_laser_control**. `Mode` 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**.
Hinweise	• Jede andere evtl. zuvor aktivierte McBSP/SPI-Übertragung für eine Online-Positionierung und jede andere zuvor aktivierte “Processing on the fly”-Anwendung mit Encoder-Signalen oder mit Positionsangaben wird zuvor beendet. • Der Befehl aktiviert eine Eingabemöglichkeit über die McBSP/SPI-Schnittstelle, bei der bis zu acht verschiedene Typen asynchron zum 10 µs-Takt übertragen werden können. Die Daten werden alle 10 µs gemäß ihrer Typ-Kodierung 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 des Befehls keine McBSP/SPI-Übertragung mehr aktiv sein. • Das jeweils erste FrameSync-Signal nach einem **load_program_file** oder **mcbsp_init** wird von der McBSP/SPI-Schnittstelle ignoriert, die evtl. bereitgestellten Daten werden also nicht übertragen (siehe Seite 56). • Der Benutzer muss die Typ-Zuordnung für die Übertragung in die drei niederwertigsten Bits des Datenwortes gemäß der Formel McBSPValue = (Value << 3) | Type kodieren. Die RTC5 speichert das Datenwort gemäß der Formel 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. Ggf. gibt es ein Clipping oder einen Datenüberlauf. • Die vier Extraparameter sind nicht mit den (ehem. vier, jetzt acht) freien Variablen identisch (siehe Kapitel 6.9.1 “Freie Variablen”, Seite 102, 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.

Kontrollbefehl	set_multi_mcbsp_in
Hinweise (Forts.)	• Bei mehr als vier abgeschlossenen McBSP/SPI-Übertragungen innerhalb eines 10 µs-Taktes können vorherige Werte überschrieben werden. • Sofern die "Processing on the fly"-Option freigeschaltet ist, aktiviert der Befehl 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. Auch Z wird in 20-Bit-Auflösung verwendet. • 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", jedoch ohne den Softstart zu unterbinden. • Diese Methode der Laserleistungsvariation kann mit der vektorkontrollierten Lasersteuerung (mit parametrisierten Befehlen) kombiniert werden, jedoch nicht mit anderen Methoden der automatischen Lasersteuerung. Sie überschreibt die anderen Variationen, sofern 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/SPI-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.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_multi_mcbsp_in_list**, **read_multi_mcbsp**, **set_wobbel_control**, **set_wobbel_vector**

Normaler Listenbefehl	**set_multi_mcbsp_in_list**
Funktion	Identisch mit **set_multi_mcbsp_in**, aber ein Listenbefehl.
Einschränkung	Falls die “Processing on the fly”-Option nicht freigeschaltet ist, schaltet der Befehl den “Processing on the fly”-Prozess aus (auch wenn er nie angewesen sein kann).
Aufruf	`set_multi_mcbsp_in_list( Ctrl, P, Mode )`
Parameter	`Ctrl`, `P` Siehe **set_multi_mcbsp_in**.
Hinweise	• Siehe **set_multi_mcbsp_in**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_multi_mcbsp_in**, **read_multi_mcbsp**, **set_wobbel_control**, **set_wobbel_vector**

Variabler Listenbefehl	**set_n_pixel**	
Funktion	Definiert im Pixelausgabemodus die Lasersteuerungsparameter (Pulslänge und Analogspannungswert) für mehrere unmittelbar aufeinander folgende identische Pixel einer Bildzeile.	
Aufruf	`set_n_pixel( PulseLength, AnalogOut, Number )`	
Parameter	`PulseLength`	Länge des Pixelpulses (als 32-Bit-Wert ohne Vorzeichen). *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
	`AnalogOut`	12-Bit-Ausgabewert (Analogspannungswert am mit **set_pixel_line** ausgewählten Analogausgang) für das Pixel als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert (siehe auch **write_da_x**).
	`Number`	Anzahl der Pixel als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [1 … (2^{32}–1)]. 0 wird automatisch auf 1 geclippt.
Hinweise	• Zur Verwendung des Befehls siehe auch Kapitel 8.8 "Pixelausgabemodus – Rasterbilder (Bitmaps) scannen", Seite 217. • Vor dem ersten **set_n_pixel**-Befehl einer Zeile muss der Befehl **set_pixel_line** erteilt worden sein. • Der **set_n_pixel**-Befehl definiert die Parameter für die nachfolgenden `Number` (identische) Pixel einer Bildzeile. Jeder **set_n_pixel**-Befehl muss direkt auf den Befehl **set_pixel_line** oder auf einen anderen **set_n_pixel**-Befehl folgen. Bevor die Bildzeile komplett ist, darf kein anderer Befehl in die Liste eingetragen werden. Der erste Befehl einer Liste, der auf den **set_pixel_line**-Befehl folgt und *kein* **set_n_pixel**-Befehl ist, deaktiviert den Pixelausgabemodus. • Soll nur ein einzelner Pixel definiert werden, kann alternativ zu **set_n_pixel** auch der **set_pixel**-Befehl verwendet werden. Dieser ist synonym zu **set_n_pixel** mit Parameter `Number` = 1. • Außerhalb des Pixelausgabemodus (wenn dem Befehl kein **set_pixel_line** oder **set_n_pixel** unmittelbar vorausgegangen ist) ist **set_n_pixel** ein kurzer Listenbefehl und wird ansonsten ignoriert. Unter Umständen kann ein **list_nop** eingefügt werden (siehe Seite 250). • Am Ende der Bildzeile (zu Beginn des Default-Pixels) werden die Pixelpulslänge `PulseLength` und der Analogwert `AnalogOut` jeweils auf einen Default-Wert gesetzt. Der Default-Pixelpulslängen-Wert ist 0, wenn nicht zuvor mit **set_default_pixel** oder **set_default_pixel_list** ein anderer Wert festgelegt wurde. Der Default-Analogwert ist 0xFFF (=4095), sofern nicht zuvor mit **set_port_default** oder **set_laser_off_default** ein anderer Wert festgelegt wurde.	
RTC4→ RTC5	Neuer Befehl. Zum RTC4-Kompatibilitätsmodus: siehe **set_pixel**.	
Verweise	**set_pixel**, **set_pixel_line**, **set_pixel_line_3d**	

Kontrollbefehl	set_offset
Funktion	Definiert einen Offset für nachfolgende Koordinatentransformationen (siehe "Koordinatentransformationen", Seite 183).
Aufruf	`set_offset( HeadNo, XOffset, YOffset, at_once )`
Parameter	`HeadNo` Siehe **set_offset_xyz**.
	`XOffset, YOffset` Siehe **set_offset_xyz**.
	`at_once` Siehe **set_offset_xyz**.
Hinweise	• Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 183. • Der Befehl wirkt wie **set_offset_xyz**, lässt jedoch `ZOffset` unverändert.
RTC4→ RTC5	unveränderte Grundfunktion (Offset-Definition), allerdings: • Die Parameter `HeadNo` und `at_once` sind neu. • Erweiterter Wertebereich für die Offset-Werte. • Siehe auch "Hinweise für RTC4 Benutzer", Seite 186. Die Bildfeldkoordinaten für die X- und Y-Achse werden bei der RTC5 als 20-Bit-Werte angegeben. Im RTC4-Kompatibilitätsmodus dagegen werden die wie bei der RTC4 als 16-Bit-Werte angegebenen Offsetwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–32.768 … +32.767]).
Versionsinfo	Letzte Änderungen: • Version DLL 525, OUT 527: Verhalten für `at_once` = 3. • Version DLL 536, OUT 536: `HeadNo` = 4.
Verweise	**set_offset_list**, **set_offset_xyz**, **set_angle**, **set_matrix**, **set_scale**

Variabler Listenbefehl	set_offset_list
Funktion	Identisch mit **set_offset**, aber ein Listenbefehl.
Aufruf	`set_offset_list( HeadNo, XOffset, YOffset, at_once )`
Parameter	`HeadNo` Siehe **set_offset_xyz**.
	`XOffset, YOffset` Siehe **set_offset_xyz**.
	`at_once` Siehe **set_offset_xyz_list**.
RTC4→ RTC5	Siehe **set_offset_xyz**.
Versionsinfo	Letzte Änderung Version DLL 525, OUT 527: Verhalten für `at_once` = 3.
Verweise	**set_offset**, **set_offset_xyz_list**

Kontrollbefehl	set_offset_xyz	
Funktion	Definiert einen Offset für nachfolgende Koordinatentransformationen, siehe Kapitel 8.2 "Koordinatentransformationen", Seite 183.	
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 *primären* Scan-Kopf-Anschluss. = 2: Die Definition wirkt nur auf den *sekundären* Scan-Kopf-Anschluss (Freischaltung erforderlich). = 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.
	`XOffset`, `YOffset`	Offset für die x- und y-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 den Randwert geclippt.
	`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: [–32.768 … +32.767]. Überlaufende Werte werden auf den Randwert geclippt.
	`at_once`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird: = 0: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten Listenbefehls berechnet und auf die dann aktuelle Position angewandt. = 1: Die neue Gesamttransformation wird sofort berechnet und auf die aktuelle Position angewandt (oder vor dem nächsten Listenbefehl, sofern gerade eine Liste BUSY oder die Karte INTERNAL-BUSY ist). Die "Laser active"-Lasersteuersignale werden gegebenenfalls zuvor ausgeschaltet. = 2: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten **jump_abs**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls berechnet und auf die dann aktuelle Position angewandt. = 3: wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: wie `at_once` = 2.
Hinweise	• Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 183. • `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) beispielsweise für den Ansteuerwert `(0|0|0)` um d = `ZOffset` / K_z in die Ebene z = d in Richtung der z-Koordinatenachse (und entgegen einer mit **set_defocus** festgelegten Fokusverschiebung). Dabei ist K_z der Kalibrierungsfaktor für die z-Richtung (siehe auch Kapitel 8.6.3 "3D-Markierbefehle", Seite 194). • Die mit `HeadNo` = 4 definierte Koordinatentransformation ist nur während einer mit **set_fly_2d** eingeleiteten "Processing on the fly"-Anwendung wirksam (siehe Abschnitt "Koordinatentransformationen im virtuellen Bildfeld", Seite 208). Der Parameter `at_once` wird hier ignoriert. Für `HeadNo` = 4 wird Parameter `ZOffset` ignoriert.	

Kontrollbefehl	set_offset_xyz
RTC4→ RTC5	Neuer Befehl. Die Bildfeldkoordinaten für die X- und Y-Achse werden bei der RTC5 als 20-Bit-Werte angegeben. Im RTC4-Kompatibilitätsmodus dagegen werden die wie bei der RTC4 als 16-Bit-Werte angegebenen Offsetwerte für die X- und Y-Achse (nicht für die Z-Achse) von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–32.768 … +32.767]).
Versionsinfo	Letzte Änderungen: • Version DLL 525, OUT 527: Verhalten für `at_once` = 3. • Version DLL 536, OUT 536: `HeadNo` = 4.
Verweise	**set_offset_xyz_list**, **set_offset**, **set_angle**, **set_matrix**, **set_scale**, **set_defocus**

Variabler Listenbefehl	set_offset_xyz_list	
Funktion	Identisch mit **set_offset_xyz**, aber ein Listenbefehl.	
Aufruf	`set_offset_xyz_list( HeadNo, XOffset, YOffset, ZOffset, at_once )`	
Parameter	`HeadNo`	Siehe **set_offset_xyz**.
	`XOffset`, `YOffset`, `ZOffset`	Siehe **set_offset_xyz**.
	`at_once`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird: = 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 "Laser active"-Lasersteuersignale 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**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewandt. = 3: wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: wie `at_once` = 2
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Letzte Änderung Version DLL 525, OUT 527: Verhalten für `at_once` = 3.	
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`.
Hinweise	• Wenn gerade eine Liste ausgeführt wird und die folgende Bedingung (identisch zu anderen konditionalen Befehlen, siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244) erfüllt ist: `((IOvalue AND Mask1) = Mask1) AND (((not IOvalue) AND Mask0) = Mask0)` (d.h. 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 gleichzeitig anliegendes /STOP-Signal hat Vorrang vor dem bedingten **pause_list**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 543, OUT 543.
Verweise	**pause_list**, **restart_list**

Variabler Listenbefehl	**set_pixel**
Funktion	Definiert im Pixelausgabemodus die Lasersteuerungsparameter (Pulslänge und Analogspannungswert) eines einzelnen Pixels einer Bildzeile.
Aufruf	`set_pixel( PulseLength, AnalogOut )`
Parameter	`PulseLength` Länge des Pixelpulses (als 32-Bit-Wert ohne Vorzeichen). *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
	`AnalogOut` 12-Bit-Ausgabewert (Analogspannungswert am mit **set_pixel_line** ausgewählten Analogausgang) für das Pixel als 32-Bit-Wert ohne Vorzeichen. Höherwertige Bits werden ignoriert (siehe auch **write_da_x**).
Hinweise	• Der Befehl ist synonym zu **set_n_pixel** mit Parameter `Number` = 1 (siehe Hinweise dort).
RTC4→ RTC5	• Das Einlesen analoger Spannungen (siehe RTC4-Parameter `ADChannel` und RTC4-Befehl **read_pixel_ad**) wird von der RTC5 *nicht* unterstützt. • Die Pixelpulse (am LASER1-Ausgang) werden bei der RTC5 direkt über den Parameter `PulseLength` gesteuert, nicht mehr wie bei der RTC4-Karte über das LASERON-Signal. • `PulseLength` wird bei der RTC5 in Einheiten von 1/64 µs angegeben. Im RTC4-Kompatibilitätsmodus werden die wie bei der RTC4 in Einheiten von 1/8 µs angegebenen Werte von der RTC5 auf 1/64 µs-Einheiten umgerechnet (d.h. mit 8 multipliziert). Außerdem wird `AnalogOut` mit 4 multipliziert (bei der RTC4 waren die Analogausgänge nur für 10 Bit-Werte ausgelegt). Die zulässigen Wertebereiche für die Parameter verringern sich entsprechend.
Verweise	**set_n_pixel**, **set_pixel_line**, **set_pixel_line_3d**

Normaler Listenbefehl	**set_pixel_line**	
Funktion	Aktiviert den Pixelausgabemodus und definiert verschiedene Pixelausgabeparameter.	
Aufruf	`set_pixel_line( Channel, HalfPeriod, dX, dY )`	
Parameter	`Channel`	Nummer des Analogausgangs (als 32-Bit-Wert ohne Vorzeichen), an welchem die mit nachfolgenden **set_pixel**/**set_n_pixel**-Befehlen festgelegten Analogspannungswerte ausgegeben werden. Zulässiger Bereich: [1, 2] (1: ANALOG OUT1, 2: ANALOG OUT2).
	`HalfPeriod`	*Halbe* Pixelausgabeperiode in *Bits* als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [104 … (2^{32}–1)].
	`dX, dY`	Abstand (in x- und y-Richtung) zwischen zwei aufeinanderfolgenden Pixeln in *Bits* (im 64-Bit-IEEE-Gleitkommaformat).
Hinweise	• Jede Bildzeile eines Rasterbildes muss mit dem Befehl **set_pixel_line** begonnen werden. Dem Befehl sollte jeweils ein Jump- oder Mark-Befehl auf den jeweiligen Startpunkt der Bildzeile vorausgehen. • Unmittelbar nach dem Befehl **set_pixel_line** folgt die gewünschte Anzahl von **set_pixel**/**set_n_pixel**-Befehlen, welche die Parameter `PulseLength` und `AnalogOut` übertragen. Der erste Befehl einer Liste, der auf den **set_pixel_line**-Befehl folgt und *kein* **set_pixel**/**set_n_pixel**-Befehl ist, beendet den Pixelausgabemodus (auch der Befehl **set_pixel_line** beendet den Pixelausgabemodus ehe er ihn wieder startet). Dabei wird ein Default-Pixel eingefügt. • Für `Channel` < 1 oder `Channel` > 2 wird **set_pixel_line** durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_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 Seite 217). `HalfPeriod` darf nicht kleiner als 104 sein (wird andernfalls automatisch geclippt). Dies entspricht einer Pixelfrequenz von ca. 308 kHz. • Die DA-Wandlung wird für Pixelfrequenzen über ca. 100 kHz (d.h. für `HalfPeriod` < ca. 320) nicht in jedem Fall vollständig abgeschlossen. Bei solchen Pixelfrequenzen muss die ausreichende Funktionsfähigkeit von Benutzern sorgfältig überprüft werden. • Der Pixelausgabemodus ist inkompatibel mit dem Softstart-Modus (siehe Seite 152). • Der Pixelausgabemodus sollte nicht zusammen mit der automatischen Lasersteuerung (siehe Seite 159) verwendet werden, wenn dort die 12-Bit-Ausgabewerte am ANALOG OUT1- oder ANALOG OUT2- Ausgang, die Pulslängen (`PulseLength`) oder Ausgabeperioden (`HalfPeriod`) der Lasersignale LASER1 und LASER2 angepasst werden. • Der Pixelausgabemodus kann mit "Processing on the fly" (siehe Seite 199) kombiniert werden. • Der Pixelausgabemodus kann *nicht* mit Sky-Writing (siehe Seite 127) oder Wobbel (siehe Seite 189) kombiniert werden. • Siehe auch Kapitel 8.8 "Pixelausgabemodus – Rasterbilder (Bitmaps) scannen", Seite 217.	

Normaler Listenbefehl	**set_pixel_line**
RTC4→ RTC5	• Die RTC5 stellt nur *einen* Pixelmodus zur Verfügung (dieser entspricht RTC4-`PixelMode` = 1) . • Bei der RTC5 kann der Ausgabekanal `Channel` für die Analogsignale ausgewählt werden (bei der RTC4 konnten im Pixelausgabemodus Analogspannungswerte nur am ANALOG OUT2-Ausgang ausgegeben werden). • Die Pixelausgabeperiode wird bei der RTC5 als *halbe* Ausgabeperiode `HalfPeriod` angegeben (bei der RTC4 dagegen als *volle* Ausgabeperiode `PixelPeriod`). • Der Befehl **set_pixel_line** belegt bei der RTC5 wie andere normale Listenbefehle nur *einen* Listenspeicherplatz (bei der RTC4 dagegen *zwei* Speicherplätze). • Im RTC4-Kompatibilitätsmodus muss `HalfPeriod` in 1/8 µs angegeben werden. Die Parameter-Werte werden dann intern auf 1/64 µs-Einheiten umgerechnet (d.h. mit 8 multipliziert). Der zulässige Wertebereich verringert sich entsprechend (der kleinste zulässige Wert ist dann 13). Die angegebenen Koordinatenwerte `dX,dY` werden im RTC4-Kompatibilitätsmodus von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	Letzte Änderung Version OUT 515.
Verweise	**set_pixel**, **set_n_pixel**, **set_pixel_line_3d**

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`	Siehe **set_pixel_line** (Nummer des Analogausgangs).
	`HalfPeriod`	Siehe **set_pixel_line** (*halbe* Pixelausgabeperiode).
	`dX, dY`	Siehe **set_pixel_line**: Abstand in x- und y-Richtung zwischen zwei aufeinanderfolgenden Pixeln in *Bits* (im 64-Bit-IEEE-Gleitkommaformat)
	`dZ`	Abstand in z-Richtung zwischen zwei aufeinanderfolgenden Pixeln in *Bits* (im 64-Bit-IEEE-Gleitkommaformat). Wie alle Z-Koordinaten hat auch `dZ` eine 16-fach kleinere Skalierung als `dX` und `dY`.
Hinweise	• Anders als **set_pixel_line** kann mit **set_pixel_line_3d** der Pixelabstand zwischen zwei benachbarten Bildpunkten in einer Zeile über einen *3D*-Vektor definiert werden. • Anders als **set_pixel_line** benötigt **set_pixel_line_3d** außerdem *zwei* Listenspeicherplätze, sofern der Parameter `dZ` ungleich 0 ist. Der erste Teil wird als unverzögerter kurzer Listenbefehl vor dem Hauptteil (normaler Listenbefehl) ausgeführt. Ein evtl. noch anliegender verzögerter kurzer Listenbefehl wird noch davor ausgeführt. • Ansonsten wirkt **set_pixel_line_3d** wie **set_pixel_line** (siehe Hinweise dort).	
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_pixel_line**	
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.	
Verweise	**set_pixel_line**, **set_pixel**, **set_n_pixel**	

Kontrollbefehl	**set_port_default**
Funktion	Definiert den Default-Ausgabewert für den ausgewählten Ausgabe-Port.
Aufruf	`set_port_default( Port, Value )`
Parameter	`Port` Auswahl des Ausgabe-Ports, für den ein Defaultwert `Value` definiert werden soll als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: ANALOG OUT1-Ausgang (siehe auch Seite 48) = 1: ANALOG OUT2-Ausgang (siehe auch Seite 48) = 2: 8-Bit-Digital-Ausgang (siehe auch Seite 52) = 3: 16-Bit-Digital-Ausgang (siehe auch Seite 51) = 4: 2-Bit-Digital-Ausgang (siehe auch Seite 48)
	`Value` Gewünschter Defaultwert als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: für `Port` = 0/1: 12-Bit-Werte [0 … 4095], höherwertige Bits werden ignoriert für `Port` = 2: 8-Bit-Werte [0 … 255], höherwertige Bits werden ignoriert für `Port` = 3: 16-Bit-Werte [0 … (2^{16}–1)], 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).
Hinweise	• Bei der Initialisierung der RTC5 (mit **load_program_file**) werden die Defaultwerte für alle Ausgabe-Ports auf "–1" (= 2^{32}–1) gesetzt. • Wenn für einen Ausgabe-Port ein Defaultwert ≠ "–1" festgelegt wurde, dann wird der Ausgabe-Port auf diesen Defaultwert gesetzt, sobald das Abarbeiten einer Liste mit **stop_execution** oder durch ein externes Stoppsignal beendet wird. Bei einem Default-wert von "–1" wird der entsprechende Ausgabe-Port *nicht* beschrieben (der evtl. noch hochohmige Zustand bei den Digital-Ausgängen bleibt erhalten). • Defaultwerte (≠ "–1") für die Ausgabe-Ports 0/1 (ANALOG OUT1 oder ANALOG OUT2) werden auch am Ende des Pixelausgabemodus gesetzt (siehe Kapitel 8.8 "Pixelausga-bemodus – Rasterbilder (Bitmaps) scannen", Seite 217). • Nach erfolgter Initialisierung der positions- und/oder geschwindigkeitsabhängigen Lasersteuerung mit **set_auto_laser_control** wird der entsprechende Defaultwert (für den Ausgabe-Port 0, 1, 2 oder 3, je nach gewähltem Lasersignalparameter `Ctrl`) auch jedesmal dann ausgegeben, wenn der Laser am Ende einer Beschriftung ausgeschaltet wird oder wenn die positions- und/oder geschwindigkeitsabhängige Lasersteuerung mit **set_auto_laser_control** auf einen anderen `Ctrl`-Parameter gesetzt oder mit **set_auto_laser_control** (`Ctrl` = 0) deaktiviert wird (siehe Seite 161). Steht hierbei der Defaultwert auf "–1", so wird der maximal zulässige Wert (4095, 4095, 255 bzw. 65535) ausgegeben. • Für einen unzulässigen Wert von `Port` wird **set_port_default** nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`). • Die Defaultwerte für die Ausgabe-Ports 0-2 können auch mit dem Befehl **set_laser_off_default** festgelegt werden.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus wird `Value` für `Port` = 0/1 mit 4 multipliziert (bei der RTC4 waren die Analogausgänge nur für 10 Bit-Werte ausgelegt).
Verweise	**set_laser_off_default**, **set_default_pixel**

Kontrollbefehl	set_pulse_picking
Funktion	Schaltet den Pulse-Picking-Lasermodus ein.
Aufruf	`set_pulse_picking( No )`
Parameter	`No` 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … 63]. = 0: auf LASER2 wird das LASERON-Signal ausgegeben. = 1…63: auf LASER2 jeder `No`-te LASER1-Puls ausgegeben. Für `No` > 63 wird `No` auf 63 geclippt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).
Hinweise	• zum Pulse-Picking-Lasermodus siehe Kapitel 7.4.8 "Pulse-Picking-Lasermodus", Seite 157. • Die Pulse-Picking-Signale werden ausgegeben, bis mit **set_laser_mode** eine anderer Lasermodus eingestellt wird (der Pulse-Picking-Lasermodus wird durch Einschalten eines jeden anderen Laser-Modus mit **set_laser_mode** ausgeschaltet). • Der Pulse-Picking-Lasermodus kann *nicht* mit **set_laser_mode** eingeschaltet werden.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 529, OUT 530, RBF 521.
Verweise	**set_laser_mode**, **set_pulse_picking_list**

Kontrollbefehl	set_pulse_picking_length
Funktion	Definiert eine konstante Pulslänge für das "Laser active"-LASER2-Signal im Pulse-Picking-Lasermodus.
Aufruf	`set_pulse_picking_length( Length )`
Parameter	`Length` Pulslänge als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … 65535]. Höherwertige Bits werden ignoriert. *1 Bit entspricht 1/64 µs*. Die Standard-Einstellung nach **load_program_file** ist 0.
Hinweise	• Damit der Wert wirksam ist, müssen der Pulse-Picking-Lasermodus mit **set_pulse_picking** und der "konstante Pulslänge"-Modus mit **set_laser_control**(Bit #7 = 1) aktiviert sein. Der Wert ist dann sofort wirksam, auch während einer aktiven Markierung. • Ist zwar der Pulse-Picking-Lasermodus aktiviert, jedoch nicht der "konstante Pulslänge"-Modus (**set_laser_control**(Bit #7 = 0)), übernimmt das Pulse-Picking-Signal die Pulslänge des LASER1-Signals (siehe Pulse-Picking-Lasermodus Seite 157). • Solange weder der Pulse-Picking-Lasermodus 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 stets das LASERON-Signal ausgegeben, kein "konstante Pulslänge"-Signal.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 533, OUT 534, RBF 524.

Unverzögerter kurzer Listenbefehl	**set_pulse_picking_list**
Funktion	Identisch mit **set_pulse_picking**, aber ein Listenbefehl.
Aufruf	`set_pulse_picking_list( No )`
Parameter	`No` Siehe **set_pulse_picking**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 529, OUT 530, RBF 521.
Verweise	**set_pulse_picking**

Kontrollbefehl	**set_qswitch_delay**
Funktion	Legt in den YAG-Versionen die Länge des Delays des ersten Q-Switch-Pulses bzgl. des FirstPulseKiller-Signals fest (siehe auch Abb. 51).
Aufruf	`set_qswitch_delay( Delay )`
Parameter	`Delay` Q-Switch-Delay als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0…(2^{26}–1)].
Hinweise	• Werte über (2^{26}–1) werden geclippt. • Die YAG-Versionen werden mit **set_laser_mode** ([1, 2, 3 oder 5]) ausgewählt. • Auch bei einer mit **set_laser_mode**([1-3]) eingestellten YAG-Version kann die Länge des Q-Switch-Delays mit diesem Befehl auch noch nachträglich geändert werden, bei YAG-Version 2 auch mit **set_firstpulse_killer** oder **set_firstpulse_killer_list**.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus muss der Delay-Wert in 1/8 µs angegeben werden. Er wird dann intern auf 1/64 µs-Einheiten umgerechnet (d.h. mit 8 multipliziert). Der zulässige Wertebereiche verringert sich entsprechend.
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	Identisch mit **set_qswitch_delay**, aber ein Listenbefehl.
Aufruf	`set_qswitch_delay_list( Delay )`
Parameter	`Delay` Q-Switch-Delay als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0…(2^{26}–1)].
RTC4→ RTC5	Neuer Befehl.
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, Y` Lage des Rotationszentrums bezogen auf den Mittelpunkt (0,0) des Bildfeldes als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [-2^{24} … ($2^{24}-1$)].
Hinweise	• Zur "Processing on the fly"-Korrektur siehe Kapitel 8.7.3 "Kompensation von Rotationsbewegungen", Seite 204. • 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 Bildfeldes liegen (der zulässige Bereich entspricht etwa dem 30-fachen Bildfeld). • Ein zweiter Kopf kann sinnvoll nur verwendet werden, wenn er exakt auf dasselbe Rotationszentrum justiert ist.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Verweise	**set_fly_rot**, **set_fly_rot_pos**

Verzögerter kurzer Listenbefehl	set_rot_center_list
Funktion	Identisch mit **set_rot_center**, aber ein Listenbefehl.
Aufruf	`set_rot_center_list( X, Y )`
Parameter	`X, Y` Siehe **set_rot_center**.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_rot_center**

Kontrollbefehl	set_rtc4_mode
Funktion	Setzt den RTC4-Kompatibilitätsmodus als aktuellen DLL-Betriebsmodus.
Aufruf	`set_rtc4_mode()`
Hinweise	• Der RTC4-Kompatibilitätsmodus ist ein optionaler Betriebsmodus der DLL. Er wird zur Verfügung gestellt, damit Anwenderprogramme, die für die RTC4 geschrieben sind, auch von der RTC5 (weitgehend) ohne Programmcode-Änderung abgearbeitet werden können. Voraussetzung hierbei ist allerdings, dass der Programmcode nur RTC4-Befehle enthält, die mit unveränderter Funktionalität auch als RTC5-Befehle existieren. In der Befehlsauflistung dieses Handbuchs wird gegebenenfalls jeweils in der Zeile "RTC4→ RTC5" auf entsprechende Änderungen hingewiesen. • Standardmäßig ist der RTC5-Modus als DLL-Betriebsmodus voreingestellt und kann auch nachträglich wieder mit **set_rtc5_mode** eingestellt werden. • Der aktuelle DLL-Betriebsmodus kann mit **get_rtc_mode** ausgelesen werden. • Der Befehl **set_rtc4_mode** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **set_rtc4_mode** ist nicht als Multi-Board-Befehl verfügbar. • Der Befehl wirkt nicht kartenspezifisch, sondern global für die DLL und damit auf alle RTC5-Karten, für die das Anwenderprogramm zugriffsberechtigt ist. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **set_rtc4_mode** weder erzeugt noch verändert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_rtc_mode**, **set_rtc5_mode**

Kontrollbefehl	set_rtc5_mode
Funktion	Setzt den RTC5-Modus als aktuellen DLL-Betriebsmodus (Standardeinstellung).
Aufruf	`set_rtc5_mode()`
Hinweise	• Mit **set_rtc4_mode** kann der RTC4-Kompatibilitätsmodus als DLL-Betriebsmodus eingestellt werden. • Der aktuelle DLL-Betriebsmodus kann mit **get_rtc_mode** ausgelesen werden. • Der Befehl **set_rtc5_mode** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **set_rtc5_mode** ist nicht als Multi-Board-Befehl verfügbar. • Der Befehl wirkt nicht kartenspezifisch, sondern global für die DLL und damit auf alle RTC5-Karten, für die das Anwenderprogramm zugriffsberechtigt ist. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **set_rtc5_mode** weder erzeugt noch verändert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**get_rtc_mode**, **set_rtc4_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 183.	
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 *primären* Scan-Kopf-Anschluss. = 2: Die Definition wirkt nur auf den *sekundären* Scan-Kopf-Anschluss (Freischaltung erforderlich). = 0, 3: Die Definition wirkt auf *beide* Scan-Kopf-Anschlüsse. Es werden nur die beiden niederwertigsten Bits ausgewertet.
	`Scale`	Skalierungsfaktor 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`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird: = 0: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten Listenbefehls berechnet und auf die dann aktuelle Position angewandt. = 1: Die neue Gesamttransformation wird sofort berechnet und auf die aktuelle Position angewandt (oder vor dem nächsten Listenbefehl, sofern gerade eine Liste BUSY oder die Karte INTERNAL-BUSY ist). Die "Laser active"-Lasersteuersignale werden gegebenenfalls zuvor ausgeschaltet. = 2: Die neue Gesamttransformation (Gesamtmatrix und Offset) wird erst bei der Ausführung des nächsten **jump_abs**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls berechnet und auf die dann aktuelle Position angewandt. = 3: wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: wie `at_once` = 2.
Hinweise	• Siehe auch Kapitel 8.2 "Koordinatentransformationen", Seite 183.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	Letzte Änderung Version DLL 525, OUT 527: Verhalten für `at_once` = 3.	
Verweise	**set_scale_list**, **set_angle**, **set_offset**, **set_matrix**	

Variabler Listenbefehl	**set_scale_list**
Funktion	Identisch mit **set_scale**, aber ein Listenbefehl.
Aufruf	`set_scale_list( HeadNo, Scale, at_once )`
Parameter	`HeadNo` Siehe **set_scale**.
	`Scale` Siehe **set_scale**.
	`at_once` Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die definierte Transformation wirksam wird: = 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 “Laser active”-Lasersteuersignale 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**-, **jump_rel**-, **goto_xy**- oder **goto_xyz**-Befehls zusammen mit allen bis dahin nur gesammelt und zwischengespeicherten Transformationswerten sofort berechnet und auf die dann aktuelle Position angewandt. = 3: wie `at_once` = 1, die Lasersteuersignale bleiben aber unverändert. > 3: wie `at_once` = 2
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 525, OUT 527: Verhalten für `at_once` = 3.
Verweise	**set_scale**

Verzögerter kurzer Listenbefehl	**set_scanner_delays**
Funktion	Definiert die Scanner-Delays.
Aufruf	`set_scanner_delays( Jump, Mark, Polygon )`
Parameter	`Jump, Mark, Polygon` Scanner-Delays als 32-Bit-Werte ohne Vorzeichen. *1 Bit entspricht 10 µs*. Zulässiger Bereich: [0…(2^{32}–1)].
Hinweise	• Die Scanner-Delays werden im Kapitel 7.2.2 "Scanner-Delays", Seite 113 beschrieben. • Variable Delays für Sprünge und Polygonzüge können über **set_delay_mode** eingestellt werden. • Die angegebenen Delays werden – falls nötig – von der RTC5 automatisch angepasst, um Fehler in der Laser-Ansteuerung zu vermeiden (siehe "Automatische Delay-Anpassung" auf Seite 121). • Die Default-Einstellung nach **load_program_file** entspricht `set_scanner_delays( 9, 6, 3 )`.
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich).
Verweise	**set_delay_mode**, **set_laser_delays**

Kontrollbefehl	**set_serial**
Funktion	Setzt die Startseriennummer für den mit **select_serial_set** zuletzt ausgewählten Seriennummernsatz (bzw. nach **load_program_file** für den Seriennummernsatz 0) 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)].
Hinweise	• Der Befehl **set_serial** ist synonym zu **set_serial_step** mit `Step` = 1 (siehe Hinweise dort).
RTC4→ RTC5	Unveränderte Funktionalität (außer erweiterter Wertebereich). Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
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 (bzw. nach **load_program_file** für den Seriennummernsatz 0).	
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 stets nur die letzten 4 Dezimalziffern verwendet.
Hinweise	• Bei der Initialisierung wird die Startseriennummer auf 0 und die Schrittweite auf 1 gesetzt. • 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 des Befehls siehe Kapitel 7.5.2 "Markieren von Seriennummern", Seite 170.	
RTC4→ RTC5	Neuer Befehl.	
Verweise	**mark_serial**, **mark_serial_abs**, **set_serial**, **select_serial_set**, **select_serial_set_list**	

Normaler Listenbefehl	set_serial_step_list
Funktion	Setzt die Startseriennummer und die Schrittweite für den mit **select_serial_set_list** zuletzt ausgewählten Seriennummernsatz (bzw. nach **load_program_file** für den Seriennummernsatz 0).
Aufruf	`set_serial_step_list( 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 stets nur die letzten 4 Dezimalziffern verwendet.
Hinweise	• Bei der Initialisierung wird die Startseriennummer (für **mark_serial** und **mark_serial_abs**) auf 0 und die Schrittweite auf 1 gesetzt. • 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 des Befehls siehe Kapitel 7.5.2 "Markieren von Seriennummern", Seite 170.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_serial_step**, **select_serial_set_list**, **get_list_serial**, **mark_serial**, **mark_serial_abs**

Kontrollbefehl	set_sky_writing
Funktion	Aktiviert bzw. deaktiviert den Sky-Writing-Modus und stellt die zugehörigen Parameter ein.
Aufruf	`set_sky_writing( Timelag, LaserOnShift )`
Parameter	`Timelag` Siehe **set_sky_writing_para**.
	`LaserOnShift` Siehe **set_sky_writing_para**.
Hinweise	• Der Befehl 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 127.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_sky_writing_para**, **set_sky_writing_list**

Kontrollbefehl	set_sky_writing_limit
Funktion	Legt den Grenzwert für die Sky-Writing-Umschaltung im Modus 3 fest.
Aufruf	`set_sky_writing_limit( Limit )`
Parameter	`Limit` Gewünschter Grenzwert im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich: [–1.0 … +1.0]. Überlaufende Werte werden auf den Randbereich geclippt.
Hinweise	• Zur Verwendung des Befehls siehe Kapitel 7.2.4 "Sky-Writing", Seite 127. • `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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	**set_sky_writing_limit_list**

Unverzögerter kurzer Listenbefehl	set_sky_writing_limit_list
Funktion	Identisch mit **set_sky_writing_limit**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_limit_list( Limit )`
Parameter	`Limit` Siehe **set_sky_writing_limit**.
Hinweise	• Siehe **set_sky_writing_limit**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 531, OUT 532.
Verweise	**set_sky_writing_limit**

Normaler Listenbefehl	set_sky_writing_list
Funktion	Identisch mit **set_sky_writing**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_list( Timelag, LaserOnShift )`
Parameter	`Timelag` Siehe **set_sky_writing_para**. `LaserOnShift` Siehe **set_sky_writing_para**.
Hinweise	• Siehe **set_sky_writing**, **set_sky_writing_para** und **set_sky_writing_para_list**.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_sky_writing**, **set_sky_writing_para**, **set_sky_writing_para_list**

Kontrollbefehl	**set_sky_writing_mode**
Funktion	Schaltet den Sky-Writing-Modus um bzw. aus.
Aufruf	`set_sky_writing_mode( Mode )`
Parameter	`Mode` Gewünschter Sky-Writing-Modus als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … (2^{32}–1)]. = 0: Sky-Writing (Modus 1 oder gegebenenfalls Modus 2 oder 3) wird deaktiviert. = 1: Sky-Writing-Modus 1 wird aktiviert. = 2: Sky-Writing-Modus 2 wird aktiviert. > 2: Sky-Writing-Modus 3 wird aktiviert.
Hinweise	• Sky-Writing-Modi 2 und 3 können nur dann (mit `Mode` > 1) aktiviert werden, wenn zuvor mindestens einmal (zuletzt) **set_sky_writing_para** mit `Timelag` ≥ 1/64 aufgerufen (und damit Sky-Writing-Modus 1 aktiviert) wurde. • Nach **set_sky_writing_mode**(`Mode` = 0) ist eine nachträgliche Sky-Writing-Reaktivierung mit **set_sky_writing_mode**(`Mode` > 0) möglich, nicht dagegen nach **set_sky_writing_para** mit `Timelag` < 1/64 (hier ist zur Reaktivierung erneut **set_sky_writing_para** mit `Timelag` ≥ 1/64 erforderlich). • 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 Mark- oder Bogenbefehl noch *ohne* Sky-Writing fertig ausgeführt. • Wird Sky-Writing-Modus 1 während Ausführung einer Liste *deaktiviert* (Umschalten von `Mode` = 1 nach 0), so wird ein im Sky-Writing-Modus 1 angefangener Mark- oder Bogenbefehl noch im Sky-Writing-Modus 1 beendet und anschließend ein Mark-Delay eingefügt. • Ein Aktivieren oder Deaktivieren von Sky-Writing-Modus 2 bzw. 3 (Umschalten nach oder von `Mode` = 2 bzw. 3) ist nicht möglich, wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten). Der Befehl wird dann ignoriert (mit **get_last_error**-Returncode: `RTC5_BUSY`). Der Befehl wird dagegen immer ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Letzte Änderung (Version DLL 531, OUT 532): Modus 3.
Verweise	**set_sky_writing_mode_list**, **set_sky_writing_para**

Normaler Listenbefehl	**set_sky_writing_mode_list**
Funktion	Identisch mit **set_sky_writing_mode**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_mode_list( Mode )`
Parameter	`Mode` Siehe **set_sky_writing_mode**.
Hinweise	• Sky-Writing-Modus 2 bzw. 3 können mit **set_sky_writing_mode_list** (anders als mit **set_sky_writing_mode**) auch innerhalb einer Liste aktiviert oder deaktiviert werden (Umschalten nach oder von `Mode` = 2 bzw. 3). Dabei wird vorübergehend in den Sky-Writing Modus 1 umgeschaltet. • 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, sofern kein anderes Delay (z. B. ein Jump-Delay) wirksam ist.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Letzte Änderung (Version DLL 531, OUT 532): Modus 3.
Verweise	**set_sky_writing_mode**

Kontrollbefehl	set_sky_writing_para	
Funktion	Aktiviert bzw. deaktiviert den Sky-Writing-Modus und stellt die zugehörigen Parameter ein.	
Aufruf	`set_sky_writing_para( Timelag, LaserOnShift, Nprev, Npost )`	
Parameter	`Timelag`	Sky-Writing-Parameter im 64-Bit-IEEE-Gleitkommaformat. *1,0 entspricht 1 µs*. Bis DLL 536 gilt: ≥ 1/4: Sky-Writing-Modus 1 wird aktiviert. < 1/8: Sky-Writing (Modus 1 oder gegebenenfalls Modus 2 bzw. 3) wird deaktiviert. Ab DLL 537 gilt: ≥ 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 (Modus 1 oder gegebenenfalls Modus 2 bzw. 3) wird deaktiviert.
	`LaserOnShift`	Verschiebung (für positive Werte Verzögerung) des Einschaltzeitpunkts der Lasersteuersignale als 32-Bit-Wert mit Vorzeichen (siehe Abb. 45). *1 Bit entspricht 0,5 µs*. Negative Werte kleiner als der gesamte Vorlauf werden geclippt, d.h. es gilt automatisch: – `LaserOnShift` ≥ ca. (–40) × `Nprev` (Modus 1). – `LaserOnShift` ≥ ca. (–20) × `Nprev` (Modus 2 bzw. 3).
	`Nprev`	Dieser Parameter (32-Bit-Wert ohne Vorzeichen) definiert die Dauer der Vorlaufphase (siehe Abb. 45). *1 Bit entspricht 10 µs*. – Vorlaufdauer / µs = 20 × `Nprev` (Modus 1). – Vorlaufdauer / µs = 10 × `Nprev` (Modus 2 bzw. 3). – Zulässiger Bereich: 0 ≤ `Nprev` ≤ (2^{32}–1).
	`Npost`	Dieser Parameter (32-Bit-Wert ohne Vorzeichen) definiert die Dauer der Nachlaufphase (siehe auch Abb. 45). *1 Bit entspricht 10 µs*. – Nachlaufdauer / µs = 20 × `Npost` (Modus 1). – Nachlaufdauer / µs = 10 × `Npost` (Modus 2 bzw. 3). – Zulässiger Bereich: 0 ≤ `Npost` ≤ (2^{32}–1).

Kontrollbefehl	set_sky_writing_para
Hinweise	• Zum Sky-Writing und zur Bedeutung der Parameter siehe Kapitel 7.2.4 "Sky-Writing", Seite 127. • Für `Nprev` ≥ 65535 wirkt **set_sky_writing_para** wie **set_sky_writing**: `Nprev` wird auf einen Wert von ca. (0,15 × `Timelag`) eingestellt, d.h. – Vorlaufdauer / µs = ca. 3 × `Timelag` (Modus 1) – Vorlaufdauer / µs = ca. 1,5 × `Timelag` (Modus 2 bzw. 3) `Nprev` wird dabei gegebenenfalls auf den maximalen Wert 65535 geclippt. • Für `Npost` ≥ 65535 wirkt **set_sky_writing_para** wie **set_sky_writing**: `Npost` wird auf einen Wert von ca. (0,1 × `Timelag`) eingestellt, d.h. – Nachlaufdauer / µs = ca. 2 × `Timelag` (Modus 1) – Nachlaufdauer / µs = ca. 1 × `Timelag` (Modus 2 bzw. 3) `Npost` wird dabei gegebenenfalls auf den maximalen Wert 65535 geclippt. • **set_sky_writing_para** kann nicht dazu verwendet werden, zwischen Sky-Writing-Modus 1, 2 und 3 hin- und herzuschalten (dafür ist **set_sky_writing_mode** vorgesehen). • Wird **set_sky_writing_para** (`Timelag` ≥ 1/64) bei deaktiviertem Sky-Writing aufgerufen, so wird Sky-Writing-Modus 1 aktiviert. Erfolgt der Befehlsaufruf während der Ausführung einer Liste, so ist Modus 1 erst ab dem nächsten Listenbefehl wirksam (ein bereits angefangener Mark- oder Bogenbefehl wird gegebenenfalls noch *ohne* Sky-Writing fertig ausgeführt). • Wird **set_sky_writing_para** bei aktiviertem Sky-Writing-Modus 1 aufgerufen, so wirken die angegebenen Parameter ab dem nächsten Listenbefehl. Erfolgt der Befehlsaufruf während der Ausführung einer Liste, – so wird für (`Timelag` ≥ 1/64) das Sky-Writing für einen bereits angefangenen Mark- oder Bogenbefehl gegebenenfalls noch mit den bisherigen Parametern fertig ausgeführt, – so wird für (`Timelag` < 1/64) ein bereits (im Sky-Writing-Modus 1) angefangener Mark- oder Bogenbefehl gegebenenfalls auch noch im Sky-Writing-Modus 1 beendet und anschließend ein Mark-Delay eingefügt. • Wird **set_sky_writing_para** bei aktiviertem Sky-Writing-Modus 2 bzw. 3 aufgerufen, wenn gerade der BUSY-Status der Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten), so wird der Befehl ignoriert (**get_last_error**-Returncode: `RTC5_BUSY`). Der Befehl wird dagegen ausgeführt, wenn gerade keine Liste ausgeführt wird oder wenn eine Liste mit **set_wait** angehalten wurde: – Für `Timelag` < 1/64 wird Sky-Writing-Modus 2 bzw. 3 deaktiviert. – Für `Timelag` ≥ 1/64 wirken die angegebenen Parameter ab dem nächsten Listenbefehl. `Nprev` = 0 und/oder `Npost` = 0 werden allerdings für Sky-Writing-Modus 2 bzw. 3 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→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 537, OUT 537: `Timelag`.
Verweise	**set_sky_writing**, **set_sky_writing_para_list**

Normaler Listenbefehl	**set_sky_writing_para_list**
Funktion	Identisch mit **set_sky_writing_para**, aber ein Listenbefehl.
Aufruf	`set_sky_writing_para_list( Timelag, LaserOnShift, Nprev, Npost )`
Parameter	`Timelag` Siehe **set_sky_writing_para**.
	`LaserOnShift` Siehe **set_sky_writing_para**.
	`Nprev` Siehe **set_sky_writing_para**.
	`Npost` Siehe **set_sky_writing_para**.
Hinweise	• Anders als **set_sky_writing_para** kann **set_sky_writing_para_list** auch bei aktiviertem Sky-Writing-Modus 2 bzw. 3 innerhalb einer Liste ausgeführt werden. Dabei wird vorübergehend von Sky-Writing-Modus 2 bzw. 3 in Modus 1 umgeschaltet: Gegebenfalls wird ein unmittelbar vorausgehender Markierbefehl im Modus 1 beendet und auch der nächste Markierbefehl im Modus 1 begonnen. Danach ist wieder Modus 2 bzw. 3 aktiviert, sofern `Timelag` ≥ 1/64 ist. Weiter gilt: – Für `Timelag` ≥ 1/64 wirken die angegebenen Parameter ab dem nächsten Listenbefehl (`Nprev` = 0 und/oder `Npost` = 0: siehe **set_sky_writing_para**). – Für `Timelag` < 1/64 wird Sky-Writing endgültig deaktiviert. • Ebenso wie mit **set_sky_writing_para** kann mit **set_sky_writing_para_list** Sky-Writing-Modus 2 bzw. 3 nicht dauerhaft in Sky-Writing Modus 1 umgeschaltet werden. • Bei der Deaktivierung des Sky-Writing-Modus 1, 2 oder 3 mit **set_sky_writing_para_list** wird ein vor der Sky-Writing-Aktivierung definiertes Mark-Delay eingefügt, sofern kein anderes Delay (z. B. ein Jump-Delay) wirksam ist.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 531, OUT 532: Modus 3.
Verweise	**set_sky_writing_para**

Kontrollbefehl	**set_softstart_level**
Funktion	Definiert die Softstartwerte.
Aufruf	`set_softstart_level( Index, Level )`
Parameter	Alle Parameter sind 32-Bit-Werte ohne Vorzeichen. Parameter — Zulässige Werte — Beschreibung `Index` — 0…1023 — Indexnummer des Softstartwerts. `Level` — 0…2^{12}–1 — Falls Softstart-Mode = 1, 2, 11 oder 12 (siehe **set_softstart_mode**). Der Wert 0 entspricht einem Analogspannungswert von 0 V. Der Wert 2^{12}–1 entspricht 10 V. 0…(2^{32}–1) — Falls Softstart-Mode = 3 oder 13 (siehe **set_softstart_mode**). 1 Bit entspricht einer Pulslänge von 1/64 µs (im RTC5-Modus) bzw. 1/8 µs (im RTC4-Kompatibilitätsmodus).
Hinweise	• Der **set_softstart_level**-Befehl legt den `Level`-Wert für die Pulsnummer `Index` fest. Für jeden einzelnen `Level`-Wert muss der Befehl separat aufgerufen werden. • `Index` muss aus [0…1023] sein, sonst wird der **set_softstart_level**-Befehl nicht ausgeführt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Für `Index` ≥ `Number` (`Number` wird mit **set_softstart_mode** festgelegt) wird **set_softstart_level** zwar ausgeführt, der definierte Wert jedoch beim Softstart nicht verwendet. • Vergewissern Sie sich, so viele Softstartwerte anzugeben, wie Sie mit **set_softstart_mode** festgelegt haben (nicht angegebene Softstartwerte haben undefinierte Werte). • Der Softstart-Modus wird mit dem Befehl **set_softstart_mode** aktiviert. • Für Softstart-Mode = 1, 2, 11 oder 12 wird `Level` auf 12 Bit begrenzt, überzählige Bits werden ignoriert (siehe auch **write_da_x** oder **write_da_x_list**). • Ist für Softstart-Mode = 3 oder 13 die angegebene Softstart-Pulslänge größer als die unter **set_laser_pulses** oder **set_laser_timing** spezifizierte Lasersignal-Periodendauer (2 × `HalfPeriod`), dann bleibt der Laser dauernd an. • Insbesondere wenn die Softstartwerte als Analogspannungswerte ausgegeben werden sollen (Softstart-Mode = 1, 2, 11 oder 12), muss der Befehl **set_softstart_mode** aufgerufen werden, *bevor* der erste **set_softstart_level**-Befehl aufgerufen wird, weil sonst die angegebenen Werte nicht korrekt in Analogwerte umgerechnet werden. • Beachten Sie auch die Hinweise im Abschnitt "Softstart-Modus" auf Seite 152. • Wenn der Wobbelmode "Frei definierbare Wobbelfigur" aktiviert ist, bleibt dieser Befehl wirkungslos, siehe Kapitel 8.4 "Wobbelmode", Seite 189.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus werden Analogspannungswerte intern mit 4 multipliziert (2^{10}–1 entsprechen hier 10 V; es wird von 10 auf 12 Bit umgerechnet) und Pulslängenwerte mit 8 multipliziert (1 Bit entspricht hier 1/8 µs; es wird von 1/8 µs-Einheiten nach 1/64 µs-Einheiten umgerechnet; die 1 µs-Zeitbasis – siehe **set_laser_timing** – wird nicht mehr unterstützt). Die zulässigen Wertebereiche für `Level` verringern sich entsprechend. Wegen dieser unterschiedlichen Behandlung von Analogspannungs- und Pulslängenwerten sollte **set_softstart_mode** unbedingt vor **set_softstart_level** aufgerufen werden.
Verweise	**set_softstart_mode**, **set_softstart_mode_list**, **set_softstart_level_list**

Normaler Listenbefehl	**set_softstart_level_list**
Funktion	Ähnlich wie **set_softstart_level**, aber ein Listenbefehl.
Aufruf	`set_softstart_level_list( Index, Level1, Level2, Level3 )`
Parameter	Siehe **set_softstart_level**.
Hinweise	• Der **set_softstart_level_list**-Befehl definiert drei `Level`-Werte für die Pulse Nummer `Index` bis `Index+2`. Der Befehl muss für jedes individuelle `Level1/2/3`-Wertetripel separat aufgerufen werden. • `Index` muss im Bereich [0…1023] liegen, sonst wird der **set_softstart_level_list**-Befehl schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Level-Werte, für die die Indexzählung über `Number` hinausreicht (`Number` wird mit **set_softstart_mode** festgelegt), werden beim Softstart nicht verwendet. • Ansonsten ist der Befehl identisch mit dem Kontrollbefehl **set_softstart_level** (siehe Hinweise dort). • Wenn der Wobbelmode "Frei definierbare Wobbelfigur" aktiviert ist, bleibt dieser Befehl wirkungslos, siehe Kapitel 8.4 "Wobbelmode", Seite 189.
RTC4→ RTC5	Neuer Befehl. RTC4-Kompatibilitätsmodus: siehe **set_softstart_level**.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**set_softstart_level**, **set_softstart_mode**, **set_softstart_mode_list**

Kontrollbefehl	**set_softstart_mode**		
Funktion	Schaltet den Softstart-Modus ein bzw. aus.		
Aufruf	`set_softstart_mode( Mode, Number, Delay )`		
Parameter	Alle Parameter sind 32-Bit-Werte ohne Vorzeichen.		
	Parameter	Zulässige Werte	Beschreibung
	`Mode`	= 0	Deaktiviert den Softstart-Modus, löscht aber *nicht* die zuvor geladenen Softstartwerte.
		= 1, 11	Die mit **set_softstart_level** definierten Softstartwerte werden als Analogspannungswerte am ANALOG OUT1-Ausgang ausgegeben.
		= 2, 12	Die mit **set_softstart_level** definierten Softstartwerte werden als Analogspannungswerte am ANALOG OUT2-Ausgang ausgegeben.
		= 3, 13	Die mit **set_softstart_level** definierten Softstartwerte definieren die Pulslängen der ersten `Number` Lasersignalpulse am LASER1-Ausgang (weder im Lasermode 4 noch im Lasermode 6).
	`Number`	1 … 1024	Zahl der zu übergebenden Softstartwerte.
	`Delay`	0 … (2^{32}–1) **(1 Bit entspricht 10 µs)**	Definiert die Zeitspanne, für die der Laser ausgeschaltet gewesen sein muss, ehe der Softstart beim nächsten Einschalten aktiviert wird.
Hinweise	• Der Befehl **set_softstart_mode** sollte unbedingt aufgerufen werden, *bevor* der erste **set_softstart_level**-Befehl aufgerufen wird, insbesondere wenn die Softstartwerte als Analogspannungswerte ausgegeben werden sollen (Standardeinstellung für die Softstartwerte-Behandlung: `Mode` = 3). • Die einzelnen Softstartwerte `Level`(0) … `Level`(`Number` – 1) müssen separat und jeweils mit eigenem Kontrollbefehl **set_softstart_level** übergeben werden. Indices, für die keine `Level`-Werte übertragen wurden, sind undefiniert. • Für ungültige Werte von `Mode` oder `Number` wird der Softstart-Modus ausgeschaltet. • Beim Ausschalten des Softstart-Modus werden die bereits übertragenen Werte nicht gelöscht und können nach dem Wiedereinschalten weiter verwendet werden. Es ist aber darauf zu achten, dass dabei nicht versehentlich zwischen Analogspannungswerten (`Mode` = 1, 2, 11 oder 12) und Pulslängen (`Mode` = 3 oder 13) umgeschaltet wird. Zwischen ANALOG OUT1 (`Mode` = 1 oder 11) und ANALOG OUT2 (`Mode` = 2 oder 12) kann aber jederzeit gewechselt werden. • Der Softstart-Modus ist inkompatibel mit dem Pixelausgabemodus (siehe Seite 217). Der Befehl **set_softstart_mode** wird zwar ausgeführt, bis zum Ende des Pixelausgabemodus wird aber kein Softstart ausgeführt. • Der Softstart-Modus sollte nicht zusammen mit der automatischen Lasersteuerung (siehe Seite 159) verwendet werden, wenn dort die 12-Bit-Ausgabewerte am ANALOG OUT1- oder ANALOG OUT2- Ausgang, die Pulslängen (PulseLength) oder Ausgabeperioden (HalfPeriod) der Lasersignale LASER1 und LASER2 angepasst werden. Der Softstart-Modus ist nicht mit der "Frei definierbaren Wobbelfigur" kombinierbar. Wenn eine solche definiert und aktiviert ist, bleibt der Befehl ohne Wirkung. • Beachten Sie auch die Hinweise in Kapitel 7.4.5 "Softstart-Modus", Seite 152.		

Kontrollbefehl	set_softstart_mode
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität (außer erhöhter Wertebereich). Statt einer Übernahme mit der 2. Flanke der Lasersignalpulse (wie es bei der RTC4 mit `Mode` = 1, 2 oder 3 eingestellt werden konnte) kann bei der RTC5 eine Übernahme mit einem um 180° phasenverschobenen Lasersignal realisiert werden (siehe Hinweise im Kapitel 7.4.5 "Softstart-Modus", Seite 152). Eine Indexverschiebung liegt in keinem Fall mehr vor.
Versionsinfo	Letzte Änderung Version OUT 515.
Verweise	**set_softstart_level**, **set_softstart_level_list**, **set_softstart_mode_list**

Variabler Listenbefehl	set_softstart_mode_list
Funktion	Identisch mit **set_softstart_mode**, aber ein Listenbefehl.
Aufruf	`set_softstart_mode_list( Mode, Number, Delay )`
Parameter	Siehe **set_softstart_mode**.
Hinweise	• **set_softstart_mode_list** schaltet die "Laser active"-Lasersteuersignale durch Ausführen von **list_nop** aus und wartet ein LaserOff-Delay ab. • Ansonsten ist der Befehl identisch mit dem Kontrollbefehl **set_softstart_mode** (siehe Hinweise dort).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**set_softstart_mode**, **set_softstart_level**, **set_softstart_level_list**

Kontrollbefehl	**set_standby**
Funktion	Definiert die Ausgabeperiode und die Pulslänge der Standby-Pulse für den "Laser standby"-Betrieb bzw. – im Lasermode 4 und Lasermode 6 – der freilaufenden Lasersignale 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 Bereich: [0 … $+(2^{32}–1)$].
	`PulseLength` Pulslänge der Standbypulse als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … $(2^{32}–1)$].
Hinweise	• Nach einem Hardware-Reset (und nach dem Befehl **load_program_file**) muss zur erstmaligen Aktivierung der Ausgänge LASER1, LASER2 und LASERON der Befehl **set_laser_control** aufgerufen werden, ehe mit **set_standby** oder **set_standby_list** Standby-Pulse aktiviert werden können (siehe Seite 144). Gleichzeitig wird mit **set_laser_control** der Signalpegel der Standby-Pulse eingestellt. • Die Standbypulse sind in **allen** Laserversionen (YAG 1/2/3/5, CO_2, Lasermode 4 und 6) verfügbar. Sie können deaktiviert (ausgeschaltet) werden, indem die Standby-Pulslänge und/oder die -Ausgabeperiode auf Null (Standardeinstellung) gesetzt wird. • Beachten Sie, dass unter `HalfPeriod` die *halbe* Standby-Ausgabeperiode angegeben werden muss (siehe Abb. 50 und Abb. 52). • `HalfPeriod` sollte bei Verwendung des Softstart-Modus (siehe Seite 152) nicht kleiner als 104 sein (dies wird aber nicht automatisch kontrolliert). Dies entspricht einer Laserpulsfrequenz von ca. 308 kHz. • Wenn `PulseLength` größer gleich der Ausgabeperiode (2 × `HalfPeriod`) ist, bleibt der Laser dauernd an. • Die Taktfrequenz für die Standby-Pulse ist immer 64 MHz (1 Bit entspricht also 1/64 µs). • Die Lasersteuerungsversion wird mit dem Befehl **set_laser_mode** ausgewählt (siehe Kapitel 7.4 "Lasersteuerung", Seite 144). Die aktuelle Ausgabeperiode und die Pulslänge der Lasersteuersignale für den "Laser active"-Betrieb (außer im Lasermode 4 und 6) werden mit den Befehlen **set_laser_pulses_ctrl**, **set_laser_pulses** oder **set_laser_timing** festgelegt.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität (außer erhöhter Wertebereich), allerdings: • `HalfPeriod` und `PulseLength` werden bei der RTC5 in Einheiten von 1/64 µs angegeben, im RTC4-Kompatibilitätsmodus dagegen wie bei der RTC4 in Einheiten von 1/8 µs. Im RTC4-Kompatibilitätsmodus werden die angegebenen Werte von der RTC5 mit 8 multipliziert (die zulässigen Wertebereiche verringern sich entsprechend; der kleinste sinnvolle Wert für `HalfPeriod` mit Softstart-Modus ist dann 13). • U.U. muss vor **set_standby** der Befehl **set_laser_control** aufgerufen werden (s.o.).
Verweise	**set_standby_list**, **get_standby**

Verzögerter kurzer Listenbefehl	**set_standby_list**
Funktion	Identisch mit **set_standby**, aber ein Listenbefehl.
Aufruf	`set_standby_list( HalfPeriod, PulseLength )`
Parameter	`HalfPeriod` *halbe* Standby-Ausgabeperiode als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … +(2^{32}–1)].
	`PulseLength` Pulslänge der Standbypulse als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 1/64 µs*. Zulässiger Bereich: [0 … (2^{32}–1)].
RTC4→ RTC5	wie **set_standby**

Kontrollbefehl	**set_start_list**
Funktion	Öffnet den Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Input-Pointer an den Listenanfang der gewünschten Liste (“Liste 1” oder “Liste 2”). Der nächste Listenbefehl wird an dieser Adresse gespeichert und alle nachfolgenden Listenbefehle an den darauffolgenden Adressen in der gewählten Liste.
Aufruf	`set_start_list( ListNo )`
Parameter	`ListNo` Listennummer der Liste, in die der Input-Pointer gesetzt werden soll, als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [ungerade: “Liste 1”, gerade: “Liste 2”].
Hinweise	• Der Befehl **set_start_list** ist synonym zu **set_start_list_pos** mit `Pos` = 0. • Alternativ können die Befehle **set_start_list_1** und **set_start_list_2** (ohne Parameter) verwendet werden.
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**execute_list**, **read_status**, **set_start_list_pos**

Kontrollbefehl	**set_start_list_1**
Funktion	Siehe **set_start_list**.
Aufruf	`set_start_list_1()`
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**set_start_list**

Kontrollbefehl	**set_start_list_2**
Funktion	Siehe **set_start_list**.
Aufruf	`set_start_list_2()`
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**set_start_list**

Kontrollbefehl	set_start_list_pos	
Funktion	Öffnet den Listenspeicher zum Beschreiben mit Listenbefehlen und setzt den Input-Pointer an die angegebene (relative) Position in der gewünschten Liste (“Liste 1” oder “Liste 2”). Der nächste Listenbefehl wird an dieser Adresse gespeichert und alle nachfolgenden Listenbefehle an den darauffolgenden Adressen in der gewählten Liste.	
Aufruf	`set_start_list_pos( ListNo, Pos )`	
Parameter	`ListNo`	Listennummer der Liste, in die der Input-Pointer gesetzt werden soll, als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: [ungerade: “Liste 1”, gerade: “Liste 2”].
	`Pos`	Position des Input-Pointers (Offset bzgl. des Listenanfangs der jeweiligen Liste) als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … (2^{20}–1)].
Hinweise	• Die ausgewählte 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 75 und **load_list**). • Der Input-Pointer kann *nicht* in den geschützten Bereich (“Liste 3”) gesetzt werden. Soll der geschützten Bereich (in voller Eigenverantwortung des Anwenders) mit **set_start_list_pos** beschrieben werden, kann dieser Bereich vorübergehend mit `ConfigList( Mem1, –1 )` der “Liste 2” zugeschlagen werden. Nach Beenden der Schreibaktion sollte die Konfiguration zum Schutz des Bereichs aber wieder rückgesetzt werden: `ConfigList( Mem1, Mem2 )`. • Für ungerade Werte von `ListNo` wird “Liste 1”, andernfalls “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 stets “Liste 1” geöffnet. • Wird `Pos` größer als der Speicherbereich der jeweiligen Liste gewählt (`Pos` > `Mem1` bzw. `Pos` > `Mem2`), so wird `Pos` auf 0 gesetzt. • Die Listenstatuswerte der gewählten Liste (siehe auch **read_status**) werden wie folgt gesetzt: LOAD gesetzt, READY rückgesetzt, USED rückgesetzt. Der LOAD-Status der jeweils anderen Liste wird rückgesetzt. • Der **set_start_list_pos**-Befehl löst einen Flush der gepufferten Listeneingabe aus (siehe Seite 75). • **set_start_list_pos** deckt auch die Spezialisierungen **set_start_list_1**, **set_start_list_2**, **set_start_list** bzw. **set_input_pointer** ab. • *Vorsicht: Wenn das Ende des jeweiligen Listenbereichs erreicht ist, wird der Input-Pointer automatisch an den Anfang desselben Listenbereichs zurückgesetzt. Vergewissern Sie sich, dass Sie keine Befehle überschreiben, die in Ihrem Anwenderprogramm noch benötigt werden.*	
RTC4→ RTC5	Neuer Befehl.	
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^{20}–1)].
Hinweise	• Für `Index` > 1023 und/oder `Pos` > (2^{20}–1) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode `RTC5_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 werden und/oder über den Index aufgerufen werden. • Mit **set_sub_pointer** kann aber auch ein indiziertes Unterprogramm oder Zeichen (bzw. Zeichenfolge) erneut referenziert werden, so dass es auch über einen zweiten Index aufgerufen werden kann. Zur Indexverwaltung sollte aber bevorzugt der Befehl **copy_dst_src** verwendet werden. • Die Anfangsadressen von Befehlslisten, die mit **set_sub_pointer** referenziert werden sollen, können – *vor* dem Abspeichern der Befehlslisten – mit dem Befehl **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 stets die Anfangsadresse eines tatsächlichen Unterprogramms sein, welches mit **list_return** abgeschlossen worden ist und kein **set_end_of_list** enthält.	
RTC4→ RTC5	Neuer Befehl.	
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^{20}–1)].
Hinweise	• Indizierte Zeichenfolgen können bei der Beschriftung von Uhrzeit, Datum und Seriennummern eingesetzt werden (siehe “Aufruf von indizierten Zeichenfolgen”, Seite 89). • Für `Index` > 41 und/oder `Pos` > (2^{20}–1) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode `RTC5_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 werden und/oder über den Index aufgerufen werden. • Mit **set_text_table_pointer** kann aber auch ein indiziertes Unterprogramm oder Zeichen (bzw. Zeichenfolge) erneut referenziert werden, so dass es auch über einen zweiten Index aufgerufen werden kann. Zur Indexverwaltung sollte aber bevorzugt der Befehl **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 dem Befehl **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 stets die Anfangsadresse eines tatsächlichen Unterprogramms sein, welches mit **list_return** abgeschlossen worden ist und kein **set_end_of_list** enthält. • Der Befehl ist synonym zum Befehl **set_char_table**, der vorher auch schon bei der RTC *SCAN*alone-Karte (Standalone-Version der RTC4-Karte) zur Verfügung stand.
RTC4→ RTC5	Neuer Befehl.
Verweise	**load_text_table**, **mark_date**, **mark_serial**, **mark_time**

Verzögerter kurzer Listenbefehl	**set_trigger**
Funktion	Startet die Messwertspeicherung der ausgewählten Messsignale.
Aufruf	`set_trigger( Period, Signal1, Signal2 )`
Parameter	`Period` Messperiode als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 10 µs*. Zulässiger Wertebereich: [0 … (2^{32}–1)].
	`Signal1, Signal2` Gewünschter Signaltyp für Speicherkanal 1 und 2 als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: wie unten aufgelistet. = 0: LASERON-Signal (1 = Lasersignal an, 0 = Lasersignal aus). = 1: StatusAX (Statussignal der X-Achse des primären Scan-Kopf-Anschlusses). = 2: StatusAY (Statussignal der Y-Achse des primären Scan-Kopf-Anschlusses). = 3: Reserviert. = 4: StatusBX (Statussignal der X-Achse des sekundären Scan-Kopf-Anschlusses). = 5: StatusBY (Statussignal der Y-Achse des sekundären Scan-Kopf-Anschlusses). = 6: Reserviert. = 7: SampleX (Kartesischer Ansteuerwert für die X-Achse). = 8: SampleY (Kartesischer Ansteuerwert für die Y-Achse). = 9: SampleZ (Kartesischer Ansteuerwert für die Z-Achse). = 10: SampleAX_Corr (korrigierter Ansteuerwert für die X-Achse des primären Scan-Kopf-Anschlusses). = 11: SampleAY_Corr (korrigierter Ansteuerwert für die Y-Achse des primären Scan-Kopf-Anschlusses). = 12: SampleAZ_Corr (korrigierter Ansteuerwert für die Z-Achse, wenn XY am primären Scan-Kopf-Anschluss angeschlossen sind; identisch zum tatsächlichen Ausgabewert für die Z-Achse). = 13: SampleBX_Corr (korrigierter Ansteuerwert für die X-Achse des sekundären Scan-Kopf-Anschlusses). = 14: SampleBY_Corr (korrigierter Ansteuerwert für die Y-Achse des sekundären Scan-Kopf-Anschlusses). = 15: SampleBZ_Corr (korrigierter Ansteuerwert für die Z-Achse, wenn XY am sekundären Scan-Kopf-Anschluss angeschlossen sind; identisch zum tatsächlichen Ausgabewert für die Z-Achse). = 16: StatusAX+LASERON (StatusAX wenn Lasersignal an, –524.288 wenn Lasersignal aus). = 17: StatusAY+LASERON (StatusAY wenn Lasersignal an, –524.288 wenn Lasersignal aus). = 18: StatusBX+LASERON (StatusBX wenn Lasersignal an, –524.288 wenn Lasersignal aus). = 19: StatusBY+LASERON (StatusBY wenn Lasersignal an, –524.288 wenn Lasersignal aus).

Verzögerter kurzer Listenbefehl	set_trigger		
	`Signal1`, `Signal2` (Forts.)	= 20:	SampleAX_Out (tatsächlicher Ausgabewert für die X-Achse des primären Scan-Kopf-Anschlusses; gegebenenfalls auch inkl. “Offset”- und “Gain”-Kompensation, siehe Hinweise; nicht für Z-Achsen-Ausgabewerte verwendbar).
		= 21:	SampleAY_Out (tatsächlicher Ausgabewert für die Y-Achse des primären Scan-Kopf-Anschlusses; nicht für Z-Achsen-Ausgabewerte verwendbar).
		= 22:	SampleBX_Out (tatsächlicher Ausgabewert für die X-Achse des sekundären Scan-Kopf-Anschlusses; nicht für Z-Achsen-Ausgabewerte verwendbar).
		= 23:	SampleBY_Out (tatsächlicher Ausgabewert für die Y-Achse des sekundären Scan-Kopf-Anschlusses; nicht für Z-Achsen-Ausgabewerte verwendbar).
		= 24:	Lasersteuerparameter der automatischen Lasersteuerung (siehe **set_auto_laser_control**).
		= 25:	SampleAX_Trans (transformierter Ansteuerwert für die X-Achse des primären Scan-Kopf-Anschlusses).
		= 26:	SampleAY_Trans (transformierter Ansteuerwert für die Y-Achse des primären Scan-Kopf-Anschlusses).
		= 27:	SampleAZ_Trans (transformierter Ansteuerwert für die Z-Achse, wenn XY am primären Scan-Kopf-Anschluss angeschlossen sind).
		= 28:	SampleBX_Trans (transformierter Ansteuerwert für die X-Achse des sekundären Scan-Kopf-Anschlusses).
		= 29:	SampleBY_Trans (transformierter Ansteuerwert für die Y-Achse des sekundären Scan-Kopf-Anschlusses).
		= 30:	SampleBZ_Trans (transformierter Ansteuerwert für die Z-Achse, wenn XY am sekundären Scan-Kopf-Anschluss angeschlossen sind).
		= 31:	Lasersteuerparameter der vektorkontrollierten Lasersteuerung (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_auto_laser_control**, **set_vector_control** und Kapitel 9.1.4 “12-Bit-Analog-Ausgänge”).
		= 34:	12-Bit-Ausgabewert am ANALOG OUT2-Ausgang (siehe **set_auto_laser_control**, **set_vector_control** und Kapitel 9.1.4 “12-Bit-Analog-Ausgänge”).
		= 35:	Ausgabewert am 16-Bit-Digital-Ausgang (siehe **set_auto_laser_control**, **set_vector_control** und Kapitel 9.1.1 “16 Bit-Digital-Ausgang”).

Verzögerter kurzer Listenbefehl	set_trigger	
	`Signal1`, `Signal2` (Forts.)	= 36: Ausgabewert am 8-Bit-Digital-Ausgang (siehe **set_auto_laser_control**, **set_vector_control** und Kapitel 9.1.2 "8 Bit-Digital-Ausgang"). = 37: Pulslänge (PulseLength) der Lasersignale LASER1 und LASER2 (siehe **set_auto_laser_control**, **set_vector_control**). = 38: Ausgabeperiode (HalfPeriod) der Lasersignale LASER1 und LASER2 (siehe **set_auto_laser_control**, **set_vector_control**). = 39: FreeVariable0. = 40: FreeVariable1. = 41: FreeVariable2. = 42: FreeVariable3. = 43: Zählerstand von Encoder-Zähler Encoder0. = 44: Zählerstand von Encoder-Zähler Encoder1. = 45: Markiergeschwindigkeit (aus **set_mark_speed**, **set_mark_speed_ctrl**). = 46: 16-Bit-Digital-Eingang (EXTENSION 1). = 47: Zoomwert (nur für intelli*WELD* II). = 48: FreeVariable4. = 49: FreeVariable5. = 50: FreeVariable6. = 51: FreeVariable7. = 52: Zeitstempel-Zähler (siehe Kapitel 8.13 "Zeitmessungen", Seite 231). = 53: Wobbel-Amplitude (siehe **set_wobbel**, **set_wobbel_mode** und Kapitel 8.4 "Wobbelmode", Seite 189). = 54: ReadAnalogIn (siehe **read_analog_in**).
Hinweise	• Sind `Signal1` und `Signal2` nicht aus obiger Liste, wird der Befehl auch für `Period` = 0 durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Nach einem **set_trigger**-Befehl (mit `Period` > 0) wird sofort und dann immer im Abstand der angegebenen Messperiode jeweils ein Datenwertpaar gespeichert. Die Datenspeicherung erfolgt so lange, bis der Befehl erneut (mit `Period` = 0) aufgerufen wird, bis **set_trigger4**(`Period` = 0) aufgerufen wird oder bis der Datenspeicher voll ist (automatische Endabschaltung bei maximal 2^{20} Datenpaaren). • Falls vor **set_trigger** mit **load_correction_file** eine Korrekturdatei als Korrekturtabelle `No`=3 oder `No`=4 geladen wurde, ist die obere Hälfte des für die Datenaufzeichnung mit **set_trigger** vorgesehenen Speicherbereichs belegt. Es können dann mit **set_trigger** nur noch 2^{19} Datenwertpaare aufgezeichnet werden. Die Messwert-Aufzeichnung endet automatisch bei diesem Wert. Falls das Laden einer Korrekturtabelle `No`=3 oder `No`=4 erfolgt, nachdem mit **set_trigger** bereits mehr als 2^{19} Datenpaare aufgezeichnet worden sind, sind die Werte jenseits von 2^{19} nicht mehr verfügbar. **load_program_file** stellt den Ursprungszustand mit bis zu 2^{20} möglichen Messwertpaaren wieder her.	

Verzögerter kurzer Listenbefehl	set_trigger
Hinweise (Forts.)	• Eine mit **set_trigger** gestartete Messwertspeicherung erfolgt nur während der Ausführung einer Liste, auch wenn sie nicht mit **set_trigger** (`Period` = 0) oder **set_trigger4**(`Period` = 0) beendet wurde. In diesem Fall wird die Aufzeichnung mit dem Start einer neuen Liste automatisch fortgesetzt (der Status der Messwertspeicherung wird durch **set_end_of_list** nicht zurückgesetzt). Der Status wird während der Ausführung einer Liste entweder durch **set_trigger** (`Period` = 0) zurückgesetzt, oder durch **stop_execution**. Außerhalb einer aktiven Liste kann der Status mit **stop_trigger** zurückgesetzt werden (**stop_execution** wird da nicht ausgeführt). • Sample-Werte und vom Scan-System rückübertragene Statuswerte (`Signal1, Signal2` = 1-23, 25-30), die durch **set_trigger** auf der RTC5 gegespeichert werden, sind stets im RTC5-20 Bit-Bereich, auch wenn die DLL auf den RTC4-Kompatibilitätsmodus eingestellt ist. Die gespeicherten Messwerte können anschließend mit dem Befehl **get_waveform** zur Auswertung an den PC übertragen werden. Es wird 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 und müssen von Benutzern geeignet ausgewertet werden (siehe Hinweise bei **get_value**). • Den aktuellen Status der Messung kann man mit dem Befehl **measurement_status** abrufen. • Zum Abbrechen einer Messung mit **set_trigger**(`Period` = 0) oder **set_trigger4**(`Period` = 0) sind die Werte für `Signal1` und `Signal2` irrelevant, solange sie im zulässigen Wertebereich liegen. • Wird eine Messung mit **set_trigger**(`Period` = 0) oder **set_trigger4**(`Period` = 0) abgebrochen, gehen die zuvor aufgenommenen Messwerte *nicht* verloren und der Messwertzähler bleibt auf seinem letzten Wert stehen. Dadurch kann auch noch nachträglich mit **measurement_status** die Anzahl der erfolgten 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. • 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 i*DRIVE*-Scan-Systemen (intelli*SCAN*, intelli$SCAN_{de}$, intelli*DRILL*, intelli*cube*, intelli*WELD*, vario$SCAN_{de}$) kann mit dem **control_command**-Befehl eingestellt werden, welche Informationen auf den Statuskanälen zurückgegeben wird. • 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 sekundären Scan-Kopf-Anschlusses angeschlossen (**select_cor_table**([1 oder 2], 0)), kann ihr Statussignal über StatusBX (`Signal1/2` = 4) ausgelesen werden.

Verzögerter kurzer Listenbefehl	set_trigger
Hinweise (Forts.)	• Die kartesischen Ansteuerwerte `Sample<X..Z>` (`Signal1/2`= 7…9) berücksichtigen gegebenenfalls eine Wobbel-Korrektur und eine "Processing on the fly"-Korrektur, aber weder eine Bildfeldkorrektur noch eine Koordinatentransformation. Die "transformierten" Ansteuerwerte `Sample<AX..BZ>_Trans` (`Signal1/2`= 25…30) werden von der RTC5 gegebenenfalls unter Berücksichtigung der mit **set_matrix**, **set_offset**, **set_scale** oder **set_angle** (bzw. mit den entsprechenden Listenbefehlen) festgelegten Koordinatentransformationen, des mit **set_offset_xyz** oder **set_offset_xyz_list** bzw. mit **set_defocus** oder **set_defocus_list** festgelegten Offsets auf die z-Koordinate bzw. die Fokuslänge aus den kartesischen Ansteuerwerten `Sample<X..Z>` berechnet. Die "korrigierten" Ansteuerwerte `Sample<AX..BZ>_Corr` (`Signal1/2`= 10…15) werden von der RTC5 unter Berücksichtigung der zugewiesenen Korrekturtabellen aus den transformierten Werten `Sample<X..Z>_Trans` berechnet. Die tatsächlichen Ausgabewerte `Sample<AX..BY>_Out` (`Signal1/2`= 20……23), die von der RTC5 an das Scan-System übertragen werden, berücksichtigen gegebenenfalls darüber hinaus eine mit **auto_cal** oder **set_hi** aktivierte Kompensation für den "Offset" und den "Gain" der Scanner zur automatischen Selbstkalibrierung des Scan-Systems (siehe auch Seite 141). • Die Signale 12 und 15 sowie 27 und 30 sind jeweils identisch: `SampleAZ_Corr` = `SampleBZ_Corr` und `SampleAZ_Trans` = `SampleBZ_Trans`. • Statussignale `Status<…>` hinken den Ansteuersignalen `Sample<…>` um einige Takte hinterher. • Für `Signal1, Signal2` = 0, 24 und 31…42 werden die Ausgabewerte aus dem jeweils vorherigen Takt aufgezeichnet. Innerhalb des laufenden Taktes können die Werte nach dem Aufzeichnen automatisch oder über Kontroll- bzw- Listenbefehle noch einmal geändert werden. • Für `Signal1, Signal2` = 24 wird der Signalparameter aufgezeichnet, der durch die positions- und/oder geschwindigkeitsabhängige oder encodergeschwindigkeitsabhängige Lasersteuerung automatisch angepasst wird (siehe **set_auto_laser_control**). Eine Aufzeichnung erfolgt nur, wenn die automatische Lasersteuerung aktiviert und während der Laser an ist. Bei ausgeschaltetem Laser wird 0 aufgezeichnet. • Für `Signal1`, `Signal2` = 31 wird der für die vektorkontrollierte Lasersteuerung 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 "Laser active"-Lasersteuersignale (nicht für Standby-Signale) aufgezeichnet. Im Softstart-Modus (siehe Seite 152) und Pixel-Ausgabe-Modus (siehe Seite 217) ändern sich Pulslänge bzw. Ausgabeperiode u.U. schneller als im 10 µs-Takt (Zeitauflösung der Aufzeichnung mit **set_trigger**). • Zu `Signal1, Signal2` = 39…42 siehe Kapitel 6.9.1 "Freie Variablen", Seite 102.

Verzögerter kurzer Listenbefehl	set_trigger
RTC4→ RTC5	Unveränderte Grundfunktion (Messwertspeicherung), allerdings: • Die Messung kann (anders als bei der RTC4) abgebrochen werden, ohne dass die zuvor aufgenommenen Messwerte verloren gehen (s.o. `Period` = 0). • Die Signale `StatusAZ` und `StatusBZ` (RTC4: `Signal1/2` = 3, 6) sind nicht mehr verfügbar. Von `SampleAZ_Corr` und `SampleBZ_Corr` (`Signal1/2` = 12, 15) liefert stets nur eines sinnvolle Signale (siehe auch obigen Hinweis zu 3D-Scan-Systemen). • Die Signale `Sample<X..Z>` und `Sample<AX..BZ>_Corr` haben bei der RTC5 eine andere Bedeutung als bei der RTC4 (siehe Hinweise oben). • Bei der RTC5 beziehen sich *alle* Koordinatenwerte auf den Bildmittelpunkt 0 und liegen im Wertebereich von [-2^{19} … (2^{19}–1)]. Bei der RTC4 liegen dagegen die tatsächlichen Ausgabewerte, die an das Scan-System übertragen werden, im verschobenen Wertebereich von [0 … (2^{16}–1)]. Letzteres gilt auch, wenn das Scan-System von einer RTC5 zusammen mit dem XY2-100-Konverter angesteuert wird (siehe auch Seite 141).
Versionsinfo	Letzte Änderung (DLL 543, OUT 543, RBF 524): `Signal1, Signal2` = 52…54.
Verweise	**get_waveform**, **measurement_status**, **control_command**, **get_value**, **get_values**, **set_mcbsp_out**, **set_mcbsp_out_ptr**, **set_trigger4**

Verzögerter kurzer Listenbefehl	**set_trigger4**
Funktion	Startet die Messwertspeicherung der ausgewählten Messsignale (wirkt wie **set_trigger**, jedoch mit 4 statt 2 Messsignalen gleichzeitig).
Aufruf	`set_trigger4( Period, Signal1, Signal2, Signal3, Signal4 )`
Parameter	`Period` Siehe **set_trigger**.
	`Signal1, Signal2, Signal3, Signal4` Siehe **set_trigger**.
Hinweise	• Siehe Hinweise zu **set_trigger**. • Bei **set_trigger4** stehen im Gegensatz zu **set_trigger** pro Messkanal nur 2^{19} Einträge zur Verfügung (automatische Endabschaltung bei maximal 2^{19} Daten-Quadrupel). • **set_trigger**(`Period` = 0) und **set_trigger4**(`Period` = 0) schalten beide eine aktive Datenaufzeichnung ab, unabhängig davon, mit welchem der beiden Befehle die Datenaufzeichnung gestartet wurde. Wird eine Messung 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. Siehe auch die entsprechenden Hinweise unter **set_trigger**. • Die Datenkanäle 3 und 4 belegen die obere Hälfte des Messwertspeichers, ebenso die Korrekturdateien `No`=3 und `No`=4. Beide Optionen können daher nicht gleichzeitig verwendet werden. Falls vor **set_trigger4** mit **load_correction_file** eine Korrekturdatei als Korrekturdatei `No`=3 oder `No`=4 geladen wurde, überschreibt die Messwert-Aufzeichnung diesen Bereich und die Korrekturdatei ist nicht mehr verwendbar. Falls das Laden einer Korrekturdatei `No`=3 oder `No`=4 erfolgt, nachdem mit **set_trigger4** irgendwelche Daten-Quadrupel (oder mit **set_trigger** bereits mehr als 2^{19} Datenpaare) aufgezeichnet worden sind, sind die Werte für die Kanäle 3 und 4 (oder jenseits von 2^{19} Datenpaaren) nicht mehr verfügbar. • **load_program_file** stellt den Ursprungszustand des Messwertspeichers wieder her. • Zur Datenübertragung der aufgenommenen Werte an den PC kann wie bei **set_trigger** der Befehl **get_waveform** verwendet werden.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung (DLL 543, OUT 543, RBF 524): `Signal1, Signal2` = 52…54.
Verweise	**set_trigger**, **stop_trigger**, **get_waveform**

Unverzögerter kurzer Listenbefehl	**set_vector_control**	
Funktion	Initialisiert bzw. deaktiviert die vektorkontrollierte Lasersteuerung (siehe Seite 166).	
Aufruf	`set_vector_control( Ctrl, Value)`	
Parameter	`Ctrl`	Kontrollparameter (32-Bit-Wert ohne Vorzeichen) zur Initialisierung bzw. Deaktivierung der vektorkontrollierten Lasersteuerung (für `Ctrl` = 1…6 identisch zu **set_auto_laser_control**). = 1…7: Definiert, welcher Signalparameter durch die vektorkontrollierte Lasersteuerung 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 Lasersignale LASER1 und LASER2. = 5: Ausgabeperiode (`HalfPeriod`) der Lasersignale LASER1 und LASER2. = 6: Ausgabewert am 16-Bit-Digital-Ausgang. = 7: Fokusverschiebung. = 0 oder > 7: Deaktiviert die vektorkontrollierte Lasersteuerung (für `Ctrl` > 7: **get_last_error**-Returncode `RTC5_PARAM_ERROR`).
	`Value`	Legt den Anfangswert für den über `Ctrl` ausgewählten Parameter fest (32-Bit-Wert ohne Vorzeichen). Zulässige Werte (überlaufende Werte geclippt): Für `Ctrl` = 1/2: 12-Bit-Werte [0 … 4095]. 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 … (2^{16}–1)]. Für `Ctrl` = 7: [0 … 65.535], `Value` = Fokusverschiebung + 32.768.
Hinweise	• Bei ungültigem Wert für `Ctrl` wird die vektorkontrollierte Lasersteuerung 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. • Der Befehl wirkt sich nur auf Para-Mark- und Para-Jump-Befehle aus. • Wenn eine positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängige Lasersteuerung mit demselben Kontrollparameter aktiviert ist (`Ctrl` = 1…6), dann setzt **set_vector_control** den 100%-Wert der positions- und/oder geschwindigkeits- oder encodergeschwindigkeitsabhängigen Lasersteuerung auf den Wert `Value`.	

Unverzögerter kurzer Listenbefehl	set_vector_control
Hinweise (Forts.)	• Für `Ctrl` = 7 wird bei nachfolgenden parametrisierten 2D- und 3D-Mark- und Jump-Befehlen die Fokusverschiebung linear (wie mit **set_defocus** bzw. **set_defocus_list**, siehe Hinweise dort) bis zum angegebenen Endwert variiert (die 3D-Option muss dazu freigeschaltet und eine 3D-Korrekturdatei geladen und zugewiesen sein). Stimmt beim ersten parametrisierten Mark- oder Jump-Befehl die aktuell eingestellte Fokusverschiebung nicht mit dem unter **set_vector_control** spezifizierten Anfangswert `Value` überein, wird zu Beginn des parametrisierten Befehls ein harter Sprung in der z-Koordinate auf `Value` ausgeführt. Die eventuell mit **set_defocus** bzw. **set_defocus_list** festgelegte Einstellung geht verloren. Im Gegensatz zu **set_defocus** bzw. **set_defocus_list** muss bei **set_vector_control** für die Fokusverschiebung (`Value`) statt eines vorzeichenbehafteten Wertes im Bereich [–32.768 … 32.767] ein um 32.768 nach oben verschobener, vorzeichenloser Wert angegeben werden: `Value` = Fokusverschiebung + 32.768. • Die automatische Lasersteuerung sollte nicht zusammen mit der variablen Laserleistung aus **set_multi_mcbsp_in** oder der "Frei definierbaren Wobbelfigur" verwendet werden.
RTC4→ RTC5	Neuer Befehl. Es gibt für diesen Befehl keinen RTC4-Kompatibilitätsmodus für `Value`.
Verweise	**set_auto_laser_control**

Kontrollbefehl	set_verify
Funktion	Aktiviert bzw. deaktiviert eine Download-Überprüfung, siehe Kapitel 6.8.1 "Download-Überprüfung", Seite 99.
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	Einstellparameter `Verify`, der vor Aufruf des Befehls aktiv war (als 32-Bit-Wert ohne Vorzeichen)
Hinweise	• Die Download-Zeiten verlängern sich mit aktivierter Überprüfung. • Für eine komplette Download-Überprüfung, wie unter "Download-Überprüfung", Seite 99 beschrieben, ist die Treiberversion 1.0.4.0 oder höher erforderlich. Ältere DLLs funktionieren auch mit dem neuen Treiber, jedoch beschränkt sich hiermit die Download-Überprüfung auf den Listenbefehl-Download (DLL-Version 511), bzw. ist überhaupt nicht implementiert (DLL-Version < 511). Neuere DLLs (Version >511) liefern mit älteren Treiber-Versionen bei **load_program_file** den Fehlercode 9 (DriverCall-Fehler) zurück. • Die Überprüfung von Korrekturdatei-Downloads funktioniert nur bei Korrekturdateien, in die eine Check-Summe eingetragen ist (andernfalls wird der Fehlercode `RTC5_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. Dieser Befehl trägt die Check-Summe, sofern sie bisher noch nicht eingetragen ist, in die Korrekturdatei ein. • Die Befehle **set_verify** und **n_set_verify** sind auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. Diese Befehle ändern nur die Einstellungen in der DLL für die angegebene (bzw. Default-) Karte. Sie haben keine Auswirkungen auf die Karte selbst. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe "Error-Handling", Seite 98) werden von **set_verify** weder erzeugt noch verändert. • Ab DLL 535 kann **set_verify** beim Einschalten der Verify-Option einen **get_last_error**-Fehlercode `RTC5_OUT_OF_MEMORY` erzeugen, wenn eine DLL-interne Windows-Speicheranforderung fehlschlägt. In diesem Fall bleibt die Verify-Option abgeschaltet.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 511, OUT 510, RBF 504. • Änderung Version DLL 512: Erweiterte Download-Überprüfung. • Letzte Änderung Version DLL 535 (siehe Hinweis).
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 der Wartemarkierung als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [1 … (2^{32}–1)]. 0 wird auf 1 korrigiert.
Hinweise	• Die Ausführung einer Liste wird an jedem Unterbrechungspunkt unterbrochen und so lange angehalten, bis mit **release_wait** fortgefahren wird. Dadurch können Synchronisierungen realisiert werden. • Die "Laser active"-Lasersteuersignale werden durch **set_wait** ausgeschaltet und es wird ein evtl. mit **home_position** oder **home_position_xyz** definierter Home-Jump ausgeführt (für die Dauer des Home-Jumps ist der INTERNAL-BUSY-Status gesetzt). Ein Fortfahren mit **execute_list_pos**, **restart_list** oder mit einem externen Start ist in der Folge unterbunden. Ein **stop_execution** oder externer Listen-Stopp ist dagegen möglich. **release_wait**, **stop_execution** oder ein externer Listen-Stopp heben die Unterdrückung des Listen-Starts wieder auf. • Durch **set_wait** wird der (mit **get_status** abrufbare) PAUSED-Status gesetzt und der BUSY-Status rückgesetzt. Das umgekehrte geschieht bei einem anschließenden **release_wait**-Befehl (siehe auch "Listenausführungsstatus", Seite 77). • Wurde eine Ausführung an einem Unterbrechungspunkt unterbrochen, dann gibt der Befehl **get_wait_status** die Nummer dieses Unterbrechungspunkts zurück.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: Bei der RTC5 gibt es auch einen PAUSED-Status. Dieser wird durch **set_wait** gesetzt.
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 Seite 189).	
Aufruf	`set_wobbel( Transversal, Longitudinal, Freq )`	
Parameter	`Transversal`	Amplitude der ellipsenförmigen bzw. achtförmigen Bewegung *senkrecht* zur momentanen oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung in *Bits* (32-Bit-Wert ohne Vorzeichen). Zulässiger Bereich: [0 … 131.071 ($=2^{17}$-1)]. Zu große Werte werden geclippt.
	`Longitudinal`	Amplitude der ellipsenförmigen bzw. achtförmigen Bewegung *parallel* zur momentanen Bewegungsrichtung oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung in *Bits* (32-Bit-Wert ohne Vorzeichen). Zulässiger Bereich: [0 … 131.071 ($=2^{17}$-1)]. Zu große Werte werden geclippt.
	`Freq`	Frequenz der Wobbelbewegung in *Hz* (Anzahl der Ellipsen pro Sekunde, im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [–6000 … 6000]. Zu große Werte werden geclippt.
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). Auch bei Kreisbögen folgt die Wobbelbewegung stets (Ausnahme siehe unten) der momentanen Markierungsrichtung. Die effektive Linienbreite ist also unabhängig vom kartesischen Winkel immer gleich groß. Ab Version DLL 537 kann auch mit **set_wobbel_direction** eine beliebige “Bewegungsrichtung“ vorgegeben werden. Ebenso sind auch “Frei definierbare Wobbelfiguren“ möglich (siehe **set_wobbel_vector**). • 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. Bei identischen Amplituden (`Longitudinal = Transversal`) ist der Wobbelstartpunkt fest bezogen auf das Koordinatensystem, also unabhängig von der momentanen Bewegungsrichtung (wie bei RTC4).	

Verzögerter kurzer Listenbefehl	set_wobbel
Hinweise (Forts.)	• 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". • Ab Version DLL 543, OUT 543, RBF 524 kann die momentane Wobbelamplitude mit dem Triggersignal 53 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**). Das Format der Daten ist ((Tansversal << 16) + Longitudinal).
RTC4→ RTC5	Von der RTC4 wurden nur bzgl. des Koordinatensystems feste (elliptische) Wobbelbewegungen ausgeführt. Bei der RTC5 kann über die Parameter `Transversal` und `Longitudinal` eine ellipsenförmige Bewegung definiert werden (und über das Vorzeichen der Frequenz deren Drehrichtung), die der momentanen Bewegungsrichtung folgt. Die Bildfeldkoordinaten für die X- und Y-Achse werden bei der RTC5 als 20-Bit-Werte angegeben, im RTC4-Kompatibilitätsmodus dagegen wie bei der RTC4 als 16-Bit-Werte. Im RTC4-Kompatibilitätsmodus werden die angegebenen Werte für die Amplituden `Longitudinal` und `Transversal` von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [0 … 8191]).
Verweise	**set_mark_speed**, **set_wobbel_mode**

Unverzögerter kurzer Listenbefehl	**set_wobbel_control**
Funktion	Definiert die Lasersteuerungsparameter 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: = 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 `RTC5_PARAM_ERROR`). `Value` Nominelle Laserleistung P0 (100%) als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich siehe **set_auto_laser_control**, maximal jedoch [0 … 65535] oder 0xFFFFFFFF. Zu große Werte werden geclippt. `MinValue`, `MaxValue` Grenzwerte, die nicht überschritten werden können, als 32-Bit-Werte ohne Vorzeichen (siehe **set_auto_laser_control**). Die zulässigen Wertebereiche hängen vom ausgewählten `Ctrl`-Parameter und von `Value` ab.
Hinweise	• Ein evtl. noch anliegender verzögerter kurzer Listenbefehl wird noch davor ausgeführt. • Für `Ctrl` = 0 und `Ctrl` > 6 wird die Laserleistungsvariation abgeschaltet (Initialisierungszustand nach **load_program_file**). Die Werte aus der Multi-McBSP/SPI-Übertragung werden dann auch nicht verwendet. • Für `Value`, `MinValue`, `MaxValue` gelten dieselben Bedingungen wie bei der automatischen Lasersteuerung (siehe **set_auto_laser_control**), maximal jedoch [0 … 65535]. Initialisierungswerte sind `MinValue` = 0 und `MaxValue` = 0xFFFFFFFF, also keinerlei Einschränkung. • Die Laserleistung kann mit der vektorkontrollierten Lasersteuerung kombiniert werden, wenn die entsprechenden `Ctrl`-Parameter gleich gewählt sind. • Wenn eine entlang der Wobbelfigur variierende Laserleistung gewünscht ist, sollte der Befehl noch vor der Aktivierung der "Frei definierbaren Wobbelfigur" ausgeführt werden. Andernfalls wird keine Laserleistung moduliert, auch dann nicht, wenn in den Datensätzen eine Änderung angegeben ist. • Sonderfall: Für `Value` = 0xFFFFFFFF wird die nominelle Laserleistung nicht aus dem Befehl 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 – und andere gegebenenfalls anstehende verzögerte kurze Listenbefehle – werden noch vor dem Befehl **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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_wobbel_vector**, **set_multi_mcbsp_in**, **set_multi_mcbsp_in_list**

Unverzögerter kurzer Listenbefehl	**set_wobbel_direction**
Funktion	Definiert einen Richtungsvektor für Wobbelbewegungen.
Aufruf	`set_wobbel_direction( dX, dY )`
Parameter	`dX`, `dY` Komponenten des Richtungsvektors in Bits als 32- Bit-Werte mit Vorzeichen.
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. Der Richtungsvektor wird von der RTC5 normiert. • 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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_wobbel**, **set_wobbel_mode**

Verzögerter kurzer Listenbefehl	**set_wobbel_mode**	
Funktion	Schaltet den Wobbelmode ein oder aus und legt die Parameter für eine ellipsenförmige bzw. achtförmige Wobbelbewegung fest (siehe Seite 189).	
Aufruf	`set_wobbel_mode( Transversal, Longitudinal, Freq, Mode )`	
Parameter	`Transversal`	Amplitude der ellipsenförmigen bzw. achtförmigen Bewegung *senkrecht* zur momentanen oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung in *Bits* (32-Bit-Wert ohne Vorzeichen). Zulässiger Bereich: [0 … 131.071 (=2^{17}-1)]. Zu große Werte werden geclippt.
	`Longitudinal`	Amplitude der ellipsenförmigen bzw. achtförmigen Bewegung *parallel* zur momentanen Bewegungsrichtung oder der mit **set_wobbel_direction** festgelegten Bewegungsrichtung in *Bits* (32-Bit-Wert ohne Vorzeichen). Zulässiger Bereich: [0 … 131.071 (=2^{17}-1)]. Zu große Werte werden geclippt.
	`Freq`	Frequenz der Wobbelbewegung in *Hz* (Anzahl der Ellipsen bzw. Achter pro Sekunde, im 64-Bit-IEEE-Gleitkommaformat) Zulässiger Bereich: [–6000 … 6000]. Zu große Werte werden geclippt.
	`Mode`	Auswahl der 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”
Hinweise	• Für `Mode` = 0 funktioniert der Befehl **set_wobbel_mode** identisch wie der **set_wobbel**-Befehl (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 des Befehls in einer Liste eine “Frei definierbare Wobbelfigur” definiert ist (siehe **set_wobbel_vector**, und Kapitel 8.4 “Wobbelmode”, Seite 189), 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. • 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…).	

Verzögerter kurzer Listenbefehl	set_wobbel_mode
Hinweise (Forts.)	• 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 (wie bei RTC4). • 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. • 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. Sie wird beim nächsten Einschalten des Wobbelmodes mit `Mode` > 1 wieder verwendet. • 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". • Ab Version DLL 543, OUT 543, RBF 524 kann die momentane Wobbelamplitude mit dem Triggersignal 53 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**). Das Format der Daten ist ((Tansversal << 16) + Longitudinal). • Der Wobbelmode kann nicht kombiniert werden mit: – Sky-Writing – Pixelausgabemodus – Jumps – **laser_on_list**
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Werte für die Amplituden `Longitudinal` und `Transversal` von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [0 … 8191]).
Versionsinfo	Letzte Änderung mit Version DLL 537, OUT 537: `Mode`s > 0.
Verweise	**set_wobbel**, **set_wobbel_control**, **set_wobbel_vector**

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`, `OffsetLong` Transversaler und longitudinaler Offset als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: ±32767. Zu große Werte werden geclippt. Initialisierungswerte nach **load_program_file**: (0,0).
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 189) anwendbar. • Die gesamte Wobbelamplitude darf incl. 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 Sprungbefehl (analog zu longitudinalen Amplituden bei den “klassischen” Wobbelfiguren).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus multipliziert die RTC5 die Offsets mit 16. Der zulässige Wertebereich reduziert sich entsprechend auf ±2047.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
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`, `dLong` Mikroschritt eines linearen Wobbelfigurabschnitts *in Bits* im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich [–256.0 … +255.0]. `dTrans` ist die Mikrovektorkomponente quer zur Bewegungsrichtung (das ist die Lasertrajektorie oder die mittels **set_wobbel_direction** definierte Bewegungsrichtung). `dLong` ist entsprechend längs dazu. `Period` 32- Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … 65535]. = 1 … 65535: Anzahl der Mikroschritte = 0: die Wobbelfigur wird abgeschaltet `dPower` Mikroschritt der relativen Laserleistung im 64-Bit-IEEE-Gleitkommaformat. Zulässiger Wertebereich [–1.0 … +1.0]. Unzulässige Werte werden auf die jeweiligen Randbereiche geclippt.
Hinweise	• **set_wobbel_vector** kann nicht gleichzeitig mit der Softstart-Funktion verwendet werden, da beide Funktionen einen gemeinsamen Speicherbereich auf der RTC5 benutzen. Um (ungewollte) gegenseitige Überschreibungen zu vermeiden, muss der Softstart unbedingt explizit deaktiviert sein. Andernfalls bleibt der Befehl ohne Wirkung. **set_wobbel_vector** blockiert seinerseits nachträgliche Veränderungen des Speichers mittels **set_softstart_level** oder **set_softstart_level_list**. Diese Befehle bleiben ohne Wirkung, solange die Wobbelfigur nicht explizit abgeschaltet ist. • `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 512 Wobbelfigurenabschnitte definiert werden. Nach 512 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, sofern 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 ggf. mit einem harten 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. • Sofern mit **set_wobbel_control** aktiviert, variiert die RTC5 die aktuelle Laserleistung P im Wobbelfigurabschnitt gemäß $P = P0 \times (1 + \texttt{dPower} \times n)$, mit $0 \le n \le \texttt{Period}$. n bezeichnet die laufende Nummer des jeweiligen Mikroschrittes. Die nominelle Laserleistung P0 wird über den Listenbefehl **set_wobbel_control** eingestellt. • Sofern mit **set_multi_mcbsp_in** oder **set_multi_mcbsp_in_list** aktiviert, wird die nominelle Laserleistung P0 multiplikativ über den Laserleistungsparameter `P` der mehrfachen McBSP/SPI-Übertragung verändert: $P0_{McBSP} = P0 \times \texttt{P} / 16384$. • Dabei ist zu beachten, dass für jede der beiden Korrekturen nur eine maximale Laserleistung von P0 × 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 Lasersteuerung kann der Multiplikationsfaktor nicht invertiert werden. • Mit einer "leeren" Wobbelfigur `set_wobbel_vector(0.0, 0.0, 1, 0.0)` kann die nominelle Laserleistung P0 auch multiplikativ verändert werden, ohne explizit wobbeln zu müssen (für eine weitere Alterative siehe auch **set_multi_mcbsp_in**). • Beim Ausschalten der Wobbelfigur (nicht beim Deaktivieren über **set_wobbel_mode**) wird die nominelle Laserleistung P0 an dem über `Ctrl` zugeordneten Port ausgegeben, sofern **set_wobbel_control** zuletzt mit `Value` = 0xFFFFFFFF aufgerufen wurde, ansonsten die mit dem externen Faktor aus **set_multi_mcbsp_in** multiplizierte Laserleistung P0. • 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 Lasersteuerung oder der vektorkontrollierten Lasersteuerung (parametrisierte Markbefehle) kombiniert werden. Die Wobbelvariation überschreibt die anderen Variationen, sofern 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".
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 537, OUT 537.
Verweise	**set_multi_mcbsp_in**, **set_multi_mcbsp_in_list**, **set_wobbel_control**

Kontrollbefehl	**simulate_encoder**
Funktion	Aktiviert bzw. deaktiviert die Encoder-Simulation 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 Encoder-Zähler Encoder0 hochgezählt. = 2: Es werden ENCODER Y-Pulse simuliert und damit Encoder-Zähler Encoder1 hochgezählt. = 3: Es werden ENCODER X- *und* ENCODER Y-Pulse simuliert. = 0: Die Encoder-Simulation wird deaktiviert.
Hinweise	• Die Encoder-Simulation erfolgt über einen internen Takt von 1 MHz (siehe auch "Encoder-Simulation" auf Seite 247). • **simulate_encoder** führt keinen Reset der Encoder-Zähler aus. • Für `EncoderNo` > 3 wird **simulate_encoder** ignoriert (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).
RTC4→ RTC5	Unverändert.
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 Listen-Start aus.
Aufruf	`simulate_ext_start( Delay, EncoderNo )`
Parameter	`Delay` Streckenverzögerung (Zählerschritte des mit `EncoderNo` ausgewählten Encoder-Zählers) als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: $[-2^{31} \ldots +(2^{31}-1)]$. `EncoderNo` Nummer des zu verwendenden Encoder-Zählers als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0 = 1: Encoder-Zähler Encoder1
Hinweise	• Zu externen Listen-Starts siehe Seite 240. • Die Streckenverzögerung wird in (relativen) Zähleinheiten des ausgewählten Encoder-Zählers angegeben (die RTC5-Encoder-Zähler werden durch ein externes oder simuliertes Encoder-Signal getriggert, siehe Seite 246). Der Start-Trigger für den Listen-Start erfolgt erst, wenn der interne Encoder-Zähler um die angegebene Streckenverzögerung hoch (bzw. nach unten) gezählt wurde. Externe Listen-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 Listen-Start aus, wenn er nicht erfolgt, während der BUSY-Status (wie bei Listenausgabe), der INTERNAL-BUSY-Status (wie z. B. bei **goto_xy**) oder/und der PAUSED-Status (nach **pause_list**, **stop_list** oder **set_wait**) gesetzt ist. Sonst wird Bit#11 des **get_startstop_info**-Rückgabewertes gesetzt. Wird eine ungeeignete Streckenverzögerung (z. B. `Delay` = 0) angegeben, so wird daher kein Listen-Start ausgelöst. Wird der Befehl **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 Listen-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* Encoder-Reset 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 Encoder-Reset aus, falls er auf einen der "Processing on the fly"-Befehle **set_fly_x**, **set_fly_y** oder **set_fly_rot** folgt. • Für `EncoderNo` > 1 wird **simulate_ext_start** durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).

Normaler Listenbefehl	simulate_ext_start
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich). Im RTC4-Kompatibilitätsmodus wird der angegebene `Delay`-Wert von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
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 Listen-Start aus.
Aufruf	`simulate_ext_start_ctrl()`
Hinweise	• Zu externen Listen-Starts siehe Seite 240. • Der Start-Trigger für den Listen-Start erfolgt erst nach einer gegebenenfalls zuvor mit **set_ext_start_delay**, **set_ext_start_delay_list** oder **simulate_ext_start** eingestellten Streckenverzögerung (d.h. wenn der mit diesen Befehlen ausgewählte interne Encoder-Zähler um die angegebene Streckenverzögerung hoch bzw. nach unten gezählt wurde). Bei der Initialisierung (mit **load_program_file**), durch einen externen Listen-Stopp und durch **stop_execution** wird die Steckenverzögerung deaktiviert. Sie kann auch mit **set_control_mode** (Bit#2) deaktiviert werden. Externe Listen-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 Listen-Start aus, wenn er nicht während der Ausgabe einer Liste erfolgt (sonst wird Bit#11 des **get_startstop_info**-Rückgabewertes gesetzt). • **simulate_ext_start_ctrl** führt *keinen* Encoder-Reset 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 Encoder-Reset aus, falls er auf einen der "Processing on the fly"-Befehle **set_fly_x**, **set_fly_y** oder **set_fly_rot** folgt. • Ab Version DLL 543, OUT 543 kann **simulate_ext_start_ctrl** mit **set_control_mode** Bit #4=1 oder **set_control_mode_list** Bit #4=1 deaktiviert werden.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 515, OUT 514, RBF 512.
Verweise	**simulate_ext_start**

Kontrollbefehl	simulate_ext_stop
Funktion	Löst einen simulierten externen Listen-Stopp aus.
Aufruf	`simulate_ext_stop()`
Hinweise	• Zu externen Listen-Stopps siehe Seite 239. • Durch **simulate_ext_stop** werden sowohl die Master-Karte als auch alle Slave-Karten gleichzeitig angehalten, mit **stop_execution** dagegen nur die Master-Karte.
RTC4→ RTC5	Neuer Befehl.
Verweise	**simulate_ext_start**, **simulate_ext_start_ctrl**, **stop_execution**

Kontrollbefehl	start_loop
Funktion	Aktiviert einen sich wiederholenden, automatischen Listenwechsel.
Aufruf	`start_loop()`
Hinweise	• Ein mit **start_loop** aktivierter Listenwechsel bzw. Listenneustart wird so lange wiederholt, bis die Ausführung durch Aufruf des Befehls **quit_loop** beendet wird. • Der Befehl **start_loop** kann zu jedem beliebigen Zeitpunkt erteilt werden, wird aber erst mit dem nächsten **set_end_of_list**-Befehl 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 "Automatischer Listenwechsel", Seite 79. • Damit der **start_loop**-Befehl überhaupt wirkt, muss die gerade aktive Liste unbedingt mit **set_end_of_list** abgeschlossen worden sein, die neue Liste sollte bereits geladen sein bzw. der Input-Pointer einen gewissen Vorsprung vor dem Output-Pointer haben (andernfalls 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 der **start_loop**-Befehl *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 der **start_loop**-Befehl. • Die aktuellen Listenstatuswerte und Listenausführungsstatuswerte können mit den Befehlen **read_status** und **get_status** gelesen werden. • Der **start_loop**-Befehl löst einen Flush der gepufferten Listeneingabe aus (siehe Seite 75).
RTC4→ RTC5	Unverändert.
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`, `Pos2`	Absolute Zielposition in CLOCK-Puls-Einheiten für Schrittmotor-Ausgang 1 bzw. 2 als 32-Bit-Werte mit Vorzeichen. Zulässiger Bereich: [-2^{31}… $+(2^{31}-1)$].
	`WaitTime`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die Funktion spätestens zurückkehrt. *1 Bit entspricht 1 s*. Zulässiger Bereich: [0 … $+(2^{32}-1)$].
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoransteuerung", Seite 234. • **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 Hinweis Seite 234). 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: `RTC5_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 der Befehl sofort zurück, damit die Anwendung sofort wieder die Kontrolle über das System hat.	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).	
Verweise	**stepper_abs_no**, **stepper_rel**, **stepper_rel_no**, **stepper_abs_list**	

Unverzögerter kurzer Listenbefehl	**stepper_abs_list**
Funktion	Identisch mit **stepper_abs**, aber ein Listenbefehl und ohne `WaitTime`-Parameter.
Aufruf	`stepper_abs_list( Pos1, Pos2 )`
Parameter	`Pos1`, `Pos2` Siehe **stepper_abs**.
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→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).
Verweise	**stepper_abs**

Kontrollbefehl	**stepper_abs_no**	
Funktion	Löst am gewünschten 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 der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`).
	`Pos`	Absolute Zielposition in CLOCK-Puls-Einheiten als 32-Bit-Wert mit Vorzeichen. Zulässiger Bereich: $[-2^{31} \ldots +(2^{31}-1)]$.
	`WaitTime`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die Funktion spätestens zurückkehrt. *1 Bit entspricht 1 s*. Zulässiger Bereich: $[0 \ldots +(2^{32}-1)]$.
Hinweise	• Es wird nur am mit `No` angegebenen Schrittmotor-Ausgang eine Zielpositionsfahrt ausgeführt. Ansonsten ist der Befehl identisch mit **stepper_abs** (siehe Hinweise dort).	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	• Siehe **stepper_abs**.	
Verweise	**stepper_abs**, **stepper_abs_no_list**	

Unverzögerter kurzer Listenbefehl	**stepper_abs_no_list**
Funktion	Identisch mit **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 der Befehl schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).
	`Pos` Siehe **stepper_abs_no**.
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→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).
Verweise	**stepper_abs_no**

Kontrollbefehl	**stepper_control**
Funktion	Setzt die Pulsperioden für die CLOCK-Signale der Schrittmotorsteuerung.
Aufruf	`stepper_control( Period1, Period2 )`
Parameter	`Period1, Period2` Pulsperiode für die CLOCK-Signale von Schrittmotor-Ausgang 1 bzw. 2 als 32-Bit-Werte mit Vorzeichen. *1 Bit entspricht 10 µs*. Zulässiger Bereich: [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.
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Seite 234. • `Period` = 0 kann für einen Not-Halt genutzt werden (siehe auch Abschnitt "Anhalten von Endlosfahrten", Seite 236).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).
Verweise	**stepper_control_list**

Unverzögerter kurzer Listenbefehl	stepper_control_list
Funktion	Identisch mit **stepper_control**, aber ein Listenbefehl.
Aufruf	`stepper_control_list( Period1, Period2 )`
Parameter	`Period1`, `Period2` Siehe **stepper_control**.
Hinweise	• Siehe **stepper_control**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Return-code: `RTC5_TYPE_REJECTED`).
Verweise	**stepper_control**

Kontrollbefehl	stepper_disable_switch
Funktion	Steuert die Verwendung der SWITCH-Signale der Schrittmotorsteuerung.
Aufruf	`stepper_disable_switch( Disable1, Disable2 )`
Parameter	`Disable1`, `Disable2` Anweisung, wie die SWITCH-Signale aus Schrittmotor-Eingang 1 und 2 verwendet werden sollen. Als 32-Bit-Werte mit Vorzeichen. Zulässiger Bereich: [-2^{32} … $+(2^{32}-1)$]. > 0: Das SWITCH-Signal am jeweiligen Schrittmotor-Eingang wird nicht verwendet. = 0: Das SWITCH-Signal am jeweiligen Schrittmotor-Eingang wird verwendet. < 0: Die Verwendung des jeweiligen Schrittmotor-Eingangssignals bleibt unverändert.
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoransteuerung", Seite 234. • 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→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 542, OUT 542.
Verweise	**stepper_init**

Kontrollbefehl	stepper_enable
Funktion	Verändert die ENABLE-Signale der Schrittmotorsteuerung.
Aufruf	`stepper_enable( Enable1, Enable2 )`
Parameter	`Enable1, Enable2` ENABLE-Signal für Schrittmotor-Ausgang 1bzw. 2 als 32-Bit-Werte mit Vorzeichen. Zulässiger Bereich: [-2^{32} … $+(2^{32}-1)$]. > 0: Das ENABLE-Signal am jeweiligen Schrittmotor-Ausgang wird gesetzt. = 0: Das ENABLE-Signal am jeweiligen Schrittmotor-Ausgang wird zurückgesetzt, < 0: das jeweilige Schrittmotor-Ausgangssignal bleibt unverändert.
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Seite 234.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).
Verweise	**stepper_enable_list**

Unverzögerter kurzer Listenbefehl	stepper_enable_list
Funktion	Identisch mit **stepper_enable**, aber ein Listenbefehl und ohne Rückgabe.
Aufruf	`stepper_enable_list( Enable1, Enable2 )`
Parameter	`Enable1, Enable2` Siehe **stepper_enable**.
Hinweise	• Siehe **stepper_enable**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).
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 der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`).
	`Period`	Pulsperiode für die CLOCK-Signale als 32-Bit-Wert ohne Vorzeichen. *1 Bit entspricht 10 µs*. Zulässiger Bereich: [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 und Zielpositionswert in CLOCK-Puls-Einheiten als 32-Bit-Wert mit Vorzeichen (siehe Hinweise). Zulässiger Bereich: [–2^{31}… +(2^{31}-1)].
	`Tol`	Toleranzwert für die Referenzfahrt (siehe Hinweise) als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [1 … +(2^{32}-1)]. Für `Dir` < 0 und/oder `Period` = 0 ist der Wert `Tol` = 0 irrelevant, ansonsten wird für `Tol` = 0 der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_PARAM_ERROR`).
	`Enable`	ENABLE-Signal als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [0 … +(2^{32}-1)]. = 0: Das ENABLE-Signal wird zurückgesetzt. > 0: Das ENABLE-Signal wird gesetzt.
	`WaitTime`	Mit diesem Parameter (32-Bit-Wert ohne Vorzeichen) wird festgelegt, wann die Funktion spätestens zurückkehrt (siehe Hinweise) *1 Bit entspricht 1 s*. Zulässiger Bereich: [0 … +(2^{32}-1)]

Kontrollbefehl	stepper_init
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoransteuerung", Seite 234. • **stepper_init** stoppt unmittelbar jede zuvor gestartete Schrittmotor-Bewegung des mit `No` angegebenen Schrittmotors. • Das ENABLE-Signal 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 mehrmals 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. Sowohl die interne Positionsvariable (für die aktuelle Position) als auch der interne Zielpositionswert werden auf den über den Parameter `Pos` festgelegten Wert gesetzt. Dieser Wert entspricht also einer gegenüber dem Endschalter um `Tol` verschobenen Position. – Wird *kein* Endschalter gefunden (z.B. weil in der angegebenen Richtung gar kein Endschalter vorhanden ist), so führt der Schrittmotor eine Endlosfahrt aus. Der Befehl kehrt dann zwar 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**-Befehl oder mit **stepper_control**(`Period` = 0) gestoppt wird. Siehe hierzu auch Abschnitt "Anhalten von Endlosfahrten", Seite 236. • Für `Period` = 0, `Dir` < 0 und/oder `WaitTime` = 0 kehrt der Befehl 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. • `Period` = 0 und/oder `Dir` < 0 können für einen Not-Halt genutzt werden. Es wird dann gegebenenfalls eine zuvor gestartete Schrittmotor-Bewegung gestoppt, aber keine Referenzfahrt ausgeführt. Auch hier werden Positionsvariable und Zielpositionswert auf den Wert `Pos` gesetzt. Dieser Wert entspricht dann der jeweils aktuellen Schrittmotor-Position. Das DIRECTION-Signal bleibt unverändert. • Für `Period` = 0 werden Init- und Busy-Status zurückgesetzt. Es werden keine weiteren Clock-Pulse mehr ausgegeben, bis `Period` wieder auf einen positiven Wert gesetzt wird (dies beeinflusst dann auch nachfolgende Zielpositions-Befehle wie **stepper_abs** oder **stepper_rel**).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl nicht ausgeführt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).

Kontrollbefehl	stepper_rel
Funktion	Löst an den beiden Schrittmotor-Ausgängen Zielpositionsfahrten zu den angegebenen relativen Zielpositionen aus.
Aufruf	`stepper_rel( dPos1, dPos2, WaitTime )`
Parameter	`dPos1`, `dPos2` Relative Zielposition in CLOCK-Puls-Einheiten für Schrittmotor-Ausgang 1 bzw. 2 als 32-Bit-Werte mit Vorzeichen. Zulässiger Bereich: [-2^{31}… $+(2^{31}-1)$].
	`WaitTime` Siehe **stepper_abs**.
Hinweise	• Die gewünschten Zielpositionen werden relativ zu den aktuellen internen Zielpositionswerten angegeben (die internen Zielpositionswerte werden entsprechend neu gesetzt). Ansonsten ist der Befehl identisch mit **stepper_abs** (siehe Hinweise dort).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Siehe **stepper_abs**.
Verweise	**stepper_abs**, **stepper_rel_list**

Unverzögerter kurzer Listenbefehl	stepper_rel_list
Funktion	Identisch mit **stepper_rel**, aber ein Listenbefehl und ohne `WaitTime`-Parameter.
Aufruf	`stepper_rel_list( dPos1, dPos2 )`
Parameter	`dPos1`, `dPos2` Siehe **stepper_rel**.
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→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).
Verweise	**stepper_rel**

Kontrollbefehl	stepper_rel_no	
Funktion	Löst am gewünschten Schrittmotor-Ausgang eine Zielpositionsfahrt zur angegebenen relativen Zielposition aus.	
Aufruf	`stepper_rel_no( No, dPos, WaitTime )`	
Parameter	`No`	Siehe **stepper_abs_no**.
	`dPos`	Relative Zielposition in CLOCK-Puls-Einheiten als 32-Bit-Wert mit Vorzeichen. Zulässiger Bereich: [-2^{31}... $+(2^{31}-1)$].
	`WaitTime`	Siehe **stepper_abs_no**.
Hinweise	• Die gewünschte Zielposition wird relativ zum aktuellen internen Zielpositionswert angegeben (der interne Zielpositionswert wird entsprechend neu gesetzt) und es wird nur am mit `No` angegebenen Schrittmotor-Ausgang eine Zielpositionsfahrt ausgeführt. Ansonsten ist der Befehl identisch mit **stepper_abs** (siehe Hinweise dort).	
RTC4→ RTC5	Neuer Befehl.	
Versionsinfo	• Siehe **stepper_abs**.	
Verweise	**stepper_abs_no**, **stepper_abs**, **stepper_rel_no_list**	

Unverzögerter kurzer Listenbefehl	stepper_rel_no_list	
Funktion	Identisch mit **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 der Befehl schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).
	`dPos`	Siehe **stepper_rel_no**.
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→ RTC5	Neuer Befehl.	
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).	
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.
Hinweise	• Zur Programmierung der Schrittmotorsignale siehe Kapitel 9.1.5 "Schrittmotoransteuerung", Seite 234. • Falls für den angegebenen Schrittmotor-Ausgang zuvor *keine* Schrittmotor-Bewegung gestartet wurde, benötigt der Befehl trotzdem 10 µs zur Ausführung, bleibt aber ansonsten ohne Auswirkung. • **stepper_wait** beeinflusst weder die "Laser active"-Lasersteuersignale, noch den Listen- oder Listenausführungsstatus.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 527, OUT 529, RBF 519. • Bei älteren RTC5-Karten mit DSP-Versionsnummer < 2 (**get_rtc_version** Bits #16-23) wird der Befehl weder ausgeführt noch durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_TYPE_REJECTED`).

Kontrollbefehl	stop_execution
Funktion	Beendet die Ausführung der Liste sofort und deaktiviert ohne Verzug die "Laser active"-Lasersteuersignale.
Aufruf	`stop_execution()`
Hinweise	• Mit Versionen OUT 534 und früher wird der Befehl nur ausgeführt, wenn eine Liste BUSY oder PAUSED ist (sonst **get_last_error**-Returncode: `RTC5_BUSY`). Ab Version OUT 535 deaktiviert **stop_execution** die "Laser active"-Lasersteuersignale dagegen auch dann, wenn keine Liste aktiv ist (der Befehl hat aber dann ansonsten keine Auswirkungen; auch hier gilt: **get_last_error**-Returncode: `RTC5_BUSY`). • Die Spiegel 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 Spiegel deshalb mit **goto_xy** in eine definierte Stellung gedreht werden. • Die externe START-Eingabe wird deaktiviert (siehe "Externer Listen-Start", Seite 240). • Die "Processing on the fly"-Korrektur wird ausgeschaltet (nur bei der "Processing on the fly"-Option). • Die BUSY-Listenstatuswerte (siehe **read_status**) und der BUSY-Listenausführungsstatus (siehe **get_status**) werden rückgesetzt. • Eine Liste, die mit **stop_execution** unterbrochen wurde, kann *nicht* fortgesetzt werden, sondern muss mit **execute_list_pos** (o.ä.) neu gestartet werden. Soll eine Liste nur vorübergehend unterbrochen und später fortgesetzt werden, so können die Befehle **pause_list** oder **stop_list** verwendet werden. • **stop_execution** wirkt nur auf die angesprochene Karte und wird auch in einer Master/Slave-Kette nicht zu den nachgeschalteten Slave-Karten durchgereicht. Sollen alle Karten einer Master/Slave-Kette gleichzeitig angehalten werden, so muss ein **simulate_ext_stop** oder ein externer Listen-Stopp an die Master-Karte ausgegeben werden (siehe auch Seite 93).
RTC4→ RTC5	Unverändert.
Versionsinfo	Letzte Änderung mit Version OUT 535 (siehe Hinweis).
Verweise	**get_startstop_info**

Kontrollbefehl	stop_list
Funktion	Unterbricht die Ausführung einer Liste und deaktiviert die "Laser active"-Lasersteuersignale.
Aufruf	`stop_list()`
Hinweise	• Der Befehl **stop_list** ist synonym zum Befehl **pause_list** – siehe Hinweise dort.
RTC4→ RTC5	Prinzipiell unveränderte Funktionalität, allerdings: bei der RTC5 gibt es auch einen PAUSED-Status. Dieser wird durch **stop_list** gesetzt.
Verweise	**pause_list**

Kontrollbefehl	stop_trigger
Funktion	Setzt den mit **measurement_status** auslesbaren Status der Messwertspeicherung zurück (auf `Busy`= 0).
Aufruf	`stop_trigger()`
Hinweise	• Dieser Befehl wird nur benötigt, wenn mit **set_trigger** oder **set_trigger4** eine Messwertspeicherung gestartet, aber nicht anschließend mit einem weiteren **set_trigger**-Befehl (`Period` = 0) oder **set_trigger4**-Befehl (`Period` = 0) beendet wurde. Der Status der Messwertspeicherung kann damit auch außerhalb einer aktiven Liste zurückgesetzt werden (siehe Hinweis bei **set_trigger**). • Der Befehl wird nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn der BUSY-Status der angesprochenen Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.
Verweise	**measurement_status**, **set_trigger**, **set_trigger4**

Unverzögerter kurzer Listenbefehl	store_encoder
Funktion	Speichert die aktuellen Zählerstände der beiden RTC5-Encoder-Zähler in einen Zwischenspeicher der RTC5.
Aufruf	`store_encoder( Pos )`
Parameter	`Pos` Auswahl der 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.
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 die zwischengespeicherten Werte anschließend mit **read_encoder** ausgelesen werden. • Der Zwischenspeicher wird ausschließlich zur Speicherung von Encoderwerten verwendet. • Siehe auch "“Processing on the fly” (optional)", Seite 199 und "Synchronisation durch Encoder-Signale", Seite 246.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 520, OUT 519.
Verweise	**read_encoder**, **get_encoder**, **set_fly_x**, **set_fly_y**, **set_fly_rot**, **wait_for_encoder**

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].
Hinweise	• Der Befehl **sub_call** liest die Startadresse des indizierten Unterprogramms entsprechend dem ausgewählten Index aus der internen Verwaltungstabelle und ruft dann den Befehl **list_call** auf (siehe auch Hinweise dort), der dann den Sprung auf das Unterprogramm auslöst. • Der Befehl **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 wird mit dem der aufrufenden Stelle unmittelbar folgenden Befehl fortgefahren. Evtl. wird ein **list_nop** 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" (d.h. 2^{32}–1) zurück. • für `Index` > 1023 wird **sub_call** schon beim Laden durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Absolute Vektor- und Bogenbefehle 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-, Bogen- oder Jump-Befehle enthalten oder es ist der Befehl **sub_call_abs** zu verwenden.
RTC4→ RTC5	Neuer Befehl.
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- und Bogenbefehle (entsprechend der aktuellen Koordinaten zum Zeitpunkt des Aufrufs) mit einem Offset beaufschlagt.
Aufruf	`sub_call_abs( Index )`
Parameter	`Index` Index des aufzurufenden indizierten Unterprogramms als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … 1023].
Hinweise	• Der Befehl **sub_call_abs** liest die Startadresse des indizierten Unterprogramms entsprechend dem ausgewählten Index aus der internen Verwaltungstabelle und ruft dann den Befehl **list_call_abs** auf (siehe auch Hinweise dort), der dann den Sprung auf das Unterprogramm auslöst. • Enthält ein aufzurufendes Unterprogramm keine absoluten Befehle, besteht zwischen **sub_call_abs** und **sub_call** kein Unterschied.
RTC4→ RTC5	Neuer Befehl.
Verweise	**sub_call**, **sub_call_abs_cond**

Unverzögerter kurzer Listenbefehl	**sub_call_abs_cond**
Funktion	*Bedingter Aufruf (AbsCall) eines indizierten Unterprogramms*: Dieser Befehl führt den Befehl **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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`sub_call_abs_cond( Mask1, Mask0, Index )`
Parameter	`Mask1, Mask0` 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Index` Index des aufzurufenden indizierten Unterprogramms als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … 1023].
Hinweise	• Siehe **sub_call_abs**. • Siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Neuer Befehl.
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 als 32-Bit-Wert ohne Vorzeichen.
Hinweise	• Der Befehl `sub_call_abs( Index )` ist synonym zu `sub_call_abs_repeat( Index, 1 )`. • Siehe **sub_call_repeat** und **sub_call_abs**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 538, OUT 538.
Verweise	**sub_call_repeat**, **sub_call_abs**, **sub_call**

Unverzögerter kurzer Listenbefehl	**sub_call_cond**
Funktion	*Bedingter Aufruf eines indizierten Unterprogramms*: Dieser Befehl führt den Befehl **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)` (d.h. falls die in `Mask1` gesetzten Bits von `IOvalue` gleich `1` und die in `Mask0` gesetzten Bits von `IOvalue` gleich `0` sind). Andernfalls wird direkt der nächste Listenbefehl ausgeführt.
Aufruf	`sub_call_cond( Mask1, Mask0, Index )`
Parameter	`Mask1`, `Mask0` 16-Bit-Masken als 32-Bit-Werte ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet. `Index` Index des aufzurufenden indizierten Unterprogramms als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … 1023]
Hinweise	• Siehe **sub_call**. • Siehe auch Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.
RTC4→ RTC5	Neuer Befehl.
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.
Hinweise	• Der Befehl `sub_call( Index )` ist synonym zu `sub_call_repeat( Index, 1 )`. • Der Befehl **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 dem Befehl **sub_call_repeat** können z. B. Trajektorien für Anlauf- und Auslaufkurven aus Mikrovektor-Befehlen mit Figuren aus Unterprogrammen nahtlos aneinander gefügt werden.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 538, OUT 538.
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). Zulässige Werte: [1 … 16]. `ShiftBits` Nummer des niederwertigsten auszuwertenden Bits des 16 Bit-Digital-Eingangs (als 32-Bit-Wert ohne Vorzeichen). Zulässige Werte: [0 … 15].
Hinweise	• Für unzulässige Werte von `MaskBits` oder `ShiftBits` und für (`MaskBits` + `ShiftBits`) > 16 wird der Befehl durch **list_nop** ersetzt (**get_last_error**-Returncode: `RTC5_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 "16-Bit-Digital-Eingang und -Ausgang", Seite 51 und Kapitel 9.3.2 "Bedingte Befehlsausführung", Seite 244.
Beispiel (Pascal)	• Zur Laufzeit betrage der aktuelle Wert am 16 Bit-Digital-Eingang $F152. **switch_ioport**($0008, $0004) führt dann **list_jump_rel**($0016), also einen relativen Listensprung mit einer Sprungweite von 22 Listenpositionen aus.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version DLL 517, OUT 516.
Verweise	**list_jump_rel**, **list_jump_rel_cond**

Kontrollbefehl	sync_slaves
Funktion	Synchronisiert alle Slave-Karten, die in einer Master/Slave-Kette mit dem SLAVE-Anschluss der angesprochenen RTC5-Karte verbunden sind, stabil auf den 10 µs-Takt der angesprochenen Karte.
Aufruf	`sync_slaves()`
Hinweise	• Zur Verwendung siehe Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93. • SCANLAB empfiehlt, die Synchronisation unmittelbar nach Initialisierung aller Karten (per **load_program_file** und **load_correction_file**) durchzuführen. Andernfalls sollten alle beteiligten Karten (d.h. die Master-Karte und die nachgeschalteten Slave-Karten, die sich im Besitz der Anwendung befinden) bereits vor dem Aufruf von **sync_slaves** angehalten sein. Um Störungen während der Synchronisation zu vermeiden, dürfen außerdem während der Ausführung des Befehls weder externe Stopp-Signale an die Karten angelegt noch externe Listen-Starts ausgelöst werden (dies wird nicht automatisch unterbunden). • Im Laufe des **sync_slaves**-Befehls wird u.a. an die angesprochene Karte ein **simulate_ext_stop**-Befehl 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 der Anwendung befinden). Der Benutzer ist selbst dafür verantwortlich, dafür zu sorgen, dass eventuell laufende Prozesse dadurch nicht gestört werden. • Nach erfolgter Ausführung des Befehls befinden sich die Scan-System-Achsen aller beteiligten Karten entweder in Koordinaten-Mittelstellung (0, 0 [,0]) oder in Home-Jump-Position (jeweils unter Beibehaltung etwaiger Offset-Einstellungen z.B. über **set_offset**, **set_defocus** oder **set_hi**). • **sync_slaves** wirkt bzgl. der Synchronisierung nur auf diejenigen RTC5-Karten, die in einer Master/Slave-Kette mit dem MASTER-Anschluss der angesprochenen RTC5-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 sollte deshalb mit **sync_slaves** die Master-Karte der Master/Slave-Kette angesprochen werden. Falls am MASTER-Anschluss der angesprochenen Karte keine Karte angeschlossen ist, hat der Befehl keine Auswirkung. • Die Synchronisation der nachgeschalteten Slave-Karten mit **sync_slaves** erfolgt auch dann, wenn der BUSY-Status oder der INTERNAL-BUSY-Status dieser Karten gesetzt ist (sie werden automatisch angehalten). Synchronisiert werden jedoch nur diejenigen Slave-Karten, für die die Anwendung eine Zugriffsberechtigung besitzt (der Besitz wird nicht automatisch angefordert). Falls die Anwendung außer für die angesprochene Karte für keine weitere Karte eine Zugriffsberechtigung besitzt, hat der Befehl keine Auswirkung. • Zusätzlich werden im Laufe des **sync_slaves**-Befehls für alle existierenden Karten, die im Besitz der Anwendung sind (auch für vorgeschaltete Karten), alle **get_startstop_info**-Fehlerbits gelöscht. • Für jede nachgeschaltete Slave-Karte wird kurzzeitig der 10 µs-Takt unterbrochen und neu synchronisiert. Es können daher je nachgeschaltete Karte für maximal 20 µs Störungen der Datenübertragung zwischen RTC5 und Scan-System auftreten. Diese können u.U. harte Galvanometerscanner-Sprünge veranlassen, wenn die tatsächlichen Ausgabewerte nicht auf (0,0,[0]) stehen.

Kontrollbefehl	sync_slaves
Hinweise (Forts.)	• Der Befehl wird nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn der BUSY-Status der angesprochenen Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	• Verfügbar ab Version DLL 515, OUT 514, RBF 512. • Aktualisierung mit Version DLL 518, OUT 517, RBF 514 (Verbesserte Master/Slave-Synchronisation, siehe Seite 692). Es wird empfohlen, den Befehl erst ab der aktualisierten Version zu verwenden. • Aktualisierung mit Version DLL 523, OUT 524: führt gegebenenfalls auch eine Anpassung der Kurzbefehlszählung durch (siehe Seite 94).
Verweise	**get_master_slave**, **get_sync_status**

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()`
Hinweise	• Der Befehl **time_fix** ist synonym zu **time_fix_f_off** mit `FirstDay` = 0 und `Offset` = 0 (siehe Hinweise dort).
RTC4→ RTC5	Unveränderte Funktionalität. Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
Versionsinfo	Letzte Änderung Version OUT 517.
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` Siehe **time_fix_f_off**.
Hinweise	• Der Befehl **time_fix_f** ist synonym zu **time_fix_f_off** mit `Offset` = 0 (siehe Hinweise dort).
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Letzte Änderung Version OUT 517.
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 bzw. 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`	Dieser Parameter (32-Bit-Wert ohne Vorzeichen) 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).
	`Offset`	Vordatierung in Sekunden als 32-Bit-Wert ohne Vorzeichen. Zulässiger Wertebereich: [0 … (2^{32}–1)].
Hinweise	• Vor dem Aufruf von **time_fix_f_off**, **time_fix_f** oder **time_fix** sollte (nach jedem **load_program_file** mindestens einmalig) mit **time_update** ein Abgleich zwischen RTC5 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. Der Befehl **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 bzw. 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→ RTC5	Neuer Befehl.	
Versionsinfo	Verfügbar ab Version DLL 524, OUT 526.	
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 RTC5 auf die aktuelle PC-Zeit ein.
Aufruf	`time_update()`
Hinweise	• Die RTC5-Karte besitzt keine batteriegepufferte Uhr. Der Befehl **time_update** muss daher nach jedem **load_program_file** erteilt werden, wenn die 24-Stunden-Uhr und der Kalender der RTC5 kalibriert 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.
RTC4→ RTC5	Unverändert. Der Befehl stand nur bei der RTC *SCAN*alone-Karte, der Standalone-Version der RTC4-Karte, zur Verfügung.
Verweise	**time_fix**, **time_fix_f**, **time_fix_f_off**, **mark_date**, **mark_time**

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 zweidimensionalen Bildfeld.	
Aufruf	`timed_arc_abs( X, Y, Angle, T )`	
Parameter	`X, Y`	Absolute Koordinaten des Bogenzentrums in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt.
	`Angle`	Bogenwinkel in ° im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet "im Uhrzeigersinn". Zulässiger Wertebereich: [–3600.0° … +3600.0°] (±10 Vollkreise). Überlaufende Werte werden auf den Randbereich geclippt.
	`T`	Dauer der Kreisbogenmarkierung in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Im Gegensatz zu **arc_abs** führt **timed_arc_abs** die Markierung nicht mit der (über **set_mark_speed** oder **set_mark_speed_ctrl**) spezifizierten Markiergeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikrostufen) der angegebenen Dauer des Kreisbogens angepasst (siehe Kapitel 8.10 "Zeitbemessene Vektor- und Bogenbefehle", Seite 223). Die gesamte Markierzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • Die "Laser active"-Lasersteuersignale werden automatisch am Anfang der Markierung eingeschaltet (bzw. bleiben nach einem unmittelbar vorhergehenden Mark- oder Bogenbefehl eingeschaltet). Die eingestellten Scanner- und Laser-Delays werden dabei berücksichtigt (siehe Kapitel 7.2 "Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung", Seite 111). Beachten Sie, dass im Sky-Writing-Modus (siehe Seite 127) andere Delays ausgeführt werden. Ausnahme: Bogenbefehl der Bogenlänge 0 (siehe Hinweise Seite 115). • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u.a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141). • Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (Bildfeldgröße) geclippt.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die für das Bogenzentrum angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–524.288 … 524.287]).	
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 zweidimensionalen Bildfeld.	
Aufruf	`timed_arc_rel( dX, dY, Angle, T )`	
Parameter	`dX, dY`	*Relativ*koordinaten des Bogenzentrums in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt.
	`Angle`	Bogenwinkel in ° im 64-Bit-IEEE-Gleitkommaformat. Ein positives Vorzeichen bedeutet "im Uhrzeigersinn". Zulässiger Wertebereich: [–3600.0° … +3600.0°] (±10 Vollkreise). Überlaufende Werte werden auf den Randbereich geclippt.
	`T`	Dauer der Kreisbogenmarkierung in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Bogenzentrums werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_arc_abs** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.). • Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (Bildfeldgröße) geclippt.	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die für das Bogenzentrum angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend auf [–524.288 … 524.287]).	
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 zweidimensionalen Bildfeld.	
Aufruf	`timed_jump_abs( X, Y, T )`	
Parameter	`X, Y`	Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`T`	Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Im Gegensatz zu **jump_abs** führt **timed_jump_abs** den Sprung nicht mit der (über **set_jump_speed** oder **set_jump_speed_ctrl**) spezifizierten Sprunggeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikrostufen) der angegebenen Sprungdauer des Vektors angepasst (siehe Kapitel 8.10 "Zeitbemessene Vektor- und Bogenbefehle", Seite 223). Die gesamte Sprungzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • Die "Laser active"-Lasersteuersignale sind während des Sprungs ausgeschaltet (werden gegebenenfalls vor dem Sprung ausgeschaltet). • Nach einem Jump-Befehl wird ein (variables) Jump-Delay eingefügt. Ausnahme: Das einem Sprungvektor der Länge 0 folgende (variable) Jump-Delay wird nicht ausgeführt. Der Befehl selber benötigt dafür aber dennoch einen 10 µs-Takt zur Ausführung. • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141).	
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich, und `T` < 5 führt keinen **timed_jump_abs** aus, sondern **jump_abs**). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).	
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_jump_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`T` Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **timed_jump_abs**-Befehl (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert. • Der Parameter `DirectMove3D` des Befehls **set_delay_mode** bestimmt die Art der Z-Achsen-Bewegung (linear oder schrittweise korrigiert). • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Seite 141).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
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 zweidimensionalen Bildfeld.
Aufruf	`timed_jump_rel( dX, dY, T )`
Parameter	`dX, dY` *Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`T` Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_jump_abs** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich, und `T` < 5 führt keinen **timed_jump_rel** aus, sondern **jump_rel**). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_jump_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`T` Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_jump_abs_3d** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
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 zweidimensionalen Bildfeld.
Aufruf	`timed_mark_abs( X, Y, T )`
Parameter	`X, Y` Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`T` Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Im Gegensatz zu **mark_abs** führt **timed_mark_abs** die Markierung nicht mit der (über **set_mark_speed** oder **set_mark_speed_ctrl**) spezifizierten Markiergeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikrostufen) der angegebenen Dauer des Markvektors angepasst (siehe Kapitel 8.10 "Zeitbemessene Vektor- und Bogenbefehle", Seite 223). Die gesamte Markierzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • Die "Laser active"-Lasersteuersignale werden automatisch am Anfang der Markierung eingeschaltet (bzw. bleiben nach einem unmittelbar vorhergehenden Mark- oder Bogenbefehl eingeschaltet). Die eingestellten Scanner- und Laser-Delays werden dabei berücksichtigt (siehe Kapitel 7.2 "Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung", Seite 111). Beachten Sie, dass im Sky-Writing-Modus (siehe Seite 127) andere Delays ausgeführt werden. Ausnahme: Mark-Befehl der Länge 0 (siehe Hinweise Seite 115). • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141).
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich, und `T` < 5 führt keinen **timed_mark_abs** aus, sondern **mark_abs**). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_mark_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`T` Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **timed_mark_abs**-Befehl (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert. • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Seite 141).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
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 zweidimensionalen Bildfeld.	
Aufruf	`timed_mark_rel( dX, dY, T )`	
Parameter	`dX, dY`	*Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`T`	Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_mark_abs** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.).	
RTC4→ RTC5	Unveränderte Funktionalität (außer erhöhter Wertebereich, und `T` < 5 führt keinen **timed_mark_rel** aus, sondern **mark_rel**). Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).	
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 dreidimensionalen Arbeitsvolumen.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_mark_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) “Processing on the fly”-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`T` Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_mark_abs_3d** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.	
Aufruf	`timed_para_jump_abs( X, Y, P, T )`	
Parameter	`X, Y`	Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`P`	Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T`	Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
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**) spezifizierten Sprunggeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikrostufen) der angegebenen Sprungdauer des Vektors angepasst (siehe Kapitel 8.10 "Zeitbemessene Vektor- und Bogenbefehle", Seite 223). Die gesamte Sprungzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • **timed_para_jump_abs** benötigt zwei Listenspeicherplätze. Der Befehlsteil auf dem ersten Speicherplatz wird zur Laufzeit als kurzer Listenbefehl ausgeführt, anschließend der zweite Befehlsteil als normaler Listenbefehl. Beide Befehlsteile werden also innerhalb desselben 10 µs-Taktes ausgeführt, sofern nicht weitere kurze Listenbefehle davor ein **list_nop** zwischen beiden Teilen erzwingen. • Für `T` < 5 wirkt **timed_para_jump_abs** wie **para_jump_abs** und benötigt dann auch nur einen Listenspeicherplatz. • Die "Laser active"-Lasersteuersignale sind während des Sprungs ausgeschaltet (werden gegebenenfalls vor dem Sprung ausgeschaltet). • Nach einem Jump-Befehl wird ein (variables) Jump-Delay eingefügt. Ausnahme: Das einem Sprungvektor der Länge 0 folgende (variable) Jump-Delay wird nicht ausgeführt. Der Befehl selber benötigt dafür aber dennoch einen 10 µs-Takt zur Ausführung. • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141).	

Normaler Listenbefehl	timed_para_jump_abs
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_para_jump_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T` Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **timed_para_jump_abs**-Befehl (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert. • Der Parameter `DirectMove3D` des Befehls **set_delay_mode** bestimmt die Art der Z-Achsen-Bewegung (linear oder schrittweise korrigiert). • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Seite 141). • Der Befehl benötigt für T ≥ 5 zwei Listenspeicherplätze. Der erste Teil wird als unverzögerter kurzer Listenbefehl vor dem Hauptteil (normaler Listenbefehl) ausgeführt.

Multipler Listenbefehl	**timed_para_jump_abs_3d**
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.
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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.	
Aufruf	`timed_para_jump_rel( dX, dY, P, T )`	
Parameter	`dX, dY`	*Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`P`	Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T`	Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_para_jump_abs** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.).	
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).	
Versionsinfo	• Verfügbar ab Version DLL 517, OUT 516, RBF 512. • Letzte Änderung Version DLL 518.	
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_para_jump_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Sprungvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T` Dauer des gesamten Sprungvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Sprungvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_para_jump_abs_3d** (siehe Hinweise dort). • Der Befehl benötigt für T ≥ 5 zwei Listenspeicherplätze. Der erste Teil wird als unverzögerter kurzer Listenbefehl vor dem Hauptteil (normaler Listenbefehl) ausgeführt.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
Versionsinfo	• Verfügbar ab Version DLL 517, OUT 516, RBF 512. • Letzte Änderung Version DLL 518.
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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.	
Aufruf	`timed_para_mark_abs( X, Y, P, T )`	
Parameter	`X, Y`	Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`P`	Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T`	Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
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**) spezifizierten Markiergeschwindigkeit aus. Stattdessen wird die Geschwindigkeit (d.h. die Zahl der Mikrostufen) der angegebenen Dauer des Markvektors angepasst (siehe Kapitel 8.10 "Zeitbemessene Vektor- und Bogenbefehle", Seite 223). Die gesamte Markierzeit ist (für `T` ≥ 5) die Summe aus der angegebenen (gerundeten) Zeit und den eingestellten Delays. • **timed_para_mark_abs** benötigt zwei Listenspeicherplätze. Der Befehlsteil auf dem ersten Speicherplatz wird zur Laufzeit als kurzer Listenbefehl ausgeführt, anschließend der zweite Befehlsteil als normaler Listenbefehl. Beide Befehlsteile werden also innerhalb desselben 10 µs-Taktes ausgeführt, sofern nicht weitere kurze Listenbefehle davor ein **list_nop** zwischen beiden Teilen erzwingen. • Für `T` < 5 wirkt **timed_para_mark_abs** wie **para_mark_abs** und benötigt dann auch nur einen Listenspeicherplatz. • Die "Laser active"-Lasersteuersignale werden automatisch am Anfang der Markierung eingeschaltet (bzw. bleiben nach einem unmittelbar vorhergehenden Mark- oder Bogenbefehl eingeschaltet). Die eingestellten Scanner- und Laser-Delays werden dabei berücksichtigt (siehe Kapitel 7.2 "Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung", Seite 111). Beachten Sie, dass im Sky-Writing-Modus (siehe Seite 127) andere Delays ausgeführt werden. Ausnahme: Mark-Befehl der Länge 0 (siehe Hinweise Seite 115). • Bei der Umrechnung der spezifizierten Koordinatenwerte in Ausgabewerte für das Scan-System werden nach erfolgter Mikrovektorisierung u. a. zuvor definierte Koordinatentransformationen und zugewiesene Korrekturtabellen berücksichtigt (siehe Seite 141).	

Normaler Listenbefehl	timed_para_mark_abs
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_para_mark_abs**. Die Mikrovektorisierung 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, Y, Z` Absolute Koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T` Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Abgesehen von der zusätzlichen Bewegung in der dritten Raumrichtung, funktioniert der Befehl gleich wie der **timed_para_mark_abs**-Befehl (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.). • Die X- und Y-Achse können mit einer Auflösung von 20 Bit, die Z-Achse mit einer Auflösung von 16 Bit angesteuert werden. Die z-Koordinatenwerte werden von der RTC5 (sowohl im RTC5-Modus als auch im RTC4-Kompatibilitätsmodus) automatisch intern auf 20-Bit-Werte hochskaliert. • Bei Berechnung der z-Ausgabewerte für das Scan-System wird auch ein zuvor eingestellter Offset auf die z-Koordinate bzw. Fokuslänge berücksichtigt (siehe Seite 141). • Der Befehl benötigt zwei Listenspeicherplätze. Der erste Teil wird als unverzögerter kurzer Listenbefehl vor dem Hauptteil (normaler Listenbefehl) ausgeführt.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.
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 zweidimensionalen Bildfeld und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Aufruf	`timed_para_mark_rel( dX, dY, P, T )`
Parameter	`dX, dY` *Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z. B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T` Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**.
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_para_mark_abs** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.).
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend).
Versionsinfo	• Verfügbar ab Version DLL 517, OUT 516, RBF 512. • Letzte Änderung Version DLL 518.
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 dreidimensionalen Arbeitsvolumen und variiert simultan den mit **set_vector_control** ausgewählten Signalparameter linear auf den angegebenen Wert.
Einschränkung	Falls die 3D-Option nicht freigeschaltet ist bzw. wenn keine 3D-Korrekturtabelle zugewiesen ist (siehe **select_cor_table**), wirkt der Befehl wie **timed_para_mark_rel**. Die Mikrovektorisierung 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,dY,dZ` *Relativ*koordinaten des Markvektorendpunkts in *Bits* als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: • Für `X` und `Y`: [–8.388.608 … 8.388.607]. Überlaufende Werte werden auf den Randbereich geclippt. Der volle Wertebereich von [–8.388.608 … 8.388.607] ist nur als virtuelles Bildfeld z.B. für (freigeschaltete) "Processing on the fly"-Anwendungen nutzbar. Die aktuellen Koordinaten werden erst zur Laufzeit unmittelbar vor dem Anwenden der Korrekturtabelle auf [–524.288 … 524.287] (reale Bildfeldgröße) geclippt. • Für `Z`: [–32.768 … 32.767]. Überlaufende Werte werden auf den Randbereich geclippt.
	`P` Endwert des Signalparameters als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: abhängig von der Auswahl mit **set_vector_control** (Parameter `Ctrl`), identisch wie bei **set_vector_control** (Parameter `Value`).
	`T` • Dauer des gesamten Markvektors in *Mikrosekunden* (im 64-Bit-IEEE-Gleitkommaformat). Zulässiger Bereich: [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**).
Hinweise	• Die Koordinaten des Markvektorendpunkts werden als Relativkoordinaten bezüglich der aktuellen Position angegeben. Ansonsten ist der Befehl identisch mit **timed_para_mark_abs_3d** (siehe Hinweise dort, Ausnahme: `T` < 5, s.o.). • Der Befehl benötigt zwei Listenspeicherplätze. Der erste Teil wird als unverzögerter kurzer Listenbefehl vor dem Hauptteil (normaler Listenbefehl) ausgeführt.
RTC4→ RTC5	Neuer Befehl. Im RTC4-Kompatibilitätsmodus werden die angegebenen x- und y-Koordinatenwerte von der RTC5 mit 16 multipliziert (der zulässige Wertebereich verringert sich entsprechend). Der Wertebereich für die z-Koordinate ist im RTC5- und RTC4-Kompatibilitätsmodus identisch.
Versionsinfo	• Verfügbar ab Version DLL 517, OUT 516, RBF 512. • Letzte Änderung Version DLL 518.
Verweise	**timed_para_mark_abs_3d**, **para_mark_rel_3d**, **timed_mark_rel_3d**, **timed_para_mark_rel**

Kontrollbefehl	transform	
Funktion	Führt eine Rücktransformation von einzelnen Positionswerten durch.	
Aufruf	`TransformErrorCode = transform( &Sig1, &Sig2, Ptr, Code )`	
Parameter und Parameter-rückgabe	`Sig1`, `Sig2`	Parameter: zu transformierende Positionswerte als Pointer auf 32-Bit-Werte mit Vorzeichen. Parameterrückgabe: transformierte Positionswerte als 32-Bit-Werte mit Vorzeichen (die Eingabewerte werden überschrieben).
Parameter	`Ptr`	Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. 32-Bit- bzw. 64-Bit-Wert ohne Vorzeichen) auf den Arbeitsspeicherbereich des PCs, in den die Korrektur- und Transformationseinstellungen, die zur Rücktransformation verwendet werden sollen, zuvor mit **upload_transform** übertragen wurden.
	`Code`	Dieser Parameter (32-Bit-Wert ohne Vorzeichen) steuert Details der Rücktransformation, insbesondere welche Teiltransformationen durchgeführt werden: Soll eine Teiltransformationen *nicht* durchgeführt werden, so muss das entsprechende Bit (#2…#5) auf 1 gesetzt werden. Der Parameter hat die gleiche Bedeutung wie bei **get_transform** (`Sig1` entspricht `Ptr1` und `Sig2` `Ptr2`). Für Bit #0 = 0 werden die beiden angegebenen Positionswerte (`Sig1` und `Sig2`) als xy-Koordinaten rücktransformiert: Bit #1 = 0: Der unter `Sig1` angegebene Wert wird als x-Koordinate und der unter `Sig2` angegebene Wert als y-Koordinate rücktransformiert. = 1: Der unter `Sig1` angegebene Wert wird als y-Koordinate und der unter `Sig2` angegebene Wert als x-Koordinate rücktransformiert. Bit #2 = 0: Die Gain/Offset-Korrektur der automatischen Selbstkalibrierung wird rücktransformiert. Bit #3 = 0: Die Bildfeldkorrektur wird rücktransformiert. Bit #4 = 0: Der Offset der definierten Koordinatentransformation wird rücktransformiert. Bit #5 = 0: Die Gesamtmatrix der definierten Koordinatentransformation wird rücktransformiert. Bits #6…#31 Reserviert.

Kontrollbefehl	transform		
Parameter (Forts.)	`Code` (Forts.)	Für Bit #0 = 1 wird einer der beiden angegebenen Positionswerte (wahlweise `Sig1` oder `Sig2`) als z-Koordinate rücktransformiert:	
		Bit #1 = 0:	Der unter `Sig1` angegebene Wert wird als z-Koordinate rücktransformiert (`Sig2` bleibt unverändert).
		= 1:	Der unter `Sig2` angegebene Wert wird als z-Koordinate rücktransformiert (`Sig1` bleibt unverändert).
		Bit #2 = 0:	Der mit **set_defocus** oder **set_defocus_list** definierte Offset auf die Fokuslänge wird rücktransformiert.
		Bit #3 = 0:	Die ABC-Korrektur wird rücktransformiert.
		Bit #4 = 0:	Der mit **set_offset_xyz** oder **set_offset_xyz_list** definierte Offset auf die z-Koordinate wird rücktransformiert.
		Bits #5… #31	Reserviert.
Rückgabe	Fehlercode als 32-Bit-Wert ohne Vorzeichen:		
	Wert	Beschreibung	
	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.	
Hinweise	• Zur Rücktransformation von Positionswerten siehe Seite 174. • 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 (rückgegebener 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 Rücktransformation 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 bzw. die z-Koordinate mit 0 und die Werte A, B, C mit 0, 1, 0 initialisiert (1zu1-Rücktransformation). • Bei der Rücktransformation 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.		

Kontrollbefehl	transform
Hinweise (Forts.)	• Der Befehl **transform** greift nicht auf eine RTC5-Karte zu und ist daher auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-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 RTC5-Karte am PCI-Bus vorhanden sein). • Der Befehl **transform** ist nicht als Multi-Board-Befehl verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe "Error-Handling", Seite 98) werden von **transform** weder erzeugt noch verändert.
RTC4→ RTC5	Neuer Befehl. Alle rücktransformierten Werte (auch die z-Werte) sind auch im RTC4-Kompatibilitätsmodus im RTC5-20-Bit-Bereich und müssen gegebenenfalls von Benutzern selbst umgerechnet (durch 16 dividiert) werden.
Versionsinfo	• Verfügbar ab Version DLL 516, OUT 515, RBF 512. • Änderung Version DLL 517: Parameter `Code` geändert.
Verweise	**upload_transform**, **get_transform**, **get_values**

Kontrollbefehl	upload_transform
Funktion	Überträgt alle Korrektur- und Transformationseinstellungen, die dem Scan-System aktuell zugeordnet sind, von der RTC5-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: Primärer Scan-Kopf-Anschluss. = 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich). `Ptr` Pointer (bei C und C++ Datentyp ULONG_PTR, d.h. 32-Bit- bzw. 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 (Gain der automatischen Selbstkalibrierung für Galvanometerscanner 2 = 0. Bit #1 =1: y-Gain (Gain der automatischen Selbstkalibrierung für Galvanometerscanner 1 = 0. 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 ist BUSY oder INTERNAL-BUSY (**get_last_error**-Returncode `RTC5_BUSY`). Bits #8… #31 Reserviert.
Hinweise	• Die ausgelesenen und übertragenen Daten können für eine Rücktransformation von Ist-Positions-Werten mit **transform** oder **get_transform** verwendet werden (siehe auch Seite 174). • Zur Speicherung eines ausgelesenen Datensatzes muss von der Anwendung jeweils (an einer über `Ptr` angegebenen Adresse) ein Speicherbereich der Größe 528520 Bytes auf dem PC bereitgestellt werden. • Im Falle eines Fehlers (außer für Bit#6 = 1) wird in `Ptr` eingetragen, dass die Daten fehlerhaft sind. Von den Befehlen **transform** und **get_transform** wird diese Fehler-Information erkannt und führt dort dazu, dass die Rücktransformation nicht ausgeführt wird (von **transform** wird dann ein entsprechender Fehlercode, bei **get_transform** ein **get_last_error**-Returncode `RTC5_PARAM_ERROR` erzeugt). • Der Befehl wird nicht ausgeführt (**get_last_error**-Returncode: `RTC5_BUSY`), wenn der BUSY-Status der angesprochenen Karte gesetzt ist (Liste wird abgearbeitet oder wurde mit **pause_list** angehalten) oder wenn der INTERNAL-BUSY-Status der Karte gesetzt ist. Der Befehl wird dagegen ausgeführt, wenn eine Liste mit **set_wait** angehalten wurde (PAUSED-Status gesetzt). • Während der Ausführung des Befehls werden externe Listen-Starts unterbunden.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 516, OUT 515, RBF 512.
Verweise	**transform**, **get_transform**

Kontrollbefehl	verify_checksum
Funktion	Prüft bzw. erzeugt die Check-Summe einer Korrekturdatei.
Aufruf	`verify_checksum( Name )`
Parameter	`Name` Name der Korrekturdatei als Pointer 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.).
Hinweise	• Die Überprüfung von Korrekturdatei-Downloads funktioniert nur bei Korrekturdateien, in die eine Check-Summe eingetragen ist (siehe Seite 99 und **set_verify**) . • Der Befehl **verify_checksum** ist auch ohne explizite Zugriffsberechtigung für irgendeine RTC5-Karte verfügbar. • Der Befehl **verify_checksum** ist nicht als Multi-Board-Befehl 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 "Error-Handling", Seite 98) werden von **verify_checksum** weder erzeugt noch verändert.
RTC4→ RTC5	Neuer Befehl.
Verweise	**set_verify**

Normaler Listenbefehl	wait_for_encoder
Funktion	Wartet, bis am ausgewählten Encoder-Zähler der angegebene Zählwert erstmalig über- bzw. 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 Encoder-Zählers als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0. = 1: Encoder-Zähler Encoder1.
Hinweise	• Der Befehl ist synonym zu **wait_for_encoder_mode** mit Parameter `Mode` = 0 (siehe Hinweise dort).
RTC4→ RTC5	Neuer Befehl.
Verweise	**wait_for_encoder_mode**

Multipler Listenbefehl	**wait_for_encoder_in_range**
Funktion	Wartet, bis beide Encoder-Zähler gleichzeitig innerhalb des angegebenen Bereichs liegen (Grenzen eingeschlossen).
Aufruf	`wait_for_encoder_in_range( EncXmin, EncXmax, EncYmin, EncYmax )`
Parameter	`EncXmin`, `EncXmax`, `EncYmin`, `EncYmax` Grenzwerte als 32-Bit-Werte mit Vorzeichen. Zulässiger Wertebereich: [-2^{31} … $+(2^{31}-1)$].
Hinweise	• Zur Verwendung des Befehls siehe "Synchronisation durch Encoder-Signale", Seite 246 und "Synchronisation von "Processing on the fly"-Anwendungen", Seite 211. • Der Befehl benötigt zwei 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 der Befehl lediglich eine Wartezeit (ohne Galvanometerscanner-Bewegung). • Für `EncXmin` = `EncXmax` (oder `EncYmin` = `EncYmax`) kann auf das Erreichen eines konkreten Encoderwertes gewartet werden. • Der Befehl ist auch verfügbar, wenn die "Processing on the fly"-Option nicht freigeschaltet ist. • Der Befehl lässt die Lasersteuersignale unverändert. Soll der Laser während des Wartens aus sein, muss vor dem Befehl irgend ein anderer Befehl stehen, der die "Laser active"-Lasersteuersignale ausschaltet, z. B. ein **list_nop**. • Ob die Galvanometerscanner während des Wartens stehen bleiben oder sich mit den Encoder-Änderungen mitbewegen, hängt vom aktiven "Processing on the fly"-Modus ab (siehe Seite 211): Mit **set_fly_2d** bewegen sie sich mit, ansonsten bleiben sie stehen.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 536, OUT 536.
Verweise	**wait_for_encoder_mode**, **park_position**, **park_return**

Normaler Listenbefehl	**wait_for_encoder_mode**
Funktion	Wartet, bis am ausgewählten Encoder-Zähler der angegebene Zählwert erstmalig über- bzw. unterschritten wurde.
Aufruf	`wait_for_encoder_mode( Value, EncoderNo, Mode )`
Parameter	`Value` Zählwert als 32-Bit-Wert mit Vorzeichen. Zulässiger Wertebereich: [-2^{31} … $+(2^{31}-1)$].
	`EncoderNo` Nummer des zu verwendenden Encoder-Zählers als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Encoder-Zähler Encoder0. = 1: Encoder-Zähler Encoder1.
	`Mode` 32-Bit-Wert mit Vorzeichen. = 0: Wartet auf Über-/Unterschreitung (positionsabhängig). > 0: Wartet auf Überschreitung (positionsunabhängig). < 0: Wartet auf Unterschreitung (positionsunabhängig).
Hinweise	• Zur Verwendung des Befehls siehe Kapitel 9.3.3 “Synchronisation durch Encoder-Signale”, Seite 246. • 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 der Befehl auf eine Überschreitung, sonst auf eine Unterschreitung. Ist `Value` positiv und bereits kleiner als der aktuelle Encoder-Zählwert, wartet der Befehl 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 der Befehl 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 `RTC5_PARAM_ERROR`). • Wenn keine encoderbasierte “Processing on the fly”-Korrektur aktiv ist, bewirkt der Befehl 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. • Der Befehl ist auch verfügbar, wenn die “Processing on the fly”-Option nicht freigeschaltet ist. • Für `Mode` = 0 ist der Befehl synonym zu **wait_for_encoder**. • Der Befehl lässt die Lasersteuersignale unverändert. Soll der Laser während des Wartens aus sein, muss vor dem Befehl irgend ein anderer Befehl stehen, der die “Laser active”-Lasersteuersignale ausschaltet, z.B. ein **list_nop**. • Ob die Galvanometerscanner während des Wartens stehen bleiben oder sich mit den Encoder-Änderungen mitbewegen, hängt vom aktiven “Processing on the fly”-Modus ab (siehe Seite 211): Mit **set_fly_2d** bewegen sie sich mit, ansonsten bleiben sie stehen.
RTC4→ RTC5	Neuer Befehl.
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/SPI-Schnittstelle den angegebenen Wert erstmalig erreicht, über- bzw. unterschritten hat.
Aufruf	`wait_for_mcbsp( Axis, Value, Mode )`
Parameter	`Axis` Auswahl, welches Halbwort des Eingangswerts zur Prüfung herangezogen wird (s.u.) als 32-Bit-Wert ohne Vorzeichen. Zulässige Werte: = 0: Unteres Halbwort (Achse X, Galvanometerscanner 2). = 1: Oberes Halbwort (Achse Y, Galvanometerscanner 1). `Value` Grenzwert als 32-Bit-Wert mit Vorzeichen. `Mode` 32-Bit-Wert mit Vorzeichen. = 0: Wartet auf Gleichheit. > 0: Wartet auf Überschreitung. < 0: Wartet auf Unterschreitung.
Hinweise	• Zur Verwendung des Befehls siehe "Synchronisation und Online-Positionierung mit McBSP/SPI-Signalen", Seite 248. • Der Befehl ist vergleichbar zu **wait_for_encoder_mode**, es wird jedoch die McBSP/SPI-Schnittstelle abgefragt. • Falls **set_fly_x_pos** und **set_fly_y_pos** *gleichzeitig* aktiviert sind, besteht `Value` aus zwei 16 Bit-Halbworten (siehe Seite 202). Nur in diesem Fall hat `Axis` die Bedeutung der Auswahl, welches Halbwort zur Prüfung herangezogen wird. In allen anderen Fällen ist `Axis` bedeutungslos und `Value` wird als 32 Bit-Wert mit Vorzeichen interpretiert. • `Axis` muss stets 0 oder 1 sein (auch wenn es bedeutungslos ist). Andernfalls wird der Befehl durch **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Wenn weder **set_fly_x_pos**, **set_fly_y_pos** noch **set_fly_rot_pos** aktiviert ist, bewirkt der Befehl 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. • Der Befehl ist auch verfügbar, wenn die "Processing on the fly"-Option nicht freigeschaltet ist. • Der Befehl lässt die Lasersteuersignale unverändert. Soll der Laser während des Wartens aus sein, muss vor dem Befehl irgend ein anderer Befehl stehen, der die "Laser active"-Lasersteuersignale ausschaltet, z.B. ein **list_nop**.
RTC4→ RTC5	Neuer Befehl.
Verweise	**wait_for_encoder_mode**

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.
Hinweise	• Siehe auch Kapitel 9.1.2 "8 Bit-Digital-Ausgang", Seite 233.
RTC4→ RTC5	Unveränderte Funktionalität.
Verweise	**write_8bit_port_list**

Verzögerter kurzer Listenbefehl	write_8bit_port_list
Funktion	Identisch mit **write_8bit_port**, aber ein Listenbefehl.
Aufruf	`write_8bit_port_list( Value )`
Parameter	`Value` 8-Bit-Ausgabewert (DATA0…DATA7) als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 8 Bits ausgewertet.
Hinweise	• Siehe **write_8bit_port**.
RTC4→ RTC5	Unveränderte Funktionalität.

Kontrollbefehl	write_abc_to_file
Funktion	Schreibt die ABC-Werte direkt in die angegebene Korrekturdatei im PC.
Aufruf	`ErrorNo = write_abc_to_file( Name, A, B, C )`
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).
Parameter	`Name` Name der Korrekturdatei als Pointer auf einen nullterminierten ANSI-String. `A, B, C` Koeffizienten (im 64-Bit-IEEE-Gleitkommaformat) der parabolischen Funktion $z_{out} = A + Bl + Cl^2$, die zur Berechnung der z-Ausgabewerte verwendet wird. Zulässige Wertebereiche: siehe Befehl **load_z_table**.
Hinweise	• Der Befehl **write_abc_to_file** ist auch ohne explizite Zugriffsberechtigung auf irgendeine RTC5-Karte verfügbar. • Der Befehl **write_abc_to_file** ist nicht als Multi-Board-Befehl verfügbar. • Die kartenspezifischen Fehlervariablen `LastError` und `AccError` (siehe Kapitel 6.8 "Error-Handling", Seite 98) werden von **write_abc_to_file** weder erzeugt noch verändert. • Wenn ein Fehlercode ungleich 0 oder 12 zurückgegeben wird, werden die ABC-Werte nicht ausgegeben.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 538, OUT 538.
Verweise	**read_abc_from_file**

Kontrollbefehl	write_da_1
Funktion	Siehe **write_da_x**.
Aufruf	`write_da_1( Value )`
Parameter	`Value` 12-Bit-Ausgabewert für den Analogausgang 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.
RTC4→ RTC5	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.

Verzögerter kurzer Listenbefehl	write_da_1_list
Funktion	Siehe **write_da_x_list**.
Aufruf	`write_da_1_list( Value )`
Parameter	`Value` 12-Bit-Ausgabewert für den Analogausgang 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.
RTC4→ RTC5	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.

Kontrollbefehl	write_da_2
Funktion	Siehe **write_da_x**.
Aufruf	`write_da_2( Value )`
Parameter	`Value` 12-Bit-Ausgabewert für den Analogausgang 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.
RTC4→ RTC5	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.

Verzögerter kurzer Listenbefehl	**write_da_2_list**
Funktion	Siehe **write_da_x_list**.
Aufruf	`write_da_2_list( Value )`
Parameter	`Value` 12-Bit-Ausgabewert für den Analogausgang 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.
RTC4→ RTC5	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.

Kontrollbefehl	**write_da_x**
Funktion	Schreibt einen Ausgabewert auf einen der analogen Ausgänge der RTC5.
Aufruf	`write_da_x( x, Value )`
Parameter	`x` Nummer des Analogausgangs als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [1, 2] (1: ANALOG OUT1, 2: ANALOG OUT2). `Value` 12-Bit-Ausgabewert für den gewählten Analogausgang 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.
Hinweise	• Siehe auch "12-Bit-Analog-Ausgänge", Seite 233. • Der Ausgabebereich an den Analogausgängen ist 0 V … 10 V. • Für `x` < 1 oder `x` > 2 wird der Befehl ignoriert (**get_last_error**-Returncode `RTC5_PARAM_ERROR`). • Alternativ zu **write_da_x** können die Befehle **write_da_1** / **write_da_2** (ohne Parameter `x`) verwendet werden.
RTC4→ RTC5	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus wird `Value` jeweils mit 4 multipliziert (bei der RTC4 hatten die Analogausgänge nur eine Auflösung von 10 Bit).

Verzögerter kurzer Listenbefehl	**write_da_x_list**	
Funktion	Identisch mit **write_da_x**, aber ein Listenbefehl.	
Aufruf	`write_da_x_list( x, Value )`	
Parameter	`x`	Nummer des Analogausgangs als 32-Bit-Wert ohne Vorzeichen. Zulässiger Bereich: [1, 2] (1: ANALOG OUT1, 2: ANALOG OUT2).
	`Value`	12-Bit-Ausgabewert für den gewählten Analogausgang 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.
Hinweise	• Für `x` < 1 oder `x` > 2 wird der Befehl durch ein **list_nop** ersetzt (**get_last_error**-Returncode `RTC5_PARAM_ERROR`).	
RTC4→ RTC5	Unveränderte Funktionalität. Im RTC4-Kompatibilitätsmodus: wie **write_da_x**.	

Kontrollbefehl	write_hi_pos
Funktion	Schreibt die angegebenen Werte als ASC-Referenzwerte (= Home-In-Referenzpositionen, nicht zu verwechseln mit Home-In-Positionen) ins EEPROM der RTC5-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: Primärer Scan-Kopf-Anschluss. = 2: Sekundärer Scan-Kopf-Anschluss (Freischaltung erforderlich).
	`X1, X2, Y1, Y2` Referenzpositionen in Bits als 32-Bit-Werte mit Vorzeichen.
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.
Hinweise	• Der Befehl **write_hi_pos** wird bei einem Wechsel des Scan-Kopfs von RTC5-Karte “A” zu RTC5-Karte “B” benutzt, um dessen Home-In-Referenzpositionen von der RTC5-Karte “A” nach RTC5-Karte “B” zu übertragen. • Lesen Sie mit `get_hi_pos( HeadNo )` die Home-In-Referenzpositionen von RTC5-Karte “A” aus (nur nach **init_rtc5_dll**, siehe **get_hi_pos**-Funktionsbeschreibung). • Der Befehl **write_hi_pos** wird auch ausgeführt, wenn der Scan-Kopf ausgeschaltet ist oder gar kein Scan-Kopf angeschlossen ist. • Auf der RTC5-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. Andernfalls liefert **write_hi_pos** den Fehler Bit #5 = 1 zurück. Schließen Sie ggf. den Scan-Kopf an RTC5-Karte “B” an und führen Sie `auto_cal(HeadNo, 4)` aus, bevor Sie **write_hi_pos** ausführen.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 538, OUT 538.
Verweise	**get_hi_pos**, **get_auto_cal**, **auto_cal**

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.
Hinweise	• Mit den Befehlen **set_io_cond_list** und **clear_io_cond_list** können individuelle Bits des 16-Bit Digital-Ausgangs, abhängig vom aktuellen Zustand des Digital-*Eingangs*, gesetzt bzw. rückgesetzt werden. • Mit **write_io_port_mask** bzw. **write_io_port_mask_list** können auch nur ausgewählte Bits am 16-Bit-Digital-Ausgang wie gewünscht verändert werden. • Siehe auch Kapitel 9.1.1 "16 Bit-Digital-Ausgang", Seite 232.
RTC4→ RTC5	Unveränderte Funktionalität.
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	Identisch mit **write_io_port**, aber ein Listenbefehl.
Aufruf	`write_io_port_list( Value )`
Parameter	`Value` 16-Bit-Ausgabewert (DIGITAL OUT0 … DIGITAL OUT15) als 32-Bit-Wert ohne Vorzeichen. Es werden nur die unteren 16 Bits ausgewertet.
Hinweise	• Siehe **write_io_port**.
RTC4→ RTC5	Unveränderte Funktionalität.

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.
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 232.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.
Verweise	**write_io_port**, **write_io_port_mask_list**

Verzögerter kurzer Listenbefehl	write_io_port_mask_list
Funktion	Identisch mit **write_io_port_mask**, aber ein Listenbefehl.
Aufruf	`write_io_port_mask_list( 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.
Hinweise	• Siehe **write_io_port_mask**.
RTC4→ RTC5	Neuer Befehl.
Versionsinfo	Verfügbar ab Version DLL 517, OUT 516, RBF 512.
Verweise	**write_io_port_mask**, **write_io_port_list**

10.3 Nicht mehr unterstützte RTC2/RTC3/RTC4-Befehle

Einige RTC3/RTC4-Befehle sowie einige RTC2-Befehle, die von der RTC3/RTC4 noch emuliert wurden, werden von der RTC5 nicht mehr unterstützt. Sie können aber größtenteils durch RTC5-Befehle ersetzt werden. Die folgende Tabelle listet alle nicht mehr unterstützen Befehle zusammen mit den entsprechenden RTC5-Ersatz-Befehlen auf.

Kontrollbefehl	**aut_change**
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**auto_change**

Kontrollbefehl	**dsp_start**
Unterstützung	Dieser RTC2/RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	Dieser Befehl ist nicht mehr erforderlich. Der Signalprozessor (DSP) startet automatisch nach dem Laden der Programmdatei über den Befehl **load_program_file**.

Listenbefehl	**field_jump**
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**home_position**, der “field jump” wird automatisch ausgeführt.

Kontrollbefehl	**get_rtc2_mode**
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	Bei der RTC5 muss der Laser-Mode nicht mehr abgefragt werden, da der Laser-Mode nicht über die Hardware, sondern über den Software-Befehl **set_laser_mode** eingestellt wird.

Kontrollbefehl	**get_rtc2_version**
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**get_rtc_version**

Kontrollbefehl	**get_version**
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**get_hex_version**

Kontrollbefehl	get_xy_pos
Unterstützung	Dieser RTC2/RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	Ersatzbefehl: **get_value**

Kontrollbefehl	get_xyz_pos
Unterstützung	Dieser RTC2/RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	Ersatzbefehl: **get_value**

Listenbefehl	home_jump
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**home_position**, der “home jump“ wird automatisch ausgeführt.

Listenbefehl	laser_on
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**laser_on_list**

Kontrollbefehl	load_cor
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**load_correction_file**

Kontrollbefehl	load_pro
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**load_program_file**

Listenbefehl	pola_abs, polb_abs, polc_abs
Unterstützung	Diese RTC2-Befehle werden nicht von der RTC5 unterstützt.
Ersatz-Befehl	**mark_abs**

Kontrollbefehl	read_pixel_ad
Unterstützung	Dieser RTC2/RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	**Es gibt keinen äquivalenten RTC5-Befehl.**

Kontrollbefehl	rtc3_count_cards / rtc4_count_cards
Unterstützung	Diese RTC3/RTC4-Befehle werden nicht von der RTC5 unterstützt.
Ersatz-Befehl	rtc5_count_cards

Kontrollbefehl	select_list
Unterstützung	Dieser RTC2/RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	**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" (bzw. 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.

Kontrollbefehl	set_base
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**Es gibt keinen äquivalenten RTC5-Befehl**, weil die RTC5 eine plug-and-play-Karte ist, deren Basisadresse automatisch gesetzt wird.

Kontrollbefehl	set_co2_standby
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	set_standby

Listenbefehl	set_co2_standby_list
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	set_standby_list

Listenbefehl	set_delays
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	set_laser_delays, set_scanner_delays

Listenbefehl	set_encoder_values
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	set_fly_x, set_fly_y

Kontrollbefehl	set_gain
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**set_matrix**, **set_offset**

Kontrollbefehl	set_list_mode
Unterstützung	Dieser RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	Siehe "Ringpuffer-Modus", Seite 90.

Kontrollbefehl	set_mode
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**Es gibt keinen äquivalenten RTC5-Befehl**, weil der Bildfeld-Korrektur-Algorithmus bei der RTC5 immer aktiviert ist. Stattdessen kann eine 1-zu-1-Korrektur-Datei geladen werden.

Kontrollbefehl	set_piso_control
Unterstützung	Dieser RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	Der Befehl ist bei der RTC5 nicht mehr nötig. Bei einer Datenübertragung gemäß SL2-100-Protokoll ist die bidirektionale Kommunikation zwischen Scan-System und RTC5 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 aber über die Jumper am XY2-100-Konverter, siehe Kapitel 4.3.2 "XY2-100-Konverter (optional)", Seite 43.

Kontrollbefehl	set_speed
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**set_jump_speed**, **set_mark_speed**

Kontrollbefehl	set_wobbel_xy
Unterstützung	Dieser RTC4-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**set_wobbel**, **set_wobbel_mode**

Kontrollbefehl	set_yag_parameter
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	**Es gibt keinen äquivalenten RTC5-Befehl.**

Kontrollbefehl	write_da
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	write_da_1

Listenbefehl	write_da_list
Unterstützung	Dieser RTC2-Befehl wird nicht von der RTC5 unterstützt.
Ersatz-Befehl	write_da_1_list

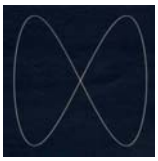
Kontrollbefehl	z_out
Unterstützung	Dieser RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	Um den z-Wert in einem 3-Achsen-Scan-System einzustellen, kann bei der RTC5 mit freigeschalteter 3D-Option der Befehl set_defocus verwendet werden. Nach load_z_table(0.0, 1.0, 0.0) hat set_defocus(Z) die gleiche Wirkung wie **z_out**(Z) (siehe auch set_offset_xyz).

Listenbefehl	z_out_list
Unterstützung	Dieser RTC3/RTC4-Befehl wird nicht von der RTC5 unterstützt.
RTC4→ RTC5	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).

11 Demo-Programme

Das RTC5-Software-Paket enthält im Verzeichnis `DemoFiles` einige Programmcode-Beispiele (siehe auch Abschnitt "Demo-Dateien", Seite 22). Sie demonstrieren außer der Initialisierung von DLL und RTC5 mit Fehlerbehandlung die Verwendung diverser Kontroll- und Listenbefehle.

Sie enthalten die benötigten Aufrufsequenzen der RTC5-Befehle, die Sie ohne Schwierigkeiten in Ihre bevorzugte Programmiersprache übersetzen können. Die folgende Tabelle gibt einen Überblick über die Charakteristik der Demo-Programme.

Dateiname	Demo1	Demo2	Demo3	Demo4	Demo5	Demo6	Demo7
Markier-Aufgabe	Quadrat und Dreieck 	Lissajous-Figuren 	Archimedische Spiralen 	Rasterbilder 	Quadrate und Dreiecke 	Rasterbilder, "Processing on the fly"	Textausgabe, intelli*SCAN*
Lasertyp	CO_2	YAG	YAG	CO_2	YAG	CO_2	YAG
Markier-methode	Vektor	Vektor	Vektor	Pixelmodus	Vektor	Pixelmodus, Flymode	Vektor
Zusatz-Features			Home-Jump		Home-Jump	Datenauf-zeichnung, Zeitmessung, 16-Bit-IO-Port	Zeichensatz, Datenauf-zeichnung, Zeitmessung, i*DRIVE*-Funktionen
DLL-Linken	implizit	explizit	explizit	explizit	explizit	implizit	implizit
Listenhandling	Verwendung *einer* Liste	eine Liste, kontinuier-licher Transfer (Ringpuffer)	zwei alternie-rende Listen, kontinuier-licher Transfer	zwei alternie-rende Listen, kontinuierli-cher Transfer	zwei Listen	zwei alternie-rende Listen, kontinuier-licher Transfer	zwei Listen (Default), geschützter Bereich
Externe Steuer-eingaben					/START, /STOP	erlaubt	
Ausnahme-behandlung	load_list	get_status, set_wait, release_wait, pause_list, restart_list, stop_execution	get_status, load_list, pause_list, restart_list, stop_execution	load_list	get_status, load_list, stop_execution	get_status, load_list, stop_execution	get_status
Sonstige Befehle	config_list, set_end_of_list, execute_list_pos	config_list, set_start_list_pos, execute_list_pos, set_end_of_list	config_list, set_start_list_pos, set_end_of_list, execute_list_pos, auto_change	config_list, set_end_of_list, execute_list_pos, auto_change, set_pixel_line, set_n_pixel	config_list, set_end_of_list, execute_list_pos, set_extstartpos, set_control_mode, get_startstop_info bounce_supp	config_list, set_end_of_list, execute_list_pos, auto_change, set_control_mode, simulate_encoder, set_pixel_line, set_n_pixel, set_fly_x fly_return set_trigger, save_and_restart-_timer, write_io_port_list, get_io_status	control_command, load_char, mark_text, mark_text_abs, set_start_list_pos, set_trigger, measurement_status, get_waveform, set_start_list_pos, save_and_restart-_timer, set_end_of_list, execute_list_pos

12 Fehlersuche

Problem	Behebung
Rechner bootet nicht	Schalten Sie den Rechner ab und prüfen Sie die folgenden Aspekte: • Überprüfen Sie, ob die Karte richtig im 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.
RTC5 lässt sich nicht ansprechen	• Überprüfen Sie, ob die Treiber richtig installiert sind. Siehe Kapitel 5.4 "Installation der RTC5-Software", Seite 62. • 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 angesteuert wird, überprüfen sie ob die Lötjumper des Konverters passend zur Kabellänge eingestellt sind. Siehe Abb. 8. Wenn ja: Überprüfen Sie, ob die Funktionen der DLL richtig in das Anwenderprogramm importiert wurden. Siehe Kapitel 6.2.2 "Import von Befehlen", Seite 66.
Anwenderprogramm stürzt ab	• Überprüfen Sie, ob die Treiber richtig installiert sind. Siehe Kapitel 5.4 "Installation der RTC5-Software", Seite 62. • Überprüfen Sie, ob die RTC5 fehlerfrei im Anwenderprogramm initialisiert wurde. • Überprüfen Sie die DLL-Importdeklarationen im Anwenderprogramm. Siehe Kapitel 6.2.2 "Import von Befehlen", Seite 66.
Fehler in der Scan-Kopf-Steuerung	• Überprüfen Sie, ob der Scan-Kopf über das Datenkabel richtig mit der RTC5 verbunden ist. Vergewissern Sie sich, ob die empfohlenen Spezifikationen für das Datenkabel erfüllt sind. Siehe Kapitel 4.3.3 "Datenkabel (Zubehör)", Seite 45. • 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 Lasersteuerung	• Überprüfen Sie die Funktionsfähigkeit der Schnittstelle zwischen Laser und RTC5.
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 121.
Der Laser ist während Jump-Befehlen nicht ausgeschaltet	• Überprüfen Sie, ob die Laser- und Scanner-Delays richtig definiert wurden. Siehe Kapitel 7.2 "Delayeinstellungen zur Synchronisation der Scan-Kopf- und Lasersteuerung", Seite 111.

Für eine Fehlersuche sind außerdem u. U. 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 "Error-Handling", Seite 98).
- mit **set_verify** kann sichergestellt werden, dass alle Downloads (Befehle, Tabellen) fehlerfrei ausgeführt wurden (siehe Abschnitt "Download-Überprüfung", Seite 99).
- mit **get_value** oder **get_values** können Sie einzelne vom Scan-System zurückgegebene Werte auslesen oder mit **set_trigger**/**set_trigger4** und **get_waveform** eine ganze Serie von Rückgabewerten aufzeichnen (siehe Abschnitt "Statusüberwachung und Diagnose", Seite 143; bei i*DRIVE*-Scan-Systemen siehe auch Kapitel 8.1, Seite 172).
- 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-Überlauf", Seite 142).
- 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 Lasersignale, 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 Seite 250) 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 121).

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 RTC5 sollten ausschließlich von SCANLAB durchgeführt werden. Bei Manipulationen an der RTC5 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 RTC5, die nicht von SCANLAB durchgeführt worden sind,
- für Schäden, die durch eine unsachgemäße Verpackung bei einer Rücklieferung an SCANLAB entstehen.
- bei Manipulationen an der RTC5,
- 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 RTC5 wird von SCANLAB gegen eine Gebühr zurückgenommen und fachgerecht entsorgt.

14 Technische Spezifikationen der RTC5 PCI-Karte

Systemanforderungen

Windows-PC mit freiem PCI-Steckplatz

Betriebssystem	Microsoft Windows 10, 8, 7, Vista, XP SP2, XP SP3 als 32-Bit oder 64-Bit Version

Abmessungen

Länge	160 mm
Breite	100 mm

Schnittstellen zum Scan-System

Anschlüsse	Primärer Scan-Kopf-Anschluss SCAN HEAD (9-polige Sub-D-Buchse)
• optional aktiviert:	Sekundärer Scan-Kopf-Anschluss 2. SCAN HEAD (10-polige Stiftleiste)
Signale	SL2-100-Protokoll
• optional:	XY2-100-Protokoll über XY2-100-Konverter

Ansteuerung eines Scan-Systems

Zahl der Speicherbereiche	bis zu 3 (konfigurierbar)
Kapazität des Listenspeichers	1 Million Befehle
Ausgabeintervall der Mikrovektoren	10 µs
maximaler Wertebereich der Bildfeldkoordinaten	–524.288 bis +524.287 (20 Bit mit Vorzeichen)
virtuelles Bildfeld (z.B. für "Processing on the fly")	–8.388.608 bis +8.388.607 (24 Bit mit Vorzeichen)

Schnittstellen zum Laser und zur Peripherie

- **LASER Laseranschluss-Buchse**

Anschluss	15-polige Sub-D-Buchse, female
Lasersignale	LASER1, LASER2, LASERON
• TTL-Pegel	5 V, high- oder low-aktiv (programmierbar)
• max. Last	20 mA
• Bezugspunkt	GND (GND2 bei optionaler Optoentkopplung)
Analoge Ausgänge	ANALOG OUT1, ANALOG OUT2
• 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

Eingänge für externe Start- und Stoppsignale	TTL low-aktiv, intern über Pull-up-Widerstände (4,7 kΩ) mit +3,3 V verbunden
• /START	flankensensitiv
• /STOP	pegelsensitiv
• Bezugspunkt	GND
Sonstige Signale	
• BUSY OUT	5 V, TTL high-aktiv, max. 10 mA
• +5 V	max. 100 mA
• Bezugspunkt	GND

- **EXTENSION 1-Anschluss**

Anschluss	40-polige Stiftleiste, Ausgangssignalpegel konfigurierbar (über **JP1**)
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 Bits
• LOW-Pegel	< 0,5 V
• HIGH-Pegel	> 2,6 V … 24 V
• Max. Eingangs-spannungsbereich	–0,5 V … +26 V
• Eingangswider-stand	> 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

- **EXTENSION 2-Anschluss**

Anschluss	26-polige Stiftleiste, konfigurierbar (über JP2-JP8)
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
Lasersignale	LASERON, LASER1, LASER2 (siehe LASER Laseranschluss-Buchse)
Sonstige Signale	
• +5 V	max. 100 mA
• Bezugspunkt	GND

- **MARKING ON THE FLY-Anschluss**

Anschluss	16-polige Stiftleiste
2 Encoder-Eingänge für Inkremental-Encoder	ENCODER X(1±,2±) und ENCODER Y(1±,2±), jeweils für ein Paar von standardisierten differentiellen Eingangssignalen (RS422). HIGH-Pegel ≥ 2.0 V LOW-Pegel ≤ 0.8 V) f ≤ 4 MHz
Analoger Ausgang	ANALOG OUT2 (siehe LASER Laseranschluss-Buchse)
Eingänge für externe Start- und Stoppsignale	/START2, /STOP2 (siehe /START, /STOP der LASER Laseranschluss-Buchse)
Sonstige Signale	
• BUSY OUT	5 V, TTL high-aktiv, max. 10 mA
• +5 V	max. 100 mA.
• Bezugspunkt	GND

- **RS232-Schnittstelle**

Anschluss	10-polige Stiftleiste
Eingang	RxD
• Spannungsbereich	max. –25 V … +25 V
Ausgang	TxD
• Spannungsbereich	max. –13 V … +13 V
Bezugspunkt	GND
Baud-Rate	300 … 115.200

- **"SPI / I2C"-Anschluss**

Anschluss	10-polige Stiftleiste

In McBSP-Konfiguration, siehe Seite 54:

• Sendersignalpegel	3,3 V TTL
• Empfängersignal-pegel	3,3 V oder 5 V TTL
• McBSP-Mode	Single Phase Frame Single Element per Frame 32 Bits per Element DataDelay N Bit
• Bezugspunkt	GND

In SPI-Konfiguration, siehe Seite 57:

• Sendersignalpegel	3,3 V TTL
• Empfängersignal-pegel	3,3 V oder 5 V TTL
• SPI-Mode	Single Phase Frame Single Element per Frame 32 Bits per Element DataDelay 1 Bit
• Bezugspunkt	GND

- **"STEPPER MOTOR"-Anschluss**

Anschluss	10-polige Stiftleiste

Signale zur Ansteuerung von zwei Schrittmotoren:

• ENABLE und DIRECTION-Ausgänge	5 V, TTL
• CLOCK-Ausgang	5 V, TTL high-aktiv, 5 µs-Puls
• SWITCH-Eingang	TTL low-aktiv, intern über Pull-up-Widerstände (10 kΩ) mit +3,3 V verbunden
• Bezugspunkt	GND

14.1 Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV)

Die Konformität der RTC5 PCI-Karte mit der EG-Richtlinie 2004/108/EG (Elektromagnetische Verträglichkeit) wurde festgestellt.

Dazu wurde die RTC5-Karte in einen PC eingebaut und zusammen mit einem hurry*SCAN*® 25 Scan-Kopf (SL2-100-Schnittstelle) getestet.

Prüfbestimmungen

Nachweis zur Erfüllung der Schutzziele der EG-Richtlinie 2004/108/EG (CE-Konformität für EMV) auf der Basis von

– EN 61000-6-2: 2005
– EN 61000-6-4: 2007 + A1: 2011

Ergebnis

Die Prüflinge erfüllen die Spezifikationen.

14.2 Konformität mit FCC-Bestimmungen

Die RTC5 PCI-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 RTC5-Karte in einer kommerziellen Umgebung betrieben wird. Die RTC5-Karte erzeugt, verwendet und emittiert möglicherweise Hochfrequenzenergie und kann, sofern nicht in Übereinstimmung mit dieser Bedienungsanleitung installiert und verwendet, Störungen im Funkverkehr verursachen. Der Betrieb der RTC5-Karte in einem Wohngebiet wird wahrscheinlich schädliche Störungen verursachen; in diesem Fall ist der Benutzer verpflichtet, die Störungen auf eigene Kosten zu beseitigen.

15 Anhang A: Die RTC5 PC/104-*Plus*-Karte

15.1 Produktübersicht

15.1.1 Verwendung – Vergleich mit der RTC5 PCI-Karte

Die RTC5 PC/104-*Plus*-Karte von SCANLAB ist wie die RTC5 PCI-Karte zur synchronen Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten vorgesehen. Hardware-bedingt unterscheidet sie sich in den folgenden Punkten von der RTC5 PCI-Karte:

- Die RTC5 PC/104-*Plus*-Karte ist ein PC/104-*Plus*-Modul. Sie wird daher nicht – wie die RTC5 PCI-Karte – in einen PC-Steckplatz eingesteckt, sondern über ihre Stack-through-Steckverbinder (direkt oder indirekt) mit einer PC/104-*Plus*-CPU-Karte verbunden. Es können maximal vier RTC5 PC/104-*Plus*-Karten an einer CPU-Karte betrieben werden.
- Die RTC5 PC/104-*Plus*-Karte ist mit zusätzlichen Jumpern ausgestattet, was eine Anpassung der Karte bzgl. der Position in einem PC/104-Stack und der Spannungsversorgung ermöglicht.
- Anzahl, Art und Anordnung der Anschlussstecker zur Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten sind anders als auf der RTC5 PCI-Karte.

Ansonsten bietet die RTC5 PC/104-*Plus*-Karte die gleiche Funktionalität zur Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten wie die RTC5 PCI-Karte. Es gelten die gleichen Spezifikationen für die Signal-Eingänge und -Ausgänge wie bei der RTC5 PCI-Karte. Die Software-Installation und die Programmierung von Anwenderprogrammen erfolgt identisch zur RTC5 PCI-Karte.

RTC5 PC/104-*Plus*-Karte und RTC5 PCI-Karte nutzen das gleiche Softwarepaket (Treiber, DLL, Importdeklarationen, etc.) und den gleichen Befehlssatz.

15.1.2 Systemanforderungen

Hardware

Die RTC5 PC/104-*Plus*-Karte ist ein PC/104-*Plus*-Modul mit entsprechendem Formfaktor. Die Abmessungen der RTC5 PC/104-*Plus*-Karte sind in Abb. 64 aufgeführt (beachten Sie, dass der 100-polige MULTI-Stecker etwas über die Kante der Platine der RTC5 PC/104-*Plus*-Karte hinausragt). Die RTC5 PC/104-*Plus*-Karte ist mit PCI- und ISA-Stack-through-Steckverbindern ausgestattet und kann direkt oder indirekt (in einem Stapel aus mehreren PC/104-*Plus*-Modulen) mit einer PC/104-*Plus*-CPU-Karte verbunden werden. Die elektrische Anpassung an die Position der Karte in einem Stapel erfolgt gegebenenfalls mit Jumpern (siehe Seite 665).

Der Datentransfer erfolgt über den PCI-Bus der Karte konform mit der PCI-Local-Bus-Spezifikation Version 2.2. Es können Stapel von maximal vier PC/104-*Plus*-Modulen (zusätzlich zur PC/104-*Plus*-CPU-Karte) gebildet werden. PC/104-Standard-Module (ISA-Module, die nicht mit einem PCI-Bus ausgestattet sind) müssen gegebenenfalls in höhere Stapelpositionen gebracht werden.

RTC5 PC/104-*Plus*-Karten, die Master/Slave-synchronisiert werden sollen, müssen im Stapel unmittelbar aufeinander folgen.

Zur Spannungsversorgung der RTC5 PC/104-*Plus*-Karte muss die CPU-Karte mindestens eine Betriebsspannung von +5 V am PCI- oder ISA-Bus bereitstellen. Über einen Jumper auf der RTC5 PC/104-*Plus*-Karte kann eingestellt werden, über welchen Bus die Spannungsversorgung erfolgen soll (siehe Seite 666).

Von Vorteil, aber nicht zwingend erforderlich, ist eine zusätzliche ±12 V-Versorgungsspannung am PCI-Bus (siehe Seite 666).

Software

Für die RTC5 PC/104-*Plus*-Karte können die gleichen Treiber und DLL-Dateien wie für die RTC5 PCI-Karte verwendet werden. Zusätzlich zu den in Kapitel 2.2.2 "Software", Seite 26 für die RTC5 PCI-Karte aufgeführten Betriebssysteme unterstützen die RTC5-Treiber und RTC5-DLL-Dateien auch das Microsoft-Betriebssystem Windows XP Embedded (ab RTC5-Software-Paketen Version 2011_09_29 und später bzw. DLL-Versionen ≥ DLL 528: die 32-Bit- und 64-Bit-Versionen von Windows XP Embedded ≥ SP2). Es können maximal vier RTC5 PC/104-*Plus*-Karten an einer CPU-Karte betrieben werden.

Vorsicht!

- Die RTC5 PC/104-*Plus*-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Siehe hierzu auch Hinweis auf Seite 26.

15.1.3 Optionale Funktionalitäten

Die RTC5 PC/104-*Plus*-Karte kann mit den gleichen Funktionalitäten ausgestattet werden wie die RTC5 PCI-Karte (siehe Seite 27). Optionale Funktionalitäten müssen von SCANLAB freigeschaltet bzw. installiert werden.

15.1.4 Kennzeichnung

Die Seriennummer der RTC5 PC/104-*Plus*-Karte finden Sie auf einem weißem Aufkleber auf der Karte (Beschriftung "RTC SN…").

Die Artikelnummer und die Konfiguration der Karte werden auf dem jeweiligen Lieferschein aufgeführt (siehe auch "Optionale Funktionalitäten", Seite 27 und "Typenbezeichnung", Seite 656).

15.1.5 Typenbezeichnung

Aus der Artikelnummer einer RTC5 PC/104-*Plus*-Karte ergibt sich, in welcher Jumperkonfiguration (**JP1**-**JP8**) die Karte ausgeliefert wird. Die jeweilige Jumperkonfiguration wird zusätzlich durch eine dreiziffrige Typenbezeichung dargestellt:

Ziffer 1	=0: Jumper **JP1** offen (kein Signal) =1: Jumper **JP1** in Position 1-2 (5 V) =2: Jumper **JP1** in Position 2-3 (3,3 V)
Ziffer 2 (betr. Pin B40)	=0: Jumper **JP2**-**JP4** offen (kein Signal) =1: Jumper **JP2** geschlossen (+5 V) =2: Jumper **JP3** geschlossen (DATA7) =3: Jumper **JP4** geschlossen (GND)
Ziffer 3 (betr. Pin B42)	=0: Jumper **JP5**-**JP8** offen (kein Signal) =1: Jumper **JP5** geschlossen (+5 V) =2: Jumper **JP6** geschlossen (DATA7) =3: Jumper **JP7** geschlossen (GND) =4: Jumper **JP8** geschlossen (LATCH)

Beispiele:

- "Typ 000": Jumper **JP1**-**JP8** offen
- "Typ 124": **JP1** in Position1-2 (5 V Signal-Pegel), **JP3** geschlossen (DATA7 an Pin B40), **JP8** geschlossen (LATCH an Pin B42)

Die Jumper **JP10**, **JP12**, **JP17**, **JP18** und **JP19** stehen bei Auslieferung immer in der jeweiligen Standardeinstellung (siehe Seite 665).

15.1.6 Auspackanleitung und typischer Lieferumfang

- Nehmen Sie die RTC5 PC/104-*Plus*-Karte vorsichtig aus der Verpackung.
- Bewahren Sie sowohl die Umverpackung als auch den antistatischen Beutel, in dem die RTC5 PC/104-*Plus*-Karte ausgeliefert wird, auf, damit die Karte im Fall einer Reparatur wieder optimal verpackt an SCANLAB geschickt werden kann.
- 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 RTC5 PC/104-*Plus*-Karte und einer Daten-CD (mit der zugehörigen Software (s. u.) und diesem Manual). Unter Umständen sind aber auch Zusatzkomponenten wie Datenkabel oder Konverter enthalten (siehe "Optionale Zusatzkomponenten", Seite 657).

Mitgelieferte Software

Das mitgelieferte Software-Paket ist identisch zum RTC5 Software-Paket (siehe Seite 22). Es enthält u. a. Treiber für die (32 Bit oder 64 Bit) Microsoft-Betriebssysteme Windows 10, 8, 7, Vista, XP SP2, XP SP3 und Windows XP Embedded (≥ SP2), Korrektur-, DLL- und Programm-Dateien, sowie Utility-Dateien für C, C++ und C#.

15.1.7 Optionale Zusatzkomponenten

Zusätzlich zur RTC5 PC/104-*Plus*-Karte und zum zugehörigen Software-Paket sind bei SCANLAB – wie für die RTC5 PCI-Karte – die folgenden Komponenten erhältlich (in Kombination mit der RTC5 PC/104-*Plus*-Karte sollten nur Hardwareerweiterungen von SCANLAB verwendet werden):

- XY2-100-Konverter (siehe Seite 29).
- Datenkabel (siehe Seite 29).
- Laseradapter (siehe Seite 29).
- Slotblech mit einer 15-poligen Sub-D-Buchse zur Verwendung der Eingänge und Signale am LASER-Stecker (siehe Seite 661)

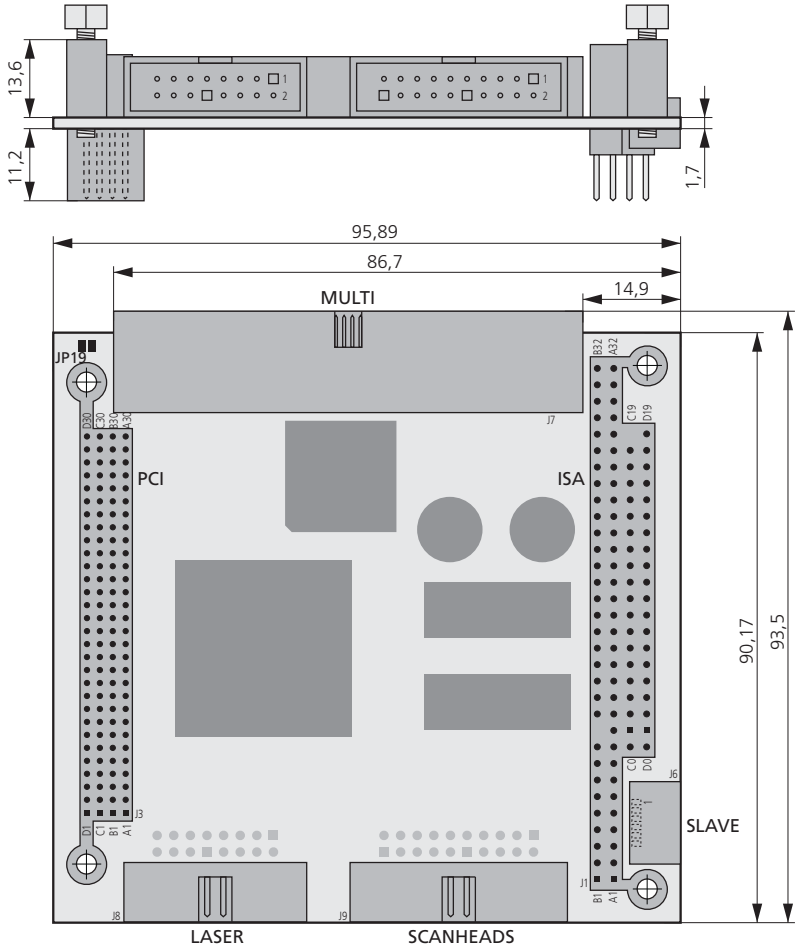
15.1.8 Ergänzende Software

Zur individuellen Anpassung von RTC-Korrekturdateien entsprechend der Daten einer Testmessung ist bei SCANLAB Spezialsoftware mit zugehörigem Handbuch aus der correXion-Reihe erhältlich (siehe auch Seite 137).

15.2 Layout und Schnittstellen

15.2.1 Stecker- und Jumperpositionen

Abb. 64 und Abb. 65 zeigen die Positionen der Anschlüsse und Jumper auf der Vorderseite und Rückseite der RTC5 PC/104-*Plus*-Karte. Alle Anschlüsse und Jumpereinstellungen werden in den folgenden Abschnitten beschrieben.



Legende

SCANHEADS . . primärer und sekundärer Scan-Kopf-Anschluss (20-poliger Wannenstecker, 2-reihig, 2.54 mm Pitch, Pinbelegung Seite 660); der sekundäre Scan-Kopf-Anschluss ist nur optional aktiviert

LASER . . . . . Laseranschluss, sowie digitale und analoge Ausgänge (16-poliger Wannenstecker, 2-reihig, 2.54 mm Pitch, Pinbelegung Seite 661)

MULTI . . . . . Multifunktionsstecker mit einem 16 Bit-Digital-Eingang, einem 16 Bit-Digital-Ausgang, einem 8-Bit-Digital-Ausgang, Encoder-Eingängen für "Processing on the fly"-Anwendungen, etc. (100-poliger Wannenstecker, 2-reihig, 1.27 mm Pitch, Pinbelegung Seite 661)

PCI . . . . . PC/104-*Plus* PCI-Bus (120-polige Buchse, 4-reihig, 2 mm Pitch, siehe Seite 660)

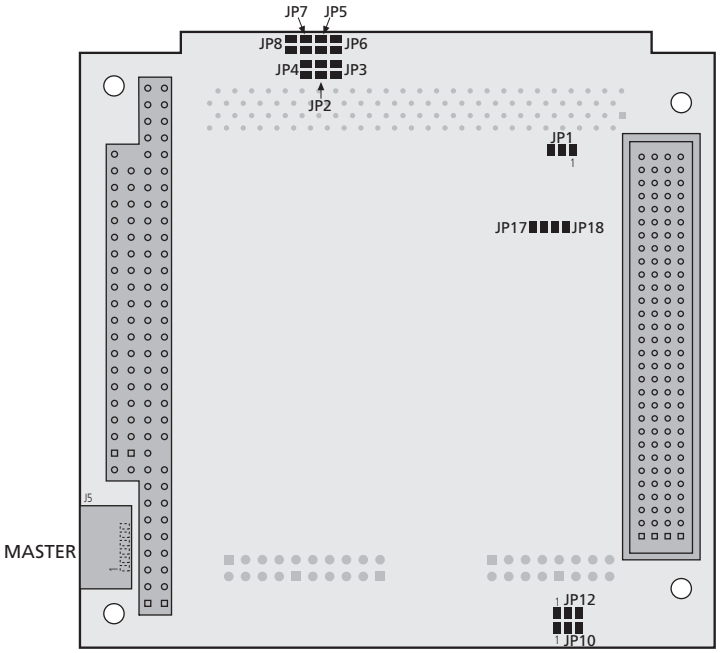
ISA . . . . . PC/104 Standard 8-Bit-ISA-Bus und 16-Bit-ISA-Bus (104-polige Buchse, 32 x 2 + 20 x 2 Pins, 2.54 mm Pitch, siehe Seite 660)

SLAVE . . . . . 6-poliger SLAVE-Anschluss (siehe "Master/Slave-Synchronisation", Seite 660)

JP19 . . . . . Jumper zum Festlegen, ob die +5 V-Spannungsversorgung der Karte über den PCI- oder über den ISA-Bus erfolgt (siehe Seite 666)

64

Layout und Abmessungen der RTC5 PC/104-*Plus*-Karte (oben: Seitenansicht; unten: Vorderseite)



Legende

Master . . . . . 6-poliger MASTER-Anschluss (siehe "Master/Slave-Synchronisation", Seite 660)
JP1 . . . . . Jumper zur Auswahl des Ausgangssignal-Pegels am MULTI-Stecker (siehe Seite 667)
JP2 - JP8 . . . . Jumper zur Konfiguration von Pin B40 und B42 des MULTI-Steckers (siehe Seite 667)
JP10+JP12 . . Jumper zum Festlegen, wie die maximale Ausgangsspannung von 10 V an den analogen Ausgängen ANALOG OUT1 und ANALOG OUT2 am MULTI-Stecker erzeugt wird (siehe Seite 666)
JP17+JP18 . . Jumper zur binären Festlegung der Stapeladresse, d.h. zur elektrischen Anpassung der Karte an die Position der Karte im Stapel (siehe Seite 665)

65

Layout der RTC5 PC/104-*Plus*-Karte (Rückseite)

15.2.2 Schnittstelle zur CPU-Karte

Der Datentransfer zwischen CPU-Karte und RTC5 PC/104-*Plus*-Karte erfolgt über den PCI-Bus. Die RTC5 PC/104-*Plus*-Karte ist mit 120-poligen PCI-Stack-through-Steckverbindern ausgestattet (siehe Abb. 64). Das Timing und die Signal-Pegel der PCI-Signale sind konform mit der PCI-Local-Bus-Spezifikation Version 2.2. 64-Bit-Erweiterungen, JTAG-, PRSNT- oder CLKRUN-Signale werden nicht unterstützt.

Am ISA-Bus verwendet die RTC5 PC/104-*Plus*-Karte nur die +5 V Versorgungsspannung an den Pins B3, B29 und D16 (zur Spannungsversorgung siehe auch Seite 666). Die anderen Signale des ISA-Busses werden lediglich über die 104-poligen ISA-Stack-through-Steckverbinder zu weiteren Karten im Stapel durchgeleitet und nicht von der RTC5 PC/104-*Plus*-Karte verwendet.

Vorsicht!

- Die RTC5 PC/104-*Plus*-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Siehe hierzu auch Hinweis auf Seite 26.

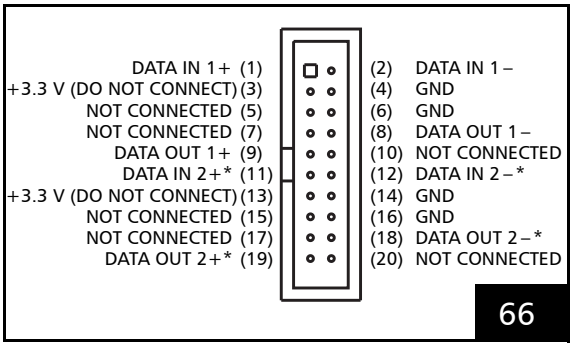
15.2.3 Master/Slave-Synchronisation

Sollen mehrere RTC5 PC/104-*Plus*-Karten mit synchroner Taktung in einem PC/104-Stack betrieben werden, dann müssen die RTC5 PC/104-*Plus*-Karten paarweise über die MASTER- und SLAVE-Anschlüsse miteinander verbunden und in benachbarte Stapelpositionen gesteckt werden. Verbinden Sie jeweils den MASTER-Anschluss einer Karte mit dem SLAVE-Anschluss einer anderen Karte. Geeignete Verbindungskabel sind bei SCANLAB erhältlich.

Siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93 und "Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation", Seite 239.

15.2.4 Schnittstellen zum Scan-System

Zur digitalen Ansteuerung von Scan-Systemen stehen am 20-poligen SCANHEADS-Stecker der primäre Scan-Kopf-Anschluss (Pins 1 bis 10) und der optional aktivierte sekundäre Scan-Kopf-Anschluss (Pins 11 bis 20) zur Verfügung (siehe Abb. 64). Dort werden Ansteuerwerte für das Scan-System ausgegeben und Statussignale empfangen, die vom Scan-System zurückgegeben werden. Abb. 66 zeigt die Pinbelegung des SCANHEADS-Steckers.



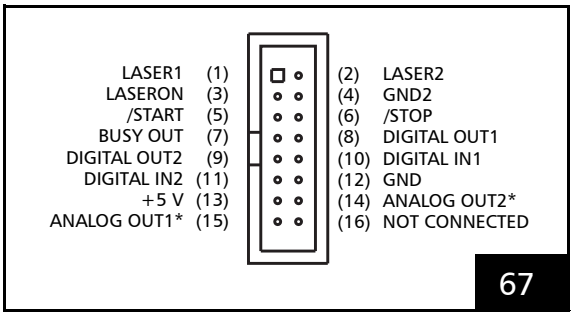
Pinbelegung des SCANHEADS-Steckers
(* Signal nur optional aktiviert)

Der SCANHEADS-Stecker stellt die gleichen Signale bereit wie die SCAN HEAD- und 2. SCAN HEAD-Anschlüsse der RTC5 PCI-Karte. Die Ansteuerung von Scan-Systemen (gegebenenfalls zusätzlich über einen XY2-100-Konverter) erfolgt identisch zur RTC5 PCI-Karte. Beachten Sie die entsprechenden Hinweise unter Kapitel 4.3 "Schnittstellen zum Scan-System", Seite 42.

15.2.5 Schnittstellen zum Laser und zur Peripherie

LASER-Stecker

Abb. 67 zeigt die Pinbelegung des 16-poligen LASER-Steckers. Dieser Stecker stellt die gleichen Signale bereit wie die 15-polige Sub-D-Laseranschluss-Buchse (LASER) der RTC5 PCI-Karte. Beachten Sie die entsprechenden Hinweise unter "Laseranschluss-Buchse", Seite 47.



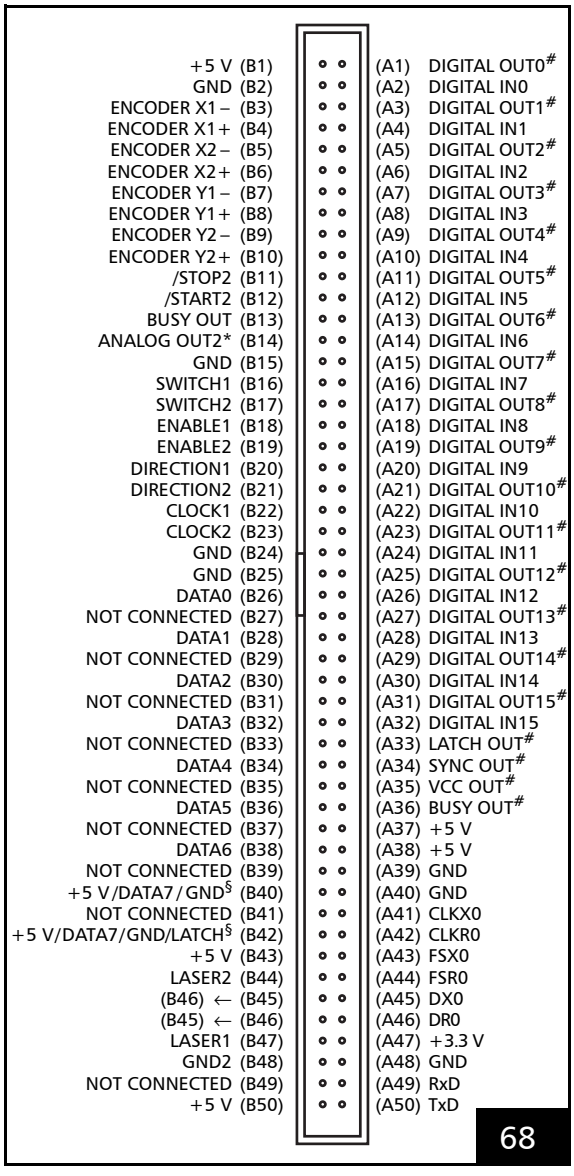
Pinbelegung des 16-poligen LASER-Steckers
(* abhängig von der Jumpereinstellung mit **JP10** und **JP12**)

Der Stecker stellt u.a. die zwei analogen Ausgänge ANALOG OUT1 und ANALOG OUT2 bereit (das ANALOG OUT2-Signal ist auch über den MULTI-Stecker verfügbar, siehe Abb. 68). Über die Jumper **JP10+JP12** auf der Rückseite der RTC5 PC/104-*Plus*-Karte kann festgelegt werden, wie die maximale Ausgangsspannung von 10 V dieser analogen Ausgänge erzeugt wird (siehe Seite 666).

Ein Slotblech mit einer 15-poligen Sub-D-Buchse zur Verwendung der Eingänge und Signale am LASER-Stecker (mit Pinbelegung wie bei der 15-poligen Sub-D-Laseranschluss-Buchse LASER der RTC5 PCI-Karte, siehe Abb. 10) ist bei SCANLAB erhältlich.

MULTI-Stecker

Abb. 68 zeigt die Pinbelegung des 100-poligen MULTI-Steckers. Dieser Multifunktionsstecker stellt die gleichen Signale bereit wie die Stiftleisten EXTENSION 1, EXTENSION 2, MARKING ON THE FLY, "RS232", "STEPPER MOTOR" und "SPI / I2C" auf der RTC5 PCI-Karte. Beachten Sie die Hinweise in der nachfolgenden Tabelle.



Pinbelegung des 100-poligen MULTI-Steckers
(* abhängig von der Jumpereinstellung mit **JP10** und **JP12**
§ abhängig von der Jumpereinstellung mit **JP2**-**JP8**
# Ausgangssignal-Pegel abh. von der Jumpereinstellung mit **JP1**)

Pin am MULTI-Stecker	Signal	Signalbeschreibung	Anschluss auf der RTC5 PCI-Karte
A1, A3, A5, A7, A9, A11, A13, A15, A17, A19, A21, A23, A25, A27, A29, A31	DIGITAL OUT0… DIGITAL OUT15	Gepufferter 16-Bit-Digital-Ausgang (Ausgangssignal-Pegel wahlweise 5 V oder 3.3 V, siehe Jumper **JP1**, Seite 667).	"EXTENSION 1-Stiftleiste", Seite 51
A33	LATCH OUT	LATCH-Signal zur Synchronisation der Datenübergabe am 16-Bit-Digital-Ausgang (Ausgangssignal-Pegel wahlweise 5 V oder 3.3 V, siehe Jumper **JP1**, Seite 667).	"EXTENSION 1-Stiftleiste", Seite 51
A2, A4, A6, A8, A10, A12, A14, A16, A18, A20, A22, A24, A26, A28, A30, A32	DIGITAL IN0… DIGITAL IN15	Geschützter 16-Bit-Digital-Eingang	"EXTENSION 1-Stiftleiste", Seite 51
A34	SYNC OUT	SYNC-Signal zur Synchronisation der Datenübergabe am 16-Bit-Digital-Eingang (Ausgangssignal-Pegel wahlweise 5 V oder 3.3 V, siehe Jumper **JP1**, Seite 667).	"EXTENSION 1-Stiftleiste", Seite 51
A36	BUSY OUT	BUSY-Status als BUSY OUT-Signal (Ausgangssignal-Pegel wahlweise 5 V oder 3.3 V, siehe Jumper **JP1**, Seite 667).	"EXTENSION 1-Stiftleiste", Seite 51
A35	VCC OUT	Mit Jumper **JP1** ausgewählter Signal-Pegel (5 V oder 3.3 V)	"EXTENSION 1-Stiftleiste", Seite 51
B26, B28, B30, B32, B34, B36, B38 (B40, B42)	DATA0…DATA6 (DATA7)	Gepufferter 8-Bit-Digital-Ausgang (DATA7 optional an B40, B42, abh. von der Jumpereinstellung mit **JP2**-**JP8**, siehe Seite 667)	"EXTENSION 2-Stiftleiste", Seite 52
(B42)	(LATCH)	LATCH-Signal zur Synchronisation der Datenübergabe am 8-Bit-Digital-Ausgang (optional an B42 abh. von der Jumpereinstellung mit **JP2**-**JP8**, siehe Seite 667)	"EXTENSION 2-Stiftleiste", Seite 52
B44, B47, B48	LASER2, LASER1, GND2	Die Lasersteuersignale LASER1 und LASER2 beziehen sich auf GND2. Ist die RTC5 PC/104-*Plus*-Karte mit Optokopplern ausgestattet, so sind GND2 und die Laserausgangssignale galvanisch von der CPU-Masse GND entkoppelt. Andernfalls sind die Massen GND2 und GND identisch.	"EXTENSION 2-Stiftleiste", Seite 52
B3 - B10	ENCODER X1/2± ENCODER Y1/2±	Encoder-Eingänge	"MARKING ON THE FLY-Stiftleiste", Seite 53
B11, B12	/STOP2, /START2	Externes Stop-Signal, Externes Start-Signal	"MARKING ON THE FLY-Stiftleiste", Seite 53

Pin am MULTI-Stecker	Signal	Signalbeschreibung	Anschluss auf der RTC5 PCI-Karte
B13	BUSY OUT	BUSY-Status als BUSY OUT-Signal (5 V Ausgangssignal-Pegel)	"MARKING ON THE FLY-Stiftleiste", Seite 53
B14	ANALOG OUT2	12-Bit-Analogausgang (Das Signal ist auch über den LASER-Stecker verfügbar – siehe Seite 661. Über die Jumper **JP10+JP12** auf der Rückseite der RTC5 PC/104-*Plus*-Karte kann festgelegt werden, wie die maximale Ausgangs-spannung von 10 V erzeugt wird – siehe Seite 666).	"MARKING ON THE FLY-Stiftleiste", Seite 53 und "Laseranschluss-Buchse", Seite 47
A49, A50	RxD, TxD	RS232-Schnittstelle	"RS232" (siehe Seite 54)
A41 - A46	CLKX0, CLKR0, FSX0, FSR0, DX0, DR0	McBSP/SPI-Schnittstelle	"SPI / I2C" (siehe Seite 54)
B16	SWITCH1/2, ENABLE1/2, DIRECTION1/2, CLOCK1/2	Signale zur Ansteuerung zweier Schrittmotoren	"STEPPER MOTOR" (siehe Seite 58)
A37, A38, B1 (B40, B42)	+5 V	5 V Ausgangssignalpegel (optional an B40, B42 abh. von der Jumpereinstellung mit **JP2-JP8**, siehe Seite 667)	EXTENSION 1, EXTENSION 2, MARKING ON THE FLY
A47	+3.3 V	3,3 V Ausgangssignalpegel	"SPI / I2C"
A39, A40, A48, B2, B15, B24, B25, B43, B50 (B40, B42)	GND	CPU-Masse (optional an B40, B42 abh. von der Jumpereinstellung mit **JP2-JP8**, siehe Seite 667)	EXTENSION 1, EXTENSION 2, MARKING ON THE FLY, "RS232", "STEPPER MOTOR", "SPI / I2C"
B45, B46		Intern miteinander verbunden	EXTENSION 2, "RS232"

15.3 Installation und Inbetriebnahme

Die Installation der RTC5 PC/104-*Plus*-Karte erfolgt in den folgenden Schritten:

(1) Konfiguration der Jumper

(2) Hardware-Installation: Einbau der Karte

(3) Installation der Treiber

(4) Installation der RTC5-Software

Gehen Sie in der vorgeschlagenen Reihenfolge vor.

Für einen Funktionstest nach der Installation kann z. B. das mitgelieferte HPGL-Converter-Programm verwendet werden.

15.3.1 Sicherheit bei Installation und Inbetriebnahme

Lesen Sie diese Betriebsanweisungen vollständig, bevor Sie mit der Installation und Inbetriebnahme der RTC5 PC/104-*Plus*-Karte beginnen. Beachten Sie insbesondere alle Sicherheitshinweise dieser Anleitung. Beachten Sie auch die Hinweise unter Kapitel 3 “Sicherheit bei Installation und Inbetriebnahme”, Seite 39.

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

Vorsicht!

- 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.
- Vermeiden Sie bei Lagerung und Betrieb der RTC5 PC/104-*Plus*-Karte elektromagnetische Felder und statische Elektrizität. Die Elektronik der Karte kann durch solche Felder zerstört werden.
- Die RTC5 PC/104-*Plus*-Karte sollte stets in einer elektrostatisch neutralen Umgebung im mitgelieferten antistatischen Beutel gelagert werden.
- Die zulässige Betriebstemperatur beträgt 15 °C bis 60 °C.
- Die Karte muss bei einer Temperatur zwischen −20 °C und +60 °C gelagert werden.

15.3.2 Jumpereinstellungen

Bevor die RTC5 PC/104-*Plus*-Karte in einen PC/104-Stack eingebaut wird, muss sie gegebenenfalls über die Jumper auf der Platine konfiguriert werden. Die RTC5 PC/104-*Plus*-Karte kann von SCANLAB in unterschiedlichen Jumperkonfigurationen ausgeliefert werden (siehe auch "Typenbezeichnung", Seite 656). Die Jumper sind Lötverbindungen und können vom Kunden mit einem Lötkolben auch nachträglich umkonfiguriert werden.

Wenn Sie die werkseigene Einstellung nicht verändern wollen, so fahren Sie mit der Installation bei "Einbau der RTC5 PC/104-Plus-Karte", Seite 668 fort.

Vorsicht!

- Achten Sie bei einer Änderung der Jumpereinstellung darauf, die Elektronik auf der Karte nicht zu beschädigen!

Jumper 17+18: Binäre Festlegung der Stapeladresse

Über die Jumper **JP17+JP18** auf der Rückseite der RTC5 PC/104-*Plus*-Karte (siehe Abb. 65) kann die Stapeladresse der Karte festgelegt und damit die Karte elektrisch an die Position der Karte im PC/104-Stack angepasst werden. SCANLAB empfiehlt, die Stapeladresse wie folgt zu vergeben (zur PCI-Stack-Position siehe Abb. 70):

PCI-Stack-Position	Stapel-adresse	JP17	JP18
1	0 *	geschlossen	geschlossen
2	1	geschlossen	offen
3	2	offen	geschlossen
4	3	offen	offen

* Standardkonfiguration

Durch die Auswahl der Stapeladresse wird sichergestellt, dass jeder RTC5 PC/104-*Plus*-Karte innerhalb eines Stapels jeweils eines der vier CLK-, IDSEL-, REQ- und GNT-Signale am PCI-Bus eindeutig zugeordnet wird:

Stapel-adresse	CLK	IDSEL	REQ	GNT
0	CLK0	IDSEL0	REQ0	GNT0
1	CLK1	IDSEL1	REQ1	GNT1
2	CLK2	IDSEL2	REQ2	GNT2
3	CLK3	IDSEL3	REQ3	GNT3

Die Auswahl der REQ- und GNT-Signale spielt nur für Bus Master-Karten eine Rolle. RTC5 PC/104-*Plus*-Karten sind Target-Geräte und daher nicht auf Master-Stapelpositionen angewiesen.

Durch die Auswahl der Stapeladresse entsprechend der PCI-Stack-Position ergeben sich optimale Relationen zwischen den (für die verschiedenen PCI-Stack-Positionen unterschiedlichen) Längen der Signalleitungen für eine stabile PCI-Bus-Datenübertragung der zugeordneten Signale.

Hinweise

- Die Stapeladresse der RTC5 PC/104-*Plus*-Karte ist bei Auslieferung standardmäßig auf 0 eingestellt. Falls die Karte direkt auf die CPU-Karte aufgesteckt wird (ohne weitere Module dazwischen), muss die Einstellung von **JP17+JP18** nicht verändert werden.

JP19: +5 V Spannungsversorgung

Die RTC5 PC/104-*Plus*-Karte benötigt eine Versorgungsspannung von +5 V. Da diese nicht von allen CPU-Karten am PCI-Bus, dagegen in der Regel immer am ISA-Bus (sofern vorhanden) bereitgestellt wird, ist die RTC5 PC/104-*Plus*-Karte bei Auslieferung so eingestellt, dass die +5 V Spannungsversorgung ausschließlich über den ISA-Bus erfolgt (**JP19** offen).

Für eine CPU-Karte ohne ISA-Bus oder falls keine ISA-Bus-Verbindung mit der CPU-Karte besteht, kann durch Umjumpern von Jumper **JP19** (auf der Vorderseite der RTC5 PC/104-*Plus*-Karte, siehe Abb. 64) die Spannungsversorgung auch über den PCI-Bus erfolgen, sofern der PCI-Bus eine +5 V Versorgungsspannung bereitstellt (überprüfen Sie die entsprechenden Spezifikationen Ihrer CPU-Karte).

JP19	+5 V Spannungsversorgung
offen *	über den ISA-Bus
geschlossen	über den PCI-Bus (nur wenn ISA-Bus *nicht* verfügbar)

* Standardkonfiguration

Vorsicht!

- **JP19** darf *nicht* geschlossen werden, wenn die RTC5 PC/104-*Plus*-Karte sowohl über den PCI-Bus als auch über den ISA-Bus mit der CPU-Karte verbunden ist und die CPU-Karte an beiden Bussen eine +5 V Versorgungsspannung bereitstellt. Andernfalls erfolgt die Spannungsversorgung gleichzeitig über beide Busse, was die Elektronik der RTC5 PC/104-*Plus*-Karte beschädigen könnte.

JP10+12: 10 V Ausgangsspannung für ANALOG OUT1 und ANALOG OUT2

Über die Jumper **JP10+JP12** auf der Rückseite der RTC5 PC/104-*Plus*-Karte (siehe Abb. 65) kann festgelegt werden, wie die maximale Ausgangsspannung von 10 V an den analogen Ausgängen ANALOG OUT1 und ANALOG OUT2 am LASER- und MULTI-Stecker erzeugt wird:

Jumper JP10+JP12	10 V Ausgangsspannung
Position 1-2 	wird aus der ±12 V Versorgungsspannung am PCI-Bus erzeugt
Position 2-3* 	wird aus der +5 V Versorgungsspannung am PCI/ISA-Bus erzeugt
offen 	wird nicht bereitgestellt

* Standardkonfiguration

Vorsicht!

- Achten Sie darauf, dass **JP10** und **JP12** nicht gemischt gejumpert werden (also z.B. nicht **JP10** in Position 1-2 und **JP12** in Position 2-3). Andernfalls wird die Elektronik der RTC5 PC/104-*Plus*-Karte beschädigt.

Gemäß der PCI-Local-Bus-Spezifikation Version 2.2 wird am PCI-Bus (Pins A29 und A30) eine Versorgungsspannung von ±12 V bereitgestellt. Tatsächlich wird diese Spannung allerdings nicht von allen CPU-Karten bereitgestellt. Deshalb ist die RTC5 PC/104-*Plus*-Karte bei Auslieferung so eingestellt, dass die 10 V Ausgangsspannung aus der +5 V-Versorgungsspannung (die entweder am PCI-Bus oder am ISA-Bus bereitgestellt wird, siehe "JP19: +5 V Spannungsversorgung") erzeugt wird (**JP10** und **JP12** stehen beide in Position 2-3). Der maximale Ausgangsstrom ist dann +5 mA (Source only).

Falls die CPU-Karte die ±12 V Versorgungsspannung aber tatsächlich bereitstellt (überprüfen Sie die entsprechenden Spezifikationen Ihrer CPU-Karte), kann wie folgt umgejumpert werden: beide Jumper (**JP10** und **JP12**) in Position 1-2. Der maximale Ausgangsstrom ist dann ±5 mA (Source and Sink).

Jumper JP1: Auswahl des Ausgangssignal-Pegels am MULTI-Stecker

Mit dem Jumper **JP1** auf der Rückseite der RTC5 PC/104-*Plus*-Karte (siehe Abb. 65) kann für den 16-Bit-Digital-Ausgang (Pins A1, A3, A5, A7, A9, A11, A13, A15, A17, A19, A21, A23, A25, A27, A29, A31), das LATCH-Signal (Pin A33), das SYNC-Signal (Pin A34), das BUSY OUT-Signal (Pin A36) und das VCC_OUT-Signal (Pin A35) des MULTI-Steckers zwischen einem Signal-Pegel von 5 V oder 3,3 V ausgewählt werden (siehe auch Seite 661). Dies entspricht der Signal-Pegel-Auswahl an der EXTENSION 1-Stiftleiste der RTC5 PCI-Karte (siehe Seite 51).

Jumper JP1		Signal-Pegel
Position 1-2		5 V
Position 2-3		3,3 V
offen		keine Ausgangssignale

Vorsicht!

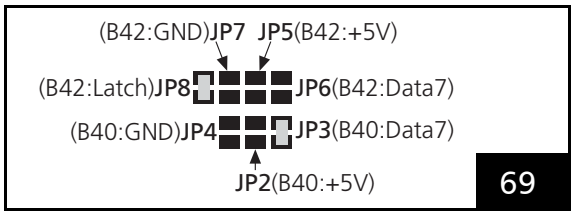
- Achten Sie darauf, dass nicht alle drei Pins des Jumpers verbunden werden. Andernfalls wird die Elektronik der RTC5 PC/104-*Plus*-Karte beschädigt.

Die RTC5 PC/104-*Plus*-Karte wird von SCANLAB in unterschiedlichen Jumperkonfigurationen für **JP1** ausgeliefert (siehe auch "Typenbezeichnung", Seite 656).

Jumper JP2-JP8: Konfiguration von Pin B40 und Pin B42 am MULTI-Stecker

Über die Jumper **JP2-JP8** auf der Rückseite der RTC5 PC/104-*Plus*-Karte (siehe Abb. 65) können Pin B40 und B42 des MULTI-Steckers konfiguriert werden (siehe auch Seite 661). Dies entspricht den Konfigurationsmöglichkeiten an Pin (15) und (17) der EXTENSION 2-Stiftleiste der RTC5 PCI-Karte (siehe Seite 52).

Das höchstwertige Bit (DATA7) des 8-Bit-Ausgangswertes kann Pin B40 oder Pin B42 zugewiesen werden. Alternativ können beide Pins auch permanent auf +5 V (HIGH) oder GND (LOW-Pegel) gelegt werden. Pin B42 kann alternativ auch das LATCH-Signal zugeordnet werden. Die Zuweisung erfolgt durch Schließen des entsprechend beschrifteten Jumpers. Abb. 69 zeigt als Beispiel die Jumper-Einstellung, mit der DATA7 Pin B40 und das LATCH-Signal Pin B42 zugeordnet wird.



69

Beispielkonfiguration für Jumper JP2-JP8:
Pin B40: DATA7, Pin B42: LATCH-Signal

Vorsicht!

- Achten Sie darauf, dass nicht mehr als *einer* der drei Jumper für Pin B40 geschlossen ist und auch nicht mehr als *einer* der vier Jumper für Pin B42. Andernfalls wird die Elektronik der RTC5 PC/104-*Plus*-Karte beschädigt.

Die RTC5 PC/104-*Plus*-Karte wird von SCANLAB in unterschiedlichen Jumperkonfigurationen für **JP2-JP8** ausgeliefert (siehe auch "Typenbezeichnung", Seite 656).

15.3.3 Einbau der RTC5 PC/104-*Plus*-Karte

Abb. 70 zeigt einen typischen Modul-Stapel (PC/104-Stack) bestehend aus einer CPU-Karte, einer RTC5 PC/104-*Plus*-Karte, einem weiteren PC/104-*Plus*-Modul, einem 16-Bit-PC/104-Modul und einem 8-Bit-PC/104-Modul.

Die Module sind in der Regel mit Stack-through-Steckverbindern ausgestattet, so dass die Signale zum jeweils nachfolgenden Modul im Stapel durchgeleitet werden. Sie können dadurch sowohl ober- als auch unterhalb von anderen Modulen und auf beiden Seiten der CPU-Karte installiert werden.

PC/104-*Plus*-Module sind mit PCI- *und* ISA-Bus ausgestattet. PC/104-Standard-Module sind dagegen nur mit einem ISA-Bus ausgestattet und unterbrechen daher gegebenenfalls den PCI-Datentransfer innerhalb eines Stapels. Da der Datentransfer bei der RTC5 PC/104-*Plus*-Karte über deren PCI-Bus erfolgt, müssen PC/104-Standard-Module gegebenenfalls in höhere Stapelpositionen gebracht werden.

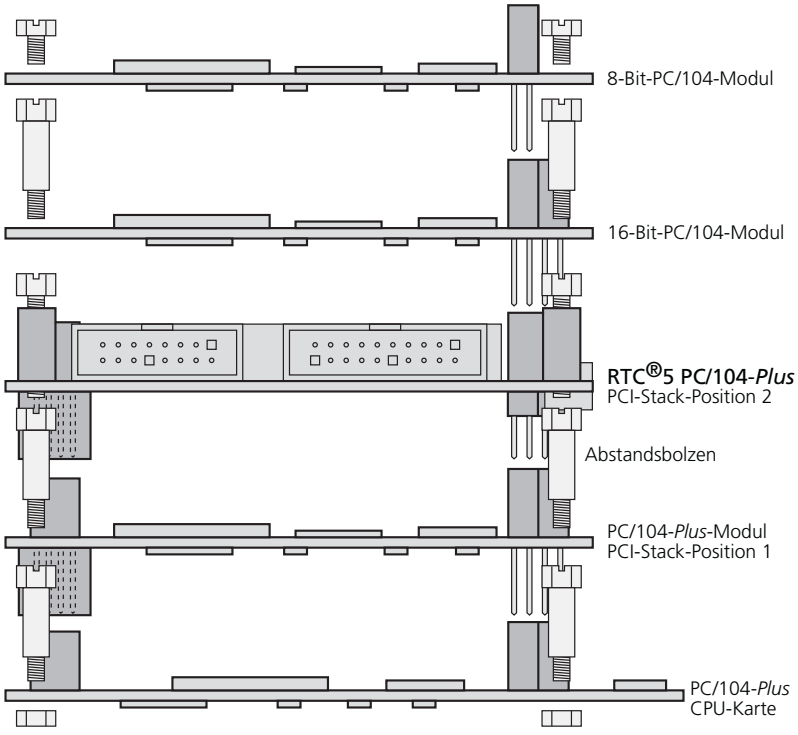
Die maximale Stapelhöhe für den PCI-Bus beträgt vier PC/104-*Plus*-Module (zusätzlich zur CPU-Karte). Die Module müssen aufeinander folgend auf der gleichen Seite der CPU-Karte gestapelt werden. Die elektrische Anpassung an die Position der Karte in einem Stapel erfolgt gegebenenfalls mit Jumpern (siehe Seite 665).

RTC5 PC/104-*Plus*-Karten, die Master/Slave-synchronisiert werden sollen, müssen im Stapel unmittelbar aufeinander folgen.

Zur Stabilisierung werden alle Module untereinander und mit der CPU-Karte mit Schrauben bzw. mit Abstandsbolzen verbunden.

Vorsicht!

- Sollen mehrere PCI-Karten in einem PC/104-*Plus*-Stapel an einer PC/104-*Plus*-CPU-Karte betrieben werden, so sollte die gesamte Konfiguration (d.h. die Funktionsfähigkeit aller Karten im Stapel) getestet werden. Bei manchen CPU-Karten kann es hier zu Problemen mit dem PCI-Bus kommen. Zur Überprüfung der Funktionsfähigkeit einer RTC5 PC/104-*Plus*-Karte im Stapel sollte mindestens getestet werden, ob **rtc5_count_cards** die richtige Anzahl von Karten liefert.



70

Typischer Modul-Stapel (Stack)

Vorsicht!

- Die Installation muss an einem Arbeitsplatz durchgeführt werden, der den ESD-Schutzvorschriften entspricht. Beachten Sie beim Einbau der Karte auch die Anweisungen in der Bedienungsanleitung Ihres Controllers.
- Berühren Sie die RTC5 PC/104-*Plus*-Karte nicht an den elektrischen Kontakten.
- Schützen Sie die Karte vor Feuchtigkeit, Staub, korrosiven Dämpfen und mechanischen Belastungen.

Der Einbau der RTC5 PC/104-*Plus*-Karte in den PC/104-Stack erfolgt in den folgenden Arbeitsschritten:

- Fahren Sie das Betriebssystem herunter, schalten Sie den Controller ab und trennen Sie ihn von der Stromversorgung.
- Trennen Sie alle Peripheriegeräte wie Monitor, Maus, Tastatur, Drucker etc. von ihrem Controller.
- Bringen Sie den Controller für den Einbau der Karte an einen Arbeitsplatz, der den ESD-Schutzvorschriften gerecht wird.
- Entfernen Sie die Abdeckung Ihres Controllers.
- Überprüfen Sie, ob die CPU-Karte mit einem PC/104-*Plus*-Anschluss ausgestattet ist und überprüfen Sie die Stapelhöhe des Stacks (ein PCI-Stack darf maximal vier PC/104-*Plus*-Module enthalten).
- Nehmen Sie die RTC5 PC/104-*Plus*-Karte aus dem antistatischen Beutel. Berühren Sie die Karte nicht an den Kontaktstellen.
- Wenn Sie die Signale an den SCANHEADS-, LASER und/oder MULTI-Stecker der RTC5 PC/104-*Plus*-Karte verwenden wollen, dann schließen Sie jetzt entsprechende Verbindungskabel an (die Verbindungskabel dürfen erst nach Installation der RTC5 PC/104-*Plus*-Karte mit den entsprechenden Geräten verbunden werden). Alle Anschlüsse und Signale der RTC5 sind in Kapitel 15.2 "Layout und Schnittstellen", Seite 658 beschrieben.
- Fügen Sie die RTC5 PC/104-*Plus*-Karte in die gewünschte Position (1 bis 4) im PC/104-Stack ein:
  - Die Position im Stack sollte zur über Jumper **JP17+JP18** festgelegten Stapeladresse passen (siehe Seite 665). Überprüfen Sie auch die anderen Jumpereinstellungen.
  - Überprüfen Sie, ob es zu Kollisionen zwischen Bauteilen der RTC5 PC/104-*Plus*-Karte und Bauteilen der CPU-Karte (bzw. eines benachbarten Moduls im Stapel) kommt.
  - Sollen mehrere RTC5 PC/104-*Plus*-Karten mit Master/Slave-Synchronisation in einem PC/104-Stack betrieben werden, dann müssen die RTC5 PC/104-*Plus*-Karten paarweise über die MASTER- und SLAVE-Anschlüsse miteinander verbunden und in unmittelbar aufeinander folgende Stapelpositionen gebracht werden (siehe auch "Master/Slave-Synchronisation", Seite 660).
  - Beachten Sie die obigen Hinweise zur Stapelung von PC/104-*Plus*-Modulen.
  - Stellen Sie sicher, dass die PCI- und ISA-Steckverbinder der Karte korrekt mit der CPU-Karte bzw. den benachbarten Modulen im Stapel verbunden sind.
  - Verbinden Sie die Karte mit Schrauben bzw. mit Abstandsbolzen mit der CPU-Karte bzw. mit benachbarten Modulen im Stapel.
- Schließen Sie die Abdeckung Ihres Controllers wieder.
- Bringen Sie den Controller an seinen Einsatzort und verbinden Sie alle Peripheriegeräte.
- Verbinden Sie gegebenenfalls einen oder mehrere Stecker der RTC5 PC/104-*Plus*-Karte mit dem Scan-System (gegebenenfalls unter Einbindung des XY2-100-Konverters), dem Laser, mit einem Handling-System oder anderen Zusatzgeräten der Gesamtmaschine.
- Überprüfen Sie die Verbindungen und schalten Sie Ihren Controller wieder an.

15.3.4 Installation der Treiber und RTC5-Software

Für die RTC5 PC/104-*Plus*-Karte werden die gleichen Treiber und sonstigen Software-Dateien benötigt wie für die RTC5 PCI-Karte. Auch die Installation erfolgt wie bei der RTC5 PCI-Karte (siehe Seite 61 und Seite 62).

Hinweise

- Die RTC5 PC/104-*Plus*-Karte wird unter Windows als PCI-Karte geführt.
- Stellen Sie sicher, dass Sie bei der Installation der Treiber mit Administratoren-Rechten angemeldet sind (nicht notwendig bei der späteren Verwendung der RTC5 PC/104-*Plus*-Karte).

Vorsicht!

- Die RTC5 PC/104-*Plus*-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Deaktivieren Sie insbesondere den Windows-Sleep-Modus (dieser ist bei einigen Windows-Betriebssystemen standardmäßig voreingestellt). Siehe hierzu auch Hinweis auf Seite 26.

15.3.5 Funktionstest, Anwenderprogramme und Demoprogramme

Zum Testen der Scan-Kopf-Steuerung können Sie den mitgelieferten HPGL-Converter HPGL.EXE verwenden (siehe Seite 63).

Die Programmierung von Anwenderprogrammen erfolgt wie bei der RTC5 PCI-Karte (siehe Seite 64). Es können die gleichen Demoprogramme wie bei der RTC5 PCI-Karte genutzt werden.

Achtung!

- **Schalten Sie immer zuerst den Controller mit der RTC5 PC/104-*Plus*-Karte und die Spannungsversorgung des Scan-Systems an, und erst dann den Laser. Andernfalls besteht die Gefahr, dass der Laserstrahl in eine unkontrollierte Richtung abgelenkt wird.**

Vorsicht!

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

15.4 Technische Spezifikationen der RTC5 PC/104-*Plus*-Karte

Systemanforderungen

PC/104-*Plus*-CPU-Karte
+ freie Position im PC/104-Stack

Betriebssystem	Microsoft (32 oder 64 Bit) Windows 10, 8, 7, Vista, XP SP2, XP SP3 und Windows XP Embedded ≥ SP2

Abmessungen

Länge x Breite	96 mm x 93,5 mm
Stapelhöhe	16,8 mm
max. Bauteilhöhe	9,3 mm

Gewicht ca. 115 g

Betriebs- und Lagerbedingungen

zulässige Betriebstemperatur	+15 °C bis +60 °C
Lagertemperatur	−20 °C bis +60 °C
Umgebungsbedingungen	nicht-kondensierend nicht-korrodierend

Anschlüsse zur PC/104-*Plus*-CPU-Karte (bzw. zum PC/104-Stack)

PC/104-*Plus* PCI-Bus	120-polige Stack-Through-Steckverbinder inkl. Kontaktschutz (Shroud), Betriebsspannung: +5 V
ISA-Bus	104-polige Stack-Through-Steckverbinder inkl. Sockel für Durchleitung der Signale und +5 V Spannungsversorgung

Sonstige Anschlüsse (Signal-Spezifikationen wie bei der RTC5 PCI-Karte

Schnittstelle zum Scan-System	SCANHEADS-Stecker (20-poliger Wannenstecker) mit primärem und sekundärem Scan-Kopf-Anschluss (der sekundäre Scan-Kopf-Anschluss ist nur optional aktiviert)
Schnittstelle zum Laser und zur Peripherie	LASER-Stecker (16-poliger Wannenstecker)
	MULTI-Multifunktions-stecker (100-poliger Wannenstecker)

16 Anhang B: Die RTC5 PCI-Express-Karte

16.1 Produktübersicht

16.1.1 Verwendung – Vergleich mit der RTC5 PCI-Karte

Die RTC5 PCI-Express-Karte von SCANLAB ist wie die RTC5 PCI-Karte zur synchronen Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten vorgesehen. Hardware-bedingt unterscheidet sie sich in den folgenden Punkten von der RTC5 PCI-Karte:

- Die RTC5 PCI-Express-Karte besitzt ein PCI-Express-Interface und muss in einen PCI-Express-Steckplatz des PCs eingesteckt werden.
- An der Stifteiste "SPI / I2C" werden statt einer I^2C-Schnittstelle zwei 10 V-Analogeingänge bereitgestellt (siehe Seite 676).

Ansonsten bietet die RTC5 PCI-Express-Karte die gleiche Funktionalität zur Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten wie die RTC5 PCI-Karte. Anzahl, Art und Anordnung der Anschlussstecker zur Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten und die Jumper zur Konfiguration sind identisch zur RTC5 PCI-Karte. Es gelten die gleichen Spezifikationen für die Signal-Eingänge und -Ausgänge wie bei der RTC5 PCI-Karte (mit oben genannter Ausnahme). Die Software-Installation und die Programmierung von Anwenderprogrammen erfolgt identisch zur RTC5 PCI-Karte. RTC5 PCI-Express-Karte und RTC5 PCI-Karte nutzen das gleiche Softwarepaket (Treiber, DLL, Importdeklarationen, etc.) und den gleichen Befehlssatz.

16.1.2 Systemanforderungen

Hardware

Die RTC5 PCI-Express-Karte kann in jeden Windows-PC mit PCI-Express-Bus und mindestens einem freien PCI-Express-Steckplatz installiert werden. Die Abmessungen der RTC5 PCI-Express-Karte sind in Abb. 71 aufgeführt.

RTC5 PCI-Express-Karten, die Master/Slave-synchronisiert werden sollen, müssen in benachbarte PCI-Express-Steckplätze gesteckt werden.

Software

Für die RTC5 PCI-Express-Karte können die gleichen Treiber und DLL-Dateien wie für die RTC5 PCI-Karte verwendet werden (zu den unterstützten Betriebssystemen siehe Abschnitt 2.2.2 auf Seite 26).

Vorsicht!

- Die RTC5 PCI-Express-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Express-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Siehe hierzu auch Hinweis auf Seite 26.

16.1.3 Optionale Funktionalitäten

Die RTC5 PCI-Express-Karte kann mit den gleichen Funktionalitäten ausgestattet werden wie die RTC5 PCI-Karte (siehe Seite 27). Optionale Funktionalitäten müssen von SCANLAB freigeschaltet bzw. installiert werden.

16.1.4 Kennzeichnung

Die Seriennummer der RTC5 PCI-Express-Karte finden Sie auf einem weißem Aufkleber auf der Karte (Beschriftung "RTC SN…").

Die Artikelnummer und die Konfiguration der Karte werden auf dem jeweiligen Lieferschein aufgeführt (siehe auch "Optionale Funktionalitäten", Seite 27 und "Typenbezeichnung", Seite 656).

16.1.5 Typenbezeichnung

Aus der Artikelnummer einer RTC5 PCI-Express-Karte ergibt sich, in welcher Jumperkonfiguration (**JP1**-**JP8**) die Karte ausgeliefert wird. Die jeweilige Jumperkonfiguration wird zusätzlich durch eine dreiziffrige Typenbezeichung dargestellt. Die Typenbezeichnung ist identisch wie bei der RTC5 PCI-Karte (siehe "Jumperkonfigurationen und Typenbezeichnung", Seite 28).

16.1.6 Auspackanleitung und typischer Lieferumfang

- Nehmen Sie die RTC5 PCI-Express-Karte vorsichtig aus der Verpackung.
- Bewahren Sie sowohl die Umverpackung als auch den antistatischen Beutel, in dem die RTC5 PCI-Express-Karte ausgeliefert wird, auf, damit die Karte im Fall einer Reparatur wieder optimal verpackt an SCANLAB geschickt werden kann.
- 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 RTC5 PCI-Express-Karte und einer Daten-CD (mit der zugehörigen Software (s. u.) und diesem Manual). Unter Umständen sind aber auch Zusatzkomponenten wie Datenkabel oder Konverter enthalten (siehe "Optionale Zusatzkomponenten", Seite 657).

Mitgelieferte Software

Das mitgelieferte Software-Paket ist identisch zum RTC5 Software-Paket (siehe Seite 22). Es enthält u. a. Treiber für die (32 Bit oder 64 Bit) Microsoft-Betriebssysteme Windows 10, 8, 7, Vista, XP SP2, XP SP3, Korrektur-, DLL- und Programm-Dateien, sowie Utility-Dateien für C, C++ und C#.

16.1.7 Optionale Zusatzkomponenten

Zusätzlich zur RTC5 PCI-Express-Karte und zum zugehörigen Software-Paket sind bei SCANLAB – wie für die RTC5 PCI-Karte – die folgenden Komponenten erhältlich (in Kombination mit der RTC5 PCI-Express-Karte sollten nur Hardwareerweiterungen von SCANLAB verwendet werden):

- XY2-100-Konverter (siehe Seite 29).
- Datenkabel (siehe Seite 29).
- Laseradapter (siehe Seite 29).
- Slotblech mit einer 9-poligen Sub-D-Buchse für die Stiftleiste 2. SCAN HEAD (siehe Seite 43).
- Slotblech mit einer 15-poligen Sub-D-Buchse für die Stiftleiste MARKING ON THE FLY (siehe Seite 53).

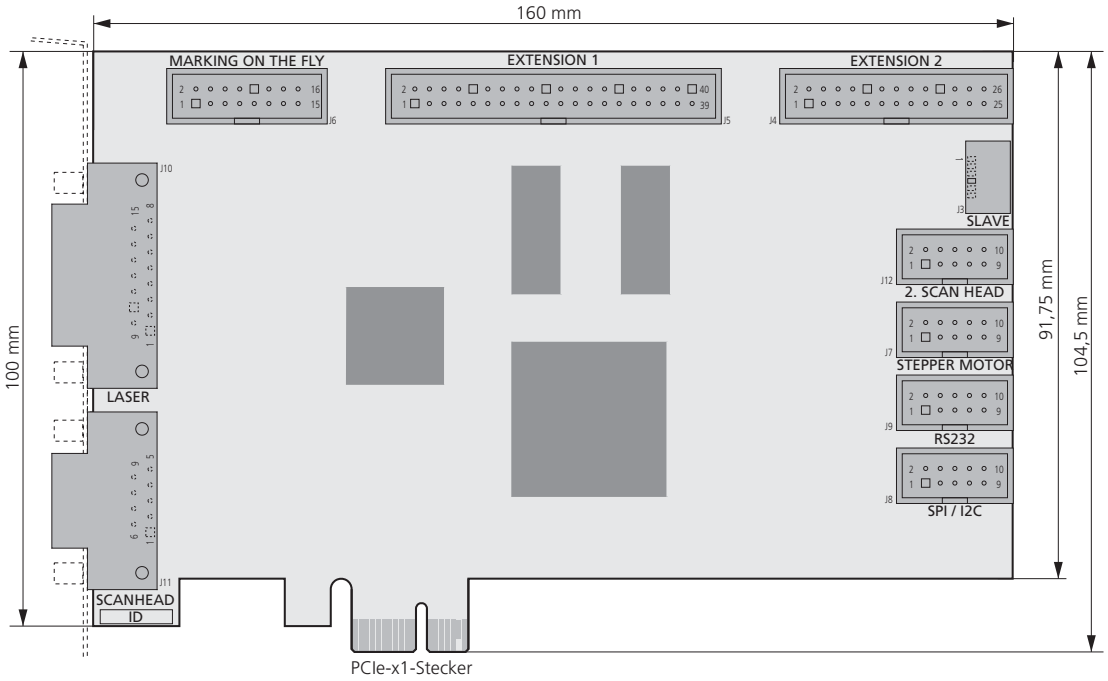
16.1.8 Ergänzende Software

Zur individuellen Anpassung von RTC-Korrekturdateien entsprechend der Daten einer Testmessung ist bei SCANLAB Spezialsoftware mit zugehörigem Handbuch aus der correXion-Reihe erhältlich (siehe auch Seite 137).

16.2 Layout und Schnittstellen

16.2.1 Stecker- und Jumperpositionen

Abb. 71 und Abb. 72 zeigen die Positionen der Anschlüsse und Jumper auf der Vorderseite und Rückseite der RTC5 PCI-Express-Karte. Die RTC5 PCI-Express-Karte unterscheidet sich bzgl. der Anschlüsse und Jumpereinstellungen nur in der Schnittstelle zum PC und in der Konfiguration der "SPI / I2C" Stiftleiste von der RTC5 PCI-Karte. Diese beiden Abweichungen von der RTC5 PCI-Karte werden in den folgenden Abschnitten beschrieben.

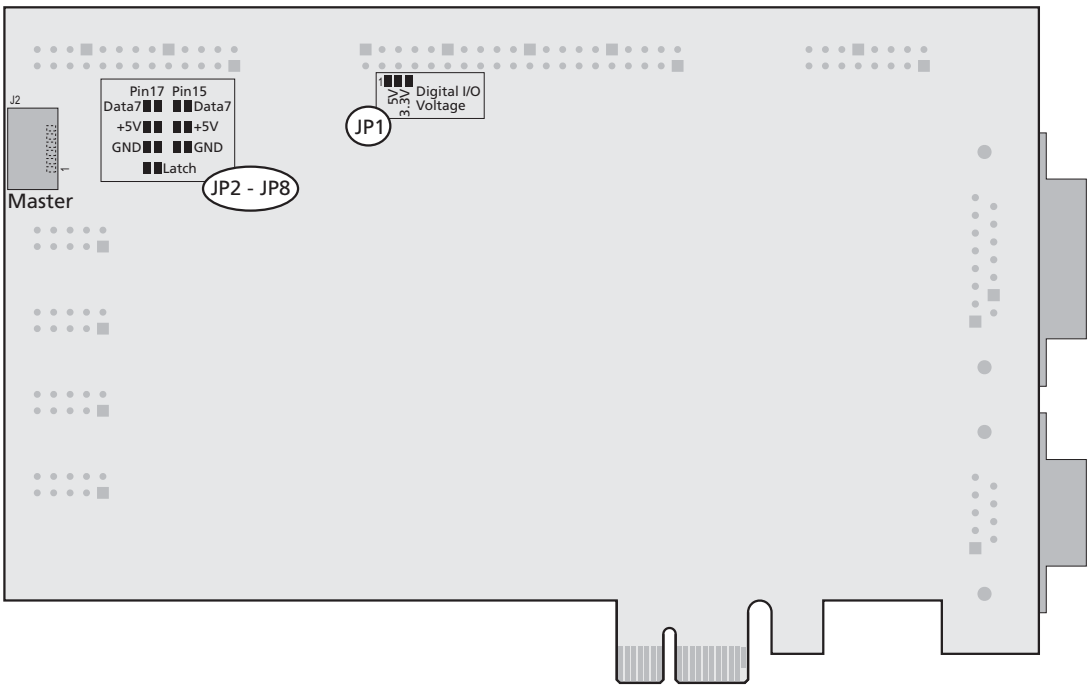


Legende

SCAN HEAD . . . . . Primärer Scan-Kopf-Anschluss (9-polige Sub-D-Buchse, Pinbelegung Seite 42)
LASER . . . . . Laseranschluss, sowie digitale und analoge Ausgänge (15-polige Sub-D-Buchse, Pinbelegung Seite 47)
MARKING ON THE FLY . . . . . 16-polige Stiftleiste mit Encoder-Eingängen, (optional) für "Processing on the fly"-Anwendungen (Pinbelegung Seite 53)
EXTENSION 1 . . . . . 40-polige Stiftleiste mit einem 16 Bit-Digital-Eingang und einem 16 Bit-Digital-Ausgang (Pinbelegung Seite 51)
EXTENSION 2 . . . . . 26-polige Stiftleiste mit einem 8-Bit-Digital-Ausgang, z. B. für eine alternative Laseransteuerung (Pinbelegung Seite 52)
SLAVE . . . . . 6-poliger SLAVE-Anschluss (siehe "Master/Slave-Synchronisation", Seite 41)
2. SCAN HEAD . . . . . 10-polige Stiftleiste für einen zweiten Scan-Kopf (optional aktiviert, Pinbelegung Seite 43)
STEPPER MOTOR . . . . . 10-polige Stiftleiste zur Ansteuerung zweier Schrittmotoren (Pinbelegung Seite 58)
RS232 . . . . . 10-polige Stiftleiste zur Ansteuerung eines externen Geräts per RS232-Schnittstelle (Pinbelegung Seite 54)
SPI / I2C . . . . . 10-polige Stiftleiste (serielle Erweiterungsschnittstelle, Pinbelegung Seite 676)
ID . . . . . Artikelnummer
PCIe-x1-Stecker . . . . . Schnittstelle zum PC (siehe Seite 675)

71

Layout der RTC5 PCI-Express-Karte (Vorderseite)



Legende

Master . . . . . 6-poliger MASTER-Anschluss (siehe "Master/Slave-Synchronisation", Seite 41)

JP1 . . . . . Jumper zur Auswahl des Ausgangssignal-Pegels an der EXTENSION 1-Stiftleiste (siehe Seite 51)

JP2 - JP8 . . . . Jumper zur Konfiguration der EXTENSION 2-Stiftleiste (siehe Seite 52)

72

Layout der RTC5 PCI-Express-Karte (Rückseite)

16.2.2 Schnittstelle zum PC

Schnittstelle zum PC ist der PCIe-x1-Stecker der RTC5 PCI-Express-Karte.

Die RTC5 PCI-Express-Karte kann in jeden Windows-PC mit PCI-Express-Bus und mindestens einem freien PCI-Express-Steckplatz installiert werden.

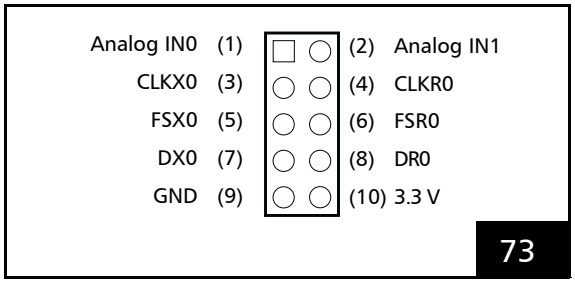
Vorsicht!

- Die RTC5 PCI-Express-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Express-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Siehe hierzu auch Hinweis auf Seite 26.

16.2.3 McBSP/SPI-Schnittstelle und Analog-Eingänge

Die “SPI / I2C” Stiftleiste stellt eine McBSP/SPI-Schnittstelle (Multi channel Buffered Serial Port) und zwei Analog-Eingänge bereit.

Die Pinbelegung ist in Abb. 73 dargestellt.



Pinbelegung der Stiftleiste “SPI / I2C” (on-board)

McBSP/SPI-Schnittstelle

Die McBSP/SPI-Schnittstelle und die zugehörigen Befehle sind identisch zur RTC5 PCI-Karte (siehe Seite 54).

Analog-Eingänge

Anders als die RTC5 PCI-Karte stellt die RTC5 PCI-Express-Karte an der “SPI / I2C” Stiftleiste statt einer I^2C-Schnittstelle (direkt, ohne zusätzliche ADC-Aufsteckplatine) zwei 10 V-Analog-Eingänge (ANALOG IN0 und ANALOG IN1) bereit.

Zur Verwendung und zu den Spezifikationen der beiden Analog-Eingänge siehe Kapitel 4.4.8 “Analog-Eingänge (Sonderzubehör)”, Seite 59.

16.3 Installation und Inbetriebnahme

Die Installation und Inbetriebnahme der RTC5 PCI-Express-Karte und der zugehörigen Software erfolgt gleich wie bei der RTC5 PCI-Karte (siehe Seite 60).

Vorsicht!

- Die RTC5 PCI-Express-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Express-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Deaktivieren Sie insbesondere den Windows-Sleep-Modus (dieser ist bei einigen Windows-Betriebssystemen standardmäßig voreingestellt). Siehe hierzu auch Hinweis auf Seite 26.

16.4 Technische Spezifikationen der RTC5 PCI-Express-Karte

Systemanforderungen

Windows-PC mit freiem PCI-Express-Steckplatz

Betriebssystem	Microsoft Windows 10, 8, 7, Vista, XP SP2, XP SP3 als 32-Bit oder 64-Bit Version

Abmessungen

Länge	160 mm
Breite	104,5 mm

Schnittstellen zum PC

PCI-Express x1, Version 1.0

Stiftleiste "SPI / I2C"

Anschluss	10-polige Stiftleiste
McBSP/SPI-Schnittstelle:	wie RTC5 PCI-Karte (siehe Seite 54)

2 Analogeingänge (ANALOG IN0, ANALOG IN1):

• Eingangsspannungs-bereich	0 V … 10 V
• Eingangsimpedanz	4 kΩ
• ADC-Auflösung	12 Bit
• Bezugspunkt	GND

Sonstige Anschlüsse und Spezifikationen

wie RTC5 PCI-Karte

16.5 Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV)

Die Konformität der RTC5 PCI-Express-Karte mit der EG-Richtlinie 2004/108/EG (Elektromagnetische Verträglichkeit) wurde festgestellt.

Dazu wurde die RTC5 PCI-Express-Karte in einen PC eingebaut und zusammen mit einem vario*SCAN*$_{de}$ 40i Flex Scan-Kopf (SL2-100-Schnittstelle) getestet.

Prüfbestimmungen

Nachweis zur Erfüllung der Schutzziele der EG-Richtlinie 2004/108/EG (CE-Konformität für EMV) auf der Basis von

– EN 61000-6-2: 2005
– EN 61000-6-4: 2007 + A1:2011

Ergebnis

Die Prüflinge erfüllen die Spezifikationen.

16.6 Konformität mit FCC-Bestimmungen

Die RTC5 PCI-Express-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 RTC5 PCI-Express-Karte in einer kommerziellen Umgebung betrieben wird. Die RTC5 PCI-Express-Karte erzeugt, verwendet und emittiert möglicherweise Hochfrequenzenergie und kann, sofern nicht in Übereinstimmung mit dieser Bedienungsanleitung installiert und verwendet, Störungen im Funkverkehr verursachen. Der Betrieb der RTC5 PCI-Express-Karte in einem Wohngebiet wird wahrscheinlich schädliche Störungen verursachen; in diesem Fall ist der Benutzer verpflichtet, die Störungen auf eigene Kosten zu beseitigen.

17 Anhang C: Die RTC5 PCIe/104-Karte

17.1 Produktübersicht

17.1.1 Verwendung – Vergleich mit der RTC5 PCI-Karte

Die SCANLAB RTC5 PCIe/104-Karte ist wie die RTC5 PCI-Karte zur synchronen Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten vorgesehen. Hardware-bedingt unterscheidet sie sich in den folgenden Punkten von der RTC5 PCI-Karte:

- Die RTC5 PCIe/104-Karte ist ein PCIe/104-Modul. Sie wird daher nicht – wie die RTC5 PCI-Karte – in einen PC-Steckplatz eingesteckt, sondern über ihre PCI-Express-Steckverbinder (direkt oder indirekt) mit einer PCIe/104-CPU-Karte verbunden. Es können maximal vier RTC5 PCIe/104-Karten an einer CPU-Karte betrieben werden.
- Anzahl, Art und Anordnung der Anschlussstecker zur Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten sind anders als auf der RTC5 PCI-Karte.

Ansonsten bietet die RTC5 PCIe/104-Karte die gleiche Funktionalität zur Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten wie die RTC5 PCI-Karte. Es gelten die gleichen Spezifikationen für die Signal-Eingänge und -Ausgänge wie bei der RTC5 PCI-Karte. Die Software-Installation und die Programmierung von Anwenderprogrammen erfolgt identisch zur RTC5 PCI-Karte.

RTC5 PCIe/104-Karte und RTC5 PCI-Karte nutzen das gleiche Softwarepaket (Treiber, DLL, Importdeklarationen, etc.) und den gleichen Befehlssatz.

17.1.2 Vergleich mit der RTC5 PC/104-*Plus*-Karte

Die RTC5 PCIe/104-Karte ist mit den gleichen Anschlüssen zur Ansteuerung von Scan-Systemen, Lasern und Peripheriegeräten wie die RTC5 PC/104-*Plus*-Karte ausgestattet. Hardware-bedingt unterscheidet sie sich aber in den folgenden Punkten von der RTC5 PC/104-*Plus*-Karte:

- Zur Verbindung zur CPU-Karte bzw. im Stapel ist die RTC5 PCIe/104-Karte statt mit PCI- und ISA-Steckverbindern (nur) mit PCI-Express-Steckverbindern ausgestattet.
- Die RTC5 PCIe/104-Karte ist *nicht* mit zusätzlichen Jumpern (zur Anpassung der Karte bzgl. der Position in einem Stapel und der Spannungsversorgung) ausgestattet.
  - Die Stapeladresse wird bei PCIe/104-Modulen automatisch eingestellt.
  - Die RTC5 PCIe/104-Karte hat keine ISA-Schnittstelle und keine zugehörigen Anpassungsmöglichkeiten der Spannungsversorgung.
  - Die maximale Ausgangsspannung von 10 V an den analogen Ausgängen ANALOG OUT1 und ANALOG OUT2 am LASER- und MULTI-Stecker wird immer aus der +5 V Versorgungsspannung am PCI-Express-Bus erzeugt.
- Optional kann die RTC5 PCIe/104-Karte mit einem zusätzlichen MARKING ON THE FLY-Anschluss ausgestattet werden (in der Standard-Version nicht enthalten).

17.1.3 Systemanforderungen

Hardware

Die RTC5 PCIe/104-Karte ist ein PCIe/104-Modul mit entsprechendem Formfaktor. Die Abmessungen der RTC5 PCIe/104-Karte sind in Abb. 74 aufgeführt (beachten Sie, dass der 100-polige MULTI-Stecker etwas über die Kante der Platine der RTC5 PCIe/104-Karte hinausragt). Die RTC5 PCIe/104-Karte ist mit PCI-Express-Steckverbindern ausgestattet und kann direkt oder indirekt (in einem Stapel aus mehreren PCIe/104-Modulen) mit einer PCIe/104-CPU-Karte verbunden werden.

Der Datentransfer erfolgt über den PCI-Express-Bus der Karte konform mit der PCI-Express-Basis-Spezifikation Version 2.0 und der PCI-Local-Bus-Spezifikation Version 2.3. Es können Stapel von maximal vier PCIe/104-Modulen (zusätzlich zur PCIe/104-CPU-Karte) gebildet werden.

RTC5 PCIe/104-Karten, die Master/Slave-synchronisiert werden sollen, müssen im Stapel unmittelbar aufeinander folgen.

Zur Spannungsversorgung der RTC5 PCIe/104-Karte muss die CPU-Karte mindestens eine Betriebsspannung von +5 V am PCI-Express-Bus bereitstellen.

Software

Für die RTC5 PCIe/104-Karte können die gleichen Treiber und DLL-Dateien wie für die RTC5 PCI-Karte verwendet werden. Zusätzlich zu den in Kapitel 2.2.2 "Software", Seite 26 für die RTC5 PCI-Karte aufgeführten Betriebssystemen unterstützen die RTC5-Treiber und RTC5-DLL-Dateien auch das Microsoft-Betriebssystem Windows XP Embedded (ab RTC5-Software-Paketen Version 2011_09_29 und später bzw. DLL-Versionen ≥ DLL 528: die 32-Bit- und 64-Bit-Versionen von Windows XP Embedded ≥ SP2). Es können maximal vier RTC5 PCIe/104-Karten an einer CPU-Karte betrieben werden.

Vorsicht!

- Die RTC5 PCIe/104-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Express-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Siehe hierzu auch Hinweis auf Seite 26.

17.1.4 Optionale Funktionalitäten

Die RTC5 PCIe/104-Karte kann mit den gleichen Funktionalitäten ausgestattet werden wie die RTC5 PCI-Karte (siehe Seite 27). Optionale Funktionalitäten müssen von SCANLAB freigeschaltet bzw. installiert werden.

17.1.5 Optionaler Anschluss

Optional kann die RTC5 PCIe/104-Karte mit einem zusätzlichen MARKING ON THE FLY-Anschluss ausgestattet werden (in der Standard-Version nicht enthalten).

17.1.6 Kennzeichnung

Die Seriennummer der RTC5 PCIe/104-Karte finden Sie auf einem weißem Aufkleber auf der Karte (Beschriftung "RTC SN…").

Die Artikelnummer und die Konfiguration der Karte werden auf dem jeweiligen Lieferschein aufgeführt (siehe auch "Optionale Funktionalitäten", Seite 27 und "Typenbezeichnung", Seite 680).

17.1.7 Typenbezeichnung

Aus der Artikelnummer einer RTC5 PCIe/104-Karte ergibt sich, in welcher Jumperkonfiguration (**JP1**-**JP8**) die Karte ausgeliefert wird. Die jeweilige Jumperkonfiguration wird zusätzlich durch eine dreiziffrige Typenbezeichung dargestellt:

Ziffer 1	=0: Jumper **JP1** offen (kein Signal) =1: Jumper **JP1** in Position 1-2 (5 V) =2: Jumper **JP1** in Position 2-3 (3,3 V)
Ziffer 2 (betr. Pin B40)	=0: Jumper **JP2**-**JP4** offen (kein Signal) =1: Jumper **JP2** geschlossen (+5 V) =2: Jumper **JP3** geschlossen (DATA7) =3: Jumper **JP4** geschlossen (GND)
Ziffer 3 (betr. Pin B42)	=0: Jumper **JP5**-**JP8** offen (kein Signal) =1: Jumper **JP5** geschlossen (+5 V) =2: Jumper **JP6** geschlossen (DATA7) =3: Jumper **JP7** geschlossen (GND) =4: Jumper **JP8** geschlossen (LATCH)

Beispiele:

- "Typ 000": Jumper **JP1**-**JP8** offen
- "Typ 124": **JP1** in Position1-2 (5 V Signal-Pegel), **JP3** geschlossen (DATA7 an Pin B40), **JP8** geschlossen (LATCH an Pin B42)

17.1.8 Auspackanleitung und typischer Lieferumfang

- Nehmen Sie die RTC5 PCIe/104-Karte vorsichtig aus der Verpackung.
- Bewahren Sie sowohl die Umverpackung als auch den antistatischen Beutel, in dem die RTC5 PCIe/104-Karte ausgeliefert wird, auf, damit die Karte im Fall einer Reparatur wieder optimal verpackt an SCANLAB geschickt werden kann.
- 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 RTC5 PCIe/104-Karte und einer Daten-CD (mit der zugehörigen Software (s. u.) und diesem Manual). Unter Umständen sind aber auch Zusatzkomponenten wie Datenkabel oder Konverter enthalten (siehe "Optionale Zusatzkomponenten", Seite 680).

Mitgelieferte Software

Das mitgelieferte Software-Paket ist identisch zum RTC5 Software-Paket (siehe Seite 22). Es enthält u. a. Treiber für die 32 Bit und 64 Bit-Versionen von Microsoft Windows 10, 8, 7, Vista, XP SP2, XP SP3, Windows XP Embedded ≥ SP2, sowie Korrektur-, DLL-Programm-, und Utility-Dateien für C, C++ und C#.

17.1.9 Optionale Zusatzkomponenten

Zusätzlich zur RTC5 PCIe/104-Karte und zum zugehörigen Software-Paket sind bei SCANLAB – wie für die RTC5 PCI-Karte – die folgenden Komponenten erhältlich (in Kombination mit der RTC5 PCIe/104-Karte sollten nur Hardwareerweiterungen von SCANLAB verwendet werden):

- XY2-100-Konverter (siehe Seite 29).
- Datenkabel (siehe Seite 29).
- Laseradapter (siehe Seite 29).
- Slotblech mit einer 15-poligen Sub-D-Buchse zur Verwendung der Eingänge und Signale am LASER-Stecker (siehe Seite 661)

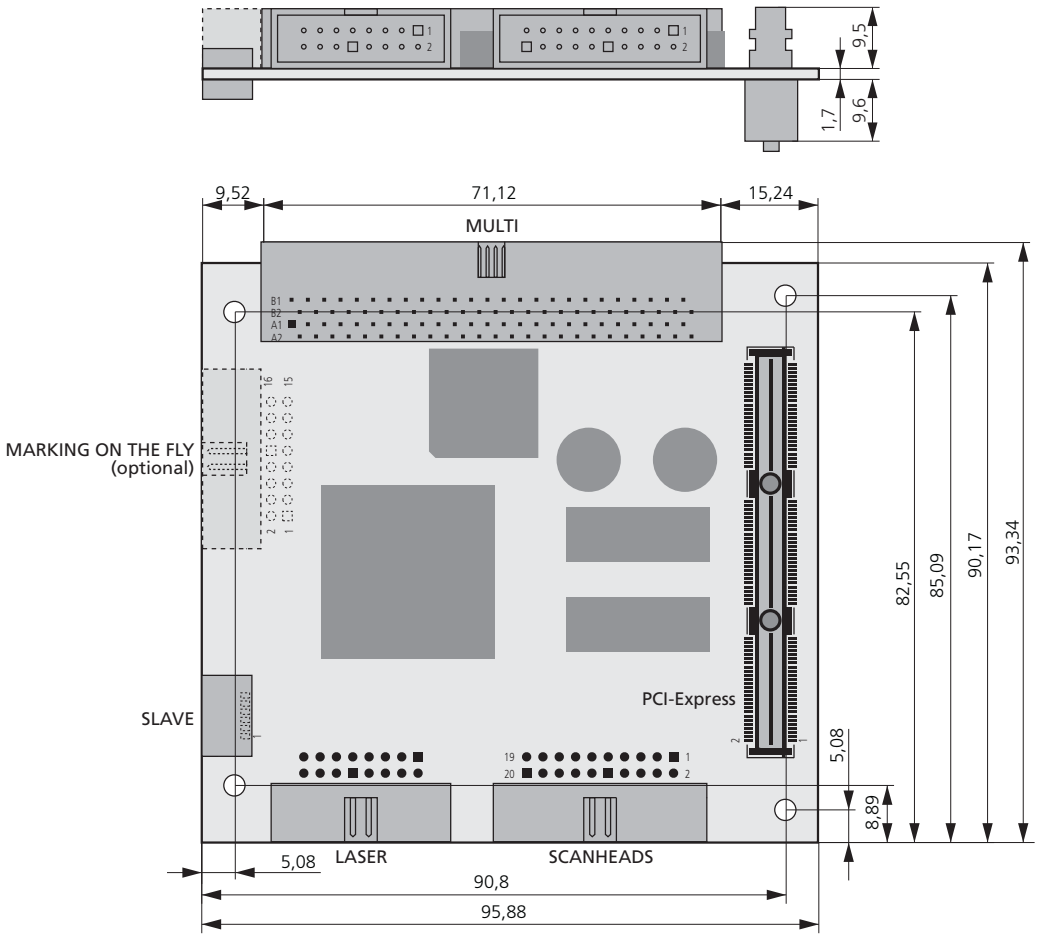
17.1.10 Ergänzende Software

Zur individuellen Anpassung von RTC-Korrekturdateien entsprechend der Daten einer Testmessung ist bei SCANLAB Spezialsoftware mit zugehörigem Handbuch aus der correXion-Reihe erhältlich (siehe auch Seite 137).

17.2 Layout und Schnittstellen

17.2.1 Stecker- und Jumperpositionen

Abb. 74 und Abb. 75 zeigen die Positionen der Anschlüsse und Jumper auf der Vorderseite und Rückseite der RTC5 PCIe/104-Karte. Alle Anschlüsse und Jumpereinstellungen werden in den folgenden Abschnitten beschrieben.



Legende

SCANHEADS. . . . . primärer und sekundärer Scan-Kopf-Anschluss (20-poliger Wannenstecker, 2-reihig, 2.54 mm Pitch, Pinbelegung Seite 660); der sekundäre Scan-Kopf-Anschluss ist nur optional aktiviert

LASER . . . . . Laseranschluss, sowie digitale und analoge Ausgänge (16-poliger Wannenstecker, 2-reihig, 2.54 mm Pitch, Pinbelegung Seite 661)

MULTI . . . . . Multifunktionsstecker mit einem 16 Bit-Digital-Eingang, einem 16 Bit-Digital-Ausgang, einem 8-Bit-Digital-Ausgang, Encoder-Eingängen für "Processing on the fly"-Anwendungen, etc. (100-poliger Wannenstecker, 2-reihig, 1.27 mm Pitch, Pinbelegung Seite 661)

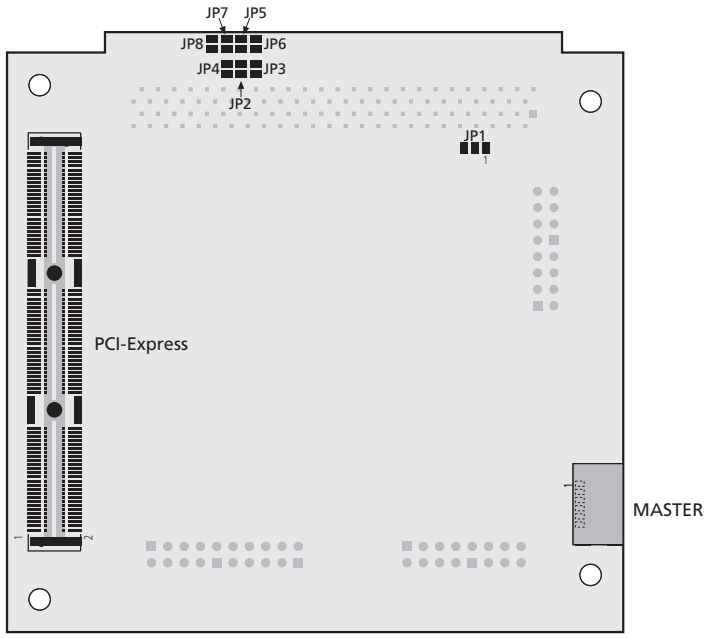
PCI-Express . . . . . PCI-Express-Steckverbinder (156-polig, siehe Seite 660)

SLAVE . . . . . 6-poliger SLAVE-Anschluss (siehe "Master/Slave-Synchronisation", Seite 660)

MARKING ON THE FLY . . . . optionaler 16-poliger Wannenstecker (2-reihig, 2.54 mm Pitch) mit Encoder-Eingängen für "Processing on the fly"-Anwendungen (Pinbelegung Seite 53)

74

Layout und Abmessungen der RTC5 PCIe/104-Karte (oben: Seitenansicht; unten: Vorderseite)



Legende

Master . . . . . 6-poliger MASTER-Anschluss (siehe "Master/Slave-Synchronisation", Seite 682)
JP1 . . . . . Jumper zur Auswahl des Ausgangssignal-Pegels am MULTI-Stecker (siehe Seite 667)
JP2 - JP8 . . . . Jumper zur Konfiguration von Pin B40 und B42 des MULTI-Steckers (siehe Seite 667)

75

Layout der RTC5 PCIe/104-Karte (Rückseite)

17.2.2 Schnittstelle zur CPU-Karte

Der Datentransfer zwischen CPU-Karte und RTC5 PCIe/104-Karte erfolgt über den PCI-Express-Bus. Die RTC5 PCIe/104-Karte ist mit 156-poligen PCIe/104-Steckverbindern ausgestattet (siehe Abb. 74). Das Timing und die Signal-Pegel der PCI-Signale sind konform mit der PCI-Express-Basis-Spezifikation Version 2.0 und der PCI-Local-Bus-Spezifikation Version 2.3.

Vorsicht!

- Die RTC5 PCIe/104-Karte unterstützt keine Energiesparmodi, in denen die Versorgungsspannungen des PCI-Bus abgeschaltet werden. Entsprechende Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren. Siehe hierzu auch Hinweis auf Seite 26.

17.2.3 Master/Slave-Synchronisation

Sollen mehrere RTC5 PCIe/104-Karten mit synchroner Taktung in einem PCIe/104-Stack betrieben werden, dann müssen die RTC5 PCIe/104-Karten paarweise über die MASTER- und SLAVE-Anschlüsse miteinander verbunden und in benachbarte Stapelpositionen gesteckt werden. Verbinden Sie jeweils den MASTER-Anschluss einer Karte mit dem SLAVE-Anschluss einer anderen Karte. Geeignete Verbindungskabel sind bei SCANLAB erhältlich.

Siehe auch Kapitel 6.6.3 "Master/Slave-Betrieb", Seite 93 und "Ausführen und Beenden von Listen mit externen Steuersignalen und Master/Slave-Synchronisation", Seite 239.

17.2.4 Schnittstellen zum Scan-System

Zur digitalen Ansteuerung von Scan-Systemen stehen am 20-poligen SCANHEADS-Stecker der primäre Scan-Kopf-Anschluss und der optional aktivierte sekundäre Scan-Kopf-Anschluss zur Verfügung. Die Pinbelegung und Signale sind identisch zum SCANHEADS-Steckers der RTC5 PC/104-*Plus*-Karte (siehe Seite 660).

17.2.5 Schnittstellen zum Laser und zur Peripherie

Als Schnittstelle zum Laser und zur Peripherie dienen der 16-polige LASER-Stecker und der 100-polige MULTI-Stecker der RTC5 PCIe/104-Karte. Deren Pinbelegungen und Signale sind identisch zu den jeweiligen Anschlüssen der RTC5 PC/104-*Plus*-Karte (siehe Seite 661).

Die Pinbelegung und Signale des optionalen MARKING ON THE FLY-Anschlusses sind identisch zum MARKING ON THE FLY-Anschluss der RTC5 PCI-Karte (siehe Seite 53).

17.3 Installation und Inbetriebnahme

Die Installation der RTC5 PCIe/104-Karte erfolgt in den folgenden Schritten:

(1) Konfiguration der Jumper

Die Konfiguration der Jumper **JP1** bzw. **JP2**-**JP8** erfolgt identisch wie bei der RTC5 PC/104-*Plus*-Karte (siehe Kapitel 15.3.2, Seite 665).

(2) Hardware-Installation
Der Einbau der Karte in den PCIe/104-Stapel erfolgt über die PCI-Express-Steckverbinder. Zusätzlich empfiehlt SCANLAB, die Karte mit Hilfe von Schrauben und Abstandsbolzen mit der CPU-Karte bzw. mit benachbarten Modulen im Stapel zu verbinden (Schrauben und Abstandsbolzen sind nicht im Lieferumfang enthalten). Eine elektrische Anpassung an die Position der Karte im Stapel ist nicht nötig. Ansonsten erfolgt die Hardware-Installation wie bei der RTC5 PC/104-*Plus*-Karte (siehe Kapitel 15.3.3, Seite 668; beachten Sie insbesondere auch alle Sicherheitshinweise in diesem Kapitel und im Kapitel 15.3.1, Seite 664).

(3) Installation der Treiber und RTC5-Software (siehe Kapitel 15.3.4, Seite 670)

Gehen Sie in der vorgeschlagenen Reihenfolge vor.

Für einen Funktionstest nach der Installation kann z. B. das mitgelieferte HPGL-Converter-Programm verwendet werden (siehe Kapitel 15.3.5, Seite 670).

17.4 Technische Spezifikationen der RTC5 PCIe/104-Karte

Systemanforderungen

PCIe/104-CPU-Karte
+ freie Position im PCIe/104-Stack

Betriebssystem	Microsoft Windows 10, 8, 7, Vista, XP SP2, XP SP3, Windows XP Embedded ≥ SP2 als 32-Bit oder 64-Bit Version

Abmessungen

Länge x Breite	96 mm x 93,5 mm
Stapelhöhe	ca. 16 mm
max. Bauteilhöhe	9,3 mm

Gewicht ca. 85 g

Betriebs- und Lagerbedingungen

zulässige Betriebstemperatur	+15 °C bis +60 °C
Lagertemperatur	−20 °C bis +60 °C
Umgebungsbedingungen	nicht-kondensierend nicht-korrodierend

Anschlüsse zur PCIe/104-CPU-Karte (bzw. zum PCIe/104-Stack)

PCI-Express-Bus	156-polige PCIe/104-Standard-Top- und Bottom-Steckverbinder (PCIe x 1, nur eine Lane verwendet), Betriebsspannung: +5 V

Sonstige Anschlüsse (Signal-Spezifikationen wie bei der RTC5 PCI-Karte)

Schnittstelle zum Scan-System	SCANHEADS-Stecker (20-poliger Wannenstecker) mit primärem und sekundärem Scan-Kopf-Anschluss (der sekundäre Scan-Kopf-Anschluss ist nur optional aktiviert)
Schnittstelle zum Laser und zur Peripherie	LASER-Stecker (16-poliger Wannenstecker), MULTI-Multifunktionsstecker (100-poliger Wannenstecker), optional: MARKING ON THE FLY-Anschluss (16-poliger Wannenstecker)

Hinweis zu ANALOG OUT1 und ANALOG OUT2 (gilt nur für die RTC5 PCIe/104-Karte): Abhängig von der angelegten Last geben die beiden analogen Ausgänge eine Mindestspannung aus, auch wenn in Befehlen wie **write_da_x** Ausgangsspannungen unterhalb dieser Mindestspannung festgelegt werden. Sofern die angelegte Last nicht die spezifizierte maximale Last überschreitet, liegt die Mindestspannung im Bereich 0 V…0,1 V.

18 Änderungsindex

Weder Korrekturen von Rechtschreib- oder Grammatikfehlern, noch sprachliche Umformulierungen oder Ergänzungen (z. B. neue Querverweise), die den technischen Inhalt unverändert lassen, sind nachfolgend explizit aufgeführt. Die angegebenen Seitenzahlen beziehen sich auf das vorliegende Dokument.

Änderungen von Version 0.9 nach Version 0.11

Seite	Name des Kapitels / Befehls	Änderung
21	Kapitel 1.2 "Kennzeichnung"	Artikelnummer steht nicht auf Karte.
45	Kapitel 4.3.3 "Datenkabel (Zubehör)"	Empfohlene Leitungsimpedanz hinzugefügt.
135	Kapitel 7.3.3 "Virtuelles Bildfeld"	Kapitel neu hinzugefügt (Verwendung auch bei Koordinatentransformationen).
226	Kapitel 8.11.3 "Ermittlung von Referenzwerten"	Referenzwerte für automatische Selbstkalibrierung werden in DLL bzw. EEPROM gespeichert.
240	Externer Listen-Start	Hinweis zu Master/Slave-Synchronisierung bei externem Start und aktiviertem Home-Jump.
250	Normale, kurze, variable und multiple Listenbefehle	Hinweise zu kurzen Listenbefehlen.
270	**auto_cal**-Befehl	Referenzwerte für automatische Selbstkalibrierung werden in DLL bzw. EEPROM gespeichert.
369	**list_nop**-Befehl	Neuer Hinweis.
464	**set_auto_laser_control**-Befehl	Korrektur bei Rückgabewert 5.
475	**set_delay_mode**-Befehl	Korrektur bzgl. RTC4-Kompatibilitätsmodus.
480	**set_end_of_list**-Befehl	Korrektur bei Hinweisen.
537	**set_port_default**-Befehl	Korrektur bei Hinweisen.
655	Kapitel 15 "Anhang A: Die RTC5 PC/104-Plus-Karte"	Kapitel neu hinzugefügt.

Änderungen von Version 0.11 nach Version 0.12

Seite	Name des Kapitels / Befehls	Änderung
44	Kapitel 4.3.2 "XY2-100-Konverter (optional)"	Geänderte Angaben zu empfohlenen Jumpereinstellungen und Kabellängen.
54	Kapitel 4.4.6 "SPI / I2C-Stiftleiste"	Kapitel zu McBSP-Schnittstelle neu hinzugefügt.
96	Kapitel 6.7 "Verwendung durch mehrere Anwenderprogramme"	Korrektur: `RTC5_TIMEOUT`-Fehler bei Versionsprüfung eliminiert.
99	Kapitel 6.8.1 "Download-Überprüfung"	Ergänzung: erweiterte Download-Überprüfung.
168	Encodergeschwindigkeitsabhängige Lasersteuerung	Abschnitt neu hinzugefügt.
199	Kapitel 8.7 ""Processing on the fly" (optional)"	Kapitel überarbeitet und ergänzt ("Processing on the fly"-Korrektur auch mit direkten Positionssignalen über die McBSP-Schnittstelle).
217	Kapitel 8.8 "Pixelausgabemodus – Rasterbilder (Bitmaps) scannen"	Kapitel ergänzt – neuer Befehl **set_n_pixel**.
224	Kapitel 8.11 "Automatische Selbstkalibrierung"	Korrektur: entsprechend den Änderungen bei den Befehlen **auto_cal** und **get_hi_pos** (s.u.).
237	Kapitel 9.1.7 "McBSP/SPI-Schnittstelle"	Kapitel neu hinzugefügt.
238	Kapitel 9.2.4 "McBSP/SPI Schnittstelle"	Kapitel neu hinzugefügt.
239	Externer Listen-Stopp	Korrektur: Die Eingänge für externe Stoppsignale können nicht verriegelt werden.
240	Externer Listen-Start	Korrektur in Abb. 61.
247	Eingänge für externe Encoder-Signale	maximal zulässige Frequenz für Encoder-Signale korrigiert.
248	Kapitel 9.3.4 "Synchronisation und Online-Positionierung mit McBSP/SPI-Signalen"	Kapitel neu hinzugefügt.
270	**auto_cal**-Befehl	Ergänzung: Voraussetzung für *iDRIVE*-Scan-Systeme Korrektur: Für `Command` = 3 werden die aktuellen Home-In-Positionen ermittelt (nicht Referenzwerte gespeichert).
302	**get_error**-Befehl	Neu: Bit #9 (Verify-Fehler).
309	**get_hi_pos**-Befehl	Korrektur: Zurückgegeben werden die aktuellen Home-In-Positionen (also normalerweise nicht die Referenzwerte).
317	**get_mcbsp**-Befehl	Neu hinzugefügt.
375	**load_correction_file**-Befehl	neue Fehlercodes.
385	**load_list**-Befehl	Korrektur: Änderung für `ListNo` = 3.
389	**load_program_file**-Befehl	neuer Fehlercode.
413	**mark_time**-Befehl	Korrektur bei Parameter `Mode` (Bit#0 = 1).
415	**mcbsp_init**-Befehl	Neu hinzugefügt.
464	**set_auto_laser_control**-Befehl	Ergänzung: encodergeschwindigkeitsabhängige Lasersteuerung über `Mode` = 5.

Seite	Name des Kapitels / Befehls	Änderung
470	**set_control_mode**-Befehl	Korrektur: Die Eingänge für externe Stoppsignale können nicht verriegelt werden.
478	**set_encoder_speed**-Befehl	Neu hinzugefügt.
479	**set_encoder_speed_ctrl**-Befehl	Neu hinzugefügt.
488	**set_fly_rot_pos**-Befehl	Neu hinzugefügt.
491	**set_fly_x_pos**-Befehl	Neu hinzugefügt.
493	**set_fly_y_pos**-Befehl	Neu hinzugefügt.
510 511 512 557 558	**set_laser_pulses**-Befehl **set_laser_pulses_ctrl**-Befehl **set_laser_timing**-Befehl **set_standby**-Befehl **set_standby_list**-Befehl	Korrektur: erlaubter Bereich für Parameter `HalfPeriod` vergrößert.
521	**set_mcbsp_out**-Befehl	Neu hinzugefügt.
529	**set_n_pixel**-Befehl	Neu hinzugefügt.
571	**set_verify**-Befehl	Ergänzung: erweiterte Download-Überprüfung.
631	**verify_checksum**-Befehl	Neu hinzugefügt.
633	**wait_for_encoder_mode**-Befehl	Neu hinzugefügt.
634	**wait_for_mcbsp**-Befehl	Neu hinzugefügt.
665	Jumper 17+18: Binäre Festlegung der Stapeladresse	Korrektur in Tabelle: Jumperzustände invertiert.
666	JP10+12: 10 V Ausgangsspannung für ANALOG OUT1 und ANALOG OUT2	Ergänzung bzgl. max. Ausgangsstrom.
685	Kapitel 18 "Änderungsindex"	Neu hinzugefügt.

Änderungen von Version 0.12 nach Version 1.0

Seite	Name des Kapitels / Befehls	Änderung
	Allgemein	Das Handbuch gilt nicht mehr als “vorläufig”.
23	Änderungshistorie	Neu hinzugefügt.
45	Kapitel 4.3.3 “Datenkabel (Zubehör)“	Korrektur: Das SL2-100-Datenkabel benötigt keinen Ferrit.
63	Kapitel 5.5 “Sichere Einschalt- und Ausschaltreihenfolge“	Neu hinzugefügt.
71	Kapitel 6.2.5 “Beispiel-Code“	Neu hinzugefügt.
93	Kapitel 6.6.3 “Master/Slave-Betrieb“	Neu hinzugefügt.
100	Kapitel 6.8.3 “Beispiel-Code“	Neu hinzugefügt.
103	Kapitel 7.1 “Markieren von Punkten, Linien und Bögen“	Neu hinzugefügt.
109	Kapitel 7.1.4 “Beispiel-Code“	Neu hinzugefügt.
122	Anpassung der Scanner-Delays	Ergänzte automatische Anpassung.
142	Takt-Überlauf	Neu hinzugefügt
144	Kapitel 7.4 “Lasersteuerung“	Überarbeitet: Einführung der Begriffe “Laser active“ und “Laser standby” zur Klarstellung (dies sind *keine* neue Funktionen). Die Begriffe werden jetzt auch an anderen Stellen im Handbuch verwendet. Korrigiert: Standby-Signale in YAG-Versionen. Neu: Kapitel 7.4.7 “Lasermode 6“, Seite 155.
242	Regelmäßige (periodische) externe Listen-Starts	Neu hinzugefügt.
253	Kapitel 10.1.3 ”Versionsinformationen”	Neu hinzugefügt.
253	Kapitel 10.1.4 “Optionale Funktionen“	Neu hinzugefügt.
270	**auto_cal**-Befehl	Korrektur bei Rückgabewert: Achse 1/2 -> Achse X/Y.
283	**control_command**-Befehl	Beschreibung zur Klarstellung überarbeitet. Korrektur: Der Kalibrierwinkel (`Data` = 0523_H) wird in Milligrad (nicht in mrad) zurückgegeben. Korrektur: Bit-Zuweisung bei `Data` = 0526_H, 0527_H, 0529_H und $053F_H$. Korrektur: Das Rückmeldeverhalten des Scan-Systems kann nur temporär geändert werden (`Data` = $0A00_H$, siehe auch Seite 182). Neu: SetControlDefinitionMode ($Code_H$ = $0E_H$).
307	**get_head_status**-Befehl	Änderung: Reservierte Bits werden jetzt (wie vorher schon bei **get_value** oder **set_trigger**) als 1 zurückgegeben.
309	**get_hi_data**-Befehl	Änderung: Voraussetzung gestrichen.
309	**get_hi_pos**-Befehl	Änderung: Voraussetzung gestrichen.

Seite	Name des Kapitels / Befehls	Änderung
314	**get_marking_info**-Befehl	Ergänzung: Befehl liefert auch Informationen über fehlerhafte Encoder-Signale zurück.
316	**get_master_slave**-Befehl	Neu hinzugefügt.
317	**get_mcbsp**-Befehl	Korrektur: Rückgabewert ist ein 32-Bit-Wert *mit* Vorzeichen.
317	**get_mcbsp_list**-Befehl	Neu hinzugefügt.
318	**get_overrun**-Befehl	Neu hinzugefügt.
320	**get_standby**-Befehl	Neu hinzugefügt.
321	**get_startstop_info**-Befehl	Ergänzung: Ab Firmware-Version 508 wird von der Firmware das Pulslängen-Fehlerbit (Bit #17 bzw. 25) zurückgegeben, wenn aktuell kein Scan-System angeschlossen ist.
332	**get_value**-Befehl	Ergänzung: aktueller Laserstatus auch außerhalb einer Listenausführung rücklesbar.
338	**goto_xy**-Befehl	Neuer Hinweis bzgl. externem Stopp-Signal.
340	**home_position**-Befehl	Korrektur: Ein Home-Jump wird auch ausgeführt, wenn eine Listenausführung mit **stop_execution** oder einem externen Stoppsignal abgebrochen wird.
341	**home_position_xyz**-Befehl	Neu hinzugefügt.
373	**load_auto_laser_control**-Befehl	Änderung: Rückgabewert 11.
379	**load_disk**-Befehl	Ergänzung: Initialisierung mit `Name` = 0.
387	**load_position_control**-Befehl	Änderung: Rückgabewert 11.
395	**load_varpolydelay**-Befehl	Änderung: Rückgabewert 11.
396	**load_z_table**-Befehl	Neu: Fehlercode 64 als Rückgabewert.
402	**mark_date**-Befehl (auch **mark_date_abs**-Befehl)	Ergänzung: Monat und Wochentag können jetzt auch mit normalen Ziffern markiert werden (Parameter Part = 6, 7).
470	**set_control_mode**-Befehl	Ergänzung: Bit#10 für regelmäßige (periodische) externe Listen-Starts.
482	**set_ext_start_delay**-Befehl	Ergänzung: RTC4-Kompatibilitätsmodus für Parameter `Delay`.
483	**set_ext_start_delay_list**-Befehl	Änderung: jetzt ein normaler Listenbefehl Ergänzung: RTC4-Kompatibilitätsmodus für Parameter `Delay`.
487	**set_fly_rot**-Befehl (analog für **set_fly_rot_pos**, **set_fly_x**, **set_fly_y**, **set_fly_x_pos** und **set_fly_y_pos**)	Änderung: deaktiviert für unzulässige Parameterwerte nur eine zuvor mit **set_fly_rot** aktivierte "Processing on the fly"-Korrektur, aber keine anderen "Processing on the fly"-Korrekturen.
508	**set_laser_mode**-Befehl	Neu: Lasermode 6.
531 532	**set_offset_xyz**-Befehl **set_offset_xyz_list**-Befehl	Neu hinzugefügt.
562	**set_trigger**-Befehl	Ergänzung: tatsächlicher Ausgabewert über Parameter `Signal1`/`Signal2` = 20-23.
583	**simulate_ext_start**-Befehl	Änderung: jetzt ein normaler Listenbefehl Ergänzung: RTC4-Kompatibilitätsmodus für Parameter `Delay`.

Seite	Name des Kapitels / Befehls	Änderung
584	**simulate_ext_start_ctrl**-Befehl	Neu hinzugefügt.
596	**stop_execution**-Befehl	Korrektur: Ein Home-Jump wird ausgeführt.
602	**switch_ioport**-Befehl	Neu hinzugefügt.
603	**sync_slaves**-Befehl	Neu hinzugefügt.
631	**verify_checksum**-Befehl	Änderung: Rückgabewert = 0 bedeutet jetzt OK. Rückgabewert > 0 Fehler oder Warnung.
647	Kapitel 11 "Demo-Programme"	Neu hinzugefügt.
648	Kapitel 12 "Fehlersuche"	Neu hinzugefügt.

Änderungen von Version 1.0 nach Version 1.1

Seite	Name des Kapitels / Befehls	Änderung
31	Kapitel 2.7 "Hinweise an Benutzer der RTC4-Karte"	Ergänzt.
64	Kapitel 5.8 "Austauschen von RTC5-Karten"	Neu hinzugefügt.
105	"Ellipse-Befehle"	Neu hinzugefügt.
139	"Inverse Tabellen"	Neu hinzugefügt.
174	Kapitel 8.1.3 "Überwachung der Positionierung"	Neu hinzugefügt.
196	"Überprüfen der Z-Achsen-Kalibrierung"	Überarbeitet.
209	Kapitel 8.7.5 "Deaktivieren der "Processing on the fly"-Korrektur"	Neu hinzugefügt.
251	"Unverzögerte und verzögerte kurze Listenbefehle (ab Version OUT 515)"	Neu hinzugefügt.
267	**arc_abs_3d**-Befehl	Neu hinzugefügt.
269	**arc_rel_3d**-Befehl	Neu hinzugefügt.
302	**get_error**-Befehl	Ergänzung: Bit #10 (DSP-Versions-Fehler).
319	**get_rtc_version**-Befehl	Ergänzung: Bits #16…#23 (DSP-Versionsnummer).
329	**get_transform**-Befehl	Neu hinzugefügt.
332	**get_value**-Befehl	Ergänzung: neue Hinweise, Beispiel-Code.
334	**get_values**-Befehl	Neu hinzugefügt.
355	**laser_on_pulses_list**-Befehl	Neu hinzugefügt.
405	**mark_ellipse_abs**-Befehl	Neu hinzugefügt.
406	**mark_ellipse_rel**-Befehl	Neu hinzugefügt.
467	**set_auto_laser_params**-Befehl	Neu hinzugefügt.
467	**set_auto_laser_params_list**-Befehl	Neu hinzugefügt.
472	**set_default_pixel**-Befehl	Neu hinzugefügt.
472	**set_default_pixel_list**-Befehl	Neu hinzugefügt.
477	**set_ellipse**-Befehl	Neu hinzugefügt.
504	**set_laser_control**-Befehl	Ergänzung: Bit#5 für **laser_on_pulses_list**.
550	**set_sky_writing_para**-Befehl	Neu hinzugefügt.
552	**set_sky_writing_para_list**-Befehl	Neu hinzugefügt.
597	**stop_trigger**-Befehl	Neu hinzugefügt.

Seite	Name des Kapitels / Befehls	Änderung
603	**sync_slaves**-Befehl	Geändert; ab DLL 518, OUT 517, RBF 514 gilt: • Eine Master/Slave-Synchronisierung kann nur mit dem Befehl **sync_slaves** ausgeführt werden. Eine Initialisierung der Karten in geeigneter Reihenfolge ist hierzu nicht ausreichend. • Eine Synchronisierung wird nur für diejenigen nachgeschalteten Slave-Karten durchgeführt, für die die Anwendung eine Zugriffsberechtigung besitzt. • **sync_slaves** führt jetzt automatisch einen **simulate_ext_stop**-Befehl aus.
617 - 626	Befehle **timed_para_jump_abs** … **timed_para_mark_rel_3d**	Acht **timed_para***-Befehle neu hinzugefügt.
627	**transform**-Befehl	Neu hinzugefügt.
630	**upload_transform**-Befehl	Neu hinzugefügt.
641	**write_io_port_mask**-Befehl	Neu hinzugefügt.
641	**write_io_port_mask_list**-Befehl	Neu hinzugefügt.

Änderungen von Version 1.1 nach Version 1.2

Seite	Name des Kapitels / Befehls	Änderung
183	Kapitel 8.2 "Koordinatentransformationen"	Ergänzt: Verhalten für `at_once` > 1 (s.u.).
224	Kapitel 8.11 "Automatische Selbstkalibrierung"	Ergänzt und korrigiert: u.a. Hinweise zu **auto_cal**(`Command` = 4) und zu i*DRIVE*-Systemen.
250	Normale, kurze, variable und multiple Listenbefehle	Änderung (ab Version OUT 518): • Jetzt auch bis zu 12 kurze Listenbefehle innerhalb eines 10 µs-Taktes möglich. • Bei Überschreiten der maximalen Anzahl wird kein **list_nop** mehr, sondern nur ein 10 µs-Takt automatisch eingefügt (der Laser bleibt im Polygonzug an).
251	"Unverzögerte und verzögerte kurze Listenbefehle (ab Version OUT 515)"	Änderung (ab Version OUT 518): Bei mehreren unmittelbar aufeinander folgenden kurzen Listenbefehlen wird ein "verzögerter kurzer Listenbefehl" nur noch dann unverzögert ausgeführt, wenn ein weiterer "verzögerter kurzer Listenbefehl" folgt, dagegen nicht mehr, wenn ein "unverzögerter kurzer Listenbefehl" folgt.
266 …	**arc_abs**-Befehl und alle entsprechenden rel-, 3d-, para- und timed-Befehle	Markierungen der Bogenlänge 0 verändern die Lasersteuersignale nicht mehr (siehe Seite 115).
270	**auto_cal**-Befehl	Geändert; ab DLL 520, OUT 519 gilt: • Mit `Command` = 4 (neu) kann die ASC-Hardware überprüft werden. Die Hardware-Überprüfung findet auch bei `Command` = 0 und u.U. auch bei `Command` = 1 und 3 statt (siehe Befehlsbeschreibung). • Der Befehl unterstützt jetzt auch Sensorsysteme vom Typ2. • Die RTC5 fährt die Galvanometerscanner auch bei `Command` = 0 und `Command` = 2 wieder in die gleiche Stellung wie vor dem Befehlsaufruf (also nicht mehr in Mittelstellung).
283	**control_command**-Befehl	Änderung: Nach einem Umschalten der Datenquelle (mit $Code_H$ = 05_H, $0E_H$ oder 21_H bzw. `Data` = 1700_H) wird automatisch eine Wartezeit von 60 µs eingefügt.
299	**get_auto_cal**-Befehl	Neu hinzugefügt.
338 … 352	**goto_xy**, **goto_xyz**, **jump_abs**, **jump_rel**-Befehle	Verhalten bei Koordinatentransformationen mit `at_once` = 2 (siehe Seite 185).
398 …	**mark_abs**-Befehl und alle entsprechenden rel-, 3d-, para- und timed-Befehle	Markierungen der Länge 0 verändern die Lasersteuersignale nicht mehr (siehe Seite 115).
405, 406	**mark_ellipse_abs**-Befehl, **mark_ellipse_rel**-Befehl	Markierungen der Bogenlänge 0 verändern die Lasersteuersignale nicht mehr (siehe Seite 115).
440	**read_encoder**-Befehl	Neu hinzugefügt.

Seite	Name des Kapitels / Befehls	Änderung
462 … 543	Befehle für Koordinaten-transformationen: **set_angle**, **set_angle_list**, **set_offset**, **set_offset_list**, **set_offset_xyz**, **set_offset_xyz_list**, **set_matrix**, **set_matrix_list**, **set_scale**, **set_scale_list**	Änderung: Verhalten für `at_once` > 1 (Sammeln wie bei `at_once` = 0, und zusammen mit **jump_abs**, **jump_rel**, **goto_xy** oder **goto_xyz** in einem gemeinsamen Sprung ausführen – siehe auch Seite 185).
597	**store_encoder**-Befehl	Neu hinzugefügt.

Änderungen von Version 1.2 nach Version 1.3

Seite	Name des Kapitels / Befehls	Änderung
54	"Verwendung als McBSP-Schnittstelle"	Änderung und Ergänzung (ab Version OUT 526): • Kein Stagnieren der Datenübertragung mehr). • Datenübertragung für die Online-Positionierung.
94	"Anpassung der Kurzbefehlszählung (ab Version DLL 523, OUT 524)"	Neu hinzugefügt.
108	Kapitel 7.1.3 "Markieren von Punkten"	Neu hinzugefügt.
116	Hinweise im Abschnitt "Mark-Delay"	Verändertes Verhalten von zeitbemessenen Mark- oder Bogenbefehlen der räumlichen Länge Null ab RTC5OUT.out Version 526 (siehe auch Hinweis im Kapitel 8.10 "Zeitbemessene Vektor- und Bogenbefehle", Seite 223).
121	Automatische Delay-Anpassung	Änderung (ab Version DLL 522, OUT 522, RBF 517): • in Abschnitt (1): Bei Überkreuzung von LaserOn mit LaserOff wird das LaserOff-Delay zwar weiterhin automatisch verlängert, aber so, dass die Lasersteuersignale nur noch für 0,5 µs (also nicht mehr für 10 µs) anbleiben. • in Abschnitt (2): Wenn ein LaserOff-Delay abläuft und zeitgleich wieder ein LaserOn gesetzt werden soll, wird jetzt kein zusätzliches Scanner-Delay (von 10 µs) mehr eingefügt.
138	Hinweise im Abschnitt Zuweisung von geladenen Korrekturtabellen	Änderung (ab Version DLL 521, OUT 521): **load_correction_file** ruft nach dem Laden einer Korrekturtabelle automatisch **select_cor_table** auf.
140	Datei-Kopf von Korrekturdateien	Änderung: Parameter 15 (**get_head_para**, **get_table_para**) ist 2,0 (statt 1,0) für 'mit Begrenzung der Feldgröße'.
187	Kapitel 8.3 "Online-Positionierung"	Neu hinzugefügt.
248	Kapitel 9.3.4	Ergänzung (ab Version OUT 526): Online-Positionierung.
264, 265	**apply_mcbsp**-Befehl, **apply_mcbsp_list**-Befehl	Neu hinzugefügt.
317, 317	**get_mcbsp**-Befehl und **get_mcbsp_list**-Befehl	Änderung (ab Version OUT 526): siehe Versions-Info in der Befehlsbeschreibung.
375	**load_correction_file**-Befehl	Änderung (ab Version DLL 521, OUT 521): **load_correction_file** ruft nach dem Laden einer Korrekturtabelle automatisch **select_cor_table** auf.
422	**move_to**-Befehl	Neu hinzugefügt.
426	**para_laser_on_pulses_list**-Befehl	Neu hinzugefügt.
443	**read_mcbsp**-Befehl	Neu hinzugefügt.
476	**set_dsp_mode**-Befehl	Neu hinzugefügt.
488… 493	**set_fly_rot_pos**, **set_fly_x_pos** und **set_fly_y_pos**	Ergänzung (ab Version OUT 526): nicht gleichzeitig mit Online-Positionierung verwendbar.

Seite	Name des Kapitels / Befehls	Änderung
523… 525	**set_mcbsp_rot**, **set_mcbsp_rot_list**, **set_mcbsp_x**, **set_mcbsp_x_list**, **set_mcbsp_y**, **set_mcbsp_y_list**	Neu hinzugefügt.
603	**sync_slaves**-Befehl	Aktualisierung mit Version DLL 523, OUT 524: führt gegebenenfalls auch eine Anpassung der Kurzbefehlszählung durch (siehe Seite 94).
605	**time_fix_f_off**-Befehl	Neu hinzugefügt.

Änderungen von Version 1.3 nach Version 1.4

Seite	Name des Kapitels / Befehls	Änderung
108	Kapitel 7.1.3 “Markieren von Punkten”	Ergänzung (ab Version OUT 527): Markieren von Punkten mit Mark- oder Bogenbefehlen der Länge Null.
176	Kapitel 8.1.5 “Sprungmodus”	Neu hinzugefügt.
213	Kapitel 8.7.9 “Überwachung der “Processing on the fly”-Korrektur”	Überarbeitet und ergänzt um Abschnitt “Kundenspezifischer Überwachungsbereich und bedingte Befehlsausführung (ab Version DLL 525, OUT 527)”.
221	Kapitel 8.9 “Mikrovektor-Befehle”	Neu hinzugefügt.
250	“Normale, kurze, variable und multiple Listenbefehle”	Ergänzung (ab Version DLL 525, OUT 527): Verzögerung von kurzen Listenbefehlen durch **list_continue** (alternativ zu **list_nop**).
278	**clear_fly_overflow**-Befehl	Neu hinzugefügt.
311	**get_jump_table**-Befehl	Neu hinzugefügt.
314	**get_marking_info**-Befehl	Änderung (ab Version DLL 525, OUT 527): Bits#4…7.
342 …	**if_fly_x_overflow**-Befehl, **if_fly_y_overflow**-Befehl, **if_not_fly_x_overflow**-Befehl **if_not_fly_y_overflow**-Befehl	Neu hinzugefügt.
346 …	**if_not_pin_cond**-Befehl, **if_pin_cond**-Befehl	Neuer Hinweis zur sicheren Verwendung Seite 244.
363	**list_continue**-Befehl	Neu hinzugefügt.
383	**load_jump_table_offset**-Befehl	Neu hinzugefügt.
398 …	**mark_abs**-Befehl und alle entsprechenden rel-, 3d- und para-Befehle (nicht dagegen bei den entsprechenden timed-Befehlen)	Änderung (ab Version OUT 527): Eine Markierung der Länge 0 lässt die Lasersteuersignale nur dann unverändert, wenn sie innerhalb eines Polygonzugs erfolgt (siehe Seite 115).
417	**measurement_status**-Befehl	Ergänzung: `Pos` = 2^{32}–1.
418…	**micro_vector_abs**-Befehl, **micro_vector_rel**-Befehl	Neu hinzugefügt.
462 … 543	Befehle für Koordinaten-transformationen: **set_angle**, **set_angle_list**, **set_offset**, **set_offset_list**, **set_offset_xyz**, **set_offset_xyz_list**, **set_matrix**, **set_matrix_list**, **set_scale**, **set_scale_list**	Änderung: Verhalten für `at_once` = 3 (hier bleiben die Lasersteuersignale unverändert).
485	**set_fly_limits**-Befehl	Neu hinzugefügt.
498	**set_jump_mode**-Befehl	Neu hinzugefügt.
501	**set_jump_mode_list**-Befehl	Neu hinzugefügt.
503	**set_jump_table**-Befehl	Neu hinzugefügt.
569	**set_vector_control**-Befehl	Änderung (ab Version DLL 525, OUT 527): Variieren der Fokusverschiebung mit `Ctrl` = 7.



Änderungen von Version 1.4 nach Version 1.5

Seite	Name des Kapitels / Befehls	Änderung
26 und 61	Kapitel 2.2.2 “Software“ und Kapitel 5.3 “Installation der Treiber“	Warnhinweis hinzugefügt: Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren.
58	Kapitel 4.4.7 “STEPPER MOTOR-Stiftleiste (Schrittmotoransteuerung)“	Ergänzt.
93	Kapitel 6.6.3 “Master/Slave-Betrieb“	Korrigiert: Master-Karte ist diejenige Karte, an deren SLAVE-Anschluss (!) keine weitere Karte angeschlossen ist.
127	Kapitel 7.2.4 “Sky-Writing“	Überarbeitet und ergänzt (Sky-Writing-Modus 2).
146	“Automatische Lasersignalunterdrückung bei Status-Fehlern des Scan-Systems“	Neu hinzugefügt.
234	Kapitel 9.1.5 “Schrittmotoransteuerung“	Neu hinzugefügt.
314	**get_marking_info**-Befehl	Ergänzt: liefert jetzt auch die Fehlerbits der automatischen Lasersignalunterdrückung zurück (Bits#10…15 und 26…31).
316	**get_master_slave**-Befehl	Korrigiert: Master/Slave-Erkennung über Bit#0 und 1.
324	**get_stepper_status**-Befehl	Neu hinzugefügt.
338, 339	**goto_xy**-Befehl, **goto_xyz**-Befehl	Geändert (ab Version DLL 527, OUT 529): Befehlsrückkehr erst nach Ende der Bewegung.
354	**laser_on_list**-Befehl	Korrigiert: Die Lasersteuersignale werden erst durch einen nachfolgenden Listenbefehl ausgeschaltet.
375	**load_correction_file**-Befehl	Geändert (ab Version DLL 527, OUT 529): Befehlsrückkehr erst nach Sprung auf die korrigierte Galvanometerposition.
382	**load_jump_table**-Befehl	Korrigiert: hat keinen Parameter `Offset`.
383	**load_jump_table_offset**-Befehl	Neu hinzugefügt: Funktionsweise so wie in Version 1.4 für den **load_jump_table**-Befehl beschrieben (mit Parameter `Offset`).
548, 549	**set_sky_writing_mode**-Befehl, **set_sky_writing_mode_list**-Befehl	Neu hinzugefügt.
550, 552	**set_sky_writing_para**-Befehl, **set_sky_writing_para_list**-Befehl	Erweitert: Sky-Writing-Modus 2.

Seite	Name des Kapitels / Befehls	Änderung
586 … 595	Befehle zur Schrittmotorsteuerung: **stepper_abs**, **stepper_abs_list**, **stepper_abs_no**, **stepper_abs_no_list**, **stepper_control**, **stepper_control_list**, **stepper_enable**, **stepper_enable_list**, **stepper_init**, **stepper_rel**, **stepper_rel_list**, **stepper_rel_no**, **stepper_rel_no_list**, **stepper_wait**	Neu hinzugefügt.
603	**sync_slaves**-Befehl	Korrigiert: Master-Karte ist diejenige Karte, an deren SLAVE-Anschluss (!) keine weitere Karte angeschlossen ist.
651	Kapitel 14 "Technische Spezifikationen der RTC5 PCI-Karte"	Ergänzt: Spezifikationen zum "SPI / I2C"-Anschluss und "STEPPER MOTOR"-Anschluss.
654	Kapitel 14.1 "Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV)"	Neu hinzugefügt.
654	Kapitel 14.2 "Konformität mit FCC-Bestimmungen"	Neu hinzugefügt.
656 und 670	"Software" und Kapitel 15.3.4 "Installation der Treiber und RTC5-Software"	Warnhinweis hinzugefügt: Standby- oder Sleep-Modi des Betriebssystems sind zu deaktivieren.
672	Kapitel 16 "Anhang B: Die RTC5 PCI-Express-Karte"	Neu hinzugefügt.
439	**read_analog_in**-Befehl	Neu hinzugefügt.

Änderungen von Version 1.5 nach Version 1.6

Seite	Name des Kapitels / Befehls	Änderung
22	Kapitel 1.3.1 "Mitgelieferte Software" und weitere Kapitel	Ergänzt: Das RTC5-Software-Paket enthält jetzt auch Treiber für 64-Bit-Windows-Betriebssysteme sowie Win32- und Win64-basierte DLLs, dagegen keine MS Visual Studio Laufzeitbibliotheken mehr. Beachten Sie hierzu auch die entsprechenden Hinweise unter "Installation der RTC5-Software", Seite 62, "Anwenderprogramme und Demoprogramme", Seite 64, "Initialisierung und Programmstart", Seite 66 und "Datentypen", Seite 260.
26	Kapitel 2.2.2 "Software"	Ergänzt: Anforderungen bei neuen und älteren RTC5-Software-Paketen (neue RTC5-DLLs benötigen mindestens Windows XP SP2).
48	"Ausgangsbeschaltung" für Laseranschluss-Buchse	Neu hinzugefügt.
102	Kapitel 6.9.1 "Freie Variablen"	Neu hinzugefügt.
128	"Modus 3" für Sky-Writing	Neu hinzugefügt (neu ab Version DLL 531, OUT 532).
157	Kapitel 7.4.8 "Pulse-Picking-Lasermodus"	Neu hinzugefügt.
187	Kapitel 8.3 "Online-Positionierung"	Änderung (ab Version DLL 531, OUT 532): Stehen zum Zeitpunkt der Anforderung durch **apply_mcbsp** (oder **apply_mcbsp_list**) zwei gleiche Datentypen (mit identischen Codierungs-Bits #31) auf den beiden Speicherstellen, dann wird jeweils der zuletzt übertragene Wert verwendet (bei älteren Versionen: immer der Wert auf Speicherstelle 2).
199	Kapitel 8.7 ""Processing on the fly" (optional)"	Korrektur über McBSP-Schnittstelle mit zusätzlicher McBSP-Eingabe (siehe neue Abschnitte Seite 203 und Seite 205 und und Kapitel 8.7.5 "Deaktivieren der "Processing on the fly"-Korrektur").
237	Kapitel 9.1.7 "McBSP/SPI-Schnittstelle"	Ergänzt (**set_mcbsp_out_ptr**).
241	Neuer Hinweis zum "Externer Listen-Start"	Externe Starts sind nach **pause_list**, **stop_list** oder **set_wait** unterbunden.
304	**get_free_variable**-Befehl	Neu hinzugefügt.
326	**get_sync_status**-Befehl	Neu hinzugefügt.
332	**get_value**-Befehl	Ergänzt (ab Version DLL 531, OUT 532): `Signal` = 25…42.
355	**laser_on_pulses_list**-Befehl	Veränderte Funktionalität für $2^{31} \leq$ `Period` $\leq (2^{32}-1)$ ab Version DLL 530, OUT 531, RBF 522.
443	**read_mcbsp**-Befehl	Ergänzt (ab Version DLL 531, OUT 532): `No` = 3 und **set_mcbsp_in**-Eingabe.

470	**set_control_mode**-Befehl, **set_control_mode_list**-Befehl	Ergänzt: Bit#1 (ab Version DLL 528, OUT 530).
495	**set_free_variable**-Befehl, **set_free_variable_list**-Befehl	Neu hinzugefügt.
498	**set_jump_mode**-Befehl	Veränderter Errorcode für Rückgabe >1 ab Version DLL 530, OUT 531, RBF 522.
517	**set_mcbsp_freq**-Befehl	Neu hinzugefügt.
518	**set_mcbsp_in**-Befehl, **set_mcbsp_in_list**-Befehl	Neu hinzugefügt.
521	**set_mcbsp_out**-Befehl	Ergänzt (ab Version DLL 531, OUT 532): `Signal1, Signal2` = 25...42.
522	**set_mcbsp_out_ptr**-Befehl	Neu hinzugefügt.
538	**set_pulse_picking**-Befehl, **set_pulse_picking_list**-Befehl	Neu hinzugefügt.
547	**set_sky_writing_limit**-Befehl, **set_sky_writing_limit_list**-Befehl	Neu hinzugefügt (neu ab Version DLL 531, OUT 532).
548	**set_sky_writing_mode**-Befehl, **set_sky_writing_mode_list**-Befehl	Ergänzt (ab Version DLL 531, OUT 532): Modus 3.
562	**set_trigger**-Befehl	Ergänzt (ab Version DLL 531, OUT 532): `Signal1, Signal2` = 25...42.
678	Kapitel 17 "Anhang C: Die RTC5 PCIe/104-Karte"	Neu hinzugefügt.

Änderungen von Version 1.6 nach Version 1.7

Seite	Name des Kapitels / Befehls	Änderung
26	Kapitel 2.2.2 "Software"	Ergänzung: Hinweise zu neuen Treibern.
59	Kapitel 4.4.8 "Analog-Eingänge (Sonderzubehör)"	Neu hinzugefügt.
61	Kapitel 5.3 "Installation der Treiber"	Ergänzung: Installation unter Windows 8 und neues Kapitel 5.3.1 "Software-Paket-Upgrades", Seite 62.
146	Kapitel 7.4.2 "Konfigurieren der Laseranschluss-Buchse (LASER)"	Neu hinzugefügt.
157	Kapitel 7.4.8 "Pulse-Picking-Lasermodus"	Ergänzung: "konstante Pulslänge"-Modus.
169	Kapitel 7.4.10 "Ausgabe-Synchronisation"	Neu hinzugefügt.
187	Kapitel 8.3 "Online-Positionierung"	Ergänzung: Matrixoperation (**set_mcbsp_matrix**, **set_mcbsp_matrix_list**).
210	Kapitel 8.7.6 "Virtuelles Bildfeld"	Ergänzung: Beispiele zur Verwendung des virtuellen Bildfelds bei "Processing on the fly"-Anwendungen.
215	Kapitel 8.7.10 "Schleppverzugskompensation von Encoderwerten für "Processing on the fly"-Anwendungen"	Neu hinzugefügt.
270	**auto_cal**-Befehl	Änderung mit Version DLL 535: Rückgabewert 8.
280	**config_laser_signals**-Befehl	Neu hinzugefügt (ab DLL 533, OUT 534, RBF 524).
302	**get_error**-Befehl	Ergänzung mit Version DLL 535: Bits #11, 12, 16.
308	**get_hex_version**-Befehl	Änderung mit Version DLL 535.
319	**get_rtc_version**-Befehl	Änderung mit Version DLL 535.
354	**laser_on_list**-Befehl	Korrektur: geändertes Verhalten ab Version DLL 530, OUT 531, RBF 522(wie bei **laser_on_pulses_list**).
489	**set_fly_tracking_error**-Befehl	Neu hinzugefügt (ab DLL 533, OUT 534, RBF 524)
504	**set_laser_control**-Befehl	Ergänzung: Bit#6 für die Ausgabe-Synchronisation und Bit#7 für den "konstante Pulslänge"-Modus (ab Version DLL 530, OUT 531, RBF 522 bzw. DLL 533, OUT 534, RBF 524).
517	**set_mcbsp_freq**-Befehl	Korrektur: Verhalten bei überlaufenden Parameterwerten.
520	**set_mcbsp_matrix**-Befehl, **set_mcbsp_matrix_list**-Befehl	Neu hinzugefügt (ab DLL 533, OUT 534, RBF 524).
538	**set_pulse_picking_length**-Befehl	Neu hinzugefügt (ab DLL 533, OUT 534, RBF 524).
571	**set_verify**-Befehl	Ergänzung mit Version DLL 535: automatische Fehlererkennung einer fehlgeschlagenen Speicheranforderung.
596	**stop_execution**-Befehl	Änderung mit Version OUT 535.

Änderungen von Version 1.7 nach Version 1.8

Seite	Name des Kapitels / Befehls	Änderung
91	Kapitel 6.5.5 "Schleifen"	Neu hinzugefügt (Version DLL 536, OUT 536).
144	Kapitel 7.4.1 "Freigeben, Aktivieren und Schalten der Lasersteuersignale"	Korrigiert: Ein (hochohmiger) tristate-Zustand der Lasersignal-Ausgänge (LASERON, LASER1 und LASER2) stellt sich auch dann ein, wenn die Lasersteuersignale nachträglich durch einen Befehl unterdrückt werden (siehe auch **disable_laser**, **pause_list** und **set_laser_control**(Bit#2 = 1)).
169	Kapitel 7.4.10 "Ausgabe-Synchronisation"	Hinweise geändert (Version RBF 527).
179	"Hinweise zum Laden der ermittelten Jump-Delay-Werte" (Kapitel 8.1.5 "Sprungmodus")	Korrigiert: der maximale *Length*-Parameter-Wert für Jump-Delay-Tabellen ist nicht 524288.0, sondern 1048576.0.
192	Kapitel 8.5 "Verwendung mehrerer verschiedener Korrekturtabellen"	Geändert (Version DLL 533, OUT 534): es können bis zu 4 Korrekturdateien geladen werden.
198	Kapitel 8.6.5 "Erweiterte 3D-Korrektur"	Neu hinzugefügt (Version DLL 536, OUT 536).
199	Kapitel 8.7 ""Processing on the fly" (optional)"	Änderungen (Version DLL 536, OUT 536): • neu: Abschnitt "Übersicht". • neu: Kapitel 8.7.4 "Kompensation zweidimensionaler Bewegungen" (inkl. Abschnitt "2D-Encoder-Kompensation für XY-Tische" und Abschnitt "Koordinatentransformationen im virtuellen Bildfeld"). • überarbeitet: Kapitel 8.7.6 "Virtuelles Bildfeld". • ergänzt: Kapitel 8.7.7 "Synchronisation von "Processing on the fly"-Anwendungen". • neu Kapitel 8.7.8 "Encoder-Resets".
252	"Multiple Listenbefehle"	Neu hinzugefügt.
262	**activate_fly_2d**-Befehl, **activate_fly_xy**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
304	**get_fly_2d_offset**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
314	**get_marking_info**-Befehl	Änderung (Version DLL 536, OUT 536): Bit #9.
327	**get_table_para**-Befehl	Geändert (Version DLL 533, OUT 534): Wertebereich von `TableNo`.
332	**get_value**-Befehl, **get_values**-Befehl	Geändert (Version DLL 536, OUT 536): Weitere auswählbare Datensignale (`Signal` = 43 und 44).
336	**get_waveform**-Befehl	Geändert (Version DLL 536, OUT 536): `Channel` = 3 und 4.
344	**if_not_activated**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
347	**init_fly_2d**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
370	**list_repeat**-Befehl, **list_until**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
375	**load_correction_file**-Befehl	Geändert (Version DLL 533, OUT 534): Wertebereich von `No` (es können bis zu 4 Korrekturdateien geladen werden).
381	**load_fly_2d_table**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).

392	**load_stretch_table**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
418	**micro_vector_abs**-Befehl, **micro_vector_rel**-Befehl	Änderung (Version DLL 536, OUT 536): `LasOn`, `LasOff` < 2^{15}.
419	**micro_vector_abs_3d**-Befehl, **micro_vector_rel_3d**-Befehl	Neu hinzugefügt (ab Version DLL 536, OUT 536).
431	**park_position**-Befehl, **park_return**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
457	**select_cor_table**-Befehl, **select_cor_table_list**-Befehl	Geändert (Version DLL 533, OUT 534): Wertebereich der Parameter.
476	**set_delay_mode_list**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
478	**set_encoder_speed**-Befehl, **set_encoder_speed_ctrl**-Befehl	Geändert (Version DLL 536, OUT 536): `EncoderNo` = 2 und 3.
484	**set_fly_2d**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
504	**set_laser_control**-Befehl	Geändert (Version DLL 536, OUT 536): Bit#28.
515	**set_matrix**-Befehl	Änderung (Version DLL 536, OUT 536): `HeadNo` = 4.
530	**set_offset**-Befehl, **set_offset_xyz**-Befehl	Änderung (Version DLL 536, OUT 536): `HeadNo` = 4.
536	**set_pixel_line_3d**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
554	**set_softstart_level_list**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
556	**set_softstart_mode_list**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
562	**set_trigger**-Befehl	Geändert (Version DLL 536, OUT 536): `Signal1`, `Signal2` = 43 und 44.
568	**set_trigger4**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).
610	**timed_para_jump_abs_3d**-Befehl, **timed_para_jump_rel_3d**-Befehl, **timed_para_mark_abs_3d**-Befehl, **timed_para_mark_rel_3d**-Befehl	Korrigiert: das sind *multiple* Listenbefehle (d.h. nicht normale Listenbefehle).
632	**wait_for_encoder_in_range**-Befehl	Neu hinzugefügt (Version DLL 536, OUT 536).

Änderungen von Version 1.8 nach Version 1.9

Seite	Name des Kapitels / Befehls	Änderung
170	Kapitel 7.5.2 "Markieren von Seriennummern"	Ergänzt (ab Version DLL 537, OUT 537): Markierung von Seriennummern jetzt auch mit auswählbarem Seriennummernsatz.
189	Kapitel 8.4 "Wobbelmode"	Ergänzt: Richtungsvorgabe und "Frei definierbare Wobbelfiguren".
190	Kapitel 8.4.1 "Wobbelfiguren – Wichtige Hinweise zur Wahl angemessener Parameterwerte"	Neu hinzugefügt.
203	Kapitel "Korrektur über McBSP/SPI-Schnittstelle mit erweiterter McBSP/SPI-Eingabe"	"Processing on the fly"-Korrektur in Z-Richtung und die Anpassung der Laserleistung ist jetzt möglich.
278	**clear_fly_overflow**-Befehl	Änderung ab Version DLL 531, OUT 532: Bits #4, 5.
280	**config_laser_signals_list**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
298	**fly_return_z**-Befehl	Neu hinzugefügt (ab Version DLL 531, OUT 532).
313	**get_list_serial**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
343	**if_fly_z_overflow**-Befehl	Neu hinzugefügt (ab Version DLL 531, OUT 532).
346	**if_not_fly_z_overflow**-Befehl	Neu hinzugefügt (ab Version DLL 531, OUT 532).
389	**load_program_file**-Befehl	Ergänzt (ab Version DLL 537, OUT 537): PCI-Fehler jetzt mit eigenem Fehlercode 16.
436	**range_checking**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
444	**read_multi_mcbsp**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
461, 461	**select_serial_set**-Befehl, **select_serial_set_list**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
546	**set_serial_step_list**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
486	**set_fly_limits_z**-Befehl	Neu hinzugefügt (ab Version DLL 531, OUT 532).
526	**set_multi_mcbsp_in**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
528	**set_multi_mcbsp_in_list**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
550	**set_sky_writing_para**-Befehl	Parameter `Timelag` geändert (ab Version DLL 537, OUT 537).
575	**set_wobbel_control**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
576	**set_wobbel_direction**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
577	**set_wobbel_mode**-Befehl	Änderung ab Version DLL 537, OUT 537: `Mode`s > 0.
579	**set_wobbel_offset**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).
580	**set_wobbel_vector**-Befehl	Neu hinzugefügt (ab Version DLL 537, OUT 537).

Änderungen von Version 1.9 nach Version 1.10

Seite	Name des Kapitels / Befehls	Änderung
30	Kapitel 2.5.6 "RTC5 varioSCAN FLEX Extension-Karte"	Neu hinzugefügt.
30	Kapitel 2.6 "Ergänzende Software"	Ergänzung.
54	Kapitel 4.4.6 "SPI / I2C-Stiftleiste"	Ergänzung (ab Version DLL 538, OUT 538): Die "SPI / I²C" Stiftleiste kann jetzt auch mit SPI-Funktionalität (Serial Peripheral Interface) initialisiert werden: "SPI-Schnittstelle". Der in diesem Handbuch lange etablierte Begriff "McBSP-Schnittstelle" wird um "/SPI" ergänzt und so weiterhin verwendet.
84	Abschnitt Aufrufe mit wiederholter Ausführung	Neu hinzugefügt.
102	Kapitel 6.9.1 "Freie Variablen"	Geändert: ab DLL 539, OUT 539 sind 8 freie Variablen möglich.
128	Abschnitt Modus 3 (Sky-Writing)	Ergänzung.
129	Abb. 44	Berichtigt.
142	Abschnitt "Vorausberechnung und Diagnose"	Neu hinzugefügt.
229	Kapitel 8.12 "Camming"	Neu hinzugefügt.
249	Kapitel 9.4 "Periodische I/O-Signale"	Neu hinzugefügt.
270	**auto_cal**-Befehl	Ergänzung des Textes: Fehlercode 9, 90.
276	**camming**-Befehl	Neu hinzugefügt (ab Version DLL 512, OUT 511).
304	**get_free_variable**-Befehl	Geändert (DLL 539, OUT 539): Wertebereich des Parameters `No` erhöht.
305	**get_galvo_controls**-Befehl	Neu hinzugefügt (ab Version DLL 539, OUT 539).
307	**get_head_status**-Befehl	Ergänzung.
415	**mcbsp_init_spi**-Befehl	Neu hinzugefügt (ab Version DLL 512, OUT 511).
434	**periodic_toggle**-Befehl	Neu hinzugefügt (ab Version DLL 538, OUT 538).
435	**periodic_toggle_list**-Befehl	Neu hinzugefügt (ab Version DLL 538, OUT 538).
438	**read_abc_from_file**-Befehl	Neu hinzugefügt (ab Version DLL 538, OUT 538).
495	**set_free_variable**-Befehl	Geändert (DLL 539, OUT 539): Wertebereich des Parameters `No` erhöht.
562	**set_trigger**-Befehl	Geändert (DLL 538, OUT 538): `Signal1`, `Signal2` = 45 und 46.
562	**set_trigger**-Befehl	Geändert (DLL 539, OUT 539): `Signal1`, `Signal2` = 47…51.
580	**set_wobbel_vector**-Befehl	Berichtigt: Formel zur Variation der aktuellen Laserleistung P im Wobbelfigurabschnitt.
600	**sub_call_abs_repeat**-Befehl	Neu hinzugefügt (ab Version DLL 538, OUT 538).
601	**sub_call_repeat**-Befehl	Neu hinzugefügt (ab Version DLL 538, OUT 538).
635	**write_abc_to_file**-Befehl	Neu hinzugefügt (ab Version DLL 538, OUT 538).
639	**write_hi_pos**-Befehl	Neu hinzugefügt (ab Version DLL 538, OUT 538).

Seite	Name des Kapitels / Befehls	Änderung
651	Kapitel 14 "Technische Spezifikationen der RTC5 PCI-Karte"	Unter "SPI / I2C"-Anschluss die Werte der SPI-Konfiguration hinzugefügt.
677	Kapitel 16.5 "Konformität mit der EU-Richtlinie zur elektromagnetischen Verträglichkeit (EMV)"	Neu hinzugefügt.

Änderungen von Version 1.10 nach Version 1.11

Seite	Name des Kapitels / Befehls	Änderung
22	Kapitel 1.3.1 "Mitgelieferte Software" u.a.	Ergänzung (Windows 10 wird unterstützt).
81	Abschnitt "Nichtindizierte Unterprogramme"	Ergänzung.
95	Abschnitt "Beispiel-Code"	Neu hinzugefügt (Ermittlung der Master/Slave Karten).
165	Abschnitt"Geschwindigkeitsabhängige Lasersteuerung"	Ergänzung (Hinweis zu **set_auto_laser_control** `Mode` = 6).
287	**control_command**-Befehl	Geändert: Formel für Kühlflussrate bei 05_H3B_H.
287	**control_command**-Befehl	Geändert: Formel für Schutzglasfenster-Streulichtwert bei 05_H3C_H.
360	**list_call_abs_repeat**-Befehl	Neu hinzugefügt (ab Version DLL 540, OUT 540).
362	**list_call_repeat**-Befehl	Neu hinzugefügt (ab Version DLL 540, OUT 540).
422	**move_to**-Befehl	Die Beschreibung des Befehls wurde verschoben in das Handbuch "Installation and Operation RTC Step Motor Extension for the RTC4 and RTC5 PC interface boards" (nur in Englisch verfügbar).
464	**set_auto_laser_control**-Befehl	Geändert (Version DLL 540, OUT 540): `Mode` = 6.
590	**stepper_enable**-Befehl	Korrigiert: der Befehl hat keine Rückgabe.
684	Kapitel 17.4 "Technische Spezifikationen der RTC5 PCIe/104-Karte"	Ergänzung (Hinweis zu ANALOG OUT1 und ANALOG OUT2 bei der RTC5 PCIe/104-Karte).

Änderungen von Version 1.11 nach Version 1.12

Seite	Name des Kapitels / Befehls	Änderung
22	Kapitel 1.3.1 "Mitgelieferte Software"	Ergänzung (das RTC5-Software-Paket enthält jetzt auch Dateien für CMake).
62	Kapitel 5.3.2 "Tausch von RTC-Karten und Update von RTC-Karten-Treibern"	Neu hinzugefügt. Weitere Informationen finden Sie in der (mit Version DLL 542 aktualisierten) Datei `ReadMe_ScanlabClassChecker.pdf`.
208	Abschnitt "Koordinatentransformationen im virtuellen Bildfeld"	Ergänzung (Koordinatentransformationen im virtuellen Bildfeld sind jetzt auch mit den Befehlen **set_fly_x** und/oder **set_fly_y** möglich; ab Version DLL 541, OUT 541).
215	Kapitel 8.7.11 ""Processing on the fly"-Korrektur für die Z-Achse — "FlyZ-Korrektur" (ab Version DLL 530, OUT 531)"	Neu hinzugefügt.
231	Kapitel 8.13 "Zeitmessungen"	Neu hinzugefügt.
287	**control_command**-Befehl	Geändert: Wertebereich verringert (von 2250 auf 1992) für Temperatur von Spiegel 2 bei 05_H36_H.
287	**control_command**-Befehl	Geändert: Wertebereich verringert (von 2250 auf 1992) für Temperatur von Spiegel 1 bei 05_H37_H.
311	**get_lap_time**-Befehl	Neu hinzugefügt (ab Version DLL 541, OUT 541).
369	**list_next**-Befehl	Neu hinzugefügt (ab Version DLL 541, OUT 541).
494	**set_fly_z**-Befehl	Neu hinzugefügt (ab Version DLL 530, OUT 531).
539	**set_pulse_picking_list**-Befehl	Korrigiert: dieser Befehl ist ein unverzögerter kurzer Listenbefehl.
568	**set_trigger4**-Befehl	Korrigiert: dieser Befehl ist ein verzögerter kurzer Listenbefehl.
589	**stepper_disable_switch**-Befehl	Neu hinzugefügt (ab Version DLL 542, OUT 542).

Änderungen von Version 1.12 nach Version 1.13

Seite	Name des Kapitels / Befehls	Änderung
52	Kapitel 4.4.3 "EXTENSION 2-Stiftleiste", Abb. 17	Korrigiert: Das Signal an Pin (26) ist LASERON (nicht: NOT CONNECTED).
59	Kapitel 4.4.8 "Analog-Eingänge (Sonderzubehör)"	Ergänzung: Die analogen Eingabewerte (ANALOG IN0 und ANALOG IN1, siehe **read_analog_in**) können mit dem Triggersignal 54 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**) (ab Version DLL 543, OUT 543, RBF 524).
189	Kapitel 8.4 "Wobbelmode"	Ergänzung: Die momentane Wobbelamplitude (aus **set_wobbel** oder **set_wobbel_mode**) kann mit dem Triggersignal 53 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**) (ab Version DLL 543, OUT 543, RBF 524).
231	Kapitel 8.13 "Zeitmessungen"	Ergänzung: Zeitstempel-Zähler. Kann mit dem Triggersignal 52 aufgezeichnet werden (siehe **set_trigger** oder **set_trigger4**) (ab Version DLL 543, OUT 543, RBF 524).
244	Kapitel 9.3.2 "Bedingte Befehlsausführung"	Ergänzung: Kontrollbefehl **set_pause_list_cond** (ab Version DLL 543, OUT 543).
434	**periodic_toggle**-Befehl	Geändert: Endloses Toggeln bei `Count` = 2^{32}–1 (ab Version DLL 543, OUT 543).
435	**periodic_toggle_list**-Befehl	Geändert: Endloses Toggeln bei `Count` = 2^{32}–1 (ab Version DLL 543, OUT 543).
470	**set_control_mode**-Befehl	Ergänzung: Bit #4 (ab Version DLL 543, OUT 543).
472	**set_control_mode_list**-Befehl	Ergänzung: Bit #4 (ab Version DLL 543, OUT 543).
533	**set_pause_list_cond**-Befehl	Neu hinzugefügt (ab Version DLL 543, OUT 543).
562	**set_trigger**-Befehl	Geändert: `Signal1`, `Signal2` = 52…54 (ab Version DLL 543, OUT 543, RBF 524).
568	**set_trigger4**-Befehl	Geändert: `Signal1`, `Signal2` = 52…54 (ab Version DLL 543, OUT 543, RBF 524).
652	Abschnitt "EXTENSION 2-Anschluss"	Korrigiert: fehlendes LASERON-Signal hinzugefügt.