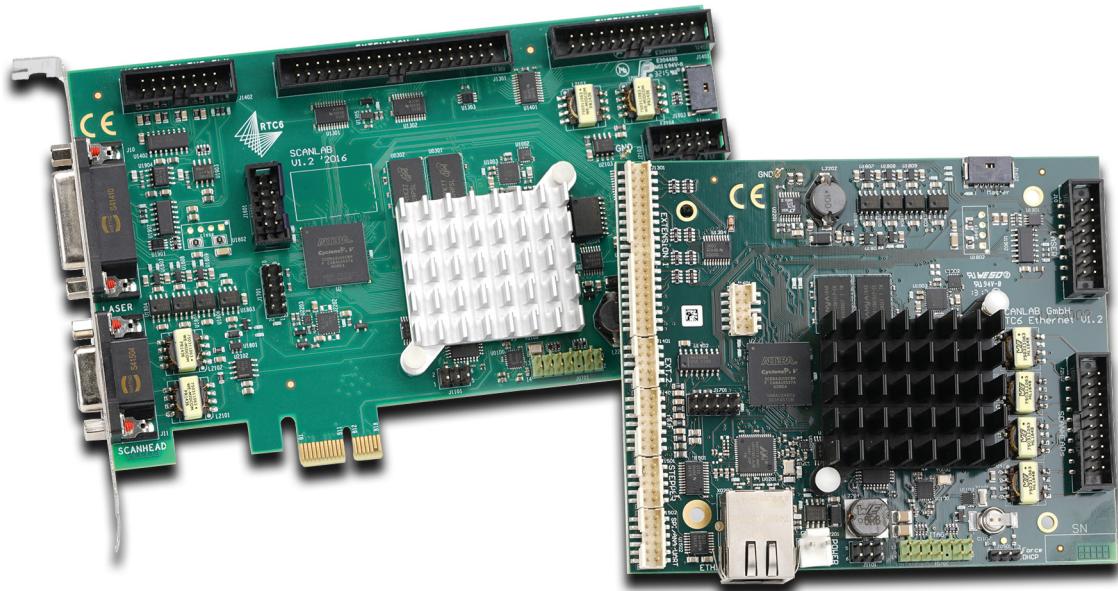




# Installation and Operation

## RTC6 PCIe Board and RTC6 Ethernet Board

for Real Time Control of Scan Heads and Lasers  
RTC6 Software Package V1.7.0



SCANLAB GmbH  
Siemensstr. 2a  
82178 Puchheim  
Germany

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

[info@scanlab.de](mailto:info@scanlab.de)  
[www.scanlab.de](http://www.scanlab.de)

© SCANLAB GmbH 2020  
(Doc. Rev. 1.0.4 en-US - March 13, 2020)

SCANLAB GmbH reserves the right to change the information in this document without notice.

No part of this document may be processed, reproduced or distributed in any form (photocopy, print, microfilm or by any other means), electronic or mechanical, for any purpose without the written permission of SCANLAB GmbH.

RTC is a registered trademark of SCANLAB GmbH.

Other mentioned trademarks are hereby acknowledged as properties of their respective owners.



## Contents

<b>1</b>	<b>Introduction</b>	<b>23</b>
1.1	Manufacturer	23
1.2	Labeling	24
1.3	Unpacking Instructions and Typical Scope of Delivery	24
1.3.1	Delivered RTC6 Software Package	24
Folder Correction Files	24	
Folder CorrectionFileConverter	24	
Folder Demo Files	25	
Folder HPGL	25	
Folder iSCANCfg	25	
Folder RTC6 Driver	25	
Folder RTC6 Tools	25	
Folder RTC6 Tools	26	
<b>2</b>	<b>Product Overview</b>	<b>27</b>
2.1	Intended Use	27
2.2	System Requirements	29
2.2.1	Hardware	29
2.2.2	Software	29
2.3	Options	30
2.4	Jumper Settings and Type Designations	32
2.4.1	Solder Jumper Field A – Output Signal Level at the EXTENSION 1 Socket Connector Configuration	33
2.4.2	Solder Jumper Field B – Pin (17) of the EXTENSION 2 Socket Connector Configuration	34
2.4.3	Solder Jumper Field C – Pin (15) of the EXTENSION 2 Socket Connector Configuration	35
2.5	Accessories for the RTC6 PCIe Board	36
2.5.1	XY2-100 Converter	36
2.5.2	Laser Adapter	36
2.5.3	Data Cables	36
2.5.4	Slot Cover with 9-pin D-SUB Connector for Using the "2. SCANHEAD" Socket Connector	36
2.5.5	Slot Cover with 15-pin D-SUB Connector for Using the "MARKING ON THE FLY" Socket Connector	37
2.6	Supplementary Software	37
2.7	Notes for RTC4 Users	38
2.7.1	Hardware Changes	38
Controlling Scan Systems	38	
Controlling the Laser	38	
EXTENSION 1 Socket Connector	38	
EXTENSION 2 Socket Connector	39	
MARKING ON THE FLY Socket Connector	39	
Other Hardware Interfaces	39	
2.7.2	Porting RTC4 Source Code to the RTC6 PCIe Board	39
Changed Initialization	39	
Command Changes	40	
Increased Parameter Resolution	41	
Changed Timing Behavior	42	
2.7.3	New and Changed Functionality	42
Interface to the PC	42	
Controlling Scan Systems	42	
Controlling the Laser	43	



Interfaces for Peripheral Equipment .....	44
General Programming .....	45
Laser Marking .....	45
Special Functions .....	46
2.8 Notes for RTC5 Users .....	47
2.8.1 RTC6 PCIe Boards vs. RTC5 Boards .....	47
Functionality changes .....	47
Parameter changes .....	48
2.8.2 RTC6 PCIe Boards and Scan Systems with SCANAhead Servo Control .....	48
2.8.3 Restrictions with RTC6 PCIe Boards .....	49
Availability of Technical Variants .....	49
Peripheral Interfaces .....	49
Functionality .....	49
2.9 Source Code for RTC6 User Programs .....	49
2.9.1 Creating New RTC6 Source Code .....	49
2.9.2 Adapting RTC5 Source Code for the RTC6 PCIe Board .....	49
Mandatory Steps .....	49
Optional Steps and Notes on Further Modification and Optimization .....	50
2.10 Changes to the RTC6 Command Set .....	51
2.10.1 Changes to the RTC6 Command Set .....	51
2.10.2 Removed from RTC6 Command Set .....	51
3 Safety During Installation and Operation .....	52
3.1 Safe Operation .....	52
3.2 Laser Safety .....	52
4 RTC6 PCIe Board – Layout and Interfaces .....	53
4.1 Layout – Upper Side .....	53
4.2 Layout – Lower Side .....	54
4.3 Interface to the PC .....	55
4.4 Master Socket Connector, Slave Socket Connector .....	55
4.5 Interfaces to Scan System .....	56
4.5.1 Scan Head Connectors and Transfer Protocol .....	56
SCANHEAD Connector (Connector for First Scan Head) .....	57
2. SCANHEAD Socket Connector (Connector for Second Scan Head) .....	57
4.5.2 XY2-100 Converter (Accessory) .....	58
4.5.3 Data Cables (Accessories) .....	60
4.6 Interfaces for the Laser and Peripheral Equipment .....	62
4.6.1 LASER Connector .....	62
Laser Output Signals .....	62
External Control Signals .....	63
BUSY Status .....	63
2-Bit Digital Input Port .....	63
2-Bit Digital Output Port .....	63
12-Bit Analog Output Port 1 and 2 .....	63
Input and Output Wiring .....	63
Laser Adapter (Accessory) .....	65
4.6.2 EXTENSION 1 Socket Connector .....	67
Configuring the Output Signal Level .....	67
16-Bit Digital Input and 16-Bit Digital Output .....	67
Synchronization of Data Acquisition .....	68
BUSY Status .....	68
4.6.3 EXTENSION 2 Socket Connector .....	69



Configuration by Solder Jumpers .....	69
Laser Output Signals .....	70
8-Bit Digital Output Port .....	70
4.6.4 MARKING ON THE FLY Socket Connector .....	71
Encoder Inputs .....	71
External Control Signals .....	71
Analog Output Port .....	71
BUSY Status .....	71
Slot Cover (Accessory) .....	72
4.6.5 RS232 Socket Connector .....	72
4.6.6 McBSP/ANALOG Socket Connector .....	73
McBSP Interface .....	73
Analog Input Ports .....	76
4.6.7 STEPPER MOTOR Socket Connector .....	77
<b>5 Installation and Start-Up .....</b>	<b>78</b>
5.1 Checking Jumper Settings .....	78
5.2 Installing the RTC6 PCIe Board .....	78
5.3 Installing the RTC6 Board Driver .....	79
5.4 Installing the RTC6 Software .....	80
5.5 Safe Start-up and Shutdown Sequences .....	81
5.6 Functionality Test .....	81
5.7 User Programs and Demo Programs .....	82
<b>6 Developing RTC6-User Programs .....</b>	<b>83</b>
6.1 RTC6 Software Concept Basics .....	83
6.1.1 Controlling Scan Systems and Lasers – An Introductory Example .....	83
6.1.2 Control Commands and List Commands .....	84
6.2 Initialization and Program Start-Up .....	85
6.2.1 DLL Calling Convention .....	85
6.2.2 Importing Commands .....	85
Pascal .....	86
C .....	86
C++ .....	86
C# .....	86
6.2.3 Initializing the RTC6 DLL and RTC6 Board Management .....	87
6.2.4 Start of RTC6 PCIe Board Operation .....	88
Initializing the Board .....	88
Configuring the Board .....	88
Initializing the Scan System Control .....	89
Initializing the Laser Control .....	89
Loading and Executing Lists .....	89
6.2.5 Example Code .....	90
6.3 RTC6 List Memory .....	92
6.3.1 Lists and the Protected List Memory Area .....	92
"List 1" and "List 2" .....	92
"List 3" – Protected List Memory Area .....	92
6.3.2 Configuring the RTC6 List Memory .....	93
6.4 List Handling .....	95
6.4.1 Loading Lists .....	95
"Unconditional" Loading .....	95
Loading with Protection .....	96
Terminating Lists .....	96



6.4.2	List Status .....	97
6.4.3	List Execution Status .....	98
6.4.4	Starting and Stopping Lists .....	99
6.4.5	Interrupting Lists for Synchronization of Processing .....	100
6.4.6	Changing Lists Automatically .....	100
	One-Time List Change .....	100
	Alternating List Changes .....	101
6.5	Structured Programming .....	102
6.5.1	Subroutines .....	102
	Non-Indexed Subroutines .....	102
	Indexed Subroutines .....	103
	General Information on Calling Subroutines .....	104
	Index Management and Defragmentation .....	105
	Subsequent Protection and Conversion of Non-Indexed Subroutines .....	106
	Unprotecting Subroutines .....	107
6.5.2	Character Sets and Text Strings .....	108
	Defining Indexed Character Sets .....	108
	Calling Indexed Characters .....	109
	Defining Indexed Text Strings for Time, Date and Serial Number .....	109
	Calling Indexed Text Strings .....	110
	Managing Indexed Characters and Text Strings .....	110
6.5.3	Jumps .....	110
6.5.4	RTC4-Circular Queue Mode .....	111
6.5.5	Loops .....	111
6.6	Using Several RTC6 PCIe Boards in One PC .....	113
6.6.1	Multi-Board Programming .....	113
	Examples (Pascal) .....	113
6.6.2	Single-Board Programming .....	114
6.6.3	Master/Slave Operation .....	114
	Initialization with RTC6 Software Package $\geq$ 1.5.0 ( $\geq$ RBF 619) .....	115
	Initialization with RTC6 Software Package $<$ V1.5.0 ( $\leq$ RBF 618) .....	115
	Synchronous Starts and Stops .....	116
	Example Code .....	117
6.7	Usage of RTC6 PCI Express Boards by Several User Programs .....	118
6.7.1	Notes on Board Acquisition by a User Program .....	118
6.8	Error Handling .....	120
6.8.1	Download Verification .....	121
6.8.2	Checking for Overruns .....	121
6.8.3	Example Code .....	122
6.9	Miscellaneous .....	124
6.9.1	Free Variables .....	124
7	Basic Functions for Scan Head Control and Laser Control .....	125
7.1	Marking Points, Lines and Arcs .....	125
7.1.1	Marking with Vector and Arc Commands .....	125
	Jump Commands .....	126
	Mark Commands .....	126
	Arc Commands .....	126
	Polylines .....	126
	Ellipse Commands .....	127
	[*]Para[*] Commands .....	128
7.1.2	Microstepping .....	129
7.1.3	Marking Single Points .....	130



7.1.4	Example Code .....	131
7.2	Delay Settings – Coordinating Scan Head Control and Laser Control .....	134
7.2.1	Laser Delays .....	134
LaserOn Delay .....	134	
LaserOff Delay .....	135	
7.2.2	Scanner Delays .....	136
Jump Delay .....	136	
Variable Jump Delay .....	137	
Mark Delay .....	138	
Polygon Delay .....	139	
Variable Polygon Delay .....	140	
Loading User-defined Variable Polygon Delays .....	142	
7.2.3	Notes on Optimizing the Delays .....	144
Recommended Sequence .....	144	
Automatic Delay Adjustments .....	144	
Potential Errors when Optimizing the Delays .....	146	
7.2.4	Sky Writing .....	149
Sky Writing Mode 1 .....	149	
Sky Writing Mode 2 .....	150	
Sky Writing Mode 3 .....	151	
Synchronization .....	151	
7.3	Scan Head Control .....	156
7.3.1	Reference System .....	156
7.3.2	Image Field Size and Image Field Calibration .....	156
Typical Image Field .....	157	
Compatibility Modes .....	157	
7.3.3	Virtual Image Field .....	158
Coordinate Transformations in the Virtual Image Field .....	158	
7.3.4	3D Image Field .....	159
Adjustment .....	159	
Checking the z axis Calibration .....	159	
Test for 3-Axis Scan Systems with F-Theta Objective .....	161	
7.3.5	Image Field Correction and Correction Tables .....	162
Field Distortion .....	162	
Field Correction Algorithm .....	163	
Activating Image Field Correction .....	163	
2D and 3D Correction Files .....	163	
Loading Correction Tables .....	164	
Assigning Loaded Correction Tables .....	164	
1to1 Correction Tables .....	166	
Inverse Tables .....	166	
ct5 Correction File Header .....	167	
Converting Correction Files .....	169	
7.3.6	Output Values to the Scan System .....	170
Calculation .....	170	
Value Ranges .....	170	
Precalculation and Diagnosis .....	170	
Clock Overruns .....	171	
7.3.7	Status Monitoring and Diagnostics .....	172
Status Information Returned from the Scan System .....	172	
7.4	Laser Control .....	173
7.4.1	Enabling, Activating and Switching Laser Control Signals .....	173



Signals for "Laser Active" Operation .....	175
Signals for "Laser Standby" Operation .....	175
Automatic Suppression of Laser Control Signals .....	176
7.4.2 Configuring the LASER Connector .....	176
7.4.3 CO <sub>2</sub> Mode .....	177
7.4.4 YAG Modes 1, 2, 3, 5 .....	179
Q-Switch Signal .....	179
FirstPulseKiller Signal .....	179
Differences Between the YAG Modes .....	179
Lamp Current (Laser Power) .....	180
7.4.5 Laser Mode 4 .....	182
7.4.6 Laser Mode 6 .....	183
7.4.7 Softstart Mode (not yet implemented) .....	184
7.4.8 Pulse Picking Laser Mode .....	185
7.4.9 "Automatic Laser Control" .....	186
Position-Dependent Laser Control .....	188
Speed-Dependent Laser Control .....	190
Encoder-Speed-Dependent Laser Control .....	192
Loading and Determining the Nonlinearity Curve .....	193
Vector-Defined Laser Control .....	195
7.4.10 Synchronization of the RTC6 Clock Cycle and an External Clock Signal .....	196
7.5 Marking Dates, Times and Serial Numbers .....	197
7.5.1 Marking the Time and Date .....	197
7.5.2 Marking Serial Numbers .....	197
<b>8 Advanced Functions for Scan Head Control and Laser Control .....</b>	<b>199</b>
8.1 iDRI <sup>E</sup> Functions .....	199
8.1.1 Transfer Protocol .....	199
8.1.2 Configuring the Data Signal Return Behavior of the Scan System .....	200
Setting Data Types .....	200
Reading Out Data .....	200
8.1.3 Monitoring the Positioning .....	201
8.1.4 Selecting the Dynamics Setting (Tuning) .....	202
8.1.5 Jump Mode .....	203
Functional Principle .....	203
Requirements and Activation .....	204
Jump-Length-Dependent Jump Delays .....	204
Determining Jump Delay Values Experimentally .....	205
Notes on Loading Determined Jump Delay Values .....	205
Automatic Determination of the Jump Delay Table .....	207
8.1.6 Configuring the PosAcknowledge Threshold Value .....	208
8.1.7 Configuring the Effective Calibration .....	208
8.1.8 Configuring the Start Behavior .....	209
8.1.9 Fault Diagnosis and Functional Test .....	209
8.2 Coordinate Transformations .....	210
8.3 Online Positioning .....	214
8.3.1 "Local Online Positioning" .....	214
Configuring "Local Online Positioning" .....	215
8.3.2 "Global Online Positioning" .....	217
Configuring "Global Online Positioning" .....	217
8.4 Wobble Mode .....	218
Example Code .....	219
8.4.1 Wobble Shapes – Important Notes on Choosing Appropriate Parameter Values .....	220



"Classic" Wobble Shapes .....	220
"Freely Definable Wobble Shapes" .....	221
8.5 Controlling 2D Scan Systems and 3D Scan Systems .....	222
8.5.1 2D Scan Systems .....	222
8.5.2 3D Scan Systems .....	222
Intended Use .....	222
Connection and Initialization .....	223
3D Commands .....	223
Adjusting Tracking Errors .....	225
Enhanced 3D Correction .....	225
8.5.3 Using Several Correction Tables .....	226
8.6 Processing-on-the-fly .....	227
8.6.1 Intended Use and Initialization .....	227
Overview .....	227
8.6.2 Compensation of Linear Movements .....	228
Correction via Encoder Counters .....	228
Correction via McBSP Interface .....	230
Correction via McBSP Interface with Additional McBSP Input .....	231
Correction via McBSP Interface with Enhanced McBSP Input .....	231
8.6.3 Compensation of Rotary Movements .....	232
Correction via Encoder Counter .....	232
Correction via McBSP Interface .....	233
Correction via McBSP Interface with Additional McBSP Input .....	233
8.6.4 Compensating 2D Motions .....	234
2D Encoder Compensation for xy Positioning Stages .....	234
8.6.5 Deactivating Processing-on-the-fly Correction .....	236
8.6.6 Virtual Image Field with Processing-on-the-fly .....	237
8.6.7 Synchronizing Processing-on-the-fly Applications .....	237
8.6.8 Encoder Resets .....	239
8.6.9 Monitoring Processing-on-the-fly Corrections .....	240
Customer-Defined Monitoring Area .....	241
8.6.10 Tracking Error Compensation of Encoder Values for Processing-on-the-fly Applications – NOT YET IMPLEMENTED .....	242
8.6.11 Processing-on-the-fly Correction for the z Axis .....	243
8.6.12 "Fly Extension" Commands .....	245
Notes on How to Use "Fly Extension" Commands .....	246
8.7 Pixel Output Mode – Marking Pixel Images (Bitmaps) .....	249
8.7.1 Principle of Operation .....	249
8.7.2 RTC6 Commands .....	249
8.7.3 Laser Control .....	251
8.7.4 Synchronization .....	254
8.8 Microvector Commands .....	259
8.9 Timed Vector Commands and Timed Arc Commands .....	260
8.10 Automatic Self-Calibration .....	261
8.10.1 Use for Drift Compensation .....	261
8.10.2 How it Works .....	261
8.10.3 Determining Reference Values .....	263
8.10.4 Calibration During the Application .....	263
Automatic Self-Calibration .....	263
Customer-Specific Calibration .....	264
Supplemental Information about Calibration .....	264
8.11 Camming .....	265



8.12 Time Measurements .....	267
<b>9 Programming Peripheral Interfaces .....</b>	<b>268</b>
<b>9.1 Signal Output .....</b>	<b>268</b>
9.1.1 16-Bit Digital Output Port .....	268
9.1.2 8-Bit Digital Output Port .....	269
9.1.3 2 Bit Digital Output Port .....	269
9.1.4 12-Bit Analog Output Ports .....	269
9.1.5 Controlling Stepper Motors .....	270
Output Signals .....	270
Reference Movements and Position Initialization .....	271
Set-Position Movements .....	271
Querying Signals and Status Values .....	272
Terminating Infinite Movements .....	272
WaitTime Parameter .....	272
9.1.6 RS-232 Interface .....	273
9.1.7 McBSP Interface .....	273
<b>9.2 Signal Input .....</b>	<b>274</b>
9.2.1 16-Bit Digital Input Port .....	274
9.2.2 2-Bit Digital Input Port .....	274
9.2.3 RS-232 Interface .....	274
9.2.4 McBSP Interface .....	274
<b>9.3 Control by External Signals .....</b>	<b>275</b>
9.3.1 Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization .....	275
External Stop .....	275
External Start .....	276
External Start with Track Delay .....	278
Regular (Periodic) External Starts .....	279
9.3.2 Conditional Command Execution .....	281
Example Code (PASCAL) .....	282
9.3.3 Synchronization by Encoder Signals .....	284
Intended Use .....	284
Inputs for External Encoder Signals .....	285
Encoder Simulation .....	285
9.3.4 Synchronization and Online Positioning by McBSP Signals .....	286
<b>9.4 Periodical I/O Signals .....</b>	<b>287</b>
<b>10 RTC6 Commands .....</b>	<b>288</b>
<b>10.1 Overview .....</b>	<b>288</b>
<b>10.1.1 Nomenclature .....</b>	<b>288</b>
Multi-Board Commands .....	288
Normal, Short, Variable and Multiple List Commands .....	288
Undelayed Short List Commands, Delayed Short List Commands .....	289
Multiple List Commands .....	290
<b>10.1.2 Compatibility .....</b>	<b>290</b>
<b>10.1.3 Version Information .....</b>	<b>291</b>
<b>10.1.4 Optional Functions .....</b>	<b>291</b>
<b>10.1.5 Control Commands .....</b>	<b>292</b>
<b>10.1.6 List Commands .....</b>	<b>296</b>
<b>10.1.7 Data Types .....</b>	<b>300</b>
<b>10.2 RTC6 Command Set .....</b>	<b>301</b>
acquire_rtc .....	302



activate_fly_1_axis .....	304
activate_fly_2_axes .....	305
activate_fly_2d .....	307
activate_fly_2d_encoder .....	308
activate_fly_xy .....	310
activate_fly_xy_encoder .....	310
activate_scanahead_autodelays .....	311
activate_scanahead_autodelays_list .....	311
apply_mcbsp .....	312
apply_mcbsp_list .....	313
arc_abs .....	314
arc_abs_3d .....	315
arc_rel .....	316
arc_rel_3d .....	317
auto_cal .....	318
auto_change .....	321
auto_change_pos .....	322
bounce_supp .....	323
camming .....	324
clear_fly_overflow .....	326
clear_io_cond_list .....	327
config_laser_signals .....	328
config_laser_signals_list .....	329
config_list .....	330
control_command .....	332
copy_dst_src .....	341
create_dat_file .....	342
disable_laser .....	343
enable_laser .....	344
eth_assign_card .....	345
eth_assign_card_ip .....	346
eth_boot_dcmd .....	347
eth_check_connection .....	348
eth_convert_ip_to_string .....	349
eth_convert_string_to_ip .....	350
eth_count_cards .....	351
eth_found_cards .....	352
eth_get_card_info .....	353
eth_get_card_info_search .....	354
eth_get_com_timeouts .....	355
eth_get_error .....	356
eth_get_ip .....	357
eth_get_ip_search .....	357
eth_get_last_error .....	358
eth_get_port_numbers .....	360
eth_get_serial_search .....	360
eth_get_static_ip .....	361
eth_max_card .....	362
eth_remove_card .....	363
eth_search_cards .....	364
eth_search_cards_range .....	365
eth_set_com_timeouts .....	366



eth_set_port_numbers .....	367
eth_set_search_cards_timeout .....	368
eth_set_static_ip .....	369
execute_at_pointer .....	370
execute_list .....	371
execute_list_1 .....	371
execute_list_2 .....	371
execute_list_pos .....	372
fly_return .....	373
fly_return_1_axis .....	374
fly_return_2_axes .....	375
fly_return_3_axes .....	376
fly_return_z .....	377
free_rtc6_dll .....	378
get_auto_cal .....	379
get_bios_version .....	379
get_card_type .....	380
get_char_pointer .....	381
get_config_list .....	382
get_counts .....	382
get_dll_version .....	383
get_encoder .....	383
get_error .....	384
get_fly_2d_offset .....	387
get_free_variable .....	387
get_galvo_controls .....	388
get_head_para .....	390
get_head_status .....	391
get_hex_version .....	393
get_hi_data .....	393
get_hi_pos .....	394
get_input_pointer .....	395
get_io_status .....	395
get_jump_table .....	396
get_lap_time .....	396
get_laser_pin_in .....	397
get_last_error .....	398
get_list_pointer .....	398
get_list_serial .....	399
get_list_space .....	399
get_marking_info .....	400
get_master_slave .....	402
get_mcbsp .....	402
get_mcbsp_list .....	403
get_out_pointer .....	404
get_overrun .....	404
get_RTC_mode .....	405
get_RTC_version .....	406
get_scanahead_params .....	407
get_serial .....	408
get_serial_number .....	408
get_standby .....	409



get_startstop_info .....	410
get_status .....	412
get stepper_status .....	414
get_sub_pointer .....	415
get_sync_status .....	416
get_table_para .....	418
get_text_table_pointer .....	419
get_time .....	419
get_transform .....	420
get_value .....	423
get_values .....	425
get_wait_status .....	426
get_waveform .....	427
get_waveform_offset .....	428
get_z_distance .....	429
goto_xy .....	430
goto_xyz .....	431
home_position .....	432
home_position_xyz .....	433
if_cond .....	434
if_fly_x_overflow .....	434
if_fly_y_overflow .....	435
if_fly_z_overflow .....	435
if_not_activated .....	436
if_not_cond .....	437
if_not_fly_x_overflow .....	438
if_not_fly_y_overflow .....	438
if_not_fly_z_overflow .....	439
if_not_pin_cond .....	439
if_pin_cond .....	440
init_fly_2d .....	441
init_rtc6_dll .....	442
jump_abs .....	444
jump_abs_3d .....	445
jump_rel .....	446
jump_rel_3d .....	447
laser_on_list .....	448
laser_on_pulses_list .....	449
laser_signal_off .....	450
laser_signal_off_list .....	450
laser_signal_on .....	451
laser_signal_on_list .....	451
list_call .....	452
list_call_abs .....	454
list_call_abs_cond .....	455
list_call_abs_repeat .....	455
list_call_cond .....	456
list_call_repeat .....	457
list_continue .....	458
list_jump_cond .....	459
list_jump_pos .....	460
list_jump_pos_cond .....	461



list_jump_rel .....	462
list_jump_rel_cond .....	463
list_next .....	464
list_nop .....	464
list_repeat .....	465
list_return .....	466
list_until .....	467
load_auto_laser_control .....	468
load_char .....	470
load_correction_file .....	471
load_disk .....	475
load_fly_2d_table .....	478
load_jump_table .....	479
load_jump_table_offset .....	480
load_list .....	482
load_position_control .....	484
load_program_file .....	486
load_stretch_table .....	489
load_sub .....	491
load_text_table .....	492
load_varpolydelay .....	493
load_z_table .....	495
load_z_table_no .....	496
long_delay .....	498
mark_abs .....	499
mark_abs_3d .....	500
mark_char .....	501
mark_char_abs .....	502
mark_date .....	503
mark_date_abs .....	505
mark_ellipse_abs .....	506
mark_ellipse_rel .....	507
mark_rel .....	508
mark_rel_3d .....	509
mark_serial .....	510
mark_serial_abs .....	512
mark_text .....	513
mark_text_abs .....	514
mark_time .....	515
mark_time_abs .....	517
master_slave_config .....	518
mcbsp_init .....	519
mcbsp_init_spi .....	520
measurement_status .....	521
micro_vector_abs .....	522
micro_vector_abs_3d .....	523
micro_vector_rel .....	524
micro_vector_rel_3d .....	525
move_to .....	525
number_of_correction_tables .....	526
para_jump_abs .....	527
para_jump_abs_3d .....	528



para_jump_rel .....	529
para_jump_rel_3d .....	530
para_laser_on_pulses_list .....	531
para_mark_abs .....	533
para_mark_abs_3d .....	534
para_mark_rel .....	535
para_mark_rel_3d .....	536
park_position .....	537
park_position_1_axis .....	539
park_position_2_axes .....	540
park_return .....	541
park_return_1_axis .....	543
park_return_2_axes .....	544
pause_list .....	545
periodic_toggle .....	546
periodic_toggle_list .....	547
quit_loop .....	548
range_checking .....	549
read_abc_from_file .....	551
read_analog_in .....	552
read_encoder .....	553
read_image_eth .....	554
read_io_port .....	555
read_io_port_buffer .....	556
read_io_port_list .....	557
read_mcbsp .....	558
read_multi_mcbsp .....	559
read_status .....	560
read_user_data .....	562
release_rtc .....	563
release_wait .....	564
reset_error .....	565
restart_list .....	566
rs232_config .....	566
rs232_read_data .....	567
rs232_write_data .....	568
rs232_write_text .....	568
rs232_write_text_list .....	569
rtc6_count_cards .....	570
save_and_restart_timer .....	571
save_disk .....	572
select_char_set .....	574
select_cor_table .....	575
select_cor_table_list .....	577
select_rtc .....	578
select_serial_set .....	580
select_serial_set_list .....	580
send_user_data .....	581
set_angle .....	582
set_angle_list .....	584
set_auto_laser_control .....	585
set_auto_laser_params .....	589



set_auto_laser_params_list .....	589
set_char_pointer .....	590
set_char_table .....	591
set_control_mode .....	592
set_control_mode_list .....	594
set_default_pixel .....	594
set_default_pixel_list .....	595
set_defocus .....	596
set_defocus_list .....	597
set_defocus_offset .....	597
set_defocus_offset_list .....	598
set_delay_mode .....	599
set_delay_mode_list .....	600
set_dsp_mode .....	601
set_ellipse .....	602
set_encoder_speed .....	603
set_encoder_speed_ctrl .....	604
set_end_of_list .....	605
set_eth_boot_control .....	606
set_eth_boot_timeout .....	606
set_extstartpos .....	607
set_extstartpos_list .....	607
set_ext_start_delay .....	608
set_ext_start_delay_list .....	609
set_firstpulse_killer .....	609
set_firstpulse_killer_list .....	610
set_fly_1_axis .....	611
set_fly_2_axes .....	612
set_fly_2d .....	614
set_fly_3_axes .....	616
set_fly_limits .....	618
set_fly_limits_z .....	619
set_fly_rot .....	620
set_fly_rot_pos .....	621
set_fly_tracking_error .....	622
set_fly_x .....	623
set_fly_x_pos .....	624
set_fly_y .....	625
set_fly_y_pos .....	626
set_fly_z .....	627
set_free_variable .....	628
set_free_variable_list .....	628
set_hi .....	629
set_input_pointer .....	630
set_io_cond_list .....	631
set_jump_mode .....	632
set_jump_mode_list .....	635
set_jump_speed .....	636
set_jump_speed_ctrl .....	637
set_jump_table .....	638
set_laser_control .....	639
set_laser_delays .....	642



set_laser_mode .....	643
set_laser_off_default .....	643
set_laser_pin_out .....	644
set_laser_pin_out_list .....	644
set_laser_power .....	645
set_laser_pulses .....	646
set_laser_pulses_ctrl .....	647
set_laser_timing .....	648
set_list_jump .....	649
set_mark_speed .....	650
set_mark_speed_ctrl .....	651
set_matrix .....	652
set_matrix_list .....	654
set_max_counts .....	655
set_mcbsp_freq .....	655
set_mcbsp_global_matrix .....	656
set_mcbsp_global_matrix_list .....	656
set_mcbsp_global_rot .....	657
set_mcbsp_global_rot_list .....	657
set_mcbsp_global_x .....	658
set_mcbsp_global_x_list .....	658
set_mcbsp_global_y .....	659
set_mcbsp_global_y_list .....	659
set_mcbsp_in .....	660
set_mcbsp_in_list .....	662
set_mcbsp_matrix .....	663
set_mcbsp_matrix_list .....	664
set_mcbsp_out .....	665
set_mcbsp_out_ptr .....	666
set_mcbsp_rot .....	667
set_mcbsp_rot_list .....	667
set_mcbsp_x .....	668
set_mcbsp_x_list .....	668
set_mcbsp_y .....	669
set_mcbsp_y_list .....	669
set_multi_mcbsp_in .....	670
set_multi_mcbsp_in_list .....	673
set_n_pixel .....	674
set_offset .....	675
set_offset_list .....	675
set_offset_xyz .....	676
set_offset_xyz_list .....	677
set_pause_list_cond .....	678
set_pause_list_not_cond .....	679
set_pixel .....	680
set_pixel_line .....	681
set_pixel_line_3d .....	684
set_port_default .....	685
set_port_default_list .....	686
set_pulse_picking .....	687
set_pulse_picking_length .....	687
set_pulse_picking_list .....	688



set_qswitch_delay .....	688
set_qswitch_delay_list .....	689
set_rot_center .....	690
set_rot_center_list .....	690
set_RTC4_mode .....	691
set_RTC5_mode .....	692
set_RTC6_mode .....	693
set_scale .....	694
set_scale_list .....	694
set_scanahead_laser_shifts .....	695
set_scanahead_laser_shifts_list .....	695
set_scanahead_line_params .....	695
set_scanahead_line_params_list .....	695
set_scanahead_params .....	695
set_scanahead_speed_control .....	695
set_scanner_delays .....	696
set_serial .....	697
set_serial_step .....	697
set_serial_step_list .....	698
set_sky_writing .....	698
set_sky_writing_limit .....	699
set_sky_writing_limit_list .....	699
set_sky_writing_list .....	699
set_sky_writing_mode .....	700
set_sky_writing_mode_list .....	701
set_sky_writing_para .....	702
set_sky_writing_para_list .....	704
set_softstart_level .....	705
set_softstart_level_list .....	705
set_softstart_mode .....	706
set_softstart_mode_list .....	706
set_standby .....	707
set_standby_list .....	708
set_start_list .....	708
set_start_list_1 .....	709
set_start_list_2 .....	709
set_start_list_pos .....	710
set_sub_pointer .....	711
set_text_table_pointer .....	712
set_timelag_compensation .....	713
set_trigger .....	714
set_trigger4 .....	721
set_vector_control .....	722
set_verify .....	724
set_wait .....	725
set_wobbel .....	726
set_wobbel_control .....	728
set_wobbel_direction .....	730
set_wobbel_mode .....	731
set_wobbel_mode_phase .....	733
set_wobbel_offset .....	734
set_wobbel_vector .....	735



simulate_encoder .....	738
simulate_ext_start .....	739
simulate_ext_start_ctrl .....	741
simulate_ext_stop .....	742
spot_distance .....	743
spot_distance_ctrl .....	743
start_loop .....	744
stepper_abs .....	745
stepper_abs_list .....	746
stepper_abs_no .....	746
stepper_abs_no_list .....	747
stepper_control .....	748
stepper_control_list .....	749
stepper_disable_switch .....	749
stepper_enable .....	750
stepper_enable_list .....	750
stepper_init .....	751
stepper_rel .....	753
stepper_rel_list .....	753
stepper_rel_no .....	754
stepper_rel_no_list .....	754
stepper_wait .....	755
stop_execution .....	756
stop_list .....	756
stop_trigger .....	757
store_encoder .....	757
store_program .....	758
store_timestamp_counter .....	759
store_timestamp_counter_list .....	759
sub_call .....	760
sub_call_abs .....	761
sub_call_abs_cond .....	761
sub_call_abs_repeat .....	762
sub_call_cond .....	762
sub_call_repeat .....	763
switch_iport .....	764
sync_slaves .....	765
time_control_eth .....	767
time_fix .....	768
time_fix_f .....	768
time_fix_f_off .....	769
time_update .....	770
timed_arc_abs .....	771
timed_arc_rel .....	772
timed_jump_abs .....	773
timed_jump_abs_3d .....	774
timed_jump_rel .....	775
timed_jump_rel_3d .....	776
timed_mark_abs .....	777
timed_mark_abs_3d .....	778
timed_mark_rel .....	779
timed_mark_rel_3d .....	780



timed_para_jump_abs .....	781
timed_para_jump_abs_3d .....	782
timed_para_jump_rel .....	783
timed_para_jump_rel_3d .....	784
timed_para_mark_abs .....	785
timed_para_mark_abs_3d .....	786
timed_para_mark_rel .....	787
timed_para_mark_rel_3d .....	788
transform .....	789
uart_config .....	792
upload_transform .....	793
verify_checksum .....	795
wait_for_1_axis .....	796
wait_for_2_axes .....	798
wait_for_encoder .....	800
wait_for_encoder_in_range .....	801
wait_for_encoder_in_range_mode .....	802
wait_for_encoder_mode .....	803
wait_for_mcbsp .....	805
wait_for_timestamp_counter .....	806
write_8bit_port .....	807
write_8bit_port_list .....	807
write_abc_to_file .....	808
write_da_1 .....	809
write_da_1_list .....	809
write_da_2 .....	810
write_da_2_list .....	810
write_da_x .....	811
write_da_x_list .....	812
write_hi_pos .....	813
write_image_eth .....	814
write_io_port .....	815
write_io_port_list .....	815
write_io_port_mask .....	816
write_io_port_mask_list .....	816
10.3 Unsupported RTC4/RTC5 Commands .....	817
<b>11 Demo Programs .....</b>	<b>819</b>
<b>12 Troubleshooting .....</b>	<b>820</b>
<b>13 Customer Service .....</b>	<b>822</b>
13.1 Servicing and Repairs .....	822
13.2 Warranty .....	822
13.3 Contacting SCANLAB .....	822
13.4 Product Disposal .....	822
<b>14 Technical Specifications of the RTC6 PCIe Board .....</b>	<b>823</b>
14.1 Compliance with EC Guidelines for Electromagnetic Compatibility (EMC) .....	827
14.2 Compliance with FCC Rules .....	827



<b>15 Appendix A: The RTC6 Ethernet Board</b> .....	<b>828</b>
15.1 Product Overview .....	828
15.1.1 RTC6 Ethernet Board vs. RTC6 PCIe Board – Usage and Comparison .....	828
15.1.2 System Requirements .....	829
Hardware .....	829
Software .....	829
15.1.3 Options .....	829
15.1.4 Labeling .....	829
15.1.5 Type Identification .....	829
15.1.6 Unpacking Instructions and Typical Scope of Delivery .....	830
15.1.7 Delivered Software .....	830
15.1.8 Accessories for the RTC6 PCIe Board .....	830
15.1.9 Supplementary Software .....	830
15.2 Layout, Interfaces, Jumper Settings .....	831
15.2.1 Layout – Upper Side .....	831
15.2.2 Layout – Lower Side .....	832
15.2.3 Dimensions and Connector Positions .....	833
15.2.4 LASER Socket Connector .....	834
Laser Output Signals .....	834
External Control Signals .....	834
BUSY Status .....	834
2-Bit Digital Input and 2-Bit Digital Output .....	834
12-Bit Analog Output .....	835
Input and Output Wiring .....	835
15.2.5 SCANHEADS Socket Connector .....	837
15.2.6 POWER Socket Connector .....	839
15.2.7 ETH Connector .....	839
15.2.8 SPI/ANA/UART Socket Connector .....	840
Analog Inputs .....	840
McBSP Interface .....	840
RS-232 Interface .....	840
15.2.9 STEPPER Socket Connector .....	841
15.2.10 MOF Socket Connector .....	841
Encoder Inputs .....	842
External Control Signals .....	842
BUSY OUT Status .....	842
15.2.11 EXTENSION 1 Socket Connector .....	842
Configuring the Output Signal Level .....	843
16-Bit Digital Output and 16-Bit Digital Input .....	843
Synchronization of Data Acquisition .....	843
BUSY Status .....	843
15.2.12 EXT. 2 Socket Connector .....	843
Configuration by Solder Jumpers .....	844
8-Bit Digital Output Port .....	844
15.2.13 Master Socket Connector, Slave Socket Connector .....	845
15.2.14 Jumper Settings .....	846
Solder Jumper Field A – Configuring the Output Signal Level at the EXTENSION 1 Socket Connector .....	846
Solder Jumper Field B – Configuring pin (09) of the EXT. 2 Socket Connector .....	847
Solder Jumper Field C – Configuring pin (08) of the EXT. 2 Socket Connector .....	848
Jumper Field 'Force DHCP' .....	849
15.2.15 Real-Time Clock .....	849



15.3 Installation and Operation .....	850
15.3.1 Hardware Installation .....	850
15.3.2 Software Installation .....	850
15.3.3 Connecting to a Network .....	850
15.4 Notes on Migrating Existing and Programming New RTC6 User Programs .....	851
15.4.1 Finding RTC6 Ethernet Boards in the Network and Querying their Properties .....	851
15.4.2 Example Code (C++): Initialization Covering RTC6 Ethernet Boards .....	852
15.5 RTC6 Ethernet Board Commands and Functions .....	855
15.5.1 Notes on Working with IP Addresses .....	855
15.5.2 About Searching RTC6 Ethernet Boards .....	855
15.5.3 About the RTC6 Board Management .....	856
15.5.4 Checking the Connection to the RTC6 Ethernet Board .....	857
15.5.5 Command Set for the RTC6 Ethernet Board .....	857
15.6 Safe Startup and Shutdown Sequences .....	858
15.7 Standalone Functionality .....	859
15.7.1 Upgrading to RTC6BIOSETH_26 .....	860
15.7.2 Preparing the "Standalone Basic State" .....	860
15.7.3 Preparing the "Standalone Full State" .....	861
15.7.4 Boot Image .....	862
Creating a Boot Image on the PC .....	862
Copying Boot Image to Board(s) .....	862
Procedure after an Transmission Abortion .....	862
15.7.5 Control Commands Allowed for Automatic Booting .....	863
15.7.6 Automatic Booting - Process in Detail .....	865
15.8 Technical Specifications for the RTC6 Ethernet Board .....	866
15.9 Compliance with EC Guidelines for Electromagnetic Compatibility (EMC) .....	871
15.10 Compliance with FCC Rules .....	871
16 Appendix B: The UPM Extension Board .....	872
17 Appendix C: The RTC6 Ethernet Plug-in + Box .....	875
17.1 Mounting .....	878
17.2 Cabling .....	878
17.3 Installation and Operation .....	878
18 Glossary and Abbreviations .....	879
19 Change Index .....	881



## 1 Introduction

This manual describes the available SCANLAB RTC6 boards and their usage for synchronous control of scan systems, lasers and peripheral equipment.

The main chapters of this manual use the RTC6 PCIe Board (= "RTC6 PCI Express Board") to exemplify. For a product overview, see [Chapter 2 "Product Overview", page 27](#).

Other RTC6 board variants are described in the Appendix:

- RTC6 Ethernet Board, see [Chapter 15 "Appendix A: The RTC6 Ethernet Board", page 828](#)

The manual is a part of the product. Read these instructions carefully before you proceed with installing and operating.

In particular, observe all safety guidelines in this manual. If there are any questions regarding the contents of this manual, contact SCANLAB, see [Chapter 1.1 "Manufacturer"](#).

Keep the manual available for servicing, repairs and product disposal. This manual should accompany the product if ownership changes hands.

SCANLAB reserves the right to update this operating manual at any time and without notification.

This manual refers to:

- RTC6 Software Package V1.7.0

DLL file for 32 bit user programs <sup>(a)</sup>	<a href="#">RTC6DLL.dll</a>	Version 618 (DLL 618) <sup>(b)</sup>
DLL file for 64 bit user programs <sup>(a)</sup>	<a href="#">RTC6DLLx64.dll</a>	Version 618 (DLL 618) <sup>(b)</sup>
Program file for the PCIe-DSP	<a href="#">RTC6OUT.out</a>	Version 618 (OUT 618) <sup>(b)</sup>
Program file for the Eth-DSP	<a href="#">RTC6ETH.out</a>	Version 618 (OUT 618) <sup>(b)</sup>
Firmware file for the FPGA	<a href="#">RTC6RBF.rbf</a>	Version 623 (RBF 623) <sup>(b)</sup>
Binary auxiliary file	<a href="#">RTC6DAT.dat</a>	Version 603 (DAT 603) <sup>(b)</sup>

(a) Software for laser-scan processes, which controls RTC6 boards based on this DLL file is consistently denoted as "user program" in this manual.

(b) Abbreviated version in this manual.

The version numbers of the supplied **RTC6 DLL** and **RTC6 files** are indicated in the names of the corresponding zip files, see [Section "Folder RTC6 Tools", page 25](#).

To identify the version numbers of your files after installation, use `get_dll_version`, `get_hex_version` and `get_RTC_version`.

### 1.1 Manufacturer

SCANLAB GmbH  
Siemensstr. 2a  
82178 Puchheim  
Germany  
Tel. +49 (89) 800 746-0  
Fax: +49 (89) 800 746-199  
[info@scanlab.de](mailto:info@scanlab.de)  
[www.scanlab.de](http://www.scanlab.de)



## 1.2 Labeling

The serial number of the RTC6 PCIe Board is printed on a label attached to the board.

The article number and configuration of the board are described in the packaging list, see [Chapter 2.3 "Options", page 30](#), and [Chapter 2.4 "Jumper Settings and Type Designations", page 32](#).

## 1.3 Unpacking Instructions and Typical Scope of Delivery

- (1) Carefully remove the RTC6 PCIe Board from the package.
- (2) Keep the packaging, including the antistatic bag the RTC6 PCIe Board is delivered in, so that in case of repair the board can be properly repackaged and returned to SCANLAB.
- (3) Also remove all other articles from the package. Check that all parts have been delivered. Refer to the corresponding packaging list.  
The scope of delivery typically includes an RTC6 PCIe Board and a data CD (with the RTC6 Software Package, see below, and this manual). Possibly additional components such as data cables or converters are also contained, see [Chapter 2.5 "Accessories for the RTC6 PCIe Board", page 36](#).

### 1.3.1 Delivered RTC6 Software Package

The delivered RTC6 Software Package contains the RTC6 board driver<sup>(1)</sup> for the 32-bit and 64-bit versions of the operating systems Microsoft Windows 10, 8, 7. *Windows XP and Windows Vista are not supported!* Observe the safety notice on legacy RTC board drivers in [Chapter 2.2.2 "Software", page 29](#).

The data CD contains all files as unzipped versions. The complete RTC6 Software Package is also delivered zipped for easy identification and management of different software versions:

- RTC6\_Software\_Package\_Rev.n.n.n.<Datum>.zip

The content of the RTC6 Software Package is as follows:

- Readme.txt  
[Description in English](#)
- Liesmich.txt  
[Description in German](#)
- IMPORTANT\_RELEASE\_NOTES.txt  
[Bilingual English and German](#)
- RTC6\_Release\_Notes\_YYYY\_MM\_DD.pdf  
[Mentions known restrictions. Bilingual English and German](#)

#### Folder Correction Files

- Cor\_1to1.ct5  
[correction file<sup>\(2\)</sup>](#)

#### Folder CorrectionFileConverter

- The Win32-based program CorrectionFileConverter.exe converts RTC4 correction files \*.ctb into RTC5/RTC6 correction files \*.ct5 and vice versa. The corresponding manual is supplied in English and German.

(1) The RTC6 board driver is not required for RTC6 Ethernet Boards.

(2) Additional correction files (D2\_XXX.CT5, D3\_YYY.CT5) and Readme files are not part of the RTC6 Software Package.



## Folder Demo Files

- Currently not available.

## Folder HPGL

- Hpgl.exe is a Win32-based HPGL-to-RTC6 converter. See also [Chapter 5.6 "Functionality Test", page 81](#).
- The folder contains some \*.plt files (Hewlett Packard HPGL format (vector graphic plotter files) for test purposes).
- Hpgl.exe needs [RTC6 files](#) and the [RTC6DLL.dll](#) in the same directory.

## Folder iSCANcfg

- iSCANcfg.exe is a Win32-based diagnosis and configuration program for iDRIVE scan systems<sup>(1)</sup>. RTC4, RTC5 and RTC6 (PCI/PCIe as well as Ethernet variants) are supported.
- Needs [RTC6 files](#) and the [RTC6DLL.dll](#) in the same directory and in addition RtcHalDLL.dll. The corresponding manual is supplied in English (Manual\_iSCANcfg\_v1-7.pdf) and German (Handbuch\_iSCANcfg\_v1-7.pdf).

## Folder RTC6 Driver

(RTC6 board driver for Windows)

- RTC6DRV.sys, RTC6DRVx64.sys, RTC6DRVx86.sys  
**RTC6 board driver files**
- RTC6DRV.inf  
**Installation file (setup information)**
- RTC6DRVx64.cat, RTC6DRVx86.cat  
**Security catalog files**
- amd64/WdfCoInstaller01009.dll,  
x86/WdfCoInstaller01009.dll  
**Installation assistant help files**
- AfterInstallation/ScanlabClassChecker.cmd,  
**Security installation script with description in ReadMe\_ScanlabClassChecker.pdf**

## Folder RTC6 Tools

- **RTC6 DLL**
  - RTC6DLL.dll  
**Win32-based RTC6 dynamic link library**
  - RTC6DLLx64.dll  
**Win64-based RTC6 dynamic link library**
- **RTC6 files**
  - RTC6OUT.out  
**Program file for the PCIe-DSP**
  - RTC6ETH.out  
**Program file for the Eth-DSP**
  - RTC6RBF.rbf  
**Firmware file for the FPGA**
  - RTC6DAT.dat  
**Binary auxiliary file**
- **Utility files for C, C++ and C#**
  - RTC6expl.c  
**C functions for RTC6 DLL handling for explicit linking**
  - RTC6expl.h  
**C function prototypes of the RTC6 for explicit linking of the RTC6 DLL**
  - RTC6DLL.lib  
**Visual C++ import libraries for implicit linking of the RTC6 DLL for Win32-based applications**
  - RTC6DLLx64.lib  
**Visual C++ import libraries for implicit linking of the RTC6 DLL for Win64-based applications**
  - RTC6impl.h  
**C function prototypes of the RTC6 for implicit linking of the RTC6 DLL**
  - RTC6impl.hpp  
**C++ function prototypes of the RTC6 for implicit linking of the RTC6 DLL**
  - RTC6Wrap.cs  
**Import declarations of the wrapper class for implicit linking in C#**
- **Utility file for Delphi**
  - RTC6Import.pas  
**Import declarations for Delphi**

(1) See glossary entry on [page 879](#).



Differing versions of **RTC6 files** and **RTC6 DLL** cannot be arbitrarily combined with another. Therefore, for easy identification of the versions, the following **zip** files are provided (each includes a text file with version and compatibility information):

- RTC6DAT\_<current DAT version number>.zip
- RTC6DLL\_<current DLL version number>.zip  
**(includes DLLs and related utility files)**
- RTC6OUT\_<current OUT version number>.zip  
**(includes RTC6OUT.out and RTC6ETH.out)**
- RTC6RBF\_<current RBF version number>.zip

### Folder RTC6 Tools

- RTC6conf.exe  
**"RTC6 Configuration Tool"**  
with the following main features
  - for RTC6 PCIe Boards and  
RTC6 Ethernet Boards: viewing board information (serial number, enabled options, **BIOS** version), enabling options (requires an auxiliary file which is to be purchased from SCANLAB), upgrading **BIOS**
  - for RTC6 Ethernet Boards additionally: searching boards in (configurable) network segments (subnets), show and change network settings for static IP configuration.
- RTC6conf.pdf  
**Description of the use of RTC6conf.exe**
- RTC6BIOSOUT\_23.out  
**DSP program file for RTC6 PCIe Board BIOS**  
(version 0x23)
- RTC6BIOSETH\_26.out  
**DSP program file for RTC6 Ethernet Board BIOS**  
(version 0x26)
- The **RTC6 files** and **RTC6 DLL** of this RTC6 Software Package
  - RTC6DAT.dat
  - RTC6DLL.dll
  - RTC6ETH.out
  - RTC6OUT.out
  - RTC6RBF.rbf

- SleepMode.cmd  
Script to deactivate *all* Windows sleep and hibernate modes.
- ReadMe\_SleepMode.pdf  
**Description of the use of SleepMode.cmd**
- RTC6\_Software\_RevisionHistory\_<Date>\_<Rev>.pdf  
**Description of RTC6 Software Package changes in English**
- RTC6\_Software\_Aenderungshistorie\_<Date>\_<Rev>.pdf  
**Description of RTC6 Software Package changes in German**



## 2 Product Overview

### 2.1 Intended Use

The SCANLAB RTC6 PCIe Board and its associated RTC6 Software Package is intended for synchronous real-time control of scan systems, lasers and peripheral equipment by a Windows PC with a PCIe bus interface.

RTC6 boards are available in different hardware variants, [Chapter 1 "Introduction", page 23](#).

The delivered DLLs ([RTC6DLL.dll](#), [RTC6DLLx64.dll](#)) provide an extensive command set for control. This allows a quick and flexible software development for laser-scan processes.

The RTC6 PCIe Board is equipped with a fast digital signal processor ([DSP](#)). During execution of control commands it also handles more complex signal processing, such as simultaneous control of two scan systems or coordinate transformations.

Moreover, you can store controlling commands (= list commands) on the RTC6 PCIe Board and start their execution at a later time. Command execution by the can then take place independently of the host PC. This makes it possible to meet the stringent demands of real-time control for scan systems, lasers and peripheral equipment, even if the PC must simultaneously respond to other tasks such as machine control and network communication.

The interface to the scan system, together with the associated software commands, allows bidirectional communication with the scan system, thereby providing both control and monitoring capabilities for the scan system.

With the RTC6 PCIe Board, commonly used laser types can be controlled. To control lasers, the supplies interfaces that output laser control signals and are software-configurable for each application's requirements.

Users can choose among different laser modes and set the signal parameters (for example, the signal level – active-HIGH or active-LOW) or the output frequency to a suitable value.

For controlling peripheral equipment and incorporating external control signals, the RTC6 PCIe Board provides a range of interfaces (for example, a 16-bit digital input port, a 16-bit digital output port, two 12-bit analog output ports and an RS-232 interface, see [Chapter 4 "RTC6 PCIe Board – Layout and Interfaces", page 53](#)) and associated software commands.

As many RTC6 PCIe Board as the PCIe bus permits can be operated simultaneously in a single PC. The total number of RTC6 PCIe Boards and RTC6 Ethernet Boards must not exceed 255.

Moreover, the [RTC6 DLL](#) allows multi-threading as well as multi-processing; therefore several applications (user programs) can be used simultaneously.

No board can be simultaneously used by multiple applications. Multiple threads of one user program can use the same board, but can not send commands to it at the same time. The [RTC6 DLL](#) serializes these accesses automatically.

The RTC6 PCIe Board is available in various configurations, see [Chapter 2.3 "Options", page 30](#), and [Chapter 2.4 "Jumper Settings and Type Designations", page 32](#).

The RTC6 PCIe Board interfaces are described on [Chapter 4 "RTC6 PCIe Board – Layout and Interfaces", page 53](#), installation and start-up on [Chapter 5 "Installation and Start-Up", page 78](#), and programming on [Chapter 6 "Developing RTC6-User Programs", page 83](#). Individual command descriptions are listed beginning with [Chapter 10 "RTC6 Commands", page 288](#).

The technical specifications of the RTC6 PCIe Board are summarized on [Chapter 14 "Technical Specifications of the RTC6 PCIe Board", page 823](#).



## Caution!

- Do not operate the RTC6 PCIe Board outside of the PC.
- The RTC6 PCIe Board is intended only for industrial usage. It is designed to be incorporated in machines (normally laser systems). It does *not* meet all criteria of ready-to-use products. Do not operate the RTC6 PCIe Board unless it is incorporated in a machine which itself complies with the regulations of all applicable standards and directives (of the country concerned). It is *not* suitable to be used as toy, in household or under unfavourable environment conditions (for example, in the open). Appropriate precautions to avoid such unforeseeable misapplications must be taken by users.
- Installation and operation must only be performed by trained specialists, among other things knowledgeable in the safe and proper use of electrical devices. Only carry out installation and maintenance work when supply voltages and lasers have been switched off.
- The RTC6 PCIe Board is a class A product. In a domestic environment this product may cause radio interferences in which case the user may be required to take adequate measures.

## 2.2 System Requirements

### 2.2.1 Hardware

The RTC6 PCIe Board requires a Windows PC with a PCIe bus interface and at least one free PCIe slot.

RTC6 PCIe Boards intended for synchronized master/slave operation should (recommended) be installed in adjacent PCIe slots.

### 2.2.2 Software

To operate the RTC6 PCIe Board, RTC6 board driver and **RTC6 DLL** files for Microsoft Windows operating systems must be used. These are included in the scope of delivery (RTC6 Software Package). For the supported Windows versions, see [Chapter 1.3.1 "Delivered RTC6 Software Package", page 24](#).

The RTC6 board driver supports the plug and play capability of the RTC6 PCIe Board as well as the simultaneous operation of up to 255 RTC6 PCIe Boards. The **RTC6 DLL** files contain the command set to control laser scan systems.

#### Notice!

- The RTC6 PCIe Board does not support power-saving modes, that switch off power to the PCIe bus. Accordingly, you must disable standby or sleep modes of the operating system. See also following [Section "Notes", page 29](#).

#### Notice!

- If on your PC an WDM technology-based RTC3/4/5 board driver
  - is yet installed or
  - was installed and has been removed or
  - you are not sure in this regard:
    - (1) Install the RTC6 board driver.
    - (2) Run `ScanlabClassChecker.cmd` as Administrator (part of the delivered RTC6 Software Package; see there also the background information in `ReadMe_ScanlabClassChecker.pdf`). Step 2 can be skipped on brand new PCs on which an RTC board driver never has been installed.

#### Notes

- RTC6 Software Packages only contain WDF<sup>(1)</sup> drivers (version 6.1.7600.16385). WDM<sup>(2)</sup> drivers (legacy technology) are not contained.
- If `init_RTC6_dll` is called, then the RTC6 board driver prevents automatic activation of standby or sleep modes (continuously until the next system restart). This enables RTC6 PCIe Boards to continue processing lists autonomously even when the initiating user program has already been terminated. However, standby or sleep modes cannot be prevented if triggered manually or by discharged batteries. Afterward, loaded lists and other settings of the RTC6 PCIe Board are lost. After a wake-up, the board might no longer be correctly addressable. Users themselves must prevent standby or sleep modes prior to the first call of `init_RTC6_dll`. Before calling `init_RTC6_dll` for the first time, make sure that standby or sleep modes of the operating system are deactivated. The script `SleepMode.cmd` which is contained in the RTC6 Software Package (under RTC6 Tools) deactivates all sleep and hibernation modes, see `ReadMe_SleepMode.pdf`.

(1) Windows Driver Framework

(2) Windows Driver Model

## 2.3 Options

The RTC6 PCIe Board can be equipped with options (features) which simple basic versions do not have. For new cards, the options ordered are enabled/installed at the SCANLAB factory on delivery. For information on retrofitting already delivered boards, see [Section "Notes", page 31](#).

**RTC6 commands**

To query which options are actually enabled on a certain board, [get\\_RTC\\_version](#) is used.

The following options are available for RTC6 PCIe Boards:

- **Option Processing-on-the-fly**

Allows that a Processing-on-the-fly correction can be activated, see [Chapter 8.6 "Processing-on-the-fly", page 227](#).

- **Option "3D"**

- **Controlling a 3-Axis Scan System**

Allows that an RTC6 PCIe Board can synchronously control a third axis (z axis, for example, a varioSCAN as a dynamic focus unit) along with the scan system's x and y axes (usually two galvanometer scanners) by its two scan head connectors, see [Chapter 8.5.2 "3D Scan Systems", page 222](#).

- **Option "Second Scan Head Control"**

- **Controlling Two Scan Systems Simultaneously**

Allows that an RTC6 PCIe Board can simultaneously control two xy-scan systems via its two scan head connectors, see also [Chapter 4.5.1 "Scan Head Connectors and Transfer Protocol", page 56](#) and [Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 222](#).

- **Option "SCA-na"**

- **Controlling Scan Systems with SCA-nahead Servo Control**

Allows that scan systems equipped with SCA-nahead servo control (for example, excelliSCAN series) can (also) be controlled. For further information, see the "excelliSCAN scan heads – Functional Principle of SCA-nahead Servo Control and Operation by RTC6 Boards" manual.

- **Option "UFP"**

- **Pixel Output Modes with pixel output frequencies more than 800 kHz<sup>(1)</sup>**

Allows that pixel output frequencies of > 800 kHz...3,2 MHz can be achieved in certain pixel output modes, see also [set\\_pixel\\_line](#).

- **Option "syncA"**

- **Support of the syncAXIS control Software**

Allows that SCANLAB syncAXIS control software can be used (board and software are part of SCANLAB's scope of delivery for XL SCAN systems).

- **Option "DC/DC Converter"**

These boards are equipped with an extra DC/DC converter (optoelectronic coupler) ex works. Therefore, the laser control signals LASERON, LASER1 and LASER2 at the LASER connector and EXTENSION 2 socket connector are galvanically decoupled from the PC ground<sup>(2)</sup>, see [Section "Laser Output Signals", page 62](#), and [Section "Laser Output Signals", page 70](#).

(1) UFP, Ultra Fast Pixel Mode.

(2) With RTC6 Ethernet Boards: ground at POWER connector.



## Notes

- Each RTC6 PCIe Board article number indicates which of the options above are enabled and/or installed. These are stated in the packing list. The naming there is as follows:
  - "Fly" for [Option Processing-on-the-fly](#)
  - "3D" for [Option "3D"](#)
  - "SSHC" for [Option "Second Scan Head Control"](#)
  - "SCANa" for [Option "SCANa"](#)
  - "UFPM" for [Option "UFPM"](#)
  - "syncA" for [Option "syncA"](#)
  - "DCDC" for [Option "DC/DC Converter"](#)
- To query which options are actually enabled on a board, [get\\_RTC\\_version](#) is used.
- If you need one or more options which are not activated out of the factory on your RTC6 PCIe Board: order an corresponding licence file from SCANLAB, specifying the serial number and the options you want. Then, to apply the upgrade to the board, specify it in [RTC6conf.exe](#).
- For retrofitting your RTC6 PCIe Board with the extra DC/DC converter (optoelectronic coupler) you need to send it to SCANLAB.
- For optical data transfer between the RTC6 PCIe Board and the scan system, solely a suitable data cable is required, see [Chapter 4.5.3 "Data Cables \(Accessories\)", page 60](#). Neither the RTC6 PCIe Board side nor the scan system side requires a special optical data interface for that.
- With software package version ≤ V1.4.4, the "Processing-on-the-fly" functionality cannot be used for scan systems with SCANAhead control.

## 2.4 Jumper Settings and Type Designations

SCANLAB ships RTC6 PCIe Boards in various jumper configurations. Jumpers are connections which are either open or closed. The *factory* solder jumper configuration of an RTC6 PCIe Board can be identified by its article number.

In addition, a three-digit type code scheme is used, for example,

- “TYPE 000”:
  - No signals (that is, all solder jumpers are open)
- “TYPE 124”:
  - 5 V output signal level at the EXTENSION 1 socket connector
  - DATA7 at pin 15 of the EXTENSION 2 socket connector
  - LATCH at pin 17 of the EXTENSION 2 socket connector

Digit 1	Refers to the output signal level at the EXTENSION 1 socket connector <sup>(1)</sup> .
	=0: No signals. =1: 5 V. =2: 3,3 V.
Digit 2	Refers to pin 15 of the EXTENSION 2 socket connector <sup>(3)</sup> .  =0: No signals. =1: +5 V. =2: DATA7. =3: GROUND.
Digit 3	Refers to pin 17 of the EXTENSION 2 socket connector <sup>(2)</sup> .  =0: No signals. =1: +5 V. =2: DATA7. =3: GROUND. =4: LATCH.

At a later time users can reconfigure solder jumpers by using a soldering iron. The assignment of a desired signal is done by closing or opening (applying or removing solder or zero-ohm resistor) of corresponding solder jumpers as described in the following<sup>(1)(2)(3)</sup>.

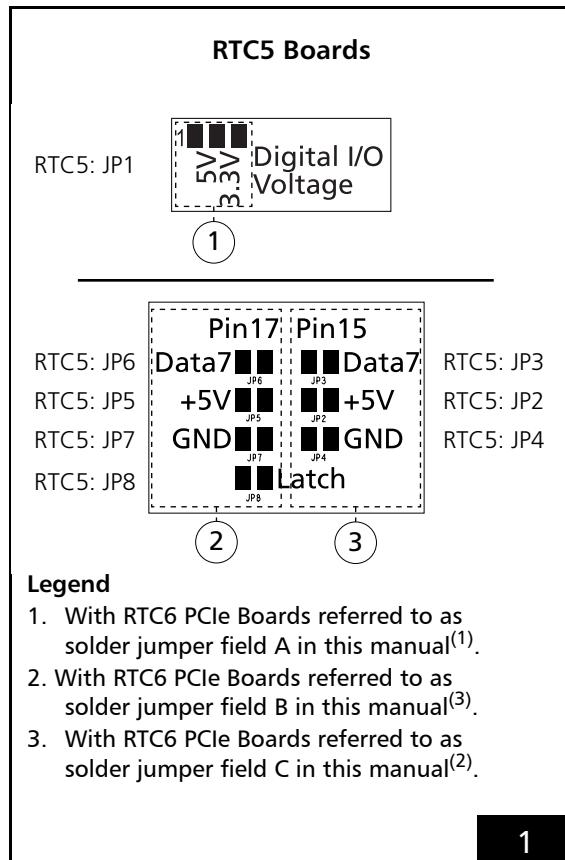
### Notice!

- Only configure allowed jumper settings.  
Otherwise, the board gets damaged!

- (1) For the solder jumper setting, see Chapter 2.4.1 “Solder Jumper Field A – Output Signal Level at the EXTENSION 1 Socket Connector Configuration”, page 33.
- (2) For the solder jumper setting, see Chapter 2.4.2 “Solder Jumper Field B – Pin (17) of the EXTENSION 2 Socket Connector Configuration”, page 34.
- (3) For the solder jumper setting, see Chapter 2.4.3 “Solder Jumper Field C – Pin (15) of the EXTENSION 2 Socket Connector Configuration”, page 35.

### Notes for RTC5 Users

- In contrast to RTC5 boards, jumper numbers are no longer imprinted on RTC6 PCIe Boards. Therefore, RTC5 jumper designations cannot longer be used in this RTC6 Manual. The relation of the designations establishes [figure 1](#).



Jumper designations with RTC5 boards and RTC6 PCIe Boards.

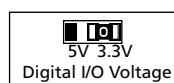
Figure shows RTC5 board.

### 2.4.1 Solder Jumper Field A – Output Signal Level at the EXTENSION 1 Socket Connector Configuration

The solder jumper field A is located on the lower side of the RTC6 PCIe Board, see [figure 3](#).

It is used to set the level (5 V or 3.3 V) of all output signals at the EXTENSION 1 socket connector, see the following table.

See also [Section "Configuring the Output Signal Level", page 67](#).

Allowed jumper setting	EXTENSION 1 socket connector
* closed open	Output signal level 5 V. 
* open closed	Output signal level 3.3 V. 
open open	No signal output. 

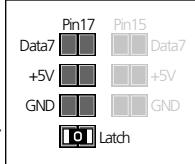
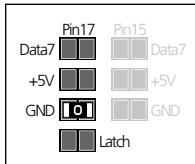
\*Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

## 2.4.2 Solder Jumper Field B – Pin (17) of the EXTENSION 2 Socket Connector Configuration

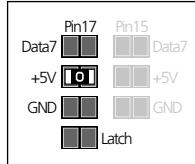
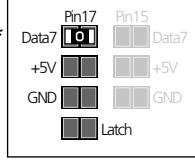
The solder jumper field B is located on the lower side of the RTC6 PCIe Board, see [figure 3](#).

It serves to configure the signal at pin (17) of the EXTENSION 2 socket connector, see the following table.

See also "[Configuration by Solder Jumpers](#)", page 69.

Allowed jumper setting	Output at the EXTENSION 2 socket connector pin (17)
open open open open	No signal. 
open open open closed*	LATCH signal. 
open open closed open	GROUND (low level). 

\*Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

Allowed jumper setting (cont'd)	Output at the EXTENSION 2 socket connector pin (17) (cont'd)
open closed open open	+5 V (high level). 
closed* open open open	DATA7 <sup>(a)</sup> . 

\*Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

(a) Synonym: Data Bit #7. **MSB** of the 8-bit output value.

### Notes

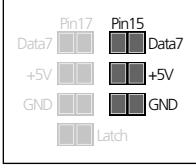
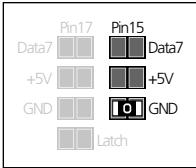
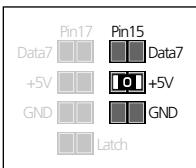
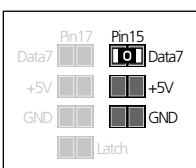
- Configurations of solder jumper field B and solder jumper field C are independent from each other.
- On RTC6 Ethernet Boards the printed label of solder jumper field B is 'Pin17'. This has been chosen deliberately in order to keep consistency with already existing RTC boards. Even though with RTC6 Ethernet Boards pin (09) of the EXT. 2 socket connector, see [Chapter 15.2.12 "EXT. 2 Socket Connector"](#), page 843, is actually configured.

### 2.4.3 Solder Jumper Field C – Pin (15) of the EXTENSION 2 Socket Connector Configuration

The solder jumper field C is located on the lower side of the RTC6 PCIe Board, see [figure 3](#).

It serves to configure the signal at pin (15) of the EXTENSION 2 socket connector, see the following table.

See also “[Configuration by Solder Jumpers](#)”, page 69.

Allowed jumper setting	Output at the EXTENSION 2 socket connector pin (15)
	open open open No signal.
	closed* open open GROUND (low level).
	open closed* open +5 V (high level).
	open open closed* DATA 7 <sup>(a)</sup> .

\* Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

#### Notes

- Configurations of solder jumper field C and solder jumper field B are independent from each other.
- On RTC6 Ethernet Boards the printed label of solder jumper field B is ‘Pin15’. This has been chosen deliberately in order to keep consistency with already existing RTC boards. Even though with RTC6 Ethernet Boards pin (08) of the EXT. 2 socket connector, see [Chapter 15.2.12 “EXT. 2 Socket Connector”](#), page 843, is actually configured.

(a) Synonym: Data Bit #7. **MSB** of the 8-bit output value.

## 2.5 Accessories for the RTC6 PCIe Board

Only hardware extensions from SCANLAB should be used in combination with the RTC6 PCIe Board. In addition to the RTC6 PCIe Board and its software package, the following accessories can be obtained from SCANLAB:

- XY2-100 Converter, page 36
- Laser Adapter, page 36
- Data Cables, page 36
- Slot Cover with 9-pin D-SUB Connector for Using the "2. SCANHEAD" Socket Connector, page 36
- Slot Cover with 15-pin D-SUB Connector for Using the "MARKING ON THE FLY" Socket Connector, page 37

### Notes

- On the UFP Ext Board for the RTC6 PCIe Board<sup>(1)</sup>, see [Chapter 16 "Appendix B: The UFP Ext Board"](#), page 872.
- The following component can be used with RTC6 PCIe Boards:
  - Extension board  
RTC5/6 varioSCAN 40 FLEX Extension, #128683<sup>(2)</sup>
- The following components *cannot* be used with RTC6 PCIe Boards:
  - EXT1 extension board for the RTC5, #123804<sup>(3)</sup>
  - ADC add-on board, #121126
  - RTC Step Motor Extension board, #112097
  - I/O extension board for RTC3 and RTC4, #108285, #121721

(1) Recommended for users with ANALOG controlled lasers, who want to achieve pixel output frequencies > 100 kHz, see [set\\_pixel\\_line](#).

(2) Can also be used with RTC6 Ethernet Boards. Requires Y cable #116050 (see [figure 74](#)) and flat ribbon cable #105446. In addition, the customer has to supply a 1:1 cable (plug 1: identical in construction to Würth 61201023021, 10-pin, female; plug 2: see under Notes, [page 841](#)).

(3) Developed for RTC5. Cannot be used with RTC6 PCIe Boards due to mechanical reasons (collision with cooling element; replacement: UFP Ext Board, [page 872](#)).

### 2.5.1 XY2-100 Converter

The SCANLAB XY2-100 converter allows the RTC6 PCIe Board to control scan systems which are equipped with a conventional XY2-100 interface. See also and [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)"](#), page 58.

### 2.5.2 Laser Adapter

The SCANLAB laser adapter is plugged into the 15-pin LASER connector of the RTC6 PCIe Board. Then a 9-pin female D-SUB connector of this adapter provides the same signals and pin-out as the 9-pin laser connector of the RTC4/RTC5. See also [Section "Laser Adapter \(Accessory\)"](#), page 65.

### 2.5.3 Data Cables

To connect the RTC6 PCIe Board to scan systems, SCANLAB offers appropriate cables in a variety of lengths – either conventional cables for electrical data transfer or polymer optical fiber cables or optical data transfer. See also [Chapter 4.5.3 "Data Cables \(Accessories\)"](#), page 60).

### 2.5.4 Slot Cover with 9-pin D-SUB Connector for Using the "2. SCANHEAD" Socket Connector

For connecting a second scan head or a z axis to the second scan head connector (10-pin 2. SCANHEAD socket connector, see [Section "2. SCANHEAD Socket Connector \(Connector for Second Scan Head\)"](#), page 57), a slot cover with a 9-pin D-SUB connector is available from SCANLAB. See also [Section "Slot Cover \(Accessory\)"](#), page 57.



## 2.5.5 Slot Cover with 15-pin D-SUB Connector for Using the "MARKING ON THE FLY" Socket Connector

For using the inputs and signals of the MARKING ON THE FLY socket connector (16-pin socket connector, see [Chapter 4.6.4](#)

"[MARKING ON THE FLY Socket Connector](#)", page 71, a slot cover with a 15-pin D-SUB connector is available from SCANLAB. See also [Section "Slot Cover \(Accessory\)", page 72](#).

## 2.6 Supplementary Software

To facilitate customizing RTC correction files basing on data of your own test measurements, SCANLAB offers the correXion pro software with accompanying manual, see also [Section "Field Correction Algorithm", page 163](#).

By using the SCANLAB laserDESK software, own laser marking and material-processing programs ("laser jobs") can be created and executed without software development. Many of the RTC6 functions are supported. For further information on laserDESK, refer to the SCANLAB homepage.

## 2.7 Notes for RTC4 Users

This chapter provides an overview of key changes introduced with RTC6 PCIe Boards for comparable functions of RTC4 boards.

For example, [Chapter 2.7.2 "Porting RTC4 Source Code to the RTC6 PCIe Board", page 39](#) discusses a possible approach to porting RTC4 programs to run on the RTC6 PCIe Board.

The individual command descriptions in particular note changes.

### 2.7.1 Hardware Changes

When migrating from the RTC4 to the RTC6 PCIe Board, you need to consider the following hardware changes for correct cabling of the system components.

#### Controlling Scan Systems

- To control scan systems with XY2-100 interface or XY2-100 Enhanced interface, you additionally need the XY2-100 converter (available from SCANLAB), see [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)", page 58](#).
- The RTC6 PCIe Board *cannot* control scan systems with XY2-100-O interfaces (for optical data transfer).
- To control a scan system with the SL2-100 interface, you need a different data cable (available from SCANLAB), see [Chapter 4.5.3 "Data Cables \(Accessories\)", page 60](#).
- To use the second scan head connector, you need a different adapter cable (including slot cover available from SCANLAB), see [Section "Slot Cover \(Accessory\)", page 57](#).
- For controlling one 3-axis scan system, both scan head connectors must be used, see [Chapter 8.5.2 "3D Scan Systems", page 222](#).
- Simultaneous control of two 3-axis scan systems requires two RTC6 PCIe Boards, see [Chapter 8.5.2 "3D Scan Systems", page 222](#).

#### Controlling the Laser

- The D-SUB LASER connector at the RTC6 PCIe Board slot cover has 15 pins (RTC4: 9 pins). Its pinout is not jumper-configurable.
  - The RTC6 PCIe Board does not require jumpers X6 and X7 of the RTC4 because all signals are available at the RTC6 PCIe Board LASER connector.
  - The voltage range of the analog outputs is always 0...10 V (0 V...2.50 V is no longer supported; the jumper X3 of the RTC4 does not exist on the RTC6 PCIe Board).
- If you want to connect a laser to the RTC6 PCIe Board by the same (9-pin) cable that you previously used with the RTC4, then you need an adapter with a 9-pin female D-SUB connector (available from SCANLAB).
  - For use of the SCANLAB laser adapter, see [Section "Laser Adapter \(Accessory\)", page 65](#), two jumpers are provided for configuring the pin-outs (JP1 corresponds to jumper X7 of the RTC4, JP2 corresponds to jumper X6 of the RTC4).
- The signal levels of the laser control signals are no longer determined by configuring jumpers. Instead, they can/must be defined by an RTC6 command, see [set\\_laser\\_control](#).
  - Jumper X10 of the RTC4 does not exist on the RTC6 PCIe Board.

#### EXTENSION 1 Socket Connector

- The RTC6 PCIe Board EXTENSION 1 socket connector is – except for the additionally provided signals at pin (33)...(35) – identical to the EXTENSION 1 socket connector of the RTC4, see [figure 18](#).
- With the RTC6 PCIe Board, the level of all output signals at the EXTENSION 1 socket connector can be configured for 5 V or 3.3 V by a jumper, see [Section "Configuring the Output Signal Level", page 67](#).

## EXTENSION 2 Socket Connector

- The RTC4 EXTENSION 2 socket connector does not exist on the RTC6 PCIe Board. Therefore, the "I/O extension board for RTC3 and RTC4" (#108285, #121721) cannot be attached to the RTC6 PCIe Board.
- The RTC6 PCIe Board EXTENSION 2 socket connector is – except for the LATCH signal at pin (17) which can be optionally set – identical to the LASER EXTENSION socket connector of the RTC4, see [figure 20](#).
  - For configuring pin-outs the RTC6 PCIe Board provides the solder jumper field B and C, see also [Section "Configuration by Solder Jumpers", page 69](#).

These correspond to RTC4 jumpers X8 and X9.

## MARKING ON THE FLY Socket Connector

- The RTC6 MARKING ON THE FLY socket connector and the RTC4 MARKING ON THE FLY socket connector are identical, see [figure 21](#).

## Other Hardware Interfaces

- The following interfaces only exist on the RTC6 PCIe Board:
  - Master and Slave, see [Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 55](#)
  - RS232, see [Chapter 4.6.5 "RS232 Socket Connector", page 72](#)
  - McBSP/ANALOG, see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#)
  - STEPPER MOTOR, see [Chapter 4.6.7 "STEPPER MOTOR Socket Connector", page 77](#)

## 2.7.2 Porting RTC4 Source Code to the RTC6 PCIe Board

User programs written for the RTC4 can only run on the RTC6 PCIe Boards after suitable code revision. This applies even when the actual program flow remains unchanged.

### Changed Initialization

The program's initialization section should be revised at least as follows:

- At the beginning of the program, an `init_rtc6_dll` must be inserted for initializing the [RTC6 DLL](#) and RTC6 board management, see [Chapter 6.2.3 "Initializing the RTC6 DLL and RTC6 Board Management", page 87](#).
- The files for initializing the board by `load_program_file` are different than with the RTC4, see command description.
- Scan system initialization by `load_correction_file` and `select_cor_table` utilizes different correction files (with file extension \*.ct5), see [Chapter 7.3.5 "Image Field Correction and Correction Tables", page 162](#).
- For laser control initialization, `set_laser_control` must be additionally inserted, see [Chapter 7.4 "Laser Control", page 173](#).



## Command Changes

All unsupported RTC4 commands must be removed or replaced. See also [Chapter 10.3 "Unsupported RTC4/RTC5 Commands"](#), page 817.

Changed or enhanced RTC4 commands must be handled differently in the program (for example, by modifying supplied parameter values or evaluating returned values differently). Changes to still supported commands are listed in the individual command descriptions (in [Chapter 10.2 "RTC6 Command Set"](#), page 301), "RTC4→RTC6" row.

RTC4 commands that need to be replaced or checked are:

- `auto_cal` changed
- `auto_change_pos` changed
- `control_command` changed
- `dsp_start` not supported
- `get_head_status` changed
- `get_hi_data` changed
- `get_list_space` changed
- `get_marking_info` changed
- `get_RTC_version` changed
- `get_startstop_info` changed
- `get_status` changed
- `get_value` changed
- `get_waveform` changed
- `get_xy_pos` not supported
- `get_xyz_pos` not supported
- `goto_xy` changed
- `goto_xyz` changed
- `list_jump_cond` changed
- `list_nop` changed
- `load_correction_file` changed
- `load_program_file` changed
- `read_pixel_ad` not supported
- `read_status` changed
- `rtc4_count_cards` not supported
- `select_cor_table` changed
- `select_list` not supported
- `select_rtc` changed
- `set_control_mode` enhanced
- `set_control_mode_list` enhanced
- `set_laser_mode` enhanced
- `set_laser_timing` Firmware changed
- `set_list_mode` not supported
- `set_matrix` changed
- `set_matrix_list` changed
- `set_offset` changed
- `set_offset_list` changed
- `set_piso_control` not supported
- `set_pixel` changed
- `set_pixel_line` changed
- `set_softstart_level` changed
- `set_softstart_mode` changed
- `set_trigger` enhanced
- `set_wobbel` changed
- `set_wobbel_xy` not supported
- `timed_jump_abs` changed
- `timed_jump_rel` changed
- `timed_mark_abs` changed
- `timed_mark_rel` changed
- `z_out` not supported
- `z_out_list` not supported



## Increased Parameter Resolution

When switching from the RTC4 to the RTC6 PCIe Board – even for some commands not mentioned above – take note that the resolution has been increased for several parameters. Examples:

- For commands such as `mark_abs` or `jump_rel`, the real-image-field x and y coordinate values are specified with 20-bit resolution for the RTC6 PCIe Board (whereas with 16-bit resolution for the RTC4), see also [Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156](#).
- Moreover, for Processing-on-the-fly applications, an extended, 29-bit value range is available with the RTC6 PCIe Board, see also [Chapter 7.3.3 "Virtual Image Field", page 158](#).
- For commands such as `write_da_x`, analog output values are specified with 12-bit resolution for the RTC6 PCIe Board (whereas with 10-bit resolution for the RTC4).
- For `set_laser_timing`, output period and pulse length are specified with 1/64  $\mu\text{s}$  resolution for the RTC6 PCIe Board (whereas with 1/8  $\mu\text{s}$  or 1  $\mu\text{s}$  resolution for the RTC4).
- For `set_laser_delays`, the laser delays are specified with 1/64  $\mu\text{s}$  resolution for the RTC6 PCIe Board (whereas with 1  $\mu\text{s}$  resolution for the RTC4).

Alternatively, the `RTC6 DLL` can be set to the `RTC4 Compatibility Mode` by `set_rtc4_mode`. Then the `RTC6 DLL` automatically converts parameter values so that many RTC4 command sequences can run unchanged on the RTC6 PCIe Board.

`RTC4 Compatibility Mode` affects the following RTC4 commands (descriptions of the respective commands include relevant information under the heading "`RTC4→RTC6`"):

- `arc_abs`
- `arc_rel`
- `fly_return`
- `get_z_distance`
- `goto_xy` (changed)
- `goto_xyz` (changed)
- `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` (changed)
- `set_pixel_line` (changed)
- `set_rot_center`
- `set_softstart_level`
- `set_standby`
- `set_standby_list`
- `simulate_ext_start`
- `timed_jump_abs` (changed)
- `timed_jump_rel` (changed)
- `timed_mark_abs` (changed)
- `timed_mark_rel` (changed)
- `write_da_1`
- `write_da_1_list`
- `write_da_2`
- `write_da_2_list`
- `write_da_x`
- `write_da_x_list`

The previously mentioned revision of initialization and checking of unsupported or changed RTC4 commands needs to be carried out regardless of whether the program is to be executed in `RTC6 Standard Mode` or `RTC4 Compatibility Mode`.



### Changed Timing Behavior

The following RTC4 list commands are processed by the RTC6 PCIe Board as "short list commands". This can result in a changed timing behavior during user program execution, see [Section "Normal, Short, Variable and Multiple List Commands", page 288](#).

- `clear_io_cond_list`
- `list_call`
- `list_call_cond`
- `list_jump_cond` (changed)
- `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 changed)
- `set_list_jump`
- `set_mark_speed`
- `set_scanner_delays`
- `set_standby_list`
- `set_trigger` (enhanced)
- `set_wobbel` (changed)
- `write_8bit_port_list`
- `write_da_1_list`
- `write_da_2_list`
- `write_da_x_list`
- `write_io_port_list`

Likewise, automatic delay adjustments can produce a changed timing behavior, see [Section "Automatic Delay Adjustments", page 144](#).

### 2.7.3 New and Changed Functionality

#### Interface to the PC

- The RTC6 board driver supports simultaneous control of any number of RTC6 PCIe Boards in a single PC, see [Chapter 6.6 "Using Several RTC6 PCIe Boards in One PC", page 113](#).
- Connectors and commands for master/slave synchronization of several RTC6 PCIe Boards, see [Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 55](#).

#### Controlling Scan Systems

- New interface to the scan system, see [Chapter 4.5.1 "Scan Head Connectors and Transfer Protocol", page 56](#):
    - 9-pin female D-SUB connector at the RTC6 PCIe Board slot cover and 10-pin socket connector
    - SL2-100 transfer protocol
    - 2 data channels each for both scan head connectors
    - Galvanically isolated signals
    - 20-bit positioning resolution, see [Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156](#)
    - Enhanced status return from the scan system, see [Chapter 7.3.7 "Status Monitoring and Diagnostics", page 172](#)
    - Control and status channels for enhanced data transfer with iDRIVE scan systems<sup>(1)</sup>
  - An XY2-100 converter is provided for data transfer according to the XY2-100 protocol, see [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)", page 58](#).
    - 25-pin female D-SUB connector
    - 16-bit positioning resolution
    - Status return according XY2-100 or XY2-100 Enhanced protocol
    - Transfer synchronization is configurable for long data cables by solder jumpers in the XY2-100 converter
- The RTC6 PCIe Board provides power for this converter.

(1) See glossary entry on [page 879](#).

- For optical data transfer between the RTC6 PCIe Board and scan systems, no special variant of the RTC6 PCIe Board (with XY2-100-O interface) is required. Optical data transfer can be realized by a SCANLAB data cable with electrical-to-optical conversion in its D-SUB connector housing, see [Chapter 4.5.3 "Data Cables \(Accessories\)", page 60](#)
- For controlling a 3-axis scan system, see [Chapter 8.5.2 "3D Scan Systems", page 222](#):
  - Both scan head connectors must be used
  - Simultaneous control of two 3-axis scan systems requires two RTC6 PCIe Boards
- Image field correction
  - New correction files are needed, see [Chapter 7.3.5 "Image Field Correction and Correction Tables", page 162](#):
    - File name extension ".ct5"
    - Correction with higher resolution
    - Correction data information in the file header is queryable
    - For the RTC6 PCIe Board, RTC4 correction files (.ctb) need to be newly calculated or converted by `CorrectionFileConverter.exe` which is part of the RTC6 Software Package.
    - Up to 8 correction files can be loaded to the RTC6 PCIe Board
    - Enhanced 3D image field correction by stretch correction tables, see [Section "Enhanced 3D Correction", page 225](#)
  - Coordinate transformations, see [Chapter 8.2 "Coordinate Transformations", page 210](#):
    - The correction file is no longer transformed (rotation, shift, extension) at download
    - Matrix transformations are only applied after microstepping – this may cause the mark speed to change
    - "[Local Online Positioning](#)", see [Chapter 8.3.1 ""Local Online Positioning"", page 214](#)
- Coordinate transformations in the virtual image field (incl. "[Global Online Positioning](#)"), see [Chapter 7.3.3 "Virtual Image Field", page 158](#)
  - 29-bit position coordinates (virtual image field): objects larger than the real image field are possible
- Position monitoring of iDRIVE scan systems<sup>(1)</sup> by backward transformation of actual position values, see [Chapter 8.1.3 "Monitoring the Positioning", page 201](#)
- Automatic self-calibration, see [Chapter 8.10 "Automatic Self-Calibration", page 261](#):
  - Optimization of previous functions
  - ASC hardware check
- Jump mode, see [Chapter 8.1.5 "Jump Mode", page 203](#)
- Cycle synchronization, see [Chapter 7.4.10 "Synchronization of the RTC6 Clock Cycle and an External Clock Signal", page 196](#)

## Controlling the Laser

- The signal levels of the laser control signals are no longer determined by configuring jumpers. Instead, they can/must be software-configured, see [set\\_laser\\_control](#)
- 15-pin female D-SUB LASER connector with all laser signals at the RTC6 PCIe Board slot cover, see [Chapter 4.6.1 "LASER Connector", page 62](#), 9-pin female D-SUB connector only by the SCANLAB laser adapter, see [Section "Laser Adapter \(Accessory\)", page 65](#)
- 15-pin female D-SUB LASER connector configurable by software command, see [Chapter 7.4.2 "Configuring the LASER Connector", page 176](#)
- Laser control signals with 15 ns resolution and 20 mA output current
- Standby signals in YAG modes, see [Chapter 7.4.4 "YAG Modes 1, 2, 3, 5", page 179](#)
- YAG Mode 5: Time between FirstPulseKiller signal and first laser pulse in YAG mode is freely programmable, see [Chapter 7.4.4 "YAG Modes 1, 2, 3, 5", page 179](#)
- Laser Mode 6: LASERON signal synchronized with a continuously-running LASER1 signal, see [Chapter 7.4.6 "Laser Mode 6", page 183](#)

(1) See glossary entry on [page 879](#).

- Pulse picking laser mode, see [Chapter 7.4.8 "Pulse Picking Laser Mode"](#), page 185
- Laser pulse period, pulse length or analog output are also programmable within a [Polyline](#) between two vectors – where the laser remains on<sup>(1)</sup>, see "short list commands" in [Section "Normal, Short, Variable and Multiple List Commands"](#), page 288
- Commands for position-dependent, speed-dependent, vector-defined and encoder-speed-dependent laser control, see [Chapter 7.4.9 "Automatic Laser Control"](#), page 186

### Interfaces for Peripheral Equipment

- 16-bit digital output, see [Section "16-Bit Digital Input and 16-Bit Digital Output"](#), page 67, and [Chapter 9.1.1 "16-Bit Digital Output Port"](#), page 268:
  - Level of output signals selectable by a jumper (3.3 V or 5 V)
  - LATCH signal for synchronization of data transmission
- 8-bit digital output, see [Section "8-Bit Digital Output Port"](#), page 70 and [Chapter 9.1.2 "8-Bit Digital Output Port"](#), page 269:
  - Provided at the EXTENSION 2 socket connector (on the RTC4, this socket connector is named "LASER EXTENSION")
  - LATCH signal for synchronization of data transmission
  - Adjustable "stop output value"
- Analog outputs, see [Section "12-Bit Analog Output Port 1 and 2"](#), page 63, and [Chapter 9.1.4 "12-Bit Analog Output Ports"](#), page 269:
  - 12 bit resolution
  - 0...10 V (0 V...2.50 V no longer available)
  - Adjustable "stop output value"
- 16-bit digital input, see [Section "16-Bit Digital Input and 16-Bit Digital Output"](#), page 67, and [Chapter 9.2.1 "16-Bit Digital Input Port"](#), page 274.
  - SYNC signal for synchronization of data transmission
- Programmable debouncing of external start signals, see [bounce\\_supp](#) and [Section "External Start"](#), page 276
- Regular (periodic) external starts, see [Section "Regular \(Periodic\) External Starts"](#), page 279
- New interfaces:
  - 2-bit digital input and 2-bit digital output at the D-SUB LASER connector, see [Section "2-Bit Digital Input Port"](#), page 63 and [Section "2-Bit Digital Output Port"](#), page 63
  - RS-232 interface by an 10-pin socket connector, see [Chapter 4.6.5 "RS232 Socket Connector"](#), page 72
  - Stepper motor signals for 2 motors by an on-board 10-pin socket connector, see [Chapter 4.6.7 "STEPPER MOTOR Socket Connector"](#), page 77
  - McBSP and ANALOG IN by an 10-pin socket connector, see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector"](#), page 73
- For the RTC6 PCIe Board, there is no longer an IO extension board. Therefore, it does not have a socket connector for installing such a board (on the RTC4, this socket connector is named "EXTENSION 2"; the RTC6 PCIe Board EXTENSION 2 socket connector corresponds to the RTC4 "LASER EXTENSION" socket connector).

(1) Is switched off with the RTC4.



## General Programming

- Utility files for C, C++, C# and Delphi, see [Chapter 6.2.2 "Importing Commands", page 85](#), but no longer utility files for Basic
- Commands for changing access rights to RTC6 PCIe Boards, see [Chapter 6.7 "Usage of RTC6 PCI Express Boards by Several User Programs", page 118](#)
- Improved and extended list handling, see [Chapter 6.4 "List Handling", page 95](#)
  - List memory with 8,388,608 storage positions
  - List memory free configurable (in two list memory areas and a protected list memory area)
  - Defining protected subroutines
  - Loading lists with protection
  - Loops in lists and subroutines
  - Circular queue mode is not available (but can be coded alternatively)
  - List status and list execution status
- "Short" list commands (for example, to change speed, analog output, I/O port, etc.) can be executed without time losses (multiple "short" list commands can be executed within one 10 µs clock period, see [Section "Normal, Short, Variable and Multiple List Commands", page 288](#))
- Functions for error handling and download verification, see [Chapter 6.8 "Error Handling", page 120](#)

## Laser Marking

- Vectors and arcs
- Commands for marking ellipses, see [Section "Ellipse Commands", page 127](#)
- Commands for marking spirals, see [Chapter "3D Commands", page 223](#)
- Timed arc commands, timed 3D vector commands, see [Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260](#)
- Para-mark and para-jump commands for "vector-controlled laser control", see [Section "Vector-Defined Laser Control", page 195](#)
- Characters and texts
  - Defining character sets and text strings, see [Section "Defining Indexed Character Sets", page 108](#)
  - Commands for marking individual characters and for marking texts (with selectable character set), see [Section "Calling Indexed Characters", page 109](#)
  - Commands for marking dates, times and serial numbers (with selectable character set and selectable serial-number-set), see [Chapter 7.5 "Marking Dates, Times and Serial Numbers", page 197](#)
- Micro vector commands (direct position output without microstepping), see [Chapter 8.8 "Microvector Commands", page 259](#).



## Special Functions

- Synchronization of scan system and laser control
  - Sky Writing, see [Chapter 7.2.4 "Sky Writing", page 149](#)
- Pixel output mode (scanning bitmaps), see [Chapter 8.7 "Pixel Output Mode – Marking Pixel Images \(Bitmaps\)", page 249](#):
  - Pixel output frequencies up to 800 kHz, beyond that with [Option "UFPM"](#) up to 3.2 MHz
  - RTC4-Pixelmode 0 with 10 µs clock synchronous output is no longer supported
  - 15 ns resolution
  - 0...100% laser pulse length
  - Pixel-Mode 0 not supported
  - Reading of analog voltages is not supported
  - 3D pixel lines with [`set\_pixel\_line\_3d`](#)
- Commands for conditional execution of any list command, see [Chapter 9.3.2 "Conditional Command Execution", page 281](#)
- Possible wobble motion shapes include not only circles, but also ellipses, horizontal figure-of-8s, vertical figure-of-8s, and “freely definable wobble shapes”. Options for the orientation of the wobble shapes are: stationary in space, continuously and automatically adjusted to the current direction of motion, or any other freely assigned motion direction. With “Freely definable wobble shapes”, also the laser power can be varied, see [Chapter 8.4 "Wobbel Mode", page 218](#).
- Camming functionality, see [Chapter 8.11 "Camming", page 265](#)
- Enhanced signal recording, see [`set\_trigger`](#) and [`set\_trigger4`](#)
- Processing-on-the-fly, see [Chapter 8.6 "Processing-on-the-fly", page 227](#)
  - two encoder inputs (RS-422) with 32-bit counter for Processing-on-the-fly correction with encoder signals on 2 axes, see [Chapter 9.3.3 "Synchronization by Encoder Signals", page 284](#); alternatively: Processing-on-the-fly correction with McBSP signals, see [Chapter 9.3.4 "Synchronization and Online Positioning by McBSP Signals", page 286](#)
  - 29-bit coordinates (virtual image field): objects larger than the real image field are possible, see [Chapter 7.3.3 "Virtual Image Field", page 158](#), and [Chapter 8.6.6 "Virtual Image Field with Processing-on-the-fly", page 237](#)
  - up to 8 objects within the Processing-on-the-fly track delay (between trigger and marking position), see [Section "External Start", page 276](#)
  - Accurate “external start”: If accordingly configured by [`set\_control\_mode`](#), the encoder counter can be reset by external start signals for synchronizing a Processing-on-the-fly process. The reset occurs fully simultaneously (without 10 µs jitter) with the external start signal.
  - Compensation of 2D motions (xy-positioning stage)
  - Encoder-based Processing-on-the-fly correction for the z axis (“FlyZ correction”), see [Chapter 8.6.11 "Processing-on-the-fly Correction for the z Axis", page 243](#)



## 2.8 Notes for RTC5 Users

RTC6 PCIe Board can control:

- Scan systems with SL2-100 interface
- Scan systems with XY2-100 interface  
(in conjunction with the XY2-100 converter)

The following is an overview of key differences between RTC6 PCIe Boards and RTC5 boards.

Also described are possible approaches for migrating RTC5 program code for use with RTC6 PCIe Boards.

### Notes

- Users of excelliSCAN scan heads must *additionally* read the "excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" manual.

### 2.8.1 RTC6 PCIe Boards vs. RTC5 Boards

Compared to the RTC5 board family, the RTC6 PCIe Board uses newer and more powerful hardware components. This applies in particular to the **DSP**, the **FPGA** and the main memory.

Existing user programs only require a few changes, see [Chapter 2.9.2 "Adapting RTC5 Source Code for the RTC6 PCIe Board"](#), page 49.

#### Functionality changes

- The RTC6 PCIe Board executes short vectors<sup>(1)</sup> faster than RTC5 boards, see [set\\_dsp\\_mode](#).
- The RTC6 PCIe Board lists memory features 8,388,608 (=  $2^{23}$ ) list positions. List 1 and List 2 are each initialized with 4,194,304 (=  $2^{22}$ ) list positions.
- The pixel output mode is significantly enhanced, see [set\\_pixel\\_line](#).
  - RTC6 PCIe Boards without **Option "UFPM"** can achieve pixel output frequencies of up to 800 kHz out-of-the-box.
  - RTC6 PCIe Boards, with **Option "UFPM"**, even up to 3.2 MHz.
  - With [set\\_pixel\\_line](#) it is now possible to digitally output laser power control values. This can be done either at the 8-bit or 16-bit output (EXTENSION 2 socket connector or EXTENSION 1 socket connector).
- Extended measurement-value recording with the RTC6 PCIe Board: 4 recording channels, each storing 8,388,608 data values, see [set\\_trigger4](#). Practically any amount of data can be recorded (ring buffer).

(1) "dashed lines"



- Eight 3D correction tables can be used on the RTC6 PCIe Board simultaneously with four recording channels (see [set\\_trigger](#), [set\\_trigger4](#)) and the complete list memory. With [number\\_of\\_correction\\_tables](#), a user-defined limit to less than 8 can be set to check for incorrect user input.
- The virtual image field of the RTC6 PCIe Board is now  $\pm 28$  bit=29 bit.
- You can set an appropriate time-lag compensation value for 3D systems with different tracking-error behavior between the xy axes and z axis, see [set\\_timelag\\_compensation](#).
- The output synchronization of the RTC5 is replaced by a cycle synchronization with the RTC6 PCIe Board, see [Chapter 7.4.10 "Synchronization of the RTC6 Clock Cycle and an External Clock Signal", page 196](#).

#### Parameter changes

- The laser delays (parameter [LaserOnDelay](#) and [LaserOffDelay](#) of [set\\_laser\\_delays](#)) as well as [LaserOnShift](#) with Sky Writing commands have a higher resolution of  $1/64 \mu\text{s}$ .  
For downward compatibility with RTC5 boards, see [step 3, page 50](#).
- In fact the parameter [Timelag](#) of [set\\_sky\\_writing\\_para](#) is specified in 64-bit IEEE floating point format ("double";  $1.0 = 1 \mu\text{s}$ ) but on the RTC6 PCIe Board it is actually executed with a resolution of  $1/64 \mu\text{s}$  resolution.

## 2.8.2 RTC6 PCIe Boards and Scan Systems with SCANahead Servo Control

- Only the RTC6 PCIe Board allows usage of SCANahead servo control, for example, offered by scan heads of the excelliSCAN series.
- If you configure the RTC6 PCIe Board by [set\\_scanahead\\_params](#) for operation with an excelliSCAN scan head, the laser control then automatically takes the SCANahead servo technology's PreviewTime time into account (that is, processing time for calculating and executing scanner trajectories<sup>(1)</sup>). This ensures fuller usage of dynamics and precision.
- [activate\\_scanahead\\_autodelays](#) lets you simply and quickly place the excelliSCAN into operation. Neither laser delays nor scanner delays need to be determined or set.
- The following restrictions currently apply for excelliSCAN scan heads:
  - See [RTC6\\_Release\\_Notes\\_<Datum>.pdf](#) in the RTC6 Software Package.
- For further details, see the separate manual "excelliSCAN scan heads – Functional Principle of SCANahead Servo Control and Operation by RTC6 Boards", which also describes SCANahead-specific commands.

(1) See Glossary entry on [page 880](#).

### 2.8.3 Restrictions with RTC6 PCIe Boards

#### Availability of Technical Variants

RTC6 boards are available as RTC6 PCIe Boards and RTC6 Ethernet Boards.

#### Peripheral Interfaces

The ADC add-on card (#121126, optional accessory for the RTC5 PCI board) cannot be used with the RTC6 PCIe Board. The RTC6 PCIe Board supplies the two analog inputs ANALOG IN0 and ANALOG IN1 at the McBSP/ANALOG socket connector directly, see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#).

#### Functionality

The RTC6 PCIe Board's McBSP/ANALOG socket connector cannot be used in SPI mode, see also [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#).

Other preliminary restrictions are mentioned in [RTC6\\_Release\\_Notes\\_<Datum>.pdf](#), which is part of the delivered RTC6 Software Package.

## 2.9 Source Code for RTC6 User Programs

### 2.9.1 Creating New RTC6 Source Code

To create new source code for RTC6 user programs, proceed as described in [Chapter 6.2 "Initialization and Program Start-Up", page 85](#) while observing [Chapter 2.9.2 "Adapting RTC5 Source Code for the RTC6 PCIe Board", page 49](#).

### 2.9.2 Adapting RTC5 Source Code for the RTC6 PCIe Board

User programs written for the RTC5 only become usable on the RTC6 PCIe Board, if you modify the program code as described in the following.

This is true even if program flow remains unchanged.

#### Mandatory Steps

(1) Ensure the following prerequisites are met:

- RTC6 PCIe Board is installed
- RTC6 board driver is installed for your operating system
- The following files are present:  
`RTC6DLL.dll` or `RTC6DLLx64.dll`,  
`RTC6OUT.out` (DSP program file for RTC6 PCIe Boards),  
optionally `RTC6ETH.out` (DSP program file for RTC6 Ethernet Boards),  
`RTC6RBF.rbf` (firmware file),  
`RTC6DAT.dat` (binary help file),  
`*.CT5` (correction file(s))  
Note, `RTC6OUT.out`, optionally `RTC6ETH.out`, `RTC6RBF.rbf`, `RTC6DAT.dat` must be together in the same folder.

- The import declarations were imported in your software and the calling convention `std_call` was applied, see [Chapter 6.2 "Initialization and Program Start-Up", page 85](#).
- (2) Take into account these command name changes of the RTC6 command set:
- Do *not* generally change `_rtc5_` to `_rtc6_`!
  - Change:  
`init_rtc5_dll` to `init_rtc6_dll`  
`free_rtc5_dll` to `free_rtc6_dll`  
`rtc5_count_cards` to `rtc6_count_cards`
  - Notice: Do *not* generally change `set_rtc5_mode` to `set_rtc6_mode` – here see step 3!
- (3) `set_rtc6_mode` is set automatically at program start. Then an expanded parameter resolution is available for
- z coordinates of 20 bits
  - laser delays of 1/64 µs
- In contrast, the control command `set_rtc5_mode` ensures downward compatibility of the parameters ("RTC5 Compatibility Mode").
- Laser delays:  
In [RTC5 Compatibility Mode](#), the RTC6 board multiplies the laser delays (parameter `LaserOnDelay` and `LaserOffDelay` of `set_laser_delays`) automatically by 32. This also applies to the parameter `LaserOnShift` of `set_sky_writing` and the parameter `LaserOnShift` of `set_sky_writing_para`.
  - z coordinates:  
In [RTC5 Compatibility Mode](#), the RTC6 board multiplies the z coordinates and defocus values (for example, parameter `Shift` of `set_defocus`) automatically by 16.

For further modifications and optimizations of your code, see the following chapter.

## Optional Steps and Notes on Further Modification and Optimization

- Commands which have been removed from the RTC6 command set or are no longer supported must be removed or replaced in the program code, see [Chapter 2.10.2 "Removed from RTC6 Command Set", page 51](#).
- Check if command changes might affect your user program, see [Chapter 2.10.1 "Changes to the RTC6 Command Set", page 51](#), and modify your program code accordingly.
- Take into account the changed behavior of `load_program_file`.  
Notice: `load_program_file` now stops lists without warning. The single-board command can even be used again after a version conflict.
- **Note on enhanced parameter resolution**  
Take into account the enhanced parameter resolution of z coordinates when using `set_rtc6_mode`. See also step 3, page 50.
- **Note on changed time behavior**  
The replacement of the automatic delay adjustment can result in faster time behavior for short vectors.  
By `set_dsp_mode( 2 )`, the original RTC5 time behavior can be restored, see [set\\_dsp\\_mode](#).



## 2.10 Changes to the RTC6 Command Set

The most RTC5 commands are covered in the RTC6 command set. The following chapter describes the few differences:

- *Changed* commands, see [Chapter 2.10.1 "Changes to the RTC6 Command Set", page 51](#)
- *Removed* commands, see [Chapter 2.10.2 "Removed from RTC6 Command Set", page 51](#)

list commands list command

### 2.10.1 Changes to the RTC6 Command Set

(n_)	<a href="#">config_list</a> con(1) .....	330
	<a href="#">get_dll_version</a> con .....	383
(n_)	<a href="#">get_hex_version</a> con .....	393
(n_)	<a href="#">get_RTC_version</a> con .....	406
(n_)	<a href="#">get_status</a> con .....	412
(n_)	<a href="#">get_sync_status</a> con .....	416
(n_)	<a href="#">*init_fly_2d</a> con .....	441
(n_)	<a href="#">load_correction_file</a> con .....	471
(n_)	<a href="#">*load_fly_2d_table</a> con .....	478
(n_)	<a href="#">load_program_file</a> con .....	486
(n_)	<a href="#">*load_stretch_table</a> con .....	489
(n_)	<a href="#">mcbsp_init_spi</a> con .....	520
(n_)	<a href="#">number_of_correction_tables</a> con .....	526
(n_)	<a href="#">select_cor_table</a> con .....	575
(n_)	<a href="#">*set_auto_laser_control</a> con .....	585
(n_)	<a href="#">*set_auto_laser_params</a> con .....	589
(n_)	<a href="#">*set_auto_laser_params_list</a> dl (2) .....	589
(n_)	<a href="#">set_dsp_mode</a> con .....	601
(n_)	<a href="#">*set_pixel_line</a> nor (3) .....	681
(n_)	<a href="#">set_trigger4</a> nor (3) .....	721

\* Note: RTC6 command with extended functionality.

### 2.10.2 Removed from RTC6 Command Set

<a href="#">free_rtc5_dll</a> .....	817
<a href="#">init_rtc5_dll</a> .....	817
<a href="#">rtc5_count_cards</a> .....	817

(1) con control command

(2) dl delayed short list command

(3) nor normal list command

### 3 Safety During Installation and Operation

Read these operating instructions completely before you proceed with installing and operating the RTC6 PCIe Board.

If there are any questions regarding the contents of this manual, contact SCANLAB.

The following conventions apply to safety instructions in this manual:



- Safety notices which draw attention to severely injuries or even death are identified by the hazard symbol and the signal word "Warning!".
- Safety notices which draw attention to a health hazard are identified by the hazard symbol and the signal word "Caution!".
- Safety instructions that recommend proper use of the device or warn against possible damage to property are (without hazard sign) only identified by the signal word "Notice!".



#### Notice!

- For storage and operation of the board, avoid electromagnetic fields and static electricity. These can damage the electronics on the board. For storage, always use the antistatic bag the board is delivered in.
- The allowed operating temperature range is 15 °C to 60 °C.
- The storage temperature should be between -20 °C and +60 °C.

### 3.2 Laser Safety

The RTC6 PCIe Board is intended for controlling scan systems and lasers. Therefore all relevant laser safety directives must be known and applied before installation and operation. The customer is solely responsible for ensuring the laser safety of the entire system.

#### 3.1 Safe Operation

##### Notice!

- Carefully check your user program before running it. Programming errors can cause a break down of the system. In this case neither the laser nor the scan system can be controlled.
- Protect the board from humidity, dust, corrosive vapors and mechanical stress.

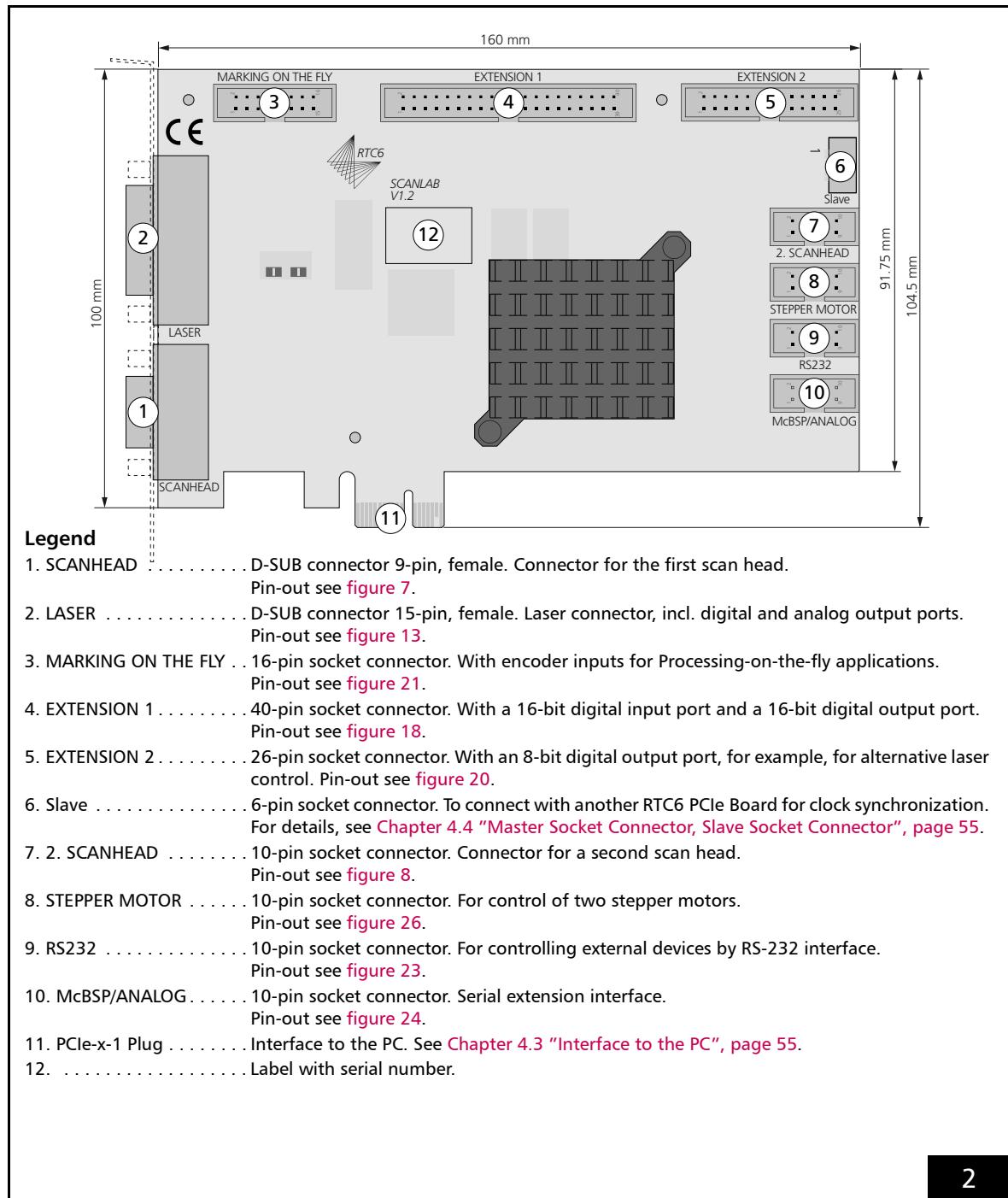


##### Caution!

- All applicable laser safety directives must be adhered to. Safety regulations may differ from country to country. It is the responsibility of the customer to comply with all local regulations.
- Observe all laser safety instructions as described in your scan system manual, chapter "Safety during Installation and Operation".
- Always turn on the PC and the power supply for the scan head first before turning on the laser. Otherwise there is the danger of uncontrolled deflection of the laser beam.*  
SCANLAB recommends the use of a shutter to prevent uncontrolled emission of laser radiation.

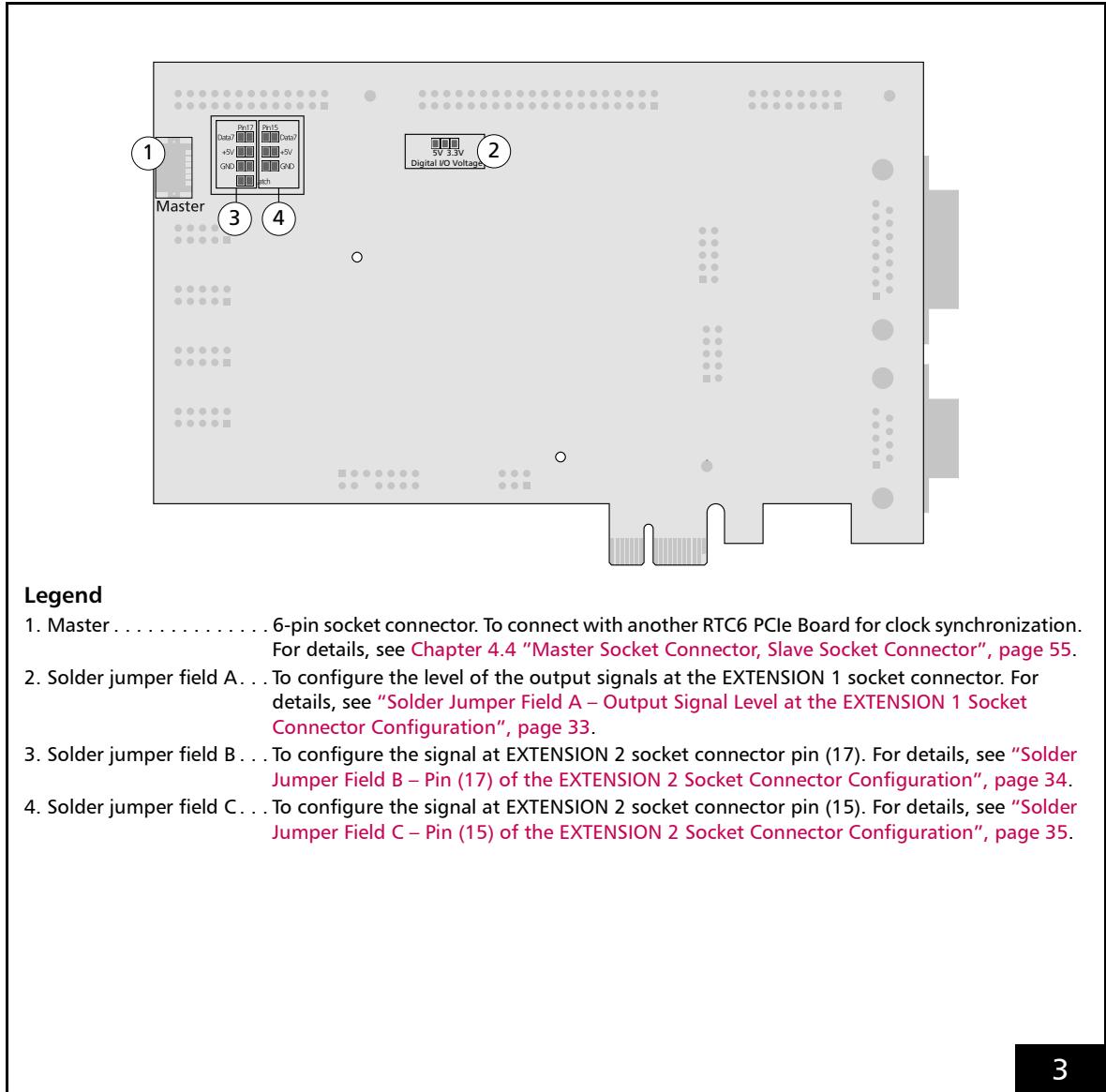
## 4 RTC6 PCIe Board – Layout and Interfaces

### 4.1 Layout – Upper Side



RTC6 PCIe Board: upper side.

## 4.2 Layout – Lower Side



RTC6 PCIe Board: lower side.

### 4.3 Interface to the PC

The interface to the PC is the PCIe-x-1 connector of the RTC6 PCIe Board, see [figure 2](#).

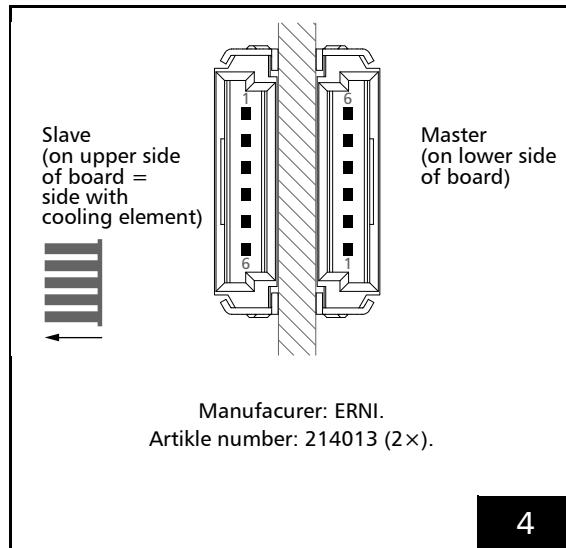
The RTC6 PCIe Board can be installed into any Windows PC with a PCI-Express bus interface and at least one free PCI-Express slot.

#### Notice!

- The RTC6 PCIe Board does not support power-saving modes that switch off power to the PCIe bus. Accordingly, you must disable standby or sleep modes of the operating system. See also [Section "Notes", page 29](#).

### 4.4 Master Socket Connector, Slave Socket Connector

The Slave socket connector as well as the Master socket connector have 6 pins, see [figure 4](#). The Slave socket connector is located on the upper side of the RTC6 PCIe Board, see [figure 2](#). The Master socket connector is located on the lower side, see [figure 3<sup>\(1\)</sup>](#).



Master socket connector and Slave socket connector.  
The pitch of the pins is 1.27 mm.

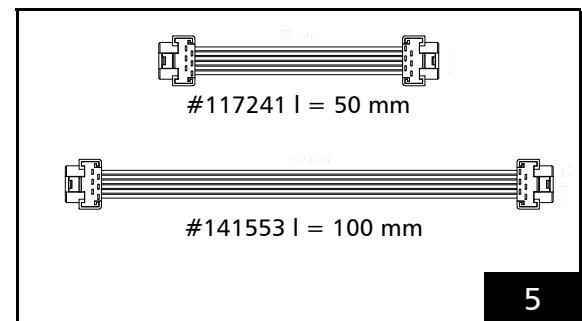
4

Purpose of the both socket connectors is to make a clock cycle synchronization of several RTC6 Ethernet Boards possible. Then they must be connected pairwise with each other by the Master and Slave socket connectors. Always connect a Master connector of a board to the Slave connector of another board by a suitable cable (available from SCANLAB, see [figure 6](#)). The necessary information for assembling your own cables is shown in [figure 5](#).

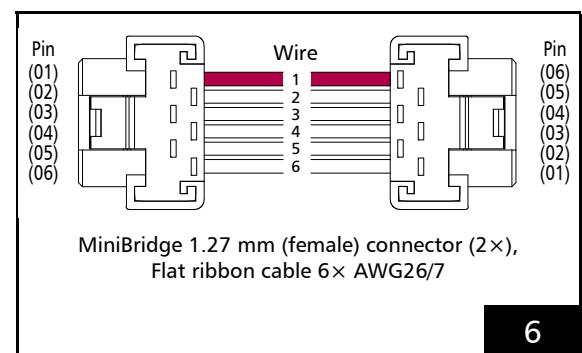
Interconnected RTC6 PCIe Boards should (recommended) be plugged into adjacent PCIe slots.

Important: In order to use the master/slave functionality, see prerequisites in [Chapter 6.6.3 "Master/Slave Operation", page 114](#).

See also [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275](#).



SCANLAB Master/Slave connecting cable.



Cable to connect the Master socket connector and Slave socket connector: requirements. Keep length as short as possible!

#### Notes

- With `master_slave_config` the master/slave interface properties can be configured individually for each RTC6 board.

(1) Vice versa with RTC6 Ethernet Boards.



## 4.5 Interfaces to Scan System

### 4.5.1 Scan Head Connectors and Transfer Protocol

The first scan head connector SCANHEAD and the optionally activated second scan head connector 2. SCANHEAD are available for digitally controlling scan systems, see [figure 7](#).

At those connectors, scan-system control values are transmitted and scan-system status signals received. Each scan head connector can transmit data for up to two axes. Consult your scan system's operating manual to determine which status signals are generated by your scan system and how they can be applied for monitoring purposes.

Data transfer between the RTC6 PCIe Board and the scan system is in accordance with the SL2-100 protocol. SCANLAB can supply an XY2-100 converter for signal transfer in accordance with the XY2-100 protocol, see [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)", page 58](#).

If neither the [Option "Second Scan Head Control"](#) nor the [Option "3D"](#) is enabled, only the first scan head connector outputs signals for an xy scan system.

If the [Option "Second Scan Head Control"](#) is enabled, two xy scan systems can be simultaneously controlled by one RTC6 PCIe Board.

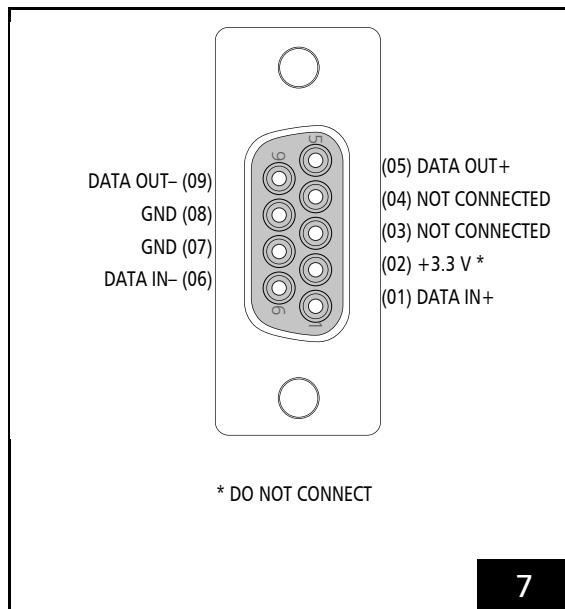
If the [Option "3D"](#) is enabled, then a 3-axis scan system can be controlled by the two scan head connectors (if the first scan head connector has been assigned a 3D correction table). Signals can then be outputted by the first scan head connector to an xy scan head – and by both channels of the second scan head connector to the third axis (z axis).

If *both* options ("second scan head control" and [Option "3D"](#)) are enabled, the assignment of the correction tables determines which signals (xy or z) are to be outputted by which connector, see also [Section "2D and 3D Correction Files", page 163](#).

If several RTC6 PCIe Boards with enabled [Option "3D"](#) are installed in a PC, then that many 3-axis systems can be simultaneously controlled.

## **SCANHEAD Connector (Connector for First Scan Head)**

The pin-out of the first scan head connector SCANHEAD (D-SUB 9-pin female) is shown in figure 7.



SCANHEAD connector (9-pin female D-SUB connector):  
pin-out.

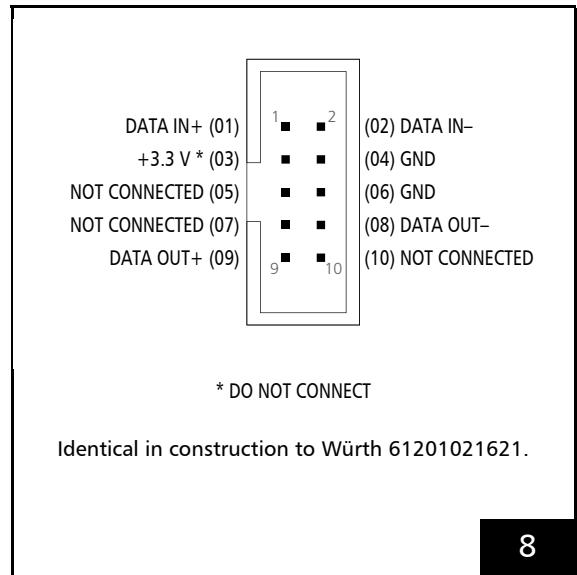
At the differential DATA OUT output port, the control values for the scan system are outputted.

The differential DATA IN input channel receives the status signals returned by the scan system.

Pin (02) supplies 3.3 V power for the SCANLAB XY2-100 converter (accessory) (or a POF<sup>(1)</sup> converter for optical data transmission). This voltage should not be used for other purposes.

## **2. SCANHEAD Socket Connector (Connector for Second Scan Head)**

The pin-out of the second scan head connector 2. SCANHEAD (10-pin socket connector) is shown in figure 8.



2. SCANHEAD socket connector: pin-out. The pitch of the pins is 2.54 mm. Signals for second scan head are only outputted with enabled Option "Second Scan Head Control".

## Slot Cover (Accessory)

SCANLAB recommends using an additional slot cover for connecting a second scan head or a z axis. A suitable slot cover with a 9-pin D-SUB connector – with the same pin-out as the SCANHEAD connector, see **figure 7**, is available from SCANLAB (#115132).

(1) POF = Polymer Optical Fiber.

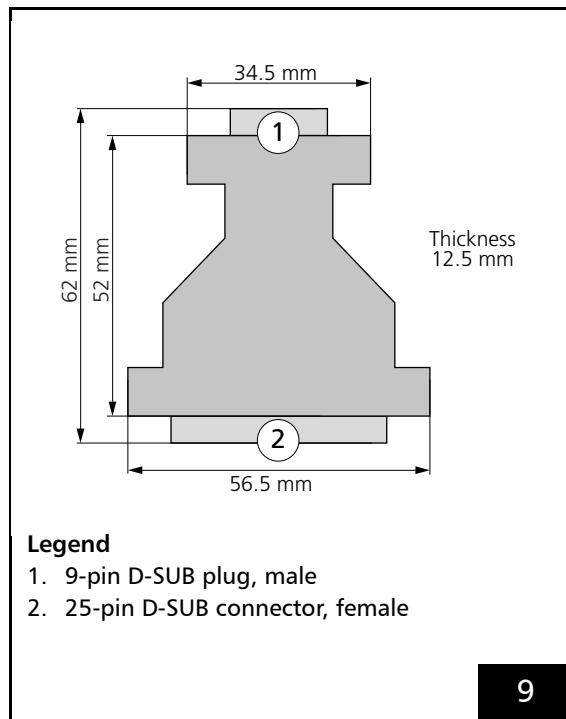
#### 4.5.2 XY2-100 Converter (Accessory)

The SCANLAB XY2-100 converter (accessory; #125377) converts:

- SL2-100-compliant control signals (20 bit) to XY2-100-compliant control signals (16 bit)
- Scan system XY2-100-compliant status signals to SL2-100-compliant status signals, see also [Chapter 7.3.7 "Status Monitoring and Diagnostics", page 172](#)

When controlling scan systems, the XY2-100 converter introduces a  $10 \mu\text{s}$  signal propagation delay. To compensate for this, the LaserOn delay and LaserOff delay must be increased by  $10 \mu\text{s}$  each, see [set\\_laser\\_delays](#).

The dimensions of the XY2-100 converter are shown in [figure 9](#).



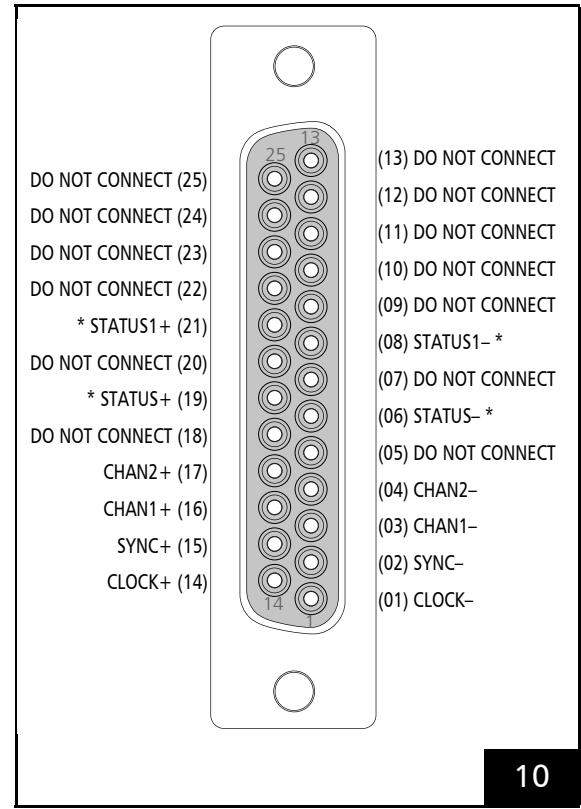
9

XY2-100 converter (accessory): dimensions.

The 9-pin D-SUB (male) plug of the XY2-100 converter should be directly plugged into the SCANHEAD connector of the RTC6 PCIe Board or, (by a short-as-possible 1:1 cable) connected to the corresponding 2. SCANHEAD connector pins of the RTC6 PCIe Board. For the pin-out, see [figure 7](#) and [figure 8](#).

The 25-pin D-SUB connector (female) of the XY2-100 converter is compatible with scan heads that provide an XY2-100 standard interface.

The pin-out is shown in [figure 10](#).



10

XY2-100 converter (accessory): pin-out of the 25-pin D-SUB connector (female).

\* For *iDRIVE* scan systems (see glossary entry on [page 879](#)), the STATUS $\pm$  channel is the status channel of axis 2 (x axis; this channel is then also called STATUS2 $\pm$ ) and the STATUS1 $\pm$  channel is the status channel of axis 1 (y axis). For other scan systems, the STATUS1 $\pm$  channel can not be used: "DO NOT CONNECT".

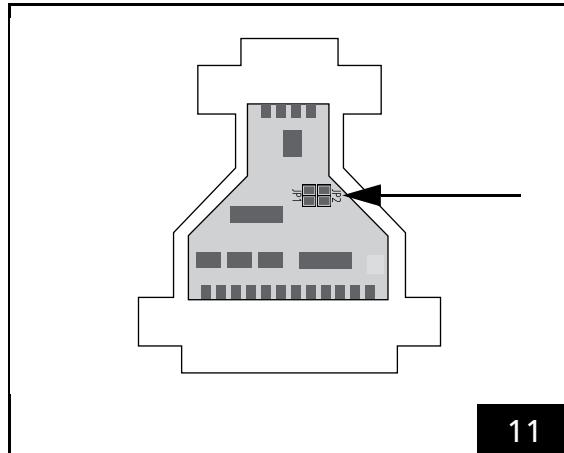
The data channels CHAN1 and CHAN2 transmit control values to the scan head. The SYNC and CLOCK channels transmit synchronization and clock signals to the scan system.

The STATUS channel (and, if appropriate, the STATUS1 channel) receives XY2-100 compliant status signals returned by the scan system.

If cables longer than 20 m (not recommended) are used for data transmission between the XY2-100 converter and the scan system, then the synchronization of bidirectional communication between the scan system and the RTC6 PCIe Board should be configured.

This can be accomplished by two solder jumpers in the XY2-100 converter.

To do so, carefully open the converter housing by its four clip latches<sup>(1)</sup>. The solder jumpers **JP1** and **JP2** are on the PCB of the converter, between the two D-SUB connectors, see arrow in [figure 11](#).



11

XY2-100 converter (accessory): positions of solder jumpers **JP1** and **JP2** on the PCB.

The following table shows the possible jumper settings and the corresponding cable lengths. Other jumper settings are not allowed. Cable lengths above 20 m are not recommended.

Jumper setting	Cable length*
<b>JP1 closed</b> <b>JP2 open</b> (default configuration)	0 m to 20 m
<b>JP1 open</b> <b>JP2 closed</b>	20 m to 40 m

\* The cable length range mentioned here for the respective jumper configuration depends on the used cable type and may differ from the values mentioned above. Cable lengths over 20 m are generally not recommended (and require extensive checks by users prior to productive use).

(1) For example, using a slotted screwdriver.

### 4.5.3 Data Cables (Accessories)

For transmission of the data signals between the RTC6 PCIe Board (or the XY2-100 converter) and the scan system, appropriate cables are obtainable from SCANLAB. Data cables are generally not included in the scope of delivery.

For SCANLAB scan systems equipped with an SL2-100 interface (9-pin female D-SUB connector), data cables are available for electrical transmission, for optical transmission (only upon request) also optical fiber cables<sup>(1)</sup>.

For SCANLAB scan systems equipped with a XY2-100 interface (25-pin female D-SUB connector) solely electrical cables are obtainable.

Scan systems equipped with an optical interface (XY2-100-O) cannot be controlled by the RTC6 PCIe Boards.

With self-constructions, SCANLAB recommends the following design for electrical data transmission:

- For SL2-100-compliant data transmission (see upper part of [figure 12](#)), the cable should be fitted with 9-pin male D-SUB connectors at both ends. The two channels DATA IN $\pm$  and DATA OUT $\pm$  must consist of twisted cable pairs and be cross-connected at both D-SUB connectors (for example, so that the DATA OUT signal of the RTC6 PCIe Board flows to the DATA IN input of the scan system). The cable length should not exceed 25 m. SCANLAB recommends a cable impedance of 110  $\Omega$ , independent from the cable length.

- For XY2-100-compliant data transmission (see lower part of [figure 12](#)), the cable must have identical 25-pin (male) D-SUB connectors at both ends. The five (or six) channels SYNC $\pm$ , CHAN1 $\pm$ , CHAN2 $\pm$ , STATUS $\pm$  (and STATUS1 $\pm$ ) and CLOCK $\pm$  must consist of twisted cable pairs. Together with a XY2-100 converter with standard jumper configuration, the data cable should not be longer than 20 m. If a longer data cable is needed, then the solder jumper setting of the XY2-100 converter need to be reconfigured, see [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)"](#), page 58.
- For XY2-100-compliant data transmission, the controller end of the data cable must be fitted with a ferrite identical in construction to Würth WE 74271132.

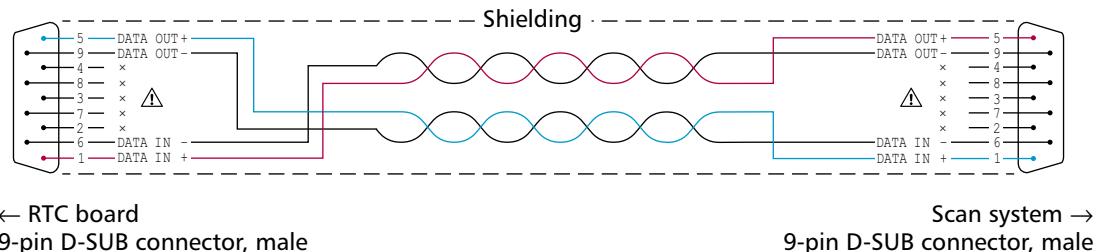
Independently from the data transmission protocol, the following applies in addition:

- The data cable must have coaxial copper braided shielding.
- The D-SUB connectors must have fully shielded metal housings.
- The electrical connection of the cable's braided shielding to the D-SUB housing *must not* be implemented as a wire. Instead, the cable's braided shielding should be *coaxially* connected to the D-SUB housing by shielded clamps.

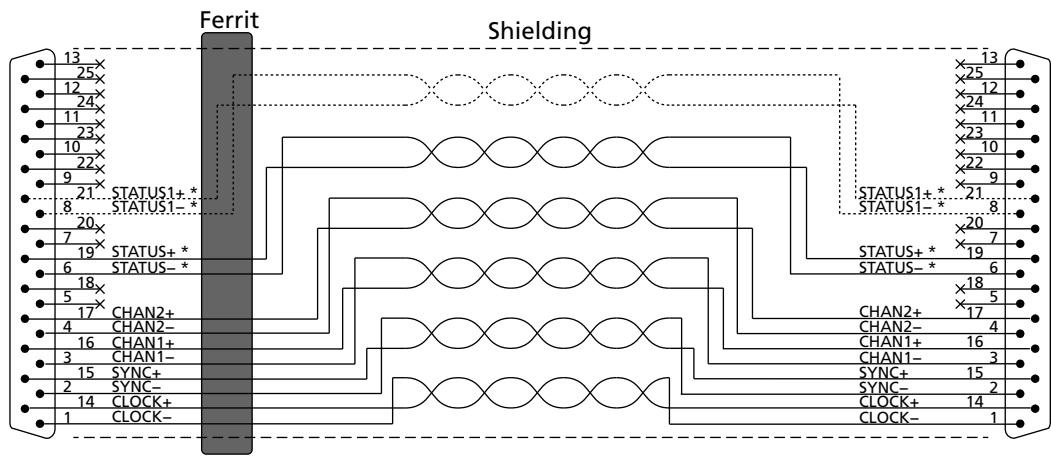
Some scan heads have only a single connector to provide both the operating voltage and the data signals. For these scan heads, SCANLAB recommends implementing a cabling solution which uses a separate cable for data. The cable should have the properties as already described, see above.

(1) The optical fiber cables, too, are attached by 9-pin D-SUB connectors. Optical conversion (POF conversion) for optical data transmission takes place inside the D-SUB connectors. The operating voltage for POF conversion is supplied at the RTC6 PCIe Board scan head connectors and the digital interface of the scan system.

Cable for SL2-100-compliant data transmission



Cable for XY2-100-compliant data transmission



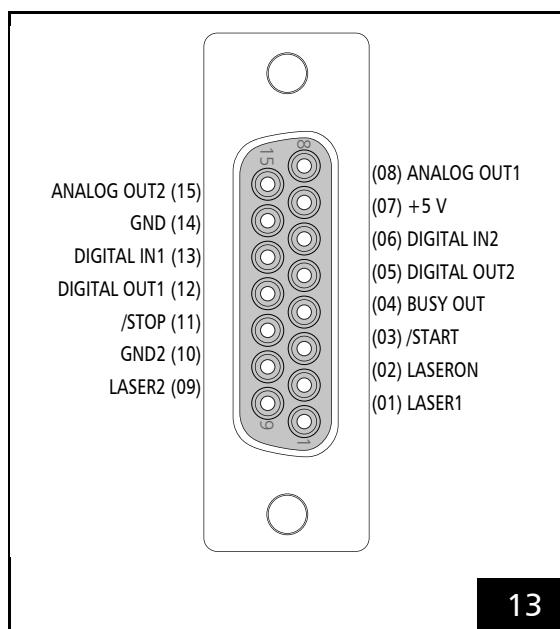
- \* For iDRIVE scan systems (see glossary entry on [page 879](#)), the following applies:  
The STATUS $\pm$  channel is the status channel for axis 2 (x axis;  
this channel is also called STATUS2 $\pm$  there).  
The STATUS1 $\pm$  channel is the status channel for axis 1 (y axis).  
For other scan systems, the STATUS1 $\pm$  channel is not needed.

## 4.6 Interfaces for the Laser and Peripheral Equipment

### 4.6.1 LASER Connector

The LASER connector is a 15-pin D-SUB connector (female). It is located on the slot cover of the RTC6 PCIe Board, see [figure 2](#).

The pin-out is shown in [figure 13](#).



13

LASER connector (15-pin D-SUB connector, female): pin-out. On RTC6 PCIe Boards without [Option "DC/DC Converter"](#), GND2 are GND identical.

#### Laser Output Signals

The laser output signals are LASERON, LASER1 and LASER2. They are digital TTL level signals and are referenced to GND2.

The laser output signals LASER1 and LASER2 differ depending on the set laser mode, see following table and the timing diagrams in [Chapter 7.4 "Laser Control", page 173](#).

	LASER1 pin (01)	LASER2 pin (09)
CO <sub>2</sub> Mode	Modulation pulse 1, standby signal	Modulation pulse 2, standby signal
YAG Mode 1, YAG Mode 2, YAG Mode 3, YAG Mode 5	Q-Switch signal	FirstPulseKiller signal
Laser Mode 4	Standby signal	FirstPulseKiller signal
Laser Mode 6	Standby signal	–

On RTC6 PCIe Boards with [Option "DC/DC Converter"](#), GND2 and the laser output signals are galvanically decoupled from GND<sup>(1)</sup>. On RTC6 PCIe Boards without [Option "DC/DC Converter"](#), GND2 are GND identical, see [Chapter 2.3 "Options", page 30](#).

For the maximum current load see [Chapter 14 "Technical Specifications of the RTC6 PCIe Board", page 824](#).

All laser output signals can be set by [set\\_laser\\_control](#) to either *active-LOW* or *active-HIGH* logic. "*active-LOW*" means that a logical 1 ("Laser On", for instance) is represented by a LOW level (0 V, TTL). "*active-HIGH*" means a logical 1 is represented by a HIGH level (+5 V, TTL). Set the TTL laser signal level according to the specifications of your laser control. Observe the documentation of your laser.

Pin (01), (02) and (09) of the LASER connector can be configured by [config\\_laser\\_signals](#) and [config\\_laser\\_signals\\_list](#), see also [Chapter 7.4.2 "Configuring the LASER Connector", page 176](#).

(1) With RTC6 PCIe Boards GND is the PC ground.  
With RTC6 Ethernet Boards GND is the ground at the POWER connector.



## External Control Signals

The external control signals are /START and /STOP (TTL active-LOW). Both input signals are connected internally to +3.3 V by pull-up resistors (4,7 k $\Omega$ ), see figure 14. The signals are referenced to GND<sup>(1)</sup>. See also Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275.

## BUSY Status

The BUSY status is available as the BUSY OUT signal at pin (04). When the BUSY status is set, the BUSY OUT signal is HIGH. See also Chapter 6.4.3 "List Execution Status", page 98. The signal is referenced to GND<sup>(1)</sup>.

## 2-Bit Digital Input Port

The RTC6 PCIe Board provides a 2-bit digital input (DIGITAL IN1 and DIGITAL IN2) and a buffered 2-bit digital output (DIGITAL OUT1 and DIGITAL OUT2).

For technical data see Chapter 14 "Technical Specifications of the RTC6 PCIe Board", page 824.

For programming the input port see Chapter 9.2.2 "2-Bit Digital Input Port", page 274.

## 2-Bit Digital Output Port

The RTC6 PCIe Board provides a buffered 2-bit digital output (DIGITAL OUT1 and DIGITAL OUT2).

For technical data see Chapter 14 "Technical Specifications of the RTC6 PCIe Board", page 824.

For programming the output port see Chapter 9.1.3 "2 Bit Digital Output Port", page 269.

## 12-Bit Analog Output Port 1 and 2

The RTC6 PCIe Board provides two 12-bit analog output ports, ANALOG OUT1 and ANALOG OUT2<sup>(2)</sup>.

For technical data see Chapter 14 "Technical Specifications of the RTC6 PCIe Board", page 824.

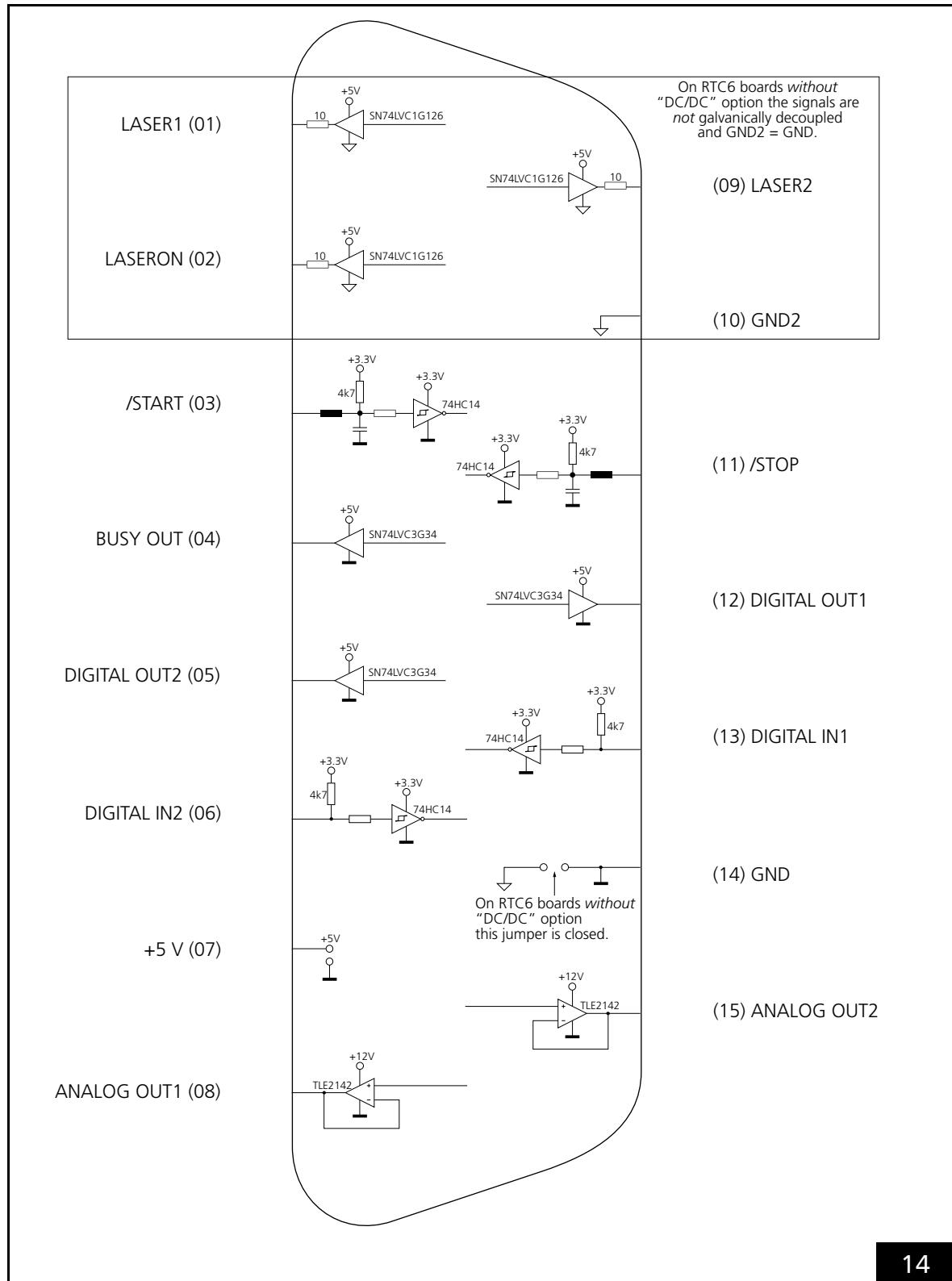
For programming the outputs, see Chapter 9.1.4 "12-Bit Analog Output Ports", page 269.

## Input and Output Wiring

The input and output wiring of the LASER connector is shown in figure 14.

(1) See footnote on page 62.

(2) The ANALOG OUT2 signal is also available by the MARKING ON THE FLY socket connector, see Section "Analog Output Port", page 71.



LASER connector, see also [figure 13](#): input/output wiring.  
See also [Section "Option "DC/DC Converter""](#), page 30.

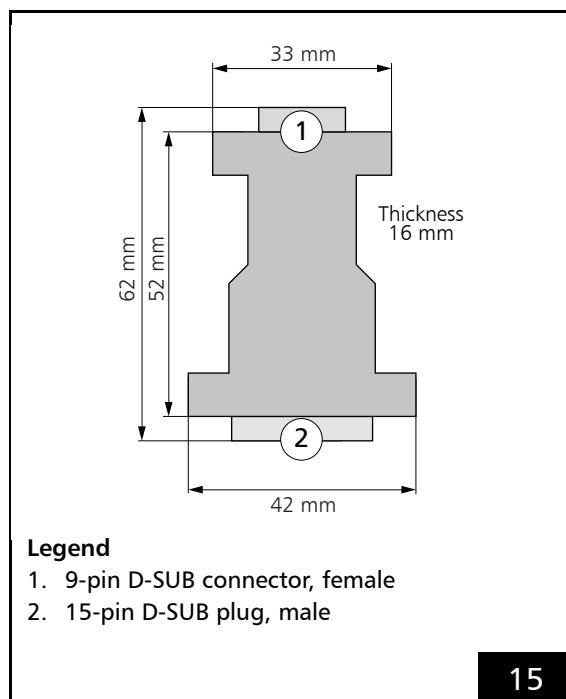
### Laser Adapter (Accessory)

The SCANLAB laser adapter (accessory; #114773) is plugged into the 15-pin LASER connector of the RTC6 PCIe Board. Then its 9-pin D-SUB connector (female) provides the same signals and pin-out as the RTC4 9-pin LASER connector.

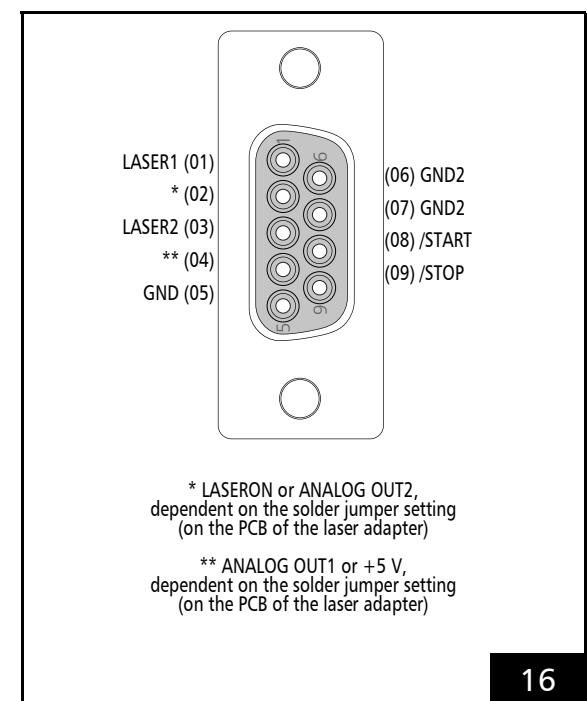
#### Notes

- The laser adapter can *not* be directly plugged into the RTC6 PCIe Board if an XY2-100 converter is already plugged in.
- The BUSY OUT signal, 2-bit digital input and 2-bit digital output are *not* available at the laser adapter's 9-pin D-SUB connector.

The dimensions of the laser adapter is shown in figure 15.



The pin-out of the 9-pin D-SUB connector is shown in figure 16.



16

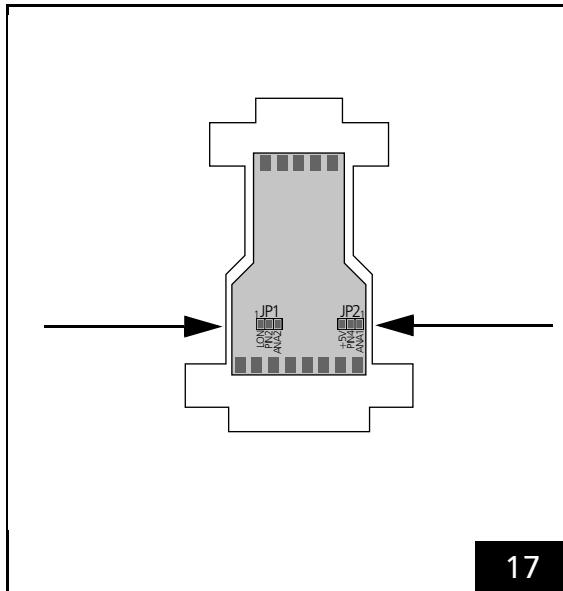
Laser adapter (accessory): pin-out of the 9-pin D-SUB connector, female.

The signals at pins (02) and (04) of the 9-pin D-SUB connector can be selected by two solder jumpers in the laser adapter. To do so, carefully open the laser adapter housing by its four clip latches (for example, using a screwdriver). The solder jumpers JP1 and JP2 are on the PCB of the laser adapter between the two D-SUB connectors.

15

Laser adapter (accessory): dimensions.

The positions of the solder jumpers on the PCB is shown in [figure 17](#).



17

Laser adapter (accessory): position of solder jumper JP1 and JP2 on the printed circuit board.

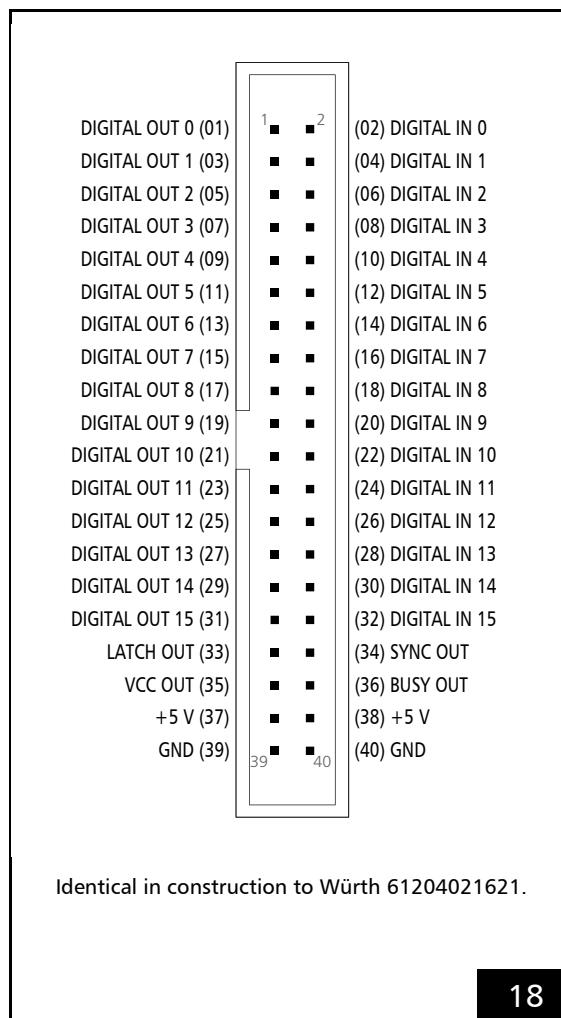
The following table shows the possible jumper settings. Other jumper settings are not allowed.

Solder jumper JP1: signal at pin (02)	Position 1-2 1 LON PIN2 ANA2	Position 2-3 1 LON PIN2 ANA2
LASERON (default configuration)		ANALOG OUT2
Solder jumper JP2: signal at pin (04)	Position 1-2 1 +5V PIN4 ANA1	Position 2-3 1 +5 V
ANALOG OUT1 (default configuration)		

#### 4.6.2 EXTENSION 1 Socket Connector

The EXTENSION 1 socket connector has 40 pins. It is located on the upper side of the RTC6 PCIe Board, see figure 2.

The pin-out is shown in figure 18.



#### Configuring the Output Signal Level

With the solder jumper field A on the lower side of the RTC6 PCIe Board, see figure 3, the level of all output signals at the EXTENSION 1 socket connector (DIGITAL OUT 0...DIGITAL OUT 15, LATCH\_OUT, SYNC\_OUT, BUSY\_OUT, VCC\_OUT) can be configured for 5 V or 3.3 V, see Chapter 2.4.1

"Solder Jumper Field A – Output Signal Level at the EXTENSION 1 Socket Connector Configuration", page 33.

For monitoring purposes, the selected signal level is continuously outputted at pin (35): signal VCC\_OUT. VCC\_OUT is referenced to GND<sup>(1)</sup>.

The maximum current load of the signal is 100 mA.

#### 16-Bit Digital Input and 16-Bit Digital Output

The 40-pin EXTENSION 1 socket connector provides a 16-bit digital TTL input and a buffered 16-bit digital TTL output, see figure 18. This requires the output signals level to be configured by the jumper setting (see section above).

For programming, see Chapter 9.1.1 "16-Bit Digital Output Port", page 268, Chapter 9.2.1 "16-Bit Digital Input Port", page 274, and Chapter 9.3.2 "Conditional Command Execution", page 281.

The input and output signals are referenced to GND<sup>(1)</sup>.

The maximum current load of the output signals is 8 mA.

#### Notes

- EXTENSION 1 socket connectors pin-outs of RTC5 boards, RTC6 PCIe Boards and RTC6 Ethernet Boards are identical.

(1) See footnote on page 62.

### Synchronization of Data Acquisition

If several bits are simultaneously transferred as data words (and not independently from each other) by the 16-bit digital output port or the 16-bit digital input, then the LATCH OUT signal or SYNC OUT signal should be used for synchronization of data acquisition.

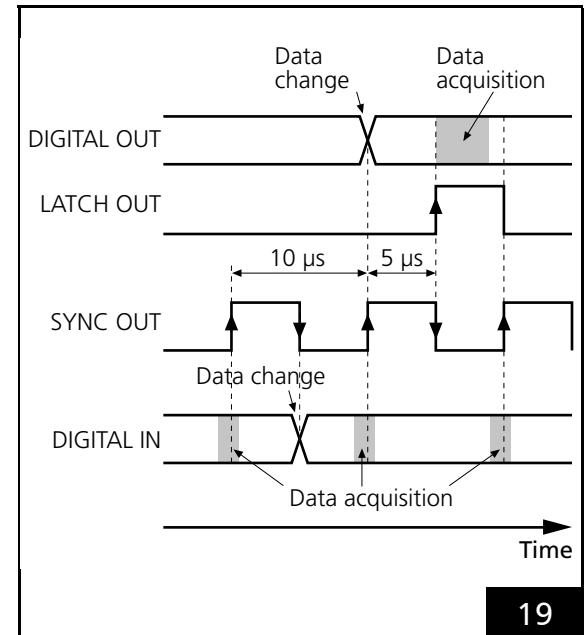
The LATCH OUT signal (a 5 µs pulse, active-HIGH) is outputted at pin (33) as a trigger signal for acquiring the output values of the 16-bit digital output port. The RTC6 PCIe Board automatically generates the LATCH OUT signal when the output value at the 16-bit digital output is written. The output value should be read-out with the rising edge of the LATCH OUT signal. The rising edge occurs 5 µs after the value has been outputted at the 16-bit digital output, see figure 19.

To synchronize data acquisition at the 16-bit digital input, a SYNC OUT signal (a square wave with 5 µs pulse length and 10 µs period) at pin (34) is continuously outputted. Value changes at the 16-bit digital input should be made with a falling edge of the SYNC OUT signal. The DSP of the RTC6 PCIe Board always accepts the currently provided value with the rising edge of the SYNC OUT signal, see figure 19.

The synchronization signals LATCH OUT and SYNC OUT are referenced to GND<sup>(1)</sup>. The maximum current load is 10 mA.

### BUSY Status

The BUSY OUT signal at pin (36) is identical to the BUSY OUT signal at the LASER connector<sup>(2)</sup>, see Section "BUSY Status", page 63.  
The signal is referenced to GND<sup>(1)</sup>.



19

Synchronization of data acquisition by LATCH OUT signal or SYNC OUT signal.

(1) See footnote on page 62.

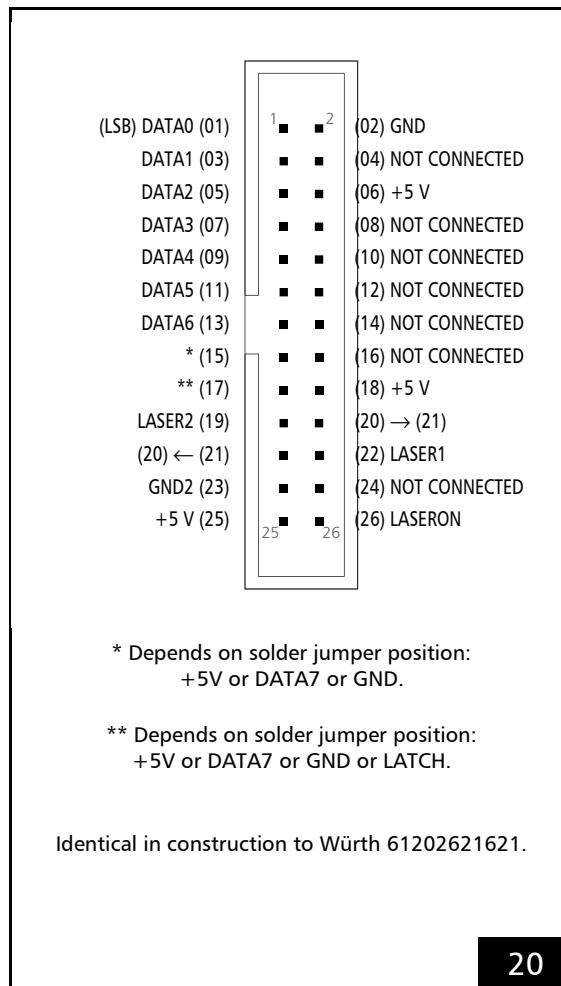
(2) RTC6 Ethernet Board: LASER socket connector.

#### 4.6.3 EXTENSION 2 Socket Connector

The EXTENSION 2 socket connector<sup>(1)(2)</sup> has 26 pins. It is located on the upper side of the RTC6 PCIe Board, see [figure 2](#).

It provides a buffered 8-bit digital output port: DATA0...DATA7.

The pin-out is shown in [figure 20](#).



#### Configuration by Solder Jumpers

Pin (15) of the EXTENSION 2 socket connector is configured by the solder jumper field C whereas Pin (17) is configured by solder jumper field B. Both jumper fields are on the lower side of the RTC6 PCIe Board, see [figure 3](#). For further information, see [Chapter 2.4.2 "Solder Jumper Field B – Pin \(17\) of the EXTENSION 2 Socket Connector Configuration"](#), page 34, and [Chapter 2.4.3 "Solder Jumper Field C – Pin \(15\) of the EXTENSION 2 Socket Connector Configuration"](#), page 35.

#### Notes

- If the DATA7 bit is assigned to pin (15), then the full 8-bit output value is available at the output port (at the odd numbered pins of the EXTENSION 2 socket connector, pin (01)...pin (15)).
- If pin (15) is set to +5 V (HIGH level), then the output values have an offset of 128. That is, the output values are between 128...255.
- If pin (15) is set to GND (LOW level), then the possible output values are 0...127.
- The DATA7 bit can be used for other purposes by assigning it to pin (17).

20

EXTENSION 2 socket connector: pin-out. The pitch of the pins is 2.54 mm.

#### Notes

- Pin (15) and pin (17) are configurable by solder jumpers.

- (1) On the RTC4, the functional corresponding socket connector is labeled LASER EXTENSION, on RTC6 Ethernet Boards EXT. 2.
- (2) RTC6 Ethernet Boards do not provide laser output signals (LASER1 and LASER2) at their (10-pin) EXT. 2 socket connectors.



## Laser Output Signals

The laser output signals LASER1 and LASER2 are identical to the LASER connector and depend on the set laser mode, see [Section "Laser Output Signals", page 62.](#)

The signals are referenced (as with the LASER connector) to GND2.

On RTC6 PCIe Boards with [Option "DC/DC Converter"](#), GND2 and the laser output signals are galvanically decoupled from GND<sup>(1)</sup>.

On RTC6 PCIe Boards without [Option "DC/DC Converter"](#), GND2 are GND identical, see [Chapter 2.3 "Options", page 30.](#)

## 8-Bit Digital Output Port

The buffered 8-bit digital output port (TTL level) is intended for lasers with digital power control (for example, to control the lamp current of a YAG laser).

Of course, it can be used for any other purpose as well. The output is in high-impedance mode until an initial value is assigned to it.

For programming the output see [Chapter 9.1.2 "8-Bit Digital Output Port", page 269.](#)

The **MSB** (DATA7) of the output value can be used for other purposes. To do this, it can be assigned to an other pin on the EXTENSION 2 socket connector (pin (15), pin (17), see above).

If the solder jumper is correspondingly set (see above), Pin (17) outputs a LATCH signal. The RTC6 PCIe Board automatically generates the LATCH signal (a 5 µs pulse, active-HIGH) when the value at the 8-bit digital output port is outputted.

If several bits are simultaneously transferred as a data word (and not independently from each other) by the 8-bit digital output port, then the LATCH signal should be used for synchronization of data transmission.

The value at the 8-bit digital output port should be read-out with the rising edge of the LATCH signal, which is generated 5 µs after the value is written at the 8-bit digital output port, see also [figure 19.](#)

The LATCH signal is referenced to GND<sup>(1)</sup>. Its maximum current load is 10 mA.

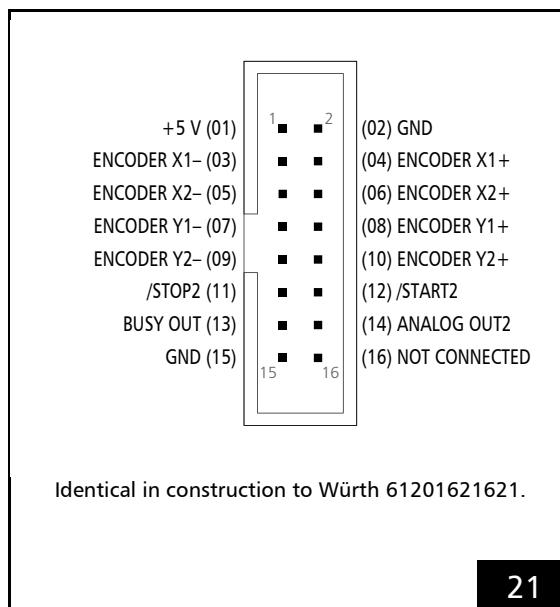
(1) See footnote on [page 62.](#)

#### 4.6.4 MARKING ON THE FLY Socket Connector

The MARKING ON THE FLY socket connector has 16 pins and is located on the upper side of the RTC6 PCIe Board, see [figure 2](#). It provides, among other things, pins for encoder inputs.

If the [Option Processing-on-the-fly](#) of the board is enabled, laser material processing of moving work-pieces (for example, on a moving conveyer belt or rotating disk) can be implemented, see [Chapter 8.6 "Processing-on-the-fly", page 227](#).

The pin-out is shown in [figure 21](#).



#### Encoder Inputs

The RTC6 PCIe Board provides two encoder inputs ENCODER X and ENCODER Y. Each of the encoder inputs are designed for a pair (1, 2) of standardized RS-422 differential signals. See also [Section "Inputs for External Encoder Signals", page 285](#).

#### External Control Signals

The external control signals /START2 and /STOP2 (TTL active-LOW) are referenced to GND<sup>(1)</sup>. Both input signals are connected internally to +3.3 V by pull-up resistors. See also [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275](#).

#### Analog Output Port

The 12-bit analog output port ANALOG OUT2 at pin (14) is identical to the ANALOG OUT2 output port at the LASER connector, see also [Section "12-Bit Analog Output Port 1 and 2", page 63](#). The signal is referenced to GND<sup>(1)</sup>.

#### BUSY Status

The BUSY OUT signal at pin (13) is identical to the BUSY OUT signal at the LASER connector, see [Section "BUSY Status", page 63](#). The signal is referenced to GND<sup>(1)</sup>.

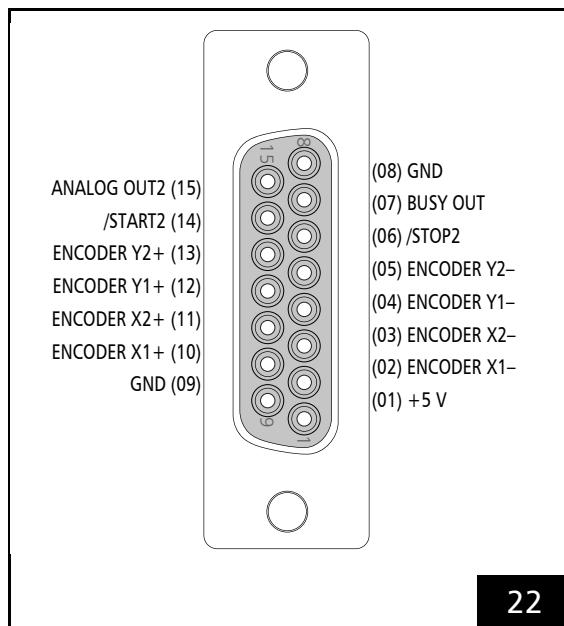
21

MARKING ON THE FLY socket connector: pin-out. The pitch of the pins is 2.54 mm.

(1) See footnote on [page 62](#).

### Slot Cover (Accessory)

SCANLAB recommends using an additional slot cover for using the inputs and signals of the MARKING ON THE FLY socket connector. A suitable slot cover with a 15-pin D-SUB connector (female) is available from SCANLAB. The pin-out is shown in figure 22.



22

MARKING ON THE FLY slot cover (accessory): pin-out of the 15-pin female D-SUB connector.

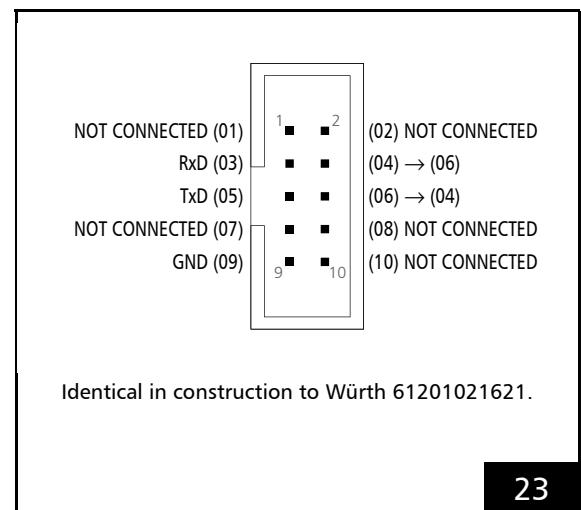
### 4.6.5 RS232 Socket Connector

The RS232 socket connector has 10 pins and is located on the upper side of the RTC6 PCIe Board, see figure 2.

The RS232 socket connector is a hardware interface intended for bidirectional data exchange.

By `uart_config` (as of DLL 612), the RS-232 interface can be configured<sup>(1)</sup>. `uart_config` returns the actual set Baud rate.

The pin-out is shown in figure 23.



23

RS232 socket connector: pin-out. The pitch of the pins is 2.54 mm.

Max. input voltage range: -25 V...+25 V.

Max. output voltage range: -13 V...+13 V.

All signals are referenced to GND<sup>(2)</sup>.

The baud rate (default: 9600 baud) can be set with `rs232_config` (to 300...115.200 Baud). Other parameters cannot be altered (data bits: 8, start bits: 1, stop bits: 1, parity: none).

For outputting data by the RS-232 interface, see Chapter 9.1.6 "RS-232 Interface", page 273.

For reading-in data by the RS-232 interface, see Chapter 9.2.3 "RS-232 Interface", page 274.

(1) Alternatively, the older `rs232_config` can be used which does not return a Baud rate. With < DLL 612, the maximum Baud rate is limited to the range [300...115200].

(2) See footnote on page 62.

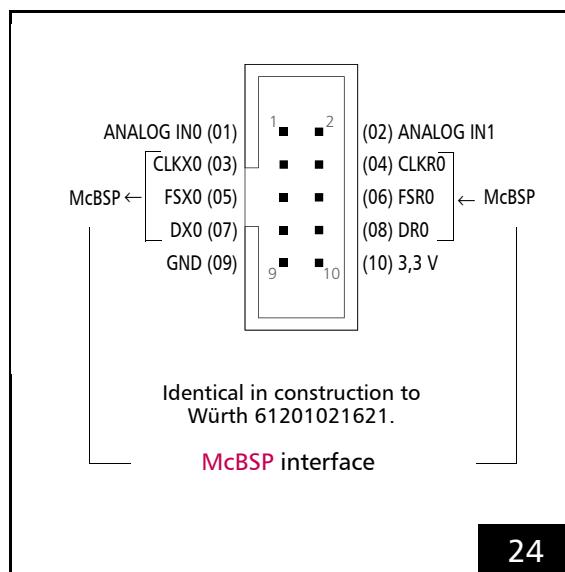
#### 4.6.6 McBSP/ANALOG Socket Connector

The McBSP/ANALOG socket connector<sup>(1)</sup> has 10 pins. It is located on the upper side of the RTC6 PCIe Board, see figure 2.

The McBSP/ANALOG socket connector is a hardware interface that

- is designed for bidirectional data exchange (McBSP), see [Section "McBSP Interface", page 73](#) and
- furthermore, provides two 10 V analog inputs: ANALOG IN0 at pin (01) and ANALOG IN1 at pin (02), see [Section "Analog Input Ports", page 76](#).

The pin-out is shown in figure 24.



24

McBSP/ANALOG socket connector: pin-out. The pitch of the pins is 2.54 mm.

#### McBSP Interface

The [McBSP interface](#), see [figure 24](#), is initialized by [load\\_program\\_file](#) with `DataDelay = 1` and 8 MHz clock frequency.

Intended for [McBSP](#) are pin (03), (05), (07) (outgoing signals) and pin (04), (06), (08) (incoming signals).

The following signals are continuously outputted after [load\\_program\\_file](#):

- the clock signal at pin (03)
- every 10 µs a frame synchronization signal FSX at pin (05)
- a data signal DX at pin (07) (Default: SampleY|SampleX), see [set\\_mcbsp\\_out](#)

The [McBSP interface](#) can be integrated (by using corresponding commands) into applications for various purposes:

- As an alternative to encoder signals, the position of a moving workpiece can be directly integrated, see also [Chapter "Correction via McBSP Interface", page 230](#). Prerequisite: Option [Processing-on-the-fly](#), see [Chapter 2.3 "Options", page 30](#).
- As an alternative to matrix and offset commands, coordinate transformations can be transmitted and integrated to align a workpiece with controllable timing ([Online Positioning](#)), see also [Chapter 9.3.4 "Synchronization and Online Positioning by McBSP Signals", page 286](#).
- With [set\\_multi\\_mcbsp\\_in](#) you can transfer not only a 3D fly correction with position information but also laser power and other parameters, see also [Chapter "Correction via McBSP Interface with Additional McBSP Input", page 231](#). Prerequisite: Option [Processing-on-the-fly](#), see [Chapter 2.3 "Options", page 30](#).
- Permanent data output in 10 µs cycles. With [set\\_mcbsp\\_out](#) and [set\\_mcbsp\\_out\\_ptr](#) it can be selected which data signals are outputted.

Information on using the [McBSP interface](#) for data output and to feed in data can be found in [Chapter 9.1.7 "McBSP Interface", page 273](#), and [Chapter 9.2.4 "McBSP Interface", page 274](#).

(1) With RTC5 boards referred to as SPI /I2C socket connector.

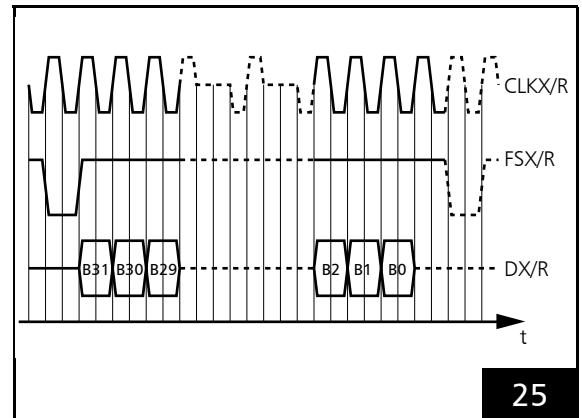
**mcbsp\_init** allows setting a data delay for the **McBSP** data transmission independently for the transmitter and receiver. Possible values range from 0 to 2 (default: 1).

All signals are referenced to GND<sup>(1)</sup>.

#### RTC6 PCIe Board as Transmitter

The following specifications apply to CLKX0, FSX0 and DX0:

- Signal level 3.3 V TTL
- **McBSP** mode:
  - Single phase frame
  - Single element per frame
  - 32 bits per element
  - DataDelay N bit
- The timing diagram of the **McBSP** signals is shown in [figure 25](#). A frame synchronization signal (active-LOW) is generated upon the rising edge of the clock and held for one clock cycle (1 data bit). The 32 data bits are transmitted after XDelay clock cycles at the rising edges of the clock and in the sequence Bit #31...Bit #0. The transmission frequency is by default 8 MHz (4 µs per data word). Alternatively, a value between 4 MHz and 16 MHz can be set by [set\\_mcbsp\\_freq](#).



Timing diagram of **McBSP** signals at 1 bit data delay  
(default: XDelay = RDelay = 1).

(1) See footnote on [page 62](#).



### RTC6 PCIe Board as receiver

The following specifications apply to CLKR0, FSR0, DRO:

- Signal level 3.3 V or 5 V TTL.
- McBSP mode:
  - Single phase frame
  - Single element per frame
  - 32 bits per element
  - DataDelay N bit
- The timing diagram of the McBSP signals is shown in [figure 25](#). The frame synchronization signal (active low) generated upon a rising edge (of an external clock signal) is detected upon the clock's next falling edge (trailing edge of the external clock pulse). The duration of the frame synchronization signal is irrelevant. After RDelay clock cycles the 32 data bits are detected with each additional falling edge of the clock signals in the sequence Bit #31...Bit #0, provided they are transmitted with rising edges.

Take account of the following information:

- The McBSP interface always ignores the first frame synchronization signal after a [load\\_program\\_file](#) or [mcbsp\\_init](#), so supplied data is not transmitted. If necessary, a dummy value should be initially sent to the interface. You can use [read\\_mcbsp](#) to check for successful transmission.
- The bit frequency (receiving frequency) is exclusively determined by the incoming clock pulses and has a maximum limit of 16 MHz.
- The last data bit (Bit #0) must be followed by transmission of at least one additional external clock cycle to ensure that the interface's DSP side acquires and buffers the data word. Simultaneously with this clock cycle, you can already initiate another new transfer by a frame synchronization signal.

- After a [load\\_program\\_file](#), the McBSP data is internally permanently acquired and buffered as soon as (new) data is transmitted. Newer data words overwrite older ones.
  - After [load\\_program\\_file](#) and/or after activation of Processing-on-the-fly correction by [set\\_fly\\_x\\_pos](#), [set\\_fly\\_y\\_pos](#) or [set\\_fly\\_rot\\_pos](#), the input values are stored to internal memory location 0.
  - After activation of an [Online Positioning](#) the input values are stored to internal memory locations 1 or 2
  - For a “[Local Online Positioning](#)” by [set\\_mcbsp\\_x](#), [set\\_mcbsp\\_y](#) and/or [set\\_mcbsp\\_rot](#) or [set\\_mcbsp\\_matrix](#), see Section “[Configuring Local Online Positioning](#)”, [page 215](#)
  - For a “[Global Online Positioning](#)” by [set\\_mcbsp\\_global\\_x](#), [set\\_mcbsp\\_global\\_y](#) and/or [set\\_mcbsp\\_global\\_rot](#) or [set\\_mcbsp\\_global\\_matrix](#), see Section “[Configuring Global Online Positioning](#)”, [page 217](#)
- The most recent fully transferred values can be queried from a corresponding memory location by [read\\_mcbsp](#).
- After activation of Processing-on-the-fly correction by [set\\_mcbsp\\_in](#) or [set\\_mcbsp\\_in\\_list](#), the input values are transferred to internal memory locations 0 and 3.
- After activation of Processing-on-the-fly correction by [set\\_multi\\_mcbsp\\_in](#) or [set\\_multi\\_mcbsp\\_in\\_list](#), the input values are transferred to internal memory locations 0 through 3.



## Analog Input Ports

The McBSP/ANALOG socket connector (RTC6 Ethernet Boards: SPI/ANA/UART socket connector) provides two analog input ports<sup>(1)</sup>:

- ANALOG IN0
- ANALOG IN1

After **read\_analog\_in** has been called the first time, voltages that are applied there are:

- automatically converted endlessly (duration approx. 0.3 ms)
- transferred to the **DSP** as 12-bit digital values

By the control command **read\_analog\_in**, the present values can be queried at any time.

### Specifications

- Input voltage range: 0 V...10 V
- Input impedance: > 5 kΩ
- ADC resolution: 12 bit

The input signals are referenced to GND<sup>(1)</sup>.

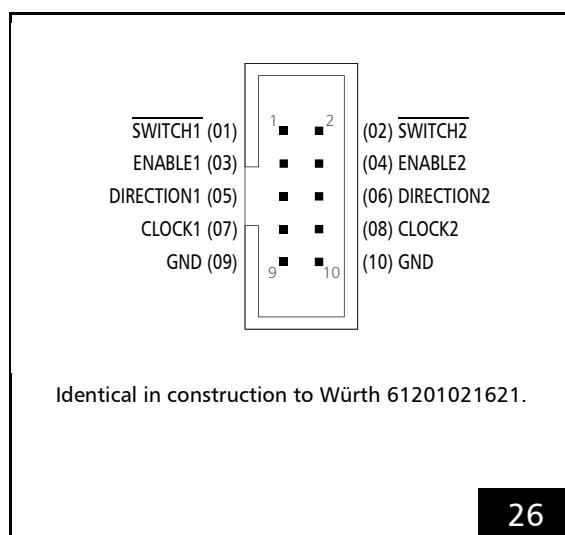
(1) As with RTC5 PCI Express Boards (but not with RTC5 PCI Boards).

#### 4.6.7 STEPPER MOTOR Socket Connector

The STEPPER MOTOR socket connector has 10 pins and is located on the upper side of the RTC6 PCIe Board, see [figure 2](#).

At the STEPPER MOTOR socket connector, signals for controlling up to two stepper motors can be outputted.

The pin-out is shown in [figure 26](#).



STEPPER MOTOR socket connector: pin-out. The pitch of the pins is 2.54 mm.

All signals are referenced to GND<sup>(1)</sup>.

The output signals (ENABLE, DIRECTION and CLOCK) are TTL level signals (5 V).

For each step that is to be carried-out the RTC6 PCIe Board generates an active-HIGH 5 µs pulse as CLOCK signal. The CLOCK signal is permanently LOW between these pulses.

With the ENABLE signals a user program can, for example, switch the motor current on and off.

The DIRECTION signals can set the direction and each CLOCK pulse can be used to execute a single step.

The two SWITCH inputs serve to connect end switches. The RTC6 PCIe Board detects an end switch position by a LOW level (active low). Internally the SWITCH inputs are connected to +3.3 V by 10 kΩ pull-up resistors.

For programming the stepper motor signals, see [Chapter 9.1.5 "Controlling Stepper Motors", page 270](#).

(1) See footnote on [page 62](#).



## 5 Installation and Start-Up

Installation of the RTC6 PCIe Board consists of the following steps:

- (1) Checking the RTC6 PCIe Board jumpers, see [Chapter 5.1 "Checking Jumper Settings", page 78](#)
- (2) Installing the RTC6 PCIe Board, see [Chapter 5.2 "Installing the RTC6 PCIe Board", page 78](#)
- (3) Installing the RTC6 board driver, see [Chapter 5.3 "Installing the RTC6 Board Driver", page 79<sup>\(1\)</sup>](#)
- (4) Installing the RTC6 software, see [Chapter 5.4 "Installing the RTC6 Software", page 80](#)
- (5) When turning on the laser system, observe the safe start-up sequence, see [Chapter 5.5 "Safe Start-up and Shutdown Sequences", page 81](#)
- (6) For conducting a post-installation functionality test, for example, the included HPGL converter program can be used, see [Chapter 5.6 "Functionality Test", page 81](#)

### 5.1 Checking Jumper Settings

SCANLAB ships RTC6 PCIe Boards in various jumper configurations.

- (1) Make sure that the to-be-installed RTC6 PCIe Board has the required jumper configuration. If you need to change the factory settings, proceed as described in [Chapter 2.4 "Jumper Settings and Type Designations", page 32](#).
- (2) Proceed with [Chapter 5.2 "Installing the RTC6 PCIe Board", page 78](#).

### 5.2 Installing the RTC6 PCIe Board

#### Notice!

- Store the board in an electrostatically neutral environment using the supplied anti-static bag.
- Carry out the installation in an area that complies with Electro Static Discharge (ESD) directives. When installing the board, observe the instructions given in the instruction for use of the PC.
- Do not touch the contacts of the board.
- Protect the board from humidity, dust, corrosive vapors and mechanical stress.

Perform the following steps to install the RTC6 PCIe Board in a PC:

- (1) Remove all data media from your PC.
- (2) Shut down the operating system and switch off the PC. Disconnect the PC from the mains.
- (3) Disconnect all peripherals (for example, monitor, mouse, keyboard, printer etc.) from the PC.
- (4) Place the PC on a work table that complies with ESD directives.
- (5) Remove the housing of the PC. Locate a free PCIe slot and remove the slot cover. A height from the slot connector's top of at least 105 mm is required.
- (6) Remove the RTC6 PCIe Board from the antistatic bag. Do not touch the contacts of the board.

(1) Step 3 does not apply to RTC6 Ethernet Boards.

- (7) If you want to use the signals on the RTC6 PCIe Board socket connectors, then attach the appropriate cables. SCANLAB recommends installing additional slot covers with suitable connectors. Then the signals are accessible even when the PC housing is closed. All RTC6 PCIe Board connectors and signals are described in [Chapter 4 "RTC6 PCIe Board – Layout and Interfaces", page 53.](#)
- (8) Install the RTC6 PCIe Board into the PCIe slot. Observe the instructions in the manual of your PC<sup>(1)</sup>.
- (9) Close the PC housing.
- (10) Place the PC at its operational location and connect all peripheral equipment.
- (11) Connect the 9-pin D-SUB connector on the RTC6 PCIe Board – using the XY2-100 converter, if necessary – to the scan system by a data cable, see [Chapter 4.5 "Interfaces to Scan System", page 56.](#)
- (12) Connect the 15-pin D-SUB connector on the RTC6 PCIe Board to the laser by a suitable interface, see [Chapter 4.6 "Interfaces for the Laser and Peripheral Equipment", page 62.](#)
- (13) If necessary, connect one or more of the RTC6 PCIe Board socket connectors (using additional slot covers, see above) to the laser, a handling system or other supplementary equipment of the overall system.
- (14) Check all connections and turn on the PC.

(1) If several master/slave-synchronized RTC6 PCIe Boards are to be used in a PC, then the boards must be connected pairwise with each other by the Master and Slave connectors and installed in preferably (recommended) adjacent PCIe slots, see also [Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 55.](#)

## 5.3 Installing the RTC6 Board Driver

### Notice!

- If on your PC an WDM technology-based RTC3/4/5 board driver
    - is yet installed or
    - was installed and has been removed or
    - you are not sure in this regard:
- (1) Install the RTC6 board driver.  
 (2) Run `ScanlabClassChecker.cmd` as Administrator (part of the delivered software package; see there also the background information in `ReadMe_ScanlabClassChecker.pdf`).  
 Step 2 can be skipped on brand new PCs on which an RTC board driver never has been installed.

To install the RTC6 board driver for Windows 10, 8, 7, proceed as follows:

- After the RTC6 PCIe Board has been installed, start the computer.
- For Windows 7:
- (1) If the "Add Hardware Wizard" does not come up automatically, call it from the Control Panel.
  - (2) In the "Add Hardware Wizard" specify the directory that includes the RTC6 board driver. During installation, the operating system automatically selects the appropriate driver file (32-bit or 64-bit).
  - (3) Run `ScanlabClassChecker.cmd` as Administrator (part of the delivered RTC6 Software Package; see there also the background information in `ReadMe_ScanlabClassChecker.pdf`).



For Windows 10, 8:

- (1) Open the Device Manager to display the device tree. Look for the "PCI device" entry and update its driver by specifying the directory that includes the RTC6 board driver.
- (2) Run `ScanlabClassChecker.cmd` as Administrator (part of the delivered software package; see there also the background information in `ReadMe_ScanlabClassChecker.pdf`).

### Notice!

- The RTC6 PCIe Board does not support power-saving modes that switch off power to the PCIe bus. Accordingly, you must disable standby or sleep modes of the operating system. Particularly disable the Windows sleep mode option (some Windows operating systems enable this option by default). You can use the `SleepMode.cmd` script included in the RTC6 Software Package (under `RTC6 Tools`) to do this.

See also [Section "Notes", page 29](#).

### Notice!

- On some PCs it may happen that an RTC6 PCIe Board is not recognized by the device manager in a certain PCIe slot. Use a different slot.

## 5.4 Installing the RTC6 Software

Additionally to installing the RTC6 board driver, the RTC6 software must be "installed", that is, copied or unzipped on the harddisk of the PC.

RTC6 software for RTC6 PCIe Boards are the following files, see also [Section "Folder RTC6 Tools", page 25](#):

- **RTC6 DLL**
  - `RTC6DLL.dll`  
Win32-based RTC6 dynamic link library
  - `RTC6DLLx64.dll`  
Win64-based RTC6 dynamic link library
- **RTC6 files**
  - `RTC6OUT.out`,  
Program file for the PCIe-DSP
  - `RTC6ETH.out`  
Program file for the Eth-DSP
  - `RTC6RBF.rbf`  
Firmware file for the FPGA
  - `RTC6DAT.dat`  
Binary auxiliary file

The files are included in the RTC6 Software Package. For easy identifying and archiving of different software versions, some of the files are also delivered zipped (the zip file names `RTC6<...>_<Version>.zip` include the version numbers).

To install the RTC6 software, follow these steps:

- Copy `RTC6DLL.dll` (or `RTC6DLLx64.dll`), `RTC6OUT.out` (with RTC6 Ethernet Boards `RTC6ETH.out`), `RTC6RBF.rbf` and `RTC6DAT.dat` (of the desired version) to your PC.  
Note that differing file versions cannot be arbitrarily combined with another (each zip file includes a text file with corresponding version information).

### Notes

- Also provide the necessary correction file(s) (existing \*.ct5 correction files can still be used; do not overwrite customized correction files!).

## 5.5 Safe Start-up and Shutdown Sequences

To assure safety during start-up, turn on the components of your laser system exactly in the following order:

- (1) Turn on the PC containing the RTC6 PCIe Board and start up the control software (initialization of the board).
- (2) Turn on the desired peripheral equipment.
- (3) Turn on the power supply for the scan system.
- (4) Turn on the laser.

When shutting down the laser system, turn off the components exactly in reverse order:

- (1) Turn off the laser.
- (2) Turn off the power supply for the scan system.
- (3) Turn off the peripheral equipment.
- (4) Close the control software and turn off the PC containing the RTC6 PCIe Board.



### Caution!

- While the PC is turned on and off, short-term level variations at the output ports of the RTC6 PCIe Board, for instance short-term arbitrary variations of the laser control signals can occur. Therefore always observe the above listed start-up and shutdown sequences. Otherwise there is the risk that the laser unexpectedly turns on for a short term.
- Always turn on the scan system after the PC and control software and turn it off prior to the PC and control software. Otherwise arbitrary scan movements of the scan system could occur. Always turn on the laser as the last component and turn off the laser as the first component. Otherwise there is the risk that the laser beam might be deflected in an uncontrolled direction.

## 5.6 Functionality Test



### Caution!

- Always turn on the PC and the power supply for the scan head first before turning on the laser. Otherwise there is the danger of uncontrolled deflection of the laser beam. SCANLAB recommends the use of a shutter to prevent uncontrolled emission of laser radiation.

The HPGL conversion program `Hpgl.exe` is supplied for testing control of the scan head, see also [Section "Folder HPGL", page 25](#). This program lets you load graphics files in Hewlett Packard HPGL format (vector graphic plotter files `*.plt`) for transfer to the RTC6 PCIe Board.

- (1) Copy `Hpgl.exe` and the supplied `*.plt` files to the same directory as the `RTC6DLL.dll`, the `RTC6` files and correction file(s).
- (2) Start `Hpgl.exe`.
- (3) Under `Options | Correction` select a correction file and under `File | HPGL-File` select a plot file (demo file).
- (4) To start output `Mark | Start Marking`.



## 5.7 User Programs and Demo Programs

The DLLs for RTC6 user programs ([RTC6DLL.dll](#), [RTC6DLLx64.dll](#)) support the RTC6 PCIe Board under 32-bit and 64-bit Microsoft operating systems Windows 10, 8, 7. The DLLs provide all necessary functions for operating the RTC6 PCIe Board.

Programming of user programs is described in detail in [Chapter 6 "Developing RTC6-User Programs"](#), page 83. [Chapter 6.2 "Initialization and Program Start-Up"](#), page 85 shows how to import the functions of the [RTC6 DLL](#) into user programs, if they are written in Pascal, C, C++ or C#.

On a 64-bit operating system, the 64-bit variant of the RTC6 board driver supports function calls from Win64 user programs as well as from Win32 user programs.

Therefore, existing Win32 user programs for the RTC6 PCIe Board are able to execute even on 64-bit systems, if the included Win32-based DLL file [RTC6DLL.dll](#) is used.

For Win64 user program, the Win64-based DLL file [RTC6DLLx64.dll](#) is included in the software package. In case a user program utilizes implicit linking to the [RTC6DLLx64.dll](#), it must be linked with the Win64-based import library [RTC6DLLx64.lib](#).

To help software developers get started, some example codes are shown in:

- [Chapter 6.2.5 "Example Code"](#), page 90
- [Chapter 6.8.3 "Example Code"](#), page 122
- [Chapter 7.1.4 "Example Code"](#), page 131

### Notice!

- Carefully check your user program before running it. Programming errors can cause a system breakdown. In this case, neither the laser nor the scan head can be controlled.

## 6 Developing RTC6-User Programs

### 6.1 RTC6 Software Concept Basics

#### 6.1.1 Controlling Scan Systems and Lasers – An Introductory Example

The SCANLAB RTC6 PCIe Board and its related software are designed for controlling scan systems and lasers.

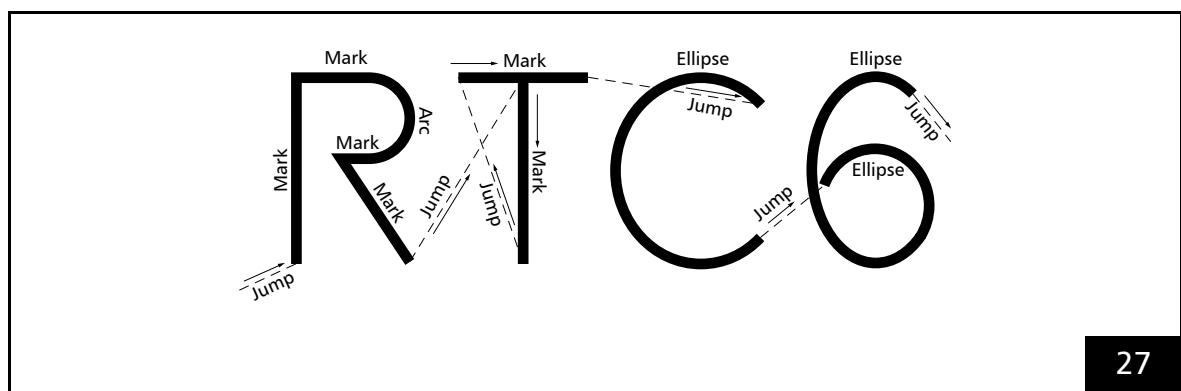
To illustrate the principle of operation, [figure 27](#) shows a simple laser marking sample<sup>(1)</sup>. The lettering is made up of straight line segments (vectors) and arc segments. The [RTC6 DLL](#) provides a set of jump, mark, arc and ellipse commands for laser processing along such segments. Each of these commands describes one vector or arc. Additional commands are available for controlling the laser during the marking process.

The RTC6 PCIe Board processes the commands it receives and precisely transmits the required marking signals to the scan head (using a pre-defined 10 µs time raster) and to the laser. The scan head's galvanometer scanners accurately position their deflection mirrors in synchronization with the incoming control signals.

Appropriate RTC6 PCIe Board commands also allow querying the current scan head status.

For laser control, the RTC6 PCIe Board provides various analog and digital signal outputs freely available.

(1) In this manual, laser marking is mentioned only as an example of the many possible laser materials processing applications.



A laser marking sample.



### 6.1.2 Control Commands and List Commands

The RTC6 command set consists of:

- Control commands
- List commands

*Control commands* are executed immediately. They are used, for instance, for initializing, controlling execution of lists, setting some general parameters, or for directly controlling the laser and scan head.

Before *list commands* can be sent to the RTC6 PCIe Board, a control command must define the input pointer to which subsequent list commands are transferred. This corresponds to opening a list, see [Chapter 6.4.1 "Loading Lists", page 95](#).

List commands sent to the RTC6 PCIe Board afterwards are not executed immediately. They are stored in a list memory first. The commands from the list memory are only processed in real time after the list has been started.

List commands include jump commands, [\[\\*\]mark\[\\*\]](#) commands and arc commands, as well as commands for setting various scanning parameters such as laser power, jump speed and mark speed.

Some of the list commands are so called *short list commands*. These do not require the full 10 µs clock period for command execution. They are executed along with the next list command, one directly after the other within a single 10 µs clock period. A control command cannot be executed in between. The total list execution time is thereby reduced.

Short list commands include [list\\_jump\\_pos](#), [list\\_call](#) etc. With [Polylines](#), for example, the laser power can be set individually for each vector by the short list command [write\\_da\\_x\\_list](#) without interrupting the [Polyline](#) (the laser remains on).

Some commands exist in two versions: as a list command and as a control command. Among these dual-version commands are the I/O commands.

All RTC6 commands are described in detail in [Chapter 10 "RTC6 Commands"](#).

An overview is provided in [Chapter 10.1 "Overview", page 288](#).

## 6.2 Initialization and Program Start-Up

To use the commands and functions of the RTC6 PCIe Board in a user program:

- A complete installation must have been carried out – this includes the RTC6 hardware, RTC6 board driver and RTC6 software, see [Chapter 5 "Installation and Start-Up", page 78](#)
- The desired **RTC6 DLL** (Win32- or Win64-based) must be linked to the user program, see [Chapter 6.2.1 "DLL Calling Convention", page 85](#)
- The **RTC6 DLL** functions must be imported to the user program, see [Chapter 6.2.2 "Importing Commands", page 85](#)

At the beginning of each user program, commands must be called:

- that initialize the **RTC6 DLL** for the calling user program and assigns access rights for the installed RTC6 PCIe Boards, see [Chapter 6.2.3 "Initializing the RTC6 DLL and RTC6 Board Management", page 87](#)
- that initialize the installed RTC6 PCIe Boards, see [Chapter 6.2.4 "Start of RTC6 PCIe Board Operation", page 88](#)

Only afterward can the user program load the desired command lists into the RTC6 list memory and start processing.

These steps are described individually in the following chapters. They are shown by an example program in [Chapter 6.2.5 "Example Code", page 90](#).

### 6.2.1 DLL Calling Convention

Link the user program to the [RTC6DLL.dll](#)/  
[RTC6DLLx64.dll](#)<sup>(1)</sup>.

The DLL calling convention is `stdcall` for Win32 and `fastcall` for Win64.

The structure alignment is 4 byte for Win32 and 8 byte for Win64.

### 6.2.2 Importing Commands

To facilitate importing the commands of the **RTC6 DLL** into a C, C++, C# or Pascal user program, the RTC6 Software Package contains corresponding utility files.

#### Notice!

- Some RTC6 commands can only be *completely* executed with appropriate options, see also [Chapter 2.3 "Options", page 30](#).
- **get\_RTC\_version** provides information about the options installed on your RTC6 PCIe Board.

(1) See [Chapter 5.4 "Installing the RTC6 Software", page 80](#).

## Pascal

Use the file `RTC6Import.pas` as a unit and call the RTC6 commands you need, like

```
goto_xy(1000, 2500);
for performing a jump to location 1000, 2500.
```

## C

In C, you can choose either implicit linking – also known as static load or load-time dynamic linking – or explicit linking – also known as dynamic load or run-time dynamic linking.

### Implicit Linking

To accomplish implicit linking, include the header file `RTC6impl.h` and link with the (C) import library `RTC6DLL.lib` (for Win32 user programs or with `RTC6DLLx64.lib` for Win64 user programs) for building the executable.

Call the RTC6 commands you need like

```
goto_xy(1000, 2500);
for causing a jump to location 1000, 2500.
```

### Explicit Linking

To accomplish explicit linking, include the header file `RTC6expl.h`. Before calling any RTC6 command, initialize the **RTC6 DLL** by calling the function `RTC6open` (which is defined in the file `RTC6expl.c`). When you have finished the RTC6 session, close the **RTC6 DLL** by calling the function `RTC6close` (also defined in the file `RTC6expl.c`).

For building the executable, link with the file `RTC6expl.obj`, which you can generate from the `RTC6expl.c` source code.

Call the RTC6 commands you need, like

```
goto_xy(1000, 2500);
for causing a jump to location 1000, 2500.
```

## Differences between Implicit and Explicit Linking

	Implicit Linking	Explicit Linking
Necessary Files	<code>RTC6impl.h</code> or <code>RTC6impl.hpp</code> , <code>RTC6DLL.lib</code> or <code>RTC6DLLx64.lib</code>	<code>RTC6expl.h</code> , <code>RTC6expl.c</code>
Advantages	Easiest linking method.	Eliminates the need to link the user program with an import library.
Disadvantages	Users need to link the user program to a compiler-specific import library.	Users need to initialize ( <code>RTC6open</code> ) and close ( <code>RTC6close</code> ) the <b>RTC6 DLL</b> .

## C++

If you want to use references instead of pointers for returning values to function parameters in C++ (for instance `UINT&Pos` instead of `UINT*Pos`), use the files `RTC6impl.hpp` instead of `RTC6impl.h`. Otherwise the command import is realized as in C (see above).

## C#

For implicit linking of the **RTC6 DLL** in C# the wrapper class is provided. It also supports the 'Any CPU' option for simultaneous usage with 32-bit and 64-bit programs.

### 6.2.3 Initializing the RTC6 DLL and RTC6 Board Management

As many RTC6 PCIe Boards as the PCIe bus allows can be operated simultaneously in one PC. However, the **RTC6 DLL** internal card management can manage a maximum of 255 RTC6 PCIe Boards and RTC6 Ethernet Boards at the same time, see [Chapter 15.5.3 "About the RTC6 Board Management", page 856](#).

The **RTC6 DLL** allows multi-processing<sup>(1)</sup> as well as multi-threading.

No RTC6 PCIe Board can be simultaneously used by several user programs. Access rights (even if temporary) to existing boards are assigned on an exclusive basis by the **RTC6 DLL**.

Multiple threads of *one* user program can use the same board, but can not send commands to it at the same time (the **RTC6 DLL** automatically serializes the command calls).

The "RTC6 board management", see [Chapter 15.5.3 "About the RTC6 Board Management", page 856](#), must be initiated by **init\_rtc6\_dll** at the beginning of each user program so that an RTC6 PCIe Board can be addressed at all. This even applies, if only *one* RTC6 PCIe Board is to be used by *only one* user program.

**init\_rtc6\_dll** does the following (for details, see the command description):

- Searches for all present RTC6 PCIe Boards
- Establishes the RTC6 board management
- Automatically assigns the user program access rights to the found boards (as long as access rights are not already assigned to another user program)
- Assigns **RTC6 DLL**-internal numbers for all found RTC6 Ethernet Boards (important for multi-board commands)
- Sets one board as the *active* board, which therefore, is the target for non-multi-board commands
- Does *not* search for RTC6 Ethernet Boards. These must be added separately, see [Chapter 15 "Appendix A: The RTC6 Ethernet Board", page 828](#).

(1) Several user programs can run simultaneously.

**acquire\_rtc**, **release\_rtc**, **select\_rtc** allow changing access rights and the active board at run-time.

Usage of several boards is described in [Chapter 6.6 "Using Several RTC6 PCIe Boards in One PC", page 113](#); usage by several user programs is described in [Chapter 6.7 "Usage of RTC6 PCI Express Boards by Several User Programs", page 118](#).

After **RTC6 DLL** initialization by **init\_rtc6\_dll**, the default DLL-operation mode is the **RTC6 Standard Mode**. If required, a different DLL-operation mode can be set (see **set\_rtc4\_mode**, **set\_rtc5\_mode** and **set\_rtc6\_mode**).

The **RTC4 Compatibility Mode** is provided so that user program written for the RTC4 can be processed by the RTC6 PCIe Board (for otherwise unchanged commands<sup>(2)</sup>) without modification. In the command descriptions, notes changes in this regard, see [Chapter 10.2 "RTC6 Command Set", page 301](#), "RTC4→RTC6" row.

See also [Chapter 2.7.2 "Porting RTC4 Source Code to the RTC6 PCIe Board", page 39](#).

The similar applies to the **RTC5 Compatibility Mode** (see RTC5→RTC6 rows od the command descriptions). See also [Chapter 2.9.2 "Adapting RTC5 Source Code for the RTC6 PCIe Board", page 49](#).

(2) Commands having identical names with analogous parameterization and meaning.  
Example: **mark\_abs** (X, Y).

### 6.2.4 Start of RTC6 PCIe Board Operation

At the beginning of every RTC6 user program – after initialization of the [RTC6 DLL](#), see [Chapter 6.2.3 "Initializing the RTC6 DLL and RTC6 Board Management", page 87](#) – it is recommended to carried out the following sequence of steps in order to start RTC6 PCIe Board operation.

- (1) [Initializing the Board, page 88](#)
- (2) [Configuring the Board, page 88](#)
- (3) [Initializing the Scan System Control, page 89](#)
- (4) [Initializing the Laser Control, page 89](#)
- (5) [Loading and Executing Lists, page 89](#)

#### Initializing the Board

- Call [load\\_program\\_file](#). [load\\_program\\_file](#) initializes the RTC6 PCIe Board, performs a [DSP](#) memory check, initializes and configures the list memory, loads the firmware [RTC6RBF.rbf](#), the program file [RTC6OUT.out](#) and the binary auxiliary file [RTC6DAT.dat](#). After execution of [load\\_program\\_file](#), the position output is automatically set to the null position (0|0)<sup>(1)</sup> and laser control is deactivated<sup>(2)</sup>. In order to be able to combine RTC6 files ([RTC6DLL.dll](#)/[RTC6DLLx64.dll](#), [RTC6OUT.out](#)/[RTC6ETH.out](#), [RTC6RBF.rbf](#), [RTC6DAT.dat](#)), they must have certain file versions, see also [Chapter 5.4 "Installing the RTC6 Software", page 80](#). Assorted files versions of the [RTC6 DLL](#) and the RTC6 files [RTC6OUT.out](#), [RTC6RBF.rbf](#) and [RTC6DAT.dat](#) cannot be arbitrarily combined with another. [load\\_program\\_file](#) performs a version compatibility check. If a version error exists (error code [7](#) and [get\\_last\\_error](#) return code [RTC6\\_VERSION\\_MISMATCH](#)), the board is released by [release\\_RTC](#). Thus, it is not available for further commands other than those not requiring granted access rights.

RTC6 PCIe Boards can only be operated if all software files are available with a compatible combination of versions, see [Chapter 5.4 "Installing the RTC6 Software", page 80](#).

If the version check fails and the board is not acquired by another user program, then [load\\_program\\_file](#) can also be used at any time to load a correct program version. After that, the board can be acquired by [select\\_RTC](#) or [acquire\\_RTC](#).

If several RTC6 PCIe Boards are master/slave connected, see [Chapter 6.6.3 "Master/Slave Operation", page 114](#).

#### Configuring the Board

- If necessary, configure the RTC6 list memory by [config\\_list](#). After [load\\_program\\_file](#) the list memory areas "List 1" and "List 2" can each store 4,194,304 list commands. The protected list memory area "List 3" cannot store list commands, see [figure 28](#).

- (1) Center of the image field.
- (2) There are no laser control signals at the corresponding pins (LASERON, LASEROFF, LASER1, LASER2). They are in high-impedance state.



## Initializing the Scan System Control

(1) Use `load_correction_file` to download the necessary correction file(s) to the RTC6 PCIe Board (you can load a correction table before or after `load_program_file` but you should load it *before* `select_cor_table`; you should at least load a 1:1 correction table).

See [Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 222](#), for information about using several different correction tables.

(2) Assign the previously loaded correction table(s) to the scan head control port(s) by `select_cor_table`<sup>(1)</sup>. This causes the intended image field correction to also be applied to the default scanner position (0|0) previously set by `load_program_file` (in some circumstances, image field correction can even shift the null position).

After `load_program_file`, the default assignment is:

- The first scan head control port uses correction table #1.
- The second scan head control port is off.

The desired image field correction becomes active only after a subsequent `select_cor_table`, `select_cor_table_list`, jump command or `Mark` command.

(3) Define the scanner delay mode by `set_delay_mode` (among others, variable polygon delay or constant polygon delay).

(4) If necessary<sup>(2)</sup>, load a table for the variable polygon delay by `load_varpolydelay`.

The remaining settings (scanner delays, jump speed and mark speed) are set by further control commands or list commands.

- (1) `load_correction_file` automatically calls `select_cor_table` with the last assigned table numbers after loading the correction table (if `load_correction_file` follows after `load_program_file`), see [Section "Notes", page 165](#).
- (2) Step 4 can be omitted, if a new `RTC6DAT.dat` is created by `create_dat_file` after loading the table. Then, this table is automatically loaded with the next `load_program_file`.

## Initializing the Laser Control

- (1) Set the laser mode by `set_laser_mode`.
- (2) Set the polarity of the laser control signals appropriate to your laser system by `set_laser_control`.
- (3) Set the FirstPulseKiller length (only for YAG modes) by `set_firstpulse_killer`.
- (4) Set the delay of the Q-Switch signal (only for YAG modes, in particular for YAG Mode 5) by `set_qswitch_delay`.
- (5) Set the stand-by pulses by `set_standby` (in particular for Laser Mode 4 and Laser Mode 6).
- (6) Enable the laser control signals (see `enable_laser`), if they have been suppressed by `set_laser_control`.

The remaining settings (such as laser timing or laser delay) are set by further control commands or list commands, for example, see [Chapter 7.1.4 "Example Code", page 131](#) or [Section "Signals for "Laser Active" Operation", page 175](#).

## Loading and Executing Lists

- (1) Load the list(s).
- (2) If necessary, enable the external start input by `set_control_mode`.

### Notice!

- Carefully check your user program before running it. Programming errors can cause a break down of the system. In this case, neither the laser nor the scan head can be controlled.



### 6.2.5 Example Code

The following C source code for a console user program (environment: Win32) illustrates the programming fundamentals of **RTC6 DLL** and **RTC6 PCIe Board** initialization (for complete demo programs, see [Chapter 11 "Demo Programs", page 819](#)).

Necessary sources: **RTC6impl.h**, **RTC6DLL.lib** (for implicit linking) or **RTC6expl.h**, **RTC6expl.c** (for explicit linking) to link the **RTC6 DLL** to the program, see [Chapter 6.2.1 "DLL Calling Convention", page 85](#). If the operating system does not find the **RTC6DLL.dll** on program startup, it produces a corresponding error message and terminates the program.

```
// System header files
#include <windows.h>
#include <stdio.h>
#include <conio.h>

// RTC6 header file for implicitly linking to the RTC6DLL.dll (for building the executable, also link
// with the (Visual C++) import library RTC6DLL.lib):
#include "RTC6impl.h"
// Alternatively: RTC6 header file for explicitly linking to the RTC6DLL.dll
// (for building the executable, link with the file RTC6expl.obj, which you can generate
// from the RTC6expl.c source code):
//#include "RTC6expl.h"

void __cdecl main( void*, void* )
{

    // only for explicitly linking:
    // if ( RTC6open() ) // error detected, RTC6open returns 0 for no error
    // {
    //     printf( "Error: RTC6DLL.dll not found\n" );
    //     terminateDLL();
    //     return;
    // }

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

    UINT ErrorCode;

    // Initializing the DLL (the following command must be called as the first RTC6 command)
    ErrorCode = init_rtc6_dll();

    // Following the init_rtc6_dll command you should include a program code to catch an error during
    // initialization, for example, for the case the desired board is not detected, access is denied
    // or for another error (for example, a version mismatch).
    // See Chapter 6.8.3 "Example Code", page 122.
}
```



```
// Initializing the RTC6 PCIe Board:  
// - Selecting the board number 1 as the active board in this user program.  
// - If desired: Selecting the RTC4 Compatibility Mode as operation mode  
//   (default: RTC6 Standard Mode)  
// - Optional: stopping any list running on RTC6 PCIe Board number 1 (if it has been used  
//   previously by another user program, a list might still be running). load_program_file does this  
//   automatically itself. Otherwise, load_correction_file cannot be executed before  
//   load_program_file.  
// - Calling load_program_file for initializing the board, loading the program file, etc.  
//   (here also a program code should be included to catch possible errors - for example, file or  
//   system errors - during initialization, see Chapter 6.8.3 "Example Code", page 122).  
// - Clearing all previous error codes  
// - (stop_execution might have created an RTC6_TIMEOUT or RTC6_BUSY error).  
// - Configuring the RTC6 list memory, default: 4,194,304 for list 1 and list 2 each.  
(void) select_rtc( 1 );  
set_rtc4_mode();  
  
// Optional: stop_execution()  
  
ErrorCode = load_program_file( 0 ); // pPath = 0: path of current working directory  
if ( ErrorCode )  
{  
    printf( "Program file loading error: %d\n", ErrorCode );  
    free_rtc6_dll();  
    return;  
}  
reset_error( -1 );  
config_list( 4000, 4000 );  
  
// Following the above initialization code you can include the program code defining  
// the laser scan process.  
// An example code is listed in Chapter 7.1.4 "Example Code", page 131.  
  
// End of main program  
terminateDLL();  
return;  
}  
  
void terminateDLL()  
{  
    printf( "- Press any key to shut down \n" );  
    while( !kbhit() );  
    (void) getch();  
    printf( "\n" );  
    // Close the RTC6 DLL  
    free_rtc6_dll();  
    // only for explicitly linking:  
    // RTC6close();  
}
```



## 6.3 RTC6 List Memory

The RTC6 list memory serves as intermediate storage for list commands.

Before list commands can be transferred to the RTC6 PCIe Board, a control command must define the input pointer to which subsequent list commands are transferred. This corresponds to opening a list, see [Chapter 6.4.1 "Loading Lists", page 95](#).

By additional control commands, the processing of the transferred list commands can be started.

### 6.3.1 Lists and the Protected List Memory Area

The RTC6 list memory offers  $8M = 8,388,608 = 2^{23}$  storage positions in total.

In general, it can be split into three areas. Their sizes are freely configurable.

- Two list memory areas, "List 1" and "List 2". In this manual, these are also simply designated as "lists".
- User-definable is in addition:  
a third "protected list memory area", "List 3".  
This is protected against unintended overwriting  
(by loading normal command lists).

#### "List 1" and "List 2"

In principle, both list memory areas "List 1" and "List 2" can be used in a manner identical to the two list memory areas of the RTC5, RTC4, RTC3 or RTC2 boards, for example, for continuous loading and processing of command lists.

However, the RTC6 PCIe Board has a bigger total list memory and furthermore, the size of each memory area can be freely configured, see [Chapter 6.3.2 "Configuring the RTC6 List Memory", page 93](#).

For list handling, see [Chapter 6.4 "List Handling", page 95](#).

#### "List 3" – Protected List Memory Area

"List 3" is intended for a protected storage of frequently needed list command sequences (as subroutines or character set definitions). It is protected against unintended overwriting (during loading of normal command lists).

There are principally two alternative ways to utilize this protection feature:

- (1) Subroutines can be written to the upper positions of the list area. These subroutines can be subsequently assigned to the protected list memory area "List 3". Such subroutines are called – both initially in the list memory area as well as subsequently in the protected list memory area "List 3" – by `list_call`, specifying an absolute memory position.
- (2) Special commands allow subroutines and character set definitions to be loaded directly in the protected list memory area "List 3" as indexed subroutines or definitions. They can then be called by providing the corresponding index. Indexed character set definitions can, for example, be used in conjunction with `mark_text` for directly marking text.

SCANLAB strongly recommends *not* intermixing usage of these two methods. Otherwise, unintended data loss by overwriting can occur even in the protected list memory area "List 3".

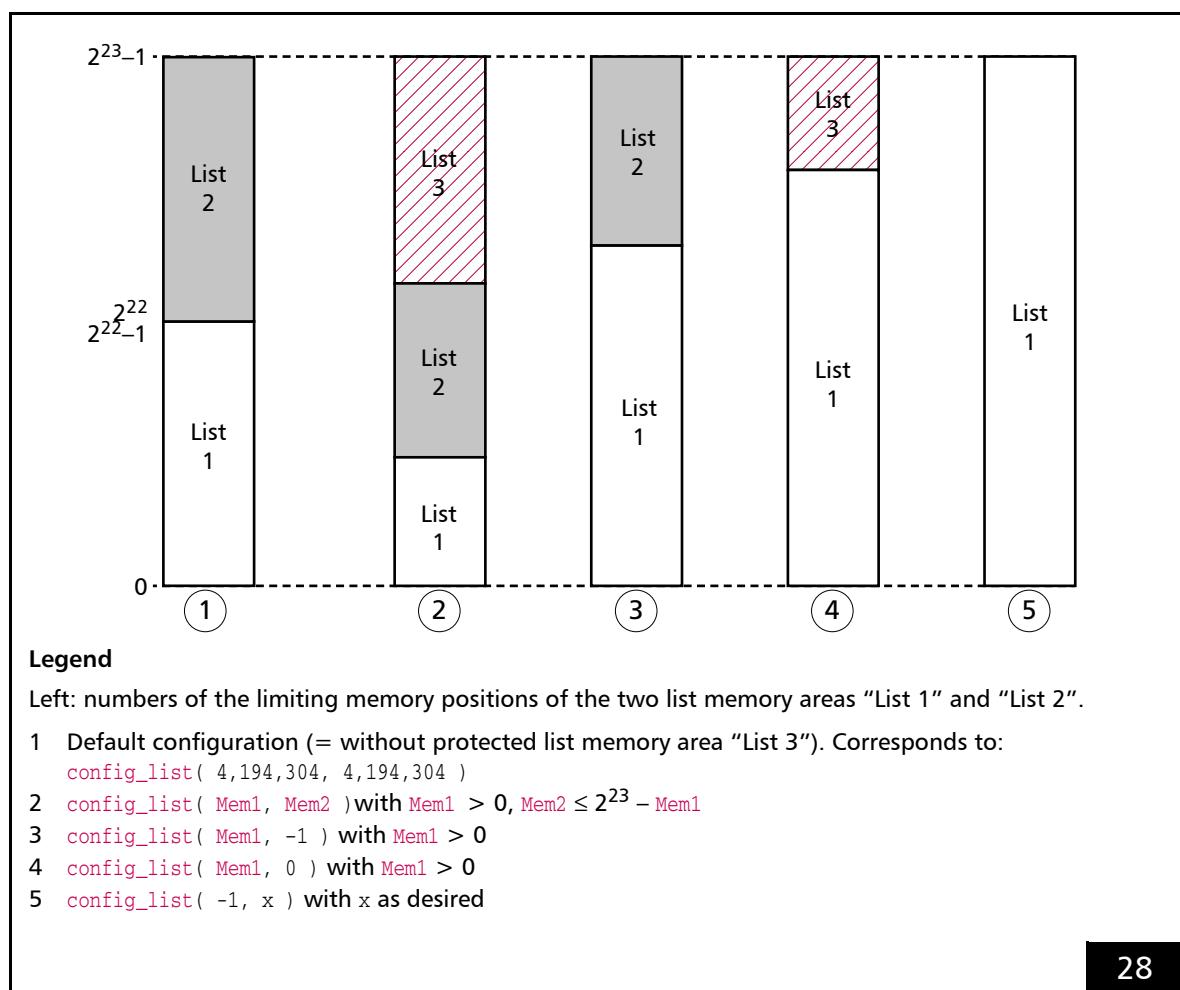
The RTC6 command set includes appropriate conversion commands so that users are not forced to continuously use only one of the two methods. The defining of subroutines and character sets, as well as their management and subsequent conversion options are detailed in [Chapter 6.5 "Structured Programming", page 102](#).

### 6.3.2 Configuring the RTC6 List Memory

The RTC6 list memory offers  $8M = 8,388,608 = 2^{23}$  storage positions in total. After **load\_program\_file** the list memory areas "List 1" and "List 2" can each store 4,194,304 list commands. The protected list memory area "List 3" cannot store list commands, see [figure 28](#).

By **config\_list**, the list memory areas can be reconfigured.

If a user program only needs *one single* list, then the total RTC6 list memory can be treated as a single list memory with a total capacity of  $2^{23}$  storage positions, see Configuration (5) in [figure 28](#).



Examples of allowed list memory configurations.



Generally, during configuration are assigned:

- lower storage position numbers to "List 1"
- medium storage positions numbers to "List 2"
- upper storage position numbers to "List 3"

When configuring the list memory areas, the following must be observed:

- "List 1" must contain at least one storage position
- the total sum of storage positions for "List 1" and "List 2" must not exceed  $2^{23}$ .

Other than that, the sizes of "List 1" and "List 2" can be set as desired. For example, "List 2" can be configured even with 0 storage positions, see configuration (4) and (5) in [figure 28](#).

With every configuration, remaining storage positions (not assigned to "List 1" or "List 2") are automatically assigned the protected list memory area "List 3".

The memory content is not altered by the configuration process. Therefore, repeating the call with differing parameters is nondestructive.

When altering the configuration, you should observe also the following:

- List boundaries should not be moved to within an eventual subroutine.
- The protection of a "List 3" range is removed, if this range is assigned to list memory area "List 1" or "List 2".
- Valid jump addresses specified in jump commands might become invalid if the configuration is altered, see [Chapter 6.5.3 "Jumps", page 110](#).
- After the protected list memory area "List 3" has been made larger, defragmentation might be needed to make the newly assigned memory area usable, see [Section "Index Management and Defragmentation", page 105](#).



## 6.4 List Handling

The two list memory areas "List 1" and "List 2" serve as intermediate storage for the continuous loading and processing of list commands. This chapter describes the control of this data transfer.

### 6.4.1 Loading Lists

"List 1" and "List 2" are enabled to be filled with list commands by `set_start_list_pos`, `load_list` or other control commands (see below). An input pointer is thereby defined for the selected list. This input pointer specifies the memory position to which the subsequent list commands are transferred.

Lists are self-contained memory blocks for list commands. When in the process of list loading the list end is reached without setting the input pointer to another list, then the input pointer is automatically reset to the start of the current list, where loading continues.

An automatic change of the input pointer to another list never occurs, particularly not to the protected list memory area "List 3".

In general, when list commands are loaded into storage positions, any list commands previously stored there are overwritten. This occurs even if they have not yet been processed or are currently being executed. User should make sure not to overwrite commands still needed by the user program (see below).

PCI transfer of the list commands into list memory is buffered to increase the speed for continuous downloads. The buffer is 16 commands in size.

Whenever the buffer is full or when the commands `set_end_of_list`, `list_return`, `set_input_pointer` (and related commands), `execute_list_pos` (and related commands), `auto_change`, `auto_change_pos`, `start_loop` or `release_rtc` are issued, this automatically results in a flush. Thereby, the still buffered list commands are transferred to the list memory.

A flush can be initiated at any time by `set_input_pointer(get_input_pointer())`, even if the buffer is yet "incomplete". This is only necessary in some circumstances when list commands should be processed and list input is not yet finished (for example, with an external start).

#### "Unconditional" Loading

The input pointer is set by `set_start_list`, `set_start_list_1` and `set_start_list_2` to the beginning of the selected list and by `set_start_list_pos` and `set_input_pointer` to the specified address of the selected list.

The next list command is written to this address regardless of the current status of the specified list, see [Chapter 6.4.2 "List Status", page 97](#).

If needed, the current positions of the input and output pointer can be queried by `get_input_pointer` or `get_list_pointer` and `get_status` or `get_out_pointer` - for example, to ensure that not-yet-processed list commands are not overwritten.



## Loading with Protection

The loading process is initialized by **load\_list**, which sets the input pointer to the specified address in the selected list (just like **set\_start\_list\_pos**). However, this occurs only, if the selected list is not currently in use. Alternatively, you can simply let the input pointer be set to a currently non-active or already processed list by **load\_list** (the RTC6 PCIe Board automatically determines the corresponding appropriate list).

The return value of **load\_list** reveals if, and in which list, the loading procedure has been successfully initialized.

Otherwise, the input pointer is set to an invalid position. Then, no further list commands can be input until the input pointer is correctly set back to a valid position again (for example, by repeating **load\_list** with a positive result or **set\_start\_list\_pos**).

This automatically prevents unintentional overwriting of commands that are still to be executed.

**load\_list** is useful in scenarios such as alternating list changes, where you want to wait specifically for a list to be processed, see [Section "Alternating List Changes", page 101](#).

## Terminating Lists

Command lists can be, but need not necessarily be, terminated by a **set\_end\_of\_list**.

However, if an unterminated command list is executed and the output pointer thereby encounters the last possible position in the list, the output pointer automatically resets to the start of the list and processing continues there.

Automatic list changing after a list is processed can only occur, if the list has been terminated by **set\_end\_of\_list**, see [Chapter 6.4.6 "Changing Lists Automatically", page 100](#).

The loading of a **set\_end\_of\_list** does *not* stop the loading procedure itself. Therefore, list commands immediately following a **set\_end\_of\_list** are still loaded into the same list.



### 6.4.2 List Status

Dependent on the command input and output statuses, lists receive particular status values.

To examine the current list status values, the control command **read\_status** can be used – separately for both lists.

- The LOAD status of a list (LOAD1 or LOAD2) indicates that the input pointer is currently in this list. In any case, the LOAD status of the other list is *not* set.
- The READY status of a list (READY1 or READY2) is set when a **set\_end\_of\_list** is written into the list during the loading procedure. The READY status is reset when the LOAD status of the list is newly set.
- The **BUSY** of a list (BUSY1 or BUSY2) status indicates that the output pointer is currently in this list after list execution (of "List 1" or "List 2") has been started. In any case, the **BUSY** status of the other list are then *not* set. The **BUSY** status of a list is reset when **set\_end\_of\_list** is executed (or is alternating set, if automatic list changing has been previously activated). If a list is opened for loading while still being processed, its **BUSY** status still remains set.
- The USED status of a list (USED1 or USED2) is set when a **set\_end\_of\_list** is reached during processing. The USED status is reset when the LOAD status of the list is set.

**set\_start\_list\_pos**(ListNo, Pos ) and execute by **execute\_list\_pos**(ListNo, Pos). If no other list has been active at this moment, then list ListNo has the status USED afterward.

- When interpreting the status values read back by **read\_status**, always take into account the programmed loading or execution processes of the lists (see command description).

#### Notes

- If the list status is queried during processing of a subroutine in the protected list memory area "List 3", then the status is returned of the list "List 1" or "List 2" in which the output pointer most recently resided (typically from where the subroutine has been originally called).
- If list execution is interrupted (by **pause\_list**, **stop\_list** or **set\_wait**), then the above-mentioned status values remain unchanged.
- If list execution is aborted (by **stop\_execution** or an external stop), the USED status is set for both lists (as by initialization). See also **load\_list**(ListNo=3).
- If you want to explicitly set the USED status for a list ListNo (for example, after abortion by **stop\_execution** or an external stop), then load a **set\_end\_of\_list** to a free position Pos of this list by



### 6.4.3 List Execution Status

In addition to the list status values, see [Chapter 6.4.2 "List Status", page 97](#), three status values are provided:

- **BUSY**  
The RTC6 PCIe Board currently processes one of the two lists
- **PAUSED**  
Processing of the list is currently interrupted
- **INTERNAL-BUSY**  
The RTC6 PCIe Board is currently busy with control commands outside of list processing

These status values can be queried by the control command `get_status`.

- The **BUSY** status is set when a list is currently being processed or when a list has been halted by the control commands `pause_list` or `stop_list`. In contrast, the **BUSY** status is not set if a list has been paused by the list command `set_wait` (and is then again set by a subsequent `release_wait`).
- The **PAUSED** status is set when a list has been halted by `pause_list`, `stop_list` or `set_wait`. It is reset by a subsequent `restart_list` or `release_wait`, see also [Chapter 6.4.5 "Interrupting Lists for Synchronization of Processing", page 100](#).
- The **INTERNAL-BUSY** status is set when the RTC6 PCIe Board is busy with executing a control command, which needs more than 10 µs for executing a scan movement (for example, `goto_xy` or possibly `set_offset`) or while a home jump or home return is executed (with `set_wait`, `set_end_of_list` or `release_wait`, if the home jump mode has been previously activated by `home_position` or `home_position_xyz`). The **INTERNAL-BUSY** status and the **BUSY** status cannot be set at the same time.

### Notes

- At the RTC6 PCIe Board, the **BUSY** status is also available as the **BUSY OUT** signal at the
  - 15-pin D-SUB LASER connector, see [figure 13](#)
  - EXTENSION 1 socket connector, see [figure 18](#)
  - MARKING ON THE FLY socket connector, see [figure 21](#)
- Some control commands are ignored (not executed), when the **BUSY** status and/or **INTERNAL-BUSY** status are set (for example, `auto_cal`, `goto_xy`, `load_correction_file`) or – with set **INTERNAL-BUSY** status – are only executed with delay after the **INTERNAL-BUSY** status has been reset again (for example, `execute_list_pos`, `set_offset`).



#### 6.4.4 Starting and Stopping Lists

List processing ("List 1" or "List 2") can be started either by the control command `execute_list` or by an external start signal, see [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275](#).

`execute_at_pointer` can be used to start output of a list at a specified address. If an external start signal is used, see [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275](#), then `set_extstartpos_list` allows definition of a start address for the external start.

The RTC6 PCIe Board starts the execution immediately. Even during 10 µs-clocked execution of the list commands, you can still send control commands to the RTC6 PCIe Board. These are immediately executed without hindering execution of the list.

This is useful, for example, for loading a second list while the first list is being processed (the PC and scan head then work in parallel). However, the second list can only be started after processing of the first list has been finished. During list processing, `execute_list` or an external start signal is ignored.

Execution of a list can also be stopped at any time, for example, for implementing an emergency shutdown. As soon `stop_execution` is called or an external stop is transferred to the RTC6 PCIe Board, the currently executed list is aborted and the "laser active" laser control signals are turned off (but not deactivated).

With `range_checking`, the processing of a list can also be terminated automatically (like with `stop_execution`).

If, during list processing, the list end is reached without encountering a `set_end_of_list`, then processing continues at the beginning of the current list. This is repeated until either `stop_execution` is called or an external stop signal is transferred to the RTC6 PCIe Board.

If during list processing a `set_end_of_list` is reached, then list execution stops - unless `auto_change`, `auto_change_pos` or `start_loop` has been previously called, a list change takes place, see [Chapter 6.4.6 "Changing Lists Automatically", page 100](#). This list change occurs only upon reaching a `set_end_of_list`.

#### Notes

- Lists are not automatically started. Regardless of how many commands are loaded, a list must be started as described in order to be processed.
- To also enable starting and stopping of list execution by external signals, the RTC6 PCIe Board provides corresponding control inputs, see [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275](#).
- `set_pause_list_cond` or `set_pause_list_not_cond` can be used to set a condition for the 16-bit input port so that a `pause_list` is executed instead of `stop_execution` when an external stop is present. In The list execution can then be continued by `restart_list`.

## 6.4.5 Interrupting Lists for Synchronization of Processing

The list command `set_wait` makes it possible to insert numbered break points ("wait markers") into a list. Each break point is associated with a number greater than zero. When the RTC6 PCIe Board reaches a break point during list execution, see [Chapter 6.5 "Structured Programming", page 102](#), output of the list is temporarily interrupted and the laser is switched off.

`get_wait_status` checks whether list processing is currently interrupted at a break point. If processing is interrupted, `get_wait_status` returns the number (`wait_word`) of the break point (otherwise the value zero).

Break points are provided for synchronization purposes. The user program should perform a handling routine for each break point. When that handling routine is finished, processing of the list can be resumed by the control command `release_wait`.

By `set_wait` the `PAUSED` status (queryable by `get_status`) is set and the `BUSY` status is reset. The opposite occurs after a subsequent `release_wait`.

List execution can be interrupted at any desired point in time by the control command `pause_list` (or by the synonym `stop_list`) and resumed by `restart_list`.

By `pause_list` the "Laser active"-laser control signals are suppressed and the scan system remains in the most recently defined state – even if in the middle of microstepping. After a subsequent `restart_list`, the scan system resumes the planned movements (of the current command) and the laser control signals are released again (in general, an interrupted marking cannot be continued without a disruption in the marking result).

By `pause_list` the `PAUSED` status (queryable by `get_status`) is set and is reset by `restart_list`. The `BUSY` status is left unchanged by both commands.

## 6.4.6 Changing Lists Automatically

If the RTC6 list memory is configured for two list memory areas "List 1" and "List 2", see [Chapter 6.3.2 "Configuring the RTC6 List Memory", page 93](#), then a second list can be loaded while the first list is still being processed.

It typically takes substantially longer to process a list than to write it into the memory. Continuous processing of arbitrarily long lists is therefore also possible, if they are divided into command blocks.

Continuous command output, which requires switching between two lists, can be achieved by automatic list changing as described in the following sections.

The commands for automatic list changing only take effect when the next following `set_end_of_list` is executed. That is, automatic list changing after processing a list can only occur if that list has been finished with a `set_end_of_list`. Otherwise, processing resumes at the beginning of the same list.

If the RTC6 list memory is configured for a list memory area ("List 2") to size 0, then all automatic list change commands lead to "List 1" to the specified position.

### One-Time List Change

`auto_change` and `auto_change_pos` activate an automatic, one-time list change between "List 1" and "List 2". After processing of the current list (when `set_end_of_list` is reached), processing of the next list is thereby automatically started.

When using `auto_change` the next list is started at position 0; when using `auto_change_pos` the next list is started at the specified start position (list memory address as an offset to the beginning of the list).



## Alternating List Changes

Another way to achieve continuous command output is by alternatingly repeating output of the two lists.

To do so, **start\_loop** must be called. This causes a continuous, automatic and alternatingly repeating processing of both lists, provided both lists are finished each with **set\_end\_of\_list**.

The alternating processing repeats until **quit\_loop** is called. **quit\_loop** terminates continuous processing as soon as the current list is finished.

The currently non-active list can be newly reloaded even as the other list is processed. This allows continuous alternating output of two lists with not only fixed content, but also constantly new content.

### Notes

- The commands for starting a one-time automatic list change and **start\_loop** to start an alternating list change can be called at any point in time. However, they do not take effect until the next **set\_end\_of\_list** is reached.
- When loading a list while another is being processed, make sure no still-needed commands are thereby overwritten. Useful here is **load\_list**, which only starts loading a list if it is currently not in use or already has been processed, see [Chapter 6.4.1 "Loading Lists", page 95](#).
- Moreover, the currently new list should have made a certain amount of loading progress before the list change occurs. The input pointer should always be adequately ahead of the output pointer (because the PCI transfer of the list commands is buffered, see [Chapter 6.4.1 "Loading Lists", page 95](#), and possibly so-called short list commands can be used, see [Chapter 6.1.2 "Control Commands and List Commands", page 84](#)). Otherwise, "old" commands might be unintentionally executed.
- The RTC6 PCIe Board does not support the RTC4 circular queue mode, see [Chapter 6.5.4 "RTC4-Circular Queue Mode", page 111](#). However, this operating mode can also be effectively replaced using an alternating list change and **load\_list** described above.

## 6.5 Structured Programming

The RTC6 command set supports structured programming and output of list commands by numerous ways to define subroutines and character sets, as well as list commands for controlling program flow.

### 6.5.1 Subroutines

- As list-command sequences, subroutines can principally be located in any part of the list memory.
- Preferably, subroutines should be written to a upper portion of the list memory, see [Section "List 3 – Protected List Memory Area", page 92](#).
- A list boundary should not run through a subroutine.
- A subroutine must be terminated with a [list\\_return](#).
- It can be defined:
  - [Non-Indexed Subroutines](#)
  - [Indexed Subroutines](#)

#### Non-Indexed Subroutines

As with “normal” list-command sequences, non-indexed subroutines are loaded into a list memory area (“List 1” or “List 2”) by list-loading commands (see [Chapter 6.4.1 “Loading Lists”, page 95](#)). Each subroutine must be terminated with a [list\\_return](#). It is called by [list\\_call](#) with a parameter specifying the absolute memory address.

After the subroutine (including the terminating [list\\_return](#) command) has been processed, it is continued with the command that follows the calling position.

#### Notes

- Non-indexed subroutines cannot be written directly to the protected list memory area “List 3”. However, they can be subsequently protected, see also [Section “Subsequent Protection and Conversion of Non-Indexed Subroutines”, page 106](#). For the subroutine to be indexed for this purpose with [set\\_sub\\_pointer](#), however, its start address must be known. Prior to loading a non-indexed subroutine into the list memory area “List 1” or “List 2”, you should therefore always read out the start address by [get\\_input\\_pointer](#).
- Make sure that there is no [list\\_return](#) in a normal list flow or in a body of a subroutine, for which there has not been a corresponding subroutine call. Otherwise, with nested subroutine calls the integrity of the nesting is destroyed. If there is no still active subroutine call, list processing is continued at the absolute position 0.  
If a subroutine begins directly after a [list\\_call](#), [list\\_call\\_abs](#), [list\\_call\\_repeat](#), or [list\\_call\\_abs\\_repeat](#), then the return address is automatically set from `Pos(list_call) + 1` to `Pos(list_return) + 1`. That is, the next processed command is the one which follows after [list\\_return](#) but not the command which follows after [list\\_call](#) (which would process the subroutine once again having an uncorrelated [list\\_return](#)).



## Indexed Subroutines

**load\_sub** assigns a desired index to a subroutine (which is defined by subsequent list commands), and loads it into the protected list memory area "List 3".

An indexed subroutine must be terminated by a **list\_return** (otherwise it is not stored). It is called by **list\_call** (with a parameter specifying the index).

A maximum of 1024 subroutines can be stored.

The management of the indexed subroutines occurs on the RTC6 PCIe Board automatically.

For more information on index management, see [Section "Index Management and Defragmentation", page 105](#).

The memory address of an indexed subroutine can be queried by **get\_sub\_pointer**. With it, the indexed subroutine can be called by specifying the absolute memory address (just as with non-indexed subroutines).

An indexed subroutine is only stored by **load\_sub** under the following circumstances:

- If prior to loading, configuration of "List 1" and "List 2" resulted in a protected list memory area "List 3" of sufficient size. For example, if all memory is assigned to one or both lists, then no indexed subroutines can be stored.
- **get\_list\_space**, if called after a **load\_sub** (but before the terminating **list\_return**), can be used for querying the amount of still-available memory in the protected list memory area "List 3"
- If the indexed subroutine is terminated by a **list\_return**
- If **list\_return** is preceded by no other command for positioning the input pointer (for example, another **load\_sub**, **set\_input\_pointer**, or **set\_start\_list\_pos**)
- If the index is within the valid range (0...1023)

After a **list\_return**, the input pointer becomes invalid. Any subsequent list commands are no longer stored.

Indexed subroutines are written to "List 3" by **load\_sub** commands in order of entry. The starting address is automatically set after the end of the last subprogram.

If an indexed subroutine is stored using an already-existing index, then the prior subroutine with that same index is not overwritten. It remains in the protected list memory area "List 3", though it is no longer indexed. Therefore, it can no longer be called through its index by **sub\_call** (whereas it can be still called through its absolute memory address by **list\_call**).

Use **get\_sub\_pointer** to query whether a subroutine is referenced by a particular index. If no subroutine is referenced, **get\_sub\_pointer** returns the value "-1" (that is,  $2^{32}-1$ ).

To load an indexed subroutine into the protected list memory area "List 3" that is already fully loaded with indexed subroutines, you must first appropriately expand the size of the protected list memory area "List 3" by **config\_list** and then defragment it by **save\_disk/load\_disk**. Note that expanding the size alone is not sufficient, see [Section "Index Management and Defragmentation", page 105](#).

## Notes on Not-Indexed Calls

- Index management or defragmentation, see [Section "Index Management and Defragmentation", page 105](#), can result in a change of the indexed subroutine absolute memory address. SCANLAB therefore advises against calling an indexed subroutine by **list\_call**.



## Rules for Programming

Observe the following guidelines when programming indexed subroutines:

- In an indexed subroutine, `set_end_of_list` is replaced by a `list_nop`.
- Absolute jumps within or out from the protected list memory area "List 3" are ignored during processing, see [Chapter 6.5.3 "Jumps", page 110](#). Therefore, absolute jumps cannot be used in indexed subroutines.
- Relative jumps that exceed the boundaries of an indexed subroutine and a jump command which initiates a jump on the command itself is ignored during execution, too.

## General Information on Calling Subroutines

Nested calls up to a maximum depth of 63 are possible.

When calling with `sub_call` or `list_call`, only relative jump and **Mark command** may be used, if the subroutine execution is to be repeated at different image field places.

To be able to use the absolute jump commands and **Mark commands**, which are often easier to handle, the so-called "AbsCalls" are provided.

### "AbsCalls"

If the subroutines contain only relative vector commands and arc commands, see [Chapter 7.1.1 "Marking with Vector and Arc Commands", page 125](#), the corresponding processes can be repeated at different places in the image field.

With "AbsCalls" the current position is taken over as offset. This offset is then taken into account for all subsequent vector and arc commands in the subroutine.

Nested calls are taken into account when the offset is determined. This can be used, for instance, to define character sets by absolute vectors.

"AbsCalls" from subroutines are made with `list_call_abs` and `sub_call_abs`.

### Conditional Calls

To enable calling of subroutines dependent on external control signals, additional commands are available for conditional branching during program execution, see [Chapter 9.3.2 "Conditional Command Execution", page 281](#).

### Repeatedly Executed Calls

`sub_call_repeat`, `sub_call_abs_repeat`, `list_call_repeat` and `list_call_abs_repeat` can be used to automatically execute the body of a subroutine several times.

## Index Management and Defragmentation

### Index Management

To duplicate, renumber or convert indexed subroutines, **copy\_dst\_src** is provided.

By **copy\_dst\_src**, an indexed subroutine is indexed one more time. **copy\_dst\_src** only alters the corresponding entries in the internal management table and does not modify the list memory content.

No longer needed indices (unneeded entries in the internal management table) can be deleted by **load\_sub** directly followed by a **list\_return**. Here, too, deletion occurs only in the internal management table, while the list commands of the previously indexed subroutine continue to reside in the list memory.

**get\_sub\_pointer** can be used to query whether a subroutine for a particular index exists. If no subroutine exists, **get\_sub\_pointer** returns a “-1” value (that is,  $2^{32}-1$ ).

A true duplicate of an indexed subroutine in the protected list memory area “List 3” can be created (after **copy\_dst\_src**) with **save\_disk/load\_disk**.

Subroutines with multiple indices are thereby written several times to the list memory. Keep this in mind in order to prevent unintended memory overflow of the protected list memory area “List 3”.

**load\_sub** always enters a new indexed subroutine after the indexed subroutine with the highest memory address. Therefore, subroutines that are no longer required and are located in the protected list memory area “List 3” may block memory positions for further indexed subroutines.

For this reason, simply increasing the size of the protected list memory are “List 3” by **config\_list** fails to produce further usable memory for storing additional indexed subroutines (the protected list memory area “List 3” can only be expanded downward, not upward).

### Defragmentation

This situation can be resolved by defragmenting with **save\_disk/load\_disk**.

Thereby, all indexed subroutines and (by **set\_sub\_pointer**) subsequently indexed subroutines are rewritten to the protected list memory area “List 3” (in index sequence, starting at the lowest memory position of the protected list memory area “List 3”).

The now-available upper memory positions can then be used for storing additional indexed subroutines.

### Notes

- Before calling **load\_disk**, be sure the protected list memory area “List 3” is of sufficient size after configuration of “List 1” and “List 2”. Indexed subroutines without sufficient space there are *not* stored by **load\_disk**. **save\_disk** returns the number of stored list commands. No-longer-needed subroutines should previously deleted from the index management by a **load\_sub** which is directly followed by a **list\_return**.
- In some circumstances, index management or defragmentation can alter the absolute memory address of an indexed subroutine. It is therefore not advisable to call an indexed subroutine by **list\_call**.
- **save\_disk** stores subroutines starting from the start address to the first-encountered **list\_return**. Relative jumps are not evaluated. So do not use branches to several **list\_return**. Instead, reclose eventual branches prior to one single **list\_return**.

## Subsequent Protection and Conversion of Non-Indexed Subroutines

Non-indexed subroutines can be (directly) written only to a list memory area ("List 1" or "List 2"), but not to the protected list memory area "List 3".

There are basically two methods to protect non-indexed subroutines subsequently:

### (1) Changing the configuration

The part of the list memory area in which the non-indexed subroutine has been written is assigned to the protected list memory area "List 3" by **config\_list**. The subroutine subsequently protected in this manner remains non-indexed (with unaltered memory address) and can, as before, be called by **list\_call**.

### (2) Converting to indexed subroutines

**set\_sub\_pointer** is used to index a non-indexed subroutine and thus include it in the memory management of the indexed subroutines.

By **save\_disk/load\_disk**, it can subsequently be copied as an indexed subroutine to the protected list memory area "List 3", see [Section "Defragmentation", page 105](#).

With a subsequent call by **sub\_call** via the index, the subroutine in the protected list memory area "List 3" is then started.

If you want to subsequently protect a non-indexed subroutine – either by method 1 or method 2 – then be aware that absolute jumps within and out from the protected list memory area "List 3" are not allowed, see [Chapter 6.5.3 "Jumps", page 110](#).

With converting to indexed subroutines (method 2), also all other programming rules for indexed subroutines must be observed, see [Section "Rules for Programming", page 104](#).

*Always try to use only one of the two methods.* This avoids unintended data loss in the protected list memory area "List 3" by overwriting.

If you begin working with method 1 but later want to also use indexed subroutines: then you should convert all non-indexed subroutines residing in the protected list memory area "List 3" to indexed subroutines using method 2, before you define the first indexed subroutine by **load\_sub**.

When doing so, observe the following Notes.

### Notes

- If method 1 is used and you remove overwrite-protection for a part of the protected list memory area "List 3", then you risk overwriting indexed subroutines or previously protected subroutines.
- Non-indexed subroutines subsequently protected with method 1 can under some circumstances be overwritten by a later **load\_sub** or **load\_disk**.
- **set\_sub\_pointer** links the supplied index with the specified start address, even if an indexed subroutine had already been previously defined for this index. The original indexed subroutine with this index is then no longer indexed and no longer callable by the index.



- If using method 2, you should use it fully. If the **set\_sub\_pointer** alone is executed, then the subroutine is already callable by **sub\_call** and its index, but the subroutine remains unprotected against overwriting. Protection is obtained only after the subroutine is subsequently copied as an indexed subroutine by **save\_disk/load\_disk** into the protected list memory area "List 3".
- **save\_disk** ignores all non-indexed subroutines, even those subsequently protected in the protected list memory area "List 3" by method 1. Be aware that they can be overwritten there by **load\_disk**.
- **save\_disk/load\_disk** automatically replaces unallowed commands (for example,, **set\_end\_of\_list**) with **list\_nop** commands.
- Indexed subroutines repeatedly indexed with **copy\_dst\_src** are duplicated in the list memory at a subsequent **save\_disk/load\_disk**. This can result in a memory overflow in the protected list memory area "List 3".
- Before executing **load\_disk**, be sure the protected list memory area "List 3" is of sufficient size after configuration of "List 1" and "List 2" (**save\_disk** returns the number of stored list commands). An indexed subroutine is *not* stored by **load\_disk** if there is not sufficient memory.
- Conversion of a subroutine by method 2 changes the absolute memory address of the subroutine.

## Unprotecting Subroutines

The protection of a subroutine stored in the protected list memory area "List 3" is removed, if it is assigned by **config\_list** to one of the list memory area "List 1" or "List 2".

The subroutine can then still be called using the same parameters (index or absolute memory address). But it no longer has protection against unintentional overwriting.



## 6.5.2 Character Sets and Text Strings

For marking tasks, it is convenient to use the RTC6 PCIe Board list memory for storing command lists as separate subroutines that define how the scan system should mark the needed characters and/or text strings.

To simplify management of characters and text strings, the RTC6 PCIe Board provides the possibility of storing indexed character definitions and text string definitions in its protected list memory area "List 3" and calling them by simple commands.

Indexed character definitions and text string definitions are essentially indexed subroutines, but definable and callable by their own commands, and managed by a dedicated internal RTC6 PCIe Board management table – separately from indexed subroutines.

The individual character and text string definitions must specify the shape and orientation (for example, parallel to the x or y axis) of the characters or text strings. Both relative and absolute vector commands can be used for this. The end position of a character or a text string should be chosen to serve as the start position of a subsequent character. Each character definition or text string definition must be terminated with **list\_return**.

### Defining Indexed Character Sets

A sequence of character-defining list commands can be directly stored in the protected list memory area "List 3" by **load\_char** (the resultant automatically-assigned memory address can be queried by **get\_char\_pointer**). Alternatively, a non-indexed subroutine can be subsequently indexed with **set\_char\_pointer** and then copied by **save\_disk/load\_disk** as an indexed character in the protected list memory area "List 3".

The RTC6 PCIe Board manages up to 4 character sets, each with 256 indexed characters.

Other than that, the same rules as for indexed subroutines are applicable, see [Section "Indexed Subroutines", page 103](#) and [Section "Subsequent Protection and Conversion of Non-Indexed Subroutines", page 106](#).

### Notes

- \0 (NUL) is a markable character, too.  
\0 also serves as a text-output delimiter (for text strings), in which case it is not marked.
- Indexed character set definitions cannot use **mark\_text**, **mark\_time**, **mark\_date** and **mark\_serial**. Otherwise, improper marking might occur during execution of the indexed character.

## Calling Indexed Characters

The marking of an individual character is started by calling **mark\_char** (or the "AbsCall" command **mark\_char\_abs**) along with the index of the corresponding indexed character definition.

To label serial numbers, indexed characters (digits) can also be called up with **mark\_serial**, see [Chapter 7.5 "Marking Dates, Times and Serial Numbers", page 197](#).

The marking of entire text passages can be started by **mark\_text** (or the "AbsCall" command **mark\_text\_abs**). The desired character set can be selected in advance by **select\_char\_set**.

When a **mark\_text** is loaded, the to-be-marked text (if more than 12 characters in length) is split into blocks of 12 characters, with each block receiving its own **mark\_text** in the list memory. Keep this in mind to prevent unintended overflow of the corresponding memory area.

## Defining Indexed Text Strings for Time, Date and Serial Number

For the marking of times, dates and serial numbers, it can be useful to define text strings such as months ("January"..."December", "Jan."..."Dec.", "/01"..."/12/" etc.) and days of the week ("Sunday"..."Saturday" or "Sun."..."Sat." etc.).

Here, you can likewise use previously-defined character sets with the **mark\_char** and **mark\_text**.

With **load\_text\_table**, a sequence of list commands defining a text string can be loaded directly into the protected list memory area "List 3" as an indexed text string (the resultant automatically-assigned memory address can be queried by **get\_text\_table\_pointer**).

Alternatively, a non-indexed subroutine can be subsequently indexed with **set\_text\_table\_pointer** and then copied by **save\_disk/load\_disk** as an indexed text string in the protected list memory area "List 3".

The RTC6 PCIe Board manages up to 42 indexed text strings.

Other than that, the same rules as for indexed subroutines are applicable, see [Section "Indexed Subroutines", page 103](#) and [Section "Subsequent Protection and Conversion of Non-Indexed Subroutines", page 106](#).

## Notes

- **set\_char\_table** is synonymous with **set\_text\_table\_pointer**.



## Calling Indexed Text Strings

Indexed text strings can be called for marking times, dates and serial numbers by the commands **mark\_time**, **mark\_date** and **mark\_serial** (or the "AbsCall" commands **mark\_time\_abs**, **mark\_date\_abs** and **mark\_serial\_abs**), see Chapter 7.5 "Marking Dates, Times and Serial Numbers", page 197.

## Managing Indexed Characters and Text Strings

The index management of indexed characters and indexed text strings occurs separately from the index management of indexed subroutines.

Index management by users (renumbering, duplicating, ...) resembles index management of indexed subroutines, see **Section "Index Management and Defragmentation"**, page 105, using **copy\_dst\_src**, **load\_char**, **load\_text\_table**, **get\_char\_pointer**, **get\_text\_table\_pointer** and **save\_disk/load\_disk**. Defragmentation of the protected list memory area "List 3" also includes indexed characters and text strings.

## 6.5.3 Jumps

**list\_jump\_pos** (synonymous with **set\_list\_jump**) and **list\_jump\_rel** allow the definition of jumps to a specified address which are carried out by the RTC6 PCIe Board at runtime.

With **list\_jump\_pos**, an absolute memory address within the configured list memory area ("List 1" and "List 2") can be specified. Jumps into and out of the protected list memory area "List 3" are not allowed with **list\_jump\_pos**. A **list\_jump\_pos** having such an unallowed jump address is ignored during execution.

With **list\_jump\_rel**, jump distances (that is, relative memory addresses) can be specified. **list\_jump\_rel** can be used in all list memory areas, even the protected list memory area "List 3". Nevertheless, when specifying jump addresses, you should be sure the jump does not exceed the boundary of the corresponding memory area. Otherwise, **list\_jump\_rel** is ignored by the RTC6 PCIe Board during processing.

If **list\_jump\_rel** is used in an indexed subroutine, you must further ensure the jump does not exceed the boundaries of the subroutine. During processing of indexed subroutines, relative jumps that exceed the boundaries of a subroutine are ignored by the RTC6 PCIe Board.

### Notes

- Reconfiguration of the list memory or conversion of a subroutine can result in an originally-valid jump address becoming invalid due to new list boundaries or an altered subroutine position in the memory. In this case, the RTC6 PCIe Board ignores the corresponding jump command – hence, the user program does probably no longer function as intended. Therefore, exercise care when programming jump commands.
- When conditional jump commands are used, execution of a jump is dependent on an external control signal, see **Chapter 9.3.2 "Conditional Command Execution"**, page 281.
- Jump commands initiating a jump to themselves as **list\_jump\_rel( 0 )** are ignored at runtime to prevent an infinite loop that excludes further activities. On the other hand, conditional jump commands as **list\_jump\_rel\_cond( Mask1, Mask0, 0 )** are allowed, for example, to wait for confirmation of a signal.



#### 6.5.4 RTC4-Circular Queue Mode

With the RTC6 PCIe Board, the RTC4 circular queue mode does not exist.

Nevertheless, users can actually replace this operational mode with the RTC6 PCIe Board by using an alternating list change and `load_list`.

`load_list ( 3, 0 )` ensures that new commands are loaded only into an already processed list (that is not **BUSY**), without needing to explicitly specifying the number of the list, see also [Section "Alternating List Changes", page 101](#) and [Section "Loading with Protection", page 96](#).

#### 6.5.5 Loops

Although list jumps, see [Chapter 6.5.3 "Jumps", page 110](#), and conditional jumps, see [Chapter 9.3.2 "Conditional Command Execution", page 281](#), let you repeat any number of list commands endlessly or under external control, precisely specifying the number of executions is not always reliably possible.

But this can be achieved by the command pair `list_repeat` and `list_until`. The command sequence between these two short list commands execute exactly as often as specified with the `list_until` command's parameter, but at least once. Here, nesting up to 8 loops deep is allowed.

`list_repeat` and `list_until` must always be used in pairs. Unpaired or supernumerous commands (`list_until` without an associated `list_repeat`, as well as `list_repeat` commands leading to a nesting depth greater than 8) are ignored. Empty loops (for example, `list_repeat` directly followed by `list_until`) terminate immediately and are not repeated.

The command pairs can be located both within lists and within subroutines.

Within subroutines, `list_until` performs a `list_jump_rel` to the address directly after the associated `list_repeat`. Loops do not function beyond the boundaries of a subroutine, because list jumps into or out of subroutines are not allowed, see [Chapter 6.5.3 "Jumps", page 110](#).

Within lists, however, `list_until` executes a `list_jump_pos` (to the address directly after the associated `list_repeat`). Thus, `list_repeat` and `list_until` can even reside in two different lists, provided that list changing is ensured (by either an explicit list jump or an automatic list change).

If, on the other hand, a list actually has been terminated (as may be the case when using `auto_change_pos`), then the `list_repeat` stack gets automatically deleted and the started loop can no longer be ended because the next `list_until` no longer finds an associated `list_repeat`.



**set\_end\_of\_list** deletes the entire loop management, if no automatic list change is pending, but **list\_return** does not.

Explicit list jumps into or out of the body of a **list\_repeat/list\_until** loop are allowed because they cannot be monitored. Careless use could therefore compromise loop management integrity so severely that started loops do not execute as expected (but subroutine calls from inside a loop are always reliably possible as long as the subroutine itself contains no unpaired **list\_repeat/list\_until** commands).

If a **list\_repeat/list\_until** loop is to be executed with an initially unknown number of repetitions, a high value (for example, greater than the highest expected number) can be specified for the **Number** parameter of **list\_until**. Within the loop, a conditional branch (for example, which is dependent on an external signal) can jump to a position outside the loop and leave the loop this way. At this point there should be a **list\_until( Number = 0 )** to end the just left loop properly.

## 6.6 Using Several RTC6 PCIe Boards in One PC

As many RTC6 PCIe Boards as the PCIe bus allows can be operated simultaneously in one PC. However, the **RTC6 DLL** internal card management can manage a maximum of 255 RTC6 PCIe Boards and RTC6 Ethernet Boards at the same time, see [Chapter 15.5.3 "About the RTC6 Board Management", page 856](#).

All RTC6 PCIe Boards work independently of each other. Command lists for each board can be loaded and executed at any time.

The RTC6 command set allows two methods to write user program when using several RTC6 PCIe Boards:

- Multi-board programming
- Single-board programming

### 6.6.1 Multi-Board Programming

In this programming method, the *multi-board* versions (command names with prefix "**n\_**") of the RTC6 commands are used.

Compared to single-board commands, the card number (32-bit unsigned value) to which the command is to be transmitted must be specified before the parameters. All other parameters are identical.

The installed RTC6 PCIe Boards are numbered in the order found during initialization (starting with 1), see [Chapter 15.5.3 "About the RTC6 Board Management", page 856](#).

The multi-board command **n\_get\_serial\_number** can be used to determine which RTC6 PCIe Boards have been assigned numbers. See also example (3) below.

**rtc6\_count\_cards** returns the number of RTC6 PCIe Boards in the RTC6 board management.

#### Notes

- Multi -board commands are sent to the active board (default board), if the specified number is > 255 or 0 (real boards begin at 1).
- If no real card is entered in the card management under the specified number, the Multi -board command is rejected.
- All multi-board commands are listed in [Chapter 10.1 "Overview", page 288](#). for nearly all single-board commands a corresponding multi-board command is available. Exceptions are explicitly noted in the command descriptions (in [Chapter 10.2 "RTC6 Command Set", page 301](#)).

#### Examples (Pascal)

(1) Write a jump command to the point (500, 500) into the current list of RTC6 PCIe Board #1:

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

(2) Process list with number **list\_no** (1 or 2) on the RTC6 PCIe Board with the number specified by the variable **RTC6\_no**:

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

(3) Return the serial number of RTC6 PCIe Board #1:

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

## 6.6.2 Single-Board Programming

During single board programming, one of the inserted RTC6 PCIe Boards is defined with `select_rtc` as default card. All single board commands following `select_rtc` are sent to the defined board until `select_rtc` is called once again.

Multi-Board commands are not influenced by `select_rtc` (if the card number is valid, see above).

Care must be taken if a process uses multiple boards by multiple threads, because `select_rtc` is not thread-specific but board-specific. It immediately redirects the output of *all* currently running threads of a process to the specified RTC6 PCIe Board.

### Notice!

- `select_rtc` defines the active RTC6 PCIe Board for all threads of one process (user program) that are currently running.  
In multi-threaded user programs, this can result in programming errors.

## 6.6.3 Master/Slave Operation

If several RTC6 PCIe Boards are to be operated clock-synchronized, then they must be connected pairwise with each other by the Master and Slave connectors and installed in preferably (recommended) adjacent PCIe slots. Connect the Master connector of one board to the Slave connector of another board. Suitable connection cables (see [figure 5](#)) are available from SCANLAB.

An RTC6 PCIe Board automatically gets the master board of a master/slave chain, if a further RTC6 PCIe Board is connected to its Master connector but no further RTC6 PCIe Board is connected to its Slave connector. All other RTC6 PCIe Boards are slave boards. See also [Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 55](#).

`get_master_slave` can be used to query separately for each RTC6 PCIe Board the master/slave status, that is, whether it is operated as a master, slave or single board.

For a source code example on how to check which RTC6 PCIe Board is the master and which one is slave, see [Section "Example Code", page 117](#).

After synchronization, the clock phase of each board is delayed by 0...3 1/64  $\mu$ s clock periods (= approx. 0 ns...50 ns) in relation to the clock phase of the preceding (upstream) board.

Without synchronization, delays of up to 10  $\mu$ s can occur. You can use `get_sync_status` to check if a slave board is synchronized to the master board (or to the preceding board in the master/slave chain).

## Initialization with RTC6 Software Package ≥ 1.5.0 (≥ RBF 619)

On all RTC6 PCIe Boards of a master/slave chain must have been executed:

- `load_program_file`

### Clock Phase Synchronization

The synchronization takes place automatically as soon as 2 RTC6 PCIe Boards are connected.

`sync_slaves` does not have to be called anymore.

The synchronization status can be queried at any time with `get_sync_status` even without a prior call of `simulate_ext_start`.

If an RTC6 PCIe Board has been synchronized to external clock cycles in the meantime (see [Chapter 7.4.10 "Synchronization of the RTC6 Clock Cycle and an External Clock Signal", page 196](#)), it is automatically resynchronized with the master board upon leaving this state.

When `load_program_file` is executed, the synchronization of all subsequent slave boards is lost for a short time. However, it is automatically reestablished.

### Notes

- The master board does not pass encoder signals to the slave board(s). They must always be individually supplied to the slave board(s). Here, you need to take into account the 0 ns...50 ns clock phase shift.
- The correction files and lists must be loaded separately onto all RTC6 PCIe Boards.
- By `get_sync_status( Bit #21...Bit #30 )`, the exact propagation time in 1/64 µs clock cycles between two RTC6 boards (outbound and return) can be read out.

## Initialization with RTC6 Software Package < V1.5.0 (≤ RBF 618)

On all RTC6 PCIe Boards of a master/slave chain must have been executed:

- `load_program_file`
- `load_correction_file`

The synchronous timing with stable phase position of a master/slave chain is severed by the first not-initialized board. If an RTC6 PCIe Board is initialized by `load_program_file` but connected as slave to a board which has not been initialized by `load_program_file`, then it is subject to its own clock with a random phase position.

### Clock Phase Synchronization

If the RTC6 PCIe Boards of a master/slave chain are to be synchronously clocked with a defined relative clock phase, then the boards must be correspondingly synchronized by `sync_slaves`.

For this, it is necessary to send `sync_slaves` one-time to the master board only. SCANLAB recommends performing the synchronization immediately after all boards have been initialized (by `load_program_file` and `load_correction_file`). It is sufficient to call `load_correction_file(0,1,2)` or to temporarily detach all scan heads.

### Notes

- The master board does not pass encoder signals to the slave board(s). They must always be individually supplied to the slave board(s). Here, you need to take into account the 0 ns...50 ns clock phase shift.
- If a board in a synchronized master/slave chain is externally clocked separately by a cycle synchronization, then this clocking refers exclusively to this board. All other boards in the master/slave chain continue to be synchronized with the original clock of the master board. If cycle synchronization is then deactivated again, the affected card remains asynchronous. It can only be synchronized again by calling `sync_slaves` once more.
- The correction file and lists must be loaded separately onto all RTC6 PCIe Boards.



## Synchronous Starts and Stops

Within a master/slave chain, external starts (if enabled with `set_control_mode`) and external stops are passed to all boards.

Therefore, a synchronous start of all synchronized boards within a master/slave chain (with presettable track delays) can be triggered by an external start signal, a `simulate_ext_start` or a `simulate_ext_start_ctrl` at any board and correspondingly a synchronous stop by an external stop or a `simulate_ext_stop`.

In contrast, starts by `execute_list` or `execute_at_pointer` as well as stops by `stop_execution` are *not* passed on. They must be separately executed even at master/slave-synchronized boards.

See also [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275](#).

### Notes

- With `master_slave_config` the master/slave interface properties can be configured individually for each RTC6 board.



## Example Code

The following example Delphi source code shows how to check which RTC6 PCIe Board is the master and which one is slave.

The code must be included in a user program, see [Chapter 6.2.5 "Example Code", page 90](#).

```
if init_RTC6_dll() <> 0 then halt;      // Initialize the RTC6 DLL
if rtc6_count_cards() <> 2 then halt;    // Are 2 RTC6 PCIe Boards in the PC?
for CardNo := 1 to 2 do                  // Load firmware and 3D correction files onto both boards
begin
  // Stop RTC6 if a task is currently still running. Optional:
  // n_stop_execution(CardNo);
  if n_load_program_file(CardNo, nil) <> 0 then halt;
  if n_load_correction_file(CardNo, nil, 1, 2) <> 0 then halt;
end;

// Check for Errors (bits #12-#15)
if (n_get_sync_status(1) and $F000 = 0) and (n_get_sync_status(2) and $F000 = 0) then
begin
  // Detect master board
  // Is board 1 the master and board 2 a single slave?
  if (n_get_master_slave(1) = 2) and (n_get_master_slave(2) = 1) then
  begin
    Master := 1;
    Slave := 2;
  end else
  // Is board 2 the master and board 1 a single slave?
  if (n_get_master_slave(1) = 1) and (n_get_master_slave(2) = 2) then
  begin
    Master := 2;
    Slave := 1;
  end
  else
    halt;                                // Something wrong with master-slave configuration
end else
halt;

n_select_cor_table(Master, 1, 0);
n_select_cor_table(Slave, 1, 0);
n_set_control_mode(Master, 1 + 8);        // Master slave, activate Start Stop control
n_set_control_mode(Slave, 1 + 8);
n_sync_slaves(Master);                   // Synchronize master and slave boards
// check synchronization status at any time
// provide an external /START or a simulated external start before checking
n_simulate_ext_start_ctrl(Master); // for ≤ RBF 618
Result = n_get_sync_status(Master) and $3FF; // must be 640
Result = n_get_sync_status(Slave) and $3FF; // must be <4

// MasterCfg, SlaveCfg must be defined elsewhere, see manual
n_master_slave_config(Master, MasterCfg);
n_master_slave_config(Slave, SlaveCfg);
```



## 6.7 Usage of RTC6 PCI Express Boards by Several User Programs

Usage of RTC6 PCIe Boards by several user programs is coordinated by the (RTC6 DLL-internal) RTC6 board management, see also [Chapter 15.5.3 "About the RTC6 Board Management", page 856](#). By `init_rtc6_dll`, it is initialized.

`init_rtc6_dll` automatically grants a user program access rights (by `acquire_rtc`) to the found boards, as long as the access right has not been already assigned to another user program (several RTC6 PCIe Boards or user programs can be used simultaneously, but no board can be simultaneously used by several user program).

Access rights (even if temporary) to boards are granted on an exclusive basis by the [RTC6 DLL](#).

Multiple threads of one user program can use the same board, but can not send commands to it at the same time (the [RTC6 DLL](#) automatically serializes the command calls).

Without access rights, a board can only be accessed by a user program through purely [RTC6 DLL](#)-internal commands that do not require access rights (for example, `get_error`, `get_last_error` and `select_rtc`).

If a user program has granted access rights for a board, then this board can be acquired by another user program only after the original user program explicitly has been released it explicitly by `release_rtc` or `free_rtc6_dll`.

When a board is acquired by `acquire_rtc` (or `init_rtc6_dll` or `select_rtc`), a version check of the [RTC6 DLL](#), [RTC6OUT.out](#), and [RTC6RBF.rbf](#) is performed.

If these files are not loaded yet, then a version check cannot be explicitly performed but the check is still regarded as successful and does thus not hinder the board acquisition. If these files are loaded and the version check detects an error, then access is denied (`get_last_error` return code `RTC6_ACCESS_DENIED` | `RTC6_VERSION_MISMATCH`).

### 6.7.1 Notes on Board Acquisition by a User Program

`init_rtc6_dll`, `acquire_rtc`, `free_rtc6_dll`, `release_rtc` and `select_rtc` affect the access rights of RTC6 PCIe Boards. They do not initialize the RTC6 PCIe Boards.

A user program acquiring a board by `acquire_rtc` (or `init_rtc6_dll` or `select_rtc`) inherits its unadjusted memory contents and operational state. The user program therefore can use the stored data and settings of the board and could intervene in the flow of any list program started (by the previous user program).

If a user program releases a board by `release_rtc` and subsequently reacquires it – without it having been acquired in the meantime by another user program – then the RTC6 PCIe Board can be further used without changes, because in this situation all [RTC6 DLL](#) configuration data remain unaltered. However, the above does not apply, if the acquired board has been released by `free_rtc6_dll` and reacquired after `init_rtc6_dll`.

When an RTC6 PCIe Board is acquired by another user program, some important information managed by the previous user program only in the **RTC6 DLL** is *not* (automatically) taken over. The acquiring user program thereby lacks information related to memory configuration, protected-area management, or the operational status.

If board acquisition is followed by a board initialization by **load\_program\_file** and all settings are newly defined anyway, such missing information would not be relevant.

On the other hand, if the acquiring user program is supposed to further use the inherited state of the RTC6 PCIe Board, then it must explicitly query the missing **RTC6 DLL** information, receive it from the previous user program and explicitly re-establish it so that **RTC6 DLL** and board remain consistent. In this regard, observe the following notes.

#### Notes

- For a correct behavior of the input pointer at the list borders, the memory configuration currently set in the **RTC6 DLL** for the acquiring user program must be consistent to the current memory configuration of the acquired board. **get\_config\_list** obtains the current memory configuration of the board and sets it in the **RTC6 DLL** correspondingly.
- The management tables of protected functions (indexed subroutines, character sets and text strings) are located on the board. All protected functions stored on the board therefore remain callable. On the other hand, information on where the next protected function should be loaded is lost. This information can only be restored by **save\_disk/load\_disk**, see Section "Index Management and Defragmentation", page 105.

An alternative restoration method is not possible. The intermixed loading of protected functions by differing applications should therefore be avoided.

- The input pointer is generally not inherited (the input pointer location currently saved in the **RTC6 DLL** for the acquiring user program is used, maybe corrected after **get\_config\_list**). On the other hand, output pointers of lists can be queried after an acquisition by **get\_status**.
- After acquisition and until the next **load...** command, the list status (in regards to LOAD and READY) might be incorrect. But for further execution this is not important, and the status is newly set after the next **load...** command.
- Other settings such as **start\_loop** or laser settings are not relevant to the **RTC6 DLL**. Though settings used by the previous user program can not generally be queried, new settings can of course be set as desired.
- Error handling is performed separately for each board and each user program. When access rights are exchanged, this data is not included.

## 6.8 Error Handling

In order to catch errors in the program flow, the RTC6 PCIe Board carries out a general error handling. With some commands there is also an individual error handling.

General errors occur, for example, if:

- the user program has no access rights for the board (`RTC6_ACCESS_DENIED`)
- the board fails to respond to a control command (`RTC6_TIMEOUT`)
- PCI communication problems occur during sending (`RTC6_SEND_ERROR`)

Individual errors occur, for example, if:

- calling a command with an unallowed (uncorrectable) parameter (`RTC6_PARAM_ERROR`, see, for example, `get_value` or `write_da_x`)
- rejected sending of a list command (`RTC6_REJECTED`, for example, due to an invalid input pointer)
- sending a control command at an improper time (`RTC6_BUSY`, for example, `goto_xy`, when a list is still being processed)

In such cases, the control commands are not executed and list commands are typically each replaced by `list_nop` (for example, for `RTC6_PARAM_ERROR` or `RTC6_IGNORED`, see `set_end_of_list` as an example).

The bits assigned to these errors are set or accumulated in the **RTC6 DLL** with each command in (card specific) error variables `LastError` and `AccError`:

- Error code `LastError`:
  - is automatically reset at the beginning of every command
  - therefore, is a listing of occurred errors from the most recently executed command
  - `LastError` can be queried by `get_last_error`<sup>(1)</sup>.
- Cumulative error code `AccError`:
  - Is reset when initializing the **RTC6 DLL**
  - Can be reset by the user program itself by `reset_error`
  - Are all accumulated error bits since the last error bit reset by `reset_error`
  - Can be queried by `get_error`<sup>(1)</sup>

Error handling takes place separately:

- for each board
- each user program

A `reset_error` does not delete the error code of another user program with current access rights to the specified board.

If access rights are exchanged, this data is not also exchanged.

Error handling only takes place during processing of commands within the **RTC6 DLL** and when sending to the RTC6 PCIe Board. Error handling takes place during execution of a list program.

An example code of how to incorporate board-specific error variables is provided in the command description of `get_error`.

Some control commands (for example, `init_rtc6_dll`, `load_correction_file` or `load_program_file`) additionally return a special error code as the result value that is not buffered and must therefore be immediately evaluated or discarded by the user program.

(1) The described mechanism only applies for commands that establish communication with the RTC6 PCIe Board. Commands that do not establish communication with the RTC6 PCIe Board (for example, `rtc6_count_cards`, `set_rtc4_mode` or `get_serial_number`) neither generate nor alter `LastError` or `AccError` (see also comments in the corresponding command descriptions).



### 6.8.1 Download Verification

Verification of RTC6 communication is vital particularly in medical applications. For this purpose, you can activate download verification separately for each board by `set_verify`.

However, this automatically results in extended download times.

If download verification is activated and an error is found, then the error code `RTC6_VERIFY_ERROR` is set, which can be queried by `get_last_error` or `get_error`. Certain operations might immediately be aborted and the board would then no longer be functional (for example, if `load_program_file` has been aborted).

With download verification activated, the following checks are performed (also note the comments in the command description of `set_verify`):

#### (1) Loading of list commands

For list-command downloads, each download is read back and compared (for equality) against the sent command. Here, only transfer to the RTC6 PCIe Board itself is checked; automatic parameter adjustments (for example, clipping) are not taken into account.

#### (2) Loading of control commands

For control commands, the corresponding parameters are read back and compared for equality against the sent parameters. Automatic parameter adjustments are not taken into account.

#### (3) `load_program_file`

For sending of `load_program_file`, the following is checked:

- The binary auxiliary file `RTC6DAT.dat` is tested by a checksum for file correctness and PCI-transfer correctness.
- The firmware file `RTC6RBF.rbf` is only checked by a bitwise transfer handshake. No other checking is possible.
- Each loaded section of the program file `RTC6OUT.out` is immediately read back for checking. If an error is detected, then the loading process aborts.

#### (4) Loading of correction files

For loading by `load_correction_file`, the integrity of the to-be-loaded correction file is checked (by the checksum) and the transfer itself checked for correctness by an immediate read back of the correction table. For this function, the correction file must contain a checksum (see command description of `set_verify` and `verify_checksum`).

#### (5) Loading of tables

For loading other tables (for example, by `load_varpolydelay`), the transfer is checked for correctness by an immediate read back of the table. In addition to the `get_last_error` return code `RTC6_VERIFY_ERROR`, the corresponding error return value of the loading command is also get set.

### 6.8.2 Checking for Overruns

By `get_overrun`, it can be checked whether overruns of the 10 µs clock period have occurred. See also Section "Clock Overruns", page 171.



### 6.8.3 Example Code

The following example C source code shows how to catch an error during initialization. It ensures the user program terminates with an error message, if:

- An error occurs during initialization with **init\_RTC6\_dll** (for example, if no RTC6 PCIe Board was detected)
- The desired RTC6 PCIe Board (here: the board with serial number 12345) is not found
- Access is denied to the desired RTC6 PCIe Board
- An error occurs during **load\_program\_file** (for example, a version mismatch, file or system error)

The code must be included in a user program, see [Chapter 6.2.5 "Example Code", page 90](#).

```
UINT ErrorCode;
ErrorCode = init_RTC6_dll();
if ( ErrorCode )
{
    // Reading the number of RTC6 boards detected during initialization with init_RTC6_dll
    const UINT RTC6CountCards = rtc6_count_cards();
    if ( RTC6CountCards )
    {
        // Detailed error analysis for all detected boards
        UINT AccError( 0 );
        for ( UINT i = 1; i <= RTC6CountCards; i++ )
        {
            // Errors which occurred during execution of init_RTC6_dll
            const UINT Error = n_get_last_error( i );
            if ( Error != 0 )
            {
                AccError |= Error;
                const UINT SerialNumber = n_get_serial_number ( i );
                printf( "RTC6 board number %d (serial number %d): Error %d detected\n",
                        i, SerialNumber, Error );
                n_reset_error( i, Error );
            }
        }
        if ( AccError )
        {
            free_RTC6_dll();
            return;
        }
    }
}
```



```
else
{
    printf( "Initializing the DLL: Error %d detected\n", ErrorCode );
    free_rtc6_dll();
    return;
}
else
{
    // Reading the internal board number for the desired RTC6 board
    const UINT SerialNumberOfDesiredBoard ( 12345 );
    const UINT RTC6CountCards = rtc6_count_cards();
    UINT InternalNumberOfDesiredBoard ( 0 );
    for ( UINT i = 1; i <= RTC6CountCards; i++ )
    {
        if ( n_get_serial_number( i ) == SerialNumberOfDesiredBoard )
        {
            InternalNumberOfDesiredBoard = i;
        }
    }
    if ( InternalNumberOfDesiredBoard == 0 )
    {
        printf( "RTC6 board with serial number %d not detected.\n", SerialNumberOfDesiredBoard );
        free_rtc6_dll();
        return;
    }
    // Selecting the desired RTC6 board as the active RTC6 board for this user program
    if ( InternalNumberOfDesiredBoard != select_rtc( InternalNumberOfDesiredBoard ) )
    {
        // Errors which occurred during execution of select_rtc
        ErrorCode = n_get_last_error( InternalNumberOfDesiredBoard );
        if ( ErrorCode & 256 ) // RTC6_VERSION_MISMATCH
        {
            if ( ErrorCode = n_load_program_file( InternalNumberOfDesiredBoard, 0 ) )
            {
                printf( "n_load_program_file returned error code %d\n", ErrorCode );
            }
        }
        else
        {
            printf( "No access to RTC6 board with serial number %d\n", SerialNumberOfDesiredBoard );
            free_rtc6_dll();
            return;
        }
        if ( ErrorCode )
        {
            printf( "No access to RTC6 board with serial number %d\n", SerialNumberOfDesiredBoard );
            free_rtc6_dll();
            return;
        }
        else
        {
            // if n_load_program_file was successful, select the desired board
            (void) select_rtc( InternalNumberOfDesiredBoard );
        }
    }
}
```



## 6.9 Miscellaneous

### 6.9.1 Free Variables

8 so-called "free" variables are available.

Users can freely assign data to them by the control command `set_free_variable` and the short list command `set_free_variable_list`.

These variable values can be:

- outputted at the `McBSP interface`, see also [Chapter 9.1.7 "McBSP Interface", page 273](#)
- read back by `get_free_variable` and `get_value` (see command descriptions)
- recorded by `set_trigger/set_trigger4` (see command descriptions)

The free variables let you, for example, transmit control commands over the `McBSP interface` to user hardware or document the operational states of the board (for example, with branches).

#### Notes

- You can use `set_free_variable` and `set_free_variable_list` to define any 32-bit unsigned values as variable values. However, the `McBSP interface` only outputs 24-bit values by `set_mcbsp_out_ptr` and 16-bit values by `set_mcbsp_out` (but `get_free_variable`, `get_value` and `set_trigger/set_trigger4` return full 32-bit values).



## 7 Basic Functions for Scan Head Control and Laser Control

### 7.1 Marking Points, Lines and Arcs

#### 7.1.1 Marking with Vector and Arc Commands

As explained in [Chapter 6.1 "RTC6 Software Concept Basics", page 83](#), positioning of the scan system's axes (and thus of the laser beam) under RTC6 PCIe Board control is achieved by calling jump, mark, arc and ellipse commands.

Each of these commands describes one vector or arc<sup>(1)</sup>. By using the micro vector commands, arbitrarily shaped trajectories<sup>(2)</sup> can be implemented.

Even numeric and alphabetic characters ultimately consist of the constituent lines, points and arcs that define them, see [Chapter 7.5 "Marking Dates, Times and Serial Numbers", page 197](#).

Vector commands (jump, mark) require as parameters the coordinates of the *end point* of the corresponding vector<sup>(3)</sup>. Each vector starts at the *current output position*, which is the end point of the preceding vector or arc.

Arc and ellipse commands require parameters for the coordinates of the arc center and the arc angle(s). Circular arcs start at the current output position. The elliptical arc start at the position specified by the command parameters. A direct connection to the last output position must be explicitly ensured by the user program itself with suitable parameters.

Otherwise there is a "Hard jump" there.

The initial output position at RTC6 PCIe Board start-up is the center of the image field, that is, the point  $(0|0)$ . Refer to [Chapter 7.3 "Scan Head Control", page 156](#) for a description of the image field coordinate system.

At run-time, each vector or arc to be traced by the scan system gets divided by the RTC6 PCIe Board into microsteps, see [Section "Microstepping", page 129](#)<sup>(4)</sup>.

Jump commands serve to move the scan system axes while the laser is *off* to a new position. In contrast, **Mark commands** perform marking motions while the laser is switched *on* (see also following description).<sup>(5)</sup>

To mark a point, the "laser active" laser control signals must be switched on for the desired time period after a jump command or **[\*]mark[\*]** command, see [Chapter 7.1.3 "Marking Single Points", page 130](#).

For line and arc marking, the RTC6 PCIe Board automatically switches the "laser active" laser control signals on at the beginning of a **Mark command** and later switches it back off (for example, at the beginning of a subsequent jump command).

Here, users can specify delays to optimize the timing of scan head and laser control signals for their particular applications, see [Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control", page 134](#).

Adjustment of laser parameters is described in [Chapter 7.4 "Laser Control", page 173](#).

A thoroughly-commented sample code for a basic marking task is shown in [Chapter 7.1.4 "Example Code", page 131](#).

(1) Here, wider line widths can be specified by **set\_wobbel\_mode**.

(2) See Glossary entry on [page 880](#).

(3) The coordinates must be specified as digital control values (without units). To avoid confusion with coordinates in [mm], SCANLAB uses the expression "coordinate values [in bits]".

(4) Only iDREVE scan systems (see glossary entry on [page 879](#)) which are equipped with an appropriate tuning can execute jumps also in jump mode, see [Chapter 8.1.5 "Jump Mode", page 203](#).

(5) Outside a list, repositioning can be achieved by **goto\_xy** or **goto\_xyz** (even while the laser control signals are on).



## Jump Commands

A jump command (`jump_abs` or `jump_rel`<sup>(1)</sup>) causes the mirrors to move from the start point to the end point of a vector. The “laser active” laser control signals are automatically switched off at the beginning of the vector and remain switched off during the jump, see also [Chapter 7.2.1 “Laser Delays”, page 134](#) and [Chapter 7.2.2 “Scanner Delays”, page 136](#). The jump speed is defined by `set_jump_speed` and `set_jump_speed_ctrl`.

If the laser system does not allow fast switching, the jump speed must be set high enough to prevent a visible marking effect on the workpiece. See also the commands `home_position` and `home_position_xyz`.

## Mark Commands

Upon execution of a `[*]mark[*]` command (`mark_abs` or `mark_rel`<sup>(1)</sup>), the laser focus is moved linearly from the start point to the end point of the vector. The RTC6 PCIe Board automatically turns on the signals for “laser active” operation at the beginning of a `[*]mark[*]` command, see also [Section “Polylines”, page 126](#).

The mark speed is defined by `set_mark_speed` and `set_mark_speed_ctrl`. It can be changed anywhere in a list by `set_mark_speed` or by `set_mark_speed_ctrl`, if no list is currently being processed.

## Arc Commands

The arc commands `arc_abs` and `arc_rel` can be used for marking circular arcs<sup>(2)</sup>. These commands require parameters for the coordinates of the arc center and the arc angle. The circular arc starts at the current output position, with angles counted positively and clockwise (contrary to the mathematical definition).

When an arc command is executed, the laser focus is guided along the specified arc at the specified speed. The “laser active” laser control signals are automatically switched on at the beginning of the execution of an arc command, see also [Section “Polylines”, page 126](#).

## Polylines

If another `[*]mark[*]` command (or arc command) follows immediately afterward (“[Polyline](#)”), the signals for “laser active” operation remain on.

Therefore, a continuous marking is possible by a direct line-up of `[*]mark[*]` commands (and arc commands).

The signals for “laser active” operation are switched off at the beginning of the (normal) command that follows the last `[*]mark[*]` command of a [Polyline](#).

See also [Section “EdgeLevel”, page 141](#).

(1) For using abs and rel commands, see [Section “AbsCalls”](#), page 104. Additionally are available: timed vector commands, see [Chapter 8.9 “Timed Vector Commands and Timed Arc Commands”, page 260](#), para vector commands, see [Section “Vector-Defined Laser Control”, page 195](#) and 3D vector commands, see [Chapter “3D Commands”, page 223](#).

(2) For using abs and rel commands, see [Section “AbsCalls”](#), page 104. Additionally, timed arc commands are available, see [Chapter 8.9 “Timed Vector Commands and Timed Arc Commands”, page 260](#).

## Ellipse Commands

The RTC6 command set also provides commands for marking elliptical arcs.

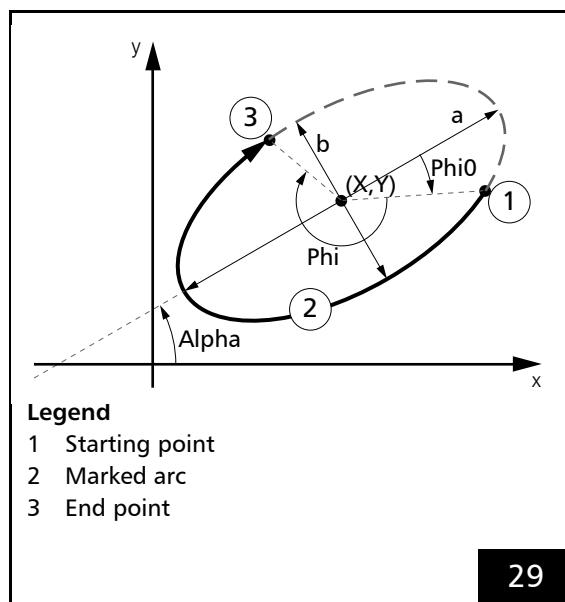
Here (unlike marking of vectors or circular arcs), you generally need to call two commands: **set\_ellipse** and one of the arc commands **mark\_ellipse\_abs** or **mark\_ellipse\_rel**<sup>(1)</sup>.

By **set\_ellipse** the arc shape is specified, see figure 29:

- Lengths  $a$  and  $b$  of the ellipse's half-axes
- The beginning phase angle  $\text{Phi}0$  (and thereby the arc starting point's position relative to the end point of half-axis  $a$ )
- The arc angle  $\text{Phi}$  (and thereby the length of the to-be-marked ellipse section)

The arc commands **mark\_ellipse\_abs** or **mark\_ellipse\_rel** specify the to-be-executed arc's position and orientation in the following manner, see figure 29:

- The coordinates  $(X, Y)$  of the ellipse's midpoint
- The angle  $\text{Alpha}$  between the ellipse's half-axis  $a$  and the  $x$  axis



## Notes

- By  $a$ , you can specify either the short or long half-axis (then use  $b$  for the other axis).  $\text{Phi}0$ ,  $\text{Phi}$  and  $\text{Alpha}$  are always relative to axis  $a$ .
- $\text{Phi}0$  and  $\text{Phi}$  are counted positively clockwise (in contrast to mathematical convention). In contrast,  $\text{Alpha}$  is counterclockwise (in accordance with mathematical convention).
- As with mark and arc commands, during an ellipse command, the laser focus moves with the defined mark speed along the specified arc. The signals for "laser active" operation are automatically turned on at the beginning of an ellipse command, see also [Section "Polylines", page 126](#). **set\_ellipse** is a short list command, see also [Section "Normal, Short, Variable and Multiple List Commands", page 288](#). Therefore, it can be called between a **Mark command** and a ellipse arc command without thereby interrupting the **Polyline** (the laser remains on).
- In contrast to mark and arc commands (which automatically begin marking at the current output position), elliptical arc commands always begin marking at the starting point determined by the above-mentioned parameters. If the arc starting point does not equal the current position, then a **Hard jump** to the starting point is executed at the beginning of marking (and jump delays are ignored).

(1) For using abs and rel commands, see [Section "AbsCalls"](#), page 104.

- Elliptical arcs can also be marked by circular arc commands (for example, `arc_abs`) if an appropriate coordinate transformation (for example, scaling that differs in the x/y directions) was specified with `set_matrix`. Here, though, the effective mark speed varies along the arc, see also the note on [page 212](#).

This contrasts with `mark_ellipse_abs` and `mark_ellipse_rel`, where in 10 µs intervals the step length gets adjusted for the ellipse's shape at the current position such that the arc is marked with a (largely) constant mark speed.

For very large eccentricities and also at high mark speeds, however, such stepwise ellipse approximation by a 10 µs clock can produce numerical inaccuracies in the end point regions of the large half-axis. Consequently, the effective mark speed there might not be precisely constant (for example, an eccentricity of  $a/b = 2$  and 100 microsteps per circumference would produce a speed deviation of approx. 3.7%).

Moreover, as closed equations do not exist for calculating an ellipse arc length, the step length of the finally-marked microstep is generally shorter and the mark speed correspondingly lower than specified. Nevertheless, the end position is always exact.

Likewise, Sky Writing might produce run-in/run-out irregularities at the large half-axis. Users should therefore check if their chosen parameters are compatible with their precision requirements.

#### [\*]Para[\*] Commands

- `para_jump_abs`
- `para_jump_abs_3d`
- `para_jump_rel`
- `para_jump_rel_3d`
- `para_laser_on_pulses_list` (special case)
- `para_mark_abs`
- `para_mark_abs_3d`
- `para_mark_rel`
- `para_mark_rel_3d`
- `timed_para_jump_abs`
- `timed_para_jump_abs_3d`
- `timed_para_jump_rel`
- `timed_para_jump_rel_3d`
- `timed_para_mark_abs`
- `timed_para_mark_abs_3d`
- `timed_para_mark_rel`
- `timed_para_mark_rel_3d`

If the “vector-controlled laser control” is activated, these commands simultaneously vary a signal parameter linearly along the mark or jump vector, see [Section “Vector-Defined Laser Control”, page 195](#).

[\*]`para_mark[*]` commands generally do not take Sky Writing into account.

### 7.1.2 Microstepping

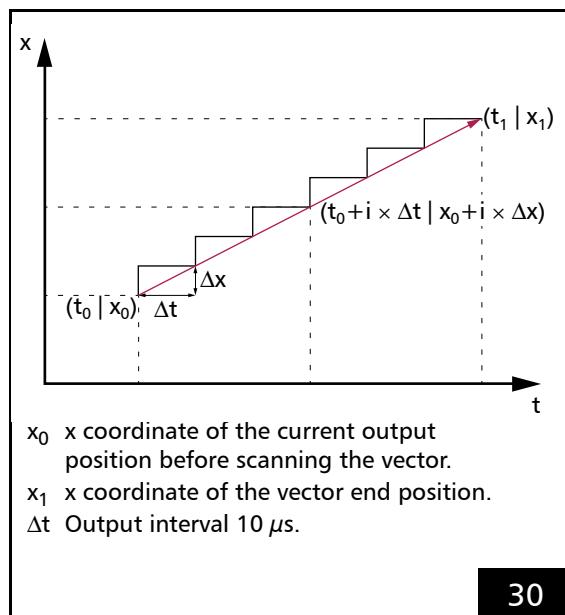
The RTC6 PCIe Board splits up each

- jump command,
- [\*]mark[\*] command and
- arc command

into so-called "microsteps"

(not: microvectors, see [Chapter 8.8 "Microvector Commands", page 259](#)).

The split-up of the x component of a vector into microsteps is shown in [figure 30](#).



The x component of a vector is split-up into microsteps.  
The y component is split-up in the same way.

All microsteps are transferred to the scan head with a constant output interval ( $\Delta t$ ) of 10  $\mu s$  and *cannot* be changed.

The length  $\Delta s$  of each microstep is:

$$\Delta s = v \times \Delta t$$

Whereby  $v$  = current jump speed or mark speed.

### Notes

- Custom curves can be implemented by using microvector commands, see [Chapter 8.8 "Microvector Commands", page 259](#).
- Only iDRIVE scan systems<sup>(1)</sup> which are equipped with an appropriate tuning can execute jumps also in jump mode, see [Chapter 8.1.5 "Jump Mode", page 203](#).

(1) See glossary entry on [page 879](#).



### 7.1.3 Marking Single Points

To mark a single point, the signals for "laser active" operation must be switched on for the desired time period, see [laser\\_on\\_list](#), [laser\\_on\\_pulses\\_list](#), [para\\_laser\\_on\\_pulses\\_list](#) and [Chapter 7.4 "Laser Control", page 173](#).

Alternatively, a single point can also be marked by a timed [Mark command](#) of length zero, see [Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260](#).



#### 7.1.4 Example Code

The following example C source code shows the commands of a simple laser scan application.

A point, a square and a circle are marked in CO<sub>2</sub> Mode. Here it is assumed that the **RTC4 Compatibility Mode** is activated.

The code must be included in a user program, see [Chapter 6.2.5 "Example Code", page 90](#).

```
// Scan system initialization
// Loading and assigning a correction file
ErrorCode = load_correction_file( 0,    // initialize like "D2_1to1.ct5",
                                  1,    // table (#1 is used by default)
                                  2 ); // use 2D only

if ( ErrorCode )
{
    printf( "Correction file loading error: %d\n", ErrorCode );
    free_rtc6_dll();
    return;
}
select_cor_table( 1, 0 ); // assigning correction table #1 to scan head connector #1 (default)

// Laser control initialization, see Chapter 7.4 "Laser Control", page 173.
// Setting the CO2 Mode
set_laser_mode( 0 );

// Setting and enabling the "laser active" laser control signals
set_laser_control( 0x18 ); // All laser signals active low (Bit #3 and #4)
                           // set_laser_control must be called at least once to activate laser signals.
                           // Later on enable_laser/disable_laser would be sufficient.

// Opens List 1
set_start_list( 1 );

// Setting the standby pulses
set_standby_list( 800, 8 );
                  // In RTC4 Compatibility Mode the standby parameters are specified in units of 1/8 µs.
                  // The RTC6 board multiplies the specified values by 8 to convert them to
                  // integer-multiple of 1/64 µs. Half of the standby output period = 100 µs.
                  // Pulse length of the standby pulses = 1 µs.

// Timing, delay and speed preset
// Setting the scanner delays (see Chapter 7.2.2 "Scanner Delays", page 136).
set_scanner_delays( 25, 10, 5 );
                     // Jump delay = 250 µs (specified in [10 µs])
                     // Mark delay = 100 µs (specified in [10 µs])
                     // Polygon delay = 50 µs (specified in [10 µs])
```



```
// Setting the jump and mark speed:  
set_jump_speed( 1000.0 );  
set_mark_speed( 250.0 );  
    // In RTC4 Compatibility Mode the RTC6 board multiplies the speed values by 16.  
    // Jump speed = 1000.0 bits/ms.  
    // Marking speed = 250.0 bits/ms.  
  
// Setting the laser timing (see Chapter 7.4 "Laser Control", page 173).  
set_laser_pulses( 800, 400 );  
    // In RTC4 Compatibility Mode the timing parameters are specified in units of 1/8 µs.  
    // The RTC6 board multiplies the specified values by 8 to convert them to  
    // integer-multiple of 1/64 µs. Laser HalfPeriod = 100 µs.  
    // Laser PulseLength = 50 µs.  
  
// Setting the laser delays (see Chapter 7.2.1 "Laser Delays", page 134).  
set_laser_delays( 100, 100 );  
    // In RTC4 Compatibility Mode the laser delays are specified in units of 1 µs.  
    // The RTC6 board multiplies the specified values by 32 to convert them to  
    // integer-multiple of 1/64 µs. LaserOn delay = 100 µs.  
    // LaserOff delay = 100 µs.  
  
// Defining the end of the list and the end of command transfer to the RTC6 board  
set_end_of_list();  
  
// Execute the list commands for initialization  
execute_list( 1 );  
  
// Marking procedure  
// Waiting for list 1 to be not busy. (load_list( 1, 0 ) returns 1 if successful, otherwise 0);  
// if list 1 is not (no longer) busy:  
// opening the list memory for writing of list commands and setting the input pointer  
// to the start of list 1  
while ( !load_list( 1, 0 ) );  
  
// In the following the list commands for marking point, square and circle are defined and transferred  
// to the RTC6 board.  
  
// Marking the center point of the image field:  
jump_abs( 0, 0 ); // Jump to center point  
    // A jump delay is automatically inserted after the jump.  
// Turning on the laser control signals for 50 µs (+ LaserOff delay - LaserOn delay):  
laser_on_list( 5 );
```



```
// Marking a square around the center point:  
jump_abs( -20000, -20000 ); // Jump to the bottom left corner of the square  
// A jump delay is automatically inserted after the jump.  
mark_abs( -20000, 20000 ); // Marking the left edge of the square  
mark_abs( 20000, 20000 ); // Marking the top edge of the square  
mark_abs( 20000, -20000 ); // Marking the right edge of the square  
mark_abs( -20000, -20000 ); // Marking the bottom edge of the square  
// The laser control signals are automatically switched on  
// with the first [*]mark[*] command after a LaserOn delay and remain on for  
// all four [*]mark[*] commands.  
// A polygon delay is automatically inserted after the first three [*]mark[*] commands, each.  
// Initiated by the following non-marking command (jump command, see below), a mark delay  
// is automatically inserted after the last [*]mark[*] command and the laser control signals  
// are automatically switched off after a LaserOff delay, because a jump command follows.  
  
// Marking a circle around the center point:  
jump_abs( 0, -10000 ); // Jump to the bottom edge of the circle  
// A jump delay is automatically inserted after the jump.  
arc_abs( 0, 0, 360.0 ); // Marking the circle  
// The laser control signals are automatically switched on with the arc command  
// after a LaserOn delay.  
// Initiated by the following non-marking command (set_end_of_list, see below),  
// a mark delay is automatically inserted after the arc command and the laser control signals  
// are automatically switched off after a LaserOff delay, because a set_end_of_list follows.  
  
// Defining the end of the list and the end of command transfer to the RTC6 board  
set_end_of_list();  
  
// Starting the transferred list  
execute_list( 1 );
```

## 7.2 Delay Settings – Coordinating Scan Head Control and Laser Control

Scan head control and laser control should suit the dynamic behavior of the system components, that is, the

- response behavior of the laser
- response behavior of the galvanometer scanners ([Tracking Error](#))
- the type of interaction between laser radiation and material

The following delays are available for this purpose:

- Laser delays
  - LaserOn delay
  - LaserOff delay
- Scanner delays
  - Jump delay (optionally: variable)
  - Mark delay
  - Polygon delay (optionally: variable)

### 7.2.1 Laser Delays

There are two different laser delays:

- LaserOn delay
- LaserOff delay

The laser delays determine when the signals for “laser active” operation are switched on and off.

As a rule, laser delay have *no* influence on the total marking time. Exceptions:

- A negative [LaserOnDelay](#) value, see [Section “LaserOn Delay”, page 134](#)
- Artificially inserted delays, see [Section “Automatic Delay Adjustments”, page 144](#)

The LaserOn delay and the LaserOff delay are set by the undelayed short list command [set\\_laser\\_delays](#). Their unit is  $1/64 \mu\text{s}$  each.

In order to avoid burn-in effects at start and end points of a marking, the laser focus should be moved at a speed which is as constant as possible. Therefore, the laser delay durations must be adjusted to the tracking delay of the scan head and the set mark speed<sup>(1)</sup>, see also [Chapter 7.2.3 “Notes on Optimizing the Delays”, page 144](#).

#### LaserOn Delay

The LaserOn delay is automatically inserted at the start of a single [Mark command](#) and at the start of a series of [Mark command](#) (“Polyline”) and delays the switching on of the laser.

It can be used for several purposes:

- At the beginning of a marking, the mirrors must be accelerated to the specified mark speed (if necessary, from a halt), see [figure 33](#). This phase can be suppressed with a sufficiently high positive LaserOn delay value until the mirrors have already reached a certain angular speed before the laser is switched on. On the other hand, the LaserOn delay must not be too long, otherwise the first part of the marking is cut off.
- Some materials/applications (for example, welding) take some time until they react as desired to the exposure to laser radiation. In this case, it can be useful to “preheat” the starting point of the marking. This can be achieved by setting a *negative* LaserOn delay value. However, this extends the total marking time because the LaserOn delay is inserted prior to the current [Mark command](#) as scanner delay.<sup>(2)</sup>

(1) In addition, automatic readjustment of the laser power during marking can be applied for optimization, see [Chapter 7.4.9 ““Automatic Laser Control””, page 186](#).

(2) This scanner delay is automatically extended, if a preceding LaserOff delay has not yet been expired, see [Section “Automatic Delay Adjustments”, page 144](#).



## LaserOff Delay

The end of a marking is not defined by the marking itself, but by the first “normal” list command that is not a **Mark command**, for example, a jump command.

The acceleration phase at the beginning of a movement leads to a time difference between the respective set position and the real position of the mirrors, see [figure 33](#).

The laser is to be switched off until the *actual value* of the end position is reached (but not already at the *set position*). Therefore, a LaserOff delay is inserted automatically after every **[\*]mark[\*]** command and arc command<sup>(1)</sup> see also [Section “Notes”, page 138](#). This can be used to compensate for the [Tracking Error](#) of the scan head.

(1) In **DSP mode < 3** (see [set\\_dsp\\_mode](#)), the following applies for short marking vectors: if a preceding **LaserOn** delay has not yet been expired, the **LaserOff** delay is temporarily automatically extended accordingly, see [Section “Automatic Delay Adjustments”, page 144](#).

### 7.2.2 Scanner Delays

There are three different types of scanner delays:

- jump delay (optionally: variable)
- mark delay
- polygon delay (optionally: variable)

After each jump command and **Mark command**, the RTC6 PCIe Board inserts one of these scanner delays before the next command is executed (unless otherwise specified).

These scanner delays are defined by **set\_scanner\_delays**. Their unit is  $10 \mu\text{s}$  each.

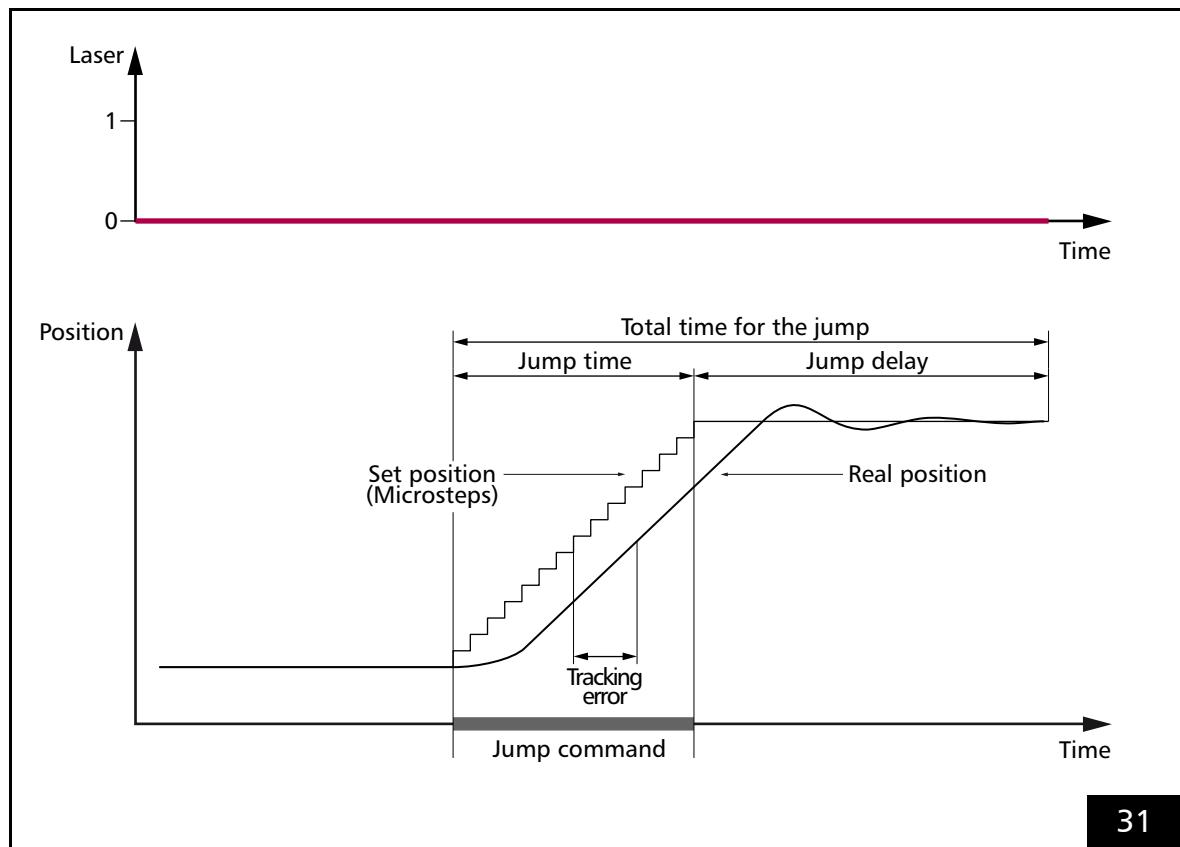
#### Jump Delay

The jump delay is specified by **set\_scanner\_delays** with the **Jump** parameter.

A typical course for a single jump command and the belonging jump delay is shown in **figure 31**.

The total time for a jump is made up of the jump time and the duration of the jump delay.

The duration of the jump delay should suit the dynamic properties of the scan head (**Tracking Error**, tuning) and the jump speed set.



Scan head control during a jump command with a constant jump delay.  
The laser remains off.

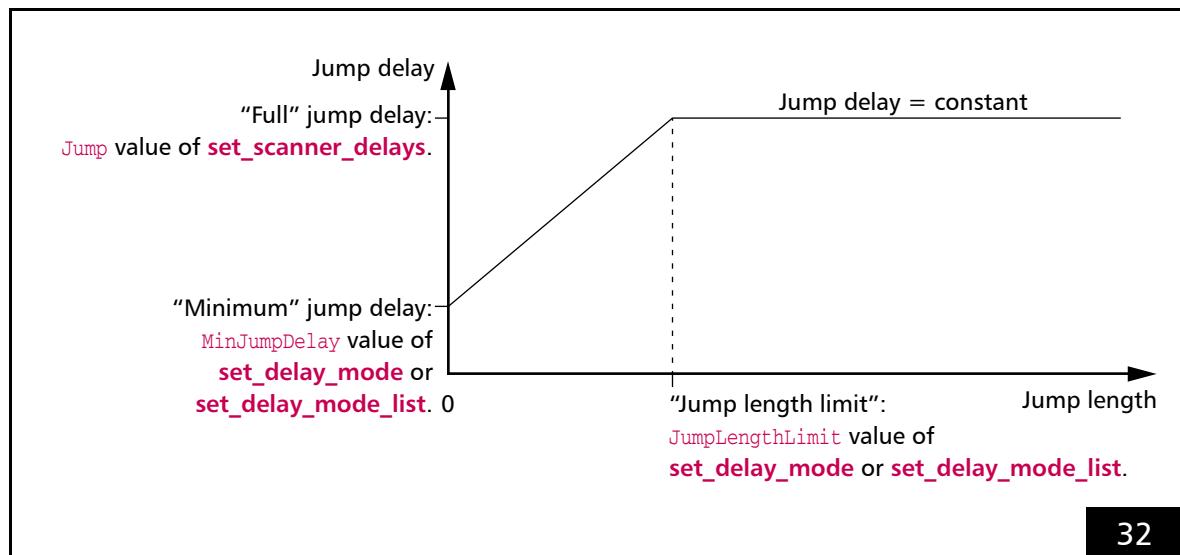
### Variable Jump Delay

With short jumps, the scan head often does not reach the full jump speed. Then *shorter* jump delays (= "Variable Jump-Delays") are sufficient for settling of the mirrors.

For this, there is the "Variable jump delay mode". It is switched on by setting the `JumpLengthLimit` value > 0 (at `set_delay_mode` or `set_delay_mode_list`). Then the RTC6 PCIe Board inserts a linearly interpolated jump delay between `MinJumpDelay` (from `set_delay_mode` or `set_delay_mode_list`) and `Jump` (jump delay from `set_scanner_delays`) for jump lengths between 0 and `JumpLengthLimit` ("Jump Length Limit"), see [figure 32](#). With jump lengths larger than `JumpLengthLimit` a "full" jump delay is always inserted.

### Notes

- With the "Variable jump delay mode", total marking time is reduced, especially when there are many short jumps.
- The "Variable jump delay mode" is switched off by `JumpLengthLimit` = 0 (at `set_delay_mode` and `set_delay_mode_list`).
- After jump vectors of length 0, the variable jump delay duration is 0.
- The "minimum" jump delay should not be larger than the "normal" jump delay. Otherwise, the variable jump delay can become very large.



"Variable jump delay mode": jump delay value depending on the jump length.

32

## Mark Delay

The mark delay is specified by `set_scanner_delays` with the `Mark` parameter.

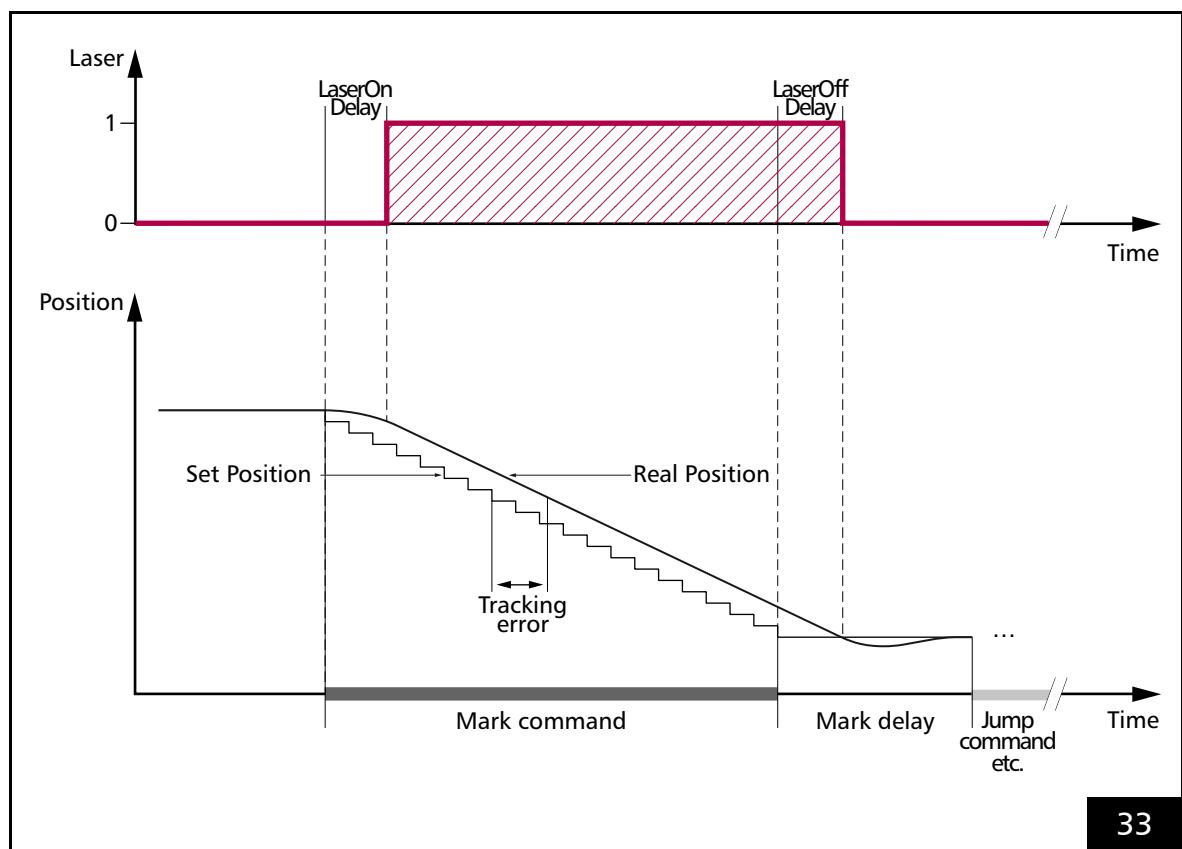
A typical course for a single `Mark command` with the and the belonging laser delays (see [Chapter 7.2.1 "Laser Delays", page 134](#)) is shown in [figure 33](#).

The duration of the mark delay should suit the dynamic properties of the scan head ([Tracking Error](#), tuning) and the mark speed set.<sup>(1)</sup>

## Notes

- If no further `Mark command` follows a `Mark command`, a mark delay is inserted automatically and the laser is switched off after a `LaserOff` delay.
- If a further `Mark command` follows a `Mark command` (= "Polyline"), a (variable) polygon delay is inserted and the laser remains switched on, see [Section "Variable Polygon Delay", page 140](#).
- Short list commands do not lead to insertion of a mark delay or polygon delay and not to a laser switch off. Note that `Mark commands` of length 0 are short list commands, if they are not timed., see [Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260](#).

- (1) About the interaction of laser delays and scanner delays in DSP mode < 3 (see `set_dsp_mode`), see [Section "Automatic Delay Adjustments", page 144](#).



Scan head control and laser control timing during a `Mark command`s or arc command with a mark delay. The laser is on in the hatched period.

## Polygon Delay

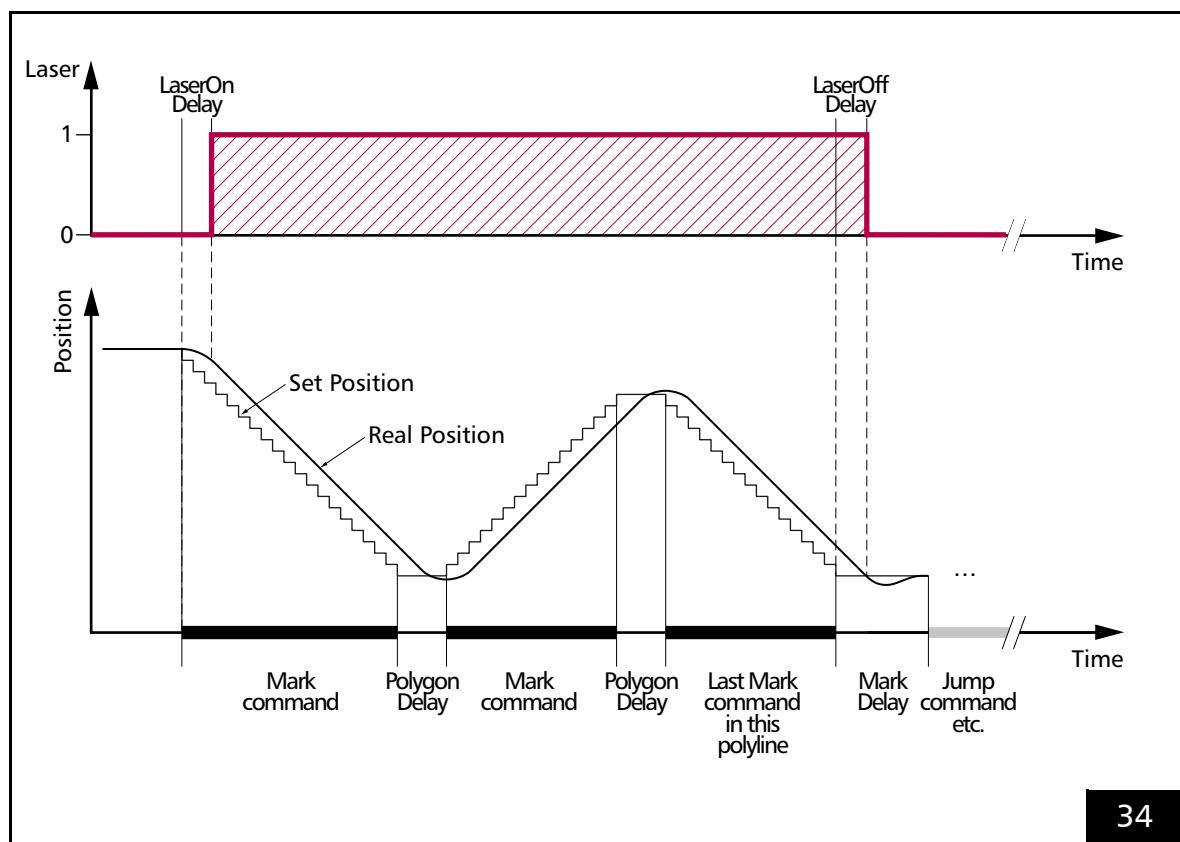
The polygon delay is specified by `set_scanner_delays` with the `Polygon` parameter.

A typical course for a series of **Mark commands** ("Polyline") where polygon delays (instead of mark delays) are inserted between the individual **Mark command** is shown in figure 34.

Short list commands do not interrupt a **Polyline**.

If the (usually smaller) angles between the markings vary only slightly usually a polygon delay value can be specified that is smaller than the mark delay value.

If, on the other hand, the angles vary significantly, then a variable polygon delay is recommended, see Section "Variable Polygon Delay", page 140.



Scan head control and laser control timing during a **Polyline** with a constant polygon delay.

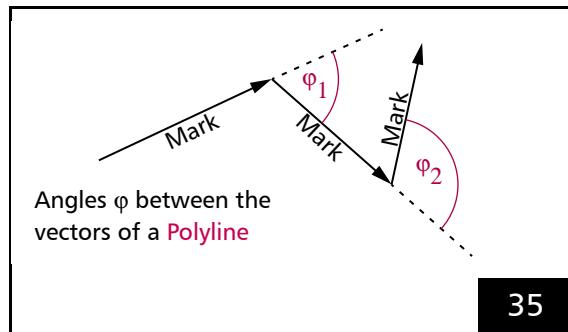
### Variable Polygon Delay

The “Variable polygon delay mode” switched on by setting the `VarPoly` value > 0 (at `set_delay_mode` or `set_delay_mode_list`).

Then, the RTC6 PCIe Board calculates the “Variable polygon delay”  $v_{\text{delay}}(\varphi)$  for every `Polyline` corner according to:

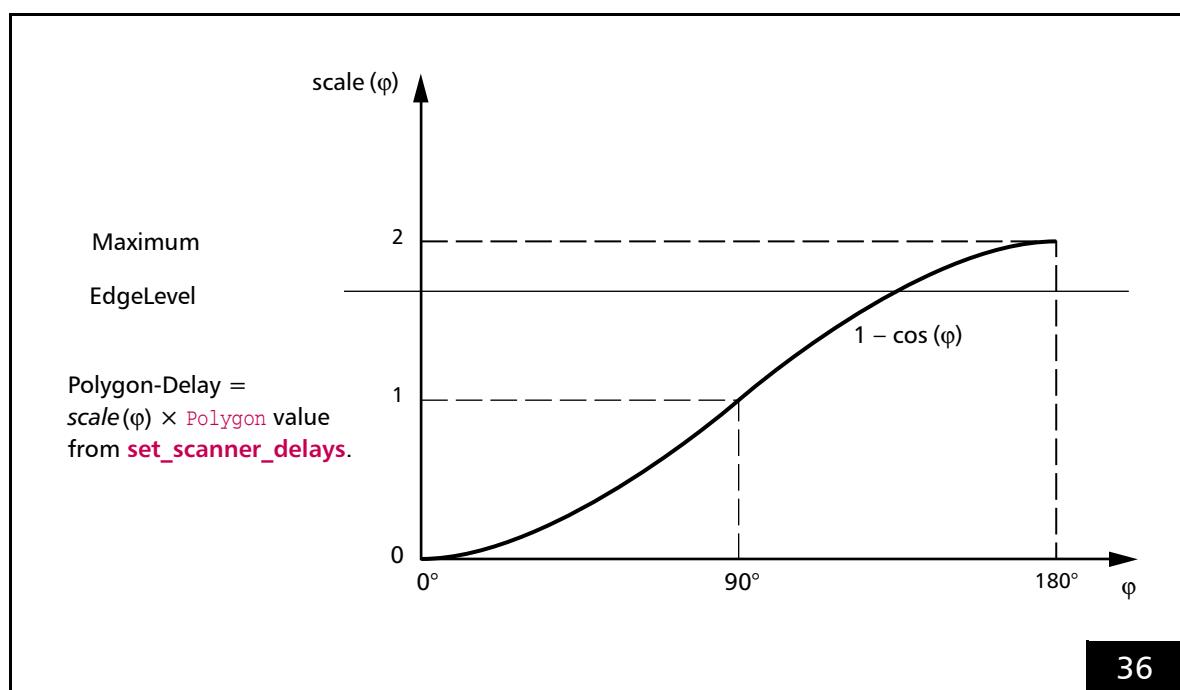
$$v_{\text{delay}}(\varphi) = \text{scale}(\varphi) \times \text{polygon\_delay}$$

For the definition of the angle  $\varphi$  using the example of mark vectors, see [figure 35](#).



Variable Polygon Delay. Definition of the angle  $\varphi$ .

For circular arcs and ellipses, analogously the tangents at the connection point are relevant. `scale( $\varphi$ )` is a scaling function for the `Polygon` value from `set_scanner_delays` (constraint  $0 \leq \text{scale}(\varphi) \leq 2$ ), see [figure 36](#). The default function after `load_program_file` is  $\text{scale}(\varphi) = 1 - \cos(\varphi)$ . It can be replaced by a user-defined function, see [Section “Loading User-defined Variable Polygon Delays”, page 142](#).



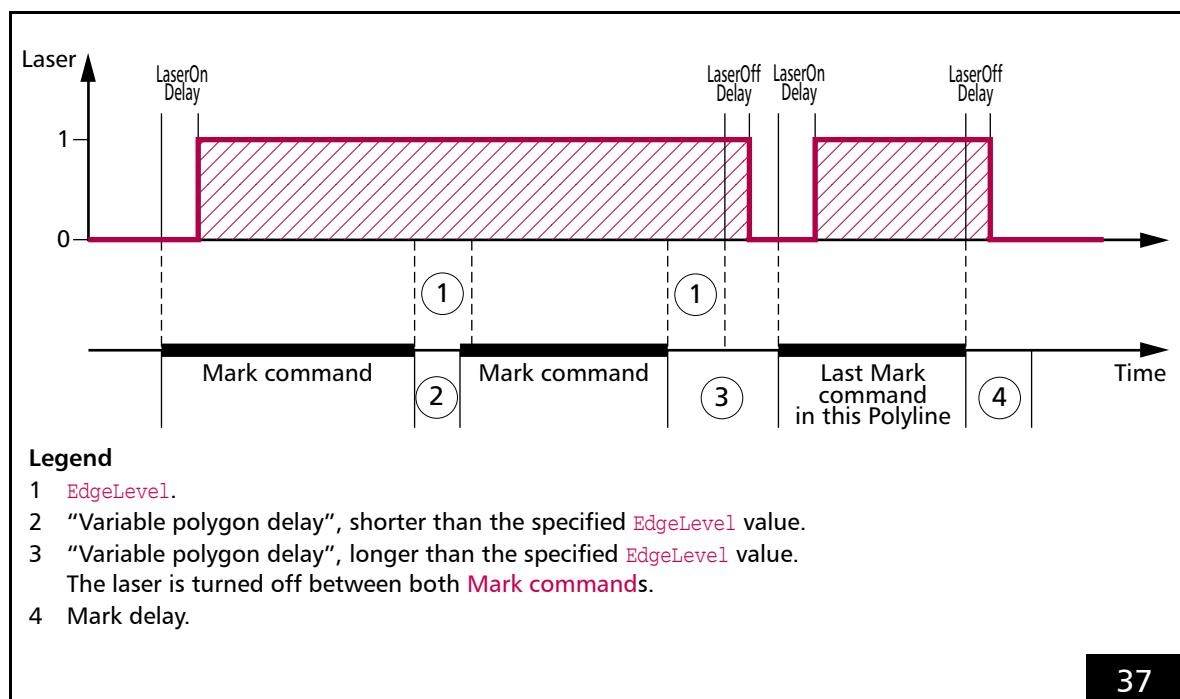
Variable Polygon Delay. Variation of the polygon delay (default function after `load_program_file`).

### EdgeLevel

The “Variable polygon delay” becomes quite long, if the angle  $\varphi$  is close to  $180^\circ$ , see [figure 36](#). This might lead to burn-in effects in sharp corners of a [Polyline](#).

If an [EdgeLevel](#) value  $> 0$  is specified (at [set\\_delay\\_mode](#)), the RTC6 PCIe Board switches off the laser with a LaserOff delay after [EdgeLevel](#) delay clock cycles at the latest and thus terminates the current [Polyline](#), see [figure 37](#).

The next [Mark command](#) starts as usual with the current “Variable Polygon Delay”.



37

Laser control timing during a [Polyline](#) with “Variable polygon delay”. An [EdgeLevel](#) value has been defined with [set\\_delay\\_mode](#).

## Loading User-defined Variable Polygon Delays

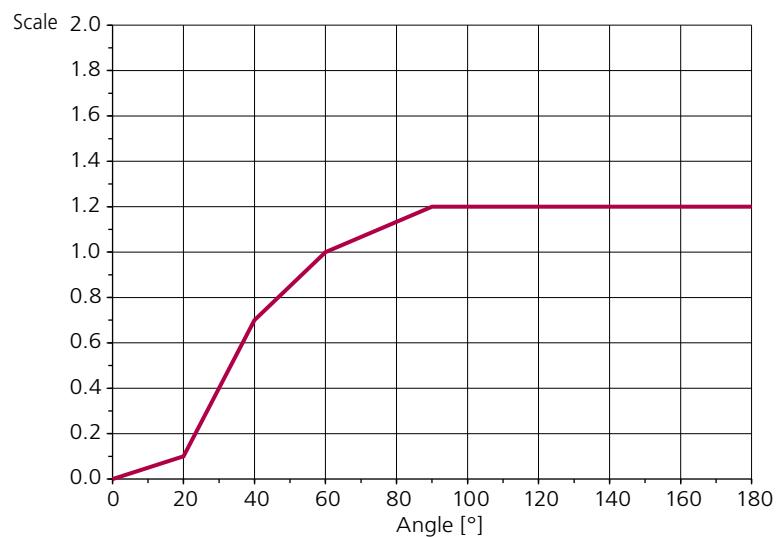
- For the `scale( $\phi$ )` scaling function, `load_varpolydelay` loads a table from an ASCII text file. See also Section "Variable Polygon Delay", page 140.
- The ASCII text file can contain one or more tables<sup>(1)</sup>.
- Each table can contain up to 50 data points ( $\phi \mid scale(\phi)$ ).
- The `scale( $\phi$ )` function is linearly interpolated from the data points.
- As an example, figure 38 shows a table with four data points and the corresponding scaling function.

(1) Even of another type, see table 1, page 142.

Table 1: Possible table types in the ASCII text file. Each type can occur more than once.

[VarPolyTable<No>]	User-defined variable polygon delays, see <a href="#">page 142</a>
[PositionCtrlTable<No>]	Scaling function , see <a href="#">page 188</a>
[AutoLaserCtrlTable<No>]	Nonlinearity curve, see <a href="#">page 193</a>
[JumpTable<No>]	Jump delay values, see <a href="#">page 205</a>
[StretchTable<No>]	2D stretch correction table, see <a href="#">page 226</a>
[Fly2DTable<No>]	2D compensation table, see <a href="#">page 235</a>

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



38

Example: table for user-defined variable polygon delays with 4 data points (left) and corresponding scaling function `scale( $\phi$ )` (right).



For the tables, the following rules apply:

- Each table must begin with the line:  
[VarPolyTable<No>]  
<No> represents the table number.
- If the table contains multiple [VarPolyTable<No>] entries with the same <No>, then only the lines after the first entry are used. Only lines up to the next '[' character (that is not preceded by a semicolon) are used.
- Each data point ( $\varphi$  | scale( $\varphi$ )) is defined as follows:  
Angle<n> = <Value>  
Scale<n> = <Value>  
where <n> must be replaced by a number ( $1 \leq <n> \leq 50$ ) which denotes the number of the data point. The values <Value> for the angle  $\varphi$  (in degrees) and the scaling factor can be specified as (unsigned) floating point numbers. Decimal separator: period (.).
- If the table contains multiple data points with the same Index <n>, then the most recently read one is used and the previous ones are ignored.
- If the table contains multiple data points with the same angle  $\varphi$ , then the data point with the largest Index <n> is used and the others ignored. Equality is checked to within  $\pm 0.01^\circ$ .
- For <Value>, the following ranges apply:  
 $0.0^\circ \leq \varphi \leq 180.0^\circ$  and  $0.0 \leq \text{scale}(\varphi) \leq 2.0$ .
- Each instruction must be in a separate line.
- Spaces and tabs in a line (for example, between '=' and <Value>) are ignored.
- Empty lines are ignored.
- Data points with invalid values are ignored.

- The data point of a particular index <n> is ignored if the corresponding Angle<n> and/or Scale<n> definition is missing.
- The semicolon ';' can be used for comments. All characters in a line following a semicolon are ignored.
- The instructions for data points in the table can be ordered as desired.
- Indices for data point pairs in the table can be selected as desired within the range [1...50] (the table is then automatically sorted by ascending angles).
- If the table contains no valid data point, then **load\_varpolydelay** has no effect (return value 1 or 13).
- The angle  $\varphi = 0^\circ$  means that two successive vectors are parallel and are marked in the same direction.  
If the table contains no explicit data for  $\varphi = 0^\circ$  (equality is checked to within  $\pm 0.01^\circ$ ), then a data point for  $\varphi = 0^\circ$  with the scaling factor  $\text{scale}(0^\circ) = 0$  is added.
- The angle  $\varphi = 180^\circ$  means that two successive vectors are marked in opposite directions.  
If the table contains no explicit data for  $\varphi = 180^\circ$  (equality is checked to within  $\pm 0.01^\circ$ ), then a data point for  $\varphi = 180^\circ$  with the largest scaling factor found in the table for  $\text{scale}(180^\circ)$  is added.

After initialization by **load\_program\_file**, the RTC6 PCIe Board uses the internal (default) table for the variable polygon delay ( $1 - \cos(\varphi)$ , see [figure 36](#)). Alternatively, this can also be achieved with Name = **NULL** in **load\_varpolydelay**.

The table can be saved by **create\_dat\_file**.

### 7.2.3 Notes on Optimizing the Delays

The delays are set by `set_scanner_delays` and `set_laser_delays`.

They should be appropriate for:

- the set jump speed
- the set mark speed
- the tracking error of the used scan head

If the delays are not optimized, the quality of the scanning results are under some circumstances reduced and scanning time is extended. The figures in [Section "Potential Errors when Optimizing the Delays", page 146](#) show the various effects of non-optimized delays on the lettering "RTC".

#### Notes

- The laser delays are specified in units of  $1/64 \mu\text{s}$ .
- In contrast, the scanner delays (jump delay, mark delay and polygon delay) are specified in units of  $10 \mu\text{s}$ .

#### Recommended Sequence

The laser delays should be optimized first. It is recommended to set jump delays and mark delays to a high value. The laser delay lengths have no influence on the total scan time as long as positive values are specified (see also following section).

After the laser delays have been set, the scanner delays can be optimized.

#### Automatic Delay Adjustments

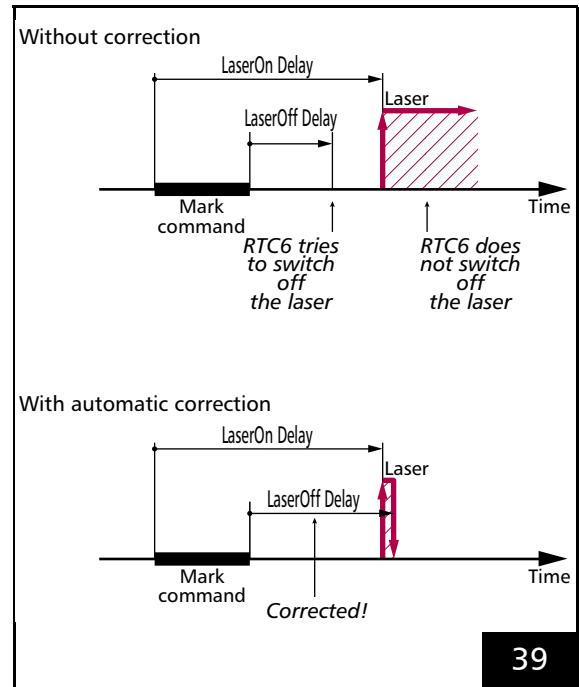
In principle, all delays can be set for any value within the corresponding allowed range. In some situations the laser control could be disturbed, for example, when Laser-On and Laser-Off overlap. Therefore, for each command that switches the laser, the RTC6 PCIe Board checks the set laser delay, corrects them and, if necessary, inserts a scanner delay "artificially". These situations, which are listed below, should actually already be avoided in the user program, because the "corrected" marking does not necessarily meet the requirements.

#### Laser-On and Laser-Off Overlap

The RTC6 PCIe Board corrects the faulty laser delay so that:

- the Laser-On occurs  $1/64 \mu\text{s}$  after the currently effective LaserOff delay
- or the Laser-Off occurs  $1/64 \mu\text{s}$  after the still running LaserOn delay

In both cases, the chronological next mark is cut off somewhat, see also [figure 39](#).



Automatic adjustment of LaserOff delay.

39

#### Notes

- For further information on the general avoidance of such automatic adjustments in [RTC5 Compatibility Mode](#), see [RTC5 Manual, Chapter 7.2.3 "Notes on Optimizing the Delays"](#).



### Laser-On and Laser-On Overlap

If the LaserOn delay is switched from a long delay to a short delay between two markings, the short LaserOn delay could switch on the laser before the long LaserOn delay of the previous marking has expired, if the laser has been switched off between the markers (otherwise the laser would stay on and the new LaserOn delay would not be effective at this point).

To avoid this overlap, the RTC6 PCIe Board in this case (like the RTC5) inserts a scanner delay "artificially". This increases the processing time and changes the dynamics of the galvanometer scanner movement.

### Laser-Off and Laser-Off Overlap

If the LaserOff delay is switched from a long delay to a short delay, the short LaserOff delay could switch off the laser before the long LaserOff delay of the previous marking has expired, if a marking had switched on the laser in the meantime (otherwise the laser would stay off and the new LaserOff delay would not be effective at this point)

To avoid this overlap, the RTC6 PCIe Board in this case (like the RTC5) extends the short LaserOff delay so long that it only expires after the previous one (and of course only after the intermediate LaserOn delay).

### Several Simultaneously Expiring Laser Delays

This can happen if a laser delay is only effective beyond the length of the next command. The RTC6 PCIe Board can process up to 256 LaserOn delays and LaserOff delays simultaneously in a pipeline, as long as Laser-On and Laser-Off alternate and Laser-On—Laser-On and Laser-Off—Laser-Off run one after the other (see automatic adjustments above). This allows markings (interrupted lines) to be "pre-programmed" up to at least 2.56 ms.

### Notes

- In contrast to the RTC6 PCIe Board, the RTC5 can only process one delay of the same type at a time. The delays are adjusted automatically. If necessary, scanner delay are inserted "artificially". In **DSP mode < 3** (see **set\_dsp\_mode**), the RTC6 PCIe Board simulates the behavior of the RTC5 in order to display the same time sequences. For further details, see RTC5 Manual, Chapter 7.2.3.

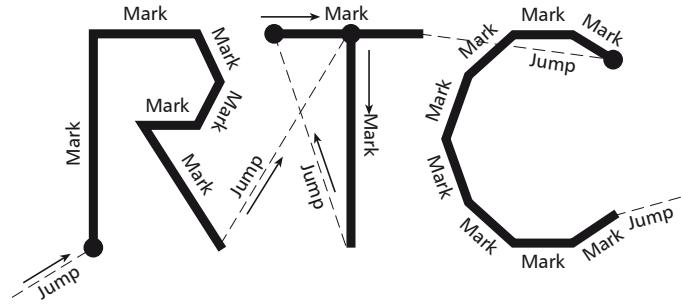
## Potential Errors when Optimizing the Delays

The following figures show the various effects of unsuitable delays on the marking result, in this example on the lettering "RTC".

### LaserOn delay too short

At the beginning of a mark vector the laser is switched on, even though the mirrors have not yet reached the necessary angular velocity.

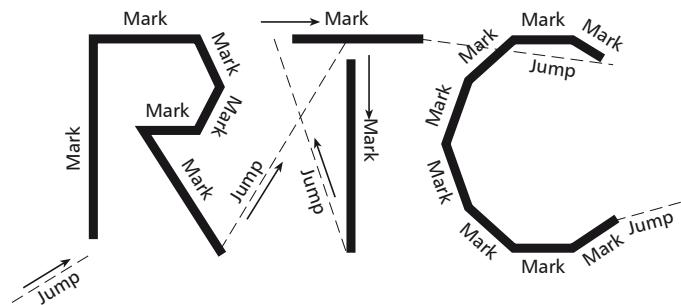
Burn-in effects at the start points of the respective vectors result.



### LaserOn delay too long

The laser is turned on too late at the beginning of a mark vector.

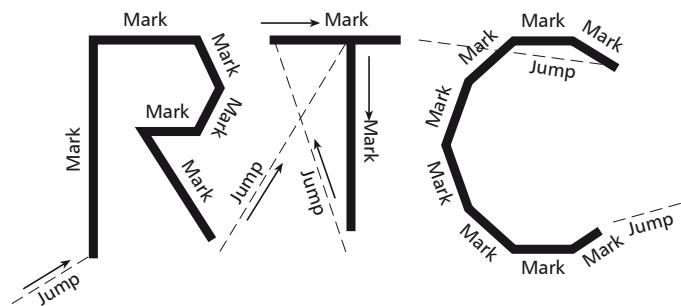
The first part of the vector is not marked.



### LaserOff delay too short

The laser is turned off after the last mark command of a line or **Polyline**, although the mirrors have not yet reached the end position of the vectors.

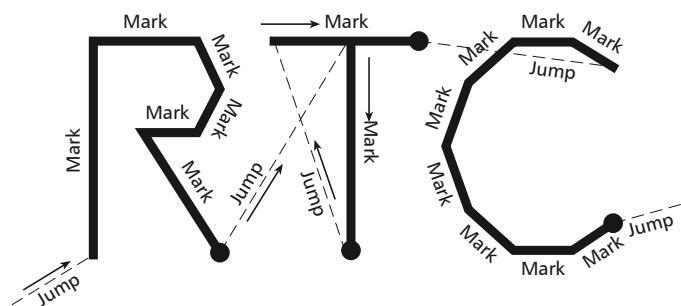
The respective vectors are not marked completely.



### LaserOff delay too long

The laser is turned off too late after the last mark command of a line or **Polyline**. The laser is still on, even though the mirrors have already stopped or move only very slowly.

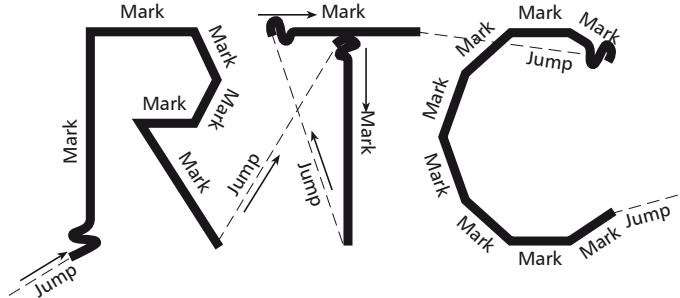
The results are burn-in effects at the end points of the respective vectors.



### Jump delay too short

After a jump, the first mark vector has already started although the scanners have not yet settled.

A running-in oscillation (overshoot) is visible.



### Jump delay too long

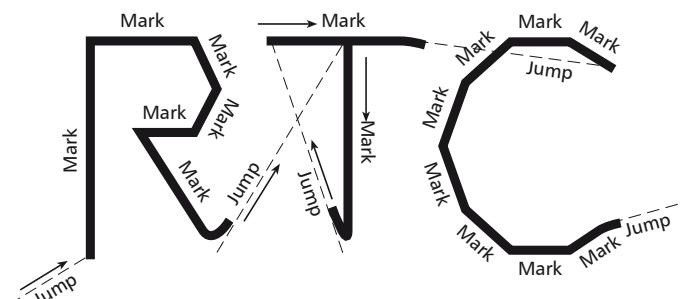
There are no visible effects if the jump delay is too long. However, the scanning time is extended.



### Mark delay too short

Though the mirrors have not yet reached the end position of the last vector of a line or **Polyline**, the command for the succeeding jump vector is already executing.

The end of the mark vector is turned towards the direction of the jump vector.



### Mark delay too long

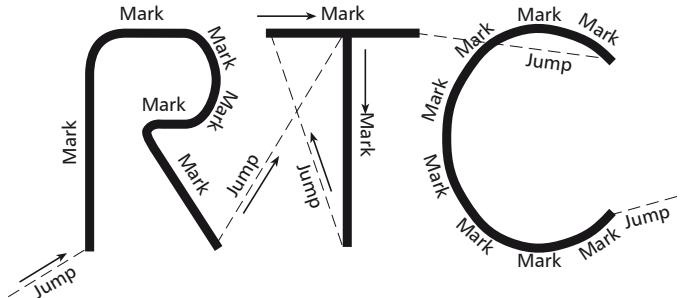
There are no visible effects if the mark delay is too long, but the scanning time is increased.



### Polygon delay too short

The subsequent mark command in a **Polyline** is already executing, although the mirrors have not yet reached the end position of the preceding mark vector.

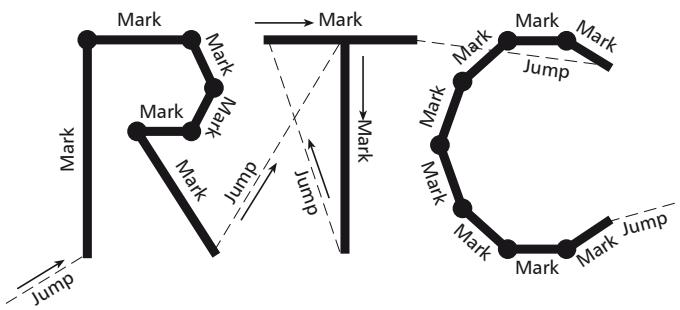
The corners of the **Polyline** are rounded off.



### Polygon delay too long

If the polygon delay is too long, the mirrors are moving too slowly or are even stopping between subsequent mark commands.

Since the laser is not turned off between these vectors, burn-in effects occur.



In the "Variable polygon delay mode", a maximum length ("EdgeLevel") can be defined, see **Section "EdgeLevel"**, page 141 for details.

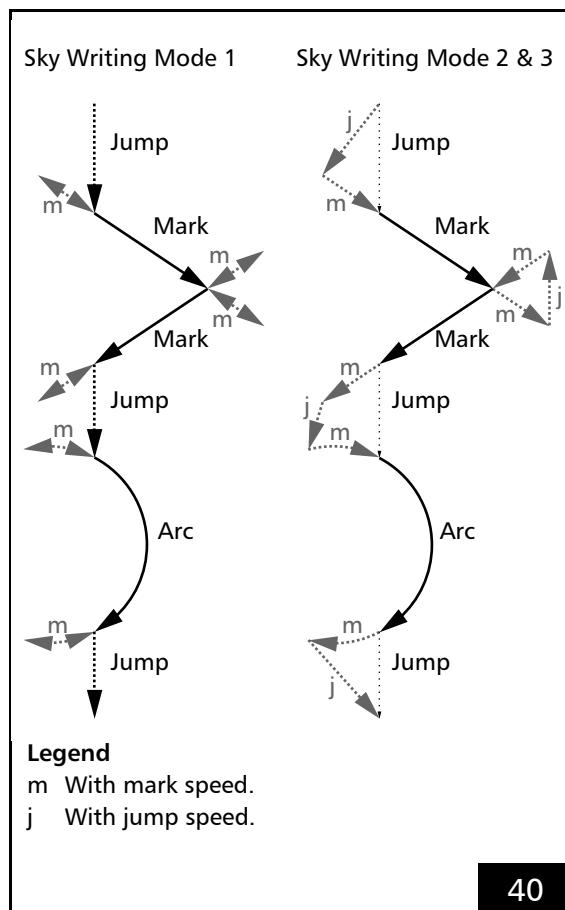
### 7.2.4 Sky Writing

For applications with elevated accuracy requirements the so-called Sky Writing can be enabled. Then, every mark vector is precisely executed at a constant mark speed over the entire vector length.

For **Mark commands** with Sky Writing enabled, the RTC6 PCIe Board automatically performs non-marking Sky Writing scan motions before and after the to-be-executed vectors and arcs, see [figure 40](#).

The following are always executed without Sky Writing:

- Microvector commands
- **[\*]para[\*]** commands



Sky Writing motions (grey). For Sky Writing mode 3 see also [figure 41](#).

By the Sky Writing command parameters you can specify the durations of these run-in and run-out motions as well as synchronization between scan motions and laser control.

Sky Writing can be used in mode 1, mode 2 or mode 3.

#### Sky Writing Mode 1

In Sky Writing mode 1, the RTC6 PCIe Board performs the following Sky Writing motions for each **Mark command** – regardless of previous or subsequent commands:

- In the run-in phase, the vector/arc is preceded by a “forerun” movement performed by the galvanometer scanners at mark speed, see [figure 40](#): the scanners are driven a short distance parallel to the vector (or along the arc extension), initially from the startpoint in the opposing direction, then back to the startpoint.
- After the vector/arc has been processed at mark speed, it is gets a short deceleration and retrace movement of the scanners (at mark speed) appended in the run-out phase.

Sky Writing mode 1 can be switched on/off by **set\_sky\_writing\_para**, **set\_sky\_writing** as well as with the corresponding list commands.

## Sky Writing Mode 2

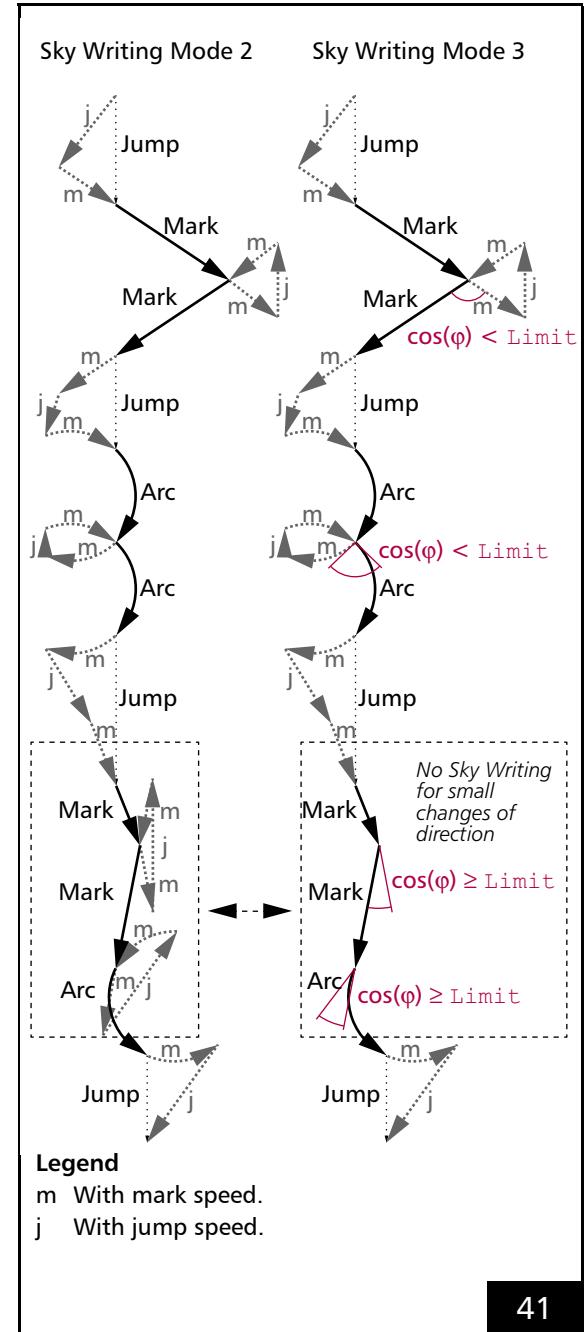
In Sky Writing mode 2, the RTC6 PCIe Board calculates *time-shortened* Sky Writing motions.

Here, too, each to-be-executed vector/arc gets preceded and appended with a run-in motion and a run-out motion in extension of the vector/arc at mark speed. Within a Sky Writing 2 marking sequence, however, neither scanner forerun motions (in the run-in phase) nor retrace motions (in the run-out phase) occur. Instead, the RTC6 PCIe Board executes Sky Writing jumps (at the currently specified jump speed) from jump vector startpoints to run-in startpoints, from run-out endpoints to run-in startpoints, and from run-out endpoints to jump vector endpoints, see figure 40.

`set_sky_writing` and `set_sky_writing_para` always switch on Sky Writing mode 1. Therefore, it is only possible to *change the mode* afterwards to Sky Writing mode 2 by `set_sky_writing_mode` or `set_sky_writing_mode_list`.

Sky Writing mode 2 activation affects the same commands as Sky Writing mode 1 (except ellipse commands, see below). *Time-shortened* Sky Writing, however, can only occur within a sequence of non-parameterized mark, arc and jump commands (may also contain timed or 3D commands). If such a sequence gets interrupted by some other “non-Sky Writing 2-capable” list command (for example, a para, ellipse or any short list command), then the RTC6 PCIe Board suspends Sky Writing mode 2 and complete the preceding command in Sky Writing mode 1 (with a scanner retrace motion). This suspension of Sky Writing mode 2 does not deactivate it: subsequent “Sky Writing mode 2-capable” commands are started at in Sky Writing mode 1 (with a forerun motion of the scan head) and then executed again in Sky Writing mode 2.

Even with Sky Writing mode 2 switched on, ellipse commands are always executed in Sky Writing mode 1.



Sky Writing motions (grey) in Sky Writing mode 2 and mode 3.

### Sky Writing Mode 3

The time cost of Sky Writing motions for vectors and arcs having only small directional changes within a **Polyline** is probably disproportionately high for the gained accuracy.

Therefore, `set_sky_writing_mode` or `set_sky_writing_mode_list` can be used to switch to Sky Writing mode 3. A switching limit angle can be defined for this with `set_sky_writing_limit` or `set_sky_writing_limit_list`.

Then only for larger angle deviations ( $\cos(\phi) < \text{Limit}$ ) between successive **Mark commands** of a **Polyline** a Sky Writing motion is executed as in Sky Writing mode 2.

In contrast, smaller angular changes ( $\cos(\phi) \geq \text{Limit}$ ) result in a (variable) polygonal delay, see Section "Variable Polygon Delay", page 140. See also figure 41.

### Notes

- In case a **Polyline** ends with a short mark vector (shorter than mark speed  $\times$  Timelag) - and a polygon delay has been executed but not a Sky Writing motion - then the length of the short vector (caused by the tracking error) may possibly not achieve the precision expected with Sky Writing.
- At the beginning and end of a **Polyline**, as well as when interrupted with "non-sky writing 2-capable" list commands, a Sky Writing Mode 1 movement always occurs in Sky Writing mode 3 (see Section "Sky Writing Mode 2", page 150).

### Synchronization

The timing diagram for scan-head and laser control in Sky Writing mode 1 is shown in figure 42. The timing diagram for Sky Writing mode 2 and 3 is similar, but here the scanner perform no reverse motion in the run-in and run-out phases.

The `Timelag`, `Nprev`, `Npost` and `LaserOnShift` parameters are specifiable for `set_sky_writing_para` and `set_sky_writing` and the corresponding list commands:

- The `Nprev` parameter is a whole number in units of  $10 \mu\text{s}$  and defines the duration of the run-in:

Mode	Duration
1	$20 \times N_{\text{prev}} [\mu\text{s}]$
2, 3	$10 \times N_{\text{prev}} [\mu\text{s}]$

- The `Npost` parameter is a whole number in units of  $10 \mu\text{s}$  and defines the duration of the run-out:

Mode	Duration
1	$20 \times N_{\text{post}} [\mu\text{s}]$
2, 3	$10 \times N_{\text{post}} [\mu\text{s}]$

- The parameters `Timelag` (64-bit IEEE floating point value rounded to integer multiple of  $1/64 \mu\text{s}$ ) and `LaserOnShift` (whole number in units of  $1/64 \mu\text{s}$ ) define the delay of the signals for "laser active" operation switch-on and switch-off time points relative to the set starting position and set ending position:
  - Delay of switch-on time point relative to the set starting position /  $\mu\text{s}$   
 $= \text{Timelag} + 1/64 \mu\text{s} \times \text{LaserOnShift}$
  - Delay of switch-off time point relative to the set ending position /  $\mu\text{s}$   
 $= \text{Timelag}$



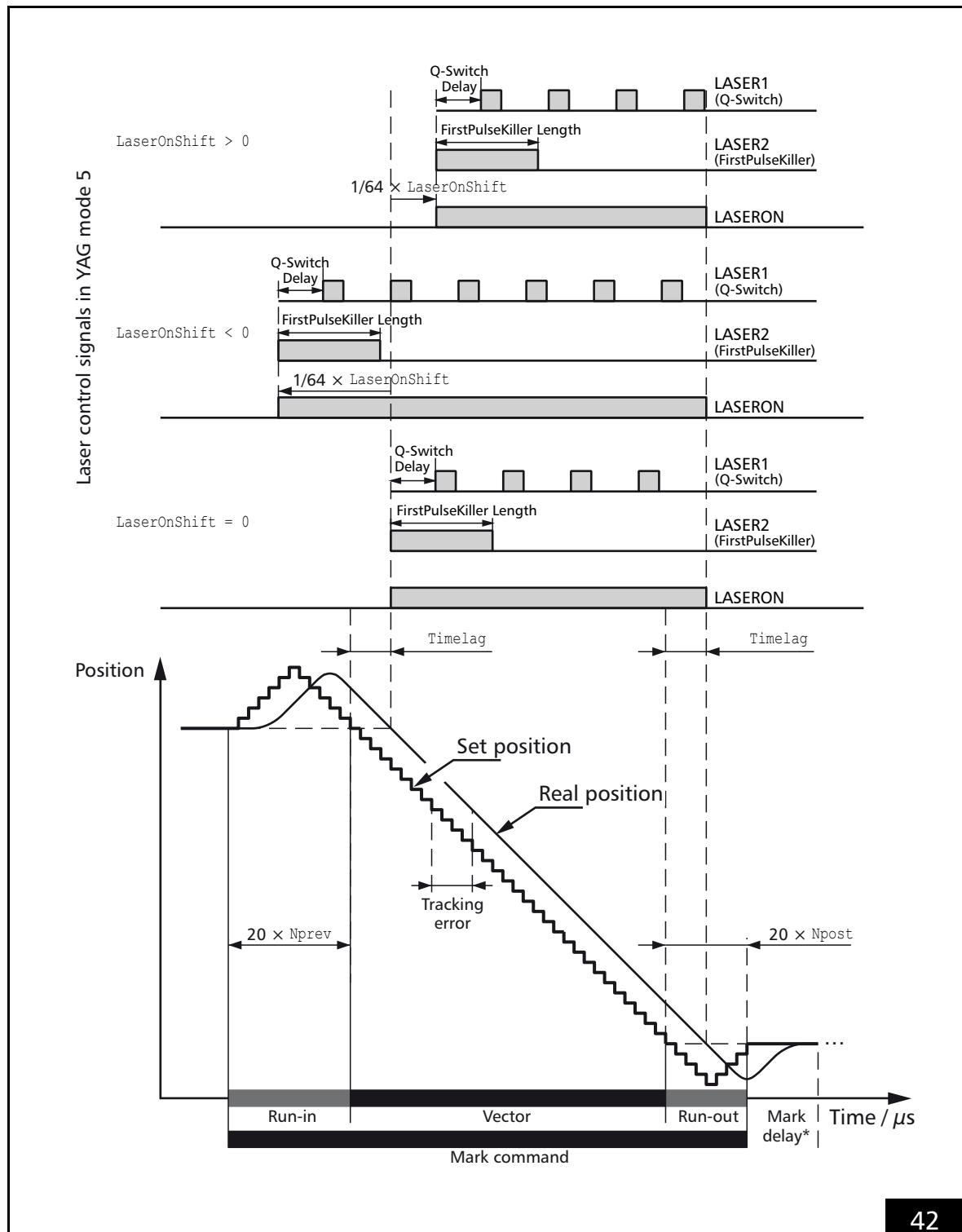
For `LaserOnShift = 0`, the signals for "laser active" operation are switched on (off) with a delay of `Timelag` relative to the set starting position (set ending position).

By setting the parameter `Timelag` to the actual tracking error (and the parameters `Nprev` and `Npost` to sufficiently large values), it is ensured that:

- the signals for "laser active" operation switch on/off precisely at the endpoints of the desired vector
- the tracking error becomes constant even before the vector's startpoint is reached
- the vector is executed right up to its endpoint with constant mark speed

The laser delays from `set_laser_delays` are not effective with Sky Writing.

The switch-on time point of the signals for "laser active" operation can be adjusted with the parameter `LaserOnShift` (for example, to the `HalfPeriod` of `set_laser_pulses` or to the reaction times of the laser itself), so that the first laser pulse occurs exactly with the set starting position. Positive `LaserOnShift` values delay the switching on of the laser control signals. Negative `LaserOnShift` values advance the switching on of the laser control signals (at the earliest, however, until the beginning of the forward run). Negative values are also necessary to "make room" for a Q-Switch signal or a First-PulseKiller signal in one of the YAG modes.



Scan-head and laser control in Sky Writing mode 1 (with parameters Timelag, Nprev, Npost and LaserOnShift)  
 (the laser control signals LASER1 and LASER2 in YAG Mode 5 are exemplarily shown)  
 (\* This mark delay is only effective with Sky Writing mode 3 or Sky Writing mode 0).

## Notes

- By `set_sky_writing` and `set_sky_writing_mode_list`, you can switch back and forth between Sky Writing modes 1 and 2 or 3. Temporarily switching off is also possible, as long as `Timelag > 0`.
  - When searching for appropriate Sky Writing parameters, initially `set_sky_writing` and Sky Writing mode 1 should be used: as start value for the `Timelag` parameter, the tracking delay of the galvanometer scanners. should be set. Then `Timelag` (with `LaserOnShift = 0`) can be fine-tuned such that marking vector *ends* are exactly marked and only afterwards the parameter `LaserOnShift` should be adjusted (keeping constant the parameter `Timelag`), so that the marking vectors' *beginnings* are exactly marked, too.
- In a second step `set_sky_writing_para` can be used to optimize the parameters `Nprev` and `Npost`: `Nprev` and `Npost` may be decreased (relating to the default settings of `set_sky_writing`, see below) to minimize marking times. Alternatively, `Nprev` may be increased to enable a correspondingly large negative `LaserOnShift`.
- With `set_sky_writing` and `set_sky_writing_list`, `Nprev` and `Npost` are predefined:
    - `Nprev = approx. 0.15 × Timelag`
    - `Npost = approx. 0.1 × Timelag`

This results in the following run-in and run-out durations:

	Mode	Duration [Timelag]
Run-in	1	3
Run-in	2, 3	1.5
Run-out	1	2
Run-out	2, 3	1

- With `set_sky_writing_para` or `set_sky_writing_para_list`, you can also specify other run-in and run-out phases, for example, shorter ones to reduce marking times (but this may be at the expense of accuracy). Shorter run-in and run-out phases are particularly beneficial when lines are marked in only one direction without interruption and jump and mark speeds are set equally. Note that in Sky Writing mode no automatic adjustment of laser and scanner delays is performed:
  - If the next `Mark command`'s run-in phase begins when the preceding `Mark command`'s `LaserOn` delay has not yet expired, then the laser does not switch on for the preceding command.
  - If the next `Mark command`'s run-out phase begins when the preceding `Mark command`'s `LaserOff` delay has not yet expired, then the laser does not switch off for the preceding command.
- In Sky Writing mode 1, a positive `LaserOnShift` time (in  $\mu\text{s}$ ) should exceed the smallest occurring marking time by no more than  $(20 \times Npost - Timelag)$  – and in Sky Writing mode 2 and 3 by no more than  $(10 \times Npost - Timelag)$ . Otherwise, the laser is not switched on for this marking.

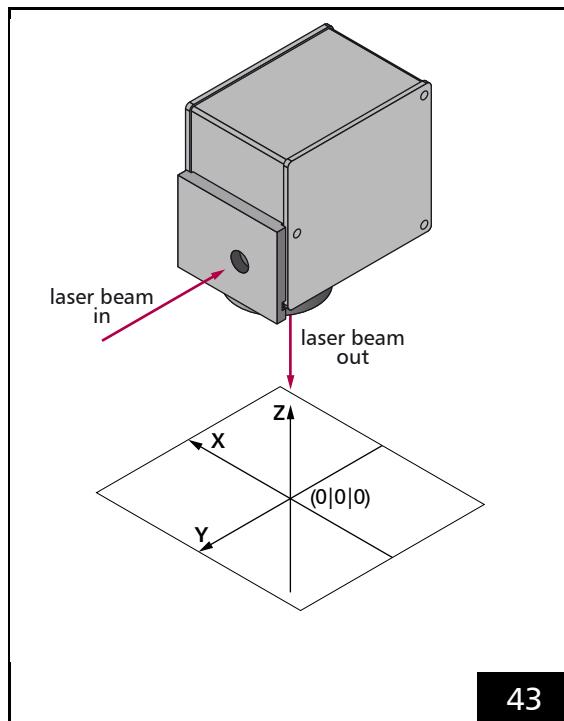
- Pulse lengths and frequencies (in YAG modes additionally the Q-Switch delay and the FirstPulseKiller signal parameters) previously defined for the “laser active” laser control signals are kept unchanged in the Sky Writing mode. On the other hand, previously defined LaserOn, LaserOff, mark, polygon or variable polygon delays are *not* taken into account in the Sky Writing mode. They are fully functional again after deactivation of Sky Writing mode. Deactivation of Sky Writing mode results in addition of a mark delay defined prior to activation (see [figure 42](#)).
- In Sky Writing mode 1, the run-in and run-out phases take place fully within the respective **Mark command**. As a result, execution of the **Mark command** is extended by a time period (in  $10 \mu\text{s}$ ) of  $(2 \times N_{\text{prev}} + 2 \times N_{\text{post}})$ . In Sky Writing mode 2 and 3 execution is extended by a time period (in  $10 \mu\text{s}$ ) of at least  $(N_{\text{prev}} + N_{\text{post}})$ .
- In Sky Writing mode 1, jump delays are performed normally. The jump delay value (in  $10 \mu\text{s}$ , see [set\\_scanner\\_delays](#)) can be reduced by approx.  $(2 \times N_{\text{prev}})$  or even to 0. In Sky Writing mode 2 and 3, the jump delay is automatically (internally, dynamically) reduced by up to  $N_{\text{prev}}$ .
- Time-based mark and arc commands can also be performed in Sky Writing mode (see [Chapter 8.9 “Timed Vector Commands and Timed Arc Commands”, page 260](#)). Here, however, a shorter marking period is coupled with a higher mark speed and therefore with longer starting and ending distances and acceleration phase in the run-in and run-out phases. Therefore,  $N_{\text{prev}}$  and  $N_{\text{post}}$  can be separately adjusted with respect to the specified mark speed.
- the Sky Writing mode is not taken into account (but also not deactivated)
  - During execution of **[\*]para[\*]** commands, for example, with activated “vector-controlled laser control”, see [Section “Vector-Defined Laser Control”, page 195](#)
  - During execution of micro vector commands, see [Chapter 8.8 “Microvector Commands”, page 259](#)
- With Sky Writing mode 2 or 3 activated, the following functionalities of the 2D jump commands **jump\_abs** and **jump\_rel** are *not* executed:
  - Automatic tuning switching in jump mode, see [Chapter 8.1.5 “Jump Mode”, page 203](#)
  - Coordinate transformations with  $\text{at\_once} = 2$ , see list item “With  $\text{at\_once} = 2 \dots$ ”, [page 212](#).

## 7.3 Scan Head Control

### 7.3.1 Reference System

The reference system for the image field which is used by the RTC6 PCIe Board is shown in [figure 43](#).

The y axis points in the *reverse* direction of the input laser beam, the z axis points in the *reverse* direction of the output laser beam. x axis, y axis and z axis form a right-handed reference system. The origin of the reference system, that is, the point (0|0|0), is in the center of the image field.



Reference system for the RTC6 coordinates.

### 7.3.2 Image Field Size and Image Field Calibration

The size of the usable image field is determined by the maximum scan angle and the focal length of the objective or the working distance (that is, the distance between the input laser beam axis and the image field).

The x, y and z coordinates of a vector must be specified as signed 20-bit values (that is, as numbers between -524,288 and +524,287).

The calibration factor  $K$  defines the ratio of the digital point coordinates in *bits*<sup>(1)</sup> and the actual position of the point in *millimeters*.

Let  $a_0$  denote the side length of the image field given by the maximum scan angle. The theoretical calibration factor is then  $K_0 = 2^{20}/a_0$  [bits per mm<sup>(1)</sup>].

SCANLAB provides a rounded value for the calibration factor  $K$ . This value is slightly larger than, but close to, the theoretical value. The actual calibration factor  $K$  can be read out from a used correction table by [get\\_table\\_para](#) or [get\\_head\\_para](#).

Given the calibration factor  $K$ , the side length  $a$  of the usable image field in *millimeters* can be calculated:

$$a = \frac{2^{20}}{K}$$

(1) The expression "bits" is here synonymous with "digital control value" (see footnote <sup>(3)</sup> on [page 125](#)).

## Typical Image Field

In general, the size of the usable (or “typical”) image field – dependent on the objective and the optical configuration of the scan system – is smaller than the maximum adjustable image field.

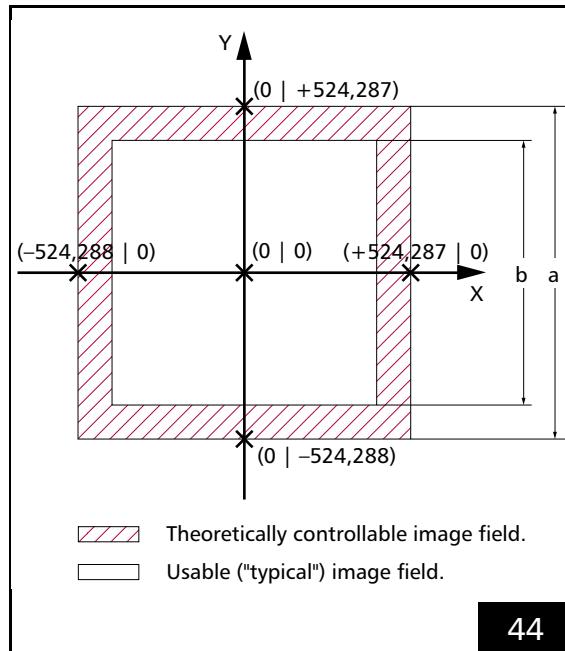


Image field size.

The scan head has a usable image field. If the laser focus moves outside this field, some vignetting of the laser beam can occur. The interior of the scan head can be damaged due to excessive absorption of laser power. Refer to your scan head operating manual’s section on objectives.

- Compare the calculated side length  $a$  of the maximum adjustable image field with the side length  $b$  of the usable image field given in the technical specifications of your scan head manual (see [figure 44](#)).
- If the laser focus is to be restricted to points within the usable image field, the absolute values of the  $x$  and  $y$  coordinates (in bits) must be smaller than the maximum value  $M$ , where  $M$  is the calibration factor  $K$  multiplied by *half* the side length of the usable image field:

$$M = K \times b/2$$

## Compatibility Modes

### RTC6 Standard Mode

(Default after [load\\_program\\_file](#) and [set\\_RTC6\\_mode](#))

The image field coordinates for the  $x$ ,  $y$  and  $z$  axis and all related parameters (for example, jump speed or wobble amplitude) are to be specified as 20-bit values.

### RTC5 Compatibility Mode

([set\\_RTC5\\_mode](#))

The image field coordinates for the  $X$  and  $Y$  axes and all associated parameters are to be specified as 20-bit values – as in [RTC6 Standard Mode](#). See also [Chapter 2.9.2 “Adapting RTC5 Source Code for the RTC6 PCIe Board”, page 49](#).

The image field coordinates of the  $z$  axis are to be specified as 16-bit values. The RTC6 PCIe Board automatically multiplies all  $z$  coordinates by  $16^{(1)}$ . As the  $z$  calibration factor, a value 16 times smaller must be used.

### RTC4 Compatibility Mode

([set\\_RTC4\\_mode](#))

The image field coordinates for the  $x$ ,  $y$  and  $z$  axis and all related parameters are to be specified as 16-bit values. The RTC6 PCIe Board automatically multiplies them by  $16^{(1)}$ . As the  $xy$  and  $z$  calibration factor, a value 16 times smaller must be used. See also [Chapter 2.7.2 “Porting RTC4 Source Code to the RTC6 PCIe Board”, page 39](#).

(1) The allowed value range decreases accordingly.

### 7.3.3 Virtual Image Field

In particular for Processing-on-the-fly applications, a virtual image field of size 29-bit<sup>(1)</sup> is available.

Therefore, vector commands and arc commands can be loaded for objects up to 512 times larger than the real image field (in the Processing-on-the-fly direction). For details on Processing-on-the-fly in the virtual image field, see [Chapter 8.6.6 "Virtual Image Field with Processing-on-the-fly", page 237](#).

At runtime, the current coordinates are clipped to the real image field [-524,288...+524,287], see [2...7](#) in [Chapter 7.3.6 "Output Values to the Scan System", page 170](#).

In addition, the extended value range of the virtual image field can also be used for utilizing the complete real 20-bit image field during coordinate transformations (such as rotations, shrinkages or shifts, see [Chapter 8.2 "Coordinate Transformations", page 210](#)). Otherwise, certain edge areas of the real image field may remain inaccessible during such coordinate transformations.

#### Coordinate Transformations in the Virtual Image Field

"Virtual coordinate transformations" (particularly translations and rotations) are sometimes needed to compensate certain mechanical tolerances of object positioning when performing continuous marking larger than the real image field.

See [3](#) in [Chapter 7.3.6 "Output Values to the Scan System", page 170](#).

"Virtual coordinate transformations" are defined by:

- `set_matrix( HeadNo = 4 )`
- `set_offset( HeadNo = 4 )`
- `set_offset_xyz( HeadNo = 4 )`
- `set_angle( HeadNo = 4 )`

They let you define matrix coefficients and 2D offsets (with `set_offset_xyz`, `zOffset` is ignored). With `at_once = 1`, they become effective immediately, if no list is currently being processed. Otherwise, they are only saved as with `at_once = 0`.

Stored transformations are automatically activated when a [Processing-on-the-fly session](#) is restarted (the changed output position is reached at jump speed). They remain active<sup>(2)</sup> even after the end of this [Processing-on-the-fly session](#).

As long as a [Processing-on-the-fly session](#) is active, the parameters for the "virtual coordinate transformation" cannot be changed, they can only be saved.

By `set_matrix( 4, 0.0, 0.0, 0.0, 0.0 )`, the "virtual coordinate transformation" can be deactivated again, see [Section "Clock Overruns", page 171](#).

The adjustment range for offsets is  $\pm 28$  bits. Matrix coefficients must not exceed an absolute value of 2.0. The offset is applied after the matrix operation.

For "virtual coordinate transformations" *cannot* be used:

- `set_scale`
- all list commands for coordinate transformations

With RTC6 Software Package  $\geq$  V1.6.1, "virtual coordinate transformations" can be also defined by a ["Global Online Positioning"](#), see [Chapter 8.3.2 "Global Online Positioning", page 217](#).

(1) Up to DLL 608: 24-bit.

(2) With RTC6 Software Package < V1.5.0, "virtual coordinate transformations" are only available during a `set_fly_2d` [Processing-on-the-fly session](#).

### 7.3.4 3D Image Field

#### Adjustment

SCANLAB 3D correction tables are calculated such that the plane of the middle focus position is controlled with  $z = 0$ .

Therefore, the mechanical distance between scan system and working plane must be adjusted accordingly. With the varioSCAN series, a specific distance to the scan system has to be maintained in addition. The values are to be taken from the manual of the respective 3-axis scan system or varioSCAN.

Deviations from these should preferably be set via the user program by `set_defocus`, `set_defocus_offset`, `set_offset_xyz` or the corresponding list commands.

Once the working distance and distance to the scan system are correctly adjusted, then subsequently the laser focus position should be fine-tuned. For this, a test pattern is to be marked in the middle of the  $z = 0$  working plane. The optimum laser focus position for processing results can be achieved:

- With the varioSCAN series, by manually turning the focusing ring, see corresponding Manual
- With a varioSCAN FLEX, by adjusting the focusing optic position, see Manual for #128683 RTC5/6 varioSCAN 40 FLEX Extension
- With a varioSCAN FC and intelliWELD FC by adjusting the coefficient A of the used 3D correction table<sup>(1)</sup>

#### Checking the z axis Calibration

The optimum output values for the z axis also depend on various parameters such as beam divergence of the used laser and tolerances of the optical components.

Such information is generally not available to SCANLAB and therefore, are not included in the calculation of 3D correction tables.

Therefore, in some cases the calculated 3D correction table might not fit optimally the individual scan system. To test whether this is the case, run a laser marking test that covers the entire **3D image field**.

Check if the laser focus meets the requirements of your application. If you find that the spot diameter varies considerably, then a recalibration of the z axis correction may help under certain circumstances.

The aim of the z axis calibration procedure is to determine suitable coefficients A, B and C. These can be transferred to the RTC6 PCIe Board by `load_z_table`.

A, B and C are coefficients of the parabolic function

$$z_{\text{out}} = A + Bl + Cl^2$$

They determine the relationship between the z output value  $z_{\text{out}}$  and the focus length value  $l$ . For each point  $(x|y|z)$  in the **3D image field**, the focus length value  $l$  corresponds to the focus length difference between the specified point  $(x|y|z)$  and the point  $(0|0|0)$ .

The ABC coefficient values of a correction file on the PC can be read-out directly with `read_abc_from_file` and written into with `write_abc_to_file`.

(1) Coefficients A, B and C can be queried by `get_head_para` and newly set by `load_z_table`. Only A should be modified. B and C should be passed over unchanged – as queried – by `load_z_table`.

### Procedure

- (1) Adjust the correct mechanical distance between the scan system and the  $z = 0$  working plane and the correct distance between the varioSCAN device and the scan system<sup>(1)</sup>.
- (2) Load the 3D correction file by **load\_correction\_file**.
- (3) Assign the 3D correction table by **select\_cor\_table**.
- (4) Read out the assigned coefficients  $A$ ,  $B$  and  $C$  by **get\_head\_para**.
- (5) varioSCAN FC and intelliWELD FC only: proceed with step 10. Steps 6...9 do not need to be performed.
- (6) Move the laser spot to the reference point<sup>(2)</sup> by **goto\_xyz**.
- (7) Set the z axis to the neutral (middle) position by **load\_z\_table(0, 0, 0)**.
- (8) Place a test object at the reference point.
- (9) Adjust the laser focus, see [page 159](#).
- (10) Move the focus to an arbitrary point  $(x|y|z)$  within the (required) 3D image field by **goto\_xyz**<sup>(3)</sup>.
- (11) Query the focus length value  $l$  (in bits) set by the RTC6 PCIe Board for this point<sup>(4)</sup> by **get\_z\_distance**.
- (12) Optimize the laser focus at this point:  
vary the Z output value  $z_{out}$  until the quality of the laser focus meets your requirements<sup>(5)</sup> by **load\_z\_table(A=z\_out, 0, 0)**.  
A starting value can be calculated in accordance with  $A + Bl + C l^2$  by using the previously read focus length value  $l$  and the previously read or used values  $A$ ,  $B$  and  $C$ .

- (1) See the corresponding manual.
- (2) The reference point is a point in the  $z = 0$  working plane for which a middle focus length value is required for a sharp laser focus. The laser beam should be focused to the reference point when the z axis is set to the neutral position (Z output value  $z_{out} = 0$ ). The coordinate values of the reference point (in mm) are provided in the `Readme.txt` file that accompanies the 3D correction file or in the 3-axis scan system's or the varioSCAN's user manual. If you are using an F-Theta objective, the reference point is generally the origin  $(0|0|0)$ .
- (3) If you are using a scan system with an F-Theta objective, it is sufficient to select various points  $(0|0|z)$  on the z coordinate axis. The maximum possible **3D image field** is specified in the corresponding user manual.

(13) Repeat steps 10...12 for as many locations  $(x|y|z)$  as possible<sup>(6)</sup> and write down the values  $(l|z_{out})$  for each new point. If possible, seek to thereby cover the entire **3D image field** required by your application.

(14) Fit the function  $z_{out} = A + Bl + C l^2$  to your value pairs  $(l|z_{out})$ .

(15) Use the resulting coefficients  $A$ ,  $B$  and  $C$  to adjust the 3D correction table by **load\_z\_table(A, B, C)**.

Values loaded by **load\_z\_table** are overwritten by a subsequent **load\_correction\_file**

(**load\_correction\_file** sets the three coefficients  $A$ ,  $B$  and  $C$  to the values of the loaded correction table). After each **load\_correction\_file** or **select\_cor\_table**, you should therefore also call **load\_z\_table(A, B, C)** again.

By **load\_z\_table\_no**, the three coefficients  $A$ ,  $B$ ,  $C$  can be assigned to correction table **No**. They are switched with **select\_cor\_table**.

Alternatively, the ABC values can be written permanently to the header of the correction file by **write\_abc\_to\_file**, see also parameter 5...7 in Section "ct5 Correction File Header", page 167.

- (4) The focus length value  $l$  can be positive or negative.
- (5) The optimal  $z_{out}$  output value can be positive or negative.
- (6) Note that larger z control values lead to shorter working distances and that the focus thereby shifts toward the scan system. The test object might have to be tracked accordingly.



## Test for 3-Axis Scan Systems with F-Theta Objective

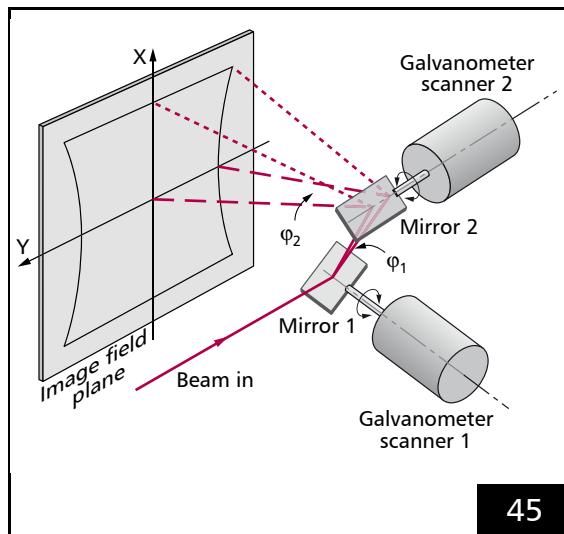
- With the adjusted 3D correction table and 3D vector commands, see [Chapter "3D Commands", page 223](#), mark two identical large squares. Mark one of the squares with the work piece in z position  $z = z_{\min}$ , the other one with  $z = z_{\max}$ .
- These two squares should match exactly. If this is not the case, your correction file is not perfectly suited to your objective. To solve this problem, measure the size (x and y) of both squares and report these values to SCANLAB. You will then receive a new 3D correction file.
- See also [Section "Enhanced 3D Correction", page 225](#).

### 7.3.5 Image Field Correction and Correction Tables

#### Field Distortion

The deflection of a laser beam with a two-mirror system results in three effects:

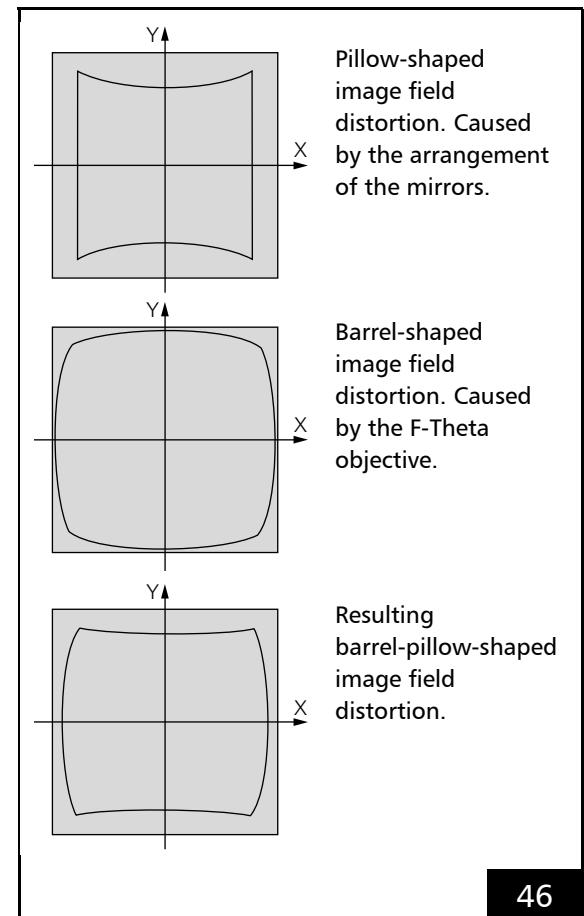
- (1) The arrangement of the mirrors leads to a certain distortion of the image field<sup>(1)</sup>, see figure 45.
- (2) The distance in the image field is not proportional to the scan angle itself, but to the tangent of the scan angle. Therefore, the mark speed of the laser focus in the image field is not proportional to the angular velocity of the corresponding scanner.
- (3) If an ordinary lens is used for focusing the laser beam, the focus lies on a sphere. In a flat image field plane, a varying spot size results.



45

Image field distortion when deflecting a beam in a two-mirror deflection system.

By focusing the deflected laser beam with an F-Theta objective, effect 2 and 3 can be avoided. However, this causes a barrel-shaped distortion of the image field, see figure 46.



46

Image field distortion caused by the arrangement of the mirrors and by the F-Theta objective.

- (1) Cause: the distance between mirror 1 and the image field depends on the size of the scan angles of mirror 1 and mirror 2. A larger scan angle leads to a longer distance.



## Field Correction Algorithm

For these field distortions, the RTC6 PCIe Board board internally uses a algorithm to compensate for it. The algorithm is based on a correction table.

An orthogonal grid of  $257 \times 257$  points is superimposed on the ideal square image field. The adjusted x and y coordinates for a corrected output of these grid points are stored in a correction table.

To move the focus to any point within the image field, the RTC6 PCIe Board calculates the corrected coordinates by interpolating from the grid points in the correction table.

The correction is executed for every single microstep, see also [Chapter 7.1.2 "Microstepping", page 129](#).

SCANLAB creates a correction file for every system type. For this purpose, the correction table is calculated based solely on general system data (such as mirror geometry, calibration and the objective specifications). Standard correction files therefore reflect neither the unique properties possessed by the customer's individual system, nor alignment errors.

For those customers requiring more accurate correction tables tailored to the unique properties of their particular scan systems, SCANLAB offers the correXion program line. These generate RTC correction files based on test measurement data. For further information, refer to the corresponding manual or contact SCANLAB).

## Activating Image Field Correction

To activate image field correction, at the beginning of a user program the required correction tables must:

- (1) be loaded from the corresponding correction files into RTC6 PCIe Board memory (by [load\\_correction\\_file](#))
- (2) assigned to the used scan head connectors (by [select\\_cor\\_table](#) or [select\\_cor\\_table\\_list](#))

2D correction tables activate an image field correction for x and y coordinates,

3D correction tables additionally for z coordinates.

## 2D and 3D Correction Files

SCANLAB supplies 2D and 3D correction files. They have the extension \*.ct5 and the following naming scheme applies:

- D2\_xxx.ct5 for 2D correction files
- D3\_yyy.ct5 for 3D correction files

Here, xxx and yyy are numbers. Each correction file is calculated for a specific optical configuration. The configuration is specified in the accompanying Readme.txt file and in parameters of the correction file header, see also [Section "ct5 Correction File Header", page 167](#).



## Loading Correction Tables

By `load_correction_file`, up to 8 correction tables can be loaded from their corresponding correction file into the RTC6 PCIe Board memory, see also [Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 222](#).

If the **Option "3D"** is not enabled, then only 2D correction tables can be loaded.

2D correction tables can also be loaded from 3D correction files (by `load_correction_file` with `Dim = 2`). The 3D part is ignored.

If the **Option "3D"** is enabled, a 3D correction table can even be loaded from a 2D correction file (by `load_correction_file` with `Dim = 3`). Thereby, the 2D correction table is automatically supplemented with a linear z correction.

The actually suitable Z correction can be loaded by `load_z_table_no` or `load_z_table` after the assignment by `select_cor_table` (see below), see [Chapter 7.3.4 "3D Image Field", page 159](#).

### Notes

- RTC5/RTC6 correction files (file extension `*.ct5`) differ from those of prior RTC products (file extension `*.ctb`) in terms of size, structure and content.
- If `*.ct5` and `*.ctb` correction files have the same file name (except for the different extensions), then they were calculated for the same optical configuration.
- For converting older correction files, see [Section "Converting Correction Files", page 169](#).

## Assigning Loaded Correction Tables

Loaded correction tables are assigned to the two scan head connectors by `select_cor_table` or `select_cor_table_list`.

If neither the **Option "Second Scan Head Control"** nor the **Option "3D"** is enabled, then a 2D correction table can be assigned only to the first scan head connector.

If the **Option "Second Scan Head Control"** is enabled, then the two scan head connectors can each be assigned their own 2D correction table.

If the **Option "3D"** is enabled but not the **Option "Second Scan Head Control"**, then a 2D correction table or a 3D correction table can be assigned exclusively to the first scan head connector. The first scan head connector outputs position signals for an xy scan system.

With a 3D correction table, the second scan head connector outputs additionally position signals for the z axis (for example, varioSCAN) via both channels.

If the **Option "Second Scan Head Control"** and the **Option "3D"** are enabled, up to two (even different) 2D correction tables or a single 3D correction table can be assigned.

In order to control a 3D system, the (only) 3D correction table must be assigned exclusively to the connector for the xy scan system. There is no assignment for the z axis. The output for the z axis occurs automatically on both channels of the other port (`select_cor_table( n, 0 )` or `select_cor_table( 0, n )`).

## Notes

- If no correction table has been loaded into the RTC6 PCIe Board memory, then the board outputs unforeseen values to the scan system. Therefore, at least a 1to1 correction table must be loaded (as 2D or 3D correction table), see [Section "1to1 Correction Tables", page 166](#), if no 2D correction file or 3D correction file is available that has been calculated for the specific optical configuration.
- Correction files should have been assigned<sup>(1)</sup>:
  - before a list is started for the first time or
  - before the galvanometer scanners of a scan system are moved by a control command (like `goto_xy`)
- If only one scan head connector has an 2D correction table assigned, then the other scan head connector outputs no *position* signals.
- During initialization, `load_program_file` assigns a correction table according to `select_cor_table( 1, 0 )`. However, it leaves the galvanometer scanners at position (0,0). `load_program_file` cannot determine whether a correction table #1 is loaded at all. Therefore, there is no internal jump to the output position that is specified in the correction file. Its contents could be random (see above).

- `load_program_file` does not initialize the memory contents of the correction tables. The correction table values are random immediately after switching on the RTC6 PCIe Board. It is recommended to explicitly call `select_cor_table`. Before returning, `select_cor_table` itself internally waits for the end of the jump.
- If `load_correction_file` is called after `load_program_file`, then:
  - `load_correction_file` automatically executes a `select_cor_table( HeadA, HeadB )` with the last used values for `HeadA` and `HeadB`
  - `load_correction_file` positions the galvanometer scanners with an internal jump at the currently set jump speed to the output position specified there
- If `load_correction_file` is called prior to `load_program_file`, then the correction table is only saved. There can be no internal jump to the intended output position.

(1) Correction files can also be loaded before the RTC6 files have been loaded by `load_program_file`.



## 1to1 Correction Tables

1to1 correction files

- Are not assigned to a particular scan system
- In general, serve test purposes only
- Alternatively, can be loaded by  
**load\_correction\_file** with parameter  
Name = **NULL**.
  - According to the further **Dim** parameter a 2D or 3D 1to1 correction table is being generated internally.

Because some user programs require a file name and do not accept 0 as the name, the 1to1 correction file Cor\_1to1.ct5 is supplied within the RTC6 Software Package. This contains a calibration factor of value 0. If a user program strictly needs an "authentic" calibration factor, a corresponding value can be specified with CorrectionFileConverter.exe, see **Section "Folder CorrectionFileConverter"**, page 24 (button **Show File Header** > field 'Field Calibration [Bit/mm]').

## Inverse Tables

The ct5-correction file format supports "inverse tables". These are required for back transforming actual scan head positions to the associated image field coordinates.

For further information, see:

- Parameter **11**, see [page 168](#)
- **Chapter 8.1.3 "Monitoring the Positioning"**, page 201
- **Section "Converting Correction Files"**, page 169

## ct5 Correction File Header

The ct5-correction file header contains 16 parameters, see following table.

These can be read out and thus directly incorporated into a user program:

- from the currently loaded correction tables by [get\\_table\\_para](#)
- from assigned correction tables by [get\\_head\\_para](#)

## Notes

- With ctb-correction files, some parameter values can exclusively be taken from its associated ReadMe file. These must be transferred manually to the user program.
- A ct5 file created by converting another ctb file, see [Section "Converting Correction Files", page 169](#), does not contain all parameter data.
- A 1to1 correction file does not contain all parameter data.
- Parameters 3...7 are only relevant for 3D marking. In 2D correction tables, they are 0.
- For the same 3-axis scan system, the ABC coefficients (parameter 5...7) in ctb-correction files and ct5-correction files are identical.

Parameter <sup>(a)</sup>	Description
0	Type of the correction table. <ul style="list-style-type: none"> <li>• = 0.0: 2D correction table</li> <li>• = 1.0: 3D correction table</li> </ul>
1	Calibration factor $K_{xy}$ [bit/mm]. See <a href="#">Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156</a> .
2	Focal length or working distance [mm]. <ul style="list-style-type: none"> <li>• For a configuration with a scan objective: the effective focal length of the objective [mm].</li> <li>• For a configuration without a scan objective: the working distance A [mm]. A = distance from the optical axis of the incident laser beam at the first deflection mirror to the image plane.</li> </ul>
3	Stretch factor for the x direction. Compensates the pyramid-shaped image field change which exists in the z direction of 3D markings.
4	Stretch factor for the y direction. See parameter 3.
5	Coefficient A of the polynomial for z axis control, see <a href="#">page 159</a> : offset part, $\pm 26$ Bit.
6	Coefficient B of the polynomial for z axis control, see <a href="#">page 159</a> : linear part, $\pm 11$ Bit.
7	Coefficient C of the polynomial for z axis control, see <a href="#">page 159</a> : square part, $\pm 4$ Bit.
8	Number of the correction file. With correction files supplied by SCANLAB, the parameter corresponds to the number in the file name (for example, 145 for D2_145.ct5 or D3_145.ct5).

Parameter <sup>(a)</sup> (cont'd)	Description (cont'd)
9	<p>Differentiation between correction with or without an F-Theta objective. The following applies: Parameter = <math>10 \times P_{Obj} + P_{Typ}</math> with</p> <ul style="list-style-type: none"> <li>• <math>P_{Obj} = 0</math>: Correction without F-Theta objective</li> <li>• <math>P_{Obj} = 1</math>: Correction with F-Theta objective</li> <li>• If correction with F-Theta objective:           <ul style="list-style-type: none"> <li><math>P_{Typ} = 0.0</math>: without distortion data</li> <li><math>P_{Typ} = 1.0</math>: with F-Theta's F-stop progression condition</li> <li><math>P_{Typ} = 2.0</math>: with image height table</li> </ul> </li> </ul>
10	<p>Indicator for the source of the correction file. The following applies: Parameter = <math>1000 \times P_{Orig} + P_{Ver}</math> with</p> <ul style="list-style-type: none"> <li>• <math>P_{Orig} = 10000</math>: Originally calculated file</li> <li>• <math>P_{Orig} = 20000</math>: converted from ctb file</li> <li>• <math>P_{Orig} = 30000</math>: reconstructed from txt file</li> <li>By manipulating a correction file using correction programs available from SCANLAB, <math>P_{Orig}</math> is increased by 1 in each case.</li> <li>– <math>P_{Ver}</math> = Version number of the program used to create the correction file</li> </ul>
11	<p>Information about the inverse table. The following applies: Parameter = <math>P_{Exist} + 2 \times P_{Calc}</math> with</p> <ul style="list-style-type: none"> <li>• <math>P_{Exist} = 1.0</math>: valid inverse table is present</li> <li>• <math>P_{Exist} = 0.0</math>: no valid inverse table present</li> <li>• If valid inverse table is present:           <ul style="list-style-type: none"> <li><math>P_{Calc} = 0</math>: inverse table calculated ab initio</li> <li><math>P_{Calc} = 1</math>: inverse table numerically calculated</li> </ul> </li> </ul>
12	<p>Angle calibration of the scan system. Mechanical angle deflection in [<math>\pm ^\circ</math>] at 96% of the maximum control.</p>
13	<p>Code for the scan head geometry used for the calculation (for internal use only), for example,</p> <ul style="list-style-type: none"> <li>• = -1.0: unknown geometry (for example, for a table converted from a ctb file)</li> <li>• = 0.0: standard geometry</li> </ul>
14	<p>Indicator for whether an additional protective window has been taken into account. The following applies: Parameter = <math>1,000,000 \times P_T + 1,000 \times P_I</math> with</p> <ul style="list-style-type: none"> <li>• <math>P_T</math> = Protective window thickness in mm (max. 2 decimal places)</li> <li>• <math>P_I</math> = Refraction index (max. 3 decimal places)</li> </ul> <p>Example: The value 3,521,450.0 corresponds to a protective window thickness of 3.52 mm and a refraction index of 1.450.</p>
15	<p>Indicator for whether the image field size has been limited in the correction file.</p> <ul style="list-style-type: none"> <li>• = 0.0: without field size limit</li> <li>• = 2.0: with field size limit</li> </ul>

(a) Numbering according [get\\_table\\_para](#) and [get\\_head\\_para](#).



## Converting Correction Files

The RTC6 Software Package includes the program `CorrectionFileConverter.exe` for converting `ctb` correction files to `ct5` correction files and vice versa. The corresponding manual is supplied in English and German.

When converting a `ctb`-correction file into a `ct5`-correction file, the `ct5`-correction file header receives a corresponding origin indicator (parameter [10, page 168](#),  $P_{\text{Orig}} = 20000$ ).

Such conversions do *not* produce fully complete `ct5`-correction files because the file header lacks several parameters. These can be subsequently entered manually by `CorrectionFileConverter.exe` (via button **Show File Header**).

Moreover, it may happen that the inverse correction table to be calculated numerically during the conversion does not cover the entire image field or cannot be calculated. In this case, a 1to1 correction table is inserted instead, see also parameter [11, page 168](#).

In contrast, converting a `ct5`-correction file into a `ctb`-correction file results in a fully complete `ctb`-correction file.

Parameter information from the `ct5`-correction file header are:

- no longer contained in a `ctb`-correction file for 2D correction table
- contained partially in a `ctb`-correction file for 3D correction tables

### 7.3.6 Output Values to the Scan System

#### Calculation

After microstepping (1), the following corrections are applied in the order given to calculate the output values from the current x, y, and z coordinate values.

- (1) The split-up of vectors and arcs to microsteps.
- (2) A wobble motion that has been set, see [Chapter 8.4 "Wobble Mode", page 218](#), for example, by `set_wobble_mode`.
- (3) A global coordinate transformation in the virtual image field, for example, by `set_matrix`, `set_angle` and `set_offset` with `HeadNo = 4`, see [Section "Coordinate Transformations in the Virtual Image Field", page 158](#)
- (4) A Processing-on-the-fly correction that has been set, see [Chapter 8.6 "Processing-on-the-fly", page 227](#), for example, by `set_fly_x` or `set_fly_x_pos`.
- (5) A 2D coordinate transformation for aligning the scan system relative to the image field, see [Chapter 8.2 "Coordinate Transformations", page 210](#), for example, by `set_matrix`, `set_scale`, `set_angle` and corresponding list commands.
- (6) A 3D coordinate shift that has been set, see [Chapter 8.5.2 "3D Scan Systems", page 222](#), for example, by `set_offset_xyz` or `set_offset_xyz_list`.
- (7) Coordinate values that exceed the real image field range (virtual image field), see [Chapter 7.3.3 "Virtual Image Field", page 158](#), are clipped to the boundary values of the real 20-bit image field.
- (8) A 2D/3D image field correction
  - correspondingly to the correction table that has been assigned by `select_cor_table` or `select_cor_table_list`, see [Chapter 7.3.5 "Image Field Correction and Correction Tables", page 162](#)
  - and a focus shift that has been set for example, by `set_defocus` or `set_defocus_list`.

- (9) A gain correction and offset correction for the x and y galvanometer scanner. These can be set:
  - explicitly by `set_hi`, see [Section "Customer-Specific Calibration", page 264](#)
  - automatically by `auto_cal`, see [Chapter 8.10 "Automatic Self-Calibration", page 261](#)

#### Notes

- Overflowing values are always clipped to the boundary values of the maximum possible value range.
- A complete 3D calculation is always performed. If the **Option "3D"** is not enabled, Z values are just not outputted.

#### Value Ranges

For all scan system axes, signed 20-bit values ( $-524,288 \dots +524,287$ ) are outputted. This also applies to values which the scan system returns to the RTC6 PCIe Board, see also:

- [Section "RTC4 Compatibility Mode", page 157](#)
- [Section "RTC5 Compatibility Mode", page 157](#)
- [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)", page 58](#)

#### Precalculation and Diagnosis

With `get_galvo_controls` the output values resulting for the given coordinates and setting parameters in the current configuration (correction table, coordinate transformations) can be determined without the galvanometer scanners moving.



## Clock Overruns

The  $10 \mu\text{s}$  clock period might not always suffice for calculating all data required by the computation-intensive jump, mark and arc commands if several of the available command options are utilized simultaneously – for example, simultaneous control of two scan systems, wobble motion, coordinate transformation in the virtual image field, Processing-on-the-fly for two axes (correction by [McBSP interface](#)), “Automatic Laser Control”, para vectors, data recording, “variable polygon delay”, short list commands (for example, in a [Polyline](#)), control commands during list execution.

This overrun situation can be internally detected and counted. You can appropriately test your user program by using [`get\_overrun`](#) to count the number of overruns. Such overruns result in one or several peripheral ports not being accessible during the current  $10 \mu\text{s}$  clock period, possibly including output to the scan head. The galvanometer scanner motion might also could pause for  $10 \mu\text{s}$ .

### 7.3.7 Status Monitoring and Diagnostics

For status monitoring and diagnostic purposes, `get_value` or `get_values` can be used to read out a variety of signals:

- A 20-bit status word which is returned by the scan system, see [Section "Status Information Returned from the Scan System", page 172](#)
- The current LASERON signal
- The current Cartesian control values (that is, the so-called "sample values" common to both scan head connectors, see [2](#), [3](#) and [4](#)).
- The control values specific to each scan head connector which take into account
  - any coordinate transformations defined by `set_matrix`, `set_scale`, `set_angle` or by the corresponding list commands, see [5](#).
  - any z coordinate offset defined by `set_offset_xyz` or `set_offset_xyz_list`, see [6](#).
  - any focal length offset defined by `set_defocus` or `set_defocus_list` and any loaded correction table, see [8](#) and [9](#).

Each of these signals can be recorded on the RTC6 PCIe Board for a longer time period by `set_trigger/set_trigger4` – with a selectable sample period. After that, `get_waveform` can be used to transfer them to the PC for analysis.

By `set_trigger( Period Bit #31 = 1 )` (ring buffer functionality), practically arbitrary amounts of data can be recorded. The recorded values can be transferred to the PC in blocks during list execution using `get_waveform_offset`. After the measurement memory has reached its capacity, measurement value recording automatically starts from the beginning.

The current status and progress of a measurement session started with `set_trigger/set_trigger4` can be queried by `measurement_status`.

The returned values are always 20-bit values.

### Status Information Returned from the Scan System

The scan system transmits the following signals to the RTC6 PCIe Board every  $10 \mu\text{s}$  via the SL2-100 protocol:

- A 20-bit status word.  
It can mean the following data types:
  - The actual status word (the upper 16 bits are identical to the 16-bit status word of the XY2-100 protocol, for a description see `get_head_status`).
  - Only for iDRI<sup>E</sup> scan systems<sup>(1)</sup>, see [Chapter 8.1 "iDRI<sup>E</sup> Functions", page 199](#): a different data type selectable with `control_command`.

The 20-bit status word can:

- be read out by `get_value/get_values`
- be recorded by `set_trigger/set_trigger4`

- 6 additional status bits.

With iDRI<sup>E</sup> scan systems<sup>(1)</sup> with SL2-100 protocol, the status bits (PowerOK, TempOK and PosAck; per axis) are transferred in parallel to and independently from the 20-bit status word. Therefore, these status bits can even be used to monitor the scan system (see [Section "Automatic Suppression of Laser Control Signals", page 176](#)) if, for example, an "Automatic Laser Control" (see [Chapter 7.4.9 ""Automatic Laser Control""](#), page 186) is active that requires a different return data type.

The 6 additional status bits can be read out by `get_head_status`.

### Notes

- Scan systems with XY2-100 protocol require the use of the SCANLAB XY2-100 converter, see [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)", page 58](#).
- For iDRI<sup>E</sup> scan systems<sup>(1)</sup> with XY2-100 protocol, the actual status word must be set as the to-be-returned data type (default type after switching on).

(1) See glossary entry on [page 879](#).

## 7.4 Laser Control

At its laser signal output ports (LASERON, LASER1 and LASER2), the RTC6 PCIe Board outputs signals which can be used for controlling various laser types (“laser control signals”).

The laser signal output ports are part of:

- the D-SUB-LASER connector, see [Chapter 4.6.1 “LASER Connector”, page 62](#)
- the EXTENSION 2 socket connector, see [Chapter 4.6.3 “EXTENSION 2 Socket Connector”, page 69](#)

The laser control mode is set by `set_laser_mode`:

- The CO<sub>2</sub> Mode (laser mode 0; default after `load_program_file`) is designed for controlling a CO<sub>2</sub> laser.
- The YAG modes (laser mode 1, 2, 3, 5) are designed for controlling lasers from the Nd:YAG family (and related).
- Laser Mode 4 and Laser Mode 6 are designed for controlling free-running lasers.<sup>(1)</sup>

All laser control signals are TTL signals. By `set_laser_control`, it is set whether they are active-HIGH or active-LOW, see also [Chapter 4.6.1 “LASER Connector”, page 62](#). Their current setting can be queried by `get_startstop_info` (Bit #13).

By `get_startstop_info` (Bit #9), the current status of the laser control signals (enabled, suppressed) can be read out.

For the maximum current load values, see [Chapter 14 “Technical Specifications of the RTC6 PCIe Board”, page 823](#).

### 7.4.1 Enabling, Activating and Switching Laser Control Signals

All laser control signals are suppressed:

- After a hardware reset and
- After initialization with `load_program_file`.

Then, all laser signal output ports (LASERON, LASER1 and LASER2) are in the high impedance mode.

Before laser control signals can be outputted at all, their polarity must first be set by `set_laser_control`. This cancels the tristate state at the same time (= “global unblock”). By default, LASERON and LASER1/LASER2 are set to their respective “Off” levels.

The tristate state is only set again:

- after a hardware reset (restart)
- a call of `load_program_file`

Furthermore, real laser control signals must be enabled:

- either simultaneously by `set_laser_control(Bit #2 = 0)` or
- afterwards with the separate command `enable_laser`

These can be suppressed again by `disable_laser` or `pause_list`. In both cases, all laser control signals are set to their respective “Off” levels.

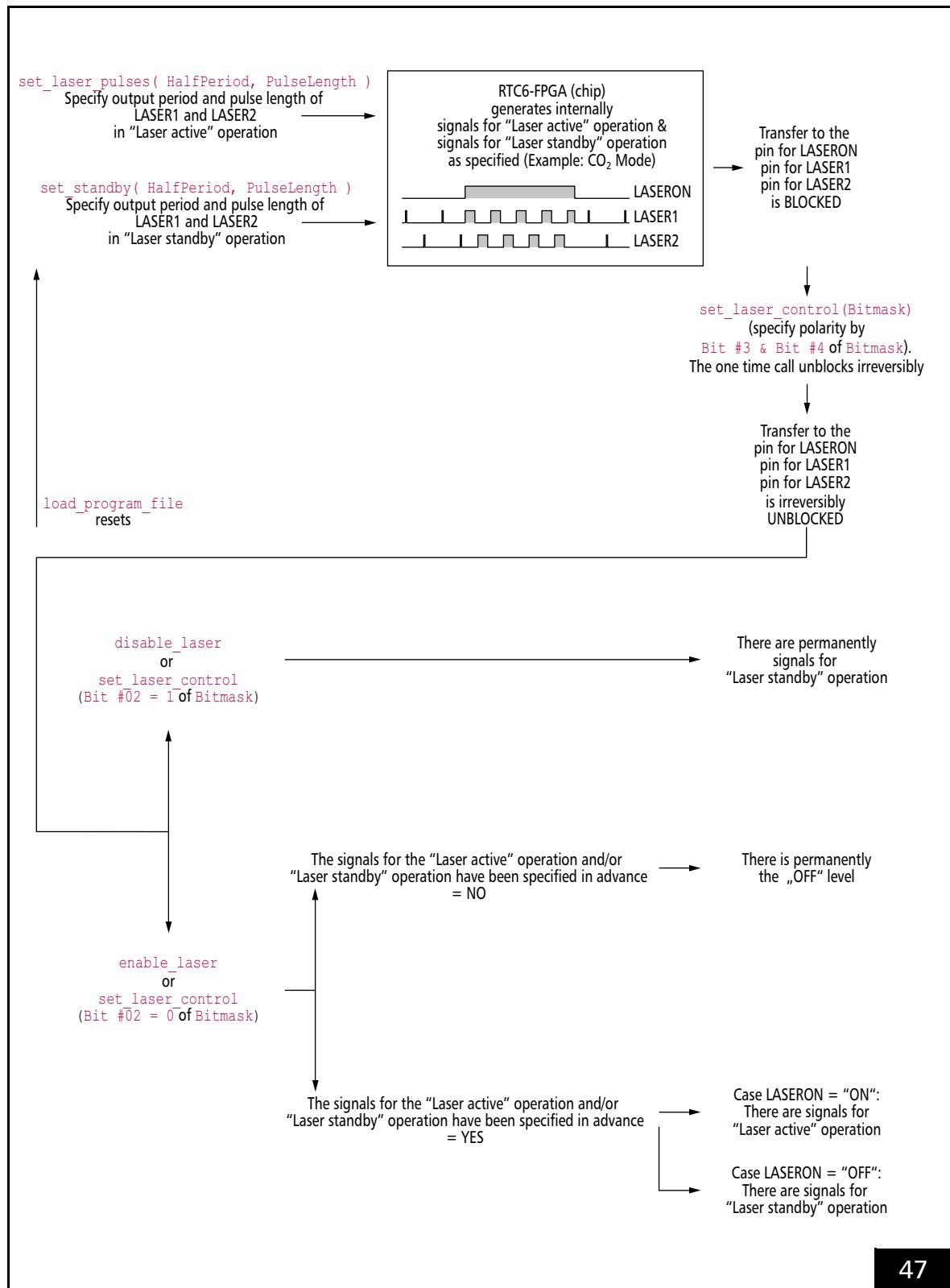
By `get_startstop_info` (Bit #9, Bit #10, Bit #13, Bit #14) it can be read out:

- The “global unblock” by `set_laser_control`
- The polarities of LASERON and LASER1/LASER2
- The enabling for “laser active operation” by `enable_laser`

General notes on the laser control signals can be found in [Section “Signals for “Laser Active” Operation”, page 175](#) and [Section “Signals for “Laser Standby” Operation”, page 175](#). Mode-specific details depend on the set laser mode, see [Chapter 7.4.3 “CO<sub>2</sub> Mode”, page 177](#) to [Chapter 7.4.8 “Pulse Picking Laser Mode”, page 185](#).

The assignment of the laser control signals to the respective output pins at the LASER connector can be configured, see [Chapter 7.4.2 “Configuring the LASER Connector”, page 176](#).

(1) For synchronization to externally controlled free-running lasers, see [Chapter 7.4.10 “Synchronization of the RTC6 Clock Cycle and an External Clock Signal”, page 196](#).



RTC6 board hardware and laser control-related RTC6 commands.

## Signals for “Laser Active” Operation

By `set_laser_pulses`, `set_laser_pulses_ctrl` or `set_laser_timing`, the signals for “laser active” operation can be

- deactivated: `HalfPeriod ≠ 0` and `PulseLength ≠ 0`
- activated: `HalfPeriod = 0` and/or `PulseLength = 0`

Even if the signals for “laser active” operation have been enabled and activated, they are *only outputted at the output ports, if they are switched on by further commands.*

They are automatically switched on, when:

- **Mark commands** are called, see [Chapter 7.1.1 “Marking with Vector and Arc Commands”, page 125](#)

They are automatically switched off, when:

- a **Mark command** is followed by a normal non-mark command
- a list is terminated by `set_end_of_list` or `stop_execution`
- a list is temporarily suspended by `set_wait`, `pause_list` or `stop_list`

In the latter case, the signals for “laser active” operation are switched on again if the list is continued by `release_wait` or `restart_list`.

They can also be switched on and off within a list with `laser_signal_on_list` and `laser_signal_off_list` or outside a list with `laser_signal_on` and `laser_signal_off` for an unlimited time.

## Pulse Completion

Whether a modulation pulse (Q-Switch pulse) started with the LASERON signal switched on is still completely executed or cut off at LASER1 if it has not yet been fully processed when the LASERON signal is switched off, can be set by `set_laser_control(Bit #0)`.

## Signals for “Laser Standby” Operation

By `set_standby` or `set_standby_list`, the signals for “laser standby” operation can be

- deactivated: `HalfPeriod ≠ 0` and `PulseLength ≠ 0`
- activated: `HalfPeriod = 0` and/or `PulseLength = 0`

The signals for the “Laser standby” operation are continuously outputted at the output ports after their activation (without further commands), if no signals for the “Laser active” operation are switched on.

If the signals for the “Laser active” operation are deactivated (for example, by `PulseLength = 0`), there is no changeover. Then the signals for the “Laser standby” operation are continuously outputted even during the execution of **Mark commands**.

If the signals for the “Laser standby” operation are deactivated, all output ports are set to the “Off” level in the “Laser standby” mode.

If activated signals for the “Laser standby” operation are to be deactivated when stopping a list with `pause_list` (here, only the signals for the “Laser active” operation are automatically deactivated), users must explicitly initiate this by calling `set_standby(PulseLength = 0)`.

The current “Laser standby” parameters that may have been changed within a list can be read out by the control command `get_standby`.

## Pulse Completion

With the signals for the “Laser standby” operation, pulse completion, see [“Pulse Completion”, page 175](#), is not supported.



## Automatic Suppression of Laser Control Signals

### Case: Scan System Status Errors

By `set_laser_control` (Bit #16...Bit #27), it can set that the laser control signals are to be automatically suppressed when the corresponding scan-system status signal (PowerOK, TempOK, PosAck of axis X/Y of head A/B) indicates an error (that is, is 0; "NOK").

As soon as at least one of the specified status signals is 0, then:

- Output of the laser control signals are automatically interrupted. They are only continued, if all selected status signals are simultaneously 1 (laser control signals disabled by `set_laser_control`, `disable_laser` or `pause_list` remain disabled regardless of the status signals' current value)
- Internal error bits are (cumulatively) set (which can be read-out by `get_marking_info`)
- If accordingly set by `set_laser_control(Bit #28 = 1)`, a `stop_execution` is automatically executed (the list stops, laser control signals get permanently switched off)
- If accordingly set by `set_laser_control(Bit #29 = 1)` in addition, the `stop_execution` is forwarded as /Master-STOP (see figure 63) to all Master/Slave connected RTC6 boards. Whether the signal is effective there can be set individually for each RTC6 board with `master_slave_config`. This ensures that the laser is automatically switched off even if the error occurs on an RTC6 board that is not intended for controlling the laser.

### Case: Galvanometer Scanner Position Exceedances

`range_checking` can be used to define that the laser control signals are to be automatically suppressed as soon as a galvanometer scanner exceeds a predefined range limit.

They are automatically switched on again as soon as the next `Mark command` starts within the permitted range. This means that an interrupted `Polyline` remains suppressed for the rest of the `Polyline`.

## 7.4.2 Configuring the LASER Connector

LASER connector pin (01), (02) and (09) can be configured by `config_laser_signals` and `config_laser_signals_list`, see figure 13.

If you employ a variety of lasers or laser operational modes, then these commands might eliminate the need to configure at the hardware level (that is, using various cables and switches).

Whereas the default setting (for normal markings) outputs the LaserOn signal as a laser start signal on the LASERON channel, you could, for example, configure the LASER2 signal as a gate signal outputted on the LASERON channel in pixel output mode with pulse picking.

For other operational modes, you could also configure the FirstPulseKiller signal as the laser start signal on the LASERON channel. This way, LASER1 signals can be outputted even before a delayed switch-on of the laser (which is not possible in the default setting).

The following description of the various laser modes applies to the default setting for laser signal output.

### 7.4.3 CO<sub>2</sub> Mode

`set_laser_mode(0)` sets the CO<sub>2</sub> Mode ("Laser Mode 0").

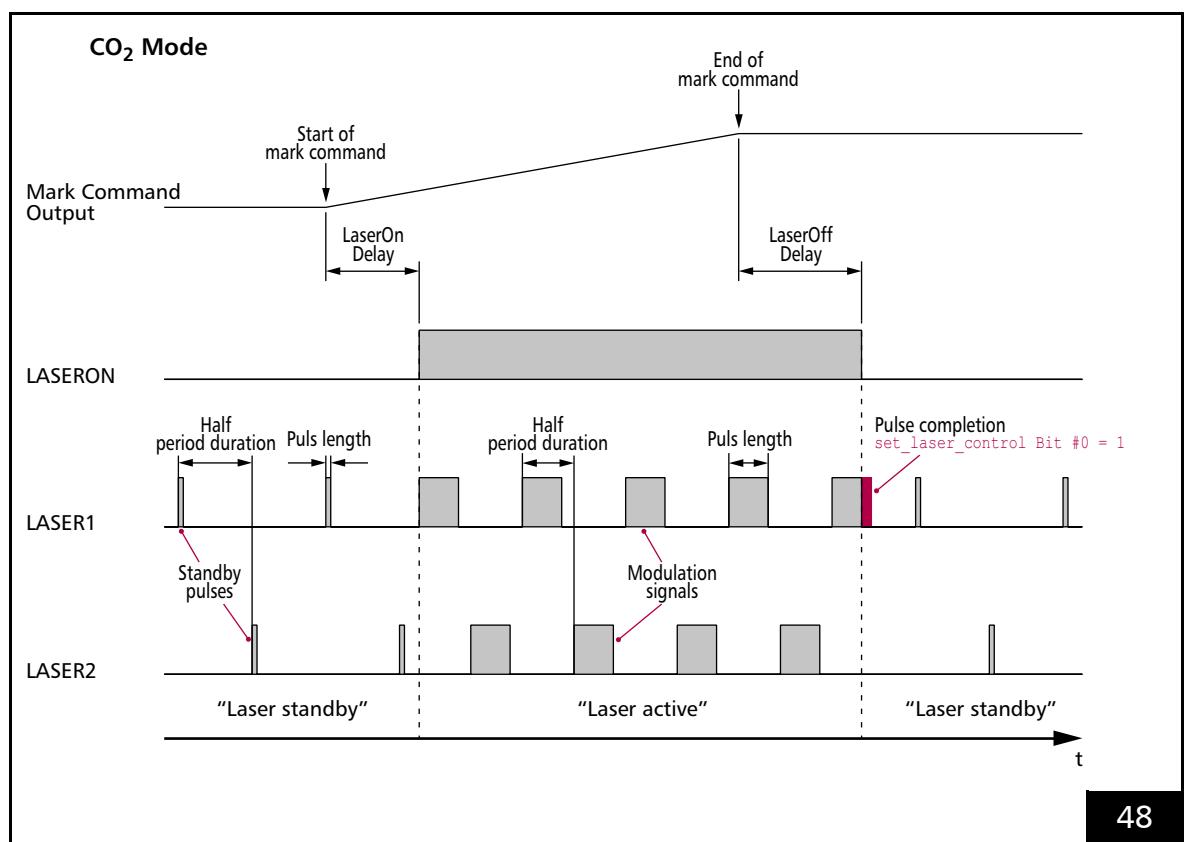
The timing diagram, see [figure 48](#), shows the corresponding signals using the example of an isolated mark command.

For "laser active" operation:

- The LASERON signal is switched on
- 2 alternating modulation signals are outputted at the LASER1 and LASER2 output port. Their pulse length and period duration can be defined by `set_laser_pulses`, `set_laser_pulses_ctrl` or `set_laser_timing`.

For "laser standby" operation:

- The LASERON signal is switched off
- Alternating standby pulses are outputted at the LASER1 and LASER2 output port. Their pulse length and period duration can be defined by `set_standby` or `set_standby_list`.



Timing diagram of the signals in CO<sub>2</sub> Mode (with active-HIGH laser control signals).  
Example: isolated mark command.



## Notes

- LASER1 signals and LASER2 signals are
  - activated by: HalfPeriod ≠ 0 and  
PulseLength ≠ 0
  - deactivated by: HalfPeriod = 0 and/or  
PulseLength = 0 (default setting after  
[load\\_program\\_file](#))
- The LASER2 signal is phase-shifted by half a signal period in relation to the LASER1 signal. It can be used for the control of a second laser tube. By [set\\_laser\\_control](#) (Bit #1 = 1), both signals can be exchanged with each other. To control laser power, the pulse length of the LASER1 and LASER2 signals can be varied. Both signals share the same pulse lengths and periods.
- For the LASER1 and LASER2 signals, *half* of the output period must be specified.
- [set\\_laser\\_pulses](#) and [set\\_laser\\_timing](#) are delayed short list commands. They can also be used to change the laser parameters between two [Mark commands](#).  
The time base for the signals is always:
  - 1/64 µs in [RTC6 Standard Mode](#),  
see also [Section "RTC6 Standard Mode"](#),  
[page 157](#)
  - 1/8 µs in [RTC4 Compatibility Mode](#),  
see also [Section "RTC4 Compatibility Mode"](#),  
[page 157](#)
- “[Pulse Completion](#)”, page 175 affects also LASER2 signals.

#### 7.4.4 YAG Modes 1, 2, 3, 5

`set_laser_mode( [1, 2, 3 or 5] )` sets one of the YAG modes.

The timing diagram, see [figure 49](#), shows the corresponding signals using the example of an isolated mark command.

In each YAG mode, for "laser active" operation:

- The LASERON signal is switched on
- A *Q-Switch signal* is outputted at the LASER1 output port, see "[Q-Switch Signal](#)", page 179.
- A adjustable *FirstPulseKiller signal* is outputted at the LASER2 output port, see "[FirstPulseKiller Signal](#)", page 179.

In each YAG mode, for "laser standby" operation:

- The LASERON signal is switched off
- Standby pulses are outputted at the LASER1 output port. Pulse length and period duration of the standby pulses can be set by `set_standby` or `set_standby_list`.

##### Notes

- LASER1 signals are
  - activated by: `HalfPeriod ≠ 0` and `PulseLength ≠ 0`
  - deactivated by: `HalfPeriod = 0` and/or `PulseLength = 0` (default setting after [load\\_program\\_file](#))

##### Q-Switch Signal

The Q-Switch signal serves to control the quality switch of the laser. The Q-Switch period duration and pulse length are set by `set_laser_pulses`, `set_laser_pulses_ctrl` or `set_laser_timing`.

##### Notes

- See also "[Pulse Completion](#)", page 175 of signals for "Laser Active" operation.
- See also "[Pulse Completion](#)", page 175 of signals for "Laser standby" operation.

##### FirstPulseKiller Signal

The FirstPulseKiller signal serves to signal the first laser pulses of a pulse sequence, if they do not correspond to the usual continuous power.

This signal is only provided. The laser itself must respond appropriately. The FirstPulseKiller signal is outputted automatically together with the LASERON signal.

The length of the FirstPulseKiller signal is set with `set_firstpulse_killer` or `set_firstpulse_killer_list`.

##### Differences Between the YAG Modes

YAG Mode 1, YAG Mode 2, YAG Mode 3 and YAG Mode 5 only differ in the relative start time of the first Q-Switch pulse with reference to the FirstPulseKiller signal, see also the timing diagrams in [figure 49](#).

After the Q-Switch delay has expired, the first Q-Switch pulse starts. The Q-Switch delay value can be specified by `set_qswitch_delay` or `set_qswitch_delay_list`; alternatively by selecting the YAG mode:

- YAG Mode 1: 0
- YAG Mode 2: length of the FirstPulseKiller signal
- YAG Mode 3: 10 µs
- YAG Mode 5: value from `set_qswitch_delay` or `set_qswitch_delay_list`

By `set_laser_mode`, the Q-Switch delay is merely *preset*. Therefore, in YAG Mode 1, YAG Mode 2, YAG Mode 3, too, the Q-Switch delay can be subsequently changed by `set_qswitch_delay` or `set_qswitch_delay_list`.

In YAG Mode 2, the Q-Switch delay is also adjusted accordingly with each `set_firstpulse_killer` or `set_firstpulse_killer_list`.



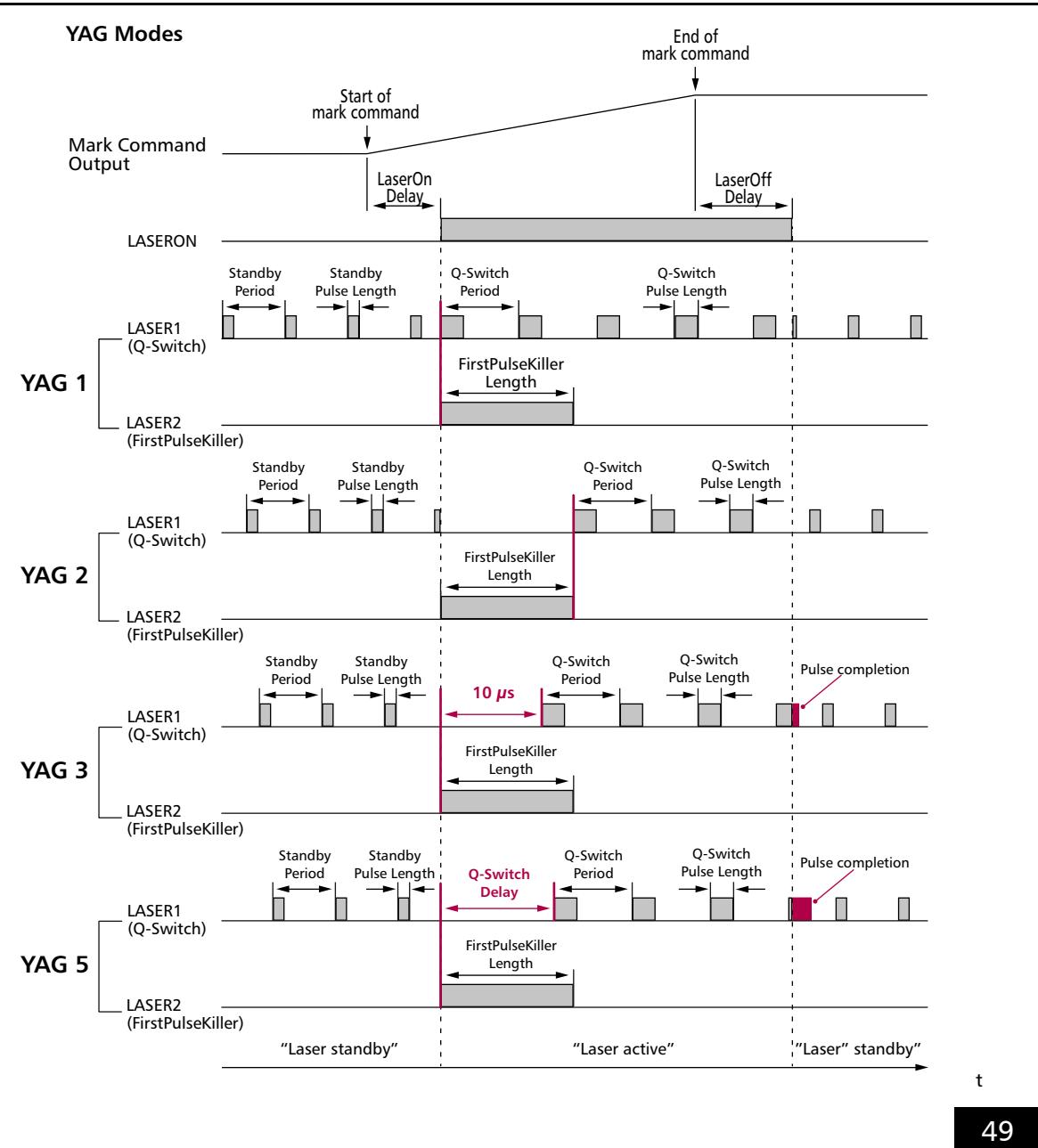
### Lamp Current (Laser Power)

To control the lamp current of a YAG laser, the 12-bit analog output ports, ANALOG OUT1 or ANALOG OUT2 can be used. They are available at the 15-pin D-SUB LASER connector, see [figure 13](#). ANALOG OUT2 is also available at the MARKING ON THE FLY socket connector, see [figure 21](#).

To define the analog output signal, the control command `write_da_x` and the delayed short list command `write_da_x_list` are provided.

Alternatively, the lamp current can be controlled digitally via the 8-bit digital output port (at the EXTENSION 2 socket connector, see [figure 20](#)), see also [Section "8-Bit Digital Output Port", page 70](#). The commands for setting the 8-bit output port are `write_8bit_port` and `write_8bit_port_list`.

When the lamp current is changed, list execution can be halted by `long_delay` until a constant laser power has been achieved.



Timing diagram of the signals in the YAG modes (with active-HIGH laser control signals). Standby signals are not synchronized with the "laser-active" signals and can have arbitrary phase alignments.  
Example: isolated mark command.

### 7.4.5 Laser Mode 4

`set_laser_mode(4)` sets the Laser Mode 4.

The timing diagram, see [figure 50](#), shows the corresponding signals using the example of an isolated mark command.

At LASER1 output port, for "laser active" operation as well as for "laser standby" operation standby pulses are outputted continuously.

Pulse length and period duration of the standby pulses can be set by `set_standby` or `set_standby_list`.

For "laser active" operation:

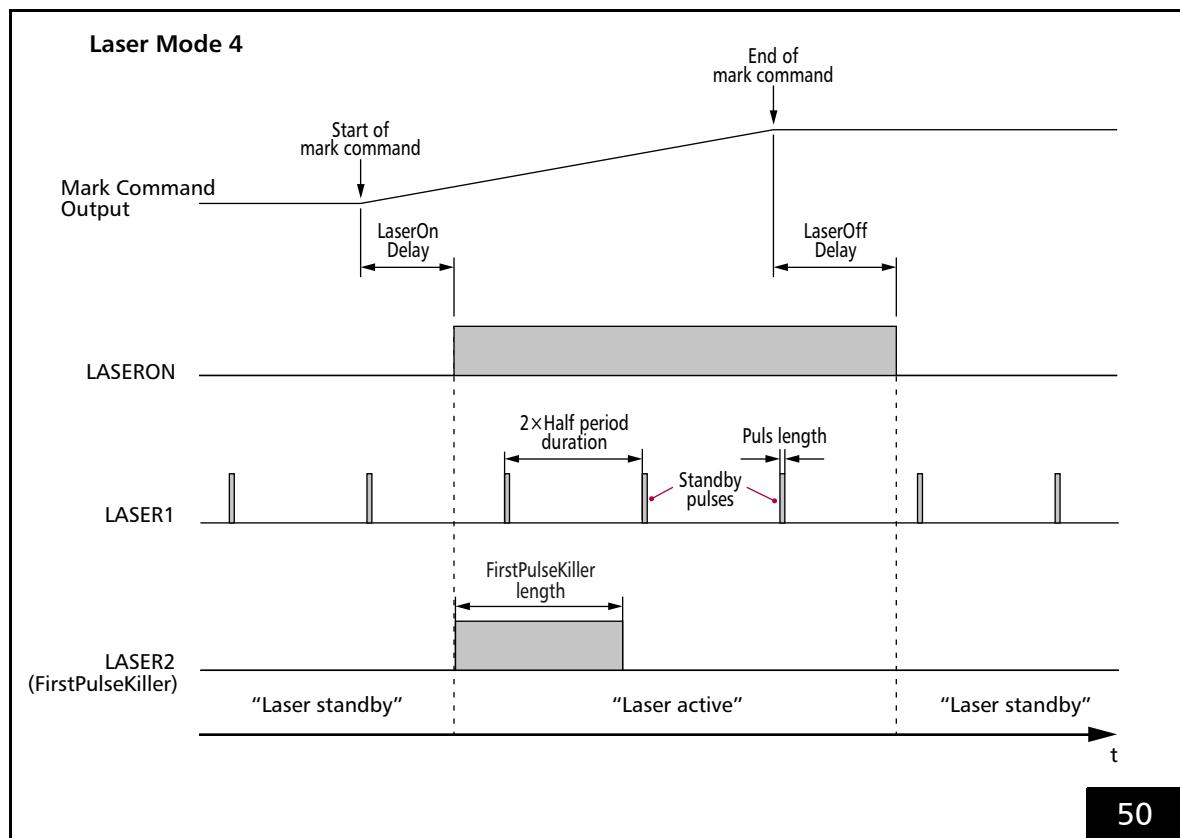
- The LASERON signal is switched on
- A programmable FirstPulseKiller signal is outputted at the LASER2 output

For "laser standby" operation:

- the LASERON signal is switched off
- and the LASER2 signal is switched off

#### Notes

- LASER1 signals are
  - activated by: `HalfPeriod ≠ 0` and `PulseLength ≠ 0`
  - deactivated by: `HalfPeriod = 0` and/or `PulseLength = 0` (default setting after `load_program_file`)
- The FirstPulseKiller signal is started together with the LASERON signal automatically. The length of the FirstPulseKiller signal is set by `set_firstpulse_killer` or `set_firstpulse_killer_list`.
- Laser Mode 4 is used for some fiber lasers.



Timing diagram of the signals in Laser Mode 4 (with active-HIGH laser control signals).  
Example: isolated mark command.

### 7.4.6 Laser Mode 6

`set_laser_mode(6)` sets the Laser Mode 6.

Laser Mode 6 is like Laser Mode 4 and is provided for those free-running lasers whose gate signals (LASERON) must *not* be changed during the duration of a pulse.

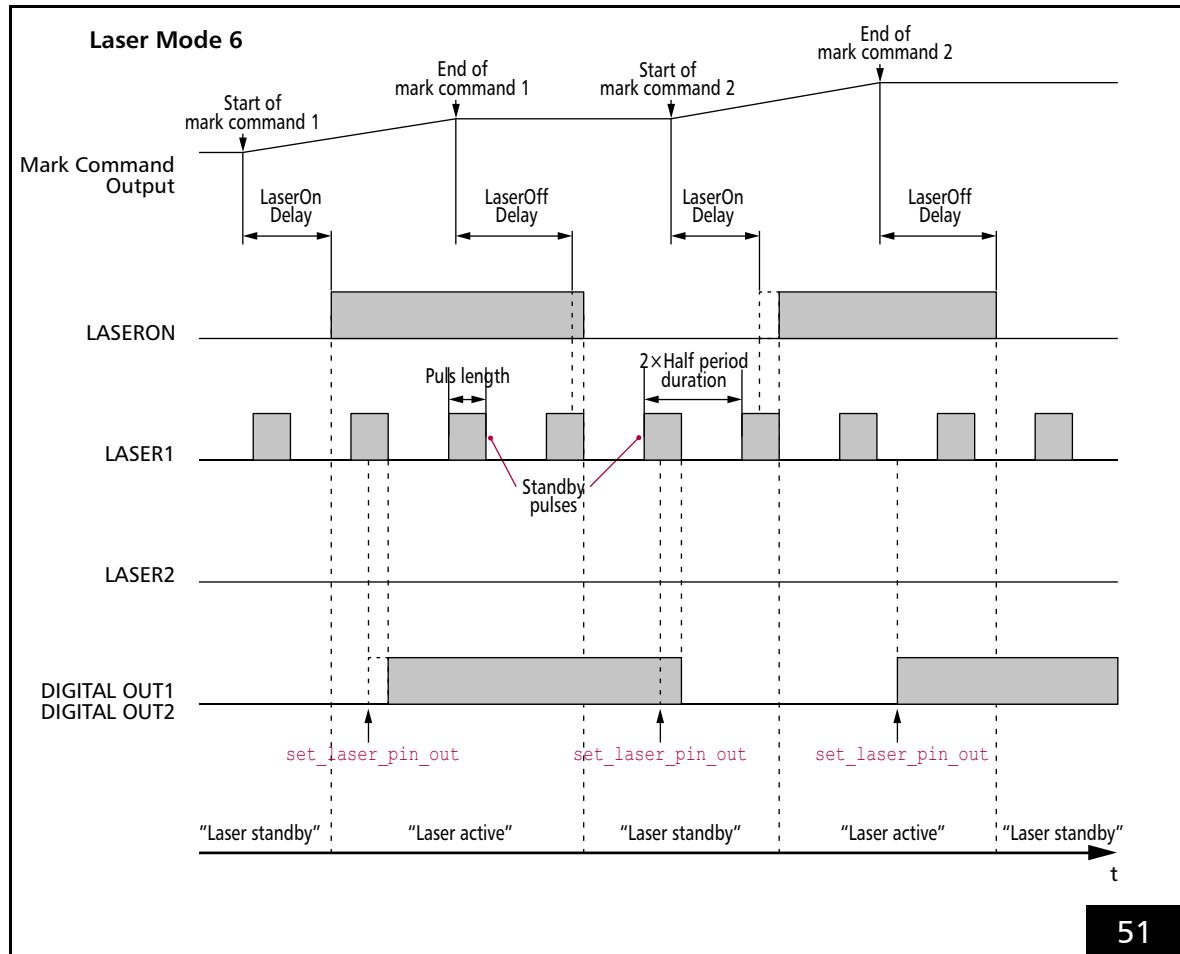
The timing diagram, see [figure 51](#), shows the corresponding signals using the example of an isolated mark command.

Laser Mode 6 signals and Laser Mode 4 signals are the same (see also Notes there, [page 182](#)) with the following exception:

- As long as a standby pulse is active, switching of the LASERON signal is delayed accordingly. LASERON switches 5/64  $\mu$ s after the end of the standby pulse.

#### Notes

- No LASER2 signal is outputted.
- The delay of the switching time when a standby pulse is still active is also valid for switching the output values at the 2-bit digital output port (DIGITAL OUT1 and DIGITAL OUT2) by `set_laser_pin_out` or `set_laser_pin_out_list`, see [figure 51](#).



Timing diagram of the signals in Laser Mode 6 (with active-HIGH laser control signals).  
Example: isolated mark commands.



#### **7.4.7 Softstart Mode (not yet implemented)**

*Not implemented yet.*

#### 7.4.8 Pulse Picking Laser Mode

By `set_pulse_picking` or `set_pulse_picking_list`, the pulse picking laser mode can be selected. The laser control timing diagram in figure 52 shows the corresponding signals.

For "laser active" operation:

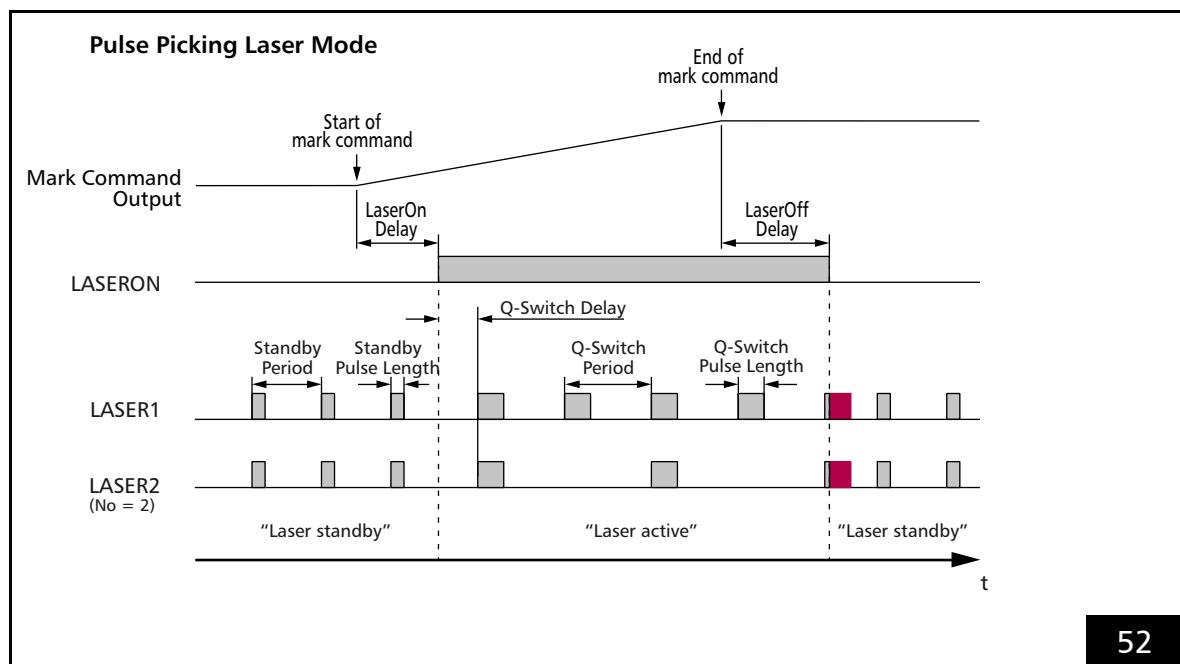
- The LASERON signal is switched on
- A modulation signal is provided at the LASER1 output with pulse length and period duration that can be defined by `set_laser_pulses`, `set_laser_pulses_ctrl` or `set_laser_timing`
- At the LASER2 output every  $No^{th}$  pulse of the signals is outputted at the LASER1 output (in phase).  $No$  is set with `set_pulse_picking` or `set_pulse_picking_list`

For "laser standby" operation:

- The LASERON signal is switched off
- Standby pulses are provided in phase at the LASER1 and LASER2 outputs with pulse lengths and periods that can be defined by `set_standby` or `set_standby_list`. Standby pulses cannot be "pulse-picked".

#### Notes

- With `set_laser_control`( Bit #7 = 1) the Pulse Picking signal at LASER2 can be set to a constant length independent of the LASER1 signal, which is set with `set_pulse_picking_length`.
- `set_pulse_picking` and `set_pulse_picking_list` overwrite a laser mode previously set with `set_laser_mode`. Vice versa, `set_laser_mode` switches off the Pulse Picking laser mode.
- For the LASER1 signals, half the period duration must be specified.
- A Q-Switch delay is effective, see also [Section "Differences Between the YAG Modes", page 179](#).
- A FirstPulseKiller signal is not outputted.
- The setting sequence of the LASER1 signals and the pulse picking laser mode is irrelevant.



Timing diagram of the signals in pulse picking laser mode (with active-HIGH laser control signals and  $No = 2$ ). Example: isolated mark command.

### 7.4.9 "Automatic Laser Control"

By `set_auto_laser_control`, automatic readjustment of "laser active" laser control signals – even dynamically during execution of **Mark commands** can be achieved.

The `Ctrl` parameter determines the output port and the `Value` parameter the output value for 100% power at the target (set) mark speed at the image field center (for the other parameters, see command description).

This can be used to compensate for variations in laser energy input caused by a changing power density, spot size or processing speed.

For "Automatic Laser Control", a position-, speed- or vector-controlled correction of the laser control signals can be activated and combined.

- *Position-dependent* laser control, see [Section "Position-Dependent Laser Control", page 188](#), allows compensation of radial laser energy input variations during execution of **Mark commands**. Such variations may be caused, for example, by a objective edge diminution or different incidence angles of the laser beam.
- *Speed-dependent* laser control, see [Section "Speed-Dependent Laser Control", page 190](#), allows compensation of laser energy input variations during execution of **Mark commands** that resulting from an uneven galvanometer scanner movement (acceleration phase and deceleration phase, time-dependent **Mark commands**). In addition, the galvanometer scanner speed can also be transformed back into a mark speed depending on the position.
- *Encoder speed-dependent* laser control, see [Section "Encoder-Speed-Dependent Laser Control", page 192](#), allows the laser energy input to be controlled based on the currently present encoder speed.

- Speed-dependent laser control and encoder speed-dependent laser control can also be combined with each another, if galvanometer scanner and workpiece move simultaneously.

- *Vector-defined* laser control, see [Section "Vector-Defined Laser Control", page 195](#), allows linear readjustment of a signal parameter along parameterized mark vectors or jump vectors (see [Section "\[\\*\]Para\[\\*\] Commands", page 128](#)).

This signal parameter can be combined with the laser power control if the control parameter `Ctrl` matches `Ctrl` from `set_auto_laser_control` (the current parameter `Para` dynamically replaces the 100% value from `set_auto_laser_control`) or as an independent control (for example, as defocus).

Selectively, "Automatic Laser Control" adjusts one of the following signal parameters:

- 12-bit output value at the ANALOG OUT1 or ANALOG OUT2 output port of the LASER connector, see also [Section "12-Bit Analog Output Port 1 and 2", page 63](#)
- Output value at the 16-bit digital output port of the EXTENSION 1 socket connector, see also [Section "16-Bit Digital Input and 16-Bit Digital Output", page 67](#)
- Output value at the 8-bit digital output port of the EXTENSION 2 socket connector, see also [Section "8-Bit Digital Output Port", page 70](#)
- Pulse length (`PulseLength`) or output period (`HalfPeriod`) of the laser control signals LASER1 and LASER2

The automatically readjusted value can be recorded by `set_trigger/set_trigger4` (Signal n = 24).



## Notes

- “Automatic Laser Control” does *not* compensate for an explicit change in mark speed between two **Mark commands** caused by a **set\_mark\_speed** call.
- “Automatic Laser Control” and **Pixel mode** should not affect the same output port at the same time.
- “Automatic Laser Control” cannot be combined with variable laser control via the **McBSP interface** (see **set\_multi\_mcbsp\_in**).

## General Notes

- Each individual contribution as well as the total correction cannot exceed a factor of 4.0 (clipping<sup>(1)</sup>).
- If laser power and/or energy input into the to-be-processed material is not strictly proportional to the output values of the selected signal parameter, then **load\_auto\_laser\_control** can be used to load a nonlinearity curve that defines this (application-specific) relationship, see **Section “Loading and Determining the Nonlinearity Curve”, page 193**.
- In addition, the selected signal parameter can neither exceed the value range allowed with **set\_auto\_laser\_control** (parameter `MinValue` and `MaxValue`) nor the value range allowed for the respective output port. It is clipped correspondingly.
- For the laser control signal a corresponding default value is outputted (see **set\_port\_default** or **set\_laser\_off\_default**), if:
  - “Laser active” signals are switched off when “Automatic Laser Control” is active
  - “Automatic Laser Control” itself is deactivatedIf no default value has been explicitly defined, the permitted maximum value is outputted. As substitute for the control parameters `HalfPeriod` and `PulseLength`, the parameter `Value` (the 100% value) from **set\_auto\_laser\_control** is used.

- Once an “Automatic Laser Control” has been switched on with **set\_auto\_laser\_control**, then the following can be changed subsequently by **set\_auto\_laser\_params** or **set\_auto\_laser\_params\_list**:
  - the signal parameter
  - the 100% value
  - limit values assigned to itThe `Mode` parameter cannot be changed subsequently.

(1) < DLL 612: overflow.

## Position-Dependent Laser Control

To activate the position-dependent laser control for the output port specified by **set\_auto\_laser\_control**, a user-defined scaling function must be loaded by **load\_position\_control**, see [Section "Notes on Loading a Scaling Function", page 188](#).

This scaling function represents a scaling factor as a function of the distance to the center of the image field. After **load\_program\_file**, it is initialized for all distances with "factor 1.0". The "position-dependent" laser control is de facto deactivated by that. The table can be saved by **create\_dat\_file**.

When calculating the correction for "position-dependent laser control", the current Cartesian control values are used as a basis:

- including wobbel correction (#2 in [Chapter 7.3.6 "Output Values to the Scan System", page 170](#))
- including coordinate transformation in the virtual image field (#3)
- including Processing-on-the-fly correction (#4)
- excluding head-specific coordinate transformation (#5)
- excluding image field correction (#6)

### Notes on Loading a Scaling Function

- For the **Scale(Position)** scaling function, **load\_position\_control** loads a table from an ASCII text file.
- The ASCII text file can contain one or several tables.<sup>(1)</sup>
- Each table can contain up to 50 data points (**Position | Scale(Position)**).
- The **Scale(Position)** function is linearly interpolated from the data points.

For the scaling function tables, the following rules apply:

- Each table must begin with the line:  
[PositionCtrlTable<No>]  
<No> represents the table number.
- If the table contains multiple [PositionCtrlTable<No>] entries with the same <No>, then only the lines after the first entry are used. Only lines up to the next '[' character (that is not preceded by a semicolon) are used.
- Each data point (**Position | Scale(Position)**) is defined as follows:

Position<n> = <Value>  
Scale<n> = <Value>  
where <n> corresponds to the index ( $1 \leq n \leq 50$ ) of the data point. The values <Value> can be specified as (unsigned) floating point numbers.  
Decimal separator: period (.)

- If the table contains multiple data points with the same Index <n>, then the most recently read one is used.
- If the table contains multiple data points with the same position value **Position**, then the data point with the largest Index <n> is used. Equality is checked to within  $\pm 0.01$ .
- The position value is specified radially as the distance between the to-be-marked point and the coordinate midpoint ( $= (x^2 + y^2)^{1/2}$ ) as percent of half the image-field side length.  
Example:  $(X_{max}|0)$  corresponds to 100%,  $(X_{max}|Y_{max})$  corresponds to  $2^{1/2} \times 100\%$ .

(1) Even of another type, see [table 1, page 142](#).



- For <Value>, the following ranges apply:  
 $0.0 \leq Position \leq 150.0$  and  
 $0.0 \leq Scale(Position) \leq 4.0$ .
- Each instruction must be in a separate line.
- Spaces and tabs in a line (for example, between '=' and <Value>) are ignored.
- Empty lines are ignored.
- Data points with invalid values are ignored.
- The data point of a particular index <n> is ignored if the corresponding Position<n> and/or Scale<n> definition is missing.
- The semicolon ';' can be used for comments. All characters in a line following a semicolon are ignored.
- The instructions for data points in the table can be ordered as desired.
- Indices for data point pairs in the table can be selected as desired within the range [1...50] (the table is then automatically sorted by ascending position values).
- If the table contains no valid data point, the command **load\_position\_control** has no effect (return value 1 or 13).
- If there is no entry for  $Position = 0.0$ , then an entry with  $Scale = \text{Min}(Scale <i>)$  is inserted (the smallest allowed value defined in the table is used for lower positions values). Likewise for  $Position = 150.0$  with  $\text{Max}(Scale <i>)$ .
- If the selected text file only contains a single valid data point with  $Scale <n> = S$ , then (for the entire position range) the scaling function  $Scale(Position) = S$  is loaded. The correction has a multiplicative effect on the laser control signal. For  $S = 1.0$ , position-dependent correction is therefore switched off. Alternatively, this can also be achieved with **Name** = 0 in **load\_position\_control**.
- The table can be saved by **create\_dat\_file**.



## Speed-Dependent Laser Control

When activating the “speed-dependent laser control” for the output port specified by `set_auto_laser_control`, the `Mode` parameter is used to select which input parameters are to be used for calculating the correction.

As reference value for the 100% speed, always the set (target) mark speed (set by `set_mark_speed` or `set_mark_speed_ctrl`) applies (its change is never compensated by the “Automatic Laser Control”).

- `Mode = 1` is intended (for consistency reasons) especially for analog scan systems. The current microstep length per 10 µs is used as input. It may deviate from the target speed due to the 10 µs clock pulse rounding or with [\*]timed[\*] commands. Variations in the acceleration and deceleration phases cannot be compensated.
- `Mode = 2` is intended as basic mode for iDRIVE scan systems<sup>(1)</sup>. It can be combined with other special corrections (see below).
- Extensions to `Mode = 2` (any combination possible):
  - `+0`  
Basic mode for intelliSCAN systems
  - `+4`  
Combines the speeds from `Mode = 2` and `Mode = 5` to a total speed. The 100% reference speed from `Mode = 5` remains unconsidered. Instead, the encoder speeds are converted into galvanometer scanner bit speeds with the fly scaling factors and then vectorially added to the galvanometer scanner speed. A corresponding Processing-on-the-fly session must be active.

– `+16`

Basic mode for excelliSCAN scan heads

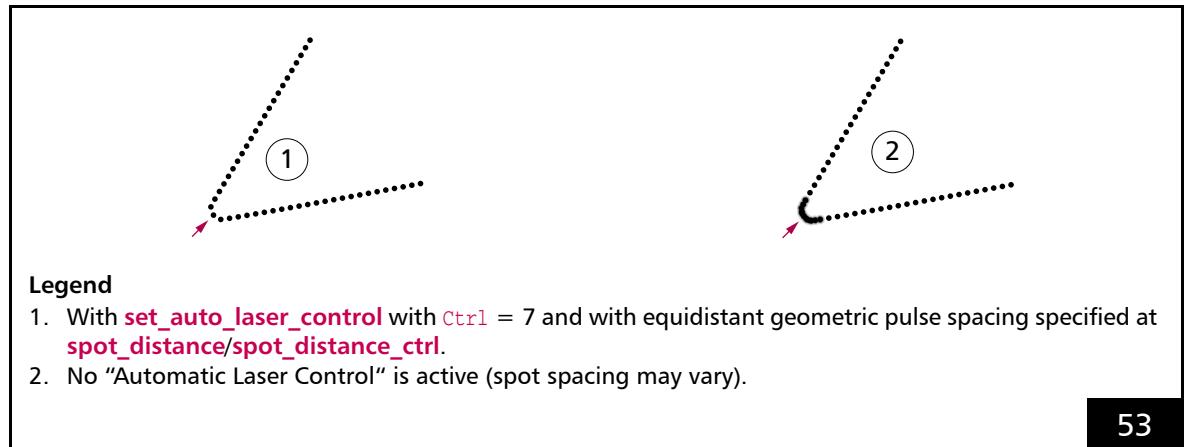
– `+32`

Position-dependent correction of the galvanometer scanner speed (in angular units) in a one for the image field (“inverse speed correction table”)

## Notes

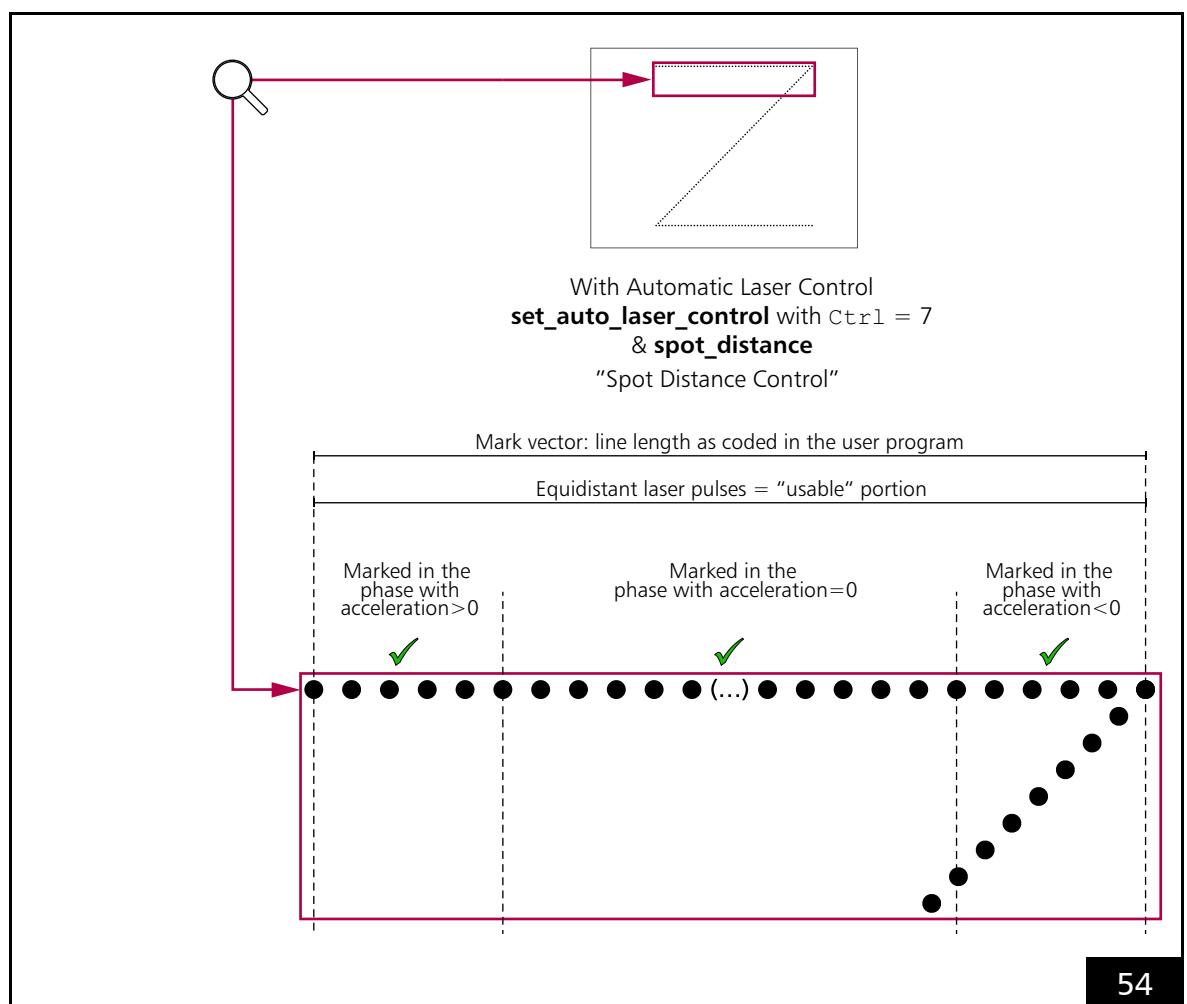
- Usually `set_auto_laser_control`(`Mode = 2`) requires a special intelliSCAN firmware, which returns an actual speed corrected by the signal runtimes between the RTC6 PCIe Board and the scan head. If you have any questions as to whether your intelliSCAN is equipped with it or is upgradeable, contact SCANLAB.
- Usually a combination of “speed-dependent laser control” and a negative LaserOn delay does not make sense.

(1) See glossary entry on page 879.



53

Example marking result: "sharp" corner.



54

For `set_auto_laser_control` with `Ctrl = 7` the marking result shows *largely* equidistant spots as long as the acceleration does not change too much within a 10  $\mu$ s cycle.



### "Spot Distance Control"

The "Spot Distance Control" functionality is only available for scan systems equipped with SCANahead servo control (for example, excelliSCAN series). With `set_auto_laser_control( Ctrl = 7 )` the output period (HalfPeriod) is controlled to a geometrically constant pulse distance, which is defined by `spot_distance` or `spot_distance_ctrl`. The parameters `Value`, `MinValue` and `MaxValue` have no meaning with the "Spot Distance Control" functionality.

The speed correction occurs with `Mode = 2 + 16 +` (optionally) `32`, (see above). "Spot Distance Control" cannot be combined with `Mode = 1, 2 or 5`.

The correction is more precise and dynamic than the comparable control with `Ctrl = 5` (HalfPeriod). For successful use, a laser with "pulse-on-demand" functionality should be used that outputs laser pulses immediately when triggered by the LASER1 signal. Frequency modulations are set with an accuracy of  $1/64 \mu\text{s}$  for the HalfPeriod.

A comparison of a typical marking result with "Automatic Laser Control" (`Ctrl = 7`) and without "Automatic Laser Control" is shown in [figure 53](#).

As an alternative to `set_auto_laser_control` with `Ctrl = 7`, the marking shown in [figure 54](#) could also be carried out with Sky Writing switched on. However, when using `set_auto_laser_control` with `Ctrl = 7` the process time is often shorter.

### Encoder-Speed-Dependent Laser Control

`set_auto_laser_control( Mode = 5 )` is intended for a pure encoder speed-dependent correction, if only the workpiece moves and the galvanometer scanners (ideally) are idle.

The 100% reference speed is defined by `set_encoder_speed_ctrl` or `set_encoder_speed`. Processing-on-the-fly should not be active at the same time.

For a combination of galvanometer scanner speeds and encoder speeds in a [Processing-on-the-fly session](#), see `Mode = 6 = 2 + 4`.



## Loading and Determining the Nonlinearity Curve

- For the *Scale(Percent)* nonlinearity curve, **load\_auto\_laser\_control** loads a table from an ASCII text file.
- The ASCII text file can contain one or several tables.<sup>(1)</sup>
- Each table can contain up to 50 data points (*Percent* | *Scale(Percent)*).
- The *Scale(Percent)* function is linearly interpolated from the data points.

For the tables, the following rules apply:

- Each table must begin with the line:  
[AutoLaserCtrlTable<No>]  
<No> represents the table number.
- If the table contains several [AutoLaserCtrlTable<No>] entries with the same <No>, then only the lines after the first entry are used. Only lines up to the next '[' character (that is not preceded by a semicolon) are used.
- Each data point (*Percent* | *Scale(Percent)*) is defined as follows:

Percent<n> = <Value>  
Scale<n> = <Value>  
where <n> corresponds to the index ( $1 \leq <n> \leq 50$ ) of the data point. The values <Value> can be specified as (unsigned) floating point numbers.  
Decimal separator: period (.).

- If the table contains multiple data points with the same Index <n>, then the most recently read one is used.
- If the table contains multiple data points with the same percent value *Percent*, then the data point with the largest Index <n> is used. Equality is checked to within  $\pm 0.01^\circ$ .

(1) Even of another type, see table 1, page 142.

- The percent value is relative to the 100% value from **set\_auto\_laser\_control** (Parameter **Value**) or dynamically from a ““vector-controlled laser control””, see **Section “Vector-Defined Laser Control”**, page 195.

In the following example, a nonlinearity factor of 1.2 is set for a 1.5x multiple of the target value:

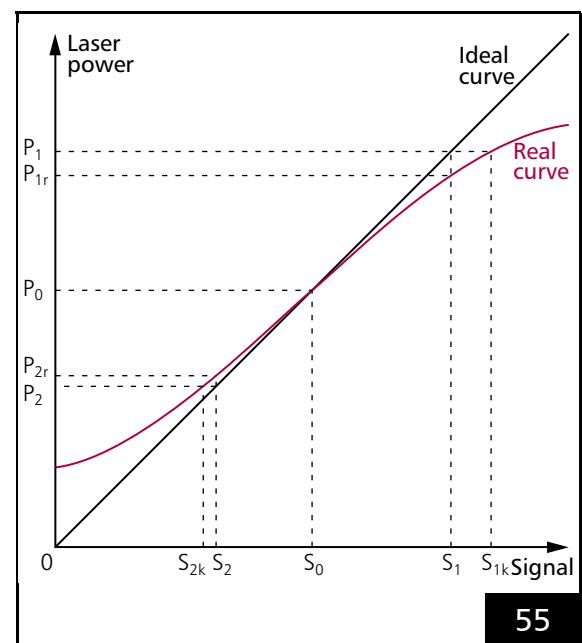
```
Percent<n> = 150
Scale<n> = 1.2
```

- For **<Value>**, the following ranges apply  
 $0.0 \leq \text{Percent} \leq 400.0$  and  
 $0.0 \leq \text{Scale}(\text{Percent}) \leq 4.0$ .
- Each instruction must be in a separate line.
- Spaces and tabs in a line (for example, between '=' and **<Value>**) are ignored.
- Empty lines are ignored.
- Data points with invalid values are ignored.
- The data point of a particular index **<n>** is ignored if the corresponding **Percent<n>** and/or **Scale<n>** definition is missing.
- The semicolon ';' can be used for comments. All characters in a line following a semicolon are ignored.
- The instructions for data points in the table can be ordered as desired.
- Indices for data point pairs in the table can be selected as desired within the range [1...50] (the table is then automatically sorted by ascending percent values).
- If the table contains no valid data point, then **load\_auto\_laser\_control** has no effect (return value 1 or 13).
- If there is no entry for **Percent = 0.0**, then an entry with **Scale = Min(Scale<i>)** is inserted (the smallest allowed value defined in the table is used for lower percent values). Likewise for **Percent = 400.0** with **Max(Scale<i>)**.

After **load\_program\_file** this function is initialized for all percentage values with “Factor 1.0”, the nonlinear laser control is “deactivated”. Alternatively, this can also be achieved with **Name = 0** in **load\_auto\_laser\_control**.

The table can be saved by **create\_dat\_file**.

The example diagram in **figure 55** illustrates how the nonlinearity curve can be determined.



55

Laser power progression – example of determining a nonlinearity curve.

The straight line in the diagram describes an ideal relationship between laser power and the laser control signal parameter (here, the term laser power also represents the pulse frequency =  $0.5/\text{LaserHalfPeriod}$ ), the curved line simulates a realistic relationship.

$S_0$  is the signal parameter value defined as the target value and  $P_0$  is the associated laser power. At point  $(S_0 | P_0)$  (this corresponds to the data point  $\text{Percent}_0 = 100, \text{Scale}_0 = 1.0$ ) the two curves are normalized to each other. The combination of a nonlinearity curve with a ““vector-controlled laser control”” is therefore generally not recommended.

An increase (decrease) of the signal parameter to  $S_1$  ( $S_2$ ) results in an ideal laser power  $P_1$  ( $P_2$ ) and a real laser power  $P_{1r}$  ( $P_{2r}$ ). For the actually desired laser power  $P_1$  ( $P_2$ ), a corrective signal parameter value  $S_{1k}$  ( $S_{2k}$ ) is needed. The following two value pairs are then to be entered as data points for the nonlinearity curve:

$$\begin{aligned}\text{Percent1} &= S_1/S_0 \times 100 = P_1/P_0 \times 100 \\ \text{Scale1} &= S_{1k}/S_1\end{aligned}$$

$$\begin{aligned}\text{Percent2} &= S_2/S_0 \times 100 = P_2/P_0 \times 100 \\ \text{Scale2} &= S_{2k}/S_2\end{aligned}$$



## Vector-Defined Laser Control

The “vector-controlled laser control” allows a signal parameter to be varied linearly along a mark vector or jump vector.

To initialize the ““vector-controlled laser control””, **set\_vector\_control** must be used to specify which signal parameter is to be varied with which initial value (parameters **Ctrl** and **Value**).

Then the signal parameter is varied linearly along a parameterized mark or jump vector. The end value is automatically the start value for a subsequent **[\*]para[\*]** command.

### Notes

- List commands for explicitly changing the signal parameter output value are temporarily effective or not at all.
- **[\*]para[\*]** commands always use the end value of the previous **[\*]para[\*]** command as their start value.  
Control commands that write to the same output port should be avoided while processing a list of **[\*]para[\*]** commands.
- If the same output port is selected via **set\_auto\_laser\_control** for **Ctrl**, the control parameter from the **[\*]para[\*]** commands acts as 100% value for the laser control.  
Special care should be taken with **set\_vector\_control** (**Ctrl** = 7) (Defocus): The setting of the signal value is always done immediately. This can lead to **Hard jumps** on the varioSCAN.
- During execution of **[\*]para[\*]** commands, the Sky Writing mode, see [Chapter 7.2.4 “Sky Writing”, page 149](#), is *not* taken into account (but also not deactivated).

### 7.4.10 Synchronization of the RTC6 Clock Cycle and an External Clock Signal

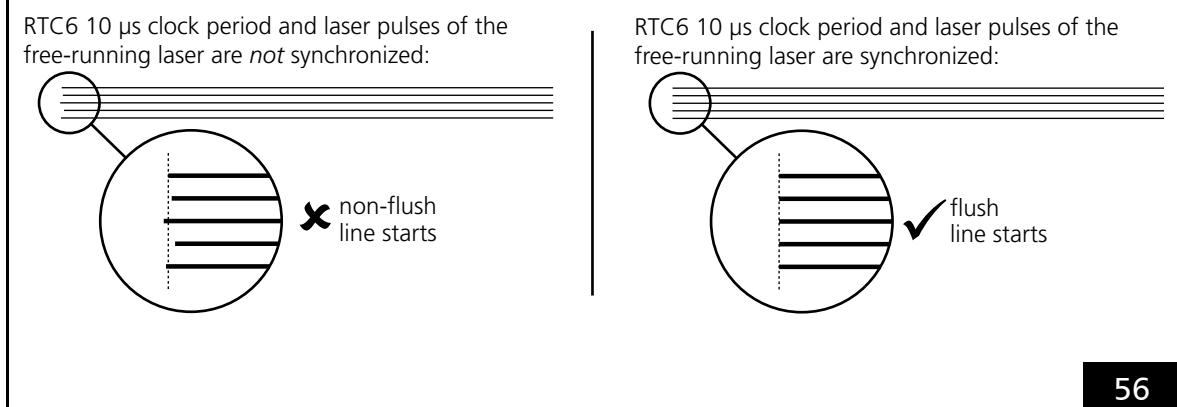
The laser pulse signals of a free-running laser and the laser control of the RTC6 PCIe Board can be synchronized. Non-flush line starts are thereby avoided, see figure 56.

Here, the RTC6 PCIe Board accepts the external clock signal as the master clock. It must be supplied at the digital input port DIGITAL IN1 of the LASER connector.

The synchronization is switched on and off by **Bit #6** of **set\_laser\_control**. By **Bit #5**, it can be set whether to use the rising or falling edge of the external clock for synchronization. This automatically synchronizes the RTC6-internal laser signals to the external clock signals.

#### Notes

- The frequency of the master clock must meet the following requirements:  
 $f = k \times 100 \text{ kHz}$  ( $k \leq 64$ ) and  
 $f = 64 \text{ MHz} / n$  ( $k$  and  $n$  integer).  
Therefore, only the following frequencies can be used: 100 kHz, 200 kHz, 400 kHz, 500 kHz, 800 kHz, 1 MHz, 1.6 MHz, 2 MHz, 3.2 MHz, 4 MHz, 6.4 MHz. The allowed external frequency deviation from an integer multiple of 100 kHz is  $\pm 15.625 \text{ ns}$  per  $10 \mu\text{s}$ .
- The supplied clock signal must be a TTL signal. The minimum pulse length or pulse pause of the clock signal should be 80 ns. See also [Chapter 4.6.1 "LASER Connector", page 62](#), [Section "2-Bit Digital Input Port", page 63](#).
- No synchronization is carried-out, if the synchronization is activated but there is no valid clock signal at DIGITAL IN1 of the LASER-connector. In this case the RTC6 PCIe Board uses its internal  $10 \mu\text{s}$  clock period.
- If the synchronization is activated by **set\_laser\_control** by setting **Bit #6**, then the synchronization of RTC6 PCIe Board and external clock is subtly (depending on the respective phase position within 1.2 ms at most).



Free-running laser and RTC6 PCIe Board – example of marked lines.

## 7.5 Marking Dates, Times and Serial Numbers

The **RTC6 DLL** contains a series of commands to mark the current time, current date or the serial number of products.

Before times, dates and serial numbers can be marked, the required characters and text strings must be defined as indexed characters and indexed text strings.

Separate text strings can be defined for marking times/dates and serial numbers. See [Section "Defining Indexed Text Strings for Time, Date and Serial Number", page 109](#).

### 7.5.1 Marking the Time and Date

By **time\_update** the PC time/date is transferred to the RTC6 PCIe Board. This is necessary after every PC restart. After that, the board (as long as it remains energized) internally counts the date/time with the quartz-controlled 10 µs clock to the second.

By **time\_fix**, **time\_fix\_f** or **time\_fix\_f\_off** the current time/date of the board is held in a cache.

The time (hours, minutes, seconds) can be marked by **mark\_time** or **mark\_time\_abs** and the date (year, month, day, day-of-the-week) by **mark\_date** or **mark\_date\_abs**.

To mark the date and time, the Gregorian or Julian date can be set as well as the 12- or 24-hour format.

For marking dates of expiry, **time\_fix\_f\_off** is available to fix a forward date based on the current date and current time.

### 7.5.2 Marking Serial Numbers

By **mark\_serial** and **mark\_serial\_abs**, up to 12-digit serial numbers can be marked. It can be specified how leading zeros are handled.

The RTC6 PCIe Board manages up to 4 serial-number-sets (each with its own serial number and increment size). Serial-number-set 0 is selected at initialization with **load\_program\_file**.

Other serial-number-sets must be selected in advance by **select\_serial\_set** or **select\_serial\_set\_list** (see notes below).

By **set\_serial**, **set\_serial\_step** or **set\_serial\_step\_list**, a starting serial number (max. 10 digits) and an increment size for each serial-number-set can be specified. At initialization with **load\_program\_file** all starting serial numbers are set to 0 and all increment sizes are set to 1.

Each call of **mark\_serial** or **mark\_serial\_abs** causes the current serial number to be incremented (yet before execution of the serial number marking) by the specified increment size.

If a serial number is to be omitted a blank marking can be executed (**Digits = 0**), which increments the serial number by 1 (*not* by the specified increment size).

#### Notes

- If a serial-number-set is to be marked by **mark\_serial** or **mark\_serial\_abs**, then you can only select that set by the list command **select\_serial\_set\_list**. **mark\_serial**, **mark\_serial\_abs** and **set\_serial\_step\_list** are always applied to the serial-number-set most recently selected by **select\_serial\_set\_list**.
- You can use the control command **get\_list\_serial** to query the number of the serial-number-set most recently selected by **select\_serial\_set\_list** as well as the current serial number of that set. This also lets you determine (among other things) whether the current number was or was not incremented after an uncontinued aborted list.
- The control command **select\_serial\_set** lets you select (independently of selection by **select\_serial\_set\_list**) a serial-number-set for the control commands **set\_serial\_step** and **set\_serial** (Note that the RTC6 PCIe Board does not prohibit modifying parameters of the serial-



number-set currently being marked). The control commands `set_serial_step` and `set_serial` always apply to the serial-number-set most recently selected by `select_serial_set`.

- `get_serial` returns the current serial number of the serial-number-set selected by `select_serial_set` (if multiple serial-number-sets exist, then the returned serial number is not necessarily the most recently marked serial number).
- `set_max_counts` allows specification of the maximum number of external starts and thus the maximum number of externally started markings. The number of already occurred external starts can be obtained with `get_counts`. When a single serial-number-set is used, these commands let you indirectly set the maximum serial number and query the current serial number. But when multiple serial-number-sets are used, these commands do not differentiate between the various serial-number-sets.



## 8 Advanced Functions for Scan Head Control and Laser Control

### 8.1 iDRIVE Functions

SCANLAB iDRIVE scan systems<sup>(1)</sup> utilize the iDRIVE technology. This servo and control approach exploits the advantages of fully digital servo electronics to deliver significantly expanded functionality. An enhanced transfer protocol between the servo electronics and the controller facilitates support of all the new features (see also the following section).

This allows users to adjust a number of settings of the respective scan system, for example,

- To select which data it has to send back to the controller
- To choose from different dynamic settings (tunings)
- To set the PosAcknowledge threshold value
- To set the effective calibration of the scan system
- To set the start behavior of the scan system
- To perform a fault diagnosis
- To perform a functional test of the data transfer

For more information, refer to the manual of the scan system.

iDRIVE functions are executed by **control\_command**<sup>(2)</sup>.

#### 8.1.1 Transfer Protocol

Data transfer between RTC6 PCIe Board and scan system is carried out according SL2-100 protocol, which supports the full functionality of iDRIVE technology.

With iDRIVE scan systems<sup>(1)</sup> this protocol allows, for example, status signals of the x and y axes to be separately and simultaneously evaluated. For a 3D scan system, z axis status signals can be simultaneously read back by the channel of the scan head connector to which the z axis is connected.

The SCANLAB XY2-100 converter, see [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)", page 58](#), can be used with iDRIVE scan systems<sup>(1)</sup> equipped with a XY2-100 interface or XY2-100 Enhanced interface.

(1) See glossary entry on [page 879](#).

(2) See also [Section "Folder iSCANCfg", page 25](#).

## 8.1.2 Configuring the Data Signal Return Behavior of the Scan System

### Setting Data Types

The digital servo architecture of iDRI<sup>VE</sup> scan systems<sup>(1)</sup> allows a wide variety of data signals to be returned from the scan system to the RTC6 PCIe Board.

Each axis is assigned its own status channel which transmits data to the RTC6 PCIe Board every 10 µs:

- The STATUS channel is designed for the x axis (galvanometer scanner 2)
- The STATUS1 channel is designed for the y axis (galvanometer scanner 1)

This opens up possibilities such as monitoring the actual values of the galvanometer scanners during an application or carrying out comprehensive troubleshooting in case of operational malfunction.

`control_command( Data = 05nnH )` allows to set which data the scan system has to return to the RTC6 PCIe Board. The available data types are described in detail in the manual of the respective scan system and (in parts) in the command reference of the `control_command`. The set data source is transferred until another data source is set.

After every power-up or reset (after the initialization has been completed), the scan system transmits (on all receive channels) the XY2-100 status word.

### Reading Out Data

At any time, data received by the RTC6 PCIe Board can be:

- Read out asynchronously by `get_value`, `get_values` or `get_head_status`
- Synchronously recorded by `set_trigger/set_trigger4`

See also [Chapter 7.3.7 "Status Monitoring and Diagnostics", page 172](#).

Note that switching of the data source is followed by a short (serial transmission-related) delay of typically 50 µs before the first data is transmitted, see also comment at [control\\_command](#), page 339.

The value ranges of these data and the possible status states are described in the command reference of [control\\_command](#).

### Notes

- `get_head_status` queries the XY2-100 status word. With SL2-100 protocol-compliant data transfer, the scan system always transfers the XY2-100 status word in parallel with other status information.

Important: If the SCANLAB XY2-100 converter is used for control, see [Chapter 4.5.2 "XY2-100 Converter \(Accessory\)", page 58](#), then it must explicitly be set that the scan system has to transfer the XY2-100 status word to the RTC6 PCIe Board. Otherwise, `get_head_status` returns unusable values.

(1) See glossary entry on [page 879](#).

### 8.1.3 Monitoring the Positioning

For some applications, it is important to monitor and, if necessary, to document the scan system positioning even during operation.

For this purpose, the actual position of the scan axes must be set to be returned from the scan system by **control\_command**. The returned actual position values can be queried then by **get\_values** or recorded by **set\_trigger/set\_trigger4**<sup>(1)</sup>.

If the returned actual positions of the scan axes are to be compared with the Cartesian target coordinate values (X, Y, Z), then these must be transformed back by the corrections made on the RTC6 PCIe Board, [Chapter 7.3.6 "Output Values to the Scan System", page 170](#).

For runtime reasons, the backward transformation needs to subsequently be performed by the PC rather than on the RTC6 PCIe Board itself. For this, all correction and transformation settings currently assigned to the scan system can be transferred from the RTC6 PCIe Board to the PC by **upload\_transform**. Afterwards, an individual xy value pair or an individual Z value can be backward transformed by **transform**. With **get\_transform** (see also **get\_waveform**), an entire series of xy value pairs or Z values previously recorded by **set\_trigger/set\_trigger4** can be backward transformed.

#### Notes

- For some forward transformations, a backward transformation is not possible:

Forward transformation	Backward transformation possible?
Wobbel motion (#2 on <a href="#">page 170</a> )	no
Global coordinate transformation (#3 on <a href="#">page 170</a> )	no
Processing-on-the-fly correction (#4 on <a href="#">page 170</a> )	no
Coordinate transformations (total matrix and offset) to the x and y coordinates (#5 and #6 on <a href="#">page 170</a> )	yes
Offset to the z coordinate (#6 on <a href="#">page 170</a> )	yes
Clipping to the limits of the controllable image field (#7 on <a href="#">page 170</a> )	no
2D/3D image field correction (#8 on <a href="#">page 170</a> )	yes
Gain and offset correction of automatic self-calibration (#9 on <a href="#">page 170</a> )	yes
Clipping to the maximum possible control values	no

- Furthermore, backward transformation is not possible if a noninvertable transformation matrix has been defined during forward transformation (notice: clipping to  $\pm 50$  for each individual matrix coefficient; see [Chapter 8.2 "Coordinate Transformations", page 210](#)). The non-invertability is already reported by **upload\_transform**.
- If forward transformation included a clipping to the edges of the positionable image field or to the edges of the maximum possible range of control values, then backward transformation (ideally) calculates the Cartesian coordinates of these edge values instead of the original values.

(1) Due to communication runtimes, the currently returned actual positions are several clock pulses later than the currently output control signals.

- By **get\_values**, four arbitrary signals can be queried at the same time. Example:
  - the actual position of galvanometer scanner 2 by StatusAX
  - actual position of galvanometer scanner 1 by StatusAY
  - the actual z axis position by StatusBX
  - an additional desired signal, for example, LaserOn
 In contrast, **get\_value** (not: **get\_values**) is not useful for monitoring xy positioning because it is only meant for querying a single signal and multiple calls unavoidably lead to XYZ values across different points of time.
- With **set\_trigger**, you can simultaneously record two desired signals by two measurement channels (with **set\_trigger4** four signals by four measurement channels).
- With **transform** and **get\_transform**, you can use the parameter `Code` to specify that values queried by **get\_values** or **set\_trigger** are to be assigned to X, Y or Z for backward transformation.
- **transform** and **get\_transform** also allow specification of which partial transformations are to be performed.
- Values queried by **get\_values**, or arbitrary synthetic values can be backward transformed by **transform**. During backward transformation of synthetic values beyond the forward transformation's achievable image field, values sometimes are only calculated by extrapolation, due to possible range exceedances or other errors.
- If the user program binarily stores both the recorded values and the transferred transformation data (see **get\_waveform**), then subsequent backward transformation by **transform** (not **get\_transform**) can also be executed offline, hence without needing to further access a RTC6 PCIe Board.

- If **control\_command** is used to specify positioning error rather than actual position as the to be returned data type by the scan system, then it is not possible to directly compare the originally defined pattern with the marked pattern. But you can check if the scan system correctly processed the RTC6 PCIe Board output values. This is particularly useful if backward transformation of actual values is not (fully) possible or when it cannot be determined if deviations between backward-transformed actual positions and originally defined coordinate values are due to scan system error or clipping during forward transformation.

#### 8.1.4 Selecting the Dynamics Setting (Tuning)

SCANLAB can optimize the dynamics setting of scan systems (tuning) to accommodate differing requirements of diverse applications regarding the laser positioning dynamics, for example:

- to execute vectors or circular arcs at a constant processing speed ("vector tuning")
- to execute jumps of minimized duration ("jump tuning")

iDRI<sup>E</sup> scan systems<sup>(1)</sup> can be optionally equipped with several tunings. For different applications, the suitable tuning can be set by **control\_command**(`Data = 0526H`).

For scan systems equipped with one or several jump tunings, you can also activate jump mode (and hereby tuning autoswitching) for 2D jumps, see Chapter 8.1.5 "Jump Mode", page 203.

The default set start behavior is that the scan system starts with tuning number 0 upon power-up or after a reset.

(1) See glossary entry on page 879.

### 8.1.5 Jump Mode

For applications such as drilling holes with defined spacing (whereby laser processing is actually point-by-point rather than along lines and curves), you can optimize process times by activating the so-called jump mode.

This requires the scan system to be equipped with a jump tuning, see also [Section "Requirements and Activation", page 204](#).

#### Functional Principle

In the default setting (after `load_program_file`), both jump commands and [Mark commands](#) are executed in vector mode:

- The jump length gets subdivided into individually executable microsteps in accordance with the current jump speed. If the scan system is only equipped with a jump tuning, then the microsteps execute using this tuning.
- A jump delay defined by `set_scanner_delays` is executed before a subsequent list command.

In contrast, when jump mode is enabled and activated by `set_jump_mode` or `set_jump_mode_list`, every 2D jump (see below) is executed as follows:

- The entire jump length of the 2D jump is controlled as a “Hard jump” over a time dimensioned jump of  $10 \mu$  duration. The target position is executed without microstepping.
- The jump executes with a jump tuning. `set_jump_mode` can be used to designate which jump tuning to use. If a different tuning was set before the jump, then the RTC6 PCIe Board automatically switches at the beginning of the jump to the tuning specified by `set_jump_mode`.
- At the end of the 2D jump, the RTC6 PCIe Board automatically switches to a vector tuning (if the scan system is equipped with one and if a corresponding setting has been made by `set_jump_mode`).

- At the end of the 2D jump, a jump-length-dependent jump delay occurs. This jump delay can be specified for the corresponding jump length by `load_jump_table_offset` or `set_jump_table`, see also [Section “Jump-Length-Dependent Jump Delays”, page 204](#). Here, an external jump delay specified by `set_scanner_delays` is not taken into account.

#### Notes

- Jump mode works exclusively on
  - `jump_abs`, `jump_rel`, `goto_xy` (not on the corresponding 3D, para or timed commands)
  - home jumps and home returns (see `home_position`)
- If a 2D jump occurs where the jump length limit (`Length` parameter) specified by `set_jump_mode` is not reached or exceeded on at least one of the two axes, then the jump executes in vector mode even if jump mode was enabled and activated. This allows exploitation of the fact that *short* jumps can in some circumstances execute faster by vector tuning than with jump tuning. But if no vector tuning is installed or none specified, then you should set the `Length` parameter to 0.
- Each switch between different tunings (servos) requires an additional  $10 \mu$  clock period. For applications such as pure drilling, this can be avoided by not specifying a vector tuning to switch back to when you call `set_jump_mode`.
- When you deactivate or disable jump mode (by `set_jump_mode` or `set_jump_mode_list`), then subsequent jumps again execute in vector mode (split-up into microsteps and without further servo autoswitching). Here, the vector tuning is used that was most recently set at the end of jump mode, unless deactivation was followed by selection of a different tuning by `control_command`. Moreover, the most recently set jump speed is again used and jumps are followed by the jump delay specified by `set_scanner_delays`.



## Requirements and Activation

The following are required for enabling and activating jump mode:

- At least one of the two scan head connectors must have been assigned a correction table.
- At least one of the two scan head connectors must be connected to an intelliSCAN, intelllicube, intelliWELD or intelliDRILL scan system.
- The software version of the attached scan system(s) must be 2078 or higher.
- The attached scan system must be equipped with at least one jump tuning. In contrast, a vector tuning is not absolutely required.
- The tuning numbers specified by `set_jump_mode` must match those stored on the board.
- The tunings specified by `set_jump_mode` must be of the proper type – vector tuning or jump tuning – (the tuning type is stored as info in the scan system's firmware) and must be suitable for rapid switching.

Before jump mode can be activated by `set_jump_mode_list`, it must have been successfully enabled at least once by `set_jump_mode` (see command description).

The `set_jump_mode` control command (but not the `set_jump_mode_list` list command) performs an appropriate check if jump mode was not already enabled.

## Jump-Length-Dependent Jump Delays

When executing a “Hard jump”, it takes the scan head some time to reach the specified position.

The RTC6 PCIe Board takes this delay (also called step response) into account by appending a jump delay at the end of the jump.

Point-by-point laser processing does not need to take other scanner delays into account and you can generally set laser delays to 0.

The specific step response behavior of the respective scan system (step response time vs. jump length) can be stored on the RTC6 PCIe Board in a user-specific jump delay table. With jump mode enabled, the RTC6 PCIe Board uses the specified jump delay table to determine the appropriate jump delay value for each 2D jump in accordance with the jump's longer edge (that is, either the x or y component of the jump).

You can determine the step response behavior experimentally and then load it onto the board as a table of values using the `load_jump_table_offset` command. Alternatively, the jump delay table can also be automatically determined by `load_jump_table_offset` (parameter `Name = NULL`). Additionally, the currently loaded jump delay table can be retrieved as a binary table by `get_jump_table` and reloaded onto the board by `set_jump_table`.

The step response time (at least for longer jumps) typically scales with the squareroot of the jump length, and `load_program_file` accordingly initializes the internal jump table – with an end value of 10.24 ms for a jump length of  $2^{20}$  bits.

When the jump delay table has been loaded and a new `RTC6DAT.dat` file is created by `create_dat_file`, then this table is automatically loaded upon the next `load_program_file`.

## Determining Jump Delay Values Experimentally

The user manual of the scan system typically specifies the step response times for each jump tuning at selected jump lengths.

To experimentally determine the step response behavior, you need to have the scan system perform jumps of various lengths and query the resulting position values by the status channel for analysis.

After you activate jump mode, perform the jumps by using **jump\_abs** or **jump\_rel**. The scan system should have been previously set to return the actual position data type by **control\_command**. You can then record the latter by **set\_trigger/set\_trigger4** and retrieve it by **get\_waveform**.

The determined jump delay values must be supplied in an ASCII text file. If the step response behaviors of both axes differ, then the higher of the two axes' jump delay values should be supplied in the ASCII text file.

## Notes on Loading Determined Jump Delay Values

- For jump length values and jump delay values, **load\_jump\_table\_offset** loads a table from an ASCII text file.
- The ASCII text file can contain one or several tables.<sup>(1)</sup>
- Each table can contain up to 50 data points (*Length* | *Delay(Length)*).
- The complete (internal) jump delay table *Delay(Length)* is linearly interpolated from the data points.

For the tables, the following rules apply:

- Each table must begin with the line:  
`[JumpTable<No>]`  
`<No>` represents the table number.
- If the table contains multiple `[JumpTable<No>]` entries with the same `<No>`, then only the lines after the first entry are used. Only lines up to the next '`'` character (that is not preceded by a semicolon) are used.
- Each data point (*Length* | *Delay(Length)*) is defined as follows:  
`Length<n> = <LengthValue>`  
`Delay<n> = <DelayValue>`  
 where `<n>` is the data point index ( $1 \leq <n> \leq 50$ ).  
 The `<Value>` numbers can be supplied as (unsigned) floating point numbers. Decimal separator: period (.) .
- If the table contains multiple data points with the same index `<n>`, then the most recently read one is used and the previous ones ignored.
- If the table contains multiple data points with the same jump length value *Length*, then the data point with the largest index `<n>` is used and the others ignored. Equality is checked to within  $\pm 0.01$ .

(1) Even of another type, see table 1, page 142.



- For <Value> the following ranges apply:  
 $0.0 \leq Length \leq 1048576.0$   
 $0.0 \leq Delay(Length) \leq 65535.0$   
Delay values are supplied in units of  $10 \mu\text{s}$ , jump lengths in bits.
- Each instruction must be in a separate line.
- Space characters and tabs within a line (for example, between '=' and <Value>) are ignored.
- Empty lines are ignored.
- Data points with invalid values are ignored.
- The data point of a particular index <n> is ignored if the corresponding Length<n> and/or Delay<n> definition is missing.
- The semicolon ';' can be used for comments. All characters in a line following a semicolon are ignored.
- The instructions for data points in the table can be ordered as desired.
- Indices for data point pairs in the table can be selected as desired within the range [1...50] (the table is then automatically sorted by ascending position values).
- If the table contains no valid data points, then **load\_jump\_table\_offset** has no effect (return value 1 or 13).
- If there is no entry of  $Length = 0.0$ , then one with  $Delay = \text{Min}(Delay<i>)$  is inserted (the smallest valid value encountered is filled downward). The same applies to  $Length = 524288.0$  with  $\text{Max}(Delay<i>)$ .
- If the specified text file contains only one valid data point with  $Delay<n> = D$ , then the jump delay table  $Delay(Length) = D$  (for the whole jump length range) is loaded.

## Automatic Determination of the Jump Delay Table

You can initiate automatic determination of the jump delay table by `load_jump_table_offset` (parameter `Name = NULL`) if you had previously enabled and activated jump mode successfully by `set_jump_mode` (`Flag = 1`).

For automatic determination, "Automatic Laser Control" is deactivated, the data type to be returned by the scan system is set to target position and the tuning set to the jump tuning that had been defined by `set_jump_mode` (the original settings are restored when `load_jump_table_offset` completes).

For automatic determination, several diagonal jumps of varying lengths are performed. For each jump, the target position returned by the scan system is recorded by `set_trigger` (not: `set_trigger4`) and retrieved by `get_waveform`. The data is analyzed for the timepoint at which the specified position tolerance (`PosAck` parameter) was last exceeded, that is, when the target position persistently remained in the jump target positional range  $\pm$ `PosAck`. This value is then reserved as the jump delay value associated with the corresponding jump length.

### Notes

- Before automatic determination, you must absolutely switch off the laser.
- Data recording requires execution of a list with a total of six commands. The commands are automatically written to list memory and processed. The parameter `ListPos` indicates the position in list memory ("list 1" or "list 2") in which storage is to occur. This position should be such that any previously entered list commands can be harmlessly overwritten.

- For automatic determination, the longest jump is performed first, followed by increasingly shorter jumps (with a maximum of up to 16 different jump lengths). For the first (longest) jump length (jump across the entire image diagonal), the measurement period is specified by the parameter `MaxDelay` [10  $\mu$ s]. For subsequent (shorter) jumps, it is defined by each previously determined jump delay. `MaxDelay` should be chosen to be adequate but not significantly larger than the jump delay for the longest jump. A larger `MaxDelay` increases the total required execution time. With `MaxDelay` = 500, the total execution time for automatic determination is typically a few seconds.
- For statistical noise reduction, four identical jumps are performed for each jump length and the results averaged. Additionally, the values for each individual measurement are low-pass filtered (2-point smoothing). This permits selection of a position tolerance `PosAck` that can also be somewhat (but not substantially) under the expected noise level (but note that XY2-100 converters only return whole multiples of 16. Here, a noise level of  $\pm 3 \times 16$  bits can be expected).
- If the `PosAck` range is not persistently reached within the measurement period, then `MaxDelay` becomes the determined jump delay.
- If the determined jump delay is smaller than `MinDelay`, then `MinDelay` becomes the determined jump delay and the measurement terminates. Then, shorter jumps are no longer performed and `MinDelay` is also the determined jump delay for these shorter jump lengths. A longer `MinDelay` reduces the total execution time for automatic determination.

- If an offset is specified for automatic determination (by the according `load_jump_table_offset` parameter), this offset is added to all automatically determined delay values before the overall jump delay table gets calculated by linear interpolation and loaded onto the board (in addition, the delay values are clipped to the value range 0...65,535). The Offset can be used to compensate for measurement runtime latencies (for example, caused by an XY2-100 converter, by tuning switching or by a runtime latency of the signal returned by the scan system) when calculating the jump delay table. It can also be used to add a safety margin to the delay values to compensate for noise-induced random deviations. The `load_jump_table` command is identical to `load_jump_table_offset` with `Offset = 0`.
- For simultaneous control of two scan systems, you should determine the jump delay values for both systems and, after comparing, use the values of the slower system.
- The resulting table can be retrieved in binary form by `get_jump_table` and reloaded onto the board by `set_jump_table`.

## 8.1.6 Configuring the PosAcknowledge Threshold Value

`control_command(Data = 15nnH)` can be used to set the PosAcknowledge threshold value nn. The default start behavior is for the scan system to set the threshold value to 0.28% of the full position range after every power-up.

If other threshold values are desired, they must be separately set for each axis.

## 8.1.7 Configuring the Effective Calibration

The servo electronic can be configured by `control_command(Data = 12nnH)` to down scale the position values received from the RTC6 PCIe Board by a specific factor (1, 1/2, 1/4 or 1/8). The position signals (optionally) returned by the scan systems to the RTC6 PCIe Board remain unaffected, as do the pre-configured calibrations of SCANLAB's scan systems. However, the effective calibration can be thereby reduced to confine the scan area to a smaller angular range – with a higher angular resolution.

The default start behavior is for the scan system to start with a scale factor of 1 upon power-up.

By `control_command(Data = 053FH)`, the currently set scale factor can be queried.

### 8.1.8 Configuring the Start Behavior

The default configuration of iDRI/VE scan systems<sup>(1)</sup> is set as follows:

- Tuning number 0, see also [Section "Selecting the Dynamics Setting \(Tuning\)", page 202](#)
- PosAcknowledge threshold value is  $B8_H$  (corresponds to 0.28% of the full position range of  $2^{16}$  bit), see also [Section "Configuring the PosAcknowledge Threshold Value", page 208](#)
- Scale factor = 1, see also [Section "Configuring the Effective Calibration", page 208](#)

These settings can be changed by [\*\*control\\_command\*\*](#). The changed settings are only temporary, however they can be additionally saved as starting settings for subsequent power-ups by [\*\*control\\_command\(Data = 0A00<sub>H</sub>\)\*\*](#).

The return behavior of the scan system can only be temporarily changed. After a power-up, the scan system transmits the XY2-100 status word, see also [Section "Configuring the Data Signal Return Behavior of the Scan System", page 200](#).

### 8.1.9 Fault Diagnosis and Functional Test

If a problem occurs, the versatile status return functions of the iDRI/VE scan system<sup>(2)</sup> can be used for scan system diagnosis, too. These functions allow to read for instance:

- The current operating state
- The operating state at the moment of the most recently occurred operation interruption
- An event code that indicates which event has been responsible for the change to an error state

To verify that data transfer capability between the RTC6 PCIe Board and a scan system is intact, by [\*\*control\\_command\(Data = 21nn<sub>H</sub>\)\*\*](#) an 8-bit value nn – separately for each axis – can be transmitted to the scan system. Subsequently, a 20-bit value is returned on the corresponding status channel: If data transfer is error-free, then the upper 8 bits of the returned 20-bit value is identical with the originally sent 8-bit value, and the next lower 8 bits are identical with the complement of the sent 8-bit value.

These 20-bit values are returned until [\*\*control\\_command\(Data = 05nn<sub>H</sub>\)\*\*](#) is used to select another return data type, see [Section "Configuring the Data Signal Return Behavior of the Scan System", page 200](#).

Prior to a transfer test, the data type currently selected for transmission can be cached by [\*\*control\\_command\(Data = 17FF<sub>H</sub>\)\*\*](#). After the test, [\*\*control\\_command\(Data = 1700<sub>H</sub>\)\*\*](#) restores the same feedback behavior as before the test.

(1) See glossary entry on [page 879](#).

(2) See glossary entry on [page 879](#).

## 8.2 Coordinate Transformations

For precise set-up of the scan system relative to the image field (or, if the **Option "Second Scan Head Control"** is enabled, two scan heads can be adjusted relative to a *common* image field), a linear coordinate transformation can be defined (separately for the first and second scan head connectors) for all X and Y output coordinates ( $x|y$ ) defined by vector or arc commands:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

The ( $2 \times 2$ ) total matrix  $M$  is thereby automatically calculated by the RTC6 PCIe Board as a product of a scaling matrix  $M_S$ , a rotation matrix  $M_R$  and a general transformation matrix  $M_T$ :

$$M = M_T \times M_R \times M_S$$

The coefficients of the three matrices ( $M_T$ ,  $M_R$ , and  $M_S$ ) and the offset values ( $x_0|y_0$ ) can be individually defined for the first and second scan head connector.

The offset ( $x_0|y_0$ ) is set by **set\_offset** or **set\_offset\_list**.

For 3-axis scan systems, **set\_offset\_xyz** or **set\_offset\_xyz\_list** enables setting of an offset  $z_0$  for the z coordinate, too ( $z_0$  has the opposite effect of **set\_defocus** or **set\_defocus\_list**). The following applies:

$$z' = z + z_0$$

The coefficients of the scaling matrix  $M_S$  are set by **set\_scale** or **set\_scale\_list** using a scaling factor  $k$  that is common to both axes:

$$M_S = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

The coefficients of the rotation matrix  $M_R$  are set by **set\_angle** or **set\_angle\_list** by specifying a rotation angle  $\alpha$  (in accordance with mathematical convention: positive angles produce counter-clockwise rotation):

$$M_R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

The coefficients  $m_{11}\dots m_{22}$  of the general transformation matrix  $M_T$  are set by **set\_matrix** or **set\_matrix\_list**:

$$M_T = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

With the general transformation matrix  $M_T$ , the two above matrices ( $M_S$  and  $M_R$ , as special case) as well as further transformations for scaling, rotating, mirroring or skewing objects can be defined:

- Scaling by the factors  $k_x$  and  $k_y$ :

$$M_T = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$$

- Rotation by the angle  $\alpha$ :

$$M_T = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

Example:

`set_matrix( 1, 0.5, -0.866, 0.866, 0.5, 1 )`  
defines a rotation by 60° (counterclockwise)  
around the center of the image field for the first  
scan head connector.  
This can also be achieved by `set_angle(1, 60)`.

- Mirroring around the y axis  
(flipping in the x direction):

$$M_T = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Mirroring around the x axis  
(flipping in the y direction):

$$M_T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- Mirroring around the first dimension diagonal  
(exchanging the x and y coordinates):

$$M_T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Skewing in the x direction by the angle  $\alpha$   
(slanting):

$$M_T = \begin{bmatrix} 1 & -\sin \alpha \\ 0 & 1 \end{bmatrix}$$

Example: `set_matrix( 1, 1, -0.25, 0, 1, 0 )`

A general transformation defined by `set_matrix` or `set_matrix_list` can also represent a combination of various transformations (users can calculate the corresponding matrix  $M_T$  by multiplying the corresponding individual matrices in the correct order).

## Notes

- The described coordinate transformations are primarily intended for small corrections when setting up the scan system relative to the image field. Separate settings for scaling and rotations thereby provide more handling flexibility in comparison to a single matrix setting.
- Initialization by `load_program_file` results in an offset of  $(0|0|0)$  and in matrices  $M_S$ ,  $M_R$  and  $M_T$ , each predefined as identity matrices.
- Each matrix or offset definition overwrites prior definitions.
- The RTC6 PCIe Board calculates the total matrix  $M$  independently of the order in which the individual transformation matrices were defined.
- The value range of scaling factor  $k$  for the scaling matrix  $M_S$  is  $[-16\dots+16]$ . The value range for the coefficients of the general transformation matrix  $M_T$  is  $[-50\dots+50]$ . Also be sure that the value range  $[-50\dots+50]$  for individual coefficients of the total matrix  $M$  are not exceeded; otherwise calculation of the corrected coordinates might result, under some circumstances, in overflows.
- Rotations take place exclusively around the centerpoint of the image field; mirroring is relative to the axes.
- For each definition, the parameter `at_once` can be used to specify whether the new setting should have immediate effect on the current position (`at_once = 1` or `3`) or whether it should only be provisionally accumulated and cached (`at_once = 0` or `2`). The most recently issued `at_once` parameter value determines when a (accumulated) transformation takes effect.

- Before the total transformation is applied to the current position, the signals for “laser active” operation are switched off with `at_once = 0...2`. They remain unchanged with `at_once = 3`.
- With `at_once = 1` or `3`, all settings (only) accumulated until then are processed immediately and simultaneously. In the process, the scan system axes are moved from the current position to the corrected position at the defined jump speed. Consequently, this can require some clock cycles. This needs to be observed especially when RTC6 PCIe Boards are master/slave synchronized and are supposed to execute different jump lengths. With this use case, the coordinate transformations should be executed prior the synchronized start with `at_once = 1` and their end should be waited for.  
Any scanner delays are not initialized. The **INTERNAL-BUSY** status is set while the jump to the corrected position is executed.
- With `at_once = 2`, the accumulated settings only become effective upon execution of the next **`jump_abs`**, **`jump_rel`**, **`goto_xy`** or **`goto_xyz`** (but not other jump commands such as **`jump_abs_3d`** or **`jump_rel_3d`**) unless afterwards another call triggers immediate execution. Correction of the current output position then occurs together with the specified coordinate jump. This eliminates unnecessary galvanometer scanner motions (incl. delays).  
Example: The following command list produces a first jump to `(0, 0)`, followed by – if `at_once = 1` or `at_once = 3` – a second jump from `(0, 0)` to `(1000, 500)` and a third from `(1000, 500)` to `(0, 500)`. But if `at_once = 2`, then only a second jump occurs from `(0, 0)` to `(0, 500)`.

```
jump_abs( 0, 0 );
set_offset_list( 1000, 500, at_once );
jump_abs( -1000, 0 );
```
- Settings via control commands by `at_once = 0` are only saved as long as no list is running. When the list is started or the list is running, the settings take effect immediately before the next list command.  
Settings via list commands with `at_once = 0` are only saved until they are retrieved elsewhere (`at_once > 0` or by a control command).
- If no correction table is assigned to the corresponding scan head connector, then the new settings for the coordinate transformations are only stored on the RTC6 PCIe Board. They take effect when a correction table is assigned.
- Coordinate transformations are applied to all to-be-outputted coordinates from all vector commands (jump or mark list commands, but also **`goto_xy`** or **`goto_xyz`**) and arc commands. See also [Chapter 7.3.6 “Output Values to the Scan System”, page 170](#).
- With a scaling matrix, the effective jump speed and mark speed changes.
- With 3D vector commands:
  - The transformation matrix only affects the x and y components
  - The offset affects all three components
- In order to utilize the complete real image field with coordinate transformations (such as rotations, shrinkages or shifts), the extended value range of the virtual image field can be used even without Processing-on-the-fly, see [Chapter 7.3.3 “Virtual Image Field”, page 158](#).
- Coordinate transformations for the virtual image field can be defined, see [Section “Coordinate Transformations in the Virtual Image Field”, page 158](#).  
See also [4 in Chapter 7.3.6 “Output Values to the Scan System”, page 170](#).



### Notes for RTC4 Users

- With RTC6 PCIe Boards, coordinate transformations can no longer be set upon loading a correction file by `load_correction_file`. Instead, coordinate transformations are separately defined for the first and second scan head connector and serve the same purpose as those of `load_correction_file` of the RTC4.
- RTC6 PCIe Boards do not support the coordinate transformations (collectively for both scan heads) before microstepping which is possible with the RTC4.  
The global coordinate transformations have a slightly different effect than the coordinate transformations above, see [3](#), [5](#) and [6](#) in [Chapter 7.3.6 "Output Values to the Scan System", page 170](#).

## 8.3 Online Positioning

The preceding [Chapter 8.2 "Coordinate Transformations", page 210](#) details how to precisely align a scan system relative to the image field, see also [Section "Coordinate Transformations in the Virtual Image Field", page 158](#).

The user program can, for example, determine the required transformation values by automatic position analysis for a workpiece on a conveyer belt and then execute the associated transformations.

However, it is not easy to achieve well-controlled timing (referenced to the  $10\ \mu s$  clock of the RTC6 PCIe Board) while positioning a workpiece and aligning the scan system by the control commands described in [Chapter 8.2 "Coordinate Transformations", page 210](#). For applications in which such timing is important, commands for so-called **Online Positioning** are available. Here, data for an offset and/or rotation coordinate transformation or a general matrix operation can be inputted by the **McBSP interface**.

The following variants are provided for different use cases:

- The previous "**Local Online Positioning**" concerns the scan system-specific coordinate transformations in the real image field analogous to **set\_offset**, **set\_angle** and **set\_matrix** with parameter `HeadNo = 1` or `2` according to [5](#) and [6](#) in [Chapter 7.3.6 "Output Values to the Scan System", page 170](#). It is described in [Chapter 8.3.1 ""Local Online Positioning""](#), page [214](#).
- The "**Global Online Positioning**" concerns global coordinate transformations in the virtual image field analogous to **set\_offset**, **set\_angle** and **set\_matrix** each with `HeadNo = 4` according to [3](#) in [Chapter 7.3.6 "Output Values to the Scan System", page 170](#). It is described in [Chapter 8.3.2 ""Global Online Positioning""](#), page [217](#).

**Online Positioning** cannot be combined with Processing-on-the-fly applications with position information, but it can be combined with encoder-based Processing-on-the-fly applications.

### Notes

- Since coordinate transformations [5](#) and [6](#) are calculated after the Processing-on-the-fly correction [4](#), larger image field rotations are not well compatible with linear Processing-on-the-fly corrections. In such cases, "[Global Online Positioning](#)" is preferable.

#### 8.3.1 "Local Online Positioning"

Reading in data for "**Local Online Positioning**" via the **McBSP interface** needs to be configured by **set\_mcbsp\_x**, **set\_mcbsp\_y** and/or **set\_mcbsp\_rot** or **set\_mcbsp\_matrix** (or by the equivalent list commands).

With **apply\_mcbsp** or **apply\_mcbsp\_list**, you can acquire the most recent fully transferred values and define the required coordinate transformations (as with **set\_offset** and/or **set\_angle** or **set\_matrix** or the equivalent list commands). Here, as with the commands described in [Chapter 8.2 "Coordinate Transformations", page 210](#) an `at_once` parameter can be used to specify when the newly defined (total) transformation should take effect.

For precise timing, execution of the list command that triggers the transformation (depending on the `at_once` parameter, this would be **apply\_mcbsp\_list** or **jump\_abs** or **jump\_rel** or any other list command) can be made dependent on the input of an external control signal (for conditional command execution, see [Chapter 9.3.2 "Conditional Command Execution", page 281](#)).

The **McBSP interface** is described in [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#).



## Configuring "Local Online Positioning"

`set_mcbsp_x`, `set_mcbsp_y` and/or `set_mcbsp_rot` (or the equivalent list commands) determine both how the values inputted at the **McBSP interface** are interpreted by the RTC6 PCIe Board and which internal memory location is used to read the values:

- Depending on which of the above commands is called, the RTC6 PCIe Board interprets the inputted values as offsets in the x and/or y directions and/or as rotation values or as matrix coefficients. The desired scaling factor always needs to be supplied as a command parameter (except with `set_mcbsp_matrix`, see command description).  
The three options `x`, `y` and `rot` can be used either separately or in any desired combination. By the appropriate command, each option can be enabled or disabled independently of the other two. In contrast, the `matrix` option cannot be used in conjunction with other options.
- As soon as one of the four options becomes activated, all values subsequently inputted at the **McBSP interface** are internally stored in memory location 1 or 2 (see below). Transferred values can subsequently be queried by `read_mcbsp` or applied in coordinate transformations by `apply_mcbsp` or `apply_mcbsp_list`.

- **x, y or rot**

If you activate only *one* of the three options, then X or Y offset correction values can be supplied as *signed* 32-bit values or rotation correction values as *unsigned* 32-bit values.

The **McBSP** input values are transferred to internal memory location 1.

- **x and y (without rot)**

If you activate X and Y offset corrections, but no rotation correction, then the two offset correction values must be supplied as a 16-bit signed value, each, combined to a 32-bit value (the x value in the lower 16 bits and the y value in the upper 16 bits).

The **McBSP** input values are transferred to internal memory location 1.

- **x or y and rot**

If you activate an X or a Y offset correction together with a rotation correction, then the offset and rotation correction values should be alternatingly supplied as 32-bit values. The **McBSP** input values are then alternatingly transferred to internal memory locations 1 and 2.

The RTC6 PCIe Board identifies the data type by examining the coding Bit #31 (Bit #31 = 0 for offset values, Bit #31 = 1 for rotation correction values).

Signed 31 bits are effectively available for transferring offset values. 31 bits *without* sign are available for rotation correction values.

- **x, y and rot**

If you activate all three options together, then the two offset correction values must be combined and supplied as one 32-bit value alternatingly supplied with the rotation correction value as a second 32-bit value. The **McBSP** input values are likewise alternatingly transferred to internal memory locations 1 and 2.

The RTC6 identifies the data type by examining the coding Bit #31 (Bit #31 = 0 for offset values, Bit #31 = 1 for rotation correction values).

Signed 15 bits are effectively available for transferring offset values (whereby x values reside in the lower 16 bits and y values in the upper 16 bits). 31 bits *without* sign are available for rotation correction values.

- The last two cases designate the data type by coding Bit #31. Though this makes the order of transmission irrelevant, always *both* data types nevertheless must be transmitted (preferably always alternatingly). If a request is made by **apply\_mcbsp** (or **apply\_mcbsp\_list**) when two values of the same data type exist in both memory locations, then the most recently transferred value is always used. If two identical Bit #31 codings are present, then the last transfer should have already ended at the time of the request.

- **matrix**

With this option activated, matrix coefficients are then transferable as 32-bit signed values. The indices are encoded in the data word (see command description). **McBSP** input values get transferred to internal memory location 1.

## Notes

- You can use “**Local Online Positioning**” in conjunction with an encoder-controlled Processing-on-the-fly application, but *not* in conjunction with a Processing-on-the-fly application controlled by **McBSP** signals:
  - When you use the commands for configuring “**Local Online Positioning**”, then Processing-on-the-fly correction activated by **set\_fly\_x\_pos**, **set\_fly\_y\_pos**, **set\_fly\_rot\_pos**, **set\_mcbsp\_in**, **set\_mcbsp\_in\_list**, **set\_multi\_mcbsp\_in** or **set\_multi\_mcbsp\_in\_list** gets automatically deactivated. Subsequently, **McBSP** input values are copied to internal memory locations 1 and 2 (see above) and are then available for “**Local Online Positioning**”.
  - In reverse, **set\_fly\_x\_pos**, **set\_fly\_y\_pos**, **set\_fly\_rot\_pos**, **set\_mcbsp\_in**, **set\_mcbsp\_in\_list**, **set\_multi\_mcbsp\_in** or **set\_multi\_mcbsp\_in\_list** deactivate a previously activated “**Local Online Positioning**”. Subsequently, **McBSP** input values are then transferred to the internal memory locations 0 to 3 depending on the command and are then available for the Processing-on-the-fly application.
  - If you switch off (intentionally or with an invalid scaling factor) “**Local Online Positioning**” by **set\_mcbsp\_x**, **set\_mcbsp\_y** or **set\_mcbsp\_rot**, then the data is continued to be copied to internal memory locations 1 or 1 and 2 (as long as data is transmitted), but it is no longer applied (**apply\_mcbsp** has no effect).

### 8.3.2 “Global Online Positioning”

“Global Online Positioning” (available as of RTC6 Software Package ≥ V1.6.1) needs to be activated by one of the following commands:

- `set_mcbsp_global_matrix`
- `set_mcbsp_global_rot`
- `set_mcbsp_global_x`
- `set_mcbsp_global_y`
- `set_mcbsp_global_matrix_list`
- `set_mcbsp_global_rot_list`
- `set_mcbsp_global_x_list`
- `set_mcbsp_global_y_list`

Once activated, all other McBSP processings are deactivated (“Processing-on-the-fly” applications controlled by McBSP signals, “Local Online Positioning” as described in Chapter 8.3.1 ““Local Online Positioning””, page 214, processings activated by `set_mcbsp_in` or `set_multi_mcbsp_in`) and vice versa.

All subsequent data transferred via McBSP are internally handled in the same way as with “Local Online Positioning” (copied to internal memory locations 1 and possibly 2; readable by `read_mcbsp`),

but instead of the scan system-specific coordinate transformations in the real image field (see Chapter 8.3.1 ““Local Online Positioning””, page 214) automatically used for coordinate transformations in the virtual image field instead.

Calling `apply_mcbsp` or `apply_mcbsp_list` is no longer necessary and even has no effect.

Coordinate transformations in the virtual image field (see also Chapter 8.6.4 “Compensating 2D Motions”, page 234) are automatically applied, as soon as a Processing-on-the-fly application is activated *afterwards*.

During a Processing-on-the-fly application, new data can only be sent and stored, but not applied, see also:

- `set_matrix( HeadNo = 4 )`
- `set_offset_xyz( HeadNo = 4 )`
- `set_angle( HeadNo = 4 )`

“Global Online Positioning” is compatible with Processing-on-the-fly applications controlled by encoder signals.

#### Notice!

- The latest transferred data value is used immediately according to the current “Global Online Positioning” mode. Therefore, make sure to transmit correct values after changing that mode and before applying the data by starting a Processing-on-the-fly session.
- Example: `set_mcbsp_global_x` and `set_mcbsp_global_y` combine the xy offsets in the lower and upper half word of the transmitted data. `set_mcbsp_global_y( Scale = 0.0 )` disables the y offset and the latest sent data value is used *in total* as x offset. Make sure to transmit the correct x offset again.

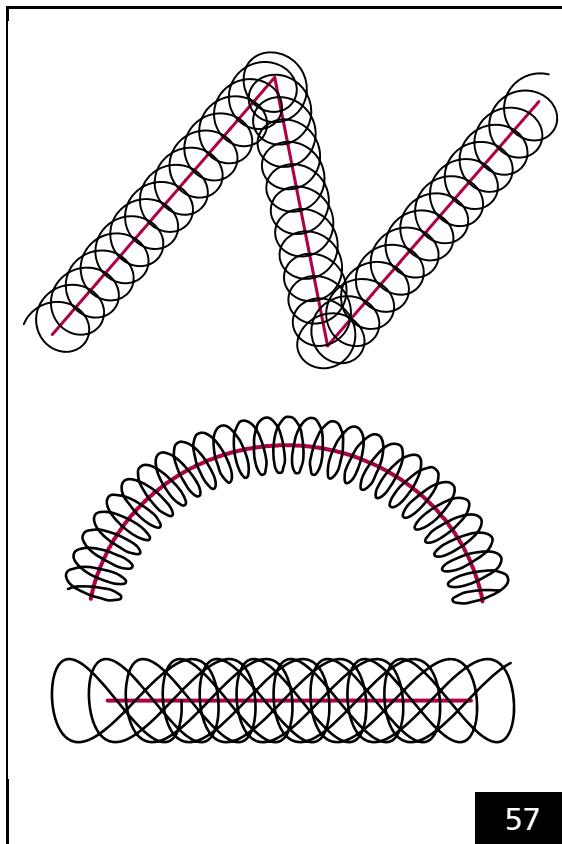
#### Configuring “Global Online Positioning”

The “Global Online Positioning” is configured the same way as the “Local Online Positioning”, see Chapter 8.3.1 ““Local Online Positioning””, page 214.

## 8.4 Wobble Mode

The Wobble mode allows varying the line width for laser marking.

For this purpose, for example, an ellipse-shaped motion is added to the regular, linear movement of the output position. This results in a spiral movement of the laser focus in the image field, see [figure 57](#). Alternatively, the motion can be combined with a horizontal or vertical figure-of-8.



57

Principle of the Wobble mode. Top: circular wobble. Middle: ellipse-shaped wobble. Bottom: figure-of-8 wobble (horizontal 8).

A broadening of the original line is obtained by choosing suitable values for the transverse and longitudinal amplitudes and the frequency of the wobble movement. With figure-of-8s, broader mid-line processing can be achieved by appropriate parameter values. If the specified transverse and longitudinal amplitudes are identical, then the wobble shape remains stationary in space; otherwise the orientation of the wobble shape follows the current direction of motion. If there is no direction of movement, no wobble movement is performed.

During Processing-on-the-fly correction by the [McBSP interface](#) or encoder interface where the scan head only compensates differences between actual external movements and intended total motion, see [Chapter 8.6.2 "Compensation of Linear Movements", page 228](#), a slight jitter in the direction of galvanometer motion might occur, particularly during exact external path motions. Here, the wobble movement superimposed onto the direction of motion does correspondingly jitter, too. You can avoid such jitter by specifying a fixed (instead of the momentary) direction of motion for the wobble shape by the [set\\_wobble\\_direction](#) list command. This is also particularly important if the complete translation movement takes place outside and the galvanometer scanners only have to carry out the actual sweep movement.

After [set\\_wobble](#) or [set\\_wobble\\_mode](#), the wobble start point is always set for the same value relative to the vector/arc startpoint and direction. The Wobble phase is then continued both within an uninterrupted [Polyline](#) and after interruptions (for example, a jump command) until [set\\_wobble](#) or [set\\_wobble\\_mode](#) are called again.

The Wobble mode cannot be combined with:

- Sky Writing
- Pixel output mode
- Jumps<sup>(1)</sup>
- [laser\\_on\\_list](#)

For further details, see [set\\_wobble](#) and [set\\_wobble\\_mode](#).

(1) See also [set\\_wobble\\_mode\\_phase](#).

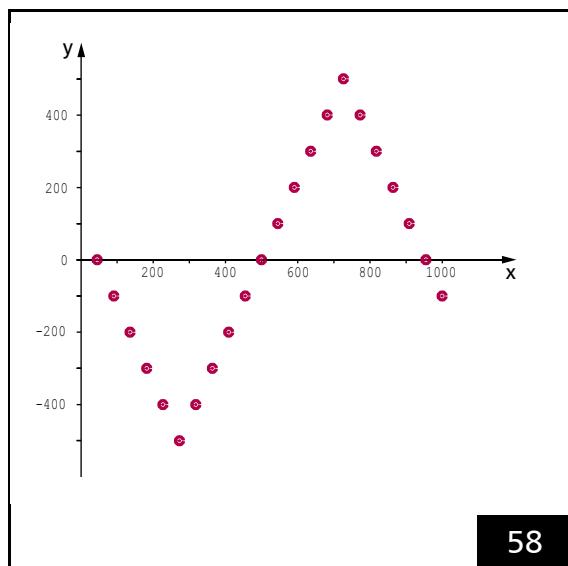
For optimum marking results, the wobble frequency and mark speed must complement each other, see [Chapter 8.4.1 "Wobble Shapes – Important Notes on Choosing Appropriate Parameter Values", page 220.](#)

For many welding applications, the default set of "classic" wobble shapes (circle, ellipse, sine, figure-8) does not produce optimal results in the area of the weld seam. For example, high-speed motion occurs parallel to translation movements and low-speed motion occurs in the opposing direction.

**set\_wobble\_vector** lets you define a wobble shape customized for your user program, consisting of up to 1023 piecewise linear sections, while also specifying variation of laser power along that shape (see also **set\_wobble\_control**). By **create\_dat\_file**, this "freely definable wobble shape" is saved, see comment on [page 342](#).

However, **set\_wobble\_vector** cannot be combined with automatic or vector-based laser control.

The present wobble excursion from **set\_wobble** or **set\_wobble\_mode** can be recorded by **set\_trigger/set\_trigger4** (signal 53). The format of the data is ((transversal << 16) + longitudinal).



See [Section "Example Code", page 219.](#)

## Example Code

The following example C++ source code shows a zigzag pattern, see [figure 58](#).

The code must be included in a user program, see [Chapter 6.2.5 "Example Code", page 90.](#)

```
// Zigzag pattern

// Transversal micro step
const double dTrans( 100.0 );
// Longitudinal micro step
const double dLong( 0.0 );

// Number of steps
const uint16 Period( 5 );

// Start a new wobble shape
set_wobble_vector( 0, 0, 0, 0 );

// 1st wobble vector
set_wobble_vector( dTrans, dLong,
                    Period, 0.0 );
// 2nd wobble vector
set_wobble_vector( -dTrans, dLong,
                    Period * 2, 0.0 );
// 3rd wobble vector
set_wobble_vector( dTrans, dLong,
                    Period, 0.0 );

// if laser power variation is needed, include here
// set_wobble_control( Ctrl, Value,
//                      MinValue, MaxValue );

// Activate freely definable wobble shapes
set_wobble_mode( 100, 0, 100, 2 );

// Mark a vector
timed_mark_rel( 1000, 0, 22*10 );
```

### 8.4.1 Wobble Shapes – Important Notes on Choosing Appropriate Parameter Values

#### “Classic” Wobble Shapes

“Classic” wobble shapes are defined by `set_wobble_mode` or `set_wobble`.

Only assign hardware appropriate values to the Transversal, Longitudinal and Freq parameters.

#### Notice!

- Too big values define control situations where very high waste heat is produced. Galvanometer and digital control board/amplifier board overheating and permanent damage may occur even in short-term operation (overload!). Take the highest possible dynamics of scan head and laser into account.
- If the frequency values are too high the galvanometers may not be able to follow the nominal curve. This may lead to unexpected marking results.

Rule of thumb to estimate appropriate maximum values:

(1) Maximum frequency:  $F = 1/(10 \times T)$

where  $T$  = tracking error<sup>(1)</sup>

(2) Wobble amplitude<sup>(2)</sup>:  $A = T \times V$

where  $V$  = typical positioning speed<sup>(1)</sup>

#### Notes

- Also take the path velocity on the wobble shape itself into account. For circular wobble shapes it is calculated as follows:

(3) Path velocity  $V_{\text{Path}} = 2 \times \pi \times A \times F$ .

- Make sure that the combination of the used values and the trajectory<sup>(3)</sup> velocity are suitable for a long-term operation without causing damages (process safety!).
- Check the temperature status already during an evaluation period (see `get_head_status`) in order to recognize potential overload situations at an early stage. With iDRIVE systems<sup>(4)</sup> you can also read-out the present temperatures of galvanometers and/or digital control boards (see `control_command`).

#### Example of Use

##### System Specification

- Tracking error  $T = 0.33$  ms.
- Calibration  $\pm 0.349 \text{ rad}_{\text{optical}}$   
 $(= 10^\circ \text{ mechanical})$  at  $\pm 503,316$  bit.
- This is an angle control AC of  
 $\approx 1.44 \times 10^6 \text{ bit/rad}_{\text{optical}}$   
 $(\approx 50,300 \text{ bit}^\circ \text{ mechanical})$ .
- Calibration factor<sup>(5)</sup>  $K = 5,000 \text{ bit/mm}$ .
- Typical positioning speed  
 $V_{\text{rad}} = 100 \text{ rad}_{\text{optical}}/\text{s}$ .  
– Converted to control values this corresponds to a typical positioning speed of  
 $V = V_{\text{rad}} \times AC$   
 $\approx 1.44 \times 10^8 \text{ bit/s} = 1,440 \text{ bit}/10 \mu\text{s}$ .
- In the image field this corresponds to a typical positioning speed of  $V/K = 28.8 \text{ m/s}$ .

(3) See Glossary entry on [page 880](#).

(4) See Glossary entry on [page 879](#).

(5) See ReadMe file of correction file or `get_table_para`.

(1) See technical specifications in the scan head manual.

(2) At the maximum frequency estimated with (1).

### *Wobbel Parameters*

- The maximum frequency estimated with rule of thumb 1:  
 $F = 1/(10 \times 0.33 \times 10^{-3} \text{ s}) \approx 300 \text{ Hz.}$
- Wobbel amplitude estimated with rule of thumb 2:  
 $A = 0.33 \times 10^{-3} \text{ s} \times 1.44 \times 10^8 \text{ bit/s}$   
 $\approx 47,500 \text{ bit.}$ 
  - In the image field this corresponds to a wobbel amplitude of  $A/K \approx 9.5 \text{ mm.}$
- Path velocity of circular wobbel shapes as given by rule of thumb 3 is:  
 $V_{\text{Path}} \approx 895 \text{ bit}/10 \mu\text{s.}$
- The *maximum* mark speed results from  $V_{\text{Path}} / K$  (in this example  $89,500,000 \text{ bit/s} / 5,000,000 \text{ bit/m} \approx 17.9 \text{ m/s.}$  Note that this value is a rough estimate. Furthermore, it is dependent on a position due to the correction file (which is non-linear).  
The calculated path velocity for wobbel only (!)
  - should be lower than the specified maximum positioning speed,
  - but there may be certain circumstances where it is much higher than the *typical* mark speed.
Make sure that your values are suitable for operation (temperature status and so on, same as above).

### **"Freely Definable Wobbel Shapes"**

When defining "freely definable wobbel shapes" (see [set\\_wobbel\\_vector](#)) take the dynamics of the scan head into account.

- The maximum repetition rate for "freely definable wobbel shapes" corresponds to the maximum wobble frequency estimated by the rule of thumb 1, [page 220](#).
- The sum of the path (trajectory) velocity and the velocity on the freely defined wobble figure should not exceed the maximum positioning velocity.
- The acceleration should not significantly exceed the value of  $2.5 \text{ V/S}$  (where  $V =$  typical positioning speed and  $S =$  tracking error).
- Also with "freely definable wobbel shapes", observe the heating of the scan system for freely definable wobble figures, see safety notice on [page 220](#) and rule of thumb 3.

## 8.5 Controlling 2D Scan Systems and 3D Scan Systems

### 8.5.1 2D Scan Systems

Up to two 2D scan systems can be controlled by one RTC6 PCIe Board.

The second one requires the [Option "Second Scan Head Control", page 30](#).

When controlling two 2D scan systems, a separate image field correction can be carried out for each of the both scan systems.

By [select\\_cor\\_table](#) or [select\\_cor\\_table\\_list](#), the two correction tables are assigned to the respective scan head connector, see also [Section "2. SCANHEAD Socket Connector \(Connector for Second Scan Head\)", page 57](#).

#### To use two correction tables in a double scan system configuration

(1) Load each of the desired 2D correction files by

```
load_correction_file(Name, n, 2)
(n = 1...8).
```

See also [Chapter 8.5.3 "Using Several Correction Tables", page 226](#).

(2) Assign a loaded 2D-correction table to the first and the second scan head each by calling the command [select\\_cor\\_table\(HeadA, HeadB\)](#) with ([HeadA](#) and [HeadB](#) = 1...8).

(3) By [set\\_offset](#), [set\\_scale](#), [set\\_angle](#) or the corresponding list commands, specify offset, scale factor and rotation to align the two image fields precisely with respect to each other.

#### Notes

- If you are not using one of the scan head connectors: assign the correction table 0 to it.
- The default setting for [select\\_cor\\_table](#) and [select\\_cor\\_table\\_list](#) is (1,0):
  - For the first scan head, correction table #1 is used
  - At the second scan head, there are *no position outputs*
- The RTC6 PCIe Board returns to this setting after every [load\\_program\\_file](#). If a different setting is to be used, [select\\_cor\\_table](#) or [select\\_cor\\_table\\_list](#) must be called again.
- The scan head connectors *cannot* be simultaneously assigned:
  - two 3D correction tables
  - a 2D correction table and a 3D correction table

### 8.5.2 3D Scan Systems

An RTC6 PCIe Board can also be used to control a 3-axis scan system<sup>(1)</sup>. This requires the [Option "3D", page 30](#).

#### Intended Use

3-axis scan systems can be used for positioning the laser focus within a flat processing field without the need for a flat field objective. Therefore, they are frequently used in applications for which flat field objectives are not available.

3-axis scan systems can also be used as 3D beam deflection systems. Here, the laser focus is guided along the contour of the workpiece being processed, thus enabling workpiece processing in 3 dimensions.

SCANLAB offers dynamic focusing units<sup>(2)</sup> that extend xy scan systems to 3-axis scan systems.

(1) Consisting of an xy scan system and a z axis dynamic focusing unit (see footnote 2) as z axis.

(2) See glossary entry on [page 879](#).

## Connection and Initialization

In order to control a 3-axis scan system by an RTC6 PCIe Board:

- The [Option "3D", page 30](#) of the RTC6 PCIe Board must be enabled (this can be checked by `get_RTC_version`)
- The xy scan system must be connected to the first scan head connector
- The z axis must be connected to the second scan head connector
- A 3D correction table (`D3_*.CT5`) must exclusively be assigned to the first scan head connector (by `select_cor_table` or `select_cor_table_list`).
- If, in addition, the [Option "Second Scan Head Control", page 30](#), is enabled, it is alternatively possible to connect:
  - the xy scan head to the second scan head connector
  - the z axis to the first scan head connector
  - In this constellation the 3D correction table must be assigned to the second scan head connector.

See also [Chapter 4.5 "Interfaces to Scan System", page 56](#) and [Section "2D and 3D Correction Files", page 163](#).

Other than that, no additional drivers or software files are needed for controlling a 3-axis scan system. Even initialization and program launching remain unchanged, see [Chapter 6.2 "Initialization and Program Start-Up", page 85](#).

The standard [RTC6 DLL \(RTC6DLL.dll\)](#) supports all commands for controlling 3-axis scan systems.

However, the full functionality of the commands – the actual *output* of z coordinates – is only available with enabled [Option "3D", page 30](#).

Several 3-axis scan systems can be simultaneously controlled (by multi-board commands) from a single PC. This requires a corresponding number of RTC6 PCIe Boards with enabled [Option "3D"](#)s to be installed in that PC.

## 3D Commands

These are, for example, the following RTC6 commands:

- `[*]3d[*]`
- `goto_xyz`
- `set_offset_xyz` and `set_offset_xyz_list`

Except for the additional motion in the third dimension, the 3D commands function identically to their corresponding 2D commands:

- Specified vectors and arcs are split-up into microsteps, see [Chapter 7.1.2 "Microstepping", page 129](#)
- The jump speed and mark speed are specified by `set_jump_speed` and `set_mark_speed`, see [Chapter 7.1.1 "Marking with Vector and Arc Commands", page 125](#). As for timed vector commands, the speeds are automatically determined from the specified jump or marking duration
- 3D image field correction is applied in accordance with the 3D correction table assigned by `select_cor_table` or `select_cor_table_list`, see [Chapter 7.3.5 "Image Field Correction and Correction Tables", page 162](#)
- For jump commands and `[*]mark[*]` commands, the laser control signals are switched on and off while taking delay settings into account, see [Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control", page 134](#)

For the value range of the data, see [Section "Compatibility Modes", page 157](#).

The calibration factor  $K_{xy}$  can be read out from the correction table used by `get_table_para`.

In [RTC4 Compatibility Mode](#) and [RTC5 Compatibility Mode](#), the three coordinate values are always scaled up to 20-bit values internally, so that the three spatial directions are treated equally with regard to the jump speed or mark speed.

The 3 coordinate values are internally always scaled up to 20-bit values, so that the three spatial directions are treated equally with regard to the jump or mark speed.

With big z jumps, observe the different dynamics of varioSCAN and 2D scan systems.



## Notes

- After “vector-controlled laser control” has been activated by **set\_vector\_control** (`Ctrl = 7`), the para-mark and para-jump commands can be used to set a dynamic focus shift linearly along the mark or jump vector, see [Section “Vector-Defined Laser Control”, page 195](#). See also **set\_defocus** or **set\_defocus\_list**.
- The size of the usable working field<sup>(1)</sup> and the focus shift in the z direction<sup>(2)</sup> (height of the [usable 3D image field](#)) can be obtained from the `Readme.txt`-file supplied with the 3D correction file as well as from the user manual of the 3-axis scan system/varioSCAN (“Technical Specifications” chapter).
- If a Z axis is to be used to hold the laser focus only in a certain plane (especially in 3D systems without a lens), then even 2D vector commands can be used. 2D vector commands leave the z position previously set with a 3D vector command unchanged and only adjust the focus length.
- If the [Option “3D”, page 30](#) is not enabled or no 3D correction table has been assigned:
  - The split-up into microsteps is calculated including the z axis (which influences the effective jump speed and mark speed in the xy plane).
  - There is merely no output for the Z axis

(1) Line “Max. Field Size (z=0): nnn.nnn mm”.

(2) Line “Max. Z-Range: +/- nn.n mm”.



## Adjusting Tracking Errors

If the xy scan head and the **Dynamic focusing unit** unit of a 3D scan system have different **Tracking Errors**, **set\_timelag\_compensation** can be used to *delay* the output of the faster component to that of the slower component.

The laser signal output is synchronized with the slower component. This allows, for example, Sky Writing with 3D commands without permanent defocusing.

In case of a **SCANahead system**, the `PreviewTime` parameter from **set\_scanahead\_params** is automatically used as "**Tracking Error**".

## Enhanced 3D Correction

The image size of 3D scan systems without objectives or with non-telecentric F-Theta objectives depends on its distance to the scan head objective (z coordinate).

The image field size typically gets stretched or squeezed linearly with the Z value. Therefore, SCANLAB 3D correction files include stretch correction factors to compensate for this effect, see **Section "ct5 Correction File Header", page 167**.

With some objectives, the typical stretching is combined with a change in the image geometry. Then additional stretch corrections are necessary which compensate the image geometry changes xy position-dependent but linear in z (2D stretch correction table). The stretch correction values are meaning the Z gradient for the location deviation at this point.

Assumption: for any chosen non-zero Z plane, (X, Y) is the desired position and (X', Y') the measured position. The corrections `<StretchX>` and `<StretchY>` are then calculated as follows:

$$\begin{aligned}\text{<StretchX>} &= (X-X')/Z \\ \text{<StretchY>} &= (Y-Y')/Z\end{aligned}$$

You can then use **load\_stretch\_table** to load the enhanced 3D correction onto the RTC6 PCIe Board from an ASCII text file and assign it to an already loaded 3D-correction table. This ASCII text file must contain corrections for a complete rectangular grid. The number of gridlines and their spacings can be freely defined and even differ in X and Y. If the absolute values of the corrections exceed 0.03125, then they are clipped to this limit value. The corrections are bilinearly interpolated for data points within the specified grid and linearly extrapolated for data points outside the grid.

Enhanced 3D correction is not active by default after **load\_program\_file**. It becomes active upon loading a valid table onto the RTC6 PCIe Board. It can be deactivated by calling **load\_stretch\_table** using a null pointer instead of the filename.

For the 2D stretch correction tables, the following rules apply:

- The ASCII text file can contain one or several tables.<sup>(1)</sup>
- The 2D stretch correction table must begin with the line:  
`[StretchTable<No>]`  
 <No> represents the table number to be specified by **load\_stretch\_table**.
- This is directly followed by a block of data points.
- If several tables with the same number exist, then only data from the first encountered table is read. The others are ignored.
- Reading of the text data terminates upon end of file or upon a line containing a caption.
- All characters to the right of a semicolon are treated as comments and ignored.
- The order of data points is up to you.
- The maximum length of a data line is 255 characters.
- A data line contains two position coordinates (in bits, signed integers) and two correction values (dimensionless, signed floating point numbers, use the period (.) or comma as the decimal separator), each separated by spaces or tabs:  
`<Xpos> <Ypos> <StretchX> <StretchY>`
- If a data point reoccurs, then the most recently read value is used.
- Empty lines or incomplete data lines are invalid and are ignored.

### 8.5.3 Using Several Correction Tables

The RTC6 PCIe Board memory can store 8 different correction tables at the same time.

It can be useful to work with several different correction tables even if only a single scan system (2D or 3D) is used. Example: a pilot laser and a working laser with different wavelengths are used.

Special applications sometimes also require different correction tables in quick succession. This change can be done, for example, by `select_cor_table( n, 0 )` or `select_cor_table_list( n, 0 )`  
`(n = 1...number_of_correction_tables)`.

**number\_of\_correction\_tables** serves to protect other commands (for example, **load\_correction\_file** and **select\_cor\_table**) from unwanted table numbers.

There is no need to change existing user programs except when user input is to be rejected in the future (by means of explicit RTC6 error messages).

(1) Even of another type, see table 1, page 142.

## 8.6 Processing-on-the-fly

- "Processing-on-the-fly" means the combination of workpiece movement and scan system movement.
- The use of the *Processing-on-the-fly* functionality requires the [Option Processing-on-the-fly](#).
- Whether the [Option Processing-on-the-fly](#) is enabled can be checked by [get\\_RTC\\_version](#).

### 8.6.1 Intended Use and Initialization

With its [Option Processing-on-the-fly](#) enabled, the RTC6 PCIe Board allows processing of parts in motion (for example, parts on a conveyor belt, rotating plate or xy positioning stage), as well as stationary parts with a moving scan system (for example, by a robot arm).

To adjust laser scan processes to the current workpiece position relative to the scan system, the position of the workpiece or scan system can be forwarded to the RTC6 PCIe Board:

- indirectly by encoder counters
- directly by the [McBSP interface](#)

If the Processing-on-the-fly correction is active, the coordinate values of all vector commands and arc commands are transformed based on the forwarded position values.

Upon forwarding the movement as encoder pulses, RTC6-internal encoder counters are triggered. The counter values correspond to the current position. They get a scaling factor (from certain RTC6 commands) assigned and are then used as *Processing-on-the-fly* correction.

See also [Chapter 9.3.3 "Synchronization by Encoder Signals"](#), page 284.

Upon forwarding the position values via the [McBSP interface](#), the input values get a scaling factor assigned and are then used as *Processing-on-the-fly* correction.

See also [Chapter 9.3.4 "Synchronization and Online Positioning by McBSP Signals"](#), page 286.

Simultaneous usage of both forwarding methods for Processing-on-the-fly correction of two independent motions is not possible, see [Section "Overview", page 227](#).

The [McBSP interface](#) cannot be simultaneously used for a Processing-on-the-fly application and an [Online Positioning](#), see [Notes, page 216](#).

Processing-on-the-fly correction is activated and deactivated by list commands. The parameters required for activation may have to be determined beforehand by a calibration process (see below).

If both scan head connectors each have a 2D correction table assigned, then a Processing-on-the-fly correction has the same effect at both scan head connectors.

For the z axis, a Processing-on-the-fly correction can be activated as well, see [Chapter 8.6.11 "Processing-on-the-fly Correction for the z Axis", page 243](#).

### Overview

The following encoder-based Processing-on-the-fly corrections of motions are available:

- [`set\_fly\_x`](#), [`set\_fly\_y`](#), even in combination  
To compensate linear motions of the workpiece (for example, by conveyor belt or xy positioning stage), see [Chapter 8.6.2 "Compensation of Linear Movements", page 228](#).
- [`set\_fly\_rot`](#)  
To compensate rotary motions of the workpiece (for example, by rotating plate), see [Chapter 8.6.3 "Compensation of Rotary Movements", page 232](#).
- [`set\_fly\_2d`](#)  
To compensate 2D motions of the workpiece (for example, xy positioning stage), see [Chapter 8.6.4 "Compensating 2D Motions", page 234](#).

The following McBSP-based Processing-on-the-fly corrections of motions are available:

- **`set_fly_x_pos`, `set_fly_y_pos`** even in combination  
To compensate 1D or 2D motions of the scan system (for example, robot arms), see [Chapter 8.6.2 "Compensation of Linear Movements", page 228](#).
- **`set_fly_rot_pos`**  
To compensate rotary motions of the scan system (for example, rotary table), see [Chapter 8.6.3 "Compensation of Rotary Movements", page 232](#).
- **`set_mcbsp_in`, `set_mcbsp_in_list`**  
To compensate 1D or 2D motions or rotary motions (for example, robot arms, rotary table), see [Chapter 8.6.2 "Compensation of Linear Movements", page 228](#) and [Chapter 8.6.3 "Compensation of Rotary Movements", page 232](#).

The following Processing-on-the-fly corrections can be combined:

- `set_fly_x` with `set_fly_y`.
- `set_fly_x_pos` with `set_fly_y_pos`
- For the special case `set_fly_z`, see [Chapter 8.6.11 "Processing-on-the-fly Correction for the z Axis", page 243](#)
- For linear 3D Processing-on-the-fly corrections with positional values, see [Section "Correction via McBSP Interface with Enhanced McBSP Input", page 231](#).

The following Processing-on-the-fly corrections *cannot* be combined:

- Encoder-based with McBSP-based
- linear and rotating

The last command always determines the overall correction unless it can be combined with previous settings.

However, mutual synchronization of any Processing-on-the-fly corrections by `wait_for_encoder_mode`, `wait_for_encoder_in_range` and `wait_for_mcbsp` is possible, see [Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237](#).

## 8.6.2 Compensation of Linear Movements

Processing-on-the-fly correction for linear movements (translations) can be activated<sup>(1)</sup> by:

- `set_fly_x` and/or `set_fly_y`
- `set_fly_x_pos` and/or `set_fly_y_pos`
- `set_mcbsp_in` (Mode = 1...3)
- `set_mcbsp_in_list` (Mode = 1...3)

A scaling factor must thereby be specified in each case.

With `set_multi_mcbsp_in` or `set_multi_mcbsp_in_list`, you can activate additional Processing-on-the-fly correction with positional values for linear motion in all three coordinate directions without bit-resolution restrictions. No scaling factor is required.

Processing-on-the-fly correction can be deactivated (simultaneously for both directions) by `fly_return`. See the corresponding notes in [Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236](#).

### Correction via Encoder Counters

To be able to pass position values via the board-internal encoder counters, the Processing-on-the-fly correction must be activated by:

- `set_fly_x` and/or `set_fly_y`

Here, the scaling factor [in bits per count] defines the relation between the shift [in bits] of the current output position in the image field and one counter pulse (count) of the corresponding encoder counter. See also [Section "Determining the Scaling Factors", page 229](#).

By `set_fly_x`, the encoder counter "Encoder0" is reset to zero. By `set_fly_y` the encoder counter "Encoder1" is reset to zero.<sup>(2)</sup>

(1) Different Processing-on-the-fly corrections cannot be combined arbitrarily, see the [Section "Overview", page 227](#).

(2) By `set_control_mode` Bit #9 it can be set beforehand whether the counter is reset at the respective Processing-on-the-fly command or at the start trigger.

Thereafter, the output value is calculated from the current output position by adding (for all spatial directions) the product of the scaling factor and the current counter value.

This correction takes place every 10 µs.<sup>(1)</sup>

#### Notes

- `set_fly_x` and `set_fly_y` can be used in combination for 1D as well as for 2D Processing-on-the-fly applications (for example, an xy positioning stage), particularly for separate or separable marking tasks in the real image field.
- For continuous marking in the virtual image field, SCANLAB recommends to preferably use `set_fly_2d`, see [Chapter 8.6.4 "Compensating 2D Motions"](#), page 234.

#### Determining the Scaling Factors

The scaling factors can be determined experimentally:

- (1) Read out the counter start value by `get_encoder`.
- (2) Start the movement.
- (3) Stop the movement.
- (4) Read out the counter end value by `get_encoder`<sup>(2)</sup>.
- (5) Measure the distance travelled in mm.
- (6) Calculate the encoder increment i as follows:

$$i = \frac{(\text{counter end value} - \text{counter start value})}{\text{distance travelled}}$$

- (7) Calculate the scaling factors Scalex and Scaley [in bits per count] as follows:

$$\text{Scalex} = \frac{K}{i_x}$$

$$\text{Scaley} = \frac{K}{i_y}$$

Whereby K is the calibration factor [in bits per mm], see [Chapter 7.3.2 "Image Field Size and Image Field Calibration"](#), page 156 and i is the encoder increment from Step 6.

If the workpiece moves at a constant speed  $v_x$  or  $v_y$  [in mm per second] and an encoder simulation was activated by `simulate_encoder`, then the scaling factors are calculated as follows:

$$\text{Scalex} = \frac{K \times v_x}{(1,000,000 \text{ counts / s})}$$

$$\text{Scaley} = \frac{K \times v_y}{(1,000,000 \text{ counts / s})}$$

- (8) Adjust the signs of the scaling factors according to the movement direction.

- (1) The encoder counters are signed 32-bit counters with overflow (= after reaching the maximum (minimum) counter value, counting continues at the minimum (maximum) counter value).
- (2) Alternatively, the counter start and end values can be stored in a cache on the RTC6 PCIe Board by the list command `store_encoder` and then retrieved from there by the control command `read_encoder`.

## Correction via McBSP Interface

To be able to pass position values via the **McBSP interface**, the Processing-on-the-fly correction must be activated by:

- **set\_fly\_x\_pos** and/or **set\_fly\_y\_pos**
- **set\_mcbsp\_in** for positional values in 2 spatial directions
- **set\_multi\_mcbsp\_in** for positional values in all 3 spatial directions (requires no scaling factors)

Here, the scaling factor [in bits per bits] defines the relation between the shift [in bits] of the current output position in the image field and the input value [in bits] at the **McBSP interface**. See also [Section "Determining the Scaling Factors", page 230](#).

Thereafter, the output value is calculated from the current output position by adding (for all spatial directions) the product of the scaling factor and the current input value at the **McBSP interface**.

This correction takes place every 10 µs.

At the **McBSP interface**, see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#), there are available:

- For 1D corrections – signed 32-bit values
- For 2D corrections – signed 16 bits per axis  
(the y value is in the lower 16 bits and the y value in the upper 16 bits of the value at the **McBSP interface**)

### Notes

- After **set\_fly\_x\_pos** and **set\_fly\_y\_pos**, the input values received at the **McBSP interface** automatically get copied to internal memory location 0. This still applies even after Processing-on-the-fly correction gets switched off by **fly\_return**, **set\_fly\_x\_pos**, **set\_fly\_y\_pos** or **set\_fly\_rot\_pos**, as well as after **load\_program\_file**. The current data at memory location 0 can be queried by **read\_mcbsp(0)**.

- In contrast, activation of Processing-on-the-fly correction by **set\_mcbsp\_in** or **set\_mcbsp\_in\_list** also result in McBSP input values being copied to internal memory locations 1, 2 and/or 3 (see [Section "Correction via McBSP Interface with Additional McBSP Input", page 231](#)).
- After Processing-on-the-fly correction is activated by **set\_multi\_mcbsp\_in** or **set\_multi\_mcbsp\_in\_list**, the transmitted data gets consecutively written to memory locations 0...3. From there, they can be read out unsorted by **read\_mcbsp** or sorted by **read\_multi\_mcbsp**.

### Determining the Scaling Factors

The scaling factors can be determined experimentally:

- (1) Read out the start input value by **read\_mcbsp**.
- (2) Start the movement.
- (3) Stop the movement.
- (4) Read out the end input value by **read\_mcbsp**.
- (5) Measure the distance travelled in mm.
- (6) Calculate the position increment i as follows:

$$i = \frac{\text{end input value} - \text{start input value}}{\text{distance travelled}}$$

- (7) Calculate the scaling factors Scalex and Scaley [in bits per bits] as follows:

$$\text{Scalex} = \frac{K}{i_x}$$

$$\text{Scaley} = \frac{K}{i_y}$$

Whereby K is the calibration factor [in bits per mm], see [Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156](#) and i is the encoder increment from Step 6.



## Correction via McBSP Interface with Additional McBSP Input

To activate Processing-on-the-fly correction for linear motions using **McBSP** input values (as an alternative to `set_fly_x_pos` or `set_fly_y_pos`), you can also call `set_mcbsp_in` or `set_mcbsp_in_list` (`Mode = 1...3`).

These commands offer the advantage of using the **McBSP interface** to input additional desired signals that should not be subjected to Processing-on-the-fly correction even when it is activated.

For this, all **McBSP** input values must be coded by Bit #31.

- Bit #31 = 0: The input value gets copied to internal memory location 0 and is applied for Processing-on-the-fly correction.
- Bit #31 = 1: The input value gets copied to internal memory location 3 but is not applied for Processing-on-the-fly correction.

### Notes

- All input values always get copied alternatingly to internal memory locations 1 and 2 and subsequently, in accordance with their Bit #31 coding, to internal memory locations 0 and 3. But after deactivation of correction by `set_mcbsp_in(0)` or `set_mcbsp_in_list(0)`, they only get copied to memory locations 1 and 2.
- You can query the data currently stored at internal memory locations 0 - 3 by `read_mcbsp`.
- A scaling factor is specified at `set_mcbsp_in` or `set_mcbsp_in_list`. For 2D corrections (`Mode = 3`), the scaling factor applies to both axes simultaneously. Calibration for determining the scaling factor is the same as for `set_fly_x_pos` and `set_fly_y_pos` (see above).
- For transferring 1D correction values, 31 bits with sign are available (Bit #31 is reserved as the coding bit). In contrast, only 15 bits with sign are available per axis for transferring 2D correction values, whereby the x value lies in the lower 16 bits and the y value in the upper 16 bits of the value at the **McBSP interface**. For a description of the interface, see Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73.

## Correction via McBSP Interface with Enhanced McBSP Input

As an alternative to the previous chapter's methods, you can also activate Processing-on-the-fly correction of linear motion with `set_multi_mcbsp_in` or `set_multi_mcbsp_in_list`.

These commands have an advantage over `set_mcbsp_in` or `set_mcbsp_in_list` in that they offer Processing-on-the-fly correction not only in the x and y directions, but also in the z direction and with laser power variation. Furthermore, up to four additional signals can be transmitted for usage as you wish.

Each 10 µs, data asynchronously transmitted to **McBSP** memory locations 0 through 3 gets sorted and copied to an additional internal memory location. From there it is available for final usage and can be read-out sorted by signal types using `read_multi_mcbsp`.

### 8.6.3 Compensation of Rotary Movements

Before activating Processing-on-the-fly correction for rotary xy movement, you must define the rotation center by `set_rot_center` or `set_rot_center_list`. The rotation center may also be situated outside the image field.

The Processing-on-the-fly correction itself can be activated by:

- `set_fly_rot`
- `set_fly_rot_pos`
- `set_mcbsp_in` (Mode = 4)
- `set_mcbsp_in_list` (Mode = 4)

Processing-on-the-fly correction can be stopped by `fly_return` (see the corresponding notes on Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236).

#### Notes

- A Processing-on-the-fly rotation correction:
  - Cannot be combined with other Processing-on-the-fly corrections, see [Section "Overview", page 227](#).
  - Simultaneously for both scan head connectors is only practical, if both attached scan systems are aligned to exactly the same rotation center.

#### Correction via Encoder Counter

To be able to pass rotation angles via the board-internal incremental encoder, the ENCODER X input<sup>(1)</sup> must be connected, see also [Section "Inputs for External Encoder Signals", page 285](#).

Then, Processing-on-the-fly correction must be activated by `set_fly_rot`. The parameter `Resolution` [in counts per revolution] must thereby be specified.

See also [Section "Determining the Resolution Parameter", page 232](#).

`set_fly_rot` resets the encoder counter "Encoder0" to zero.<sup>(2)</sup>

The output value is calculated from the current output position via a rotation matrix around the specified rotation center with rotation angle /  $360^\circ = \text{current "Encoder0" counter reading} / \text{Resolution}$ ).

This correction is performed every  $10 \mu\text{s}$ .<sup>(3)</sup>

#### Determining the Resolution Parameter

The `Resolution` parameter can be determined experimentally:

- (1) Read out the counter start value by `get_encoder`.
- (2) Start the rotation.
- (3) Count the number of revolutions.
- (4) Stop the rotation.
- (5) Read out the counter end value by `get_encoder`.<sup>(4)</sup>
- (6) Calculate the parameter `Resolution` [in counts per revolution] as follows:

$$\text{Resolution} = \frac{(\text{counter end value} - \text{counter start value})}{\text{number of revolutions}}$$

$$\text{Resolution} = (\text{counter end value} - \text{counter start value}) / \text{number of revolutions}$$

If the workpiece rotates at a constant speed  $\omega$  [in number of revolutions per second] and an encoder simulation was activated by `simulate_encoder`, then the `Resolution` parameter can be calculated as follows:

$$\text{Resolution} = \frac{(1.000.000 \text{ counts} / \text{s})}{\omega}$$

- (7) Adjust the sign of `Resolution` to the direction of rotary movement.

(1) ENCODER X = Encoder0.

(2) By `set_control_mode` Bit #9 it can be set beforehand whether the counter is reset at the respective Processing-on-the-fly command or at the start trigger.

(3) The encoder counters are signed 32-bit counters with overflow (= after reaching the maximum (minimum) counter value, counting continues at the minimum (maximum) counter value).

(4) Alternatively, the counter start and end values can be stored in a cache on the RTC6 PCIe Board by the list command `store_encoder` and then retrieved from there by the control command `read_encoder`.



### Correction via McBSP Interface

If angle-position values for Processing-on-the-fly correction of rotary movement are forwarded by the **McBSP interface**, then Processing-on-the-fly correction must be activated by `set_fly_rot_pos`.

The required `Resolution` parameter has the same meaning as with `set_fly_rot` and is similarly determined, see [Section "Determining the Resolution Parameter"](#), page 232.

McBSP input values are read out by `read_mcbsp`.

#### Notes

- McBSP input values get copied to board-internal memory location 0, see notes in [Section "Correction via McBSP Interface"](#), page 230.

### Correction via McBSP Interface with Additional McBSP Input

A Processing-on-the-fly correction for rotary movements with McBSP input values can be activated by:

- `set_mcbsp_in` (Mode = 4)
- `set_mcbsp_in_list` (Mode = 4)

These commands offer the advantage of using the **McBSP interface** to input additional desired signals that should not be subjected to Processing-on-the-fly correction even when it is activated.

For this, all **McBSP** input values must be coded by Bit #31.

#### Notes

- All input values always get copied alternatingly to internal memory locations 1 and 2 and subsequently, in accordance with their Bit #31 coding, to internal memory locations 0 and 3. But after deactivation of correction by `set_mcbsp_in(0)` or `set_mcbsp_in_list(0)`, they only get copied to memory locations 1 and 2.
- You can query the data currently stored at internal memory locations 0 - 3 by `read_mcbsp`.
- By `set_mcbsp_in` or `set_mcbsp_in_list`, a rotation resolution can be specified. Calibration for determining the rotation resolution is the same as for `set_fly_rot_pos`, see [Section "Determining the Resolution Parameter"](#), page 232.
- For transferring rotary correction values, 31 bits with sign are effectively available.

#### 8.6.4 Compensating 2D Motions

You can use the `set_fly_2d` command to activate encoder-based 2D Processing-on-the-fly correction (for example, for a xy positioning stage), particularly for continuous marking in the virtual image field.

Compared to an activation by `set_fly_x` and `set_fly_y`, this has the following advantages:

- `set_fly_2d` simultaneously resets both encoders (whereas the separate commands `set_fly_x` and `set_fly_y` do so with a slight temporal offset between both channels)
- `set_fly_2d` allows compensation of non-linear relations between encoder values and actual xy positioning stage motions, see [Section "2D Encoder Compensation for xy Positioning Stages", page 234](#)
- Even during an interruption of a `set_fly_2d` marking by `wait_for_encoder_mode` or `wait_for_encoder_in_range`, the galvanometer scanner positions receive continuous Processing-on-the-fly correction in accordance with the current xy positioning stage encoder values (whereby the laser focus remains stationary relative to the xy positioning stage). Thus, unnecessary jumps are avoided after xy positioning stage forwarding motions, see [Chapter 8.6.6 "Virtual Image Field with Processing-on-the-fly", page 237](#)

##### Notes

- A `set_fly_2d` correction cannot be combined with other Processing-on-the-fly corrections, see [Section "Overview", page 227](#).
- Coordinate transformations within the virtual image field, see [Section "Coordinate Transformations in the Virtual Image Field", page 158](#) can be activated when starting a [Processing-on-the-fly session](#) with changed values, if they have been loaded before with the corresponding control commands.

#### 2D Encoder Compensation for xy Positioning Stages

For particularly demanding marking requirements, it might be necessary to also compensate mechanical deviations of a xy positioning stage.

For this purpose, `load_fly_2d_table` can be used to load up to two 2D compensation tables, see [page 235](#), onto the RTC6 PCIe Board.

Then – during a Processing-on-the-fly application initiated by `set_fly_2d` – encoder values are 2D interpolated and compensated just like with field correction tables.

To avoid unnecessary initialization motions, you can use `init_fly_2d` to define a desired positioning stage start position as the reference value for 2D encoder compensation (and to store it on the RTC6 PCIe Board) and to select simultaneously one of the both correction tables.

Upon every subsequent encoder reset by `set_fly_2d`, the current position is automatically applied as the new reference position, so that the relation between current encoder values and the absolute position of the positioning stage is retained for compensation. This relation remains even upon termination with `fly_return` or upon a new start with `set_fly_2d`.

This relation does not remain, if you meanwhile activate other Processing-on-the-fly corrections or reset the encoders by an external /START (see `set_control_mode` (`Bit #9 = 1`)).

At any time, you can query the currently valid reference values by `get_fly_2d_offset`.

Encoder compensation is applied exclusively to Processing-on-the-fly correction. The original uncompensated encoder values use:

- `get_encoder`
- `store_encoder`
- `read_encoder`
- `wait_for_encoder`
- `wait_for_encoder_mode`
- `wait_for_encoder_in_range`
- Recording by `set_trigger`  
(`Signal1/Signal2 = 43 or 44`)
- Recording by `get_value`



For **load\_fly\_2d\_table**, you need to provide an ASCII text file that contains a 2D compensation table.

A 2D compensation table must have encoder compensations for reference points on a rectangular grid. The number of grid lines and their spacing is up to you (both may also differ in x and y). Missing reference point data is automatically assigned a compensation of 0. The largest occurring encoder reference points form a frame within which encoder compensation values are bilinearly interpolated. Encoder values outside this frame is clipped to this frame prior to interpolation. Therefore, the reference points should cover at least the range of the xy positioning stage required by your application. Reference points with values exceeding  $\pm 524288$  can result in some loss of precision.

After **load\_program\_file**, 2D encoder compensation is inactive by default.

It becomes active as soon as a valid 2D compensation table from an ASCII text file gets loaded onto the RTC6 PCIe Board. You can subsequently deactivate it by calling **load\_fly\_2d\_table** using a **NULL** pointer instead of the filename.

For the 2D compensation tables, the following rules apply:

- The ASCII text file can contain one or several tables.<sup>(1)</sup>
- The 2D compensation table must begin with the line:  
`[Fly2DTable<No>]`  
`<No>` stands for the table number `No`, which must be specified with **load\_fly\_2d\_table**. The first of the two tables is loaded with `No = <No>` and the second with `No = <No> + 65,536`.
- This is directly followed by a block of data points.
- If several tables with the same number exist, then only data from the first encountered table is read. The others are ignored.
- Reading of the ASCII text file terminates upon end of file or upon a line containing a caption.
- All characters to the right of a semicolon are treated as comments and ignored.
- The order of data points is up to you.
- The maximum length of a data line is 255 characters.
- A data line contains the two reference point coordinates for "Encoder0" and "Encoder1" (signed integers) and two compensation values (signed integers), each separated by spaces or tabs:  
`<Encoder0> <Encoder1> <Encoder0-delta>`  
`<Encoder1-delta>`
- If a data point reoccurs, then the most recently read value is used; the others are ignored.
- Empty lines or incomplete data lines are invalid and are ignored.

(1) Even of another type, see table 1, page 142.

### 8.6.5 Deactivating Processing-on-the-fly Correction

All Processing-on-the-fly corrections can be deactivated (simultaneously for both spatial directions) by **`fly_return`**.

**`fly_return`** requires a new output position to be specified, which then is reached by a normal jump.

Processing-on-the-fly correction that has been activated by **`set_multi_mcbsp_in`** should be terminated by **`fly_return_z`**.

In all other cases (see below) a “Hard jump” to a new output position may occur.

#### Notes

- If Processing-on-the-fly correction is not explicitly deactivated<sup>(1)</sup>, then:
  - it is also effective during execution of subsequent lists
  - it is not effective in the pause between two lists<sup>(2)</sup>
- Processing-on-the-fly correction enabled by **`set_fly_x`, `set_fly_y`, `set_fly_2d`, `set_fly_rot`, `set_fly_x_pos`, `set_fly_y_pos` or `set_fly_rot_pos`** also gets deactivated if the same command is called again but with invalid parameter values. This could lead to a jump to an uncorrected output position (also refer to the command descriptions).
- If Processing-on-the-fly was activated by **`set_fly_x`**, then **`set_fly_y`** does not deactivate it and vice versa. The same applies to **`set_fly_x_pos`** and **`set_fly_y_pos`**. Other than that, every Processing-on-the-fly command automatically deactivates any Processing-on-the-fly correction activated by another Processing-on-the-fly command, see also [Section “Overview”, page 227](#). This could lead to a “Hard jump” to a new output position.

- Processing-on-the-fly correction activated by **`set_mcbsp_in`** or **`set_mcbsp_in_list`** also gets deactivated if you call **`set_mcbsp_in`** with **`Mode = 0`** or **`set_mcbsp_in_list`** with **`Mode = 0`**. This could lead to a “Hard jump” to an uncorrected output position
- Processing-on-the-fly correction activated by **`set_fly_x_pos`, `set_fly_y_pos`, `set_fly_rot_pos`, `set_mcbsp_in` or `set_mcbsp_in_list`** also gets deactivated if you call the command for configuring [Online Positioning](#), see [Section “Notes”, page 216](#). This could lead to a “Hard jump” to a new output position.
- A deactivation of a Processing-on-the-fly correction occurs only after the scanner delay.
- If Processing-on-the-fly was activated by **`set_mcbsp_in`** or **`set_mcbsp_in_list`**, then the **`fly_return`** command deactivates Processing-on-the-fly correction, but it does not deactivate copying to internal memory locations (just like with **`set_mcbsp_in(5)`**). To afterward also deactivate copying to internal memory locations, you can use **`set_mcbsp_in(0)`** or **`set_mcbsp_in_list(0)`**.
- Processing-on-the-fly correction also gets deactivated (for both spacial directions simultaneously) by **`stop_execution`** or an external stop.

(1) **`set_end_of_list`** does not deactivate Processing-on-the-fly correction.

(2) Therefore, the correction does not affect the control commands **`goto_xy`** and **`goto_xyz`**.

### 8.6.6 Virtual Image Field with Processing-on-the-fly

With Processing-on-the-fly applications, the full value range (29-bit) of the virtual image field can be used, see [Chapter 7.3.3 "Virtual Image Field", page 158](#) to load and process command lists with objects that are up to 512 times larger than the real 20-bit image field.

With 1D Processing-on-the-fly applications (for example, workpieces on a conveyor belt), objects can only exceed the real image field in the very dimension parallel to the Processing-on-the-fly direction.

With 2D Processing-on-the-fly applications (for example, with an xy positioning stage), the to-be-marked objects of a command list may be distributed across the entire virtual image field.

If an entire marking task consists of several partial markings ("tiles") whose extent does not exceed the real image field, the Processing-on-the-fly functionality can also only be used to move the partial marking to be marked into the real image field and then process it there.

Coordinate values which are still outside the real image *after* Processing-on-the-fly correction are clipped to the boundary values of the real image field. Here, the `get_marking_info` error bits get set automatically, see [Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections", page 240](#).

If there is a risk of the real 20-bit image field being exceeded during list execution, specifically a forwarding motion of the conveyor belt or xy positioning stage should be executed. The forwarding motion can be waited for with `wait_for_encoder`, `wait_for_encoder_mode`, `wait_for_encoder_in_range` or `wait_for_mcbsp` by interrupting the list execution until certain encoder values or McBSP values are reached, see [Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237](#).

The appropriate break up of the entire marking task and synchronization with the 1D or 2D Processing-on-the-fly movement is the task of the user program.

### 8.6.7 Synchronizing Processing-on-the-fly Applications

Processing of command lists can be started by either a RTC6 command or an external start signal, see [Chapter 6.4.4 "Starting and Stopping Lists", page 99](#).

If the command list is not to be started immediately with the start signal, then an appropriate delay may be implemented as follows:

- When transferring positions via the McBSP interface – by a `wait_for_mcbsp` at the beginning of the command list
- When transferring positions via encoder counters – by a `set_fly_x`/`set_fly_y`, `set_fly_2d` or `set_fly_rot` (to reset the counter(s)) and a subsequent `wait_for_encoder_mode` or `wait_for_encoder_in_range` at the beginning of the command list.

`wait_for_encoder_in_range` is useful for 2D encoder-based Processing-on-the-fly applications. `wait_for_encoder_in_range` waits until both encoders are within the specified range at the same time and thus does not depend on the actual trajectory that used to reach this range<sup>(1)</sup>. When transferring positions via encoder counters, a track delay can be configured (even independently of an active Processing-on-the-fly session) to delay the execution of the actual list start relative to the triggering start signal or RTC6 command, see [Section "External Start", page 276](#)

(1) If you would use `wait_for_encoder` or `wait_for_encoder_mode` instead, you would need to call these commands individually and consecutively for each encoder. Here, simultaneous fulfilment of both encoder criteria would depend on the xy positioning stage motion's explicit trajectory and you would need to define and reproduce an appropriate trajectory even before loading the lists.

External starts triggered by an external start signal (or by `simulate_ext_start` or `simulate_ext_start_ctrl`) that do not execute immediately because of the track delay setting are held in a queue that can accommodate up to 8 starts (each start trigger is automatically generated when the delay has expired, see also [Section "External Start", page 276](#)). This helps avoid dead time between the execution of multiple (Processing-on-the-fly) list programs.

Moreover, the list command `simulate_ext_start` can be used to trigger further list starts at defined intervals.

If `wait_for_encoder`, `wait_for_encoder_mode`, `wait_for_encoder_in_range` and `wait_for_mcbsp` are used to interrupt a list for intermediate forwarding motions, see [Chapter 8.6.6 "Virtual Image Field with Processing-on-the-fly", page 237](#), then the signals for "laser active" operation are not changed before the forwarding motion.

With encoder-based Processing-on-the-fly applications, the laser focus with `park_position` can be moved to a safe parking position before an interruption and back to the starting position or any other position after an interruption with `park_return`. See command description for more details.

The behavior of the galvanometer scanners during such a forwarding motion depends on the Processing-on-the-fly correction used:

- With McBSP-based Processing-on-the-fly correction as well as the encoder-based Processing-on-the-fly corrections `set_fly_x`, `set_fly_y` and `set_fly_rot`, the galvanometer scanners are stationary relative to the real image field during list interruption. As a rule, a jump to the next marking must then be made.
- With the encoder-based `set_fly_2d` Processing-on-the-fly correction, the positions of the galvanometer scanner are continuously Processing-on-the-fly-corrected (the laser focus thus remains stationary relative to the to-be-marked object). The subsequent jump to the next marking or continuation of the same is usually very small.
- With `set_fly_2d`, clipping might occur if the Processing-on-the-fly-corrected coordinate values exceed the real image field during a long forwarding motion. To avoid this, it might make sense to switch off Processing-on-the-fly correction before the forwarding motion (by `fly_return`, see [Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236](#); the galvanometer scanners then remain stationary). To switch correction back on after the forwarding motion, you can use `activate_fly_2d` instead of `set_fly_2d` (or `activate_fly_xy` instead of `set_fly_x` and `set_fly_y`). These commands do not reset the encoders, but instead convert the last Processing-on-the-fly-uncorrected coordinate values such that the Processing-on-the-fly-corrected output matches the current output. If an error occurs when restarting with these commands (for example, because the recalculated coordinate values would then be outside the 29-bit virtual image field, or because another Processing-on-the-fly mode has been activated in the meantime), then an error bit gets set. This error bit can be read out by `get_marking_info` (Bit #9 = 1).



If, during list processing, it is necessary to immediately respond to this error, then the list command **if\_not\_activated** can be called to possibly jump to an error-handling routine.

- **activate\_fly\_2d\_encoder** or **activate\_fly\_xy\_encoder** can be used to continue at another positioning stage position when switching on again. They reset the encoders and simulate a positioning stage position using the encoder values passed as parameters. This avoids larger forwarding motions for initialization.

## 8.6.8 Encoder Resets

Many encoder-based Processing-on-the-fly applications (for example, a conveyor belt continuously traveling in the same direction) need to occasionally reset the encoders (for example, before a new marking sequence). For this, you can integrate Processing-on-the-fly reactivations into your lists by **set\_fly\_x**, **set\_fly\_y**, **set\_fly\_rot** or **set\_fly\_2d**.

In encoder-based Processing-on-the-fly applications with continuous marking (for example, using an xy positioning stage in the virtual image field), however, the relationship between the encoder values and the absolute position of the xy positioning stage should usually be preserved.

For this purpose, **set\_fly\_2d** should be used for Processing-on-the-fly activation. Before a long interruption, you can deactivate Processing-on-the-fly correction with **fly\_return**. And after the interruption you can use **activate\_fly\_2d** or **activate\_fly\_2d\_encoder** to resume correction. See also Chapter 8.6.4 "Compensating 2D Motions", page 234 and Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237.

Processing-on-the-fly correction can also be resumed with **activate\_fly\_xy** or **activate\_fly\_xy\_encoder**, but some functions are no longer available, particularly those necessary for the relation between current encoder values and absolute xy positioning stage positions, see the Section "2D Encoder Compensation for xy Positioning Stages", page 234.

By **set\_control\_mode** Bit #9 it can be set beforehand whether the counter is reset at the respective Processing-on-the-fly command or at the start trigger.

### 8.6.9 Monitoring Processing-on-the-fly Corrections

If a Processing-on-the-fly user program is not optimized for the motion of the workpiece or scan system or if considerable unintended change of workpiece or scan system speed occurs during a Processing-on-the-fly operation, then the image field's boundaries might be reached. Because the RTC6 PCIe Board clips coordinate values at the boundaries to prevent unallowed values, this could cause some parts of the to-be-marked pattern to not be scanned.

To allow user programs to monitor Processing-on-the-fly applications, the RTC6 PCIe Board sets internal error bits (`Bit #0...Bit #3`) if the image field boundaries are exceeded (and coordinate values get clipped). These can be read out by `get_marking_info`.

To avoid boundary exceedance, you should repeatedly call `get_marking_info` during test runs or normal operation of your Processing-on-the-fly application and modify your user program accordingly.

#### Notes

- Error `Bit #0...Error Bit #3` can be used for determining which edge of the image field has been exceeded. Each boundary exceedance results in setting of the corresponding error bit.
- Fehler-`Bit #0...Fehler-Bit #3` are reset during initialization by `load_program_file` and by `get_marking_info`. Therefore, `get_marking_info` returns information about errors that occurred since the last initialization or the last call of `get_marking_info`. Subsequent transformations of type 5...9, see [Chapter 7.3.6 "Output Values to the Scan System", page 170](#), are not taken into account here.
- During the adjustment phase of a Processing-on-the-fly correction, coordinate points at the edge of the image field should therefore be approached and checked for possible limitations.
- Error `Bit #9` indicates if the 29-bit virtual image field range has been exceeded during resumption of Processing-on-the-fly correction by `activate_fly_2d` or `activate_fly_xy`, see [Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237](#).



## Customer-Defined Monitoring Area

For Processing-on-the-fly applications, the RTC6 PCIe Board also checks for exceedance of a second value range. Its boundaries can be specified by `set_fly_limits`.

Such exceedances likewise result in setting internal error bits (Error Bit #4...Error Bit #7). These can be read out by `get_marking_info`.

Moreover, the conditional commands

`if_fly_x_overflow`, `if_fly_y_overflow`,  
`if_not_fly_x_overflow` or `if_not_fly_y_overflow`  
allow execution of any list command to be made dependent upon whether boundary exceedance in a customer-defined monitoring area occurred or not.

If the condition specified in the command parameter to the error bits #4...#7 is fulfilled, the immediately following list command is executed. Otherwise it is skipped.

### Notes

- Boundary exceedance of a customer-defined monitoring area does not necessarily result in clipping of the output coordinate values. Clipping (and setting Error Bit #0...Error Bit #3) only occurs if the maximum image field (-524,288...+524,287 bit) is exceeded (the customer-defined monitoring area is typically smaller).
- Error Bit #4...Error Bit #7 are reset during initialization (by `load_program_file`), but *not* by `get_marking_info`. Individual error bits get implicitly reset by the conditional commands and can also be explicitly reset by `clear_fly_overflow` (see command description).
- Error Bit #4...Error Bit #7 do not take into account transformations of type 5...9, see [Chapter 7.3.6 "Output Values to the Scan System", page 170](#).
- Also for Rotation-fly applications it is possible to monitor a rectangular area, but not a rotation angle range.



### **8.6.10 Tracking Error Compensation of Encoder Values for Processing-on-the-fly Applications – NOT YET IMPLEMENTED**

*Not yet implemented.*

### 8.6.11 Processing-on-the-fly Correction for the z Axis

The encoder-based Processing-on-the-fly correction for the z axis (referred to as "FlyZ correction" in the following) can be activated by `set_fly_z`.

You can add FlyZ to Processing-on-the-fly correction for the x and y axes that had been activated by `set_fly_x/set_fly_y/set_fly_rot` or can be effective on its own.

This makes dynamic marking possible if the workpiece plane ( $z = 0$ ) and the scan system do not move parallel to each other.

With "FlyZ correction" activated, the z component workpiece motion gets compensated by re-adjustment of the z axis (siehe [Dynamic focusing unit](#)) in accordance with the current encoder signals.

#### Prerequisites

- Both, [Option Processing-on-the-fly](#) as well as the [Option "3D"](#) must be enabled.
- The desired marking must function properly when static (that is, without workpiece motion):
  - The user program must model workpiece skew by using suitable 3D vector commands.
  - The varioSCAN must be capable of tracking the xy galvanometer scanner motions of the scan system.
- For moving workpieces, the [Dynamic focusing unit](#) must also be capable of following workpiece motion. Take care to exceed neither the maximum focus range in z direction nor the depth of field of the objective. Ensure that the precision of the correction table is sufficient in terms of edge sharpness, stretching etc. When used with `set_fly_rot`, ensure that the rotation angle across the image field is small enough and the rotation center is sufficiently far away. Users are responsible for appropriate testing. SCANLAB provides no additional functionality for this. A virtual image field for the z coordinate does not exist.
- See also the operational prerequisites for 3-axis scan systems in [Section "Connection and Initialization", page 223](#).

#### Notes on Usage

- For general information on Processing-on-the-fly correction and determining scaling factors, see [Chapter 8.6 "Processing-on-the-fly", page 227](#).
- When activating "FlyZ correction" by `set_fly_z`, you can specify which of the two encoder counters should be used. Because `set_fly_x` already uses encoder counter "Encoder0" and `set_fly_y` uses encoder counter "Encoder1" (`set_fly_rot` uses "Encoder0"), you might need to use one of the two encoders twice.
- "FlyZ correction" can be applied together with `set_fly_x/set_fly_y/set_fly_rot`.
- "FlyZ correction" can be also be used alone (only encoder-based z variation).
- Activated "FlyZ correction" can be deactivated by `fly_return_z` or `fly_return`. As a result, all Processing-on-the-fly corrections for all three spatial directions are simultaneously deactivated. You can supply a command parameter for a new output position (only the x and y coordinates for `fly_return`, in addition the z coordinate for `fly_return_z`).
- Activated "FlyZ correction" can also be deactivated by `set_fly_z` in conjunction with an invalid parameter (for example, `set_fly_z( ScaleZ=0 )`).
  - If only z correction was activated here, then a jump to an uncorrected output position might occur.
  - If Processing-on-the-fly corrections were also activated for other spatial directions (by `set_fly_x/set_fly_y/set_fly_rot`), then these do not get deactivated here (and no jump to an uncorrected output position occurs).



- If correction for all three spatial directions is activated by `set_fly_x`, `set_fly_y` and `set_fly_z`, then a subsequent `set_fly_x( ScaleX=0 )` only deactivates X correction without affecting Y and Z correction (likewise for `set_fly_y( ScaleY=0 )`). But if only z correction (by `set_fly_z`) or 2D correction (by `set_fly_z` and `set_fly_x` or `set_fly_y`) is activated, then a subsequent `set_fly_x`, `set_fly_y` or `set_fly_rot` with invalid parameter value (for example, `set_fly_x( ScaleX=0 )`) also deactivates z correction. In the latter case, a jump to a (partially) uncorrected output position might occur.
- `set_fly_z` deactivates a Processing-on-the-fly correction with positional values via `McBSP`.
- Conversely, Processing-on-the-fly correction activated by `set_fly_z` gets deactivated by such a correction. Here, a “Hard jump” to a new output position might occur.
- To avoid “Hard jumps”, use only `fly_return_z` to deactivate Processing-on-the-fly correction.
- `get_marking_info` also provides information on possible range exceedances during “FlyZ correction” (Bit #22...Bit #25).
- `set_fly_limits_z`, `if_fly_z_overflow` and `if_not_fly_z_overflow` are available for defining a customer-specific monitoring range for “FlyZ correction” and for corresponding conditional command execution, see [Chapter 9.3.2 “Conditional Command Execution”, page 281](#). You can use `clear_fly_overflow` to reset the error bits (Bit #24...Bit #25 from `get_marking_info`) for customer-specific monitoring of FlyZ applications.
- A 3D Processing-on-the-fly correction with positional values via `McBSP` can be activated by `set_multi_mcbsp_in` and `set_multi_mcbsp_in_list`.

### 8.6.12 "Fly Extension" Commands

- In order to be able to use the Processing-on-the-fly functionality flexibly and generically, the "Fly Extension" Commands<sup>(1)</sup> are available in RTC6 Software Package ≥ V1.6.1 (DLL 617, OUT 617, RBF 623), see Table 2.

Table 2: Number of supported axes in "Fly Extension" Commands.

"Fly Extension" Command	Axes
<code>set_fly_1_axis</code>	1
<code>fly_return_1_axis</code>	1
<code>activate_fly_1_axis</code>	1
<code>park_position_1_axis</code>	1
<code>park_return_1_axis</code>	1
<code>wait_for_1_axis</code>	1
<code>set_fly_2_axes</code>	2
<code>fly_return_2_axes</code>	2
<code>activate_fly_2_axes</code>	2
<code>park_position_2_axes</code>	2
<code>park_return_2_axes</code>	2
<code>wait_for_2_axes</code>	2
<code>set_fly_3_axes</code>	3
<code>fly_return_3_axes</code>	3

- The Processing-on-the-fly functionality<sup>(2)</sup> itself remains unchanged. Only its control has been made more flexible.
- "Fly Extension" Commands are a flexible system of Processing-on-the-fly applications.
  - Most of them affect:
    - only 1, only 2 or all 3 spatial axes
    - alternatively 1 Rotary axis
  - At the same time the following can be freely assigned (Table 4)
    - Encoder counter "Encoder0" or "Encoder1"
    - and/or an McBSP input on each of these axes, even mixed

- "Fly Extension" Commands are in no case compatible with the "Classic" Processing-on-the-fly commands.
- A Processing-on-the-fly application with "Classic" Processing-on-the-fly commands must have been explicitly been terminated, before the functionality of one of the "Fly Extension" Commands can be activated, and vice versa.
- With "Fly Extension" Commands, axes are specified according to Table 3.

Table 3: Parameter Axis of the "Fly Extension" Commands.

Axis	Parameter value	Axis
1		x axis
2		y axis
3		z axis
4		Rotary axis

- With the "Fly Extension" Commands, linear axes 1, 2 and 3 cannot be combined simultaneously with a Rotary axis (4).
- "Fly Extension" Commands support 1, 2 or 3 Axes, see Table 2.

- Only list commands, no control commands.
- As described in Chapter 8.6.1 "Intended Use and Initialization", page 227...Chapter 8.6.11 "Processing-on-the-fly Correction for the z Axis", page 243.



## Notes on How to Use

### "Fly Extension" Commands

- `set_fly_1_axis( Axis, Mode, Scale )` replaces one of the "Classic" Processing-on-the-fly commands:
  - `set_fly_x`
  - `set_fly_y`
  - `set_fly_z`
  - `set_fly_rot`
  - `set_fly_x_pos`
  - `set_fly_y_pos`
  - `set_fly_rot_pos`
- `set_fly_2_axes` combines the activation of any 2 linear axes and replaces for example, even `set_fly_2d`.
- `fly_return_1_axis`, `fly_return_2_axes` and `fly_return_3_axes` extend the functionality of the "Classic" Processing-on-the-fly commands `fly_return` and `fly_return_z`.
- `activate_fly_2_axes` replaces the "Classic" Processing-on-the-fly commands
  - `activate_fly_2d`
  - `activate_fly_2d_encoder`
  - `activate_fly_xy`
  - `activate_fly_xy_encoder`
- `wait_for_1_axis` can "wait" for any encoder value or a value transmitted by McBSP, thus substituting `wait_for_encoder_mode` and `wait_for_mcbsp`.
- `wait_for_2_axes` stands for `wait_for_encoder_in_range` and `wait_for_encoder_in_range_mode` and are able to wait for McBSP values in addition.
- The laser control (switch off the laser or leave it unchanged) can be set individually at `wait_for_1_axis (wait_for_2_axes)` with parameter `LaserMode (LaserMode)`.

- The galvanometer scanner movement (stand still or move along) can be specified with `wait_for_1_axis (wait_for_2_axes)` by parameter `WaitMode (WaitMode)`. Thus, it is no longer bound to `set_fly_2d` and `set_fly_x/set_fly_y`.
- "Fly Extension" Commands are also compatible with McBSP transmissions from `set_mcbsp_in` and `set_multi_mcbsp_in`. The Processing-on-the-fly corrections can be subsequently overwritten, for example, by Encoder-based corrections. However, the McBSP transmission to the memory locations themselves cannot be changed.
- "Fly Extension" Commands are – like "Classic" Processing-on-the-fly commands<sup>(1)</sup> – not compatible with an Online Positioning.
- "Fly Extension" Commands are able to "wait" for McBSP values, see Table 4.
- Modes and Encoders supported with "Fly Extension" Commands are shown in Table 4.

(1) See Footnote on page 245.



Table 4: Supported Modes and Encoders with "Fly Extension" Commands.

Mode (parameter value Mode)	Encoder ("source")	Effect
0	Reserved.	Generates <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code> .
1	Encoder0.	Only for Processing-on-the-fly-correction. Encoder counter "Encoder0"-scaled. PreviewTime-corrected.
2	Encoder1.	Only for Processing-on-the-fly-correction. Encoder counter "Encoder1"-scaled. PreviewTime-corrected.
3	Encoder2.	Only for Processing-on-the-fly-correction. Tied to encoder counter "Encoder0"-scaled. PreviewTime-corrected.
4	Encoder2.	Only for Processing-on-the-fly-correction. Tied to encoder counter "Encoder1"-scaled. PreviewTime-corrected.
5	Reserved.	Generates <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code> .
6	McBSP 32 Bit.	Processing-on-the-fly correction with position specifications, <code>read_mcbsp( 0 )</code> .
7	McBSP 31 Bit.	<code>set_mcbsp_in</code> (Mode 1, 2 or 4) and <code>read_mcbsp( 0 )</code> .
8	McBSP 29 Bit.	Memory location for the x coordinate of the Processing-on-the-fly application from <code>set_multi_mcbsp_in</code> , <code>read_multi_mcbsp( 0 )</code> .
9	Reserved.	Generates <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code> .
10	McBSP LOW 16 Bit.	Processing-on-the-fly correction with position specifications at 2 Axes.
11	McBSP LOW 15 Bit.	From <code>set_mcbsp_in</code> (Mode= 3 ).
12	McBSP 29 Bit.	Memory location for the y coordinate of the Processing-on-the-fly application from <code>set_multi_mcbsp_in</code> , <code>read_multi_mcbsp( 1 )</code> .
13	Reserved.	Generates <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code> .
14	McBSP HIGH 16 Bit.	Processing-on-the-fly correction with position specifications at 2 Axes.
15	McBSP HIGH 15 Bit.	From <code>set_mcbsp_in</code> (Mode= 3 ).
16	McBSP 29 Bit.	Memory location for the z coordinate of the Processing-on-the-fly application from <code>set_multi_mcbsp_in</code> , <code>read_multi_mcbsp( 2 )</code> .
17	Encoder0.	Only for <code>wait_for_1_axis</code> and <code>wait_for_2_axes</code> . Not for Processing-on-the-fly corrections. Encoder counter "Encoder0"-unscaled and PreviewTime-corrected.
18	Encoder1.	Only for <code>wait_for_1_axis</code> and <code>wait_for_2_axes</code> . Not for Processing-on-the-fly corrections. Encoder counter "Encoder1"-unscaled and PreviewTime-corrected.
19	Encoder0.	Like 17, but not PreviewTime-corrected.
20	Encoder1.	Like 18, but not PreviewTime-corrected.



#### Notes on Table 4

- Each of these sources can be recorded by `set_trigger/set_trigger4`.
- Mode 1...4 as well as 17...18 are also suitable for scan systems with SCANhead control.
- With `wait_for_1_axis` and `wait_for_2_axes`, Mode 17...20 must be specified but not 1...4.
- With the other “Fly Extension” Commands where an Encoder specification is possible, Mode 1...4 must be specified but not 17...20.
- With the McBSP modes:
  - Users must make sure that the data is actually transferred in the correct format
  - Users must additionally set (exception:  
Mode = 6) a corresponding Processing-on-the-fly correction in `wait_for[*]` commands (by `set_fly[*]`, `set_mcbsp_in` or `set_multi_mcbsp_in`)
  - the RTC6 board “waits” for the Encoder values even without explicit activation of the Processing-on-the-fly correction
- A `get_last_error` return code `RTC6_PARAM_ERROR` is generated, if a Mode is not allowed.
- Further details can be found in the respective command descriptions.

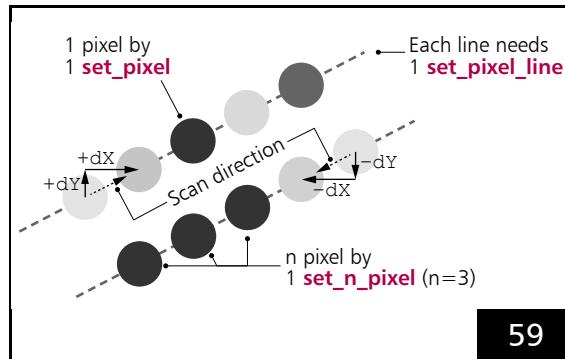
## 8.7 Pixel Output Mode – Marking Pixel Images (Bitmaps)

The vector commands described in [Chapter 7.1 "Marking Points, Lines and Arcs", page 125](#) are intended for scanning vector based images. Furthermore, the RTC6 PCIe Board allows marking pixel images (bitmaps). With suitably adjusted lasers, black-and-white images or greyscale images can be obtained. Raster and vector based images can be combined as desired.

### 8.7.1 Principle of Operation

Pixel images are created image line by image line, see [figure 59](#). Each line consists of a number of equidistant pixels. A line is generated in a single pass. During this pass, the laser focus moves – as with a normal [\*]mark[\*] command – at an (approximately) constant velocity along the entire image line (the motion is split-up into microsteps). The individual pixels are marked in passing: each pixel gets a laser pulse assigned at the appropriate location. By varying the laser energy from pixel to pixel, greyscale images (including black & white images) are produced.

Different output ports can be used to control the laser depending on mode and pixel output frequency, see command descriptions.



An individual `set_pixel_line` is required for each image line. The individual pixels of this line are then defined by successive `set_pixel`/`set_n_pixel`.

- By `set_pixel_line`, a single line is configured:
  - The *spatial* distance between the individual pixels with the parameters `dx`, `dy` (with `set_pixel_line_3d` additionally `dz`)
  - The *temporal* distance between the individual pixels with the parameter `HalfPeriod`
  - The parameter `Channel` is extended compared to the RTC5 to Channel = Mode + Port, where now the (pixel) mode is defined via the value Mode and the output port is defined via the value Port, see [Table 5](#) and [6](#)

### Notes

- The pixel output mode can be combined with Processing-on-the-fly, see [Chapter 8.6 "Processing-on-the-fly", page 227](#).
- The pixel output mode *cannot* be combined with Sky Writing, see [Chapter 7.2.4 "Sky Writing", page 149](#).<sup>(1)</sup>
- The pixel output mode *cannot* be combined with Wobbel, see [Chapter 8.4 "Wobbel Mode", page 218](#).

### 8.7.2 RTC6 Commands

Before an image line is written, a jump to the beginning of the line should be executed by a jump command.

At the beginning of each image line, the pixel output mode is activated by `set_pixel_line` or `set_pixel_line_3d`. The pixel distance and pixel output period (and, resultingly, the speed at which the image line is traversed) are simultaneously set as well. The pixel output period is defined by the parameter `HalfPeriod` (half pixel output period). This is also half the laser period. The pixel distance between two adjacent pixels in the line – and thus also the marking direction – is defined by a:

- 2D vector (`dx,dy`) with `set_pixel_line`
- 3D vector (`dx,dy,dz`) with `set_pixel_line_3d`

Directly after `set_pixel_line`/`set_pixel_line_3d`, `set_pixel` has to be called separately for each pixel of the line. This defines the laser energies to be outputted at the corresponding pixel locations.

(1) However, the pixel output mode can also be executed so that automatically generated (Sky Writing Mode 1-like) pre and post movements are included. This makes it easier to output pixel images with accurate positioning, see [Chapter 8.7.4 "Synchronization", page 254](#).

Pixel pulses are always outputted at the LASER1 output port, even for the purpose of synchronizing the laser (gating). If PulseLength is not specified as the port, the very PulseLength is outputted that has been specified at last by **set\_laser\_pulses**, however, at least 1/64  $\mu$ s.

As an alternative to a sequence of  $n$  identical **set\_pixel** calls, **set\_n\_pixel** can be used.

Then only one command is stored on the RTC6 PCIe Board, but during list processing the corresponding **set\_pixel** is executed  $n$  times. Particularly for black & white images, this can drastically reduce the size of lists. Do not confuse **set\_n\_pixel** with the **n\_set\_pixel** command (multi-board version of the **set\_pixel** command).

Prior to the end of an image line, no command other than **set\_pixel** or **set\_n\_pixel** must be inserted into the list. After **set\_pixel\_line**/**set\_pixel\_line\_3d**, the first list command that is *not* a **set\_pixel** or **set\_n\_pixel** command stops the pixel output mode and thus processing of the image line.

Each **set\_pixel**/**set\_n\_pixel** command that does not follow another **set\_pixel**/**set\_n\_pixel** or **set\_pixel\_line**/**set\_pixel\_line\_3d** command is ignored during processing and is thus a short list command, see **Section "Normal, Short, Variable and Multiple List Commands"**, page 288.

### Notes

- The number of pixels in an image line is limited only by the capacity of the RTC6 PCIe Board list memory, see **Chapter 14 "Technical Specifications of the RTC6 PCIe Board"**, page 823. It is suggested – especially for large bitmaps – to set up a new list for each image line to avoid a list change during the execution of one line.
- Each image line must start with a **set\_pixel\_line** or **set\_pixel\_line\_3d** command.
- The pixel distance ( $dx$ ,  $dy$ ) in the X and Y directions (in bits) (and with **set\_pixel\_line\_3d** also  $dz$ ) can be specified with floating point numbers. This allows scaling and rotating the image without rounding errors.
- Depending on the pixel output mode, the half pixel output period can be any integer-multiple of 1/64  $\mu$ s. It is independent from the 10  $\mu$ s position output period of the galvanometer scanners' (split-up into microsteps) motion. Very low pixel

output frequencies result in multiple galvanometer scanner steps per pixel, higher frequencies result in multiple pixel pulses per galvanometer scanner step.

- You can implement a pixel output mode in a 10  $\mu$ s raster with variable speed and/or curvilinear paths with the help of microvector commands, see **Chapter 8.8 "Microvector Commands"**, page 259.

### 8.7.3 Laser Control

Depending on the type of laser employed, the laser energy outputted at each pixel position can be varied by:

- laser pulse length
- laser power per pixel

#### “Classic Mode”

The “Classic Mode” (Mode = 0, see [page 681](#)) corresponds to the pixel output mode of the RTC4 and RTC5. Maximum pixel output frequency: 400 kHz.

- Variation Of Laser Pulse Length: `set_pixel` defines – for each pixel – the length of the pixel pulse (in units of 1/64 µs), which is outputted at the LASER1 port. For synchronization, see [Chapter 8.7.4 “Synchronization”, page 254](#).
- Variation Of Laser Power: `set_pixel` allows specification of a 12-bit analog voltage level for each pixel. The value is transferred either to the ANALOG OUT1 port or to the ANALOG OUT2 port (see above). For synchronization, see [Chapter 8.7.4 “Synchronization”, page 254](#).  
`set_n_pixel` defines the analog voltage level for n directly successive pixels of an image line.

#### “Extended Mode”

In “Extended Mode” (Mode = 16, see [page 681](#)), two pixels (32-bit values) are outputted at the specified port with each `set_pixel`. Maximum pixel output frequency: 800 kHz.

#### “Fast Mode”

In “Fast Mode” (Mode = 32, see [page 681](#)), four pixels of 16 bits each are outputted at the specified port with each `set_pixel`. Maximum pixel output frequency: 1.6 MHz.

#### “Ultra Fast Mode”

In “Ultra Fast Mode” (Mode = 64, see [page 681](#)) eight pixels of 8 bits each are outputted at the specified port with each `set_pixel`. Maximum pixel output frequency: 3.2 MHz.

#### Notes

- Pixel output frequencies > 800 kHz require the **Option “UFPM”**. Otherwise, the pixel output frequency is clipped to 800 kHz.
- It is recommended that some experiments be performed to determine an appropriate gradation curve for producing smooth greyscales. The resulting pixel greyscale values (“colors”) strongly depend on the employed material and the laser.
- The LASERON signal – as with a normal **[\*]mark[\*]** command – switches to “On” after the LaserOn delay has expired and remains “On” for the entire pixel line. After the last pixel has been outputted, it automatically goes to “Off”. See [Chapter 7.4 “Laser Control”, page 173](#).
- The LASER1 signal depends on the respective laser mode and the associated settings, as well as on the Mode (see [page 681](#)) set by `set_pixel_line/ set_pixel_line_3d`: A periodic signal with `HalfPeriod` from `set_pixel_line` and a `PulseLength` from `set_laser_pulses` (however, at least 1/64 µs long) is outputted. The following exceptions apply:
  - In Laser Mode 4 and Laser Mode 6, only a standby signal is outputted that cannot be varied. Power changes are only possible via the output ports ANALOG OUT1, ANALOG OUT2, 8-bit digital output port (EXTENSION 2) or 16-bit digital output port (EXTENSION 1).
  - If an output to `PulseLength` is specified as the Mode, `PulseLength` is varied per pixel.
  - No Softstart is performed, see [Chapter 7.4.7 “Softstart Mode \(not yet implemented\)”, page 184](#).
- The LASER2 signal follows the specifications of the respective laser mode.
- Pixel output mode and Sky Writing cannot be combined, see [Chapter 7.2.4 “Sky Writing”, page 149](#). However, a separate Sky Writing mode 1-like galvanometer scanner movement can be inserted, see [figure 60](#).
- The values for `HalfPeriod` and `PulseLength` set prior to `set_pixel_line` are not restored again. `set_laser_pulses` must be explicitly called again.

Table 5:

Mode	Max. frequency	No. of pixels per <code>set_pixel</code> / <code>set_n_pixel</code>	1 <sup>st</sup> parameter of <code>set_pixel</code> / <code>set_n_pixel</code> (name: <code>PortOutValue1</code> <sup>(a)</sup> )	2 <sup>nd</sup> parameter of <code>set_pixel</code> / <code>set_n_pixel</code> (name: <code>PortOutValue2</code> <sup>(b)</sup> )
0, 256 <sup>(c)</sup>	400 kHz	1	Pixel 1 (defined by the pulse length). The output is done as signal LASER1.	Port output (but not port 5) for Pixel 1  Pixel 1 (1 <sup>st</sup> parameter: 32-bit; 2 <sup>nd</sup> parameter: up to 16 bit, depending on port)
16	800 kHz	2	Port output for Pixel 1 (up to 32 bits per pixel, depending on port)	Port output for Pixel 2 (up to 32 bits per pixel, depending on port)  Pixel 1  Pixel 2
32 <sup>(d)</sup>	1.6 MHz	4	Port output for Pixel 1, 2 (up to 16 bits per pixel, depending on port)	Port output for Pixel 3, 4 (up to 16 bits per pixel, depending on port)  Pixel 2   Pixel 1  Pixel 4   Pixel 3
64 <sup>(d)</sup>	3.2 MHz	8	Port output for Pixel 1, 2, 3, 4 (up to 8 bits per pixel, for all ports)	Port output for Pixel 5, 6, 7, 8 (up to 8 bits per pixel, for all ports)  Pixel 4   Pixel 3   Pixel 2   Pixel 1  Pixel 8   Pixel 7   Pixel 6   Pixel 5
Bit 31.....Bit 0			Bit 31.....Bit 0	

(a) PulseLength in RTC6 Software Packages < v1.3.3.

(b) AnalogOut in RTC6 Software Packages < v1.3.3.

(c) The galvanometer scanner movement is *not* continuous in Mode = 0 and continuous in Mode = 256.

(d) > 800 kHz only possible with Option "UFPM".



Table 6:

Output Port	Where at the RTC6 PCIe Board
1	12-bit analog output port 1 <a href="#">LASER Connector, page 62</a> , ANALOG OUT1 See <a href="#">figure 13</a> .
2	12-bit analog output port 2 <a href="#">LASER Connector, page 62</a> , ANALOG OUT2 See <a href="#">figure 13</a> .  <a href="#">MARKING ON THE FLY Socket Connector, page 71</a> , ANALOG OUT2 See <a href="#">figure 21</a> .
3	8-bit digital output port <a href="#">EXTENSION 2 Socket Connector, page 69</a> See <a href="#">figure 20</a> .
4	16-bit digital output port <a href="#">EXTENSION 1 Socket Connector, page 67</a> See <a href="#">figure 18</a> .
5 <sup>(a)(b)</sup>	Pulse length <a href="#">LASER Connector, page 62</a> , LASER1 See <a href="#">figure 13</a> .

(a) Port = 5 cannot be combined with Mode = 0.

(b) Port = 5 cannot be combined with Mode = 256.

#### 8.7.4 Synchronization

The pixel output timing diagram for one image line with 3 pixels is shown in [figure 61](#).

To prepare the laser control, `set_pixel_line/set_pixel_line_3d` switches off the signals for “laser active” operation from a previous **Mark command** after a LaserOff delay (as with a **Jump command**) and waits until the laser is actually off.

During this waiting period, the galvanometer scanners do not move in “**Classic Mode**” (Mode = 0, see [page 681](#)) only. In all other modes, they continue to move at the speed defined by `HalfPeriod` and pixel distance in equidistant microsteps. This allows to program jerk-free run-in and run-out movements. Initial acceleration phases can be hidden by a **LaserOn** delay (as with a normal `[*]mark[*]` command) or by a corresponding number of “idle pixels” (see below).

After that, depending on the laser mode, pixel output starts immediately or after a Q-Switch delay, see [figure 61](#). Analog signals at ANALOG OUT1 or ANALOG OUT2 change synchronously with the leading edge of each pixel pulse. The digital-to-analog converter requires about  $1.5 \mu\text{s} \dots 3 \mu\text{s}$  to produce a stable analog output signal. With pixel output frequencies above around 100 kHz (`HalfPeriod < approx. 320`) digital-to-analog conversion cannot always be fully completed. At such pixel frequencies, it must be carefully checked whether the functionality is sufficient for the intended purpose.

**Bit #1** in `set_laser_control( Ctrl )` can be used to shift the laser pulse and thus the start of the digital-to-analog conversion by half a pixel period.

The pixel line ends with the first list command that is not a `set_pixel` or `set_n_pixel`.

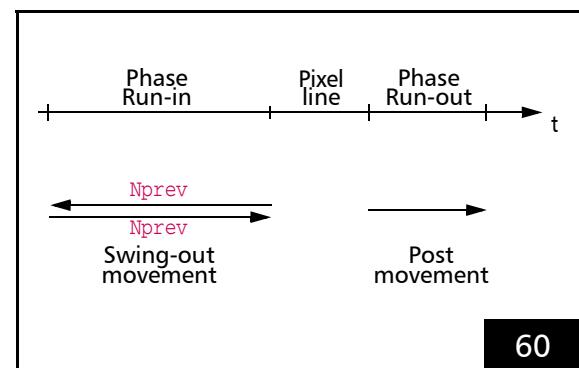
For the laser to be switched off even in the middle of a  $10 \mu\text{s}$  cycle, a default pixel is automatically outputted after the last pixel. This is repeated as often as necessary until a started  $10 \mu\text{s}$  cycle is finished. Then the laser is finally switched off.

The default pixel should be defined appropriately prior to `set_pixel_line/set_pixel_line_3d` in order to achieve a non-visible laser marking (= “idle pixels”) in the run out, see `set_port_default` and `set_default_pixel`.

The RTC6 board waits – especially at pixel frequencies  $< 100 \text{ kHz}$  – until the default pixel is outputted.

The galvanometer scanners continue to run during this time, to ensure jerk-free connection movements (programmed by the user). No scanner delay is automatically inserted. In “**Classic Mode**” (Mode = 0, see [page 681](#)), the galvanometer scanners remain idle for a few clock cycles. With Mode + 256 this can be suppressed (even for the forerun phase).

The tracking error and the hidden acceleration phase mean a pixel line shift”, see [figure 61](#). Normally, this needs to be compensated for by an adjusted run-in. To make this easier, Mode + 512 can be used to switch on Sky Writing mode 1-similar movements in the run-in and run-out phase and thus place the pixel line with pinpoint accuracy, see [figure 60](#).



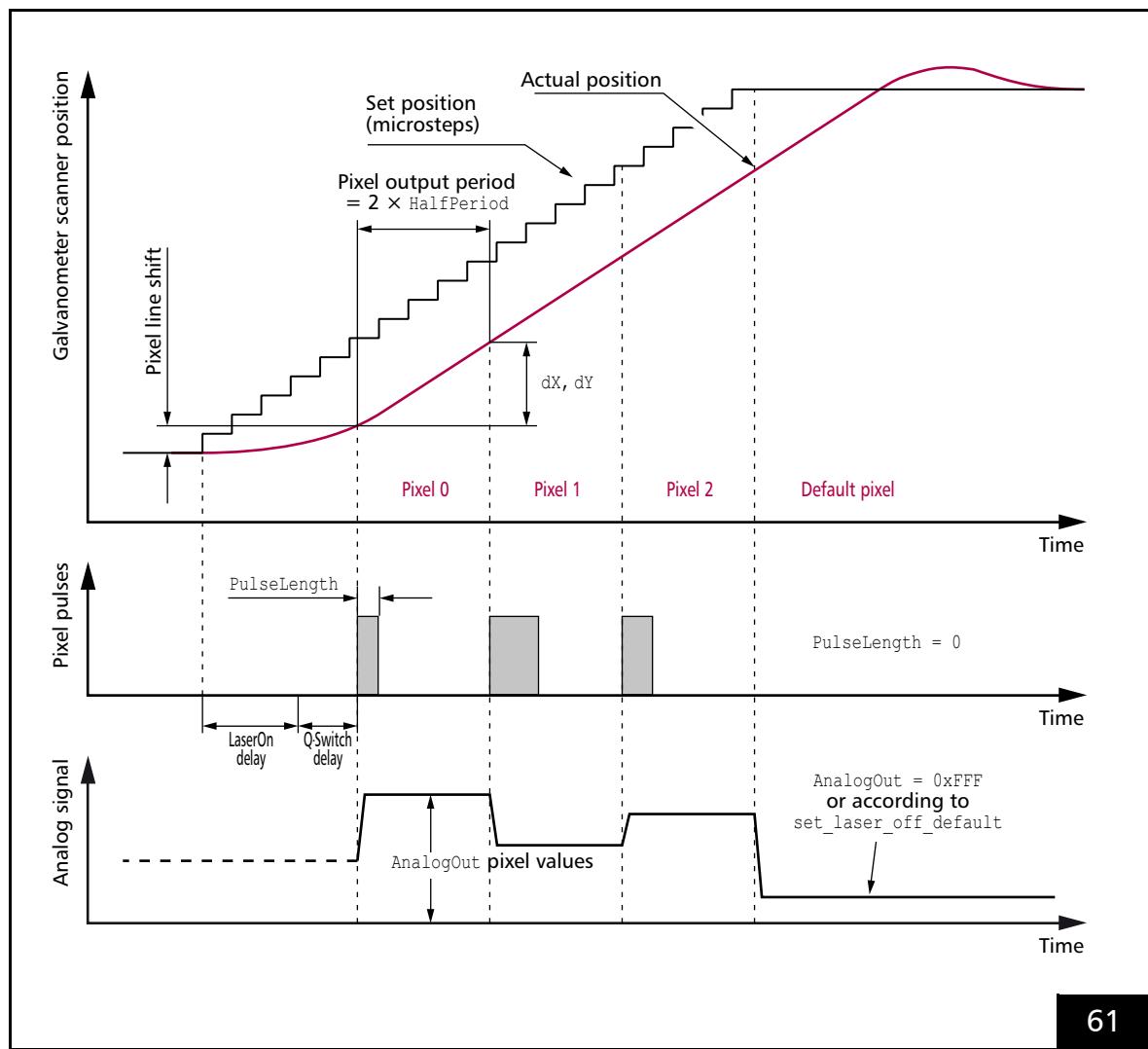
Modus + 512

These movements cannot be combined with Sky Writing mode 2. The duration of the swing-out movement must be explicitly defined via `Nprev` of `set_sky_writing_para` in advance.<sup>(1)</sup>

(1) With excelliSCAN scan heads, Sky Writing and automatic delay calculation need to be switched on, see manual “excelliSCAN scan heads – Functional Principle of SCANhead Servo Control and Operation by RTC6 Boards”.

## Notes

- With `HalfPeriod < PulseLength / 2` the laser does not switch off between the pixels.
- When specifying `HalfPeriod` and pixel distance, observe the dynamics of the scan system (marking speed) and the properties of the laser system (power modulation).



Timing of the galvanometer scanner positions and the laser control signals in the pixel output mode Mode = 0 and a YAG Mode, see [Chapter 7.4.4 "YAG Modes 1, 2, 3, 5"](#), page 179.  
The pixel output period in this example is approx. 4.5 microsteps.



### Source Code Example

The code must be included in a user program.

```

//  UINT = uint32_t
//  LONG = int32_t

// laser parameters
UINT LaserMode = 0;
UINT LaserControl = 0x0;
LONG LaserOnDelay = (LONG)round(100.0*64.0); // LaserOnDelay = 100 µs
UINT LaserOffDelay = (UINT)round(100.0*64.0); // LaserOffDelay = 100 µs
UINT HalfPeriod = (UINT)round(5.0 * 32.0); // Period = 5 µs, PixelFrequency = 200 kHz
UINT PulseLength = (UINT)round(0.5 * 64.0); // PulseLength = 0.5 µs

// Pixel output mode variables
enum E_PixelModes
{
    STANDARD      = 0,      // like RTC5-Mode; up to 400 kHz; 1 pixel per set_pixel command; 2 values of 32 bit per pixel
    ENHANCED       = 16,     // up to 800 kHz; 2 pixel per set_pixel command; 1 value of 32 bit per pixel
    FAST           = 32,     // up to 1.6 MHz; 4 pixel per set_pixel command; 1 value of 16 bit per pixel
    ULTRAFAST      = 64,     // up to 3.2 MHz; 8 pixel per set_pixel command; 1 value of 8 bit per pixel
    STANDARD_MOVE  = 256    // like STANDARD, but with continuous scanner movement
} Mode;

enum E_PixelPorts
{
    NO_OUT_PORT   = 0,
    ANALOG_OUT1   = 1,      // laser connector ANALOG OUT1
    ANALOG_OUT2   = 2,      // laser connector ANALOG OUT2
    DIGITAL_8bit  = 3,      // EXTENSION 2 socket connector (8-bit)
    DIGITAL_16Bit = 4,      // EXTENSION 1 socket connector (16-bit)
    PULSE_LENGTH  = 5       // not allowed for mode = 0 or mode = 256
} Port;

Port = DIGITAL_8Bit;          // pixel output port number
Mode = STANDARD;             // pixel output mode
UINT Channel = Port + Mode; // pixel channel

UINT Number = 1;              // set_pixel repetition number

UINT PixelArray[17] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 }; // port output values
UINT PortOutValue1 = PixelArray[0];
UINT PortOutValue2 = PixelArray[0];

LONG dx = (LONG)round(0.01 * CalibrationFactorXY); // 10 um spot distance in x direction
LONG dy = (LONG)round(0.0 * CalibrationFactorXY); // 0 um spot distance in y direction

LONG PixelLineStartPosX = (LONG)round(0.0 * CalibrationFactorXY);
LONG PixelLineStartPosY = (LONG)round(0.0 * CalibrationFactorXY);

// basic configuration
set_laser_mode(LaserMode);
set_laser_control(LaserControl);

// RTC-List
set_start_list_pos(1, 0);
set_laser_delays(LaserOnDelay, LaserOffDelay);
set_default_pixel_list(0); // sets pixel default PulseLength
UINT PortDefault = Port - 1; // CAUTION: port numbers differ between set_port_default/set_port_default_list and
                           // set_pixel_line
set_port_default_list(PortDefault, PixelArray[0]); // sets default value for specified port

```



```

switch (Mode)
{
case STANDARD:
case STANDARD_MOVE:
{
    // standard pixel output mode up to 400 kHz (Mode = 0, 256)
    // 2 values of 32 bit per pixel
    // 1 pixel per set_pixel command
    //-----
    // |      pixel1      |
    // | 32 Bit || 32 Bit |
    // | PulseLength || PortOutValue |
    //-----
    HalfPeriod = (UINT)round(5.0 * 32.0);    // Period = 5 µs, PixelFrequency = 200 kHz
    PulseLength = (UINT)round(0.5 * 64.0);    // PulseLength = 0.5 µs

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);
    set_n_pixel(1 * PulseLength, PixelArray[1], Number);           // pixel 1
    set_n_pixel(2 * PulseLength, PixelArray[2], Number);           // pixel 2
    set_n_pixel(3 * PulseLength, PixelArray[3], Number);           // pixel 3
    set_n_pixel(4 * PulseLength, PixelArray[4], Number * 2);       // (pixel 4)*2

}
break;

case ENHANCED:
{
    // enhanced pixel output mode up to 800 kHz (Mode = 16)
    // 1 value of 32 bit per pixel
    // 2 pixel per set_pixel command
    //-----
    // |      pixel1      |      pixel2      |
    // | 32 Bit || 32 Bit |
    // | PortOutValue1 || PortOutValue2 |
    //-----
    HalfPeriod = (UINT)round(2.0 * 32.0);    // Period = 2 µs, PixelFrequency = 500 kHz
    PulseLength = (UINT)round(0.5 * 64.0);    // PulseLength = 0.5 µs
    set_laser_pulses(HalfPeriod, PulseLength);

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);
    set_n_pixel(PixelArray[1], PixelArray[2], Number);           // pixel 1, pixel 2
    set_n_pixel(PixelArray[3], PixelArray[4], Number);           // pixel 3, pixel 4
    set_n_pixel(PixelArray[5], PixelArray[6], Number * 2);       // (pixel 5, pixel 6)*2
    //--> pixel output: pixel 5, pixel 6, pixel 5, pixel 6

}
break;
}

```



```

case FAST:
{
    // fast pixel output mode up to 1.6 MHz (Mode = 32) - needs UFPM-Option
    // 1 value of 16 bit per pixel
    // 4 pixel per set_pixel command
    //-----
    // |   pixel2   |   pixel1   |   pixel4   |   pixel3   |
    // |   16 Bit   |   16 Bit   |   16 Bit   |   16 Bit   |
    // |           PortOutValue1           |           PortOutValue2           |
    //-----
    HalfPeriod = (UINT)round(1.0 * 32.0);    // Period = 1 µs, PixelFrequency = 1 MHz
    PulseLength = (UINT)round(0.5 * 64.0);   // PulseLength = 0.5 µs
    set_laser_pulses(HalfPeriod, PulseLength);

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);
    // low 16 bit - first pixel, high 16 bit - second pixel
    PortOutValue1 = PixelArray[1] | (PixelArray[2] << 16);
    PortOutValue2 = PixelArray[3] | (PixelArray[4] << 16);
    set_n_pixel(PortOutValue1, PortOutValue2, Number);           // pixel 2 + pixel 1, pixel 4 + pixel 3

    PortOutValue1 = PixelArray[5] | (PixelArray[6] << 16);
    PortOutValue2 = PixelArray[7] | (PixelArray[8] << 16);
    set_n_pixel(PortOutValue1, PortOutValue2, Number * 2);      // (pixel 6 + pixel 5, pixel 8 + pixel 7)*2
    //--> pixel output: pixel 5, pixel 6, pixel 7, pixel 8, pixel 5, pixel 6, pixel 7, pixel 8

}
break;
case ULTRAFAST:
{
    // ultra fast pixel output mode up to 3.2 MHz (Mode = 64) - needs UFPM-Option
    // 1 value of 8 bit per pixel
    // 8 pixel per set_pixel command
    //-----
    // |   pixel4   |   pixel3   |   pixel2   |   pixel1   |   pixel8   |   pixel7   |   pixel6   |   pixel5   |
    // |   8 Bit    |
    // |           PortOutValue1           |           PortOutValue2           |
    //-----
    HalfPeriod = (UINT)round(0.5 * 32.0);    // Period = 0.5 µs, PixelFrequency = 2 MHz
    PulseLength = (UINT)round(0.25 * 64.0);   // PulseLength = 0.25 µs
    set_laser_pulses(HalfPeriod, PulseLength);

    jump_abs(PixelLineStartPosX, PixelLineStartPosY);
    set_pixel_line(Channel, HalfPeriod, dx, dy);

    // lowest 8 bit - first pixel, highest 8 bit - fourth pixel
    PortOutValue1 = PixelArray[1] | (PixelArray[2] << 8) | (PixelArray[3] << 16) | (PixelArray[4] << 24);
    PortOutValue2 = PixelArray[5] | (PixelArray[6] << 8) | (PixelArray[7] << 16) | (PixelArray[8] << 24);

    // pixel 4 + pixel 3 + pixel 2 + pixel 1, pixel 8 + pixel 7 + pixel 6 + pixel 5
    set_n_pixel(PortOutValue1, PortOutValue2, Number);

    PortOutValue1 = PixelArray[9] | (PixelArray[10] << 8) | (PixelArray[11] << 16) | (PixelArray[12] << 24);
    PortOutValue2 = PixelArray[13] | (PixelArray[14] << 8) | (PixelArray[15] << 16) | (PixelArray[16] << 24);

    // (pixel 12 + pixel 11 + pixel 10 + pixel 9, pixel 16 + pixel 15 + pixel 14 + pixel 13)*2
    set_n_pixel(PortOutValue1, PortOutValue2, Number * 2);
    // --> pixel output: pixel 9, pixel 10, pixel 11, pixel 12, pixel 13, pixel 14, pixel 15, pixel 16,
    //           pixel 9, pixel 10, pixel 11, pixel 12, pixel 13, pixel 14, pixel 15, pixel 16

}
break;

default:
break;

}

set_end_of_list();

```



## 8.8 Microvector Commands

Microvector commands move the galvanometer scanners directly to the specified position by a "Hard jump"

- in 2D image field:
  - `micro_vector_abs`
  - `micro_vector_rel`
- in 3D image field:
  - `micro_vector_abs_3d`
  - `micro_vector_rel_3d`

The signals for "laser active" operation can be switched on and off individually for each microvector command by the specified laser delays (parameters `LasOn` and `LasOff`). The laser delays defined by `set_laser_delays` are not overwritten by that. These remain valid for normal jump commands and `[*]mark[*]` commands.

The microvector commands can be used to mark trajectories (see Glossary entry on [page 880](#)) of arbitrary shape and at variable speed, as long as the dynamic limits of the scan system are not exceeded.

### Notes

- Microvector commands wait for a preceding scanner delay. They never trigger new scanner delays.
- Users themselves are responsible for appropriately parameterizing microvector commands:
  - Complying with the dynamic limits of the scan system
  - Avoiding cross over and take over with LaserON and LaserOff, see [Chapter 7.2.3 "Notes on Optimizing the Delays", page 144](#)
- A Wobbel motion specified by `set_wobbel` or `set_wobbel_mode`, see [Chapter 8.4 "Wobbel Mode", page 218](#) is not taken into account (but also not deactivated).
- The Sky Writing mode, see [Chapter 7.2.4 "Sky Writing", page 149](#) is not taken into account (but also not deactivated).
- The output values are calculated according [Chapter 7.3.6 "Output Values to the Scan System", page 170](#) without 1 and 2.

## 8.9 Timed Vector Commands and Timed Arc Commands

"Normal" vector commands (jump commands and `[*]mark[*]` commands) and arc commands are executed in such a way that the laser focus moves with a defined speed<sup>(1)</sup>. This is fine for most laser marking and laser material processing applications.

At the beginning of a jump, the signals for "laser active" operation are switched off. The signals for "laser standby" operation are switched on depending on the settings, see [Chapter 7.4.1 "Enabling, Activating and Switching Laser Control Signals", page 173](#).

However, some applications require that each vector command or arc command consumes exactly the same amount of time, regardless of its spacial length.

In this case, it is necessary to specify a jump *duration* or mark process *duration* (rather than the jump speed or mark speed).

Timed commands allow specification of the duration of the vector command or arc command with an accuracy of 10 µs (the output period of the microsteps) and in the range from 10 µs to 167,772,160 µs ( $\approx$  2.8 min).

A vector or arc is split-up into the specified ( $T$  Parameter) number of microsteps that correspond exactly to the specified time, see also [Chapter 7.1.2 "Microstepping", page 129](#).

Thus, jump speed and mark speed automatically depend on the length of the vectors or arcs.

Of course, this means that the *jump* or *mark* speed, that is, the velocity of the laser focus (and the angular velocity of the movement of the mirrors) depend on the length of the vectors and arcs.

The following timed commands are available:

- Vectors and arcs
  - `timed_jump_abs`
  - `timed_jump_rel`
  - `timed_mark_abs`
  - `timed_mark_rel`
  - `timed_arc_abs`
  - `timed_arc_rel`
- Parametrized vectors
  - `timed_para_jump_abs`
  - `timed_para_jump_rel`
  - `timed_para_mark_abs`
  - `timed_para_mark_rel`
- 3D vectors
  - `timed_jump_abs_3d`
  - `timed_jump_rel_3d`
  - `timed_mark_abs_3d`
  - `timed_mark_rel_3d`
- Parametrized 3D vectors
  - `timed_para_jump_abs_3d`
  - `timed_para_jump_rel_3d`
  - `timed_para_mark_abs_3d`
  - `timed_para_mark_rel_3d`

### Notes

- After a timed command, a "normal" jump delay, mark delay or polygon delay is inserted.
- Ellipses are fundamentally not available as timed commands.
- With  $T < 5 \mu s$ , a timed command is synonym to its "normal" (= without `[timed_*]`) command.
- The total execution time of a timed command is the sum of the specified time and the associated delay, see also [Chapter 7.2.2 "Scanner Delays", page 136](#).

(1) Either jump speed or mark speed.

## 8.10 Automatic Self-Calibration

### 8.10.1 Use for Drift Compensation

Long-term repeatability is very important in many applications, for example, for rapid prototyping in which the processing operation can span several hours. For such laser applications, the galvanometer scanner's long-term drift and temperature drift, which manifest as a shift (offset drift) and increase or decrease in the size (gain drift) of the working image field, can exceed the allowed tolerances.

In such applications, it is helpful to start up the application only after the galvanometer scanners have reached their operating temperature. In addition, the magnitudes of environmental fluctuations (for example, operating temperature changes to which the scan system is exposed) should be kept as small as possible and the scan system preferably operated with a constant load.

For higher long-term repeatability requirements, SCANLAB scan systems can be equipped with an additional internal sensor system for automatic self-calibration (ASC sensor system, Home-In sensors). This allows the position detectors of the galvanometer scanners to be calibrated and gain drift and offset drifts to be reliably compensated. The positioning accuracy is thus maintained over long periods of time. Remaining long-term drift effects are of the same order of magnitude as short-term repeatability accuracies.

### 8.10.2 How it Works

By [auto\\_cal](#), a measurement routine can be started for determining the exact control values for reference positions (Home-In positions) defined by the internal sensor system.

For drift *measurement*, the routine should be executed:

- When setting up the equipment – to determine reference values for the Home-In positions, see [Chapter 8.10.3 "Determining Reference Values", page 263](#)
- During the application at appropriate time intervals – to determine if and how the Home-In positions have changed, see [Chapter 8.10.4 "Calibration During the Application", page 263](#)

For drift *compensation*, appropriate gain values and offset values can be calculated and set separately for each galvanometer scanner based on the determined deviations between the current Home-In position and the reference value. See also [Chapter 7.3.6 "Output Values to the Scan System", page 170, #9](#).

The setting can also be made manually with [set\\_hi](#), if customer-specific measuring procedures are to be used, see [Section "Customer-Specific Calibration", page 264](#).

#### Notes

- Prior to performing a measurement routine, [auto\\_cal\( Command = 4 \)](#) can be used to check if:
  - the attached scan system in fact has an internal sensor system for automatic self-calibration (Home-In sensors)
  - the sensor system is functioning properly



This ASC hardware check also occurs automatically by **auto\_cal**( Command = 0 ) and if required, for **auto\_cal**( Command = 1 and 3 ), see also **auto\_cal** command description and **get\_auto\_cal**.

- During execution of the measurement routine determining the Home-In positions:
  - the laser should be switched off
  - no other commands should be sent to the scan system
- For Gain/Offset Correction:  
See [Chapter 7.3.6 "Output Values to the Scan System", page 170 #9](#).  
After an initialization by **load\_program\_file** or **auto\_cal**( Command = 2 ), the following is set:
  - Gain = 1.0
  - Offset = 0

- For iDRIVE scan systems<sup>(1)</sup>, the following applies in addition:
  - At the beginning of a measurement routine, the XY2-100 status word is automatically selected to be returned from the scan system. At the end of the measurement routine, the previously set data type is restored.
  - **auto\_cal** aborts with an error result value 7 if
    - **auto\_cal** is to be executed for the first scan head connector, and
    - an "Automatic Laser Control" in Mode = 2 (basic mode) has been activated, see [Section "Speed-Dependent Laser Control", page 190](#). The "Automatic Laser Control" must be deactivated before performing automatic self-calibration.
  - **auto\_cal** also aborts (result value 1, 10 or 11), if the scale factor has been set by **control\_command**( Data = 12xx<sub>H</sub> ) to a value < 1. This is because the sensor positions then are not reachable, see also **control\_command**( Data = 053F<sub>H</sub> ). The scale factor must have been set to 1 (default setting) by prior to automatic self-calibration
    - **control\_command**( Data = 1283<sub>H</sub> ) and
    - **control\_command**( Data = 1200<sub>H</sub> ).

(1) See glossary entry on [page 879](#).

### 8.10.3 Determining Reference Values

The reference values are determined by `auto_cal` (`Command = 0`). This starts the measurement routine for determining the current Home-In positions. These are then the reference values for later calibrations with `auto_cal` (`Command = 1 or 3`). Immediately after determination, they can be read out by `get_hi_pos` and are available for customer-specific calibration, see [Section "Customer-Specific Calibration"](#), [page 264](#). The reference values are stored in the [Flash memory](#) of the RTC6 board. This guarantees that the reference values are available even after a restart.

#### Notes

- After `auto_cal` (`Command = 0`), the galvanometer scanners are in the same real image field position as before the command.
- The reference values should be determined when the machine is set up, for example, when the scan system is installed or exchanged. When exchanging the scan system, for which reference values have already been determined earlier and read out by `get_hi_pos`, these reference values can be transferred directly to the RTC6 board by `write_hi_pos`.
- Reference values should be determined under conditions (ambient temperature, load) typical for the application and after the overall system has fully attained its operating temperature. The reference values should always be determined only after a warm-up time of more than 20 minutes and not before the TempOK signal has been activated.
- Execution of `auto_cal` (`Command = 0`) typically lasts up to 10 seconds.
- Storing the reference values takes several ms and interrupts the  $10\ \mu\text{s}$  clock period of the [DSP](#). For scan systems with clock cycle monitoring (interlock), this should be temporarily deactivated.

### 8.10.4 Calibration During the Application

#### Automatic Self-Calibration

Automatic self-calibration of the scan system during an application can be executed with `auto_cal` (`Command = 1`).

This starts a measurement routine for determining the current Home-In positions. From the deviations from the stored reference values (see [Chapter 8.10.3 "Determining Reference Values", page 263](#)) determined in the process, gain values and offset values are calculated and set separately for the x axis and y axis.

This drift compensation is effective immediately. It is valid until:

- `auto_cal` (`Command = 1`) is called again
- it is switched off
  - by `auto_cal` (`Command = 2`)
  - after `load_program_file`

#### Notes

- After `auto_cal` (`Command = 1`) the galvanometer scanners are in the same position in the real image field as before the command.
- The calibration routine started with `auto_cal` (`Command = 1`) typically lasts 1...2 seconds (depending on the strength of the drift). An ASC hardware check is automatically performed, if `auto_cal` (`Command = 0 oder 4`) has not been executed prior to the first time call of `auto_cal` (`Command = 1`). This extends the execution of `auto_cal` by a few seconds.

## Customer-Specific Calibration

Automatic self-calibration is midpoint centered, that is, the image field center remains stable.

If an alternative is preferred (for example, if the left edge should remain stable, size is irrelevant), then the calibration can also be externally calculated and set.

For such a customer-specific calibration, the Home-In positions determined by `auto_cal`(`Command` = 0) can be read out by `get_hi_pos`. From this, compensating gain values and offset values can be calculated and set by `set_hi`. The drift compensation set in this way is effective immediately.

### Notes

- `get_hi_pos` returns the last measured Home-In positions. These are the stored Home-In reference values immediately after
  - `auto_cal`(`Command` = 0)
  - initialization by `load_program_file`
- After `set_hi`, a correction of the current position occurs at jump speed.
- Gain and offset correction factors can also be set by `set_hi` for systems *without* Home-In sensors.
- After `auto_cal`(`Command` = 3) the galvanometer scanners are in exactly the same position as before the command.

## Supplemental Information about Calibration

- The accuracy of fit of the set drift compensation can decrease with increasing time. Therefore, calibration should be repeated after appropriate time intervals. The shorter the time interval between individual calibrations, the higher the attained long-term repeatability. Time intervals are typically in the range of minutes. Events such as workpiece changes or line feeds are ideal opportunities for conducting a new calibration.
- The accuracy of fit of the calibration, and the thereby attained long-term repeatability, are further enhanced by steady environmental and load conditions.
- The measurement routine determines the Home-In positions by several measurements. If deviations between the individual measurements are too large (maximum – minimum > 96 bits), the measurement routine aborts and an error 2 is returned. In this case, no reference values are

stored for the affected axis (`Command` = 0) and the gain and offset factors remain unchanged with (`Command` = 1)<sup>(1)</sup>. Then `get_hi_pos` returns 0 instead of the faulty values.

- An error within an individual measurement cycle can be caused by a brief mechanical or electrical disturbance. If significant spreading occurs within an individual measurement cycle, we recommend the following:
  - either a further measurement cycle is immediately conducted or
  - the error is initially ignored while using the current gain values and offset values until new correction values are determined by the next (successful) measurement cycle.

Significant spreading across several measurement cycles might indicate a defect in the internal sensor system or another part of the scan system. But because continuous (mechanical or electrical) external disturbances or contamination can also impair automatic self-calibration, the scan system and its environment should in such cases be appropriately inspected to assess overall functionality.

- Certain hardware error states (for example, signal faulty or not found) get permanently stored. After correcting the error, you must call `auto_cal` with `Command` = 0 or 4 to clear the error state. Until this time, correction with `Command` = 1 or 3 is not possible.

(1) (`Command` = 0) always sets gain = 1.0 and offset = 0.

## 8.11 Camming

**camming** produces a marking that simulates the classic camshaft action of moving a valve tappet – or more generally a cam disk moving a lever. An example for a camming process is shown in [figure 62](#).

The galvanometer scanner motion here is a lever movement defined as a 2D curve. It is written in a list as a closed point-by-point sequence of **mark\_abs** commands.

The entire curve must fit within a contiguous list region, see **config\_list**. Though it is not possible to switch among lists to load further portions of the curve.

“Propulsion” is furnished either by external fed in encoder pulses or by internally simulated ones.

Each outputted point is derived from the **xy** coordinate of a **mark\_abs** at the position **FirstPos + Index**, whereby **Index** =   
 $\text{Round}((\text{EncoderCurrent} - \text{EncoderStart}) \times \text{Scale})$ .   
**FirstPos** is the first **mark\_abs** command’s list position and **Scale** is a freely selectable scale factor.   
**EncoderStart** gets automatically determined when **camming** is called. There is no automatic encoder reset. The first outputted point is always **Index** = 0.

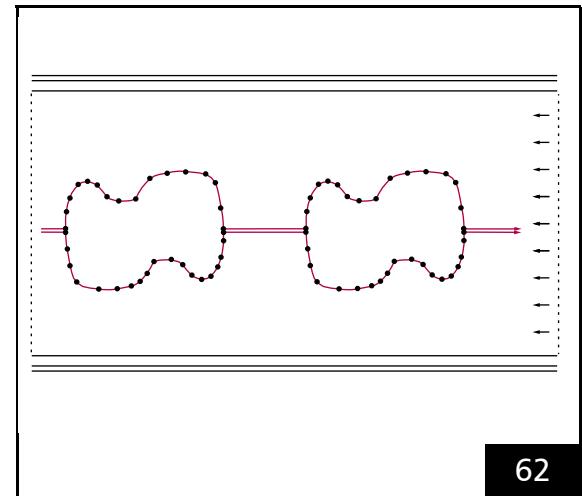
The individual points are executed every 10 µs as a “Hard jump” without scanner delays.

The distance between two points (= “resolution”) is freely selectable.

**Scale** determines how precisely the curve is sampled. The larger **Scale** is, the coarser is the piecewise linear approximation of the curve.

The number of encoder pulses per 10 µs clock period and the spacing of the points determine the actual mark speed.

“Resolution”, **Scale** and encoder speed should suit the dynamic characteristics of the connected scan system.



62

Camming example. A transport system moves a continuous workpiece. Two scan heads team up to mark the contours. Schematic depiction.

The camming process can be controlled in various ways, see **camming** command description:

- **Ctrl > 0**  
The laser is controlled externally (with **laser\_signal\_on\_list** before **camming** and **laser\_signal\_off\_list** after **camming**)
- **Ctrl = 0**  
The laser is controlled RTC6 board-internally automatically (as with a normal **Polyline** with consideration of laser delays)

The curve can be executed once and then automatically ended (**Ctrl** = 0 or **Ctrl** = 1). The list then continues by executing the next command that follows the end of the point list (the length of the point list is defined by **NPos**).

In accordance with encoder direction point lists can also run backward. Then, the curve is terminated with **Index** = 0.

The curve can be repeated indefinitely (**Ctrl** = 2 or **Ctrl** = 3). To avoid “Hard jumps” at the start of a repeat, the point list should represent a closed curve. Indefinite repeating must be cancelled by **stop\_execution** or an external /STOP.



At any time, the curve can be restarted at the address defined by **set\_extstartpos** or **set\_extstartpos\_list** (typically but not necessarily **FirstPos**) by an external /START (independently of the current index, possibly hard-jumped to the first marking position).

#### Notes

- An external /START during list execution is:
  - Suppressed during normal operation
  - Not suppressed during a camming process with indefinite repeating

If **Ctrl** = 2, then the camming process waits at the end of the point list for a new start. If **Ctrl** = 3, then processing automatically starts again from the beginning (the point list is processed as a ring buffer).

Furthermore, the camming process can be combined with Processing-on-the-fly (which can apply its own scaling if different from **camming** parameter **Scale**).

The laser control can be combined with the automatic ““vector-controlled laser control”” (**set\_vector\_control**). This requires to use **para\_mark\_abs[\*]** commands instead of **mark\_abs[\*]** commands. The value defined there is outputted immediately, thus allowing systematic variation of laser power along the curve. If automatic vector-defined laser control is not enabled, then no laser power is outputted by the **para\_mark\_abs[\*]** commands.

In addition, a combination with Encoder-speed-dependent laser control is possible, see **Section “Encoder-Speed-Dependent Laser Control”, page 192**.

Alternatively, **mark\_abs\_3d** or **para\_mark\_abs\_3d** as well as **timed\_mark\_abs\_3d** can be used.

However, the z coordinate and **T** parameter is ignored during outputting. Other commands within point lists are likewise ignored.

Point output then remains unchanged for one clock period.

#### Notes for Testing

- If a curve point list is finished by **set\_end\_of\_list** and does not cross the boundary between List 1 and List 2 (see **Chapter 6.4 “List Handling”, page 95**), then the curve shape can be tested using a normal list start, for example, by

```
execute_at_pointer( Pos )
(to some extend as a Polyline, but with a
predefined mark speed and using the currently
defined variable polygon delay).
• The point list should be closed with list_return, if
it is part of a subroutine and a sub_call call is
used for testing.
• If the test should include the “Hard jump”, then
timed_mark_abs_3d (with Time = 10 µs) can be
used as well, but not timed_para_mark_abs_3d.
```

## 8.12 Time Measurements

RTC6 PCIe Boards are equipped with an integrated timer, which is referred to as the “RTC6 timer” in the following. The RTC6 timer only counts clock cycles of list commands. Counting is paused during interruptions by `set_wait` or `pause_list`.

In order to measure the marking time consumed by any particular marking process, `save_and_restart_timer` is called before and then after the marking process. `save_and_restart_timer` saves the present RTC6 timer value and resets it to 0. The elapsed time can then be read by `get_time`, which returns the RTC6 timer value saved during the most recent call of `save_and_restart_timer`.

The present RTC6 timer value can be read out by `get_lap_time`. It returns the elapsed time since the last call of `save_and_restart_timer` but without resetting the RTC6 timer to zero. In this way the interim execution time of lengthy marking processes can be monitored.

### Time Stamp

A “time stamp counter” is additionally available. It starts counting at 0 with `load_program_file` and counts uninterruptible all 10 µs clock periods. Using the “time stamp counter” an absolute reference to the individual time periods can be established, even if the RTC6 timer has been reset by `save_and_restart_timer`. The time stamp counter can be recorded by `set_trigger/set_trigger4` (signal 52) and is read-out by `get_value(52)`.

With `store_timestamp_counter` or `store_timestamp_counter_list` the current time stamp counter can be temporarily stored in an internal variable. You can then use `wait_for_timestamp_counter` to wait in a list for a fixed time difference to the stored time stamp counter.

### Notes

- `get_time` and `get_lap_time` only take list execution times into account.
- To compare the RTC6 board-internal `save_and_restart_timer` time measurement to an external time measurement via the `BUSY` pin, you should insert a `list_nop` between `save_and_restart_timer` and `set_end_of_list`. This ensures that a scanner delay is processed before `set_end_of_list`. Without `list_nop`, `save_and_restart_timer` includes the scanner delay in its measurement even though it completes only after `set_end_of_list` (and therefore the `BUSY` pin is already LOW).



## 9 Programming Peripheral Interfaces

Scan systems are often used in equipment that needs to synchronize processing by the laser and scan system with other process steps (for example, workpiece placement, robotic motion, process monitoring etc.).

For this purpose, the RTC6 PCIe Board provides a variety of peripheral interfaces, see [Chapter 4.6 "Interfaces for the Laser and Peripheral Equipment", page 62](#).

With the commands for programming these interfaces, you can supplementally and/or synchronously control the following in addition to lasers and scan systems:

- Signals transmitted for peripheral control
- Querying and evaluation of peripheral signals
- Control and synchronization of laser scan processes and peripheral control by external control signals

### 9.1 Signal Output

For peripheral control (for example, controlling a workpiece transport system or a shutter), appropriate signals can be outputted by the interfaces described below.

The output values can be changed at any time by control commands or – during processing of a list – by list commands.

#### 9.1.1 16-Bit Digital Output Port

The EXTENSION 1 socket connector provides a buffered 16-bit digital TTL output (DIGITAL OUT0...15). The level of its output signals must be configured with a jumper, see [Section "16-Bit Digital Input and 16-Bit Digital Output", page 67](#).

The `write_io_port_list`, `write_io_port`, `write_io_port_mask_list` and `write_io_port_mask` commands specify the digital output values. The output is in high-impedance mode until an initial value is assigned to it. In addition, the command `set_port_default` (`Port` = 3) can be used to define the value to be outputted at the 16-bit digital output port, as soon as processing of a list has ended with `stop_execution` or by an external stop signal. The default value also takes effect together with the position- and/or speed-dependent laser control (see `set_port_default`).

If "Automatic Laser Control" is activated with `Ctrl=6` from `set_auto_laser_control`, then the value at the 16-bit digital output automatically gets adjusted, see [Chapter 7.4.9 "Automatic Laser Control", page 186](#). You can log this by `set_trigger/set_trigger4` (signal 24).

When the output value is changed, a LATCH signal is outputted at the EXTENSION 1 socket connector as a trigger signal for synchronization of data transmission.

The `get_io_status` command reads the current value of the digital output port.

### 9.1.2 8-Bit Digital Output Port

The EXTENSION 2 socket connector provides a (jumper-configurable) buffered 8-bit digital output port (DATA0...DATA7), see [Section "8-Bit Digital Output Port", page 70](#).

Its output values can be set by `write_8bit_port` or `write_8bit_port_list`. The output is in high-impedance mode until an initial value is assigned to it. In addition, `set_port_default` (`Port` = 2) or `set_laser_off_default` can be used to define the value to be outputted at the 8-bit digital output port, as soon as processing of a list has ended with `stop_execution` or by an external stop signal. The default value also takes effect together with the position- and/or speed-dependent laser control (see `set_port_default`).

If "Automatic Laser Control" is activated with `Ctrl` = 3 from `set_auto_laser_control`, then the value at the 8-bit digital output automatically gets adjusted, see [Chapter 7.4.9 ""Automatic Laser Control""](#), page 186. This can be recorded by `set_trigger/set_trigger4` (signal 24).

When the output value is changed, a LATCH signal is outputted at the EXTENSION 2 socket connector as a trigger signal for synchronization of data transmission (provided that pin (17) has been correspondingly configured by the jumper setting, see [Section "8-Bit Digital Output Port", page 70](#).

### 9.1.3 2 Bit Digital Output Port

The LASER connector provides a buffered 2-bit digital output port (DIGITAL OUT1 and DIGITAL OUT2), see [Section "2-Bit Digital Output Port", page 63](#).

By `set_laser_pin_out` or `set_laser_pin_out_list`, values are assigned to this output port.

In addition, `set_port_default` (`Port` = 4) can be used to define the value to be outputted at the 2-bit digital output port, as soon as processing of a list is cancelled either by `stop_execution` or an external stop signal.

### 9.1.4 12-Bit Analog Output Ports

The LASER connector provides the two 12-bit analog output ports, ANALOG OUT1 and ANALOG OUT2, see [Section "12-Bit Analog Output Port 1 and 2", page 63](#). The ANALOG OUT2 output port is also available by the MARKING ON THE FLY socket connector, see [Section "Analog Output Port", page 71](#).

The output values can be separately set by `write_da_x` or `write_da_x_list`. In addition, `set_port_default` (`Port` = 0 or 1) or `set_laser_off_default` can be used to define the values to be outputted at the 12-bit analog output ports, as soon as processing of a list has ended with `stop_execution` or by an external stop signal. The default value also takes effect together with the position- and/or speed-dependent laser control (see `set_port_default`). If accordingly preset by corresponding commands, the values at the analog output ports are also changed:

- By a softstart, see [Chapter 7.4.7 "Softstart Mode \(not yet implemented\)"](#), page 184
- In pixel output mode, see [Chapter 8.7 "Pixel Output Mode – Marking Pixel Images \(Bitmaps\)"](#), page 249

If "Automatic Laser Control" is activated with `Ctrl` = 1 or `Ctrl` = 2 from `set_auto_laser_control`, then the value at one of the 12-bit analog output ports automatically gets adjusted, see [Chapter 7.4.9 ""Automatic Laser Control""](#), page 186. This can be recorded by `set_trigger/set_trigger4` (signal 24).



## 9.1.5 Controlling Stepper Motors

### Output Signals

The signals (ENABLE, DIRECTION and CLOCK) for controlling up to two stepper motors are outputted at the "STEPPER MOTOR" socket connector:

- You can appropriately change the ENABLE signal (to switch motor current on or off) during initialization by `stepper_init` and afterward by `stepper_enable` or `stepper_enable_list`.
- The RTC6 PCIe Board generates periodic CLOCK signal pulses (during reference movements by `stepper_init` and set-position movements by `stepper_abs` etc.). With each CLOCK pulse, the stepper motor executes a single step. You can adjust the CLOCK signal's pulse period (and thereby the speed of stepper motor motion) during initialization by `stepper_init` and afterward by `stepper_control` or `stepper_control_list` (the period is specified in units of 10 µs cycles).
- You can explicitly set the DIRECTION signal (and thereby the direction of stepper motor motion) during reference movements by `stepper_init`. In contrast, the DIRECTION signal is internally controlled during set-position movements by `stepper_abs` etc.: the signal gets set (to HIGH) if the next CLOCK pulse (in accordance with the defined set-position value) would increase the internal position variable. The DIRECTION signal also remains set for the cycles between two clock periods and even when the stepper motor has reached its set position. Only upon an actual change of direction does the DIRECTION signal correspondingly change in place of a CLOCK pulse. Here, output of the next CLOCK pulse is delayed by a full CLOCK pulse period (undefined truncation of CLOCK pulse periods never occurs).

### Notes

- For signal specifications, see [Chapter 4.6.7 "STEPPER MOTOR Socket Connector", page 77](#).
- For querying signals, see [Section "Querying Signals and Status Values", page 272](#).
- Stepper motor signals are outputted independently of any executing lists. A `set_end_of_list`, `pause_list`, `set_wait`, `stop_execution` or external stops do not terminate or pause a forwarding motion.
- For changes of direction or pulse period, the new values do not become active until an already-begun period is complete. Thus, pulse intervals are never be shorter than the currently defined value. For change of direction, an additional empty period (without CLOCK pulse) gets inserted.



## Reference Movements and Position Initialization

With `stepper_init`, you can initiate reference movements to limit switches. Here, the desired direction can be specified by the `Dir` parameter. To ensure that, despite mechanical play or long signal transit times, the reached end position still does not lie beyond the limit switch, you can define a tolerance value `Tol` that moves the stepper motor in the opposite direction after reaching the limit switch.

SCANLAB recommends executing a (fast) reference movement with a short CLOCK pulse period `Period` first and then a further (shorter but slower) reference movement with a longer CLOCK pulse period.

Once the reference movement has been successfully completed, the position variable (for the current position) is set to the value defined by parameter `Pos` as the reference for subsequent set-position movements. The reached reference position is offset from the limit switch position by  $\pm Tol$  (direction dependent).

During reference movement, the status is "Init" (`Bit #5 = 1`), see [Section "Querying Signals and Status Values", page 272](#). The limit switch position is traversed 4 times and a mean limit switch position is calculated from this. Then the stepper motor moves away from the limit switch by `Tol` steps in the opposite direction to `stepper_init Dir`. `Tol` must be large enough to overcome a possible hysteresis. During this set-position movement, see [Section "Set-Position Movements", page 271](#), the status is no longer "Init" but "Busy" (`Bit #4 = 1`). If you select `Pos = Tol` with `stepper_init`, the average position of the limit switch is 0. The only resulting inaccuracy is the fluctuation of the limit switch position measurement when the limit switch position is passed over four times.

If `Period = 0` and/or `Dir < 0`, then the reference movement does not occur. Instead, the current stepper motor position becomes the new reference position (with the value newly defined in `Pos` as the position variable).

Because `stepper_init` always stops a previously begun stepper motor movement, you could also use `stepper_init` as an emergency stop for the stepper motor.

Parallel execution of reference movements for both stepper motors is also possible, but cannot be simultaneously started through a single command.

## Set-Position Movements

Set-position movements can be initiated by the commands `stepper_abs`, `stepper_rel`, `stepper_abs_no` and `stepper_rel_no` or the corresponding list commands.

Specify absolute set-position values for `_abs` commands and relative values for `_rel` commands (always as CLOCK pulse units). `_no` commands only produce set-position movements for one stepper motor output, the other set-position commands do so for both stepper motor output ports simultaneously.

With `stepper_wait`, you can interrupt further execution of a list until a started stepper motor movement is completed.

The list commands for set-position movements are short list commands. Therefore, a stepper motor movement can also execute synchronously with a galvanometer scanner movement.

If the limit switch activates during a set-position movement, then the movement immediately aborts and cannot be resumed as a normal set-position movement. You either have to request a reference motion or mechanically or via the software, see below, deactivate the limit switch. If a stepper motor movement aborts once (for example, also by `Period = 0`), then the existing set position gets overwritten by the current position value. Therefore the stepper motor movement to the original set position cannot be resumed by eliminating the cause of interruption (limit switch or CLOCK pulse period = 0). Instead, it needs to be newly triggered.

To work around this behavior, the consideration of the limit switch can be deactivated by `stepper_disable_switch`. Then, the "release" can occur without carrying-out an initialization movement once again, even if a limit switch is present. The deactivation is especially useful, if a continuously rotating rotary axis is controlled by the stepper motor. During an initialization movement a possible deactivation of a limit switch is not considered.



## Querying Signals and Status Values

The current status of stepper motor signals (ENABLE, DIRECTION, CLOCK and SWITCH), the currently defined CLOCK pulse period and the current values of internal position variables for both stepper motors can be queried by `get_stepper_status`.

The `get_stepper_status` command also returns the Busy and Init statuses of both stepper motors. The Init status is set during reference movements and the Busy status during set-position movements.

As long as the Init status is set, no set-position commands (`stepper_abs`, `stepper_rel`, etc.) are permitted; control commands (except `stepper_init`) are denied with a `get_error` return code of `RTC6_BUSY` and list commands wait until the Init status gets reset. In some circumstances, the list itself or the movement process must be aborted.

## Terminating Infinite Movements

Depending on chosen parameters, very long or even infinite stepper motor movements can be initiated by `stepper_init`, `stepper_abs`, for example, if no limit switch exists in the specified direction or if a very large set-position value is combined with a long pulse period.

- If an infinite movement is started by a control command (for example, `stepper_abs`), then this control command completes at the latest when the positive time (in seconds) supplied for the `WaitTime` parameter has expired. However, the stepper motor's infinite movement itself continues and you need to separately abort it by setting the CLOCK pulse period (for example, by the control command `stepper_control`) to 0 (emergency stop) or by defining an appropriate new set position.
- If you start an infinite movement by a list command (for example, `stepper_abs_list`) and wait for its completion by `stepper_wait`, then further list execution is blocked as long as the infinite movement is not aborted by a control command such as `stepper_control` with `Period = 0` or an appropriate new set position. You could also abort the list by `stop_execution` or an external stop. But here, too, the stepper motor's infinite movement needs to be separately stopped with `stepper_control(Period = 0)`.

## WaitTime Parameter

The `WaitTime` parameter of the control commands can be used to set them to return to the calling program after the specified time (in seconds) has elapsed, regardless of whether the stepper motor movement is complete or not.

With `WaitTime = 0`, the command returns immediately. In this case, users must ensure that no unallowed commands are called. In particular, initialization should not be cancelled before it is complete. Otherwise, this leads to incorrect reference positions.



### 9.1.6 RS-232 Interface

The RS232 socket connector provides an RS-232 interface, see [Chapter 4.6.5 "RS232 Socket Connector", page 72.](#)

For configuring the RS-232 interface, see [Chapter 4.6.5 "RS232 Socket Connector", page 72.](#) `rs232_write_data` can be used to send single data words (bytes) to the RS-232 interface. Texts can be sent to the interface by `rs232_write_text` or `rs232_write_text_list`.

### 9.1.7 McBSP Interface

At the [McBSP interface](#), see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#), a 32-bit data word every 10 µs at DX0 pin (07) is permanently outputted.

The `set_mcbsp_out` and `set_mcbsp_out_ptr` commands allow selection of the signal types (analogously to `set_trigger`) to be outputted there:

- `set_mcbsp_out` lets you specify two signal types for simultaneous output at the [McBSP interface](#). A 16-bit portion of the first signal type is packed along with a 16-bit portion of the second signal type into a 32-bit data word for output every 10 µs. For a detailed description, see [set\\_mcbsp\\_out](#)
- `set_mcbsp_out_ptr` lets you define a list of up to 8 signal types. The signals are outputted sequentially in the specified order. For every 10 µs clock period, the lower 24 bits of the corresponding data signal and the associated signal type number (8 bits) are packed into a 32-bit data word and outputted at the [McBSP interface](#). For a detailed description, see [set\\_mcbsp\\_out\\_ptr](#).

#### Notes

- For signals and operating conditions, see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73.](#)
- In the default setting, the [McBSP interface](#) always outputs bits #4...19 of the Cartesian control values for the x and y axes (SampleX and SampleY) in a common 32-bit data word. This is equivalent to specifying `set_mcbsp_out(7, 8)`.
- Signals specified by `set_mcbsp_out` or are outputted (with `set_mcbsp_out_ptr` sequentially) until you call one of these two commands again.

## 9.2 Signal Input

Signals of peripherals (for example, signals of a transport system, workpiece recognition system or process monitoring camera) can be queried by the interfaces described below through control commands at any desired time or – during processing of a list – with list commands.

### 9.2.1 16-Bit Digital Input Port

The EXTENSION 1 socket connector provides a 16-bit digital TTL input (DIGITAL IN0...15), see [Section "16-Bit Digital Input and 16-Bit Digital Output", page 67](#), for receiving 16-bit digital values. For synchronization of data transmission, the EXTENSION 1 socket connector outputs a SYNC signal.

The `read_io_port` or `read_io_port_buffer` and `read_io_port_list` commands can be used to read the current value of the digital input port.

Further commands are provided for conditional command execution, see [Chapter 9.3.2 "Conditional Command Execution", page 281](#).

### 9.2.2 2-Bit Digital Input Port

For querying 2-bit digital values, the LASER connector provides a 2-bit digital input (DIGITAL IN1 and DIGITAL IN2), see [Section "2-Bit Digital Input Port", page 63](#).

Its input value can be read by `get_laser_pin_in`.

Further commands are provided for conditional command execution (see [Chapter 9.3.2 "Conditional Command Execution", page 281](#)).

### 9.2.3 RS-232 Interface

The RS232 socket connector provides an RS-232 interface for reading signals, see [Chapter 4.6.5 "RS232 Socket Connector", page 72](#).

For configuring the RS-232 interface, see [Chapter 4.6.5 "RS232 Socket Connector", page 72](#).

Data can be read in by `rs232_read_data`.

### 9.2.4 McBSP Interface

At the [McBSP interface](#), see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#), the 32-bit data word most recently fully transmitted to the specified memory location can be queried with `read_mcbsp`.

It is up to users whether to interpret it as one 32-bit value or two 16-bit values.

Signals (position values) received by the [McBSP interface](#) can also be integrated directly into Processing-on-the-fly correction of workpiece or scan-system motion (see [Chapter 8.6 "Processing-on-the-fly", page 227](#) and [Chapter 9.3.4 "Synchronization and Online Positioning by McBSP Signals", page 286](#)) or can be used for an [Online Positioning](#), see [Chapter 8.3.1 ""Local Online Positioning"", page 214](#) and [Chapter 8.3.2 ""Global Online Positioning"", page 217](#).

#### Notes

- For signals and operating conditions, see [Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73](#).

## 9.3 Control by External Signals

The previously described input and output of peripheral signals can be synchronized with scan system control and laser control, as follows:

- The related list commands can be inserted at appropriate locations in command lists.
- Execution of related control commands can be made dependent on the current status of list execution. For this, the list status can be requested by `read_status`, see [Chapter 6.4.2 "List Status", page 97](#) and the list execution status by `get_status`, see [Chapter 6.4.3 "List Execution Status", page 98](#).
- In addition, the **BUSY** list execution status is provided by the **BUSY OUT** signal:
  - at the LASER connector,  
see [Section "BUSY Status", page 63](#)
  - at the EXTENSION 1 socket connector,  
see [Section "BUSY Status", page 68](#)
  - at the MARKING ON THE FLY socket connector,  
see [Section "BUSY Status", page 71](#)

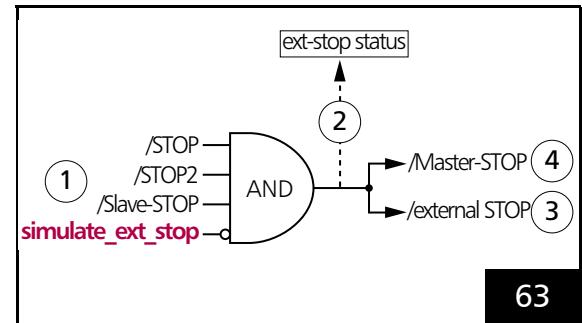
Moreover, the RTC6 PCIe Board provides commands and interfaces (described in the following sections) that allow external control signals (for example, from a light-barrier or from an encoder) to directly control and synchronize execution of command lists or individual commands (including the output of peripheral signals).

Moreover, the RTC6 PCIe Board provides interfaces to control and synchronize list execution directly with external signals.

### 9.3.1 Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization

#### External Stop

By a signal at the inputs `/STOP`, `/STOP2` or `/Slave STOP`, or by `simulate_ext_stop`, an external stop can be initiated, see (1) and (3) in [figure 63](#).



63

External stop. See text for description.

This:

- immediately cancels the currently executed list
- switches off the signals for "laser active" operation (but does not deactivate them)

Like after calling `stop_execution` (internal stop), the following output ports are then set to the previously defined (by `set_port_default`) stop-case values given these have not been defined as ≠ "-1":

- the 16-bit digital output port of the EXTENSION 1 socket connector
- the 8-bit digital output port (DATA0 to DATA7) of the EXTENSION 2 socket connector
- the 2-bit digital output port
- the two 12-bit analog output ports (ANALOG OUT1 and ANALOG OUT2) of the LASER connector

The inputs for external stop signals (1) are always unlocked so that an external stop can occur at *any* time (emergency stop).

The /STOP input is accessible by the 15-pin D-SUB LASER connector, see [Section "External Control Signals", page 63](#), the /STOP2 input at the MARKING ON THE FLY socket connector, see [Section "External Control Signals", page 71](#). Both signal inputs are internally connected to +3.3 V by pull-up resistors (4.7 kΩ). Both inputs are TTL active-LOW and level sensitive. A list abort is triggered as soon as at least one of the two inputs is set to LOW (that is, to 0 V or ground) for at least 10 μs.

If an RTC6 PCIe Board is connected to other boards by the Master or Slave connector, see [figure 4](#), then external stops from the board receiving the signal are passed on to *all* boards connected to it given the respective master/slave interface has been configured (by [master\\_slave\\_config](#)) accordingly. See also [Chapter 6.6.3 "Master/Slave Operation", page 114](#).

Stops triggered by [stop\\_execution](#) are *not* passed on. In contrast, external stops triggered by [simulate\\_ext\\_stop](#) are passed through.

By [get\\_startstop\\_info](#) the current stop status (that is, whether one of the inputs is currently set to LOW) (2) can be queried and whether or not a new external stop has occurred since the last query.

## External Start

By a signal at the inputs /START, /START2 or /Slave-START, or by the commands [simulate\\_ext\\_start](#) or [simulate\\_ext\\_start\\_ctrl](#), an external start can be initiated (see (1), (5) and (7) in [figure 64](#)).

This starts execution at the beginning of "List 1". But the commands [set\\_extstartpos](#) or [set\\_extstartpos\\_list](#) also allow pre-selection of another absolute start address. A list is only started if neither the [BUSY](#) status (as during list execution) nor the [INTERNAL-BUSY](#) status (as for example, with [goto\\_xy](#)) nor the [PAUSED](#) status (after [pause\\_list](#), [stop\\_list](#) or [set\\_wait](#)) is set at the moment.

Before the /START, /START2 or /Slave-START inputs (1) can be used, they must be enabled by [set\\_control\\_mode](#) (3).

The control command [simulate\\_ext\\_start\\_ctrl](#) can be deactivated by [set\\_control\\_mode](#)( Bit #4 = 1 ). The list command [simulate\\_ext\\_start](#) still remains active.

The /START input is accessible by the 15-pin D-SUB LASER connector, see [Section "External Control Signals", page 63](#), the /START2 input at the MARKING ON THE FLY socket connector, see [Section "External Control Signals", page 71](#). Both signal inputs are internally connected to +3.3 V by pull-up resistors (4.7 kΩ). Both inputs are TTL active-LOW and edge sensitive (HIGH to LOW level transition). A start is triggered – after activation by [set\\_control\\_mode](#) – as soon as one of the three input signals changes from HIGH to LOW (that is, to 0 V or ground).

If an RTC6 PCIe Board is connected to other boards by the Master or Slave connector, see [figure 4](#), then external starts from the board receiving the signal are passed on to *all* boards connected to it given the respective master/slave interface has been configured (by [master\\_slave\\_config](#)) accordingly. See also [Chapter 6.6.3 "Master/Slave Operation", page 114](#).

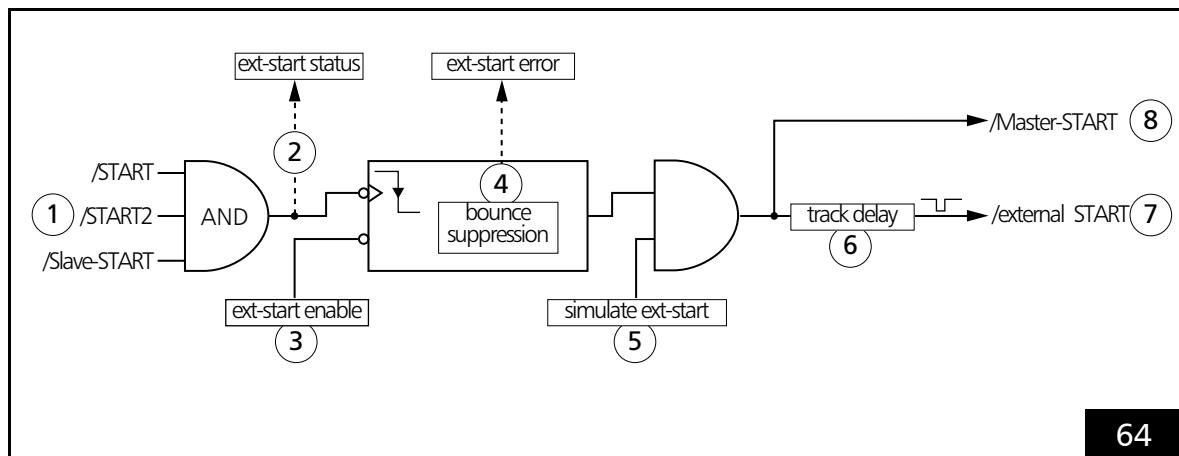
Internal starts triggered by [execute\\_list](#) or [execute\\_at\\_pointer](#) are *not* passed on. In contrast, external starts triggered by [simulate\\_ext\\_start](#) or [simulate\\_ext\\_start\\_ctrl](#) are passed on, see also [Chapter 6.6.3 "Master/Slave Operation", page 114](#).

**get\_startstop\_info** queries the current start status (that is, whether one of the inputs is set to LOW) (2) and whether a list has successfully started since the last query.

**bounce\_supp** enables debouncing of start signals received at the /START, /START2 or /Slave-START inputs (4). Start signals occurring within the defined debouncing time after a successful start signal are thereby suppressed. **get\_marking\_info** queries whether a start signal has been suppressed (4).

**simulate\_ext\_start\_ctrl** can be used to start by control command a synchronous start of master/slave-synchronized boards.

The list command **simulate\_ext\_start** can be used to trigger further starts at defined intervals after the successful one-time external start (see below).



External start. See text for description.

## External Start with Track Delay

For many applications (for example, if a workpiece must be initially transported from the light barrier to the scan system), the start must be delayed with reference to the triggering start signal.

For this purpose, the commands `set_ext_start_delay`, `set_ext_start_delay_list` or `simulate_ext_start` allow configuring a track delay (see (6) in figure 64) that postpones execution of a start relative to the triggering input signal or corresponding command. The track delay is specified in counting units of an internal encoder (encoder-counter) that itself can be triggered by an external or simulated encoder signal, see [Chapter 9.3.3 "Synchronization by Encoder Signals", page 284](#).

External starts triggered by an external start signal or by `simulate_ext_start` or `simulate_ext_start_ctrl` that do not execute immediately because of the track delay setting are held in a queue that can accommodate up to 8 starts (each start trigger is automatically generated when the delay has expired). This can be useful, for instance, when processing multiple workpieces transported to the scan system (even) at irregular intervals: here, up to 8 workpieces can simultaneously reside within the track delay (distance between the light barrier and scan system). If more external starts are triggered than can be simultaneously held in the 8-start wait loop, then an error bit is set, which can be queried by `get_startstop_info` (Bit #11). If a track delay is set, then any previous queue is canceled (see `set_ext_start_delay` and `simulate_ext_start`).

By `set_control_mode` (Bit #1 = 1) it can be set that the queue with the external start entries get explicitly cancelled upon an external stop. With `set_control_mode` (Bit #1 = 0) the queue remains existing after an external stop.

## Notes

- Note that the /START, /START2 and /Slave-START inputs are *edge* sensitive (HIGH to LOW level *transition*), whereas the /STOP, /STOP2 and /Slave-STOP inputs are *level* sensitive.
- An explicit call of the command `stop_execution` disables the /START, /START2 and /Slave-START inputs. An external stop signal also (at least temporarily) disables these inputs, that is, as long as one of the inputs /STOP, /STOP2 or /Slave-STOP is LOW. `set_control_mode` can be used to define whether or not the /START, /START2 and /Slave-START inputs also stay disabled when the external stop signal is no longer active.
- `set_control_mode` additionally allows activation or deactivation of the inputs /START, /START2 or /Slave-START and deactivation of track delay.
- External starts are also suppressed after `pause_list`, `stop_list` or `set_wait` (PAUSED status is set). `restart_list`, `stop_execution`, `release_wait` or an external stop ends suppression of the start.
- If list inputs are not yet finished, a buffer flush should be initiated before an external start, for example, by `set_input_pointer( get_input_pointer() )`, so that any still buffered list commands are fully transferred to list memory, see [Chapter 6.4.1 "Loading Lists", page 95](#).
- If a master board is started internally (for example, by `execute_list_pos`) and subsequently a slave board by `simulate_ext_start`, then the master and slave boards do not run synchronously if a home jump was previously activated by `home_position` or `home_position_xyz`: the home return executes on the master board *before* `simulate_ext_start` starts the slave board, but executes on the slave board *afterward*. While the home return executes on the slave board, the master board continues running. This asynchronicity does not occur if all boards are started by an external start signal (or by `simulate_ext_start` or `simulate_ext_start_ctrl`) or if no home jump is activated.

### Regular (Periodic) External Starts

By `set_control_mode` and `set_control_mode_list` (Bit #10), equidistant external starts can be created that are independent of the time point of the start trigger as long as they occur within the specified track delay. This strongly periodic list processing is – independently of a list's actual duration of execution and the exact time point of the external start – exactly synchronized to the 10 µs clock of the RTC6 PCIe Board.

If desired, set Bit #10 = 1 (`Mode|Bit#10`) to configure the internal encoder-counter's processing so that the track delay of an external start is *not* counted only beginning with the time point of the triggering external start signal or `simulate_ext_start` (Bit #10 = 0) but already beginning with the most recently executed external start (also executed by an external start signal or `simulate_ext_start`), see [figure 65](#). This makes the distance between consecutive external starts (in encoder pulses) constant.

For activation of this mode, an external start must have successfully occurred (only one-time) in mode Bit #10=0 (`Mode &~Bit#10`). Each subsequent external start must be requested within the specified track delay.

Example in PASCAL of a typical command sequence without use of an external start signal:

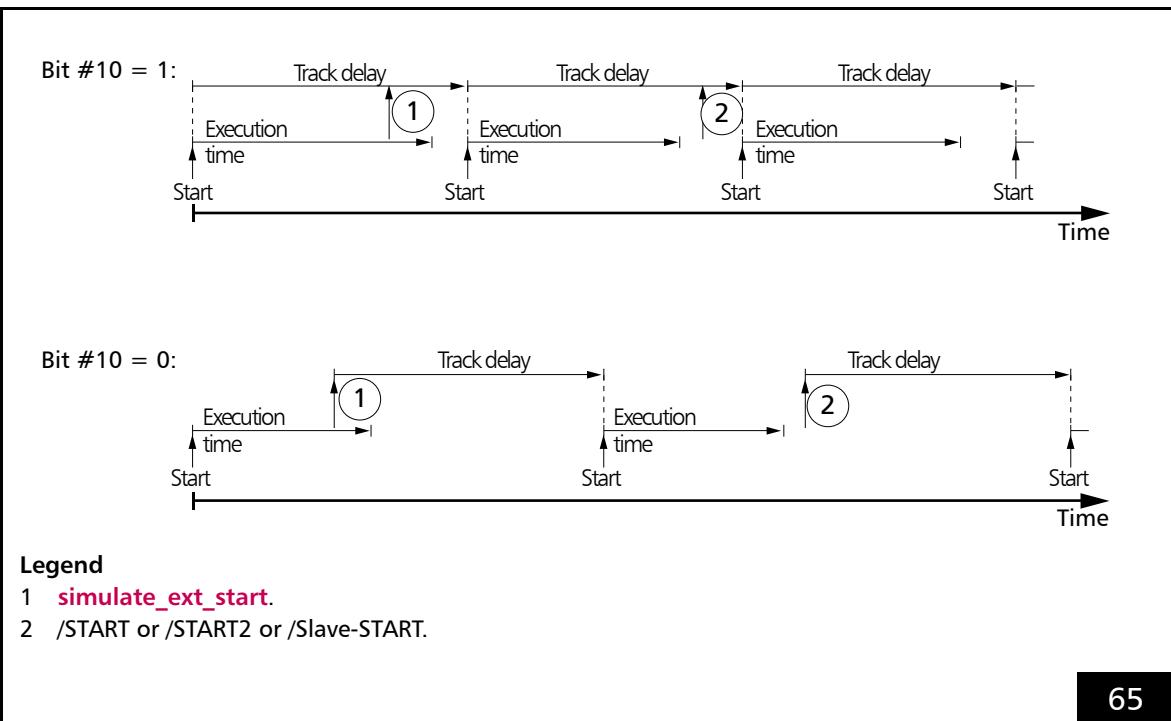
```
set_control_mode(Mode &~Bit#10);
// (one-time) reset (disable) Bit #10
// (initialization)
set_start_list_pos(ListNo, Pos);
// open some list
// afterward: some commands
simulate_ext_start(Delay,EncoderNo);
// first time start in mode Bit #10 = 0,
// otherwise in mode Bit #10 = 1
set_control_mode_list(Mode|Bit#10);
// set Bit #10 = 1
// afterward: further commands
set_end_of_list;
// close the list
execute_list_pos(ListNo,Pos);
// (one-time) start the list
```

If the first start is to be triggered externally (for example, by /START or by `simulate_ext_start_ctrl`) rather than by an `execute_list_pos` command, but all subsequent starts triggered by `simulate_ext_start`, then `set_control_mode_list` in the above example must be called before `simulate_ext_start`.

After setting `set_control_mode` (`Bit #1 = 1`), the external start queue's entries get explicitly cancelled upon an external stop (thereby, external starts can be permanently stopped by an external stop).

For `set_control_mode` (`Bit #1 = 0`) (default setting) – after an otherwise infinitely repetitive series has been stopped (for example, by `set_control_mode` (`Bit #0 = 0`) – you should deactivate the track delay and cancel the queue of not-yet-executed external starts by `set_ext_start_delay` (`Delay = 0`). Otherwise, the next "equidistant" external start does not have the correct gap. `set_control_mode` (`Bit #2 = 1`) alone is not sufficient for termination, because the track delay is reactivated by any not-yet-executed `simulate_ext_start` command.

If a further external start is missed within the track delay, then you should delete the wait loop (otherwise the encoder counter needs to run through a full 31-bit sequence before a start can again be successfully triggered). Deletion can be accomplished by resetting the track delay with `set_ext_start_delay`. In any case, Bit #10 needs to first be reset (disabled) for initialization and then a (one-time) external start must be triggered (for external start signals Bit #0 must be set) before Bit #10 can again be set (see example). Otherwise, the first track delay in the wait loop is undefined.



Regular and irregular external starts (see text for description).

### 9.3.2 Conditional Command Execution

The so-called conditional commands allow the execution of individual list commands to be made dependent on external control signals.

The conditional commands read out the current value at the 16-bit digital input port at the EXTENSION 1 socket connector, see [Section "16-Bit Digital Input and 16-Bit Digital Output", page 67](#), and their execution depend on the read out value:

- Conditional Jumps  
`list_jump_pos_cond` (synonymous with `list_jump_cond`) and `list_jump_rel_cond` result in either a jump within a buffer area or no jump. The thereby specified jump addresses must fulfill the same conditions as with `list_jump_pos` and `list_jump_rel`
- Variable-distance jump  
`switch_ioport` executes a relative list jump
- Conditional Calls of Non-Indexed Subroutines  
`list_call_cond` and `list_call_abs_cond` either call or do not call a non-indexed subroutine at a specified memory address
- Conditional Calls of Indexed Subroutines  
`sub_call_cond` and `sub_call_abs_cond` either call or do not call – depending on the queried value – an indexed subroutine with a specified index
- Conditional output of peripheral signals  
`set_io_cond_list` and `clear_io_cond_list` associate the output value of the 16-bit digital output port at the EXTENSION 1 socket connector, see [Section "16-Bit Digital Input and 16-Bit Digital Output", page 67](#), directly with the signals at the digital input port: individual bits of the output port are set or cleared

- Conditional execution of any desired list commands:

`if_cond` and `if_not_cond` have no effect if the condition for the queried value is fulfilled or not. Otherwise, they result in skipping the next list command. In this way, all list commands can be executed conditionally.

Example: the command sequence

`if_cond(...)`

`list_call(...)`

is functionally identical to

`list_call_cond(...)`.

The execution of any desired list command can also be made dependent on the current value at the 2-bit digital input port of the LASER connector. For this, `if_pin_cond` and `if_not_pin_cond` are available. These are functionally similar to the commands `if_cond` and `if_not_cond`. Reliable functioning of these conditional commands requires that the signals at the 2-bit digital input port remain unchanged for at least 10 µs.

A condition for the 16-bit digital input port of the EXTENSION 1 socket connector can be defined by the control command `set_pause_list_cond`. If this condition is met, a currently executed list is paused by an automatically set `pause_list`. The list can only be resumed by `restart_list`. The condition is checked once per 10 µs clock period. A conditional `pause_list` takes precedence over a simultaneously present /STOP signal. `set_pause_list_not_cond` does the same as `set_pause_list_cond`, if the specified condition is not fulfilled.



## Example Code (PASCAL)

### (1) Confirm a signal:

```
set_start_list(1);
...
// set bit #0 of the 16-bit digital output port
set_io_cond_list(0, 0, 1);
// loop until the signal is confirmed (i.e. bit #0 of the digital input turns HIGH)
list_jump_rel_cond(0, 1, 0);
// clear bit #0 of the 16-bit output
clear_io_cond_list(0, 0, 1);
// loop until the signal is confirmed
list_jump_rel_cond(1, 0, 0);
...
set_end_of_list;
execute_list(1);
```

### (2) If the lower four bits of the digital input have the value (0110), set Bit #1 of the 16-bit digital output.

Otherwise clear Bit #1:

```
set_start_list(2);
...
// RTC4 style: list_jump_cond($0006, $0009, get_input_pointer + 3);
// this command uses absolute addresses and is not relocatable
// the following RTC6 command uses relative addresses and is relocatable:
// skip the next two commands, if the state
// of the 16-bit input is (xxxx xxxx xxxx 0110)
// list_jump_rel_cond($0006, $0009, 3);

// clear bit #1 of the 16-bit output and ...
clear_io_cond_list(0, 0, 2);
//RTC4 style: set_list_jump(get_input_pointer + 2);
//this command uses absolute addresses and is not relocatable
//the following RTC6 command uses relative addresses and is relocatable
// ... skip the next command
list_jump_rel(2);

// set bit #1 of the 16-bit output
set_io_cond_list(0, 0, 2);

// (continue)
...
set_end_of_list;
execute_list(2);
...
bit1 := (get_io_status AND $0002)           // returns the current state of bit #1
```



**(3) Choose between 15 small subroutines at defined memory addresses:**

```
...
for i := 1 to 15 do
    // call subroutine at address i*100, if [bit #3..bit #0] (binary) = i
    list_call_cond(i, 15-i, i*100);
...
```

**(4) Choose between 15 indexed subroutines:**

```
...
for i := 1 to 15 do
    // call subroutine with index i, if [bit #3..bit #0] (binary) = i
    sub_call_cond(i, 15-i, i);
...
```

### 9.3.3 Synchronization by Encoder Signals

#### Intended Use

When processing moving workpieces, the laser scan processes need to be adapted to the current workpiece position.

To incorporate the current workpiece position, the RTC6 can evaluate signals of up to two user-supplied incremental encoders. Though incremental encoders do not register the current workpiece position, they register the motion of the transport system (conveyor belt, rotating plate, etc.)<sup>(1)</sup>: For each transport motion, they provide signals (depending on the direction of movement) to the RTC6 which can result in incrementing or decrementing of its two internal encoder counters. The states of the RTC6's encoder counters thereby correspond directly to the position of the workpiece<sup>(2)</sup>.

If workpieces are always processed at a constant speed and an encoder is therefore not mandatory, then the encoder signals can also be simulated by **simulate\_encoder**, so that the encoder counters are incremented with a constant counting rate of 1 MHz.

The current counts of both encoder counters can be queried by the control command **get\_encoder**. Alternatively, they can be stored in a buffer on the RTC6 by the list command **store\_encoder** and then retrieved from there by the control command **read\_encoder**. In addition, the RTC6 automatically evaluates the current counts if execution of the laser scan processes is controlled as follows:

- For Processing-on-the-fly-applications (see [Chapter 8.6 "Processing-on-the-fly", page 227](#)), the coordinate values of all vector and arc commands are transformed in accordance with the current encoder counts.
- By the list command **wait\_for\_encoder\_mode**, further execution of a list can be postponed until the selected encoder counter has overstepped or understepped a pre-defined value.
- With 2D movements, **wait\_for\_encoder\_in\_range** waits until the encoder values are within a given rectangle.
- For external starts, a track delay can be defined by **simulate\_ext\_start**, **set\_ext\_start\_delay** or **set\_ext\_start\_delay\_list** for postponing execution of a list start relative to the triggering input signal or corresponding command, see [Section "External Start", page 276](#).
- Encoder-speed-dependent "Automatic Laser Control", see [Section "Encoder-Speed-Dependent Laser Control", page 192](#), uses the current encoder speed (counter pulses in the most recent 10 µs interval) of an encoder counter to control a laser signal parameter.

- (1) The actual workpiece position can also be forwarded by the **McBSP interface** to the RTC6 PCIe Board (see [Chapter 9.3.4 "Synchronization and Online Positioning by McBSP Signals", page 286](#)).
- (2) The encoder counters are signed 32-bit counters. Upon reaching the maximum (minimum) counter value, counting continues with the minimum (maximum) value. A counter reset only occurs if triggered by Processing-on-the-fly-commands (see **set\_fly\_x**, **set\_fly\_y**, **set\_fly\_rot**). **set\_control\_mode**( Bit #9 = 1 ) can be used to precisely synchronize the encoder reset with external start signals.

## Inputs for External Encoder Signals

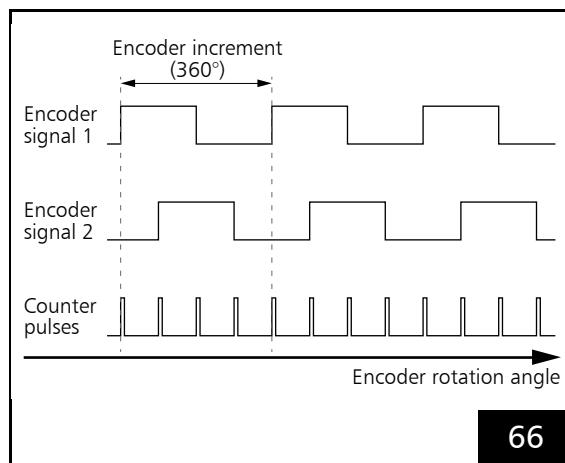
For receiving encoder signals, the MARKING ON THE FLY socket connector provides two encoder inputs (ENCODER X and ENCODER Y), see [Section "Encoder Inputs", page 71](#).

For linear movements of the parts to be processed, up to two user-supplied incremental encoders (that define, independently from each other, the workpiece motion in the x and y directions) can be connected to these inputs.

For rotational movements only one incremental encoder is necessary. For Processing-on-the-fly applications, it must be connected to the ENCODER X input.

Each encoder input is designed for a pair of standardized differential input signals (RS-422; HIGH level  $\geq 2.0$  V, LOW level  $\leq 0.8$  V).

The timing diagram of a typical encoder signal pair shows [figure 66](#). The second encoder signal is usually phase-shifted by 90° relative to the first signal. The internal RTC encoder counter triggers at each edge of both signals, that is, one encoder increment results in four counter pulses (counts). The relative 90° phase-shift of the two signals allows the RTC6 PCIe Board to detect the movement direction of the workpiece as well. Depending on the direction of movement, the counter value is increased or decreased.



The signals of encoder input ENCODER X trigger encoder counter "Encoder0", the signals of encoder input ENCODER Y trigger encoder counter "Encoder1".

The encoder increments should not exceed the maximum allowed frequency of 4 MHz (16 encoder signal edges /  $\mu$ s).

## Encoder Simulation

The encoder simulation can be activated (deactivated) by [simulate\\_encoder](#). The RTC6 PCIe Board provides a 1 MHz clock signal (counter pulses), which replaces the signals of an external incremental encoder.

Timing diagram of a typical encoder signal pair and of the corresponding RTC5 counter pulses.

### 9.3.4 Synchronization and Online Positioning by McBSP Signals

Instead of incremental encoders, the movement between the workpiece and the scan system can be synchronized using absolute position data via the [McBSP interface](#).

Alternatively, this interface also allows the alignment of the workpiece relative to the (stationary) scan system ("Online Positioning").

The input value most recently fully transmitted at the [McBSP interface](#) can be queried by [read\\_mcbsp](#). In addition, the RTC6 PCIe Board automatically evaluates the current input value if execution of the laser scan processes is controlled as follows:

- For Processing-on-the-fly-applications (see [Chapter 8.6 "Processing-on-the-fly", page 227](#)), the coordinate values of all vector and arc commands are transformed in accordance with the current input value.
- By the list command [wait\\_for\\_mcbsp](#), further execution of a list can be postponed until the input value has reached, overstepped or understepped a pre-defined value.
- By [Online Positioning](#) (see [Chapter 8.3.1 ""Local Online Positioning""](#), page 214 and [Chapter 8.3.2 ""Global Online Positioning""](#), page 217), the scan system is aligned in accordance with the current input values.

#### Notes

- To compensate linear motion, Cartesian coordinates can be forwarded by the [McBSP interface](#). Compensation of rotational motion, on the other hand, requires transmission of angle positions. During evaluation, full revolutions are suppressed as long as the input values do not exceed the allowed value range (20 full circles, see [set\\_mcbsp\\_rot](#)).
- The signals at the [McBSP interface](#) have no impact on the track delay, which can be defined for external starts (by [simulate\\_ext\\_start](#), [set\\_ext\\_start\\_delay](#) or [set\\_ext\\_start\\_delay\\_list](#)).



## 9.4 Periodical I/O Signals

By **periodic\_toggle** and **periodic\_toggle\_list**, periodically repeated signals can be outputted at a selectable IO port:

- ANALOG OUT1 output port, see [Section "12-Bit Analog Output Port 1 and 2", page 63](#)
- ANALOG OUT2 output port, see [Section "12-Bit Analog Output Port 1 and 2", page 63](#)
- 8-bit digital output port, see [Section "8-Bit Digital Output Port", page 70](#)
- 16-bit digital output port, see [Section "16-Bit Digital Input and 16-Bit Digital Output", page 67](#)
- 2-bit digital output port, see [Section "2-Bit Digital Output Port", page 63](#)

Thereby, for example, external peripheral equipment can be triggered synchronously to list execution.

### Notes

- At a **set\_end\_of\_list**, **stop\_execution** or external /STOP the periodical signals continue, even if they have been activated by the list command **periodic\_toggle\_list**.
- **periodic\_toggle** and **periodic\_toggle\_list** toggle endless with **Count =  $2^{32}-1$** .



## 10 RTC6 Commands

### 10.1 Overview

The following pages describe the complete RTC6 command set (control commands and list commands). The commands are listed according to their intended use. The page numbers refer to [Chapter 10.2 "RTC6 Command Set", page 301](#), where the RTC6 commands (alphabetically ordered) are explained in detail.

#### 10.1.1 Nomenclature

##### Multi-Board Commands

All commands marked (n\_) in the following list also exist in a version for multiple RTC6 PCIe Boards installed in one computer. See [Chapter 6.6 "Using Several RTC6 PCIe Boards in One PC", page 113](#) for detailed information about these *multi-board* commands.

Nearly all single-board commands are also available in multi-board form. Exceptions are explicitly noted in the description list (in [Chapter 10.2 "RTC6 Command Set", page 301](#)).

##### Normal, Short, Variable and Multiple List Commands

The list commands of the RTC6 command set vary somewhat in their length of command execution. To differentiate, list commands in the list description (in [Chapter 10.2 "RTC6 Command Set", page 301](#)) are designated as "normal", "short", "variable" and "multi".

- Normal list commands require a full 10 µs clock period for command execution.
- Short list commands require less time for command execution. Therefore, they can be carried out along with the next list command, one directly after the other within a single 10 µs clock period. In contrast, a short list command that immediately follows a normal list command executes in the subsequent 10 µs clock period.

The quicker execution of short list commands reduces total list processing time. In addition, during a **Polyline**, the laser power can, for instance, be varied or the IO ports can be addressed (see **write\_da\_x\_list**) between the **Polyline**'s individual **Mark commands** (see **set\_laser\_pulses**), all without interrupting the **Polyline** (the laser remains on).

In contrast, if a specific time behavior is desired (10 µs clock period), you can insert an additional **list\_nop** or **list\_continue** command after any short list command to ensure that the next command only executes in the following 10 µs clock period. Insertion of **list\_nop** (but not insertion of **list\_continue**) results in the interruption of the **Polyline** (the "laser active" laser control signals are switched off).

Up to 8 short list commands per 10 µs clock period are possible. However, the maximum number can be lower, depending on the workload of the RTC6 board and the **DSP mode**. Short list commands that alter the output pointer (for example, **sub\_call**, **list\_return** or **list\_jump\_pos**) count as two commands. If the maximum number is exceeded, a 10 µs clock period is inserted (equivalent to an additionally inserted **list\_continue**, during the **Polyline** the laser remains on).

A maximum of two short list commands per 10 µs clock period are allowed before a normal list command. If a normal list command succeeds more than two short list commands, then the short list commands execute immediately and the normal list command execute delayed by a 10 µs clock period.

The maximum number of up to 8 short list commands may change in the future. For fully future-safe applications, only one short list command should precede a normal list command. If necessary, you should explicitly insert a **list\_continue** or **list\_nop** (**list\_nop** interrupts the **Polyline**).

- The command execution lengths of variable list commands are dependent on additional parameters and the user program. Details are provided in the corresponding command description.
- About multiple list commands, see [Section "Multiple List Commands", page 290](#).

#### Notes

- The total execution time of normal and variable list commands equals the sum of the command execution time and the execution time of the process initiated by the command. A subsequent list command only runs after this total execution time has completed. Example: the total execution time of the normal list command **mark\_abs** is 10 µs (command execution time) + the execution time of the marking process. The latter is dependent on the settings of such parameters as marking length, mark speed, and delays etc.

#### Undelayed Short List Commands, Delayed Short List Commands

Most short list commands (for example, **list\_jump\_pos**, **list\_call**, **sub\_call** or **list\_return**) execute before the next list command and prior to a possible scanner delay (jump, mark or polygon delay). These commands are called “undelayed short list commands” in the corresponding command descriptions (in [Chapter 10.2 “RTC6 Command Set”, page 301](#)).

However, some short list commands only execute after the respective scanner delay (that is, directly before the next command). Such commands are called “delayed short list commands” in the corresponding command descriptions. These include commands that immediately affect output or laser power (for example, **set\_rot\_center\_list**, **set\_wobbel\_mode**, **write\_da\_x\_list**, **set\_laser\_pulses**, **set\_standby\_list**, **set\_mark\_speed**, **set\_encoder\_speed**) or commands that affect data acquisition or time measurement (for example, **set\_trigger** or **save\_and\_restart\_timer**).

Without this delayed execution, such commands would, for example, result in a laser power change already being effective at the end of a **Mark command** (that is, during a still-active mark delay or polygon delay) and not at the beginning of the following **Mark command**.

If such a power changing command, such as a peripheral output or **set\_laser\_power**, is called immediately before a **set\_end\_of\_list**, it is not executed because **set\_end\_of\_list** “clears up” the laser control. Add a **list\_nop** to be sure that the command is actually executed.

**set\_trigger** and **save\_and\_restart\_timer** would erroneously take into account the delay of a preceding command instead of the delay of the subsequent command.

If these commands are directly before a **set\_end\_of\_list**, add a **list\_nop** so that they are still executed within the list.

For several short list commands in a row, even a “delayed short list command” only executes delayed if no further “delayed short list commands” directly follow.



With several “delayed short list commands” in a sequence of short list commands, only the last “delayed” command can actually be executed delayed. All others before that are executed immediately (yet before a scanner delay).

When you sequence the commands, make sure to place the most important “delayed” command at the end, especially within a [Polyline](#).

Alternatively, you can also explicitly initiate processing of the scanner delay (for example, by [list\\_nop](#)), so that all subsequent short list commands in fact always execute “delayed”.

If a normal list command succeeds more than two short list commands, then the normal list command is executed delayed by a 10 µs clock period.

### Multiple List Commands

A multiple list command has multiple components that accordingly occupy multiple list storage positions. The initial components are always undelayed short list commands. The final component is always a short, normal or variable list command. All components immediately execute successively. Any still-pending delayed short list commands execute first. The RTC6 command set currently only contains two-component multiple list commands, for example, [wait\\_for\\_encoder\\_in\\_range](#) or [set\\_pixel\\_line\\_3d](#).

### 10.1.2 Compatibility

For RTC6 customers who previously used the RTC4 (RTC5), the command descriptions (in [Chapter 10.2 “RTC6 Command Set”, page 301](#)) note in each command’s “RTC4→RTC6” (“RTC5→RTC6”) section:

- to what extent a command differs from that of the RTC4 (RTC5),
- whether the command is new to the command set,
- whether and how parameter values are converted in [RTC4 Compatibility Mode](#) ([RTC5 Compatibility Mode](#)).

Some RTC3/RTC4 commands and a few RTC2 commands emulated by the RTC3/RTC4 are not supported by the RTC6, see [Chapter 10.3 “Unsupported RTC4/RTC5 Commands”, page 817](#).

Further information on changing from the RTC4 board to RTC6 board see [RTC4 Compatibility Mode](#) and [Chapter 2.7 “Notes for RTC4 Users”, page 38](#).

Further information on changing from the RTC5 board to RTC6 board see [RTC5 Compatibility Mode](#) and [Chapter 2.8 “Notes for RTC5 Users”, page 47](#).



### 10.1.3 Version Information

Descriptions for a number of commands include version information, listed under "Version info":

- For newly added commands: the software version in which they become available
- For some older commands: information on implemented changes.

All new commands and changes to old commands are also listed in the file

RTC6\_Software\_RevisionHistory\_<Date>.pdf included in the RTC6 software package.

### 10.1.4 Optional Functions

The RTC6 PCIe Board can be configured for functionality not available in the standard version, see [Chapter 2.3 "Options", page 30](#).

If this extended functionality is not enabled or installed by SCANLAB, then the associated commands have only partial or non-existent effect:

- Without enabled [Option Processing-on-the-fly](#), commands to activate a Processing-on-the-fly correction have no effect
- Without enabled [Option "3D"](#), 3D vector commands are executed, however, no z axis signals are outputted
- Without enabled [Option "Second Scan Head Control"](#), commands for which the scan head connector can be explicitly specified have not effect on the second scan head connector (for example, [set\\_matrix](#))

If applicable, the individual command descriptions contain corresponding notes.



## 10.1.5 Control Commands

### Initialization of the RTC6 DLL

free_RTC6_dll .....	378
get_RTC_mode .....	405
init_RTC6_dll .....	442
set_RTC4_mode .....	691
set_RTC6_mode .....	693

### Using Several RTC6 PCIe Boards in One PC

acquire_RTC .....	302
release_RTC .....	563
rtc6_count_cards .....	570
select_RTC .....	578

### List Memory Commands

(n_) config_list .....	330
(n_) get_config_list .....	382
(n_) get_list_space .....	399
(n_) load_disk .....	475
(n_) save_disk .....	572

### Board Initialization and Field Correction

(n_) get_head_para .....	390
(n_) get_sync_status .....	416
(n_) get_table_para .....	418
(n_) load_correction_file .....	471
(n_) load_program_file .....	486
(n_) load_stretch_table <sup>(1)</sup> .....	489
(n_) load_z_table <sup>(1)</sup> .....	495
(n_) load_z_table_no <sup>(1)</sup> .....	496
(n_) number_of_correction_tables .....	526
read_abc_from_file .....	551
(n_) select_cor_table .....	575
(n_) set_dsp_mode .....	601
(n_) sync_slaves .....	765
write_abc_to_file .....	808

### Laser Mode and Parameters

(n_) config_laser_signals .....	328
(n_) get_standby .....	409
(n_) load_auto_laser_control .....	468
(n_) load_position_control .....	484
(n_) set_auto_laser_control .....	585
(n_) set_auto_laser_params .....	589
(n_) set_encoder_speed_ctrl .....	604
(n_) set_firstpulse_killer .....	609
(n_) set_laser_control .....	639
(n_) set_laser_mode .....	643
(n_) set_laser_pulses_ctrl .....	647
(n_) set_pulse_picking .....	687
(n_) set_pulse_picking_length .....	687
(n_) set_qswitch_delay .....	688
(n_) set_softstart_level .....	705
(n_) set_softstart_mode .....	706
(n_) set_standby .....	707
(n_) spot_distance_ctrl .....	743

### Setting the Scanner Parameters

(n_) load_varpolydelay .....	493
(n_) set_delay_mode .....	599
(n_) set_jump_speed_ctrl .....	637
(n_) set_mark_speed_ctrl .....	651
(n_) set_sky_writing .....	698
(n_) set_sky_writing_limit .....	699
(n_) set_sky_writing_mode .....	700
(n_) set_sky_writing_para .....	702

### Coordinate Transformations

(n_) set_angle .....	582
(n_) set_defocus <sup>(1)</sup> .....	596
(n_) set_defocus_offset <sup>(1)</sup> .....	597
(n_) set_matrix .....	652
(n_) set_offset .....	675
(n_) set_offset_xyz <sup>(1)</sup> .....	676
(n_) set_scale .....	694

### Online Positioning

(n_) apply_mcbsp .....	312
(n_) set_mcbsp_global_matrix .....	656
(n_) set_mcbsp_global_rot .....	657
(n_) set_mcbsp_global_x .....	658
(n_) set_mcbsp_global_y .....	659
(n_) set_mcbsp_matrix .....	663
(n_) set_mcbsp_rot .....	667
(n_) set_mcbsp_x .....	668
(n_) set_mcbsp_y .....	669

(1) Only with enabled Option "3D".



## Status Monitoring and Diagnostics

(n_) <code>get_head_status</code>	391
(n_) <code>get_overrun</code>	404
(n_) <code>get_value</code>	423
(n_) <code>get_values</code>	425
(n_) <code>get_waveform</code>	427
(n_) <code>get_waveform_offset</code>	428
(n_) <code>measurement_status</code>	521
(n_) <code>stop_trigger</code>	757

## iDRIVE Commands

(n_) <code>control_command</code>	332
(n_) <code>get_transform</code>	420
(n_) <code>read_user_data</code>	562
(n_) <code>send_user_data</code>	581
<code>transform</code>	789
(n_) <code>upload_transform</code>	793

## Pixel Output Mode – Marking Pixel Images

### (Bitmaps)

(n_) <code>set_default_pixel</code>	594
-------------------------------------	-----

## I/O Commands

(n_) <code>get_free_variable</code>	387
(n_) <code>get_io_status</code>	395
(n_) <code>get_mcbsp</code>	402
(n_) <code>mcbsp_init</code>	519
(n_) <code>mcbsp_init_spi</code>	520
(n_) <code>periodic_toggle</code>	546
(n_) <code>read_analog_in</code>	552
(n_) <code>read_io_port</code>	555
(n_) <code>read_io_port_buffer</code>	556
(n_) <code>read_mcbsp</code>	558
(n_) <code>read_multi_mcbsp</code>	559
(n_) <code>rs232_config</code>	566
(n_) <code>rs232_read_data</code>	567
(n_) <code>rs232_write_data</code>	568
(n_) <code>rs232_write_text</code>	568
(n_) <code>set_free_variable</code>	628
(n_) <code>set_laser_off_default</code>	643
(n_) <code>set_mcbsp_freq</code>	655
(n_) <code>set_mcbsp_out_ptr</code>	666
(n_) <code>set_port_default</code>	685
(n_) <code>uart_config</code>	792
(n_) <code>write_8bit_port</code>	807
(n_) <code>write_da_1</code>	809
(n_) <code>write_da_2</code>	810
(n_) <code>write_da_x</code>	811
(n_) <code>write_io_port</code>	815
(n_) <code>write_io_port_mask</code>	816

## Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization

(n_) <code>get_counts</code>	382
(n_) <code>get_master_slave</code>	402
(n_) <code>get_startstop_info</code>	410
(n_) <code>set_control_mode</code>	592
(n_) <code>set_extstartpos</code>	607
(n_) <code>set_max_counts</code>	655
(n_) <code>simulate_ext_start_ctrl</code>	741
(n_) <code>sync_slaves</code>	765

## List Handling and List Status

(n_) <code>auto_change</code>	321
(n_) <code>auto_change_pos</code>	322
(n_) <code>get_lap_time</code>	396
(n_) <code>get_status</code>	412
(n_) <code>get_wait_status</code>	426
(n_) <code>pause_list</code>	545
(n_) <code>quit_loop</code>	548
(n_) <code>read_status</code>	560
(n_) <code>release_wait</code>	564
(n_) <code>restart_list</code>	566
(n_) <code>set_pause_list_cond</code>	678
(n_) <code>set_pause_list_not_cond</code>	679
(n_) <code>start_loop</code>	744
(n_) <code>stop_execution</code>	756
(n_) <code>stop_list</code>	756

## Input Pointer Commands

(n_) <code>get_input_pointer</code>	395
(n_) <code>get_list_pointer</code>	398
(n_) <code>load_list</code>	482
(n_) <code>set_input_pointer</code>	630
(n_) <code>set_start_list</code>	708
(n_) <code>set_start_list_1</code>	709
(n_) <code>set_start_list_2</code>	709
(n_) <code>set_start_list_pos</code>	710

## Output Pointer Commands

(n_) <code>execute_at_pointer</code>	370
(n_) <code>execute_list</code>	371
(n_) <code>execute_list_1</code>	371
(n_) <code>execute_list_2</code>	371
(n_) <code>execute_list_pos</code>	372
(n_) <code>get_out_pointer</code>	404



<b>Subroutine Commands</b>		<b>"Classic" Processing-on-the-fly Commands</b>	
(n_) <code>copy_dst_src</code> .....	341	(n_) <code>get_encoder</code> .....	383
(n_) <code>get_char_pointer</code> .....	381	(n_) <code>get_fly_2d_offset</code> .....	387
(n_) <code>get_sub_pointer</code> .....	415	(n_) <code>get_marking_info</code> .....	400
(n_) <code>get_text_table_pointer</code> .....	419	(n_) <code>init_fly_2d</code> .....	441
(n_) <code>load_char</code> .....	470	(n_) <code>load_fly_2d_table</code> .....	478
(n_) <code>load_sub</code> .....	491	(n_) <code>read_encoder</code> .....	553
(n_) <code>load_text_table</code> .....	492	(n_) <code>set_ext_start_delay</code> .....	608
(n_) <code>set_char_pointer</code> .....	590	(n_) <code>set_fly_tracking_error</code> .....	622
(n_) <code>set_char_table</code> .....	591	(n_) <code>set_mcbsp_in</code> <sup>(2)</sup> .....	660
(n_) <code>set_sub_pointer</code> .....	711	(n_) <code>set_multi_mcbsp_in</code> <sup>(2)</sup> .....	670
(n_) <code>set_text_table_pointer</code> .....	712	(n_) <code>set_rot_center</code> .....	690
		(n_) <code>simulate_encoder</code> .....	738
		(n_) <code>simulate_ext_stop</code> .....	742
<b>Direct Laser and Scan Head Control</b>		<b>Controlling Stepper Motors</b>	
(n_) <code>disable_laser</code> .....	343	(n_) <code>get stepper_status</code> .....	414
(n_) <code>enable_laser</code> .....	344	(n_) <code>stepper_abs</code> .....	745
(n_) <code>get_laser_pin_in</code> .....	397	(n_) <code>stepper_abs_no</code> .....	746
(n_) <code>get_z_distance</code> <sup>(1)</sup> .....	429	(n_) <code>stepper_control</code> .....	748
(n_) <code>goto_xy</code> .....	430	(n_) <code>stepper_disable_switch</code> .....	749
(n_) <code>goto_xyz</code> <sup>(1)</sup> .....	431	(n_) <code>stepper_enable</code> .....	750
(n_) <code>laser_signal_off</code> .....	450	(n_) <code>stepper_init</code> .....	751
(n_) <code>laser_signal_on</code> .....	451	(n_) <code>stepper_rel</code> .....	753
(n_) <code>set_laser_pin_out</code> .....	644	(n_) <code>stepper_rel_no</code> .....	754
<b>Version Commands</b>		<b>Jump Mode</b>	
(n_) <code>get bios_version</code> .....	379	(n_) <code>get_jump_table</code> .....	396
<code>get_dll_version</code> .....	383	(n_) <code>load_jump_table</code> .....	479
(n_) <code>get_hex_version</code> .....	393	(n_) <code>load_jump_table_offset</code> .....	480
(n_) <code>get_rtc_version</code> .....	406	(n_) <code>set_jump_mode</code> .....	632
(n_) <code>get_serial_number</code> .....	408	(n_) <code>set_jump_table</code> .....	638
<b>Error Commands</b>		<b>Control Commands only for Scan Systems with SCAnahead Control</b>	
(n_) <code>get_error</code> .....	384	(n_) <code>activate_scanahead_autodelays</code> .....	311
(n_) <code>get_last_error</code> .....	398	(n_) <code>get_scanahead_params</code> .....	407
(n_) <code>reset_error</code> .....	565	(n_) <code>set_scanahead_laser_shifts</code> .....	695
(n_) <code>set_verify</code> .....	724	(n_) <code>set_scanahead_line_params</code> .....	695
<code>verify_checksum</code> .....	795	(n_) <code>set_scanahead_params</code> .....	695
		(n_) <code>set_scanahead_speed_control</code> .....	695
<b>Date, Time, Serial Numbers</b>			
(n_) <code>get_list_serial</code> .....	399		
(n_) <code>get_serial</code> .....	408		
(n_) <code>select_serial_set</code> .....	580		
(n_) <code>set_serial</code> .....	697		
(n_) <code>set_serial_step</code> .....	697		
(n_) <code>time_update</code> .....	770		

(1) Only with enabled Option "3D".

(2) Limited functionality if no Option Processing-on-the-fly.



### Control Commands for RTC6 Ethernet Boards

	<code>eth_assign_card</code> .....	345
	<code>eth_assign_card_ip</code> .....	346
(n_)	<code>eth_check_connection</code> .....	348
	<code>eth_convert_ip_to_string</code> .....	349
	<code>eth_convert_string_to_ip</code> .....	350
	<code>eth_count_cards</code> .....	351
	<code>eth_found_cards</code> .....	352
	<code>eth_get_card_info</code> .....	353
	<code>eth_get_card_info_search</code> .....	354
(n_)	<code>eth_get_com_timeouts</code> .....	355
(n_)	<code>eth_get_error</code> .....	356
	<code>eth_get_ip</code> .....	357
	<code>eth_get_ip_search</code> .....	357
(n_)	<code>eth_get_last_error</code> .....	358
(n_)	<code>eth_get_port_numbers</code> .....	360
	<code>eth_get_serial_search</code> .....	360
(n_)	<code>eth_get_static_ip</code> .....	361
	<code>eth_max_card</code> .....	362
	<code>eth_remove_card</code> .....	363
	<code>eth_search_cards</code> .....	364
	<code>eth_search_cards_range</code> .....	365
(n_)	<code>eth_set_com_timeouts</code> .....	366
(n_)	<code>eth_set_port_numbers</code> .....	367
	<code>eth_set_search_cards_timeout</code> .....	368
(n_)	<code>eth_set_static_ip</code> .....	369
(n_)	<code>time_control_eth</code> .....	767

### Other Control Commands

(n_)	<code>auto_cal</code> .....	318
(n_)	<code>bounce_supp</code> .....	323
(n_)	<code>create_dat_file</code> .....	342
(n_)	<code>get_auto_cal</code> .....	379
(n_)	<code>get_card_type</code> .....	380
(n_)	<code>get_galvo_controls</code> .....	388
(n_)	<code>get_hi_data</code> .....	393
(n_)	<code>get_hi_pos</code> .....	394
(n_)	<code>get_time</code> .....	419
(n_)	<code>home_position</code> .....	432
(n_)	<code>home_position_xyz</code> .....	433
(n_)	<code>move_to</code> .....	525
(n_)	<code>set_hi</code> .....	629
(n_)	<code>set_pause_list_cond</code> .....	678
(n_)	<code>set_timelag_compensation</code> .....	713
(n_)	<code>store_timestamp_counter</code> .....	759
(n_)	<code>write_hi_pos</code> .....	813

### Standalone Functionality for

#### RTC6 Ethernet Boards

(n_)	<code>eth_boot_dcmd</code> .....	347
(n_)	<code>set_eth_boot_control</code> .....	606
(n_)	<code>set_eth_boot_timeout</code> .....	606
(n_)	<code>read_image_eth</code> .....	554
(n_)	<code>store_program</code> .....	758
(n_)	<code>write_image_eth</code> .....	814

### 10.1.6 List Commands

#### Board Initialization and Field Correction

(n\_) `select_cor_table_list` var ..... 577

#### Jump Commands

(n_) <code>jump_abs</code> nor .....	444
(n_) <code>jump_rel</code> nor .....	446
(n_) <code>para_jump_abs</code> nor .....	527
(n_) <code>para_jump_rel</code> nor .....	529
(n_) <code>timed_jump_abs</code> nor .....	773
(n_) <code>timed_jump_rel</code> nor .....	775
(n_) <code>timed_para_jump_abs</code> nor .....	781
(n_) <code>timed_para_jump_rel</code> nor .....	783

#### 3D Jump Commands<sup>(1)</sup>

(n_) <code>jump_abs_3d</code> nor .....	445
(n_) <code>jump_rel_3d</code> nor .....	447
(n_) <code>para_jump_abs_3d</code> nor .....	528
(n_) <code>para_jump_rel_3d</code> nor .....	530
(n_) <code>timed_jump_abs_3d</code> nor .....	774
(n_) <code>timed_jump_rel_3d</code> nor .....	776
(n_) <code>timed_para_jump_abs_3d</code> mul .....	782
(n_) <code>timed_para_jump_rel_3d</code> mul .....	784

#### Microvector Commands

(n_) <code>micro_vector_abs</code> nor .....	522
(n_) <code>micro_vector_abs_3d</code> nor .....	523
(n_) <code>micro_vector_rel</code> nor .....	524
(n_) <code>micro_vector_rel_3d</code> nor .....	525

#### Mark Commands

(n_) <code>arc_abs</code> nor .....	314
(n_) <code>arc_rel</code> nor .....	316
(n_) <code>mark_abs</code> nor .....	499
(n_) <code>mark_ellipse_abs</code> nor .....	506
(n_) <code>mark_ellipse_rel</code> nor .....	507
(n_) <code>mark_rel</code> nor .....	508
(n_) <code>para_mark_abs</code> nor .....	533
(n_) <code>para_mark_rel</code> nor .....	535
(n_) <code>set_ellipse</code> us .....	602
(n_) <code>timed_arc_abs</code> nor .....	771
(n_) <code>timed_arc_rel</code> nor .....	772
(n_) <code>timed_mark_abs</code> nor .....	777
(n_) <code>timed_mark_rel</code> nor .....	779
(n_) <code>timed_para_mark_abs</code> nor .....	785
(n_) <code>timed_para_mark_rel</code> nor .....	787

#### 3D Mark Commands<sup>(1)</sup>

(n_) <code>arc_abs_3d</code> nor .....	315
(n_) <code>arc_rel_3d</code> nor .....	317
(n_) <code>mark_abs_3d</code> nor .....	500
(n_) <code>mark_rel_3d</code> nor .....	509
(n_) <code>para_mark_abs_3d</code> nor .....	534
(n_) <code>para_mark_rel_3d</code> nor .....	536
(n_) <code>timed_mark_abs_3d</code> nor .....	778
(n_) <code>timed_mark_rel_3d</code> nor .....	780
(n_) <code>timed_para_mark_abs_3d</code> mul .....	786
(n_) <code>timed_para_mark_rel_3d</code> mul .....	788

#### Text Commands

(n_) <code>mark_char</code> us .....	501
(n_) <code>mark_char_abs</code> us .....	502
(n_) <code>mark_text</code> var .....	513
(n_) <code>mark_text_abs</code> var .....	514
(n_) <code>select_char_set</code> us .....	574

#### Date, Time, Serial Numbers

(n_) <code>mark_date</code> nor .....	503
(n_) <code>mark_date_abs</code> nor .....	505
(n_) <code>mark_serial</code> nor .....	510
(n_) <code>mark_serial_abs</code> nor .....	512
(n_) <code>mark_time</code> nor .....	515
(n_) <code>mark_time_abs</code> nor .....	517
(n_) <code>select_serial_set_list</code> uk .....	580
(n_) <code>set_serial_step_list</code> nor .....	698
(n_) <code>time_fix</code> nor .....	768
(n_) <code>time_fix_f</code> nor .....	768
(n_) <code>time_fix_f_off</code> nor .....	769

(1) Only with enabled Option "3D".

nor normal list command

var variable list command

us undelayed short list command

ds delayed short list command

mul multiple list command



## Status Monitoring and Diagnostics

(n_) <code>set_trigger</code>	<code>ds</code>	714
(n_) <code>set_trigger4</code>	<code>ds</code>	721

## Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization

(n_) <code>set_control_mode_list</code>	<code>nor</code>	594
(n_) <code>set_extstartpos_list</code>	<code>us</code>	607

## List Handling and Structured Programming

(n_) <code>list_continue</code>	<code>nor</code>	458
(n_) <code>list_jump_pos</code>	<code>us</code>	460
(n_) <code>list_jump_rel</code>	<code>us</code>	462
(n_) <code>list_next</code>	<code>us</code>	464
(n_) <code>list_nop</code>	<code>nor</code>	464
(n_) <code>list_repeat</code>	<code>us</code>	465
(n_) <code>list_until</code>	<code>us</code>	467
(n_) <code>long_delay</code>	<code>nor</code>	498
(n_) <code>set_end_of_list</code>	<code>nor</code>	605
(n_) <code>set_list_jump</code>	<code>us</code>	649
(n_) <code>set_wait</code>	<code>nor</code>	725

## Subroutine Commands

(n_) <code>list_call</code>	<code>us</code>	452
(n_) <code>list_call_abs</code>	<code>us</code>	454
(n_) <code>list_call_abs_repeat</code>	<code>us</code>	455
(n_) <code>list_call_repeat</code>	<code>us</code>	457
(n_) <code>list_return</code>	<code>us</code>	466
(n_) <code>sub_call</code>	<code>us</code>	760
(n_) <code>sub_call_abs</code>	<code>us</code>	761
(n_) <code>sub_call_abs_repeat</code>	<code>us</code>	762
(n_) <code>sub_call_repeat</code>	<code>us</code>	763

## Setting the Laser Parameters

(n_) <code>config_laser_signals_list</code>	<code>ds</code>	329
(n_) <code>set_auto_laser_params_list</code>	<code>ds</code>	589
(n_) <code>set_encoder_speed</code>	<code>ds</code>	603
(n_) <code>set_firstpulse_killer_list</code>	<code>us</code>	610
(n_) <code>set_laser_delays</code>	<code>us</code>	642
(n_) <code>set_laser_pin_out_list</code>	<code>us</code>	644
(n_) <code>set_laser_pulses</code>	<code>ds</code>	646
(n_) <code>set_laser_timing</code>	<code>ds</code>	648
(n_) <code>set_pulse_picking_list</code>	<code>us</code>	688
(n_) <code>set_qswitch_delay_list</code>	<code>us</code>	689
(n_) <code>set_softstart_level_list</code>	<code>nor</code>	705
(n_) <code>set_softstart_mode_list</code>	<code>var</code>	706
(n_) <code>set_standby_list</code>	<code>ds</code>	708
(n_) <code>set_vector_control</code>	<code>us</code>	722
(n_) <code>spot_distance</code>	<code>uk</code>	743

## Setting the Scanner Parameters

(n_) <code>set_delay_mode_list</code>	<code>mul</code>	600
(n_) <code>set_jump_speed</code>	<code>us</code>	636
(n_) <code>set_mark_speed</code>	<code>ds</code>	650
(n_) <code>set_scanner_delays</code>	<code>ds</code>	696
(n_) <code>set_sky_writing_limit_list</code>	<code>us</code>	699
(n_) <code>set_sky_writing_list</code>	<code>nor</code>	699
(n_) <code>set_sky_writing_mode_list</code>	<code>nor</code>	701
(n_) <code>set_sky_writing_para_list</code>	<code>nor</code>	704

## Coordinate Transformations

(n_) <code>set_angle_list</code>	<code>var</code>	584
(n_) <code>set_defocus_list</code>	<code>var (1)</code>	597
(n_) <code>set_defocus_offset_list</code>	<code>var (1)</code>	598
(n_) <code>set_matrix_list</code>	<code>var</code>	654
(n_) <code>set_offset_list</code>	<code>var</code>	675
(n_) <code>set_offset_xyz_list</code>	<code>var (1)</code>	677
(n_) <code>set_scale_list</code>	<code>var</code>	694

## Online Positioning

(n_) <code>apply_mcbsp_list</code>	<code>nor</code>	313
(n_) <code>set_mcbsp_global_matrix_list</code>	<code>us</code>	656
(n_) <code>set_mcbsp_global_rot_list</code>	<code>us</code>	657
(n_) <code>set_mcbsp_global_x_list</code>	<code>us</code>	658
(n_) <code>set_mcbsp_global_y_list</code>	<code>us</code>	659
(n_) <code>set_mcbsp_matrix_list</code>	<code>us</code>	664
(n_) <code>set_mcbsp_rot_list</code>	<code>us</code>	667
(n_) <code>set_mcbsp_x_list</code>	<code>us</code>	668
(n_) <code>set_mcbsp_y_list</code>	<code>us</code>	669

## Direct Laser and Scan Head Control

(n_) <code>laser_on_list</code>	<code>var</code>	448
(n_) <code>laser_on_pulses_list</code>	<code>var</code>	449
(n_) <code>laser_signal_off_list</code>	<code>nor</code>	450
(n_) <code>laser_signal_on_list</code>	<code>nor</code>	451
(n_) <code>para_laser_on_pulses_list</code>	<code>var</code>	531

## Pixel Output Mode – Marking Pixel Images (Bitmaps)

(n_) <code>set_default_pixel_list</code>	<code>us</code>	595
(n_) <code>set_n_pixel</code>	<code>var</code>	674
(n_) <code>set_pixel</code>	<code>var</code>	680
(n_) <code>set_pixel_line</code>	<code>nor</code>	681
(n_) <code>set_pixel_line_3d</code>	<code>mul (1)</code>	684

(1) Only with enabled Option "3D".

`nor` normal list command

`var` variable list command

`us` undelayed short list command

`ds` delayed short list command

`mul` multiple list command

### I/O Commands

(n_)	clear_io_cond_list us	327
(n_)	periodic_toggle_list us	547
(n_)	read_io_port_list us	557
(n_)	rs232_write_text_list var	569
(n_)	set_free_variable_list us	628
(n_)	set_io_cond_list us	631
(n_)	set_laser_power us	645
(n_)	set_mcbsp_out us	665
(n_)	set_port_default_list us	686
(n_)	write_8bit_port_list ds	807
(n_)	write_da_1_list ds	809
(n_)	write_da_2_list ds	810
(n_)	write_da_x_list ds	812
(n_)	write_io_port_list ds	815
(n_)	write_io_port_mask_list ds	816

### Conditional Commands

(n_)	if_cond us	434
(n_)	if_not_cond us	437
(n_)	if_not_pin_cond us	439
(n_)	if_pin_cond us	440
(n_)	list_call_abs_cond us	455
(n_)	list_call_cond us	456
(n_)	list_jump_cond us	459
(n_)	list_jump_pos_cond us	461
(n_)	list_jump_rel_cond us	463
(n_)	sub_call_abs_cond us	761
(n_)	sub_call_cond us	762
(n_)	switch_ioport us	764

### "Classic" Processing-on-the-fly Commands

(n_)	activate_fly_2d var (1)	307
(n_)	activate_fly_2d_encoder mul (1)	308
(n_)	activate_fly_xy var (1)	310
(n_)	activate_fly_xy_encoder mul (1)	310
(n_)	clear_fly_overflow us	326
(n_)	fly_return nor	373
(n_)	fly_return_z nor	377
(n_)	if_fly_x_overflow us	434
(n_)	if_fly_y_overflow us	435
(n_)	if_fly_z_overflow us	435
(n_)	if_not_activated us	436
(n_)	if_not_fly_x_overflow us	438
(n_)	if_not_fly_y_overflow us	438
(n_)	if_not_fly_z_overflow us	439
(n_)	park_position var (1)	537
(n_)	park_return var (1)	541
(n_)	set_ext_start_delay_list nor	609
(n_)	set_fly_2d nor (1)	614
(n_)	set_fly_limits us	618
(n_)	set_fly_limits_z us	619
(n_)	set_fly_rot nor (1)	620
(n_)	set_fly_rot_pos nor (1)	621
(n_)	set_fly_x nor (1)	623
(n_)	set_fly_x_pos nor (1)	624
(n_)	set_fly_y nor (1)	625
(n_)	set_fly_y_pos nor (1)	626
(n_)	set_fly_z nor (1)	627
(n_)	set_mcbsp_in_list nor (2)	662
(n_)	set_multi_mcbsp_in_list nor (2)	673
(n_)	set_rot_center_list ds	690
(n_)	simulate_ext_start nor	739
(n_)	store_encoder us	757
(n_)	wait_for_encoder nor	800
(n_)	wait_for_encoder_in_range mul	801
(n_)	wait_for_encoder_in_range_mode mul	802
(n_)	wait_for_encoder_mode nor	803
(n_)	wait_for_mcbsp nor	805

(1) Only with enabled Option Processing-on-the-fly.

(2) Limited functionality if no Option Processing-on-the-fly.

nor normal list command

var variable list command

us undelayed short list command

ds delayed short list command

mul multiple list command



### "Fly Extension" Commands

(n_) activate_fly_1_axis	var	304
(n_) activate_fly_2_axes	var	305
(n_) fly_return_1_axis	var	374
(n_) fly_return_2_axes	var	375
(n_) fly_return_3_axes	var	376
(n_) park_position_1_axis	var	539
(n_) park_position_2_axes	var	540
(n_) park_return_1_axis	var	543
(n_) park_return_2_axes	var	544
(n_) set_fly_1_axis	nor	611
(n_) set_fly_2_axes	nor	612
(n_) set_fly_3_axes	mul	616
(n_) wait_for_1_axis	nor	796
(n_) wait_for_2_axes	mul	798

### Controlling Stepper Motors

(n_) stepper_abs_list	us	746
(n_) stepper_abs_no_list	us	747
(n_) stepper_control_list	us	749
(n_) stepper_enable_list	us	750
(n_) stepper_rel_list	us	753
(n_) stepper_rel_no_list	us	754
(n_) stepper_wait	nor	755

### Jump Mode

(n_) set_jump_mode_list	nor	635
-------------------------	-----	-----

### Camming

(n_) camming	nor	324
--------------	-----	-----

### Wobbel Mode

(n_) set_wobbel	ds	726
(n_) set_wobbel_control	us	728
(n_) set_wobbel_direction	us	730
(n_) set_wobbel_mode	ds	731
(n_) set_wobbel_mode_phase	ds	733
(n_) set_wobbel_offset	ds	734
(n_) set_wobbel_vector	us	735

### List Commands only for Scan Systems with SCANAhead Control

(n_) activate_scanahead_autodelays_list	us	311
(n_) set_scanahead_laser_shifts_list	us	695
(n_) set_scanahead_line_params_list	us	695

### Other List Commands

(n_) range_checking	us	549
(n_) save_and_restart_timer	ds	571
(n_) store_timestamp_counter_list	us	759
(n_) wait_for_timestamp_counter	nor	806



### 10.1.7 Data Types

The following table defines the formats and ranges of the different data types used by the RTC6 commands:

Data Format	Range	Pascal	C, C++	C#
Unsigned 32-bit value	[0; $(2^{32}-1)$ ]	longword	unsigned long	uint
Signed 32-bit value	[ $-2^{31}$ ; $+(2^{31}-1)$ ]	longint	long	int
64-bit IEEE floating point value		double	double	double
Pointer to a NULL-terminated ANSI string (1 byte per char)	4 Byte for Win32-user programs 8 Byte for Win64-user programs	pchar	char*	string

#### Pointer to Locations in the PC Memory

Some commands (for example, [get\\_transform](#), [get\\_values](#), [get\\_waveform](#), [transform](#) or [upload\\_transform](#)) have pointers to locations in the PC's memory as parameters. In C# and Pascal, appropriate pointer data types are heretofore used (see import declarations). In C and C++, the data type ULONG\_PTR is used for this pointer parameters. The ULONG\_PTR data type is defined in the C and C++ import declarations as follows (ULONG\_PTR = unsigned 32-bit value for Win32-user programs, ULONG\_PTR = unsigned 64-bit value for Win64user programs):

```
#if !defined(ULONG_PTR)
    #if !_WIN64
        #define ULONG_PTR UINT
    #else
        #define ULONG_PTR UINT64
    #endif // !_WIN64
#endif // !defined(ULONG_PTR)
```

Usually, the data type ULONG\_PTR is also appropriately defined in the Windows header file BaseTsd.h.



## 10.2 RTC6 Command Set

The commands are in alphabetical order.

The general structure of the command tables is as follows<sup>(1)</sup>:

- (1) A program language-neutral form is used. Each real programming language has its own individual nomenclature.

Category of the command	example_command_name_one
Function	Short description describing the purpose of the command.
Call	Shows the correct spellings and the sequence of the parameters. Note, there is no semi-colon at the end of the line. A '&' (address operator) is only used in this table row and indicates a pointer. Examples:  example_command_name_one( parameter_A, &parameter_B, parameter_C ) example_variable = example_command_name_one( parameter_A )
Parameters(*)	A      Short text. Data type. (*) in some C/C++ code descriptions labeled with __out.
	C      Short text. Data type.
Returned parameter values(**)	B      Short text. Data type. (**) in some C/C++ code descriptions labeled with __out.
Result	If implemented: mentions the returned result value and data type in generic form (Example: Error code #. As an unsigned 32-bit value.). A value range is here only given, if the actual usable one is smaller than the value range of the data type. If not implemented: "None."
Comments	<ul style="list-style-type: none"> <li>Additional information on this command.</li> <li>References to other chapters and publications.</li> </ul>
RTC4→RTC6	States the differences to the command (of the same name) of the RTC4 command set.
RTC5→RTC6	States the differences to the command (of the same name) of the RTC5 command set.
Version info	For example, states the minimum versions of DLL, RBF, OUT which are required to use the command, latest change in version.
References	Links to related commands: <b>command_name_two</b> , <b>command_name_three</b>



<b>Ctrl Command</b>	<b>acquire_rtc</b>
<b>Function</b>	Acquires the specified RTC6 board for a user program.
<b>Call</b>	NoOfAcquiredCard = acquire_rtc( CardNo )
<b>Parameters</b>	CardNo <b>RTC6 DLL</b> -internal number (RTC6 board management index) of the desired board. As an unsigned 32-bit value.
<b>Result</b>	The return value is: <ul style="list-style-type: none"> <li>• CardNo, if the reservation was successful</li> <li>• 0, if the board is currently reserved for another user program or the version check detects an error</li> </ul> As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>acquire_rtc</b> is also useful for single-board systems which need to coordinate the use of a RTC6 board by different user programs.</li> <li>• <b>acquire_rtc</b> has no effect (return value 0, <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>), if: <ul style="list-style-type: none"> <li>– CardNo &gt; number of RTC6 PCIe Boardns found during initialization (see <b>rtc6_count_cards</b>) and no RTC6 Ethernet Board is entered there</li> <li>– CardNo = 0 (real boards begin with 1)</li> </ul> </li> <li>• Access rights to existing boards are granted exclusively (always to only one user program at a time). Therefore, <b>acquire_rtc</b> has no effect if the specified board is already acquired by another user program (return value 0, <b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>). This must explicitly release the RTC6 board (by <b>release_rtc</b> or <b>free_rtc6_dll</b>) before the RTC6 board can be acquired by another user program (with <b>acquire_rtc</b>). On the other hand, if the board was already freed prior to initialization of a user program, then initialization by <b>init_rtc6_dll</b> result in RTC6 board management assigning board access rights for the user program. In this case, an explicit <b>acquire_rtc</b> call is not needed and has no effect. Nevertheless, the return value is CardNo.</li> <li>• Assorted versions of the <b>RTC6 DLL</b> and the files <b>RTC6OUT.out</b>, <b>RTC6RBF.rbf</b> and <b>RTC6DAT.dat</b> cannot be arbitrarily combined with another. <b>acquire_rtc</b> performs a version compatibility check. If <b>RTC6OUT.out</b>, <b>RTC6RBF.rbf</b> and <b>RTC6DAT.dat</b> are not yet loaded, then this check cannot be explicitly executed (<b>get_last_error</b> return code <b>RTC6_TIMEOUT</b>), but the check still regarded as successful and acquisition is not hindered. If <b>RTC6OUT.out</b>, <b>RTC6RBF.rbf</b> and <b>RTC6DAT.dat</b> are loaded and the version check determines an error, then access is denied (return value 0, <b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>   <b>RTC6_VERSION_MISMATCH</b>). With RTC6 Ethernet Boards, <b>RTC6OUT.out</b> must be replaced by <b>RTC6ETH.out</b>. <b>init_rtc6_dll</b> does not automatically acquire RTC6 Ethernet Boards.</li> <li>• A board successfully acquired by <b>acquire_rtc</b> does not automatically become the “active” board. Activation of a board is only achieved by <b>select_rtc</b> or <b>init_rtc6_dll</b>.</li> <li>• Running boards are neither halted nor initialized by <b>acquire_rtc</b>.</li> </ul>



<b>Ctrl Command</b>	<b>acquire_RTC</b>
Comments (cont'd)	<ul style="list-style-type: none"><li>• <b>acquire_RTC</b> is available even without explicit access rights to a particular RTC6 board.</li><li>• <b>acquire_RTC</b> is not available as a multi-board command.</li><li>• See also <a href="#">Chapter 6.7.1 "Notes on Board Acquisition by a User Program", page 118</a>.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">init_RTC6_dll</a> , <a href="#">select_RTC</a> , <a href="#">free_RTC6_dll</a> , <a href="#">release_RTC</a> , <a href="#">RTC6_count_cards</a>



<b>Variable List Command</b>	<b>activate_fly_1_axis</b>
<b>Function</b>	"Fly Extension" Command: Activates a 1 Axisn-Processing-on-the-fly application.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>activate_fly_1_axis</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>activate_fly_1_axis( Axis, Mode, Scale, Offset )</code>
<b>Parameters</b>	<p>Axis      <b>Axis from Table 3, page 245.</b>              As an unsigned 32-bit value.              Allowed values: 1...2.</p> <p>Mode     <b>Mode from Table 4, page 247.</b>              As an unsigned 32-bit value.              Allowed values: 1...4.</p> <p>Scale    Scaling factor.              As a 64-bit IEEE floating point value.              Allowed value range:              • <math>1/256 \leq  \text{Scale}  \leq 16.000,0</math> with linear axis (1, 2 or 3)              Scale can be + or -. Only the absolute value is restricted.</p> <p>Offset   Offset to the encoder value.              As a signed 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Section ""Fly Extension" Commands", page 245</a>.</li> <li><b>activate_fly_1_axis</b> occupies two list memory positions.</li> <li><b>activate_fly_1_axis</b> requires two <math>10 \mu\text{s}</math> clock cycles for execution.</li> <li>At the specified <b>Axis1</b>, neither a Processing-on-the-fly correction nor a rotation correction must be active. However, the <b>Axis</b> can be combined as a linear axis with the third linear axis.</li> <li>An xy positioning stage compensation is automatically executed, if after the successful <b>activate_fly_1_axis</b> execution <b>Axis 1</b> and <b>Axis 2</b> are activated with two physically different encoder modes (for example, <b>Mode 1</b> and <b>3</b> mean the same physical encoder) and xy positioning stage compensation is defined and activated (see <a href="#">load_fly_2d_table</a>), then xy travel table compensation is automatically executed.</li> <li>With an unallowed parameter value, <b>activate_fly_1_axis</b> is replaced by a <b>list_nop</b> (<a href="#">get_last_error</a> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>See also comments on <a href="#">activate_fly_2_axes</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">activate_fly_2_axes</a>



<b>Variable List Command</b>	<b>activate_fly_2_axes</b>
<b>Function</b>	"Fly Extension" Command: Activates a 2-Axes-Processing-on-the-fly application.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>activate_fly_2_axes</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>activate_fly_2_axes( ModeX, ScaleX, OffsetX, ModeY, ScaleY, OffsetY)</code>
<b>Parameters</b>	<p>ModeX      Mode from <a href="#">Table 4, page 247</a>. As an unsigned 32-bit value.</p> <p>ScaleX      Scaling factor. As a 64-bit IEEE floating point value. Allowed value range: • <math>1/256 \leq  \text{Scale}  \leq 16.000,0</math> with linear axis (<a href="#">1</a>, <a href="#">2</a> or <a href="#">3</a>) Scale can be + or -. Only the absolute value is restricted.</p> <p>OffsetX      Offset to the encoder value. As a signed 32-bit value.</p> <p>ModeY      Like ModeX.</p> <p>ScaleY      Like ScaleX.</p> <p>OffsetY      Like OffsetX.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Section "Fly Extension" Commands", page 245</a>.</li> <li>The Axes are automatically <a href="#">1</a> and <a href="#">2</a>. The modes need to be two physically different encoder modes (<a href="#">Mode 1</a> and <a href="#">3</a> or <a href="#">2</a> and <a href="#">4</a> are not allowed).</li> <li><b>activate_fly_2_axes</b> occupies two list memory positions.</li> <li><b>activate_fly_2_axes</b> requires two <math>10\ \mu\text{s}</math> clock cycles for execution.</li> <li>At Axis <a href="#">1</a> and <a href="#">2</a>, neither a Processing-on-the-fly correction nor a rotation correction must be active. However, the Axes can be combined as linear axes with the third linear axis.</li> <li>With an unallowed parameter value, <b>activate_fly_2_axes</b> is replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</li> <li>If an xy positioning stage compensation is defined and active (see <a href="#">load_fly_2d_table</a>), then it is carried out automatically. See also comments on <a href="#">activate_fly_2d</a> (encoder reset, error bit, <a href="#">get_marking_info</a>).</li> </ul>



<b>Variable List Command</b>	<b>activate_fly_2_axes</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The following command calls are executed in the same way:           <ul style="list-style-type: none"> <li>- activate_fly_2_axes( 1, ScaleX, 0, 2, ScaleY, 0 ) =  <code>activate_fly_2d( ScaleX, ScaleY )</code>            or  <code>activate_fly_xy( ScaleX, ScaleY )</code></li> <li>- activate_fly_2_axes( 1, ScaleX, EncX, 2, ScaleY, EncY ) =  <code>activate_fly_xy_encoder( ScaleX, ScaleY, EncX, EncY )</code>            "xy" and "2d" are distinguished with <code>wait_for_1_axis</code> and <code>wait_for_2_axes</code>!</li> <li>- activate_fly_2_axes( 1, ScaleX, OffsetX, 2, ScaleY, OffsetY ) =  <code>{</code>  <code>activate_fly_1_axis( 1, 1, ScaleX, OffsetX );</code>  <code>activate_fly_1_axis( 2, 2, ScaleY, OffsetY );</code>  <code>}</code></li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">activate_fly_1_axis</a>



<b>Variable List Command</b>	<b>activate_fly_2d</b>
<b>Function</b>	Activates a <b>set_fly_2d</b> Processing-on-the-fly application without encoder resets.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>activate_fly_2d</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>activate_fly_2d( ScaleX, ScaleY )</code>
<b>Parameters</b>	<p>ScaleX      Scaling factor as for <b>set_fly_2d</b>.</p> <p>ScaleY      Scaling factor as for <b>set_fly_2d</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If no Processing-on-the-fly correction is active, then <b>activate_fly_2d</b> activates <b>set_fly_2d</b> Processing-on-the-fly correction, see <a href="#">Chapter 8.6.4 "Compensating 2D Motions", page 234</a>. Unlike <b>set_fly_2d</b>, <b>activate_fly_2d</b> does not thereby reset the encoders, but instead calculates coordinate values such that the Processing-on-the-fly-corrected output matches the current output. If the then-current recalculated coordinate values would have fallen outside the 29-bit virtual image field, then Processing-on-the-fly correction is not activated. Then an error bit is set that can be queried by <b>get_marking_info</b> (<b>Bit #9</b>).</li> <li>If Processing-on-the-fly correction is active, then <b>activate_fly_2d</b> is a short list command without further effect and merely sets an error bit queryable by <b>get_marking_info</b> (<b>Bit #9</b>). Therefore, <b>activate_fly_2d</b> cannot be used to modify the Processing-on-the-fly mode itself or to modify the scaling factors of the same mode.</li> <li>You can also query the error bit by the short list command <b>if_notActivated</b> in order to jump to an appropriate error handling routine.</li> <li>Successful activation by <b>activate_fly_2d</b> does not reset an error bit. It remains set for <b>get_marking_info</b> until <b>get_marking_info</b> is called.</li> <li>If unallowed parameter values are supplied (for example, for ScaleX = 0), then <b>activate_fly_2d</b> is (already during loading) replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li><b>activate_fly_2d</b> does not affect the laser signals ("laser active" laser control signals remain on/off if they are on/off).</li> </ul>
<b>RTC4→RTC6</b>	New command.  <b>RTC4 Compatibility Mode:</b> see <b>set_fly_2d</b> .
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>set_fly_2d</b> , <b>activate_fly_2d_encoder</b>

<b>Multiple List Command</b>	<b>activate_fly_2d_encoder</b>
<b>Function</b>	Activates a <b>set_fly_2d</b> Processing-on-the-fly application with encoder reset and encoder offsets.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>activate_fly_2d_encoder</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>activate_fly_2d_encoder( ScaleX, ScaleY, EncX, EncY )</code>
<b>Parameters</b>	<p>ScaleX      Scaling factor as for <b>set_fly_2d</b>.</p> <p>ScaleY      Scaling factor as for <b>set_fly_2d</b>.</p> <p>EncX        Encoder offset. As a signed 32-bit value.</p> <p>EncY        Encoder offset. As a signed 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>activate_fly_2d_encoder</b> occupies two list memory positions and also needs two 10 µs clock periods to execute (the first part of the command is <i>not</i> a short list command).</li> <li>• <b>activate_fly_2d_encoder</b> is a combination of <b>set_fly_2d</b> (encoder reset) and <b>activate_fly_2d</b> (see also comments there). However, in <b>activate_fly_2d_encoder</b> the current (reset) encoder values are not used to calculate the Processing-on-the-fly uncorrected virtual image field coordinates, but the parameter values <code>EncX</code> and <code>EncY</code>. For the error handling, see comments of <b>activate_fly_2d</b>.</li> <li>• Because of this combination, <b>activate_fly_2d_encoder</b> saves the positioning stage movement (which is often long but may be necessary for the Processing-on-the-fly activation without this command) from the initialization position (for example, at the lower left corner) to the center and back again.</li> <li>• Subsequently all encoder values are offset with <code>EncX</code> and <code>EncY</code>, before the Processing-on-the-fly correction is applied. All other encoder related commands refer to the actual encoder values and behave as before.</li> <li>• If the value of <code>EncX</code> or <code>EncY</code> is not allowed (that is, the Processing-on-the-fly-corrected virtual image field coordinates are outside the virtual image field limits), then <b>activate_fly_2d_encoder</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>• If the value of <code>ScaleX</code> or <code>ScaleY</code> is not allowed (see <b>set_fly_2d</b>), the first part is transferred to the board (and thus executes the encoder reset). However, the second part is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>• With active Processing-on-the-fly correction, <b>activate_fly_2d_encoder</b> is a short list command without further effect and merely sets an error bit queryable by <b>get_marking_info</b> (Bit #9). Therefore, <b>activate_fly_2d_encoder</b> cannot be used to modify the Processing-on-the-fly mode itself nor the scaling factors or encoder offsets of the same mode.</li> </ul>



<b>Multiple List Command</b>	<b>activate_fly_2d_encoder</b>
RTC4→RTC6	New command. <b>RTC4 Compatibility Mode:</b> see <a href="#">set_fly_2d</a> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 610, OUT 610, RBF 615.
References	<a href="#">set_fly_2d</a> , <a href="#">activate_fly_2d</a>



<b>Variable List Command</b>	<b>activate_fly_xy</b>
<b>Function</b>	Activates a <b>set_fly_x/set_fly_y</b> Processing-on-the-fly application without encoder resets.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>activate_fly_xy</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>activate_fly_xy( ScaleX, ScaleY )</code>
<b>Parameters</b>	ScaleX      Scaling factor as for <b>set_fly_x/set_fly_y</b> . ScaleY      Scaling factor as for <b>set_fly_x/set_fly_y</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>Like <b>activate_fly_2d</b> with the difference that a <b>set_fly_x/set_fly_y</b> Processing-on-the-fly session is activated.</li> </ul>
<b>RTC4→RTC6</b>	New command. <b>RTC4 Compatibility Mode:</b> see <b>set_fly_x/set_fly_y</b> .
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>set_fly_x, set_fly_y, activate_fly_xy_encoder</b>

<b>Multiple List Command</b>	<b>activate_fly_xy_encoder</b>
<b>Function</b>	Activates a <b>set_fly_x/set_fly_y</b> Processing-on-the-fly application with encoder reset and encoder offsets.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>activate_fly_xy_encoder</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>activate_fly_xy_encoder( ScaleX, ScaleY, EncX, EncY )</code>
<b>Parameters</b>	ScaleX      Scaling factor as for <b>set_fly_x/set_fly_y</b> . ScaleY      Scaling factor as for <b>set_fly_x/set_fly_y</b> . EncX        Encoder offset. As a signed 32-bit value. EncY        Encoder offset. As a signed 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>Like <b>activate_fly_2d_encoder</b> with the difference that a <b>set_fly_x/set_fly_y</b> Processing-on-the-fly session is activated.</li> </ul>
<b>RTC4→RTC6</b>	New command. <b>RTC4 Compatibility Mode:</b> see <b>set_fly_2d</b> .
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 610, OUT 610, RBF 615.
<b>References</b>	<b>set_fly_x, set_fly_y, activate_fly_xy, activate_fly_2d_encoder</b>



<b>Ctrl Command</b>	<b>activate_scanahead_autodelays</b>
Comments	<i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i> This command is described in the manual "excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards".

<b>Undelayed Short List Command</b>	<b>activate_scanahead_autodelays_list</b>
Comments	<i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i> This command is described in the manual "excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards".



<b>Ctrl Command</b>	<b>apply_mcbsp</b>
<b>Function</b>	Fetches the most recent values fully transmitted over the <b> McBSP interface</b> for <b>"Local Online Positioning"</b> and defines offset and/or rotation matrix $M_R$ or general transformation matrix $M_T$ for subsequent coordinate transformations.
<b>Call</b>	<code>apply_mcbsp( HeadNo, at_once )</code>
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value.                        = 1: The definition only affects the <i>first</i> scan head connector.                        = 2: The definition only affects the <i>second</i> scan head connector.                        = 0, 3: The definition affects <i>both</i> scan head connectors.                        Only the two least significant bits are evaluated.</p> <p>at_once     Determines when the defined transformation becomes effective.                        An unsigned 32-bit value.                        = 0: The new total transformation (total matrix and offset) is only calculated and applied to the current position when the next list command is executed.                        = 1: The new total transformation is calculated immediately (or before the next list command if a list is currently <b>BUSY</b> or the board is <b>INTERNAL-BUSY</b>) and applied to the current position.                        = 2: The new total transformation (total matrix and offset) is only calculated and applied to the current position when the next <b>jump_abs</b>, <b>jump_rel</b>, <b>goto_xy</b> or <b>goto_xyz</b> is executed.                        &gt; 2: Like <code>at_once = 2</code>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Data acquisition by the <b> McBSP interface</b> for <b>"Local Online Positioning"</b> must be activated in advance by the commands <b>set_mcbsp_x</b>, <b>set_mcbsp_y</b> and/or <b>set_mcbsp_rot</b> or <b>set_mcbsp_matrix</b> (or with the corresponding list commands). Depending on the configuration, the command <b>apply_mcbsp</b> only defines (as with <b>set_offset</b>) an X and/or Y offset or also (as with <b>set_angle</b>) a rotation matrix or (as with <b>set_matrix</b>) a general matrix operation (see <b>Chapter 8.3.1 "Local Online Positioning"</b>, page 214). As with the commands described in <b>Chapter 8.2 "Coordinate Transformations"</b>, page 210, the parameter <code>at_once</code> determines when the newly defined total transformation becomes effective.</li> <li>Transformations previously defined by <b>set_angle</b>, <b>set_offset</b> or <b>set_matrix</b> get overwritten by <b>apply_mcbsp</b>. In contrast, transformations and focus shifts previously defined by <b>set_scale</b> or <b>set_defocus</b> and z offsets defined by <b>set_offset_xyz</b> is continued to be taken into account, when the total transformation gets recalculated.</li> <li>Any new definitions made with <b>set_angle</b>, <b>set_offset</b> or <b>set_matrix</b> overwrite coordinate transformations defined by the <b> McBSP interface</b>.</li> <li>The <b> McBSP interface</b> always ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>, so available data is not transmitted, see <b>page 75</b>.</li> </ul>



<b>Ctrl Command</b>	<b>apply_mcbsp</b>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">apply_mcbsp_list</a> , <a href="#">set_mcbsp_x</a> , <a href="#">set_mcbsp_y</a> , <a href="#">set_mcbsp_rot</a> , <a href="#">set_mcbsp_matrix</a>

<b>Normal List Command</b>	<b>apply_mcbsp_list</b>
Function	Like <a href="#">apply_mcbsp</a> , but a list command.
Call	<b>apply_mcbsp_list( HeadNo, at_once )</b>
Parameters	<p>HeadNo See <a href="#">apply_mcbsp</a>.</p> <p>at_once Determines when the defined transformation becomes effective. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>= 0: The transformation settings are only accumulated and intermediately stored, but the transformation is not processed as long as this is not activated by another coordinate transformation (for example, by a list command with at_once = 1 or a corresponding control command).</li> <li>= 1: The transformation is immediately calculated (including all transformation settings that were accumulated until then) and processed prior to the next list command.</li> <li>= 2: The transformation settings are only accumulated and intermediately stored (as with at_once = 0). However, The transformation is immediately calculated (including all transformation settings that were accumulated and intermediately stored until then) and applied to the current position when the next <a href="#">jump_abs</a>, <a href="#">jump_rel</a>, <a href="#">goto_xy</a> or <a href="#">goto_xyz</a> is executed.</li> <li>&gt; 2: Like at_once = 2.</li> </ul>
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">apply_mcbsp</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">apply_mcbsp</a>



<b>Normal List Command</b>	<b>arc_abs</b>						
<b>Function</b>	Moves the laser focus from the current position at mark speed along an arc with the specified angle and center point (absolute coordinate values) within a 2D image field.						
<b>Call</b>	<code>arc_abs( X, Y, Angle )</code>						
<b>Parameters</b>	<table> <tr> <td>X</td><td>Absolute coordinates of the arc center. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td>Y</td><td></td></tr> <tr> <td>Angle</td><td>Arc angle. In degrees. As a 64-bit IEEE floating point value. A positive sign means "clockwise". Allowed value range: [-3,600.0°...+3,600.0°] (<math>\pm 10</math> full circles). Out-of-range values are clipped to the boundary values.</td></tr> </table>	X	Absolute coordinates of the arc center. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.	Y		Angle	Arc angle. In degrees. As a 64-bit IEEE floating point value. A positive sign means "clockwise". Allowed value range: [-3,600.0°...+3,600.0°] ( $\pm 10$ full circles). Out-of-range values are clipped to the boundary values.
X	Absolute coordinates of the arc center. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.						
Y							
Angle	Arc angle. In degrees. As a 64-bit IEEE floating point value. A positive sign means "clockwise". Allowed value range: [-3,600.0°...+3,600.0°] ( $\pm 10$ full circles). Out-of-range values are clipped to the boundary values.						
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the mark speed has not been previously explicitly set by <a href="#">set_mark_speed</a> or <a href="#">set_mark_speed_ctrl</a>, then the marking is executed at a predefined mark speed of 1,000 bits/ms.</li> <li>The "laser active" laser control signals are automatically turned on at the beginning of the command (or remain on after a directly preceding mark or arc command). The defined scanner and laser delays are thereby taken into account, see <a href="#">Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control", page 134</a>. Note that other delays are executed in Sky Writing mode, see <a href="#">Chapter 7.2.4 "Sky Writing", page 149</a>.</li> </ul> <p>Exception: zero-length arc commands, see <a href="#">Section "Notes", page 138</a>.</p>						
RTC4→RTC6	Unchanged functionality. In addition: increased value range.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified arc center coordinate values by 16. The allowed value ranges decrease accordingly.						
RTC5→RTC6	Unchanged functionality. In addition: increased value range.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">set_mark_speed</a> , <a href="#">set_scanner_delays</a> , <a href="#">arc_rel</a> , <a href="#">timed_arc_abs</a> , <a href="#">mark_abs</a> , <a href="#">arc_abs_3d</a> , <a href="#">mark_ellipse_abs</a>						



<b>Normal List Command</b>	<b>arc_abs_3d</b>
<b>Function</b>	Moves the laser focus at mark speed from the current position helical around an axis parallel to the z axis. The x and y components thereby characterize an arc with the specified angle around the specified axis, while the z component characterizes a linear motion from the current position to the specified end point. The position of the helical axis and the z end coordinate are specifiable as absolute coordinate values.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>arc_abs_3d</b> has the same effect as <b>arc_abs</b> .
<b>Call</b>	<b>arc_abs_3d( X, Y, Z, Angle )</b>
<b>Parameters</b>	<p>X      Position of the helical axis (parallel to the z axis) as absolute x coordinate. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.</p> <p>Y      Like X (analogously).</p> <p>Z      Absolute z end coordinate. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</p> <p>Angle    Arc angle. In degrees. As a 64-bit IEEE floating point value. A positive sign means "clockwise". Allowed value range: [-3,600.0°...+3,600.0°] (<math>\pm 10</math> full circles). Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>arc_abs_3d</b> functions similarly to <b>arc_abs</b> (see comments there).</li> <li>The z motion is not taken into account during calculation of the number of microsteps.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the values specified for X, Y and Z by 16. The allowed value range decreases accordingly.</p>
RTC5→RTC6	<p>Unchanged functionality. In addition: increased value range.</p> <p>In <b>RTC5 Compatibility Mode</b>, the RTC6 multiplies the values specified for Z by 16. The allowed value range decreases accordingly.</p>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>arc_abs, arc_rel_3d</b>



<b>Normal List Command</b>	<b>arc_rel</b>
<b>Function</b>	Moves the laser focus from the current position at mark speed along an arc with the specified angle and center point (relative coordinate values) within a 2D image field.
<b>Call</b>	<code>arc_rel( dx, dy, Angle )</code>
<b>Parameters</b>	<p><code>dx</code>      Relative coordinates of the arc center. In bits.  <code>dy</code>      As signed 32-bit values.                      Allowed value range: [-268,435,456...+268,435,455].                      Out-of-range values are clipped to the boundary values.</p> <p><code>Angle</code>    Arc angle. In degrees.                      As a 64-bit IEEE floating point value.                      A positive sign means "clockwise".                      Allowed value range: [-3,600.0°...+3,600.0°] (<math>\pm 10</math> full circles).                      Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the arc center are to be supplied as relative coordinates with respect to the current position. Otherwise, <code>arc_rel</code> is identical to <code>arc_abs</code> (see comments there).</li> </ul>
<b>RTC4→RTC6</b>	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for <code>dx</code> and <code>dy</code> by 16. The allowed value range decreases accordingly.
<b>RTC5→RTC6</b>	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_mark_speed</a> , <a href="#">set_scanner_delays</a> , <a href="#">arc_abs</a> , <a href="#">timed_arc_rel</a> , <a href="#">mark_rel</a> , <a href="#">arc_rel_3d</a> , <a href="#">mark_ellipse_rel</a>



<b>Normal List Command</b>	<b>arc_rel_3d</b>								
<b>Function</b>	Moves the laser focus at mark speed from the current position helical around an axis parallel to the z axis. The x and y components thereby characterize an arc with the specified angle around the specified axis, while the z component characterizes a linear motion from the current position to the specified end point. The position of the helical axis and the z end coordinate are specifiable as relative coordinate values.								
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>arc_rel_3d</b> has the same effect as <b>arc_rel</b> .								
<b>Call</b>	<code>arc_rel_3d( dx, dy, dz, Angle )</code>								
<b>Parameters</b>	<table> <tr> <td><code>dx</code></td><td>Position of the helical axis (parallel to the z axis) as relative coordinates. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td><code>dy</code></td><td>Relative Z end coordinate. In bits. As signed 32-bit values. Allowed value range: [-524,288...+524,287]. An out-of-range value is edge-clipped.</td></tr> <tr> <td><code>dz</code></td><td>Arc angle. In degrees. As a 64-bit IEEE floating point value. A positive sign means "clockwise". Allowed value range: [-3,600.0°...+3,600.0°] (<math>\pm 10</math> full circles). An out-of-range value is edge-clipped.</td></tr> <tr> <td><code>Angle</code></td><td></td></tr> </table>	<code>dx</code>	Position of the helical axis (parallel to the z axis) as relative coordinates. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.	<code>dy</code>	Relative Z end coordinate. In bits. As signed 32-bit values. Allowed value range: [-524,288...+524,287]. An out-of-range value is edge-clipped.	<code>dz</code>	Arc angle. In degrees. As a 64-bit IEEE floating point value. A positive sign means "clockwise". Allowed value range: [-3,600.0°...+3,600.0°] ( $\pm 10$ full circles). An out-of-range value is edge-clipped.	<code>Angle</code>	
<code>dx</code>	Position of the helical axis (parallel to the z axis) as relative coordinates. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.								
<code>dy</code>	Relative Z end coordinate. In bits. As signed 32-bit values. Allowed value range: [-524,288...+524,287]. An out-of-range value is edge-clipped.								
<code>dz</code>	Arc angle. In degrees. As a 64-bit IEEE floating point value. A positive sign means "clockwise". Allowed value range: [-3,600.0°...+3,600.0°] ( $\pm 10$ full circles). An out-of-range value is edge-clipped.								
<code>Angle</code>									
<b>Comments</b>	<ul style="list-style-type: none"> <li>The position of the helical axis (<code>dx</code>, <code>dy</code>) and the Z end coordinate (<code>dz</code>) are to be specified as relative coordinates with respect to the current position. Otherwise, <b>arc_rel_3d</b> is identical to <b>arc_abs_3d</b> (see comments there).</li> </ul>								
RTC4→RTC6	New command.  <b>RTC4 Compatibility Mode:</b> see <b>arc_abs_3d</b> .								
RTC5→RTC6	Unchanged functionality. In addition: increased value range.								
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.								
<b>References</b>	<b>arc_abs_3d</b> , <b>arc_rel</b>								



<b>Ctrl Command</b>	<b>auto_cal</b>
<b>Function</b>	Controls the functions for (automatic self-) calibration of the scan system attached to the specified scan head connector.
<b>Call</b>	ErrorCode = auto_cal( HeadNo, Command )
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value.            Allowed values:            = 1: First scan head connector.            = 2: Second scan head connector. Activation required.</p> <p>Command     Control parameter. As an unsigned 32-bit value.            Allowed value range: [0...4].            = 0: The RTC6 detects the current Home-In positions, stores them in the <b>RTC6 DLL</b> and in the <b>Flash memory</b> as Home-In reference values and initializes the gain and offset values (Gain = 1.0, Offset = 0).            = 1: The RTC6 detects the current Home-In positions, calculates and sets the new gain and offset values and thereby activates drift compensation.            = 2: The RTC6 deactivates drift compensation by initializing the gain and offset values (Gain = 1.0, Offset = 0).            = 3: The RTC6 detects the current Home-In positions (but – in comparison to <code>Command = 1</code> – leaves the gain and offset values unchanged and that is, does not activate drift compensation)            = 4: The RTC6 checks the ASC hardware whether a scan system attached to the specified scan head connector is equipped with an internal sensor system for automatic self-calibration – Home-In sensors) and returns the type and status of the detected sensor system. The detected type is also stored in the <b>Flash memory</b>.</p>
<b>Result</b>	<p>Error code or type of sensor system.            As an unsigned 32-bit value.</p> <p>3            <code>auto_cal</code> cannot be executed because the board is currently <b>BUSY</b> or <b>INTERNAL-BUSY</b>.</p> <p>6            Parameter error.</p> <p>The following error codes are only returned after <code>Command = 0...3</code>:</p> <p>0            No error.</p> <p>1, 10, 11    Home-In sensor not found (this could also mean a Home-In sensor is defective)            (1: for x axis (galvanometer scanner 2)            10: for y axis (galvanometer scanner 1)            11: for both axes).</p> <p>2, 20, 22    The spread in measured values during a measurement cycle is too high            (2: for x axis / 20: for y axis / 22: for both axes).</p> <p>4, 40, 44    Reference data not found (only for <code>Command = 1</code> and 3)            (4: for x axis / 40: for y axis / 44: for both axes).</p>



Ctrl Command	auto_cal
Result (cont'd)	<p>5, 50, 55   Calibration error (Error during calibration or error in reference data) 5: for x axis / 50: for y axis / 55: for both axes.</p> <p>9, 90   Sensor for axis X or Y is defective. Only returned after Command = 0, 1 or 3.</p> <p>The following error code is only returned after Command = 0 or 4, and even then only if no other errors occurred:</p> <p>8   Download error. The values have possibly not been saved. For this error, the <b>get_last_error</b> return code <b>RTC6_FLASH_ERROR</b> is always generated.</p> <p>The following values are only returned after Command = 4:</p> <p>100   For both axes: a sensor system of type1 is included and is functioning.</p> <p>200   For both axes: a sensor system of type2 is included and is functioning.</p> <p>19   x axis (galvanometer scanner 2): a sensor system of type1 is included and is functioning. y axis (galvanometer scanner 1): sensor system is defective.</p> <p>29   x axis (galvanometer scanner 2): a sensor system of type2 is included and is functioning. y axis (galvanometer scanner 1): sensor system is defective.</p> <p>91   x axis (galvanometer scanner 2): sensor system is defective. y axis (galvanometer scanner 1): a sensor system of type1 is included and is functioning.</p> <p>92   x axis (galvanometer scanner 2): sensor system is defective. y axis (galvanometer scanner 1): a sensor system of type2 is included and is functioning.</p> <p>99   For both axes: sensor system is defective.</p> <p>255   For both axes: there is no sensor system included in the scan system.</p>
Comments	<ul style="list-style-type: none"> <li>For usage of <b>auto_cal</b>, see <a href="#">Chapter 8.10 "Automatic Self-Calibration", page 261</a>.</li> <li>At the end of <b>auto_cal</b> execution, the RTC6 board always moves the galvanometer scanners back to the position held prior to the call, possibly with corrections attributable to changed gain and offset values.</li> <li>During determination of the current Home-In positions with <b>auto_cal</b>(Command = 0, 1 or 3), the current gain and offset corrections of the galvanometer scanners are not taken into account; neither are any head corrections or the current positions of the scanners.</li> <li>After first-time or renewed connecting a scan system, a reference value determination should be performed by <b>auto_cal</b>(Command = 0). Otherwise, the RTC6 uses the stored reference values of a previously operated (possibly different) scan system when later performing an automatic self-calibration.</li> </ul>



Ctrl Command	auto_cal
Comments (cont'd)	<ul style="list-style-type: none"> <li>After initialization of the RTC6, drift compensation is turned off (gain = 1.0, offset = 0). However, previously determined reference values are still available.</li> <li>If no appropriate Home-In reference values were stored or the scan system is not equipped with Home-In sensors, then the measurement routine for <code>auto_cal</code>(Command = 1) (axis-specific) automatically aborts and restores the prior state.</li> <li><code>auto_cal</code> is not executed, if a HeadNo or Command value is invalid (return value 6, <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>). This also applies for HeadNo = 2 if the Option "Second Scan Head Control" is not enabled (return value 6).</li> <li><code>auto_cal</code> is not executed (return value 3, <code>get_last_error</code> return code <code>RTC6_BUSY</code>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <code>pause_list</code>)</li> <li>the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li><code>auto_cal</code> is even executed, if: <ul style="list-style-type: none"> <li>a list has been paused by <code>set_wait</code> (<b>PAUSED</b> status set)</li> </ul> </li> <li>For <code>RTC6_PARAM_ERROR</code>, the <b>BUSY</b> status is not checked; therefore return codes <code>RTC6_BUSY</code> and <code>RTC6_PARAM_ERROR</code> do not occur simultaneously.</li> <li>For Command = 0, 1 or 2, a valid correction table must be loaded and assigned. Otherwise, unexpected jumps can occur when gain/offset values are updated.</li> <li>Gain and offset correction can also be directly set by <code>set_hi</code> (even for systems <i>without</i> Home-In sensors).</li> <li>Reference values determined and stored by <code>auto_cal</code>(Command = 0, 1 or 3) can be queried by <code>get_hi_pos</code>.</li> <li>ASC hardware checks are performed not just by <code>auto_cal</code>(Command = 4), but also automatically for <ul style="list-style-type: none"> <li><code>auto_cal</code>(Command = 0) and</li> <li>the first call of <code>auto_cal</code>(Command = 1) and <code>auto_cal</code>(Command = 3) if neither <code>auto_cal</code>(Command = 0) nor <code>auto_cal</code>(Command = 4) were previously executed.</li> </ul> In each case, the detected ASC hardware type gets stored in the <b>Flash memory</b> and can be subsequently queried by <code>get_auto_cal</code>. For automatic Command = 4 execution, however, no corresponding return value is generated. The return value is instead simply that of the primary Command call (see above). </li> <li>When an error occurs, a corresponding error code gets saved to the <b>Flash memory</b> (therefore, <code>get_auto_cal</code> does not return 100 or 200). As soon as hardware functionality is restored, you can clear the error from the <b>Flash memory</b> by explicitly calling <code>auto_cal</code>(Command = 0) or <code>auto_cal</code>(Command = 4). <code>auto_cal</code>(Command = 1 ) and <code>auto_cal</code>(Command = 3 ) cannot be executed as long as the error is still in the <b>Flash memory</b>. The error is <i>not</i> cleared by these calls from the <b>Flash memory</b>.</li> </ul>



<b>Ctrl Command</b>	<b>auto_cal</b>
RTC4→RTC6	The functions for automatic self-calibration (Command = 0...2) are (largely) unchanged (see version info).  New: Command = 3 and Command = 4.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_hi</a> , <a href="#">get_hi_pos</a> , <a href="#">get_auto_cal</a> , <a href="#">write_hi_pos</a>

<b>Ctrl Command</b>	<b>auto_change</b>
Function	Activates a one-time automatic list change.
Call	<code>auto_change()</code>
Comments	<ul style="list-style-type: none"> <li>• <b>auto_change</b> is synonymous with <a href="#">auto_change_pos( 0 )</a>. This starts the subsequent list at its beginning.</li> <li>• See also comments for <a href="#">auto_change_pos</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">auto_change_pos</a> , <a href="#">get_status</a> , <a href="#">read_status</a>



<b>Ctrl Command</b>	<b>auto_change_pos</b>
<b>Function</b>	Activates a one-time automatic list change and simultaneously defines the list position at which execution continues.
<b>Call</b>	auto_change_pos( Pos )
<b>Parameters</b>	Pos      Start position (list memory address) as an offset referenced to the beginning of the list to be started by the automatic list change. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>auto_change_pos</b> triggers a subsequent <i>one-time-only</i> list change or a list new start. For further list changes or list new starts, <b>auto_change_pos</b> must be called again.</li> <li>• <b>auto_change_pos</b> can be called at any time.</li> <li>• If the automatic list change is activated during processing of a list, then upon reaching <b>set_end_of_list</b> execution continues without delay at the supplied start position of the other list. If there is only <i>one</i> list (<math>\text{Mem2} = 0</math>, see <b>config_list</b>), then upon reaching <b>set_end_of_list</b> execution continues at the supplied start position of this list.</li> <li>• During processing of a list, the other list (and also the current list) can be newly loaded, see <b>Chapter 6.4.6 "Changing Lists Automatically"</b>, page 100.</li> <li>• So that <b>auto_change_pos</b> can function at all, any already active list must absolutely be finalized by <b>set_end_of_list</b>; the new list should already be loaded and the input pointer should be sufficiently ahead of the output pointer (otherwise, "old" commands are executed). If, during list execution, the end of the list is reached without encountering a <b>set_end_of_list</b>, then execution automatically continues at the beginning of the current list.</li> <li>• If an automatic list change is activated when no list is currently being processed, then checking takes place as to whether a list was already processed and the other list was started (at the supplied start position). If no list was previously executed, then "List 1" is regarded as already executed (initialization) and "List 2" is started.</li> <li>• If a list memory address outside the corresponding list area is supplied (depending on which list should be started: <math>\text{Pos} \geq \text{Mem1}</math> or <math>\text{Pos} \geq \text{Mem2}</math>), then the start position is set to the beginning of the list (<math>\text{Pos} = 0</math>).</li> <li>• If, during processing of a list, the <b>auto_change_pos( Pos &gt; 0 )</b> and <b>start_loop</b> commands are called, then upon the next <b>set_end_of_list</b> the command <b>auto_change_pos( Pos &gt; 0 )</b> is executed; and at the next one the <b>start_loop</b> command is executed.</li> </ul>



<b>Ctrl Command</b>	<b>auto_change_pos</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The current list and list execution statuses can be queried by the commands <b>read_status</b> and <b>get_status</b>.</li> <li><b>auto_change_pos</b> triggers a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a>.</li> </ul>
RTC4→RTC6	<p>Basically unchanged functionality. However:</p> <p>The list memory address (<b>Pos</b>) is supplied to the RTC6 as a relative memory address referenced to the beginning of the respective list, whereas the RTC4 is supplied an absolute memory address (0...7999). If no memory area is assigned to "List 2" by <b>config_list</b> (<b>Mem2</b> = 0), then the RTC6 command behaves like the RTC4 command.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">auto_change</a> , <a href="#">get_status</a> , <a href="#">read_status</a>

<b>Ctrl Command</b>	<b>bounce_supp</b>
Function	Debounces the external start signal.
Call	<code>bounce_supp( Length )</code>
Parameters	Length      Debouncing time in ms. As an unsigned 32-bit value. Allowed value range: [0...1023]. The 22 higher bits are ignored.
Comments	<ul style="list-style-type: none"> <li><b>bounce_supp</b> enables debouncing of start signals received at the /START, /START2 or /Slave-START inputs, see <a href="#">Section "External Start", page 276</a>. Start signals occurring within the defined debouncing time after a successful start signal are thereby suppressed.</li> <li>Recommended procedure: Start a list, which operates more than one second. If /START, /START2 or /Slave-START bounces, then an additional trigger error signal is generated, which can be detected by <b>get_marking_info</b> (Bit #8 = 1). Increase the debouncing time until this additional signal has vanished (Bit #8 = 0).</li> <li>The debouncing time default value is 0 ms.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_startstop_info</a>



<b>Normal List Command</b>	<b>camming</b>
<b>Function</b>	Enables camming functionality.
<b>Call</b>	camming ( FirstPos, NPos, EncoderNo, Ctrl, Scale, Code )
<b>Parameters</b>	<p>FirstPos      Starting address of the camming command list (absolute address in list memory). As an unsigned 32-bit value. Allowed values: [0...(2<sup>23</sup>-1)].</p> <p>NPos      Length of the camming command list (without terminating <b>set_end_of_list</b> or <b>list_return</b>). As an unsigned 32-bit value. Allowed values: [1...(2<sup>23</sup>-FirstPos)].</p> <p>EncoderNo      Number of the encoder counter, whose pulses is evaluated for controlling the camming process. As an unsigned 32-bit value. Allowed values: = 0:      Encoder counter "Encoder0". = 1:      Encoder counter "Encoder1". Higher bits are ignored.</p> <p>Ctrl      Camming control mode. As an unsigned 32-bit value. Allowed values: = 0:      RTC6 board controls laser as with a <b>[*]mark[*]</b> command. <b>camming</b> terminates automatically. = 1:      User controls laser. <b>camming</b> terminates automatically. = 2:      User controls laser. <b>camming</b> does <i>not</i> terminate automatically. The output index is clipped to [0, NPos-1]. = 3:      User controls laser. <b>camming</b> does <i>not</i> terminate automatically. The camming list is continuously processed (output index modulus NPos).</p> <p>Scale      Translation (conversion factor) between the encoder-counter value and the command index. As a 64-bit IEEE floating point value. Allowed values: <math>2^{-60} &lt;  Scale  &lt; 2^{60}</math>.</p> <p>Code      Is not used. As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 8.11 "Camming", page 265</a>.</li> <li>• If there are unallowed parameter values, <b>camming</b> is replaced by a <b>list_nop</b> already during loading (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> </ul>



<b>Normal List Command</b>	<b>camming</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Each time <b>camming</b> is called, the current encoder-counter value is ascertained for use as the new reference value. At a later time point, the corresponding command index of the camming command list is calculated for the then-current encoder-counter value (using the reference value and the <b>Scale</b> parameter). For each call of <b>camming</b>, the first command in the camming command list (index = 0) is always the first command to be processed.</li> <li><b>camming</b> waits for a scanner delay but sets no delay itself.</li> <li>If <b>Ctrl</b> = 0, then the laser is (as with a normal [*]<b>mark</b>[*] command) switched on at the beginning of the camming process and switched off after it terminates (laser delay settings are taken into account). If <b>Ctrl</b> &gt; 0, then the state of the laser is not changed; its control is the full responsibility of the user.</li> <li>If <b>Ctrl</b> = 0 or 1, then <b>camming</b> terminates automatically (in the next cycle) as soon as the index first undershoots 0 or overshoots <b>NPos</b>-1 (the final camming command to be executed is then be the one with an index of 0 or <b>NPos</b>-1). For these two modes (as always, if a list is <b>BUSY</b>), no external starts are allowed as long as <b>camming</b> has not yet terminated.</li> <li>In contrast, control modes <b>Ctrl</b> = 2 and 3 are endless. Here, the camming process can only be terminated by <b>stop_execution</b> or an external stop. However, in these two modes (as an exception) external starts are allowed if the list (or <b>camming</b>) is still active. If <b>Ctrl</b> = 2, then the index is clipped to the boundaries 0 or <b>NPos</b> - 1; for <b>Ctrl</b> = 3 it is set to modulus <b>NPos</b> (whereby the encoder speed is supposed not to be so high that a complete rotation is skipped). Thus, <b>Ctrl</b> = 3 works like a ring buffer.</li> <li><b>camming</b> is a normal list command, but with a variable execution period.</li> <li><b>camming</b> functions even if the <b>Option Processing-on-the-fly</b> is not activated.</li> <li>While <b>camming</b> is executing, <b>get_out_pointer</b> always provides the position of <b>camming</b>, not the position of the current index.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_encoder_speed</b> , <b>set_auto_laser_control</b>



<b>Undelayed Short List Command</b>	<b>clear_fly_overflow</b>
<b>Function</b>	Resets Error Bit #4...Error Bit #7, Error Bit #24 and Error Bit #25 (from <a href="#">get_marking_info</a> ) for customer-defined monitoring of Processing-on-the-fly applications.
<b>Call</b>	clear_fly_overflow( Mode )
<b>Parameters</b>	<p>Mode      Determines which error bits are to be reset.      As an unsigned 32-bit value.</p> <p>Bit #0    = 1: Error Bit #4 (underflow X) is reset.      Bit #1    = 1: Error Bit #5 (overflow X) is reset.      Bit #2    = 1: Error Bit #6 (underflow Y) is reset.      Bit #3    = 1: Error Bit #7 (overflow Y) is reset.      Bit #4    = 1: Error Bit #24 (underflow Z) is reset.      Bit #5    = 1: Error Bit #25 (overflow Z) is reset.      Bit #0...Bit #5 can be combined as desired.      Higher-order bits are ignored.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>clear_fly_overflow</b>, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>For Mode = 0, all six error bits are reset as with Mode = 15.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a>



<b>Undelayed Short List Command</b>	<b>clear_io_cond_list</b>
Function	Clears the bits of the 16-bit digital output port on the EXTENSION 1 socket connector that are set in the parameter Maskclear, if the current IOvalue at the 16-bit digital <i>input</i> port on the EXTENSION 1 socket connector meets the following condition:  $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ (that is, if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0).
Call	clear_io_cond_list( Mask1, Mask0, Maskclear )
Parameters	<p>Mask1      16-bit mask. As an unsigned 32-bit value.  Only the lower 16 bits are evaluated.</p> <p>Mask0      See Mask1.</p> <p>Maskclear    See Mask1.</p>
Comments	<ul style="list-style-type: none"> <li>• <b>clear_io_cond_list</b> clears only those bits of the digital output port that are set in the parameter Maskclear and leaves the other bits unchanged.</li> <li>• See also <b>Section "16-Bit Digital Input and 16-Bit Digital Output"</b>, page 67 and <b>Chapter 9.3.2 "Conditional Command Execution"</b>, page 281.</li> </ul>
Examples (Pascal)	<ul style="list-style-type: none"> <li>• Clear Bit #4 of the output port (DIGITAL OUT4), if Bit #0 of the input port (DIGITAL IN0) is set and bits #1 to #3 (DIGITAL IN1...3) of the input port are not set:  <b>clear_io_cond_list(\$0001, \$000E, \$0010)</b></li> <li>• Always clear Bit #15 of the output port (and leave the other bits unchanged):  <b>clear_io_cond_list(0, 0, \$8000)</b></li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_io_cond_list, write_io_port, write_io_port_mask, get_io_status, read_io_port</b>



<b>Ctrl Command</b>	<b>config_laser_signals</b>
<b>Function</b>	Configures the laser output signal types to be outputted on pin (01) (LASER1), pin (02) (LASERON) and pin (09) (LASER2) of the LASER connector.
<b>Call</b>	<code>config_laser_signals( Config )</code>
<b>Parameters</b>	<p>Config      Desired signal configuration. As an unsigned 32-bit value.</p> <p>Thereby the following bits configure: Bits # 0-1: Pin (02) (LASERON channel) Bits # 2-3: Pin (01) (LASER1 channel) Bits # 4-5: Pin (09) (LASER2 channel) Bits # 6-31 are ignored</p> <p>To the following signal type: = 0 = 00<sub>b</sub>: LASERON signal = 1 = 01<sub>b</sub>: LASER1 signal = 2 = 10<sub>b</sub>: LASER2 signal = 3 = 11<sub>b</sub>: FirstPulseKiller signal</p> <p>The default setting (after <a href="#">load_program_file</a>) is: Config = 100100<sub>b</sub> = 0x24 = 36.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The specified configuration takes effect the next time the laser is switched on. Therefore, <b>config_laser_signals</b> should not be called during an active marking procedure.</li> </ul>
<b>Examples</b>	<ul style="list-style-type: none"> <li>Config = 000111<sub>b</sub>: FirstPulseKiller signal on LASERON channel, LASER1 signal on LASER1 channel, LASERON signal on LASER2 channel. In this configuration, LASER1 signals synchronously generated with the LASERON signal can be immediately outputted, whereas the laser itself switches on only after a delay by the FirstPulseKiller signal.</li> <li>Config = 100100<sub>b</sub> (default setting): LASERON signal on the LASERON channel, LASER1 signal on the LASER1 channel, LASER2 signal on the LASER2 channel. In this configuration, LASER1 signals can only be switched on after the laser, not before it (here the LASERON signal is the laser start signal; LASER1 signals cannot be outputted before the laser start, because the RTC5/RTC6 only generates LASER1 signals if the LaserON signal is on).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">config_laser_signals_list</a>



<b>Delayed Short List Command</b>	<b>config_laser_signals_list</b>
Function	Like <a href="#">config_laser_signals</a> , but a list command.
Call	config_laser_signals_list( Config )
Parameters	Config      See <a href="#">config_laser_signals</a> .
Comments	<ul style="list-style-type: none"><li>• See <a href="#">config_laser_signals</a>.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">config_laser_signals</a>



<b>Ctrl Command</b>	<b>config_list</b>				
<b>Function</b>	Configures the list memory, that is, allocates storage positions to certain list memory areas.				
<b>Call</b>	<code>config_list( Mem1, Mem2 )</code>				
<b>Parameters</b>	<table> <tr> <td>Mem1</td> <td>Desired storage positions for list memory area "List 1". As an unsigned 32-bit value.</td> </tr> <tr> <td>Mem2</td> <td>Desired storage positions for list memory area "List 2". As an unsigned 32-bit value.</td> </tr> </table>	Mem1	Desired storage positions for list memory area "List 1". As an unsigned 32-bit value.	Mem2	Desired storage positions for list memory area "List 2". As an unsigned 32-bit value.
Mem1	Desired storage positions for list memory area "List 1". As an unsigned 32-bit value.				
Mem2	Desired storage positions for list memory area "List 2". As an unsigned 32-bit value.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>The RTC6 list memory contains <math>8,388,608 (= 2^{23})</math> storage positions in total. They can be divided into three areas ("lists") by <b>config_list</b>. The sizes of "List 1" and "List 2" are specified by the parameters <code>Mem1</code> and <code>Mem2</code>. All remaining storage positions not assigned to "List 1" or "List 2" are automatically assigned by <b>config_list</b> to the protected list memory area "List 3".</li> <li>The following rules apply to the parameters <code>Mem1</code> and <code>Mem2</code> (invalid values are automatically corrected in the specified order):       <ul style="list-style-type: none"> <li><code>Mem1 &gt; 0</code> ("List 1" must not be empty). <code>Mem1 = 0</code> is corrected to <code>Mem1 = 1</code>.</li> <li><code>Mem1 ≤ 2<sup>23</sup></code> ("List 1" can contain a maximum of <math>2^{23}</math> storage positions). <code>Mem1 &gt; 2<sup>23</sup></code> is corrected to <code>Mem1 = 2<sup>23</sup></code>.</li> <li><code>Mem1 = "-1"</code> is interpreted as <code>Mem1 = (2<sup>32</sup>-1)</code> and corrected to <code>Mem1 = 2<sup>23</sup></code>. Example: With <code>config_list( -1, x )</code> where <code>x</code> is any value (also <code>x = -1</code>), "List 1" is automatically assigned the entire list memory (<code>Mem1 = 2<sup>23</sup>, Mem2 = 0</code>, no memory for "List 3").</li> <li><code>Mem2 ≤ 2<sup>23</sup> - Mem1</code> ("List 2" can maximally receive the "rest" of list memory). <code>Mem2 = 0</code> is allowed. <code>Mem2 &gt; 2<sup>23</sup> - Mem1</code> is corrected to <code>Mem2 = 2<sup>23</sup> - Mem1</code>.</li> <li><code>Mem2 = "-1"</code> is interpreted as <code>Mem2 = (2<sup>32</sup>-1)</code> and corrected to <code>Mem2 = 2<sup>23</sup> - Mem1</code>. Example: With <code>config_list( Mem1, -1 )</code>, "List 2" is automatically assigned with the "rest" of list memory (<code>Mem2 = 2<sup>23</sup> - Mem1</code>, no memory for "List 3").</li> <li>Storage positions for "List 3": <math>2^{23} - \text{Mem1} - \text{Mem2}</math>.</li> </ul> </li> <li>The RTC6 list memory contains <math>2^{23}</math> storage positions in total. By default, it is preconfigured so that "List 1" and "List 2" can each accept 4,194,304 (<math>= 2^{22}</math>) list commands (<code>Mem1 = Mem2 = 4,194,304</code>). The protected list memory area "List 3" owns no storage positions, because <math>2^{23} - \text{Mem1} - \text{Mem2} = 0</math>.</li> <li><b>config_list</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:       <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> </ul>				



Ctrl Command	<b>config_list</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Configuration by <b>config_list</b> does not alter the contents of list memory. Repeating the call with differing parameters is therefore nondestructive. However, after a configuration change, previously loaded list commands are processed in accordance with the new configuration.</li> <li>Moreover, a configuration change could in some circumstances affect the input pointer (if, prior to the reconfiguration, it pointed to a memory position which was assigned to "List 3" by the configuration change, then it is shifted to the beginning of "List 1") or affect a previously started automatic list change. This should be taken into account when further loading or executing command lists.</li> <li>Also observe the notes in <a href="#">Chapter 6.3.2 "Configuring the RTC6 List Memory"</a>, page 93.</li> <li>If you do not know the current configuration data for list memory (<code>Mem1</code> and <code>Mem2</code>), you can find out after <code>load_list( ListNo, 0 )</code> or <code>set_start_list_pos( ListNo, 0 )</code> by using <code>get_list_space</code> (with "ownership" changes, <code>get_config_list</code> must be called in advance).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality. In addition: increased value range and changed initialization values.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#"><b>get_config_list</b></a>



<b>Ctrl Command</b>	<b>control_command</b>															
<b>Function</b>	Sends a control command to an <i>iDRIVE</i> scan system (see glossary entry on <a href="#">page 879</a> ).															
<b>Call</b>	control_command( Head, Axis, Data )															
<b>Parameters</b>	Head	Number of the scan head connector. As an unsigned 32-bit value. Allowed values: = 1: First scan head connector. = 2: Second scan head connector.														
	Axis	Number of the axis. As an unsigned 32-bit value. Allowed values: = 1: x axis (STATUS channel, galvanometer scanner 2). = 2: y axis (STATUS1 channel, galvanometer scanner 1).														
	Data	Command code with optional parameter. As an unsigned 32-bit value. The upper part (bits #16-31) of this parameter is <i>not</i> evaluated, only the lower part (bits #0-15) is evaluated. The more significant data byte of the lower part <b>Code<sub>H</sub></b> (bits #8-15) represents a command code and the less significant data byte <b>Code<sub>L</sub></b> (bits #0-7) an optional parameter. For each command code the corresponding command and its allowed parameter values are described below.  Example: With Data = 0501 <sub>H</sub> , that is, Code <sub>H</sub> = 05 <sub>H</sub> (command SetMode) and Code <sub>L</sub> = 01 <sub>H</sub> , the actual position is selected to be returned from the scan system.														
	<b>Code<sub>H</sub></b>	<b>Command and Parameter Values (Code<sub>L</sub>)</b>														
	05 <sub>H</sub>	<p><b>SetMode:</b> This command selects the data signal to be returned from the scan system for the specified axis by the respective status channel. See also <a href="#">Chapter 8.1.2 "Configuring the Data Signal Return Behavior of the Scan System", page 200</a>. Each Code<sub>L</sub> parameter value corresponds to a particular data type.</p> <ul style="list-style-type: none"> <li>Default setting: Code<sub>L</sub> = 00<sub>H</sub> (XY2-100 status word)</li> <li>The data types marked in grey are only relevant for the intelliWELD.</li> </ul> <p><i>iDRIVE</i> scan systems with SL2-100 interface return signed 20-bit values. Scan systems with XY2-100 Enhanced interface return signed 16-bit values; here the XY2-100 converter converts (by multiplying by 16) the returned values to signed 20-bit values.</p>														
	<b>Code<sub>L</sub></b>	<b>Data Signal Type returned from the Scan System</b>														
	00 <sub>H</sub>	<p>XY2-100 status word</p> <table> <tr> <td>Bit #19 (<b>MSB</b>), 11</td> <td>= 1: The scan system servo control is ready for input data ("PowerOK", actually corresponds to a "ServoOK").</td> </tr> <tr> <td>Bit #18, 10</td> <td>= 1: The galvanometer scanner temperature within optimal range ("TempOK").</td> </tr> <tr> <td>Bit #16, 15, 8, 7</td> <td>= 1: The x- and y axis position errors are within allowed range ("PosAck"-Signal).</td> </tr> <tr> <td>Bit #17, 12, 9, 4</td> <td>= 1.</td> </tr> <tr> <td>Bit #14, 6</td> <td>= 1 (For intelliWELD and for scan systems with sensors for automatic self calibration: Reserved).</td> </tr> <tr> <td>Bit #13, 5</td> <td>= 0.</td> </tr> <tr> <td>Bit #0...3</td> <td>= 0.</td> </tr> </table>	Bit #19 ( <b>MSB</b> ), 11	= 1: The scan system servo control is ready for input data ("PowerOK", actually corresponds to a "ServoOK").	Bit #18, 10	= 1: The galvanometer scanner temperature within optimal range ("TempOK").	Bit #16, 15, 8, 7	= 1: The x- and y axis position errors are within allowed range ("PosAck"-Signal).	Bit #17, 12, 9, 4	= 1.	Bit #14, 6	= 1 (For intelliWELD and for scan systems with sensors for automatic self calibration: Reserved).	Bit #13, 5	= 0.	Bit #0...3	= 0.
Bit #19 ( <b>MSB</b> ), 11	= 1: The scan system servo control is ready for input data ("PowerOK", actually corresponds to a "ServoOK").															
Bit #18, 10	= 1: The galvanometer scanner temperature within optimal range ("TempOK").															
Bit #16, 15, 8, 7	= 1: The x- and y axis position errors are within allowed range ("PosAck"-Signal).															
Bit #17, 12, 9, 4	= 1.															
Bit #14, 6	= 1 (For intelliWELD and for scan systems with sensors for automatic self calibration: Reserved).															
Bit #13, 5	= 0.															
Bit #0...3	= 0.															
	01 <sub>H</sub>	Actual (angular) position / bit [-2 <sup>19</sup> ... 2 <sup>19</sup> -1].														
	02 <sub>H</sub>	Set (angular) position / bit [-2 <sup>19</sup> ... 2 <sup>19</sup> -1].														
	03 <sub>H</sub>	Position error (= set position – actual position) / bit [-2 <sup>19</sup> ... 2 <sup>19</sup> -1].														



Ctrl Command	control_command		
	Code <sub>H</sub>	Code <sub>L</sub>	Data Signal Type returned from the Scan System
	(05 <sub>H</sub> )	04 <sub>H</sub>	Actual current (output stage current) / 16 <sup>-1</sup> mA [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		05 <sub>H</sub>	Relative galvanometer scanner control [-16000...16000] (not with intellicube) (the return value 16 corresponds to 1%).
		06 <sub>H</sub>	Actual (angular) velocity / (bit/ms) [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		12 <sub>H</sub>	Actual z axis position / bit [-2 <sup>19</sup> ...2 <sup>19</sup> -1] (only for intelliWELD with varioSCAN FC; not with varioSCAN FC i). This data signal can only be read by the channel of the scan head connector to which the z axis is connected.
		14 <sub>H</sub>	The galvanometer scanner temperature / 160 <sup>-1</sup> °C [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		15 <sub>H</sub>	Servo board temperature / 160 <sup>-1</sup> °C [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		16 <sub>H</sub>	AGC voltage (PD supply voltage) / 1600 <sup>-1</sup> V [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		17 <sub>H</sub>	DSP core supply voltage (1.8 V) / 1600 <sup>-1</sup> V [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		18 <sub>H</sub>	DSP IO voltage (3.3 V) / 1600 <sup>-1</sup> V [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		19 <sub>H</sub>	Analog section voltage (9 V) / 1600 <sup>-1</sup> V [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		1A <sub>H</sub>	AD converter supply voltage (5 V) / 1600 <sup>-1</sup> V [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		1B <sub>H</sub>	AGC current (PD supply current) / 16 <sup>-1</sup> mA [-2 <sup>19</sup> ...2 <sup>19</sup> -1].
		1D <sub>H</sub>	Relative heating output of the corresponding galvanometer scanner heater (not relevant for intellicube, dynAXIS XS and scan systems with water-cooled scanners). Bits #4...19                    Relative heating output / % [0...1000]. Bits #0...3                    = 0.
		1E <sub>H</sub>	Serial number (lower 16 bits) Bits #4...19                    Serial number (lower 16 bits) [0...2 <sup>16</sup> -1]. Bits #0...3                    = 0.
		1F <sub>H</sub>	Serial number (higher 16 bits) Bits #4...19                    Serial number (higher 16 bits) [0...2 <sup>16</sup> -1]. Bits #0...3                    = 0.
		20 <sub>H</sub>	Article number (lower 16 bits) Bits #4...19                    Article number (lower 16 bits) [0...2 <sup>16</sup> -1]. Bits #0...3                    = 0.
		21 <sub>H</sub>	Article number (higher 16 bits) Bits #4...19                    Article number (higher 16 bits) [0...2 <sup>16</sup> -1]. Bits #0...3                    = 0.
		22 <sub>H</sub>	Firmware version number Bits #4...19                    Version number [0...2 <sup>16</sup> -1]. Bits #0...3                    = 0.



Ctrl Command	control_command		
	Code <sub>H</sub>	Code <sub>L</sub>	Data Signal Type returned from the Scan System
	(05 <sub>H</sub> )	23 <sub>H</sub>	Calibration angle (mechanical scan angle ( $\pm$ ) for 96% of the maximum and minimum control value, that is, for $\pm 503316$ bit)
			Bits #4...19      Calibration angle / $10^{-3}$ ° [0...2 <sup>16</sup> -1]. Bits #0...3      = 0.
		24 <sub>H</sub>	Aperture
			Bits #4...19      Aperture / mm [0...2 <sup>16</sup> -1]. Bits #0...3      = 0.
		25 <sub>H</sub>	Wavelength
			Bits #4...19      Wavelength / nm [0...2 <sup>16</sup> -1]. Bits #0...3      = 0.
		26 <sub>H</sub>	Tuning number (as of Firmware Version 2078)
			Bits #12...19      Start setting. Bits #4...11      Current setting. Bits #0...3      = 0.
		27 <sub>H</sub>	only after control_command(Data = 17FF <sub>H</sub> ): Number of the internally (for example, by Code = 17FF <sub>H</sub> for reinstatement by Code = 1700 <sub>H</sub> ) buffered, specified to be returned data type (as of Firmware Version 2078)
			Bits #12...19      = 0. Bits #4...11      Temporarily stored setting. Bits #0...3      = 0.
		2A <sub>H</sub>	Stop Event Code = 00010 <sub>H</sub> :      The galvanometer scanner has reached a critical edge position. = 00020 <sub>H</sub> :      AD converter error. = 00030 <sub>H</sub> :      Temperature in scan system above max. allowed value. = 00040 <sub>H</sub> :      External power supply voltages have dropped below the allowed value. = 00050 <sub>H</sub> :      Flags are not valid. = 00060 <sub>H</sub> - 000C0 <sub>H</sub> :      Reserved. = 000D0 <sub>H</sub> :      Watchdog 10 $\mu$ s time out (loop time exceeded). = 000E0 <sub>H</sub> :      Position Acknowledge time out (set position not reached for long time). = 000F0 <sub>H</sub> :      Reserved.
		2F <sub>H</sub>	Running time (seconds) (as of Firmware Version 2061)
			Bits #4...19      Running time (seconds) / s [0...59]. Bits #0...3      = 0.
		30 <sub>H</sub>	Running time (minutes) (as of Firmware Version 2061)
			Bits #4...19      Running time (minutes) / min [0...59]. Bits #0...3      = 0.



Ctrl Command	control_command		
	Code <sub>H</sub>	Code <sub>L</sub>	Data Signal Type returned from the Scan System
	(05 <sub>H</sub> )	31 <sub>H</sub>	Running time (hours) (as of Firmware Version 2061) Bits #4...19                    Running time (hours) / h [0...23]. Bits #0...3                    = 0.
		32 <sub>H</sub>	Running time (days) (as of Firmware Version 2061) Bits #4...19                    Running time (days) / d [0...2 <sup>16</sup> -1]. Bits #0...3                    = 0.
	The data types marked in grey are only relevant for the intelliWELD.		
		33 <sub>H</sub>	3.3 V sensor board operating voltage Bits #4...19                    Operating voltage / 10 <sup>-3</sup> V [0...7000]. Bits #0...3                    = 0.
		34 <sub>H</sub>	Sensor board operating temperature Bits #4...19                    Operating temperature / 0.1°C [0...65,535]. Bits #0...3                    = 0.
		35 <sub>H</sub>	Purge air flow rate Bits #4...19                    (Purge air flow rate - 4 l/min) * 109.89 min/l [0 ... 2250]. Bits #0...3                    = 0.
		36 <sub>H</sub>	Temperature of mirror 2 Bits #4...19                    (Temperature + 26.6°C) / 0.05°C [0...1992]. Bits #0...3                    = 0.
		37 <sub>H</sub>	Temperature of mirror 1 Bits #4...19                    (Temperature + 26.6°C) / 0.05°C [0...1992]. Bits #0...3                    = 0.
		38 <sub>H</sub>	Protective-window temperature Bits #4...19                    Temperature / 0.044°C [0...2250]. Bits #0...3                    = 0.
		39 <sub>H</sub>	Collimator temperature Bits #4...19                    Temperature / 0.05°C [0...2250]. Bits #0...3                    = 0.
		3A <sub>H</sub>	Galvanometer-mount temperature Bits #4...19                    Temperature / 0.05°C [0...2250]. Bits #0...3                    = 0.
		3B <sub>H</sub>	Coolant-flow rate Bits #4...19                    (Coolant-flow rate + 0.93 l × min <sup>-1</sup> ) / (0.00525 l × min <sup>-1</sup> ) [0...2250]. Bits #0...3                    = 0.
		3C <sub>H</sub>	Protective-window scattered light value Bits #4...19                    Scattered light value / 0.00444 V [0...2250]. Bits #0...3                    = 0.



Ctrl Command	control_command																				
	Code <sub>H</sub>	Code <sub>L</sub>	Data Signal Type returned from the Scan System																		
	<i>(05<sub>H</sub>)</i> The data types marked in grey are only relevant for the intelliWELD.																				
		3D <sub>H</sub>	<p><b>Flags sensor board</b></p> <p>(*) if "1" then an INTERLOCK error is initiated</p> <table> <tr><td>Bit #19 (<b>MSB</b>)</td><td>= 1: Protective-window temperature value not in [0...1882].</td></tr> <tr><td>Bit #18</td><td>= 1: Temperature of mirror 1 value not in [0...2250].</td></tr> <tr><td>Bit #17</td><td>= 1: Temperature of mirror 2 value not in [0...2250].</td></tr> <tr><td>Bit #14(*)</td><td>= 1: Emerging-beam-opening temperature value not in [0...1200].</td></tr> <tr><td>Bit #7</td><td>= 1: Protective-window scattered light value not in [0...2250].</td></tr> <tr><td>Bit #6</td><td>= 1: Coolant-flow rate value not in [750...2250].</td></tr> <tr><td>Bit #5 (*)</td><td>= 1: Galvanometer-mount temperature value not in [0...1600].</td></tr> <tr><td>Bit #4 (*)</td><td>= 1: Collimator temperature value not in [0...2000].</td></tr> <tr><td>Bits #0...3</td><td>= 0.</td></tr> </table>	Bit #19 ( <b>MSB</b> )	= 1: Protective-window temperature value not in [0...1882].	Bit #18	= 1: Temperature of mirror 1 value not in [0...2250].	Bit #17	= 1: Temperature of mirror 2 value not in [0...2250].	Bit #14(*)	= 1: Emerging-beam-opening temperature value not in [0...1200].	Bit #7	= 1: Protective-window scattered light value not in [0...2250].	Bit #6	= 1: Coolant-flow rate value not in [750...2250].	Bit #5 (*)	= 1: Galvanometer-mount temperature value not in [0...1600].	Bit #4 (*)	= 1: Collimator temperature value not in [0...2000].	Bits #0...3	= 0.
Bit #19 ( <b>MSB</b> )	= 1: Protective-window temperature value not in [0...1882].																				
Bit #18	= 1: Temperature of mirror 1 value not in [0...2250].																				
Bit #17	= 1: Temperature of mirror 2 value not in [0...2250].																				
Bit #14(*)	= 1: Emerging-beam-opening temperature value not in [0...1200].																				
Bit #7	= 1: Protective-window scattered light value not in [0...2250].																				
Bit #6	= 1: Coolant-flow rate value not in [750...2250].																				
Bit #5 (*)	= 1: Galvanometer-mount temperature value not in [0...1600].																				
Bit #4 (*)	= 1: Collimator temperature value not in [0...2000].																				
Bits #0...3	= 0.																				
		3F <sub>H</sub>	<p><b>Position value scale factor setting (as of Firmware Version 2072)</b></p> <table> <tr><td>Bits #6...19</td><td>Reserved.</td></tr> <tr><td>Bits #4, 5</td><td>Scale factor = <math>1/2^n</math> with <math>n = 2 \times</math> Bit #5 + Bit #4.</td></tr> <tr><td>Bits #0...3</td><td>= 0.</td></tr> </table>	Bits #6...19	Reserved.	Bits #4, 5	Scale factor = $1/2^n$ with $n = 2 \times$ Bit #5 + Bit #4.	Bits #0...3	= 0.												
Bits #6...19	Reserved.																				
Bits #4, 5	Scale factor = $1/2^n$ with $n = 2 \times$ Bit #5 + Bit #4.																				
Bits #0...3	= 0.																				



Ctrl Command	control_command							
	Code <sub>H</sub>	Code <sub>L</sub> ( <i>Command and Parameter Values</i> )						
	0A <sub>H</sub>	<p><b>UpdatePermanentMemory</b> (as of Firmware Version 2078): This command (with its only allowed parameter value of Code<sub>L</sub> = 00<sub>H</sub>) causes the currently set servo behavior to also be the default start behavior following subsequent resets. See also <a href="#">Chapter 8.1.8 "Configuring the Start Behavior", page 209</a>.</p>						
	0E <sub>H</sub>	<p><b>SetControlDefinitionMode</b> This command specifies which information about a particular tuning (that is, the corresponding servo algorithm) is to be returned from the scan system by the selected status channel. Code<sub>L</sub> [0...3] thereby defines the tuning number. As with the SetMode command, tuning information are returned as signed 20-bit values.</p>						
	Code <sub>L</sub> <i>Data Signal Returned from Scan System</i>							
	00 <sub>H</sub>	<table> <tr> <td>Bit #19 (MSB)</td> <td>= 0: Tuning available. = 1: Tuning not available.</td> </tr> <tr> <td>...</td> <td></td> </tr> </table>	Bit #19 (MSB)	= 0: Tuning available. = 1: Tuning not available.	...			
Bit #19 (MSB)	= 0: Tuning available. = 1: Tuning not available.							
...								
	03 <sub>H</sub>	<table> <tr> <td>Bits #8...18</td> <td>Reserved.</td> </tr> <tr> <td>Bits #4...7</td> <td>Tuning type: = 0: Vector tuning (microstepping). = 1: Jump tuning. = 2,3: Reserved.</td> </tr> <tr> <td>Bits #0...3</td> <td>= 0.</td> </tr> </table>	Bits #8...18	Reserved.	Bits #4...7	Tuning type: = 0: Vector tuning (microstepping). = 1: Jump tuning. = 2,3: Reserved.	Bits #0...3	= 0.
Bits #8...18	Reserved.							
Bits #4...7	Tuning type: = 0: Vector tuning (microstepping). = 1: Jump tuning. = 2,3: Reserved.							
Bits #0...3	= 0.							
	11 <sub>H</sub>	<p><b>SelectControlDefinition</b> (as of Firmware Version 2078): For scan systems equipped with multiple tunings (that is, servo algorithms), this command selects a desired tuning. See also <a href="#">Chapter 8.1.4 "Selecting the Dynamics Setting (Tuning)", page 202</a>. Code<sub>L</sub> thereby defines the tuning number. Default setting: Code<sub>L</sub> = 00<sub>H</sub>.</p>						
	12 <sub>H</sub>	<p><b>SetPositionScale</b> (as of Firmware Version 2072): This command can be used to set a (down) scale factor for the position values (the servo electronic scales the position values received from the RTC6 by the specified factor, see also <a href="#">Chapter 8.1.7 "Configuring the Effective Calibration", page 208</a>). The command must be issued initially with the parameter Code<sub>L</sub> = 83<sub>H</sub> and a second time with the desired parameter value (see the following list). In the default setting (Code<sub>L</sub> = 00<sub>H</sub>), the scale factor is 1 (no scaling).</p>						
	Code <sub>L</sub> <i>Effect on the scale factor</i>							
	00 <sub>H</sub>	The scale factor is set to the value 1 (but first execute the command code Data = 1283 <sub>H</sub> !)						
	01 <sub>H</sub>	The scale factor is set to the value 1/2 (but first execute the command code Data = 1283 <sub>H</sub> !)						
	02 <sub>H</sub>	The scale factor is set to the value 1/4 (but first execute the command code Data = 1283 <sub>H</sub> !)						
	03 <sub>H</sub>	The scale factor is set to the value 1/8 (but first execute the command code Data = 1283 <sub>H</sub> !)						



Ctrl Command	control_command	
	Code <sub>H</sub>	Code <sub>L</sub> ( <i>Command and Parameter Values</i> )
	15 <sub>H</sub>	<p><b>SetPosAcknowledgelevel</b> (as of Firmware Version 2066):            This command sets the PosAcknowledge threshold value. See also <a href="#">Chapter 8.1.6 "Configuring the PosAcknowledge Threshold Value", page 208</a>. The parameter value Code<sub>L</sub> is the desired PosAcknowledge threshold value in bits [00<sub>H</sub>...FF<sub>H</sub>].            See also the comments on the SetPosAcknowledgelevel command, <a href="#">page 340</a>.</p>
	17 <sub>H</sub>	<p><b>Store/RestoreTransmissionMode</b> (as of Firmware Version 2078):            This command allows the currently used status return data type to be temporarily stored (Code<sub>L</sub> = FF<sub>H</sub>) and then reinstated at a later time (Code<sub>L</sub> = 00<sub>H</sub>). See also <a href="#">Code = 0527<sub>H</sub></a>.</p>
	21 <sub>H</sub>	<p><b>SetEchoMode</b> (as of Firmware Version 2078):            This command verifies whether data transfer is intact. See also <a href="#">Chapter 8.1.9 "Fault Diagnosis and Functional Test", page 209</a>. It transfers an 8-bit value (Code<sub>L</sub>) to the scan system and causes a 20-bit value to be returned on the corresponding status channel: if data transfer is error-free, then the upper 8 bits of the returned 20-bit value are identical with Code<sub>L</sub> and the next lower 8 bits are identical with the complement of Code<sub>L</sub> (NOT Code<sub>L</sub>).            Example: For <code>control_command( 1, 1, 0x210A )</code> and if data transfer is error-free, (<code>get_value(1) AND 0xFFFF0</code>) returns 0x0AF50.</p>
Comments	<p><i>General comments</i></p> <ul style="list-style-type: none"> <li>The <b>control_command</b> command can only be used in conjunction with iDRIVE scan systems (see glossary entry on <a href="#">page 879</a>). Conventional scan systems without iDRIVE technology ignore the command.</li> <li>With invalid values of Head and/or Axis, the command is <i>not</i> executed (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>Some parameters of <b>control_command</b> are not usable with older firmware versions of "intelli" scan systems. These parameters are specifically noted. All other parameters are usable with any firmware version.</li> <li>Under some circumstances, <b>control_command</b> might be unavailable at the first scan head connector if speed-dependent laser control has been activated by <b>set_auto_laser_control</b> (Mode = 2).</li> <li>For other command codes not mentioned here, see the manual of the respective scan system.</li> </ul>	



Ctrl Command	<b>control_command</b>
Comments (cont'd)	<p><i>Comments regarding the commands SetMode (Code<sub>H</sub> = 05<sub>H</sub>), SetControlDefinitionMode (Code<sub>H</sub> = 0E<sub>H</sub>), SetEchoMode (Code<sub>H</sub> = 21<sub>H</sub>) and RestoreTransmissionMode (Data = 1700<sub>H</sub>)</i></p> <ul style="list-style-type: none"> <li>The data type selected by <b>control_command</b> (Code<sub>H</sub> = 05<sub>H</sub>, 0E<sub>H</sub> or 21<sub>H</sub> or Data = 1700<sub>H</sub>) is transmitted until another data type is selected.</li> <li>Data returned from the scan system to the RTC6 can be queried by <b>get_value</b>, <b>get_values</b>, <b>set_trigger/set_trigger4</b> and <b>get_waveform</b>. Switching to a different data source causes a short (serial transmission-related) delay before transmission of the first data. After switching data sources, therefore, a delay time of up to 60 µs can occur before reading the data. <b>control_command</b> always automatically inserts a waiting time of 60 µs after the data source is switched (the previously mentioned commands always return correct values).</li> <li>All data returned from the scan system to the RTC6 are passed over as signed 20-bit values. This applies even if the <b>RTC6 DLL</b> is set to <b>RTC4 Compatibility Mode</b> and even for scan systems without SL2-100 interface, controlled by an XY2-100 converter. Queried data returned by <b>get_value</b>, <b>get_values</b> or <b>get_waveform</b> are nevertheless generally transferred to the PC as 32-bit signed values (for data evaluation, see comments for <b>get_value</b>).</li> <li>For scan systems with integrated SL2-100 interface, <b>get_head_status</b> queries the XY2-100 status word regardless of settings made by <b>control_command</b> (Code<sub>H</sub> = 05<sub>H</sub>, 0E<sub>H</sub> or 21<sub>H</sub> or Data = 1700<sub>H</sub>). It is returned in addition to the selected data. In contrast, if iDIVE scan systems (<i>without</i> an integrated SL2-100 interface) are controlled by an XY2-100 converter, <b>get_head_status</b> returns the XY2-100 status word only if this signal was previously selected to be returned from the scan system by <b>control_command</b>, see also <b>Section "Reading Out Data"</b>, page 200.</li> <li>After a reset or power-up of the scan system, it can take around 5 seconds for data to be returned from the scan system (see also <b>get_value</b>). After a reset or power-up of the scan system, always the XY2-100 status word is returned.</li> <li><b>control_command( Data = 1700<sub>H</sub> )</b> has no effect if a power-up or reset was executed after the most recent execution of the <b>StoreTransmissionMode</b> command (Data = 17FF<sub>H</sub>).</li> </ul> <p><i>Comments regarding the SetMode command (Code<sub>H</sub> = 05<sub>H</sub>)</i></p> <ul style="list-style-type: none"> <li>Set (angular) position values returned by the scan system correspond to the effective output values Sample&lt;AX...BY&gt;_Out with <b>set_trigger/set_trigger4</b> and take into account any defined wobbel and Processing-on-the-fly corrections, coordinate transformations, image field correction as well as offset and gain compensations for automatic self-calibration of the scan system. Additionally, the 20-bit set position values returned from a z axis (and in <b>RTC4 Compatibility Mode</b> from the x and y axes, too) are scaled by a factor 16 relating to the therefore specified 16-bit coordinate values.</li> <li>The angle bit-values (actual position, set position, position error) or angle bit/ms-values (actual speed) returned to the RTC6 can be converted into °-values or %/ms-values by the scan system's calibration angle. The calibration angle can be read out by Data = 0523<sub>H</sub>.</li> <li>Exact values for the internal voltages referred to in the table can vary for different versions of the scan system.</li> <li>intelliWELD scan systems can be supplied with various number of sensors. Reference the scan system's user manual. This RTC6 manual lists the command codes for all its theoretically possible sensors.</li> </ul>



Ctrl Command	<b>control_command</b>
Comments (cont'd)	<p><i>Comments on the SetPositionScale command (Code<sub>H</sub> = 12<sub>H</sub>)</i></p> <ul style="list-style-type: none"> <li>If the scale factor for scaling the position values is changed, then the user program calibration factor for calculating RTC6 control values (in bits) from the desired image field position values (in mm), see <a href="#">Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156</a>, needs to be subjected to an inverse proportional scaling. In addition, the jump and mark speed should be adjusted.</li> <li>The currently set scaling factor can be queried with Data = 053F<sub>H</sub>.</li> </ul> <p><i>Comments on the SetPosAcknowledgelevel command (Code<sub>H</sub> = 15<sub>H</sub>)</i></p> <ul style="list-style-type: none"> <li>The threshold value must be specified related to a 16-bit position range.</li> <li>The default start behavior is for the scan system to set the threshold value to 0.28% of the <i>full</i> angular range of an axis (LSB16 or 1/65536, Code<sub>L</sub> = 183 = B7<sub>H</sub>) after every power-up or reset. If other threshold values are desired, they must be separately set for each axis (Code<sub>H</sub> = 15<sub>H</sub>). SCANLAB recommends setting only threshold values (Code<sub>L</sub>) above 14<sub>H</sub> (that is, 0.03% of the full position range). Lower values can lead to frequent system safety shutdowns due to Position Acknowledge time outs (set position not reached for an excessive time).</li> </ul>
RTC4→RTC6	Unchanged functionality.  Note that all returned data selected by <b>control_command</b> are always in the RTC6's 20-bit range, even in <a href="#">RTC4 Compatibility Mode</a> (see also comments above and comments for <a href="#">get_value</a> ).
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_value</a> , <a href="#">get_values</a> , <a href="#">get_head_status</a> , <a href="#">set_trigger</a> , <a href="#">set_trigger4</a> , <a href="#">get_waveform</a>



<b>Ctrl Command</b>	<b>copy_dst_src</b>
<b>Function</b>	Creates entries in the internal management table for an indexed character, text string or subroutine with the specified index ( <b>Dst</b> ) by copying the table entries of another index ( <b>Src</b> ).
<b>Call</b>	copy_dst_src( <b>Dst</b> , <b>Src</b> , <b>Mode</b> )
<b>Parameters</b>	<p><b>Dst</b> Index of the indexed character, text string or subroutine whose entries should be copied from <b>Src</b>. As an unsigned 32-bit value. Allowed value range: [0...1023] for indexed characters or subroutines, [1024+0...1024+41] for indexed text strings.</p> <p><b>Src</b> Index of the indexed character, text string or subroutine whose entries should be copied to <b>Dst</b>. As an unsigned 32-bit value. Allowed value range: [0...1023] for indexed characters or subroutines, [1024+0...1024+41] for indexed text strings.</p> <p><b>Mode</b> Determines which management tables are to be changed. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>Bit #0 = 0: <b>Dst</b> is the index of an indexed character or the index of an indexed text string.</li> <li>Bit #0 = 1: <b>Dst</b> is the index of an indexed subroutine.</li> <li>Bit #1 = 0: <b>Src</b> is the index of an indexed character or the index of an indexed text string.</li> <li>Bit #1 = 1: <b>Src</b> is the index of an indexed subroutine.</li> <li>Bit #2...Bit #31: Not evaluated.</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• If an index value (<b>Dst</b> and/or <b>Src</b>) is invalid, then <b>copy_dst_src</b> is ignored (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>• <b>copy_dst_src</b> creates an additional reference (index) to an indexed character, text string or subroutine (that can also be called with this new index). <b>copy_dst_src</b> only alters the corresponding entry in the internal management table and does not modify the content of the list memory. This allows copying, renumbering or converting between indexed characters, text strings or subroutines without having to reload each time.</li> <li>• A real copy of an indexed character, text string or subroutine in the protected list memory area "List 3" can be created (subsequent to <b>copy_dst_src</b>) with <b>save_disk/load_disk</b>. Characters, text strings and/or subroutines with multiple references are thereby written several times to the list memory. Keep this in mind in order to prevent unintended memory overflow of the protected list memory area "List 3".</li> <li>• See also <b>Section "Managing Indexed Characters and Text Strings"</b>, page 110.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>save_disk, load_disk</b>



<b>Ctrl Command</b>	<b>create_dat_file</b>
<b>Function</b>	Generates a new <code>RTC6DAT.dat</code> file of the current version.
<b>Call</b>	<code>Version = create_dat_file ( Flag )</code>
<b>Parameters</b>	<p>Flag      As a signed 32-bit value.</p> <p>  1:      Generates a new <code>RTC6DAT.dat</code> file and returns the current version.</p> <p>  -1:     Only returns the current version.</p>
<b>Result</b>	<p>Version    As an unsigned 32-bit value.</p> <p>  &gt; 600: Current version.</p> <p>  0:      <code>create_dat_file</code> cannot be executed  <code>(get_last_error</code> return code <code>RTC6_BUSY</code>  or no program is loaded on the RTC6).</p> <p>  1:      Not enough Windows memory  <code>(get_last_error</code> return code <code>RTC6_OUT_OF_MEMORY</code>).</p> <p>  2:      The output file cannot be opened  <code>(get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).  Make sure that you have write permissions granted for the output path containing the <code>RTC6DAT.dat</code>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Earlier <code>RTC6DAT.dat</code> versions cannot be used anymore.  Otherwise, <code>load_program_file</code> returns error code 7  <code>(get_last_error</code> return code <code>RTC6_VERSION_MISMATCH</code>).</li> <li>The <code>RTC6DAT.dat</code> version information is available only after <code>load_program_file</code> has also been executed with the current <code>RTC6 DLL</code> instance.</li> <li>The <code>RTC6DAT.dat</code> version information is not taken over when acquiring an RTC6 board (no matter if it is already initialized or not).</li> <li>A newly generated <code>RTC6DAT.dat</code> (of a specific version) can be used at any time for all RTC6 boards that require the same version.</li> <li>Each newly generated <code>RTC6DAT.dat</code> contains (besides static initializations) also: <ul style="list-style-type: none"> <li>user definable tables which have been loaded earlier by <code>load_varpolydelay</code>, <code>load_auto_laser_control</code>, <code>load_jump_table</code>, <code>load_jump_table_offset</code>, <code>load_position_control</code></li> <li>a “freely definable wobble shape” which have been defined earlier by <code>set_wobble_vector</code></li> </ul> The user definable tables are automatically available after the next <code>load_program_file</code> whereas the “freely definable wobble shape” requires a <code>set_wobble_mode( Mode &gt; 1 )</code> call in addition. </li> <li>Note that <code>create_dat_file</code> overwrites an already existing <code>RTC6DAT.dat</code> without warning. Keep a copy of your original <code>RTC6DAT.dat</code> in a safe place.</li> <li><code>create_dat_file</code> cannot be executed, while a list is being processed  <code>(get_last_error</code> return code <code>RTC6_BUSY</code>).</li> </ul>



<b>Ctrl Command</b>	<b>create_dat_file</b>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 610, OUT 610, RBF 615. Last change version DAT 603.
References	<a href="#">load_program_file</a>

<b>Ctrl Command</b>	<b>disable_laser</b>
Function	Disables the signals for "laser active" operation.
Call	<code>disable_laser()</code>
Comments	<ul style="list-style-type: none"> <li>• <b>disable_laser</b> disables the signals for "laser active" operation at the output ports LASER1, LASER2 and LASERON (the signals are set to their respective "Off" level). Pin (02) of the 15-pin LASER connector is then at a fixed level. However, standby signals activated with <a href="#">set_standby</a> or <a href="#">set_standby_list</a> continue to be outputted by the LASER1 and LASER2 output ports. If the standby signals are deactivated, also pin (01) and pin (09) of the 15-pin LASER connector and pins (19) and (22) of the EXTENSION 2 socket connector are at a fixed level after <b>disable_laser</b>.</li> <li>• The signals for "laser active" operation can also be disabled by <a href="#">set_laser_control</a> and reenabled by <a href="#">set_laser_control</a> or <a href="#">enable_laser</a>.</li> <li>• After initialization of the RTC6 with <a href="#">load_program_file</a>, the laser control is <i>deactivated</i> and absolutely requires <a href="#">set_laser_control</a> for activation.</li> <li>• If the command results in an <a href="#">RTC6_TIMEOUT</a> error (for example, if no program file was loaded), the LASER1, LASER2 and LASERON output ports can only be reactivated with <a href="#">set_laser_control</a> after a <a href="#">load_program_file</a> command.</li> <li>• By <a href="#">get_startstop_info</a> (<a href="#">Bit #9</a>), the current status of the laser control signals (globally enabled, yes or no) can be queried.</li> <li>• By <a href="#">get_startstop_info</a> (<a href="#">Bit #14</a>), the status "laser disabled" can be queried.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">enable_laser</a> , <a href="#">set_laser_control</a> , <a href="#">get_startstop_info</a>



<b>Ctrl Command</b>	<b>enable_laser</b>
<b>Function</b>	Enables the signals for “laser active” operation.
<b>Call</b>	<code>enable_laser()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>After a hardware reset (and after <code>load_program_file</code>), <code>set_laser_control</code> must be called to activate the LASER1, LASER2 and LASERON output ports and define the signal level, see <a href="#">Chapter 7.4 “Laser Control”, page 173</a>. Otherwise, <code>enable_laser</code> has no effect.</li> <li>After initialization of the RTC6 with <code>load_program_file</code>, the laser control signals are <i>deactivated</i>. For first-time activation, <code>set_laser_control</code> must be called (see above). After a subsequent disabling by <code>disable_laser</code> (or <code>set_laser_control</code>), the <code>enable_laser</code> (or <code>set_laser_control</code>) command can be used for reenabling.</li> <li>Even if the laser control signals have been enabled with <code>enable_laser</code> or <code>set_laser_control</code>, they are not outputted without further commands, see <a href="#">Chapter 7.4 “Laser Control”, page 173</a>.</li> <li>By <code>get_startstop_info</code> (<code>Bit #9</code>) the current status of the laser control signals (globally enabled, yes or no) can be queried.</li> <li>By <code>get_startstop_info</code> (<code>Bit #14</code>), the status “laser enabled” can be queried.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. In some circumstances, <code>set_laser_control</code> must be called before <code>enable_laser</code> (see above).
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">disable_laser</a> , <a href="#">set_laser_control</a> , <a href="#">get_startstop_info</a> , <a href="#">set_laser_mode</a>



<b>Ctrl Command</b>	<b>eth_assign_card</b>
<b>Function</b>	Enters the RTC6 Ethernet Board with index SearchNo from the search result list at the RTC6 board management index CardNo.
<b>Call</b>	Result = eth_assign_card( SearchNo, CardNo )
<b>Parameters</b>	<p>SearchNo      Index of the search result list. As an unsigned 32-bit value. Allowed value range: [1...(Result value &gt;0 of <b>eth_found_cards</b>].</p> <p>CardNo      Index of the RTC6 board management, at which the RTC6 Ethernet Board is to be entered. As an unsigned 32-bit value. Allowed value range: [(<b>rtc6_count_cards</b> + 1)...255] or 0.</p>
<b>Result</b>	<p>Error code. As a signed 32-bit value.</p> <ul style="list-style-type: none"> <li>-2      Error: the entry cannot be made. At this index, already an RTC6 Ethernet Board is entered.</li> <li>-1      Error: the entry cannot be made. At this index, already an RTC6 PCIe Board is entered.</li> <li>0      Error: the entry cannot be made. SearchNo is invalid (0, &gt; <b>eth_found_cards</b>). Or: CardNo is invalid (1 ≤ CardNo ≤ <b>rtc6_count_cards</b>, &gt; 255).</li> <li>n (= CardNo)      Success: the card number entry has been made.</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_assign_card</b> is not available as a multi-board command.</li> <li>• To create a search result list, <b>eth_search_cards</b> and <b>eth_search_cards_range</b> are used. See also <a href="#">Chapter 15.5.2 "About Searching RTC6 Ethernet Boards", page 855</a>.</li> <li>• No RTC6 Ethernet Board or RTC6 PCIe Board must already be entered at the specified index CardNo. An RTC6 Ethernet Board must be explicitly removed by <b>eth_remove_card</b> in advance. RTC6 PCIe Boards cannot be removed.</li> <li>• When successful, <b>eth_count_cards</b> is automatically increased by 1.</li> <li>• With CardNo = 0 the RTC6 Ethernet Board is entered at <b>rtc6_count_cards</b> + <b>eth_count_cards</b> + 1. Thus, a continuous card numbering without gaps can automatically be created (see also <b>eth_remove_card</b>). CardNo = 0 and freely chosen assignment of RTC6 Ethernet Boards to indexes should not be concurrently used.</li> <li>• With all errors, the <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b> is generated.</li> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
<b>Version info</b>	Available as of version DLL 606, OUT 606, RBF 611.
<b>References</b>	<a href="#">eth_assign_card_ip</a> , <a href="#">eth_count_cards</a> , <a href="#">eth_found_cards</a> , <a href="#">eth_max_card</a> , <a href="#">eth_search_cards</a> , <a href="#">eth_search_cards_range</a> , <a href="#">eth_remove_card</a> , <a href="#">rtc6_count_cards</a>



<b>Ctrl Command</b>	<b>eth_assign_card_ip</b>
<b>Function</b>	Enters an RTC6 Ethernet Board with the IP address <code>Ip</code> at the index <code>CardNo</code> in the RTC6 board management.
<b>Call</b>	<code>Result = eth_assign_card_ip( Ip, CardNo )</code>
<b>Parameters</b>	<code>Ip</code> IP address for the RTC6 Ethernet Board. As an unsigned 32-bit value. <code>CardNo</code> Index of the RTC6 board management, at which the RTC6 Ethernet Board is to be entered. As an unsigned 32-bit value. Allowed value range: [ <code>(rtc6_count_cards + 1)…255</code> ] or 0.
<b>Result</b>	Error code. As a signed 32-bit value. -2 Error: the entry cannot be made. At this index, already an RTC6 Ethernet Board is entered. -1 Error: the entry cannot be made. At this index, already an RTC6 PCIe Board is entered. 0 Error: the entry cannot be made. CardNo is invalid ( $1 \leq \text{CardNo} \leq \text{rtc6\_count\_cards}$ , $> 255$ ). <code>n (= CardNo)</code> Success: the entry has been made.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>eth_assign_card_ip</code> is not available as a multi-board command.</li> <li>• No RTC6 Ethernet Board or RTC6 PCIe Board must already be entered at the specified index <code>CardNo</code>. An RTC6 Ethernet Board must be explicitly removed by <code>eth_remove_card</code> in advance. RTC6 PCIe Boards cannot be removed.</li> <li>• When successful, <code>eth_count_cards</code> is automatically increased by 1.</li> <li>• With <code>CardNo = 0</code> the RTC6 Ethernet Board is entered at <code>rtc6_count_cards + eth_count_cards + 1</code>. Thus, a continuous card numbering without gaps can automatically be created (see also <code>eth_remove_card</code>). <code>CardNo = 0</code> and freely chosen assignment of RTC6 Ethernet Boards to indexes should not be concurrently used.</li> <li>• With all errors, the <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code> is generated.</li> <li>• The RTC6 Ethernet Board is entered in the RTC6 board management, even if there is an Ethernet error during its initialization. Before trying to initialize once again, this RTC6 Ethernet Board must be removed explicitly from the RTC6 board management.</li> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_assign_card</a> , <a href="#">eth_count_cards</a> , <a href="#">eth_remove_card</a>



<b>Ctrl Command</b>	<b>eth_boot_dcmd</b>
<b>Function</b>	<b>Standalone Functionality:</b> Defines the next following control command as a boot command for <b>Standalone Operation Mode</b> .
<b>Prerequisite</b>	Minimum requirement: RTC6 Software Package V1.7.0 and RTC6BIOSETH_26.
<b>Call</b>	<code>eth_boot_dcmd()</code>
<b>Parameters</b>	None.
<b>Result</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The definition refers to the subsequent control command, that is, list commands in between do not matter.</li> <li>The boot definition is omitted without replacement, if the defined control command is not permitted for booting, see <a href="#">Chapter 15.7.5 "Control Commands Allowed for Automatic Booting", page 863</a>.</li> <li><b>eth_boot_dcmd</b> is only allowed with RTC6 Ethernet Boards. Otherwise, a <b>get_last_error</b> return code <b>RTC6_TYPE_REJECTED</b> is generated.</li> <li>See <a href="#">Chapter 15.7 "Standalone Functionality", page 859</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 618, OUT 618, RBF 623.
References	<a href="#"><b>set_eth_boot_timeout</b></a>



<b>Ctrl Command</b>	<b>eth_check_connection</b>
<b>Function</b>	Checks whether the RTC6 Ethernet Board responds.
<b>Call</b>	CheckOK = eth_check_connection()
<b>Parameters</b>	None.
<b>Result</b>	<p>Response behavior. As an unsigned 32-bit value.</p> <p>0: Not OK. The RTC6 Ethernet Board does not have an Ethernet connection, it does not respond or it is not an RTC6 Ethernet Board.</p> <p>1: OK. The RTC6 Ethernet Board has an Ethernet connection and it responds.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>When CheckOK = 0, the possible error cause can be queried by <a href="#">get_last_error</a>, <a href="#">eth_get_last_error</a> and <a href="#">eth_get_error</a>.</li> <li>If the board is not an RTC6 Ethernet Board, then the <a href="#">get_last_error</a> return code <a href="#">RTC6_TYPE_REJECTED</a> is set and <a href="#">eth_check_connection</a> is not executed.</li> <li>See also <a href="#">Chapter 15.5.4 "Checking the Connection to the RTC6 Ethernet Board", page 857</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 608, OUT 608, RBF 611.
References	<a href="#">eth_get_error</a> , <a href="#">eth_get_last_error</a> , <a href="#">get_last_error</a>



<b>Ctrl Command</b>	<b><a href="#">eth_convert_ip_to_string</a></b>
<b>Function</b>	Converts an IP address in big-endian byte order to the usual dotted decimal notation (for example, "192.168.250.1").
<b>Call</b>	<code>eth_convert_ip_to_string( Ip, IpString )</code>
<b>Parameters</b>	<p>Ip                    To-be converted IP address in big-endian byte order. As an unsigned 32-bit value.</p> <p>IpString            Converted IP address. As a pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit or 64-bit value) to an array of four unsigned 32-bit values, where the converted IP address as a string in usual dotted decimal notation can be found (synonymously to a character array of length 16, low byte first, with concluding \0).</p>
<b>Result</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_convert_ip_to_string</b> is not available as a multi-board command.</li> <li>• For storage of the converted IP address, the user program must provide a memory area of <math>4 \times 4</math> bytes at the address specified by <code>IpString</code>.</li> <li>• Example: <code>Ip = 33204416, IpString[0] = "192.", IpString[1] = "168.", IpString[2] = "250.", IpString[3] = "1\0xx"</code>.</li> <li>• The use of <b>eth_convert_ip_to_string</b> is useful, for example, for <b><a href="#">eth_get_static_ip</a></b>.</li> <li>• See also <a href="#">Chapter 15.5.1 "Notes on Working with IP Addresses", page 855</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<b><a href="#">eth_convert_string_to_ip</a>, <a href="#">eth_get_static_ip</a></b>



<b>Ctrl Command</b>	<b>eth_convert_string_to_ip</b>
Function	Converts an IP address in usual dotted decimal notation (for example, "192.168.250.1") to the big-endian byte order.
Call	ResultIp = eth_convert_string_to_ip( &IpString )
Parameters	IpString      To-be converted IP address in usual dotted decimal notation. As a string (char array). As a pointer to a \0-terminated ANSI string. 16 bytes max.
Result	IP address. As an unsigned 32-bit value. 0:              IpString is faulty. >0:              Converted IP address in big-endian byte order.
Comments	<ul style="list-style-type: none"> <li>• <b>eth_convert_string_to_ip</b> is not available as a multi-board command.</li> <li>• Example: With parameter IpString = "192.168.250.1" the result value is IpString = 33204416. See also comment at <b>eth_search_cards</b>.</li> <li>• The use of <b>eth_convert_string_to_ip</b> is useful, for example, for <b>eth_search_cards</b>, <b>eth_search_cards_range</b> and <b>eth_set_static_ip</b> as well as with <b>eth_get_card_info</b> and <b>eth_get_card_info_search</b>.</li> <li>• See also Chapter 15.5.1 "Notes on Working with IP Addresses", page 855.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<b>eth_convert_ip_to_string</b> , <b>eth_get_card_info</b> , <b>eth_get_card_info_search</b> , <b>eth_search_cards</b> , <b>eth_search_cards_range</b> , <b>eth_set_static_ip</b>



<b>Ctrl Command</b>	<b>eth_count_cards</b>
<b>Function</b>	Returns the number of RTC6 Ethernet Boards entered in the RTC6 board management.
<b>Call</b>	Result = eth_count_cards()
<b>Parameters</b>	None.
<b>Result</b>	Number of RTC6 Ethernet Boards recorded in the RTC6 board management. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_count_cards</b> is not available as a multi-board command.</li> <li>• Unlike <b>rtc6_count_cards</b>, you cannot deduce in each case the highest index of all RTC6 boards from the result value (see <b>eth_assign_card</b> and <b>eth_remove_card</b> with CardNo = 0, see also <b>eth_max_card</b>).</li> <li>• It is only ensured that each RTC6 Ethernet Board number is greater than the highest RTC6 PCIe Board number, see <b>rtc6_count_cards</b>.</li> <li>• Certainly <b>eth_count_cards</b> can be greater than <b>eth_found_cards</b>. This is the case, if RTC6 Ethernet Boards have been entered explicitly by its IP addresses into the RTC6 board management, see <b>eth_assign_card_ip</b>.</li> <li>• After successful execution of <b>eth_assign_card</b>, <b>eth_count_cards</b> is automatically increased by 1.</li> <li>• After successful execution of <b>eth_remove_card</b>, <b>eth_count_cards</b> is automatically decreased by 1.</li> <li>• See also <b>Chapter 15.5.3 "About the RTC6 Board Management"</b>, page 856.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_assign_card</a> , <a href="#">eth_assign_card_ip</a> , <a href="#">eth_found_cards</a> , <a href="#">eth_max_card</a> , <a href="#">eth_remove_card</a> , <a href="#">rtc6_count_cards</a>



<b>Ctrl Command</b>	<b>eth_found_cards</b>
<b>Function</b>	Returns the number of RTC6 Ethernet Boards in the search result list.
<b>Call</b>	Result = eth_found_cards()
<b>Parameters</b>	None.
<b>Result</b>	<p>Number of RTC6 Ethernet Boards in the search result list. As an unsigned 32-bit value.</p> <p>0:            There are no RTC6 Ethernet Boards in the search result list.               Possibly the search has not been carried-out yet.</p> <p>n:            Number of RTC6 Ethernet Boards contained in the search result list.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_found_cards</b> is not available as a multi-board command.</li> <li>• <b>eth_found_cards</b> does not necessarily match the <ul style="list-style-type: none"> <li>– RTC6 Ethernet Board number in the RTC6 board management</li> <li>– total number of RTC6 Ethernet Boards and RTC6 PCIe Boards.</li> </ul> </li> <li>• To create the search result list, use either <b>eth_search_cards</b> or <b>eth_search_cards_range</b>. Their result value is the same as the one of <b>eth_found_cards</b>. See also <a href="#">Chapter 15.5.2 "About Searching RTC6 Ethernet Boards", page 855</a>.</li> <li>• Information about a particular RTC6 Ethernet Board in the search result list can be queried by <b>eth_get_card_info_search</b>.</li> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_get_card_info_search</a> , <a href="#">eth_search_cards</a> , <a href="#">eth_search_cards_range</a>



<b>Ctrl Command</b>	<b>eth_get_card_info</b>
<b>Function</b>	Returns information about a particular RTC6 Ethernet Board enlisted in the RTC6 board management and has been previously acquired once or is currently acquired.
<b>Call</b>	<code>eth_get_card_info( CardNo, Ptr )</code>
<b>Parameters</b>	<p>CardNo            Index of the RTC6 Ethernet Board in the RTC6 board management. As an unsigned 32-bit value.</p> <p>Ptr                Card information. As a pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit or 64-bit value) to an array of 16 unsigned 32-bit values.</p>
<b>Result</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_get_card_info</b> is not available as a multi-board command.</li> <li>• For storage of the card information, the user program must provide (at the address specified by <code>Ptr</code>) a memory area of <math>16 \times 4</math> bytes.</li> <li>• Card information: <ul style="list-style-type: none"> <li>[0] = Firmware version</li> <li>[1] = Serial number</li> <li>[2] = IP address (Big Endian format)</li> <li>[3] = MAC address (lower 4 bytes)</li> <li>[4] = MAC address (upper 2 bytes)</li> <li>[5] = Is acquired (1 or 0)</li> <li>[6] = IP address of the connected PC (Big Endian format)</li> <li>[7] = Force DHCP (1 or 0)</li> <li>[8] = Static IP address (Big Endian format)</li> <li>[9] = Static net mask (Big Endian format)</li> <li>[10] = Static gateway (Big Endian format)</li> <li>[11] = UDP port for board searches</li> <li>[12] = UDP port for exclusive connections</li> <li>[13] = TCP port for an exclusive connection</li> <li>[14] = Reserved</li> <li>[15] = Reserved</li> </ul> </li> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856.</a></li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_count_cards</a> , <a href="#">eth_get_card_info_search</a> , <a href="#">eth_max_card</a> , <a href="#">get_card_type</a>



<b>Ctrl Command</b>	<b>eth_get_card_info_search</b>
<b>Function</b>	Returns information about a particular RTC6 Ethernet Board in the search result list.
<b>Call</b>	<code>eth_get_card_info_search( SearchNo, Ptr )</code>
<b>Parameters</b>	<p>SearchNo      Index of the search result list. As an unsigned 32-bit value.</p> <p>Ptr            Card information. As a pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit or 64-bit value) to an array of 16 unsigned 32-bit values.</p>
<b>Result</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_get_card_info_search</b> is not available as a multi-board command.</li> <li>• To create the search result list, use either <b>eth_search_cards</b> or <b>eth_search_cards_range</b>. Their result value is the same as the one of <b>eth_found_cards</b>. See also <a href="#">Chapter 15.5.2 "About Searching RTC6 Ethernet Boards", page 855</a>.</li> <li>• For <code>SearchNo = 0</code> and <code>SearchNo &gt; eth_found_cards</code> a default information is returned. Furthermore, the <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b> is generated.</li> <li>• If an search has not yet been carried-out then automatically the following applies: <code>SearchNo &gt; eth_found_cards</code>.</li> <li>• For storage of the card information, the user program must provide (at the address specified by <code>Ptr</code>) a memory area of <math>16 \times 4</math> bytes.</li> <li>• Card information: Same as <b>eth_get_card_info</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_found_cards</a> , <a href="#">eth_get_card_info</a> , <a href="#">eth_search_cards</a> , <a href="#">eth_search_cards_range</a>



<b>Ctrl Command</b>	<b>eth_get_com_timeouts</b>	
<b>Function</b>	Returns timing information about a particular RTC6 Ethernet Board enlisted in the RTC6 board management.	
<b>Call</b>	eth_get_com_timeouts( &AcquireTimeout, &AcquireMaxRetries, &SendRecvTimeout, &SendRecvMaxRetries, &KeepAlive, &KeepInterval )	
<b>Result</b>	None.	
<b>Returned parameter values</b>	AcquireTimeout	Reserved. As a pointer to an unsigned 32-bit value.  See <a href="#">AcquireTimeout</a> .
	AcquireMaxRetries	 See <a href="#">AcquireTimeout</a> .
	SendRecvTimeout	 See <a href="#">AcquireTimeout</a> .
	SendRecvMaxRetries	 See <a href="#">AcquireTimeout</a> .
	KeepAlive	 See <a href="#">AcquireTimeout</a> .
	KeepInterval	 See <a href="#">AcquireTimeout</a> .  All values are each 0, if the board is not an RTC6 Ethernet Board. All time specifications in $\mu$ s.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856</a>.</li> <li>• To change the settings, <b>eth_set_com_timeouts</b> is used (usually only in the context of a problem clarification and only on request by SCANLAB).</li> <li>• The parameters AcquireTimeout, AcquireMaxRetries, SendRecvTimeout and SendRecvMaxRetries refer only to the <b>RTC6 DLL</b>. Therefore, they can be read out even without granted access right to an RTC6 Ethernet Board.</li> <li>• The returned parameters KeepAlive and KeepInterval have the value 0, if for the addressed RTC6 Ethernet Board <ul style="list-style-type: none"> <li>– there is no access right granted (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b> and <b>RTC6_ETH_ERROR</b>)</li> <li>– there is a granted access right, but no connection is established (<b>get_last_error</b> return code <b>RTC6_ETH_ERROR</b>)</li> </ul> </li> </ul>	
RTC4→RTC6	New command.	
RTC5→RTC6	New command.	
Version info	Available as of version DLL 606, OUT 606, RBF 611. Last change version DLL 615: access to solely <b>RTC6 DLL</b> -internal parameters.	
<b>References</b>	<a href="#">eth_set_com_timeouts</a>	



<b>Ctrl Command</b>	<b>eth_get_error</b>
Function	Returns an accumulated error code. It contains all errors which occurred with the most recently executed RTC6 Ethernet Board commands.
Call	<code>EthAccError = eth_get_error()</code>
Parameters	None.
Result	Same error code as <b>eth_get_last_error</b> . As an unsigned 32-bit value. If multiple errors occurred simultaneously, then multiple bits are set.
Comments	<ul style="list-style-type: none"> <li>For error handling, see <a href="#">Chapter 6.8 "Error Handling", page 120</a>.</li> <li><b>eth_get_error</b> and <b>n_eth_get_error</b> are available even without explicit access rights to a specific RTC6 Ethernet Board.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <b>eth_get_error</b>.</li> <li>If with an RTC6 Ethernet Board command call (internally) several errors occur successively, <b>eth_get_last_error</b> probably returns only the very last error (for example, only "<code>not_acquired</code>"). The actual error causes can be identified by <b>eth_get_error</b> because it returns the accumulated errors.</li> <li>The accumulated error is reset automatically with each acquiring of the RTC6 Ethernet Board.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 608, OUT 608, RBF 611.
References	<a href="#">acquire_rtc</a> , <a href="#">eth_get_last_error</a>



<b>Ctrl Command</b>	<b>eth_get_ip</b>
<b>Function</b>	Returns the pertaining IP address for a specified RTC6 board management index.
<b>Call</b>	IP = eth_get_ip( CardNo )
<b>Parameters</b>	CardNo      Index of the RTC6 Ethernet Board in the RTC6 board management. As an unsigned 32-bit value.
<b>Result</b>	IP address. In Big Endian byte order. The value is 0, if the board is not an RTC6 Ethernet Board.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_get_ip</b> is not available as a multi-board command.</li> <li>• To convert from Big Endian byte order to the usual dotted decimal notation, <b>eth_convert_ip_to_string</b> can be used. See also <a href="#">Chapter 15.5.1 "Notes on Working with IP Addresses", page 855</a>.</li> <li>• <b>eth_get_ip</b> allows a faster IP address query from the RTC6 board management than by <b>eth_get_card_info</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_convert_ip_to_string</a> , <a href="#">eth_get_card_info</a>

<b>Ctrl Command</b>	<b>eth_get_ip_search</b>
<b>Function</b>	Returns the pertaining IP address for a specified RTC6 Ethernet Board in the search result list.
<b>Call</b>	IP = eth_get_ip_search( SearchNo )
<b>Parameters</b>	SearchNo      Index in the search result list. As an unsigned 32-bit value.
<b>Result</b>	IP address. In Big Endian byte order.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_get_ip_search</b> is not available as a multi-board command.</li> <li>• To create the search result list, use either <b>eth_search_cards</b> or <b>eth_search_cards_range</b>. Their result value is the same as the one of <b>eth_found_cards</b>. See also <a href="#">Chapter 15.5.2 "About Searching RTC6 Ethernet Boards", page 855</a>.</li> <li>• The result value is 0 for SearchNo = 0 and SearchNo &gt; <b>eth_found_cards</b>. At the same time the <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b> is generated.</li> <li>• If an search has not yet been carried-out then automatically the following applies: SearchNo &gt; <b>eth_found_cards</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 608, OUT 608, RBF 611.
References	<a href="#">eth_get_ip</a>



<b>Ctrl Command</b>	<b>eth_get_last_error</b>																																																																																						
<b>Function</b>	Returns an error code which contains only the errors that have occurred during execution of the most recent RTC6 Ethernet Board command.																																																																																						
<b>Call</b>	EthLastError = eth_get_last_error()																																																																																						
<b>Parameters</b>	None.																																																																																						
<b>Result</b>	<p>Error code. As an unsigned 32-bit value. If several errors occurred at the same time, several bits are set.</p> <table> <thead> <tr> <th>no_error = 0. No error.</th> <th>Error</th> <th>Numeric value (<math>2^{[Bit]}</math>)</th> </tr> </thead> <tbody> <tr><td>wsasstartup_failed</td><td>Bit 0</td><td>1</td></tr> <tr><td>wrong_winsock_version</td><td>Bit 1</td><td>2</td></tr> <tr><td>Reserved.</td><td>Bit 2</td><td>4</td></tr> <tr><td>udp_create_socket_error</td><td>Bit 3</td><td>8</td></tr> <tr><td>udp_bind_socket_error</td><td>Bit 4</td><td>16</td></tr> <tr><td>udp_connect_socket_error</td><td>Bit 5</td><td>32</td></tr> <tr><td>udp_excl_add_use_error</td><td>Bit 6</td><td>64</td></tr> <tr><td>udp_bc_ena_error</td><td>Bit 7</td><td>128</td></tr> <tr><td>Reserved.</td><td>Bit 8</td><td>256</td></tr> <tr><td>tcp_create_socket_error</td><td>Bit 9</td><td>512</td></tr> <tr><td>tcp_excl_add_use_error</td><td>Bit 10</td><td>1024</td></tr> <tr><td>tcp_bind_socket_error</td><td>Bit 11</td><td>2048</td></tr> <tr><td>Reserved.</td><td>Bit 12</td><td>4096</td></tr> <tr><td>tcp_connect_socket_error</td><td>Bit 13</td><td>8192</td></tr> <tr><td>tcp_connect_sel_error</td><td>Bit 14</td><td>16384</td></tr> <tr><td>tcp_connect_fds_error</td><td>Bit 15</td><td>32768</td></tr> <tr><td>tcp_nodelay_error</td><td>Bit 16</td><td>65536</td></tr> <tr><td>create_thread_failed</td><td>Bit 17</td><td>131072</td></tr> <tr><td>udp_recv_error</td><td>Bit 18</td><td>262144</td></tr> <tr><td>udp_send_error</td><td>Bit 19</td><td>524288</td></tr> <tr><td>udp_recv_timeout</td><td>Bit 20</td><td>1048576</td></tr> <tr><td>already_acquired</td><td>Bit 21</td><td>2097152</td></tr> <tr><td>not_acquired</td><td>Bit 22</td><td>4194304</td></tr> <tr><td>access_denied</td><td>Bit 23</td><td>8388608</td></tr> <tr><td>send_tgm_timeout</td><td>Bit 24</td><td>16777216</td></tr> <tr><td>card_not_found</td><td>Bit 25</td><td>33554432</td></tr> <tr><td>core1_timeout</td><td>Bit 26</td><td>67108864</td></tr> </tbody> </table>			no_error = 0. No error.	Error	Numeric value ( $2^{[Bit]}$ )	wsasstartup_failed	Bit 0	1	wrong_winsock_version	Bit 1	2	Reserved.	Bit 2	4	udp_create_socket_error	Bit 3	8	udp_bind_socket_error	Bit 4	16	udp_connect_socket_error	Bit 5	32	udp_excl_add_use_error	Bit 6	64	udp_bc_ena_error	Bit 7	128	Reserved.	Bit 8	256	tcp_create_socket_error	Bit 9	512	tcp_excl_add_use_error	Bit 10	1024	tcp_bind_socket_error	Bit 11	2048	Reserved.	Bit 12	4096	tcp_connect_socket_error	Bit 13	8192	tcp_connect_sel_error	Bit 14	16384	tcp_connect_fds_error	Bit 15	32768	tcp_nodelay_error	Bit 16	65536	create_thread_failed	Bit 17	131072	udp_recv_error	Bit 18	262144	udp_send_error	Bit 19	524288	udp_recv_timeout	Bit 20	1048576	already_acquired	Bit 21	2097152	not_acquired	Bit 22	4194304	access_denied	Bit 23	8388608	send_tgm_timeout	Bit 24	16777216	card_not_found	Bit 25	33554432	core1_timeout	Bit 26	67108864
no_error = 0. No error.	Error	Numeric value ( $2^{[Bit]}$ )																																																																																					
wsasstartup_failed	Bit 0	1																																																																																					
wrong_winsock_version	Bit 1	2																																																																																					
Reserved.	Bit 2	4																																																																																					
udp_create_socket_error	Bit 3	8																																																																																					
udp_bind_socket_error	Bit 4	16																																																																																					
udp_connect_socket_error	Bit 5	32																																																																																					
udp_excl_add_use_error	Bit 6	64																																																																																					
udp_bc_ena_error	Bit 7	128																																																																																					
Reserved.	Bit 8	256																																																																																					
tcp_create_socket_error	Bit 9	512																																																																																					
tcp_excl_add_use_error	Bit 10	1024																																																																																					
tcp_bind_socket_error	Bit 11	2048																																																																																					
Reserved.	Bit 12	4096																																																																																					
tcp_connect_socket_error	Bit 13	8192																																																																																					
tcp_connect_sel_error	Bit 14	16384																																																																																					
tcp_connect_fds_error	Bit 15	32768																																																																																					
tcp_nodelay_error	Bit 16	65536																																																																																					
create_thread_failed	Bit 17	131072																																																																																					
udp_recv_error	Bit 18	262144																																																																																					
udp_send_error	Bit 19	524288																																																																																					
udp_recv_timeout	Bit 20	1048576																																																																																					
already_acquired	Bit 21	2097152																																																																																					
not_acquired	Bit 22	4194304																																																																																					
access_denied	Bit 23	8388608																																																																																					
send_tgm_timeout	Bit 24	16777216																																																																																					
card_not_found	Bit 25	33554432																																																																																					
core1_timeout	Bit 26	67108864																																																																																					



Ctrl Command	<a href="#">eth_get_last_error</a>
Comments	<ul style="list-style-type: none"> <li>For error handling, <a href="#">Chapter 6.8 "Error Handling", page 120</a>.</li> <li><b>eth_get_last_error</b> and <b>n_eth_get_last_error</b> are available even without explicit access rights to a specific RTC6 Ethernet Board.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (<a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <b>eth_get_last_error</b>.</li> <li>Each <b>eth_get_last_error</b> error also leads to a <b>get_last_error</b> error <code>RTC6_ETH_ERROR</code>.</li> <li>If with an RTC6 Ethernet Board command call (internally) several errors occur successively, <b>eth_get_last_error</b> probably returns only the very last error (for example, only "<code>not_acquired</code>"). The actual error causes can be identified by <b>eth_get_error</b> because it returns the accumulated errors.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_get_error</a> , <a href="#">get_error</a> , <a href="#">reset_error</a> , <a href="#">set_verify</a>



<b>Ctrl Command</b>	<b>eth_get_port_numbers</b>	
<b>Function</b>	Returns the UDP ports for board searches ( <b>UDPsearch</b> ) and exclusive connections ( <b>UDPexcl</b> ) as well as the TCP port of an RTC6 Ethernet Board.	
<b>Call</b>	Result = eth_get_port_numbers( &UDPsearch, &UDPexcl, &TCP )	
<b>Result</b>	Error code. As an unsigned 32-bit value. 0: Success: OK. 1: Error: Not an RTC6 Ethernet Board.	
<b>Returned parameter values</b>	UDPsearch	UDP port for board searches. As a pointer to an unsigned 32-bit value.
	UDPexcl	UDP port for exclusive connections. As a pointer to an unsigned 32-bit value.
	TCP	TCP port for exclusive connections. As a pointer to an unsigned 32-bit value.  All values are each 0, if the board is not an RTC6 Ethernet Board.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The values can also be queried by <a href="#">eth_get_card_info</a>.</li> </ul>	
RTC4→RTC6	New command.	
RTC5→RTC6	New command.	
Version info	Available as of version DLL 606, OUT 606, RBF 611.	
<b>References</b>	<a href="#">eth_get_card_info</a> , <a href="#">eth_set_port_numbers</a>	

<b>Ctrl Command</b>	<b>eth_get_serial_search</b>	
<b>Function</b>	Returns the pertaining serial number for a specified RTC6 Ethernet Board in the search result list.	
<b>Call</b>	Serialnumber = eth_get_serial_search( SearchNo )	
<b>Parameters</b>	SearchNo      Index in the search result list. As an unsigned 32-bit value.	
<b>Result</b>	Serial number. As an unsigned 32-bit value.	
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>eth_get_serial_search</b> is not available as a multi-board command.</li> <li>To create the search result list, use either <a href="#">eth_search_cards</a> or <a href="#">eth_search_cards_range</a>. Their result value is the same as the one of <a href="#">eth_found_cards</a>. See also <a href="#">Chapter 15.5.2 "About Searching RTC6 Ethernet Boards", page 855</a>.</li> <li>The result value is 0 for SearchNo = 0 and SearchNo &gt; <a href="#">eth_found_cards</a>. At the same time the <a href="#">get_last_error</a> return code <b>RTC6_PARAM_ERROR</b> is generated.</li> <li>If an search has not yet been carried-out then automatically the following applies: SearchNo &gt; <a href="#">eth_found_cards</a>.</li> </ul>	
RTC4→RTC6	New command.	
RTC5→RTC6	New command.	
Version info	Available as of version DLL 608, OUT 608, RBF 611.	
<b>References</b>	<a href="#">get_serial_number</a> , <a href="#">eth_get_card_info_search</a>	



<b>Ctrl Command</b>	<b>eth_get_static_ip</b>				
<b>Function</b>	Returns static IP address, subnet mask and gateway address which are saved on the RTC6 Ethernet Board.				
<b>Call</b>	Result = eth_get_static_ip( &Ip, &NetMask, &Gateway )				
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <p>0: Success: OK. 1: Error. Not an RTC6 Ethernet Board.</p>				
<b>Parameters</b>	None.				
<b>returned parameter value</b>	Ip	Static IP address in Big Endian byte order. As a pointer to an unsigned 32-bit value.			
	NetMask	Sub net mask in Big Endian byte order. As a pointer to an unsigned 32-bit value.			
	Gateway	Address of the gateway in Big Endian byte order. As a pointer to an unsigned 32-bit value.			
	All values are each 0, if the board is not an RTC6 Ethernet Board or a static IP address has not been programmed yet.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>To convert from Big Endian byte order to the usual dotted decimal notation, <a href="#">eth_convert_ip_to_string</a> can be used. See also <a href="#">Chapter 15.5.1 "Notes on Working with IP Addresses"</a>, page 855.</li> <li>The values can also be queried by <a href="#">eth_get_card_info</a>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	New command.				
Version info	Available as of version DLL 606, OUT 606, RBF 611.				
<b>References</b>	<a href="#">eth_convert_ip_to_string</a> , <a href="#">eth_get_card_info</a> , <a href="#">eth_set_static_ip</a>				



<b>Ctrl Command</b>	<b>eth_max_card</b>
Function	Returns the highest index in the RTC6 board management, where an RTC6 Ethernet Board is entered.
Call	Result = eth_max_card()
Parameters	None.
Result	Highest index of an RTC6 Ethernet Board in the RTC6 board management. As an unsigned 32-bit value. 0: No RTC6 Ethernet Board recorded in the RTC6 board management. n: Highest index in the RTC6 board management, where an RTC6 Ethernet Board is entered.
Comments	<ul style="list-style-type: none"> <li>• <b>eth_max_card</b> is not available as a multi-board command.</li> <li>• <b>eth_max_card</b> does not return the total number of RTC6 Ethernet Boards in the RTC6 board management. For this purpose, <b>eth_count_cards</b> is available.</li> <li>• With <b>eth_max_card</b>, all RTC6 Ethernet Boards can be removed from the RTC6 board management by a simple loop:  <pre>while ( eth_max_card() ) { release_rtc( eth_max_card() ); eth_remove_card( eth_max_card() ); }</pre> </li> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_count_cards</a> , <a href="#">eth_remove_card</a>



<b>Ctrl Command</b>	<b>eth_remove_card</b>
<b>Function</b>	Deletes the RTC6 Ethernet Board entry from the RTC6 board management at the specified index.
<b>Call</b>	Result = eth_remove_card( CardNo )
<b>Parameters</b>	CardNo      Index of the RTC6 Ethernet Board in the RTC6 board management. As an unsigned 32-bit value.
<b>Result</b>	Error code. As a signed 32-bit value.  -2      Error: the entry cannot be deleted. At this index an RTC6 Ethernet Board is entered which is still acquired yet.  -1      Error: the entry cannot be deleted. At this index an RTC6 PCIe Board is entered.  0      Error: the entry cannot be deleted. At this index "No card" is entered or CardNo is invalid (> 255).  n (= CardNo)      Success: the entry has been deleted.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_remove_card</b> is not available as a multi-board command.</li> <li>• With all errors, the <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b> is generated.</li> <li>• Entries for "No card"s and RTC6 PCIe Boards cannot be deleted.</li> <li>• Before deleting an RTC6 Ethernet Board entry, an acquired board must be explicitly released by <b>release_rtc</b>.</li> <li>• With CardNo = 0, the RTC6 Ethernet Board entry at index (<b>rtc6_count_cards</b> + <b>eth_count_cards</b>) is deleted (see also continuous card numbering without gaps using <b>eth_assign_card</b> or <b>eth_assign_card_ip</b> and CardNo = 0). CardNo = 0 and freely chosen assignment of RTC6 Ethernet Boards to indexes should not be concurrently used.</li> <li>• All RTC6 Ethernet Board entries can be deleted from the RTC6 board management by a simple loop (provided that all RTC6 Ethernet Boards are <i>not</i> acquired):  <pre>while ( eth_remove_card(0)&gt; 0 );</pre> </li> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_assign_card</a> , <a href="#">eth_assign_card_ip</a> , <a href="#">eth_count_cards</a> , <a href="#">release_rtc</a> , <a href="#">rtc6_count_cards</a>



<b>Ctrl Command</b>	<b>eth_search_cards</b>				
<b>Function</b>	Executes a board search in form of a broadcast and returns the number of RTC6 Ethernet Boards which have answered in the specified address range.				
<b>Call</b>	Result = eth_search_cards( Ip, Netmask )				
<b>Parameters</b>	<table> <tr> <td>Ip</td> <td>IP-address in Big Endian byte order. As an unsigned 32-bit value.</td> </tr> <tr> <td>Netmask</td> <td>Subnet mask in Big Endian byte order. As an unsigned 32-bit value.</td> </tr> </table>	Ip	IP-address in Big Endian byte order. As an unsigned 32-bit value.	Netmask	Subnet mask in Big Endian byte order. As an unsigned 32-bit value.
Ip	IP-address in Big Endian byte order. As an unsigned 32-bit value.				
Netmask	Subnet mask in Big Endian byte order. As an unsigned 32-bit value.				
<b>Result</b>	Number of RTC6 Ethernet Boards in the specified address range which have answered (within <b>TimeOut</b> ). As an unsigned 32-bit value.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_search_cards</b> is not available as a multi-board command.</li> <li>• To convert from the usual dotted decimal notation to Big Endian byte order, <b>eth_convert_string_to_ip</b> can be used. See also <b>Chapter 15.5.1 "Notes on Working with IP Addresses"</b>, page 855.</li> <li>• The timeout value can be set with <b>eth_set_search_cards_timeout</b>.</li> <li>• A broadcast only reaches reliably addresses within the specified network segment. For other use cases a board search by IP scan (<b>eth_search_cards_range</b>) can be carried out.</li> <li>• The number of found RTC6 Ethernet Boards can also be queried by <b>eth_found_cards</b> at a later time.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	New command.				
Version info	Available as of version DLL 606, OUT 606, RBF 611.				
References	<b>eth_convert_string_to_ip</b> , <b>eth_found_cards</b> , <b>eth_search_cards_range</b> , <b>eth_set_search_cards_timeout</b>				



<b>Ctrl Command</b>	<b>eth_search_cards_range</b>
<b>Function</b>	Executes a search within a precisely specified IP address range and returns the number of RTC6 Ethernet Boards which have answered.
<b>Call</b>	Result = eth_search_cards_range( StartIP, EndIp )
<b>Parameters</b>	<p>StartIP      Start IP-address in Big Endian byte order. As an unsigned 32-bit value.</p> <p>EndIp      End IP-address in Big Endian byte order. As an unsigned 32-bit value.</p>
<b>Result</b>	Number of RTC6 Ethernet Boards which have answered (within <b>TimeOut</b> ). As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>eth_search_cards_range</b> is not available as a multi-board command.</li> <li>• To convert from the usual dotted decimal notation to Big Endian byte order, <a href="#">eth_convert_string_to_ip</a> can be used. See also <a href="#">Chapter 15.5.1 "Notes on Working with IP Addresses", page 855</a>.</li> <li>• <b>eth_search_cards_range</b> executes the search by sending UDP packets to each IP address within the specified address range. It reliably covers the specified address range (compare to broadcast with <a href="#">eth_search_cards</a>).</li> <li>• The <b>TimeOut</b> value can be set with <a href="#">eth_set_search_cards_timeout</a>.</li> <li>• The number of found RTC6 Ethernet Boards can also be queried by <a href="#">eth_found_cards</a> at a later time.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_convert_string_to_ip</a> , <a href="#">eth_found_cards</a> , <a href="#">eth_search_cards</a> , <a href="#">eth_set_search_cards_timeout</a>



<b>Ctrl Command</b>	<b>eth_set_com_timeouts</b>												
<b>Function</b>	Sets timing information for a particular RTC6 Ethernet Board enlisted in the RTC6 board management and has been acquired.												
<b>Call</b>	<code>eth_set_com_timeouts( AcquireTimeout, AcquireMaxRetries, SendRecvTimeout, SendRecvMaxRetries, KeepAlive, KeepInterval )</code>												
<b>Parameters</b>	<table> <tr> <td>AcquireTimeout</td> <td>Reserved. As an unsigned 32-bit value. With 0 the parameter is not changed.</td> </tr> <tr> <td>AcquireMaxRetries</td> <td><b>See</b> AcquireTimeout.</td> </tr> <tr> <td>SendRecvTimeout</td> <td><b>See</b> AcquireTimeout.</td> </tr> <tr> <td>SendRecvMaxRetries</td> <td><b>See</b> AcquireTimeout.</td> </tr> <tr> <td>KeepAlive</td> <td><b>See</b> AcquireTimeout.</td> </tr> <tr> <td>KeepInterval</td> <td><b>See</b> AcquireTimeout.</td> </tr> </table>	AcquireTimeout	Reserved. As an unsigned 32-bit value. With 0 the parameter is not changed.	AcquireMaxRetries	<b>See</b> AcquireTimeout.	SendRecvTimeout	<b>See</b> AcquireTimeout.	SendRecvMaxRetries	<b>See</b> AcquireTimeout.	KeepAlive	<b>See</b> AcquireTimeout.	KeepInterval	<b>See</b> AcquireTimeout.
AcquireTimeout	Reserved. As an unsigned 32-bit value. With 0 the parameter is not changed.												
AcquireMaxRetries	<b>See</b> AcquireTimeout.												
SendRecvTimeout	<b>See</b> AcquireTimeout.												
SendRecvMaxRetries	<b>See</b> AcquireTimeout.												
KeepAlive	<b>See</b> AcquireTimeout.												
KeepInterval	<b>See</b> AcquireTimeout.												
<b>Result</b>	None.												
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management", page 856</a>.</li> <li>• To query the settings, <b>eth_get_com_timeouts</b> is used.</li> <li>• All time specifications in <math>\mu</math>s.</li> <li>• As a rule, it is not required to change these parameter values. In case of timing problems contact SCANLAB in order to clarify your special local characteristics.</li> <li>• The parameters AcquireTimeout, AcquireMaxRetries, SendRecvTimeout and SendRecvMaxRetries refer only to the <b>RTC6 DLL</b>. Therefore, they can be set even without granted access right to an RTC6 Ethernet Board.</li> <li>• The parameters KeepAlive and KeepInterval are not set, if for the addressed RTC6 Ethernet Board <ul style="list-style-type: none"> <li>– there is no access right granted (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b> and <b>RTC6_ETH_ERROR</b>)</li> <li>– there is a granted access right, but no connection is established (<b>get_last_error</b> return code <b>RTC6_ETH_ERROR</b>)</li> </ul> </li> </ul>												
RTC4→RTC6	New command.												
RTC5→RTC6	New command.												
Version info	Available as of version DLL 606, OUT 606, RBF 611. Last change version DLL 615: access to solely <b>RTC6 DLL</b> -internal parameters.												
References	<a href="#">eth_get_com_timeouts</a>												



<b>Ctrl Command</b>	<b>eth_set_port_numbers</b>
<b>Function</b>	Saves the UDP ports for board searches ( <code>UDPsearch</code> ) and exclusive connections ( <code>UDPexcl</code> ) as well as the TCP port into the <b>Flash memory</b> of the RTC6 Ethernet Board.
<b>Call</b>	Result = <code>eth_set_port_numbers( UDPsearch, UDPexcl, TCP )</code>
<b>Parameters</b>	<p>UDPsearch      UDP port for board searches. As an unsigned 32-bit value. Allowed value range: [0...65,535]. Out-of-range values are clipped.</p> <p>UDPexcl      UDP port for exclusive connections. As an unsigned 32-bit value. Allowed value range: [0...65,535]. Out-of-range values are clipped.</p> <p>TCP      TCP port for exclusive connections. As an unsigned 32-bit value. Allowed value range: [0...65,535]. Out-of-range values are clipped.</p>
<b>Result</b>	Error code. As an unsigned 32-bit value. 0:      Success: OK. 1:      Error: No access to the RTC6 Ethernet Board. <code>get_last_error</code> return code <code>RTC6_ACCESS_DENIED</code> . 2:      Error: No RTC6 Ethernet Board. <code>get_last_error</code> return code <code>RTC6_TYPE_REJECTED</code> . 3:      Error: Programming the flash memory is not possible. The RTC6 Ethernet Board is possibly busy. <code>get_last_error</code> return code <code>RTC6_BUSY</code> or <code>RTC6_FLASH_ERROR</code> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>Prerequisites for <code>eth_set_port_numbers</code>: <ul style="list-style-type: none"> <li>No list must currently be executed on the RTC6 Ethernet Board.</li> <li>The RTC6 Ethernet Board must be entered in the RTC6 board management.</li> <li>The RTC6 Ethernet Board must have been acquired.</li> </ul> </li> <li>If 0 is specified for a parameter, it is not overwritten.</li> <li>Saved parameters are not used until the RTC6 Ethernet Board has been restarted.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_get_port_numbers</a>



<b>Ctrl Command</b>	<b>eth_set_search_cards_timeout</b>
<b>Function</b>	Sets a timeout value.
<b>Call</b>	<code>eth_set_search_cards_timeout( TimeOut )</code>
<b>Parameters</b>	TimeOut      Timeout value in $\mu$ s. As an unsigned 32-bit value.
<b>Result</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"><li>• <code>eth_set_search_cards_timeout</code> is not available as a multi-board command (<b>n_</b>).</li><li>• The default timeout value is 5 ms.</li><li>• The timeout value is relevant for <code>eth_search_cards</code> and <code>eth_search_cards_range</code>.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_search_cards</a> , <a href="#">eth_search_cards_range</a>



<b>Ctrl Command</b>	<b>eth_set_static_ip</b>						
<b>Function</b>	Writes a static IP address, a subnet mask and also a gateway address onto the RTC6 Ethernet Board.						
<b>Call</b>	Result = eth_set_static_ip( Ip, NetMask, Gateway )						
<b>Parameters</b>	<table> <tr> <td>Ip</td><td>Static IP address in Big Endian byte order. As an unsigned 32-bit value.</td></tr> <tr> <td>NetMask</td><td>Subnet mask in Big Endian byte order. As an unsigned 32-bit value.</td></tr> <tr> <td>Gateway</td><td>Address of the gateway in Big Endian byte order. As an unsigned 32-bit value.</td></tr> </table>	Ip	Static IP address in Big Endian byte order. As an unsigned 32-bit value.	NetMask	Subnet mask in Big Endian byte order. As an unsigned 32-bit value.	Gateway	Address of the gateway in Big Endian byte order. As an unsigned 32-bit value.
Ip	Static IP address in Big Endian byte order. As an unsigned 32-bit value.						
NetMask	Subnet mask in Big Endian byte order. As an unsigned 32-bit value.						
Gateway	Address of the gateway in Big Endian byte order. As an unsigned 32-bit value.						
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>0: Success: OK.</li> <li>1: Error: No access to the RTC6 Ethernet Board. <a href="#">get_last_error</a> return code <code>RTC6_ACCESS_DENIED</code>.</li> <li>2: Error: No RTC6 Ethernet Board. <a href="#">get_last_error</a> return code <code>RTC6_TYPE_REJECTED</code>.</li> <li>3: Error: Programming the flash memory is not possible. The RTC6 Ethernet Board is possibly busy. <a href="#">get_last_error</a> return code <code>RTC6_BUSY</code> or <code>RTC6_FLASH_ERROR</code>.</li> </ul>						
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Prerequisites for <code>eth_set_static_ip</code>: <ul style="list-style-type: none"> <li>– No list must currently be executed on the RTC6 Ethernet Board.</li> <li>– The RTC6 Ethernet Board must be entered in the RTC6 board management.</li> <li>– The RTC6 Ethernet Board must have been acquired.</li> </ul> </li> <li>• If no gateway is to be used, then <code>Gateway</code> must be set to 0.</li> <li>• To convert from usual dotted decimal notation to Big Endian byte order, <a href="#">eth_convert_string_to_ip</a> can be used. See also <a href="#">Chapter 15.5.1 "Notes on Working with IP Addresses", page 855</a>.</li> </ul>						
RTC4→RTC6	New command.						
RTC5→RTC6	New command.						
Version info	Available as of version DLL 606, OUT 606, RBF 611.						
References	<a href="#">eth_convert_string_to_ip</a> , <a href="#">eth_get_static_ip</a>						



<b>Ctrl Command</b>	<b>execute_at_pointer</b>
<b>Function</b>	Starts list execution ("List 1" or "List 2") at the specified address in the RTC6 list memory.
<b>Call</b>	<code>execute_at_pointer( Pos )</code>
<b>Parameters</b>	<p>Pos      Absolute address of the first list command to be executed.              As an unsigned 32-bit value.              Allowed value range: [0...(2<sup>23</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>execute_at_pointer</code> essentially functions like <code>execute_list_pos</code> (see comments there). However, <code>execute_at_pointer</code> requires a start address specified as an <i>absolute</i> memory address, whereas <code>execute_list_pos</code> requires specification of the list number and a <i>relative</i> memory address.</li> <li>• For <code>Pos ≥ Mem1 + Mem2</code> (see <code>config_list</code>), <code>Pos</code> is set to 0.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">execute_list_pos</a> , <a href="#">get_out_pointer</a>



<b>Ctrl Command</b>	<b>execute_list</b>
Function	Starts execution at the beginning of the specified list ("List 1" or "List 2").
Call	<code>execute_list( ListNo )</code>
Parameters	ListNo      Number of the list to be executed. As an unsigned 32-bit value. Allowed values: [uneven: "List 1", even: "List 2"].
Comments	<ul style="list-style-type: none"> <li>• <code>execute_list</code> is synonymous with <code>execute_list_pos</code> with <code>Pos = 0</code>.</li> <li>• <code>execute_list_1</code> and <code>execute_list_2</code> (with no parameters) can be used alternatively.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_status</a> , <a href="#">execute_at_pointer</a> , <a href="#">execute_list_pos</a>

<b>Ctrl Command</b>	<b>execute_list_1</b>
Function	See <a href="#">execute_list</a> .
Call	<code>execute_list_1</code>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">execute_list</a>

<b>Ctrl Command</b>	<b>execute_list_2</b>
Function	See <a href="#">execute_list</a> .
Call	<code>execute_list_2</code>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">execute_list</a>



<b>Ctrl Command</b>	<b>execute_list_pos</b>
<b>Function</b>	Starts list execution ("List 1" or "List 2") at the specified position.
<b>Call</b>	<code>execute_list_pos( ListNo, Pos )</code>
<b>Parameters</b>	<p>ListNo      Number of the list to be executed. As an unsigned 32-bit value. Allowed values: [uneven: "List 1", even: "List 2"].</p> <p>Pos          Address of the first list command to be executed (offset relative to the start of the respective list). As an unsigned 32-bit value. Allowed value range: [0...(2<sup>23</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>execute_list_pos</b> is not executed (<b>get_error</b> return value = <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• If the <b>INTERNAL-BUSY</b> status of the board is set, <b>execute_list_pos</b> is only executed with a delay (after <b>INTERNAL-BUSY</b> has been reset again). No checks are performed to determine if a list is currently being loaded. During list processing, the other list (or even the same list) can be simultaneously reloaded (see also <a href="#">Chapter 6.4 "List Handling", page 95</a>).</li> <li>• Programs in the protected list memory area "List 3" cannot be directly executed by <b>execute_list_pos</b>. They can only be called from a list ("List 1" or "List 2") as a subroutine. Alternatively, the corresponding area can be assigned by <b>config_list</b> to "List 1" or "List 2".</li> <li>• Uneven <b>ListNo</b> values cause "List 1" to be executed; otherwise "List 2" is executed. This allows automatically generated continuous list changing by an incremented count.</li> <li>• If "List 2" has not been assigned memory (<b>Mem2</b> = 0, see <b>config_list</b>) then "List 1" is opened.</li> <li>• If <b>Pos</b> is specified as being larger than the memory area of the respective list (<b>Pos</b> &gt; <b>Mem1</b> or <b>Pos</b> &gt; <b>Mem2</b>), then <b>Pos</b> is set to 0.</li> <li>• The <b>BUSY</b> list status of the selected list is set and the <b>BUSY</b> list status of the other corresponding list is reset (see <b>read_status</b>). The <b>BUSY</b> list execution status (see <b>get_status</b>) is set.</li> <li>• Execution stops when a <b>set_end_of_list</b> command is encountered. If the end of a list area is reached without encountering a <b>set_end_of_list</b> command, then execution continues at the beginning of the same list area instead of with the next list. The output pointer remains in the active list area unless a <b>set_end_of_list</b> command was encountered and an <b>auto_change_pos</b> or <b>start_loop</b> command was previously called. For both lists to be treated as a single list, you must set the configuration appropriately: for example, <b>config_list( Mem1+Mem2, 0 )</b>.</li> </ul>



Ctrl Command	execute_list_pos
Comments (cont'd)	<ul style="list-style-type: none"> <li>If a home jump (defined with <code>home_position</code> or <code>home_position_xyz</code>) was executed by <code>set_end_of_list</code>, then <code>execute_list_pos</code> leads to a corresponding home return (the <code>INTERNAL-BUSY</code> status is set while the home return is executed).</li> <li>The <code>execute_list_pos</code> command triggers a flush of the buffered list input (see Chapter 6.4.1 "Loading Lists", page 95) even if the start was unsuccessful.</li> <li><code>execute_list_pos</code> also covers the specialized variants <code>execute_list_1</code>, <code>execute_list_2</code>, <code>execute_list</code> and <code>execute_at_pointer</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>execute_list</code> , <code>execute_at_pointer</code> , <code>set_start_list_pos</code>

Normal List Command	<b>fly_return</b>				
Function	Deactivates the previously set Processing-on-the-fly correction and subsequently executes a jump to the defined new output position.				
Restriction	If the <code>Option Processing-on-the-fly</code> is not enabled, <code>fly_return</code> only executes the jump to the specified new output position.				
Call	<code>fly_return( X, Y )</code>				
Parameters	<table> <tr> <td>X</td><td>Absolute x coordinate of the new output position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td>Y</td><td>Like X (analogously).</td></tr> </table>	X	Absolute x coordinate of the new output position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.	Y	Like X (analogously).
X	Absolute x coordinate of the new output position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.				
Y	Like X (analogously).				
Comments	<ul style="list-style-type: none"> <li>The jump to the new output position is executed as with <code>jump_abs</code> (see comments there).</li> <li>If Processing-on-the-fly-correction was activated within a subroutine called by an "AbsCall" and subsequently gets deactivated by <code>fly_return</code>, then the coordinate values specified with <code>fly_return</code> receive an offset (based on the current coordinates at the time of the call, see also Section ""AbsCalls"", page 104).</li> <li>See also Chapter 8.6 "Processing-on-the-fly", page 227.</li> </ul>				
RTC4→RTC6	Unchanged functionality. In addition: increased value range, and "AbsCall", see above. In <code>RTC4 Compatibility Mode</code> , the RTC6 multiplies the specified coordinate values by 16. The allowed value range decreases accordingly.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<code>set_fly_x</code> , <code>set_fly_y</code> , <code>set_fly_rot</code> , <code>set_fly_x_pos</code> , <code>set_fly_y_pos</code> , <code>set_fly_rot_pos</code>				



<b>Variable List Command</b>	<b>fly_return_1_axis</b>
<b>Function</b>	"Fly Extension" Command: Deactivates 1 <b>Axis</b> of a Processing-on-the-fly application and subsequently carries-out a jump to the specified new output position.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>fly_return_1_axis</b> only carries-out the jump to the specified new output position.
<b>Call</b>	<b>fly_return_1_axis( Axis, RetPos1 )</b>
<b>Parameters</b>	<p>Axis      <b>Axis from Table 3, page 245.</b>                  As an unsigned 32-bit value.</p> <p>RetPos1    Absolute axis coordinate of the new output position. In bits.                  As a signed 32-bit value.                  Allowed value range: [-524,288...+524,287].                  Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <b>Chapter 8.6 "Processing-on-the-fly"</b>, page 227 and <b>Section "Fly Extension" Commands</b>, page 245.</li> <li>• Axis needs to be a linear axis (1, 2 or 3). <b>fly_return_2_axes</b> is to be used to deactivate a <b>Rotary axis</b>.</li> <li>• <b>fly_return_1_axis</b> only deactivates 1 <b>Axis</b> of a Processing-on-the-fly application, if it has been activated by an <b>"Fly Extension" Command</b>.</li> <li>• If no Processing-on-the-fly functionality is active at the specified <b>Axis</b>, only the jump is carried out.</li> <li>• With an unallowed parameter value, <b>fly_return_1_axis</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>• See also comments on <b>fly_return</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<b>fly_return_2_axes, fly_return_3_axes</b>



<b>Variable List Command</b>	<b>fly_return_2_axes</b>
<b>Function</b>	"Fly Extension" Command: Deactivates 2 Axes of a Processing-on-the-fly application and subsequently carries-out a jump to the specified new output position.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>fly_return_2_axes</b> only carries-out the jump to the specified new output position.
<b>Call</b>	<code>fly_return_2_axes( Axis1, RetPos1, Axis2, RetPos2)</code>
<b>Parameters</b>	<p>Axis1      <a href="#">Axis from Table 3, page 245</a>. As an unsigned 32-bit value.</p> <p>RetPos1    Absolute axis coordinate of the new output position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</p> <p>Axis2      Like Axis1.</p> <p>RetPos2    Like RetPos1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Section ""Fly Extension" Commands", page 245</a>.</li> <li>Axis1 and Axis2 need to be 2 different linear axes (1, 2 or 3) or both the Rotary axis (4). In the latter case RetPos1/RetPos2 mean the return coordinates of the Axis 1/2.</li> <li><b>fly_return_2_axes</b> only deactivates Axes of a Processing-on-the-fly application, if these have been activated by an "Fly Extension" Command.</li> <li>If no Processing-on-the-fly functionality is active at one of the specified Axes, only the jump is carried out.</li> <li>With an unallowed parameter value, <b>fly_return_2_axes</b> is replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</li> <li>See also comments on <a href="#">fly_return</a>.</li> <li>The following command calls are executed in the same way: <ul style="list-style-type: none"> <li>- <code>fly_return_2_axes( 1, X, 2, Y ) = fly_return( X, Y )</code></li> <li>- <code>fly_return_2_axes( 4, X, 4, Y ) = fly_return( X, Y )</code></li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">fly_return_1_axis</a> , <a href="#">fly_return_3_axes</a>



<b>Variable List Command</b>	<b>fly_return_3_axes</b>
<b>Function</b>	"Fly Extension" Command: Deactivates all 3 Axes of a Processing-on-the-fly application and subsequently carries-out a jump to the specified new output position.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>fly_return_3_axes</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<b>fly_return_3_axes( RetPosX, RetPosY, RetPosZ )</b>
<b>Parameters</b>	<p>RetPosX      Absolute axis coordinate of the new output position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</p> <p>RetPosY      Like RetPosX.</p> <p>RetPosZ      Like RetPosX.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See Chapter 8.6 "Processing-on-the-fly", page 227 and Section ""Fly Extension" Commands", page 245.</li> <li>• <b>fly_return_3_axes</b> only deactivates Axes of a Processing-on-the-fly application, if these have been activated by an "Fly Extension" Command.</li> <li>• If no Processing-on-the-fly functionality is active at one of the specified Axes, only the jump is carried out.</li> <li>• With an unallowed parameter value, <b>fly_return_3_axes</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>• See also comments on <b>fly_return</b>.</li> <li>• The following command calls are executed in the same way: <ul style="list-style-type: none"> <li>– <b>fly_return_3_axes( X, Y, Z )</b> = <b>fly_return_z( X, Y, Z )</b></li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<b>fly_return_1_axis, fly_return_2_axes</b>



<b>Normal List Command</b>	<b>fly_return_z</b>
<b>Function</b>	Deactivates the previously set Processing-on-the-fly correction. Subsequently executes a jump to the defined position.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, <b>fly_return_z</b> only executes the jump to the defined new output position.
<b>Call</b>	<code>fly_return_z( X, Y, Z )</code>
<b>Parameters</b>	<p>X            Absolute coordinates of the new output position. In bits.            Y            As signed 32-bit values.            Z            Allowed value range: [-524,288...+524,287].            Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The jump to the new output position is executed as with <b>jump_abs_3d</b>, see comments there.</li> <li>If Processing-on-the-fly-correction was activated within a subroutine called by an "AbsCall" and subsequently gets deactivated by <b>fly_return_z</b>, then the coordinate values specified with <b>fly_return_z</b> receive an offset (based on the current coordinates at the time of the call, see also <b>Chapter 6.5.1 "Subroutines", Section ""AbsCalls""</b>, <a href="#">page 104</a>).</li> <li>See also <b>Chapter 8.6 "Processing-on-the-fly"</b>, <a href="#">page 227</a>.</li> </ul>
<b>RTC4→RTC6</b>	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified value for the x, y and z coordinates by 16. The allowed value range decreases accordingly.
<b>RTC5→RTC6</b>	Unchanged functionality.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the specified value for the z coordinates by 16. The allowed value range decreases accordingly.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_fly_z</a>



<b>Ctrl Command</b>	<b>free_RTC6_dll</b>
<b>Function</b>	Frees up all resources allocated by the <b>RTC6 DLL</b> for a user program.
<b>Call</b>	<code>free_RTC6_dll()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>RTC6 DLL</b>-allocated resources particularly include memory area in the <b>RTC6 DLL</b> allocated for the RTC6 board management (which is created by <b>init_RTC6_dll</b>). <b>free_RTC6_dll</b> deletes the RTC6 board management of the <b>RTC6 DLL</b>. Afterward, the user program has no access to boards (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>).</li> <li>• The calling of <b>free_RTC6_dll</b> is not absolutely necessary, because <b>RTC6 DLL</b>-assigned resources are automatically freed up when the user program terminates and the <b>RTC6 DLL</b> is thereby unloaded by Microsoft Windows. However, some user program development environments (in debug mode) issue "memory leaks detected" warnings even though the <b>RTC6 DLL</b> is unloaded. The calling of <b>free_RTC6_dll</b> eliminates this annoyance.</li> <li>• <b>free_RTC6_dll</b> is not available as a multi-board command.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command. The functionalities of <b>free_RTC6_dll</b> and the RTC5 command <b>free_RTC5_dll</b> are identical.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>init_RTC6_dll, release_RTC</b>



<b>Ctrl Command</b>	<b>get_auto_cal</b>
<b>Function</b>	Returns the type of ASC hardware integrated in the attached scan system previously detected by <b>auto_cal</b> .
<b>Call</b>	ASCtype = get_auto_cal( HeadNo )
<b>Parameters</b>	HeadNo      Number of the scan head connector. As an unsigned 32-bit value. Allowed values: = 1:      First scan head connector. = 2:      Second scan head connector.
<b>Result</b>	ASC hardware type. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the ASC hardware type was previously detected by <b>auto_cal</b>, then <b>get_auto_cal</b> returns the same value as <b>auto_cal</b>( <b>Command</b> = 4 ), see comments there.</li> <li>If the ASC hardware type was not previously detected by <b>auto_cal</b>, then <b>get_auto_cal</b> returns the value 255 (initialized value).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>auto_cal</b>

<b>Ctrl Command</b>	<b>get_bios_version</b>
<b>Function</b>	Returns the <b>BIOS</b> version number of the RTC6 Board.
<b>Call</b>	BiosVersion = get_bios_version()
<b>Result</b>	<b>BCD-coded BIOS</b> version number. As an unsigned 32-bit value. Example: BiosVersion = 33 = 0x21 means <b>BIOS</b> version 21.
<b>Parameters</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>get_bios_version</b> returns reliable results only as of <b>BIOS</b> version 21, otherwise 0.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<b>get_dll_version</b> , <b>get_hex_version</b> , <b>get_RTC_version</b>



<b>Ctrl Command</b>	<b>get_card_type</b>
Function	Returns the RTC6 board type.
Call	Result = get_card_type()
Result	Board type. As an unsigned 32-bit value. 0: "No card". 1: RTC6 PCIe Board. 2: RTC6 Ethernet Board.
Parameters	None.
Comments	• See also <a href="#">Chapter 15.5.3 "About the RTC6 Board Management"</a> , page 856.
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 606, OUT 606, RBF 611.
References	<a href="#">eth_assign_card</a> , <a href="#">eth_assign_card_ip</a>



<b>Ctrl Command</b>	<b>get_char_pointer</b>
<b>Function</b>	Returns the absolute start address of an indexed character.
<b>Call</b>	CharPointer = get_char_pointer( Char )
<b>Parameters</b>	Char      Index of the indexed character. As an unsigned 32-bit value. Allowed value range: [0...1023].
<b>Result</b>	Absolute start address. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>get_char_pointer</b> reads from the internal management table the start address of the indexed character with the specified index. Whether the read address resides in a protected or the unprotected list memory area depends on whether the character was loaded into the protected list memory area "List 3" or an unprotected subroutine was only subsequently referenced.</li> <li>• If <b>Index &gt; 1023</b> or if no character was referenced with the specified index, then <b>get_char_pointer</b> returns the value "-1" (for example, <math>2^{32}-1</math>).</li> <li>• <b>get_char_pointer</b> is useful for checking if a character has already been defined or for calling an indexed character by an absolute memory address as if it were a non-indexed subroutine, for example, for conditional execution with <b>list_call_cond</b>. Be aware, though, that a subsequent <b>save_disk/load_disk</b> might alter the absolute memory address. And you should ensure that <b>get_char_pointer</b> does not return "-1"; otherwise <b>list_call_cond</b> is ignored.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_sub_pointer, get_text_table_pointer</b>



<b>Ctrl Command</b>	<b>get_config_list</b>
<b>Function</b>	Passes the parameters of the current list memory configuration (Mem1, Mem2) to the RTC6 board management of the <b>RTC6 DLL</b> and initializes it as if <b>config_list</b> would have been called.
<b>Call</b>	<code>get_config_list()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The <b>get_config_list</b> command is useful when a board changes "ownership" and the new RTC6 board management is not aware of the memory configuration (at the start of each user program, the board and board management each independently initialize <code>Mem1 = 4,194,304</code> and <code>Mem2 = 4,194,304</code>; the board by <b>load_program_file</b> and board management when starting the corresponding user program). See also <a href="#">Chapter 6.7.1 "Notes on Board Acquisition by a User Program", page 118</a>.</li> <li><b>get_config_list</b> does not return a value to the user program. The user program can, however, read the list-memory configuration data after <code>load_list( ListNo, 0 )</code> or <code>set_start_list_pos( ListNo, 0 )</code> by using <b>get_list_space</b>.</li> <li><b>get_config_list</b> is executed regardless of the <b>BUSY</b> status of the board.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">config_list</a>

<b>Ctrl Command</b>	<b>get_counts</b>
<b>Function</b>	Reads the current number of successful external starts.
<b>Call</b>	<code>Counts = get_counts()</code>
<b>Result</b>	Number of successful external starts. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The number is read from an internal counter, which is incremented each time a list is started by an external start signal. This counter can be reset to 0 by <b>set_control_mode</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_max_counts</a> , <a href="#">set_control_mode</a> , <a href="#">get_startstop_info</a>



<b>Ctrl Command</b>	<b>get_dll_version</b>
<b>Function</b>	Returns the version number of the <b>RTC6 DLL</b> .
<b>Call</b>	<code>DLLVersion = get_dll_version()</code>
<b>Result</b>	Version number. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The <b>RTC6 DLL</b> version numbers are in the range 600...699.</li> <li><b>get_dll_version</b> is available even without explicit access rights to a specific RTC6 board.</li> <li><b>get_dll_version</b> is not available as a multi-board command.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code>, see <a href="#">Chapter 6.8 "Error Handling", page 120</a>, are neither generated nor altered by <b>get_dll_version</b>.</li> </ul>
<b>RTC4→RTC6</b>	Unchanged functionality.
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">get_hex_version</a> , <a href="#">get_RTC_version</a>

<b>Ctrl Command</b>	<b>get_encoder</b>
<b>Function</b>	Returns the current counts of the two internal encoder counters.
<b>Call</b>	<code>get_encoder( &amp;Encoder0, &amp;Encoder1 )</code>
<b>Returned parameter values</b>	<p>Encoder0    Current count of encoder counter "Encoder0". As a pointer to a signed 32-bit value.</p> <p>Encoder1    Current count of encoder counter "Encoder1". As a pointer to a signed 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>get_encoder</b>, see <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 284</a>.</li> <li>If incremental encoders are used to detect the motion of the parts to be processed, encoder counter "Encoder0" is triggered by the signals at encoder input ENCODER X and encoder counter "Encoder1" by the signals at encoder input ENCODER Y.</li> <li>In contrast, if an encoder simulation has been started by <b>simulate_encoder</b>, both encoder counters are triggered by an internal periodic 1 MHz clock signal.</li> </ul>
<b>RTC4→RTC6</b>	Unchanged functionality. In addition: increased value range.
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">store_encoder</a> , <a href="#">read_encoder</a> , <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">set_fly_rot</a> , <a href="#">wait_for_encoder</a>



<b>Ctrl Command</b>	<b>get_error</b>																														
<b>Function</b>	Returns the cumulative error code. It corresponds to a list of error types occurring since the last reset or error reset.																														
<b>Call</b>	AccError = get_error()																														
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.  If multiple errors occurred, then multiple bits are set.  For the specific errors the error constants should be predefined as mentioned below.</p> <table> <thead> <tr> <th>Bit</th> <th>Error type</th> <th>Error constant</th> <th>=</th> </tr> </thead> <tbody> <tr> <td>-</td> <td>No error.</td> <td>RTC6_NO_ERROR</td> <td>0</td> </tr> <tr> <td>Bit #0 (LSB)</td> <td>= 1: No RTC6 PCIe Board found. This error can only occur with <a href="#">init_RTC6_dll</a>.</td> <td>RTC6_NO_PCIE_CARD_FOUND</td> <td>1</td> </tr> <tr> <td>Bit #1</td> <td>= 1: Access denied. This error can occur by <a href="#">init_RTC6_dll</a>, <a href="#">select_RTC</a>, <a href="#">acquire_RTC</a> or any multi-board command.</td> <td>RTC6_ACCESS_DENIED</td> <td>2</td> </tr> <tr> <td>Bit #2</td> <td>= 1: Command not forwarded. This error implies an internal, RTC6 board driver error or PCI error, for example, caused by a hardware defect or an incorrect connection.</td> <td>RTC6_SEND_ERROR</td> <td>4</td> </tr> <tr> <td>Bit #3</td> <td>= 1: No response from board. It is likely that no program has been loaded onto the RTC6. This error can especially occur in connection with control commands that expect a response, for example, <a href="#">get_hex_version</a>.</td> <td>RTC6_TIMEOUT</td> <td>8</td> </tr> <tr> <td>Bit #4</td> <td>= 1: Invalid parameter. This error can occur through all commands for which invalid parameters are not automatically corrected to valid values, for example, parameters with limited choices such as <a href="#">get_head_para</a>. If this error occurs for a list command, it is replaced with <a href="#">list_nop</a>. If this error occurs for a control command, it is not executed.</td> <td>RTC6_PARAM_ERROR</td> <td>16</td> </tr> </tbody> </table>			Bit	Error type	Error constant	=	-	No error.	RTC6_NO_ERROR	0	Bit #0 (LSB)	= 1: No RTC6 PCIe Board found. This error can only occur with <a href="#">init_RTC6_dll</a> .	RTC6_NO_PCIE_CARD_FOUND	1	Bit #1	= 1: Access denied. This error can occur by <a href="#">init_RTC6_dll</a> , <a href="#">select_RTC</a> , <a href="#">acquire_RTC</a> or any multi-board command.	RTC6_ACCESS_DENIED	2	Bit #2	= 1: Command not forwarded. This error implies an internal, RTC6 board driver error or PCI error, for example, caused by a hardware defect or an incorrect connection.	RTC6_SEND_ERROR	4	Bit #3	= 1: No response from board. It is likely that no program has been loaded onto the RTC6. This error can especially occur in connection with control commands that expect a response, for example, <a href="#">get_hex_version</a> .	RTC6_TIMEOUT	8	Bit #4	= 1: Invalid parameter. This error can occur through all commands for which invalid parameters are not automatically corrected to valid values, for example, parameters with limited choices such as <a href="#">get_head_para</a> . If this error occurs for a list command, it is replaced with <a href="#">list_nop</a> . If this error occurs for a control command, it is not executed.	RTC6_PARAM_ERROR	16
Bit	Error type	Error constant	=																												
-	No error.	RTC6_NO_ERROR	0																												
Bit #0 (LSB)	= 1: No RTC6 PCIe Board found. This error can only occur with <a href="#">init_RTC6_dll</a> .	RTC6_NO_PCIE_CARD_FOUND	1																												
Bit #1	= 1: Access denied. This error can occur by <a href="#">init_RTC6_dll</a> , <a href="#">select_RTC</a> , <a href="#">acquire_RTC</a> or any multi-board command.	RTC6_ACCESS_DENIED	2																												
Bit #2	= 1: Command not forwarded. This error implies an internal, RTC6 board driver error or PCI error, for example, caused by a hardware defect or an incorrect connection.	RTC6_SEND_ERROR	4																												
Bit #3	= 1: No response from board. It is likely that no program has been loaded onto the RTC6. This error can especially occur in connection with control commands that expect a response, for example, <a href="#">get_hex_version</a> .	RTC6_TIMEOUT	8																												
Bit #4	= 1: Invalid parameter. This error can occur through all commands for which invalid parameters are not automatically corrected to valid values, for example, parameters with limited choices such as <a href="#">get_head_para</a> . If this error occurs for a list command, it is replaced with <a href="#">list_nop</a> . If this error occurs for a control command, it is not executed.	RTC6_PARAM_ERROR	16																												



Ctrl Command	get_error			
Result (cont'd)	<p>Bit #5 = 1: List processing is (not) active. Examples: <code>for execute_list</code>, if a list is currently being processed; <code>for stop_execution</code>, if no list is currently being processed; <code>for restart_list</code>, if <code>pause_list</code> was not previously called.</p> <p>Bit #6 = 1: List command rejected, invalid input pointer. For example, for any list command directly after <code>load_char + list_return</code>: the list command is then not loaded.</p> <p>Bit #7 = 1: List command was converted to a <code>list_nop</code>. For example, <code>set_end_of_list</code> in a protected subroutine.</p> <p>Bit #8 = 1: Version error: <code>RTC6DLL.dll/RTC6DLLx64.dll</code> version, <code>RTC6RBF.rbf</code> version and <code>RTC6OUT.out/RTC6ETH.out</code> version are not compatible with each other. See also <code>load_program_file</code>.</p> <p>Bit #9 = 1: Verify error: The download verification has detected an incorrect download. See also <a href="#">Chapter 6.8.1 "Download Verification", page 121</a>.</p> <p>Bit #10 = 1: For example, an <code>[*]eth[*]</code> command has been sent to an RTC6 PCIe Board.</p> <p>Bit #11 = 1: A <code>RTC6 DLL</code>-internal Windows memory request failed.</p> <p>Bit #12 = 1: Download error. The values have possibly not been saved. May occur with <code>auto_cal</code> and <code>write_hi_pos</code>.</p>	RTC6_BUSY	= 32	
		RTC6_REJECTED	= 64	
		RTC6_IGNORED	= 128	
		RTC6_VERSION_MISMATCH	= 256	
		RTC6_VERIFY_ERROR	= 512	
		RTC6_TYPE_REJECTED	= 1024	
		RTC6_OUT_OF_MEMORY	= 2048	
		RTC6_FLASH_ERROR	= 4096	



Ctrl Command	get_error			
Result (cont'd)	Bit #13 = 1: General Ethernet error. May occur, for example, when trying to switch by <b>select_rtc</b> or <b>acquire_rtc</b> from RTC6 PCIe Board to RTC6 Ethernet Board. Further information is provided by <b>n_get_error</b> on the addressed RTC6 Ethernet Board.		RTC6_ETH_ERROR	= 8192
	Bit #14 Reserved.		-	
	Bit #15 = 1: Unsupported Windows version. May only occur during <b>init_rtc6_dll</b> .		-	= 32768
	Bit #16 = 1: Error reading PCI configuration register. May only occur during <b>init_rtc6_dll</b> .	RTC6_CONFIG_ERROR		= 65536
	Bit #17 Reserved.		-	
	...			
	Bit #31			
Comments	<ul style="list-style-type: none"> <li>For error handling see <a href="#">Chapter 6.8 "Error Handling", page 120</a>.</li> <li><b>get_error</b> and <b>n_get_error</b> are available even without explicit access rights to a specific RTC6 board.</li> <li>The board-specific error variables <b>LastError</b> and <b>AccError</b> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <b>get_error</b>.</li> <li>Bit #0 = 1 (error constant: <b>RTC6_NO_PCIE_CARD_FOUND</b> = 1) means that no RTC6 PCIe Board has been found (<b>init_rtc6_dll</b> does not search for RTC6 Ethernet Boards).</li> </ul>			
Example (C/C++)	<p>Creates an array for specifying which board has no existing access rights and resets the cumulative error code.</p> <pre>UINT NoAccess[MaxCount+1]; // MaxCount is a user-defined constant UINT Error = <b>init_rtc6_dll</b>(); // Searches for all installed RTC6 boards if (Error &amp; RTC6_ACCESS_DENIED) { // at least one board is inaccessible     UINT Count = <b>rtc6_count_cards</b>(); // number of boards found     for (UINT Num = 1; Num &lt;= Count; Num++) {         NoAccess[Num] = <b>n_get_last_error</b>(Num) &amp; RTC6_ACCESS_DENIED;         <b>n_reset_error</b>(Num, RTC6_ACCESS_DENIED);     } }</pre>			
RTC4→RTC6	New command.			
RTC5→RTC6	Unchanged functionality.			
Version info	Available as of version DLL 600, OUT 600, RBF 600.			
References	<a href="#"><b>get_last_error</b></a> , <a href="#"><b>reset_error</b></a> , <a href="#"><b>set_verify</b></a>			



<b>Ctrl Command</b>	<b>get_fly_2d_offset</b>
<b>Function</b>	Returns the current reference values (offset values) for 2D encoder compensation.
<b>Call</b>	<code>get_fly_2d_offset( &amp;OffsetX, &amp;OffsetY )</code>
<b>Returned parameter values</b>	<p>OffsetX    x reference value. As a pointer to a signed 32-bit value.</p> <p>OffsetY    y reference value. As a pointer to a signed 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For 2D encoder compensation, see <a href="#">Section "2D Encoder Compensation for xy Positioning Stages", page 234</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">init_fly_2d</a> , <a href="#">set_fly_2d</a>

<b>Ctrl Command</b>	<b>get_free_variable</b>
<b>Function</b>	Returns the current value of a free variable.
<b>Call</b>	<code>VariableValue = get_free_variable( No )</code>
<b>Parameters</b>	<p>No    Number of the free variable to be queried. As an unsigned 32-bit value. Allowed value range: [0...7]. Only the two least significant bits are evaluated.</p>
<b>Result</b>	The value currently stored in the free variable No. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 6.9.1 "Free Variables", page 124</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_free_variable</a> , <a href="#">set_free_variable_list</a>



<b>Ctrl Command</b>	<b>get_galvo_controls</b>
<b>Function</b>	Returns the corresponding control values for given input values.
<b>Restriction</b>	<b>get_galvo_controls</b> can only be executed, if no list is currently being processed ( <b>get_last_error</b> return code <b>RTC6_BUSY</b> ).
<b>Call</b>	<code>get_galvo_controls( InPtr, OutPtr )</code>
<b>Parameters</b>	<p>InPtr      Pointer (data type <b>ULONG_PTR</b> in C and C++, an unsigned 32-bit or 64-bit value) to an array of five unsigned 32-bit values, where the to-be-outputted settings are specified: X, Y, Z, Defocus, Zoom. Out-of-range values are clipped to the boundary values.</p> <p>OutPtr      Pointer (data type <b>ULONG_PTR</b> in C and C++, an unsigned 32-bit or 64-bit value) to an array of four unsigned 32-bit values, where the corresponding control values are to be stored: XA, YA, XB, YB. Value range in each case [-524,288... +524,287].</p>
<b>Returned parameter values</b>	XA, YA, XB, YB denote the control values for the xy axes of scan heads A and B.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>get_galvo_controls</b> carries-out a virtual jump to the specified coordinates. Though the galvanometer scanners are <i>not</i> moved.</li> <li>• All other settings which are not specified within <b>get_galvo_controls</b> (for example, matrix, offset, angle, etc.; also stretch table) are considered as they are set at the moment. Make sure to apply these by <code>at_once &gt; 0</code> before calling <b>get_galvo_controls!</b>. The settings are not used, if they just only have been saved by <code>at_once = 0</code>.</li> <li>• Control values are calculated only for channels where a correction table has been previously assigned by <b>select_cor_table</b>, see the following examples.</li> </ul> <p><code>select_cor_table(0,0): XA = YA = XB = YB = 0</code></p> <p>2D correction files: inputs Z, Defocus, Zoom are ignored</p> <p><code>select_cor_table(1,0): XA, YA calculated, XB, YB = 0</code></p> <p><code>select_cor_table(0,1): XA, YA = 0, XB, YB calculated</code></p> <p><code>select_cor_table(1,1): XA, YA, XB, YB calculated</code></p> <p>(Standard-)3D correction file: input Zoom is ignored</p> <p><code>select_cor_table(1,0): XA, YA calculated, XB = YB = Zout calculated</code></p> <p><code>select_cor_table(0,1): XA= YA = Zout calculated, XB, YB calculated</code></p> <p><code>select_cor_table(1,1): same as select_cor_table(0,0)</code></p> <p>3D zoom correction file (only for intelliWELD II with zoom axis):</p> <p><code>select_cor_table(1,0): XA, YA calculated, XB = Zout, YB = ZoomOut calculated</code></p> <p><code>select_cor_table(0,1): XA= Zout, YA = ZoomOut calculated, XB, YB calculated</code></p> <ul style="list-style-type: none"> <li>• The return values are 0, if a <b>get_last_error</b> return code <b>RTC6_BUSY</b> was generated.</li> </ul>



Ctrl Command	get_galvo_controls
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for x, y, z and Defocus by 16. The allowed value range decreases accordingly. Even in <b>RTC4 Compatibility Mode</b> , all returned values are in the RTC6 20-bit range.
RTC5→RTC6	Unchanged functionality.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the specified values for z and Defocus by 16. The allowed value range decreases accordingly. Even in <b>RTC5 Compatibility Mode</b> , all returned values are in the RTC6 20-bit range.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">select_cor_table</a>



<b>Ctrl Command</b>	<b>get_head_para</b>
<b>Function</b>	Returns the value of the requested parameter in the correction table assigned to the specified scan head.
<b>Call</b>	HeadPara = get_head_para( HeadNo, ParaNo )
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value. Allowed values: = 1:      First scan head connector. = 2:      Second scan head connector.</p> <p>ParaNo      Number of the parameter. As an unsigned 32-bit value. Allowed values: 0...15. Mapping: see <a href="#">Section "ct5 Correction File Header", page 167</a>.</p>
<b>Result</b>	Parameter value, see <a href="#">Section "ct5 Correction File Header", page 167</a> . As a 64-bit IEEE floating point value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The parameter values can be read out by <a href="#">get_table_para</a> from a currently loaded correction table and by <a href="#">get_head_para</a> from an assigned correction table and thus directly incorporated into a user program, see <a href="#">Section "ct5 Correction File Header", page 167</a>.</li> <li>If the parameters HeadNo and ParaNo are out of range, then the return value is 0 (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>). The return value is also 0 (no <a href="#">get_last_error</a> return code) if no correction table has been assigned to the specified head (for example, for HeadNo = 2 if the Option "Second Scan Head Control" has not been enabled) and no 3D correction table has been assigned to the other head.</li> <li>If a 3D correction table has been assigned to a head, then this 3D correction table's parameter is returned regardless of HeadNo (two 3D correction tables cannot be simultaneously assigned). HeadNo must nevertheless be 1 or 2 (see preceding comment).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_table_para</a>



<b>Ctrl Command</b>	<b>get_head_status</b>																																													
<b>Function</b>	Returns the XY2-100 status word from the specified scan head connector.																																													
<b>Call</b>	<code>get_head_status( Head )</code>																																													
<b>Parameters</b>	<p>Head = 1: Returns the status of the first scan head connector (Byte #1 = Byte #0).</p> <p>= 2: Returns the status of the second scan head connector (Byte #1 = Byte #0).</p> <p>Else: Returns the status of the first scan head connector (Byte #0) and of the second scan head connector (Byte #1).</p>																																													
<b>Result</b>	<p>XY2-100 status word. As an unsigned 32-bit value.</p> <table> <tbody> <tr> <td>Byte #0</td> <td>Bit #0</td> <td>1.</td> </tr> <tr> <td>(LSB)</td> <td>(LSB)</td> <td></td> </tr> <tr> <td></td> <td>Bit #1</td> <td>0.</td> </tr> <tr> <td></td> <td>Bit #2</td> <td>1 (reserved).</td> </tr> <tr> <td></td> <td>Bit #3</td> <td>Position Acknowledge of x axis, 1 = OK.</td> </tr> <tr> <td></td> <td>Bit #4</td> <td>Position Acknowledge of y axis, 1 = OK.</td> </tr> <tr> <td></td> <td>Bit #5</td> <td>1 (reserved).</td> </tr> <tr> <td></td> <td>Bit #6</td> <td>Temperature Status, 1 = OK.</td> </tr> <tr> <td></td> <td>Bit #7</td> <td>Power Status, 1 = OK.</td> </tr> <tr> <td>Byte #1</td> <td>Bit #8</td> <td>Bit assignments as with Byte #0.</td> </tr> <tr> <td></td> <td>...</td> <td></td> </tr> <tr> <td></td> <td>Bit #15</td> <td></td> </tr> <tr> <td>Byte #2</td> <td>Bit #16</td> <td>0.</td> </tr> <tr> <td>...</td> <td>...</td> <td></td> </tr> <tr> <td>Byte #3</td> <td>Bit #31</td> <td></td> </tr> </tbody> </table>	Byte #0	Bit #0	1.	(LSB)	(LSB)			Bit #1	0.		Bit #2	1 (reserved).		Bit #3	Position Acknowledge of x axis, 1 = OK.		Bit #4	Position Acknowledge of y axis, 1 = OK.		Bit #5	1 (reserved).		Bit #6	Temperature Status, 1 = OK.		Bit #7	Power Status, 1 = OK.	Byte #1	Bit #8	Bit assignments as with Byte #0.		...			Bit #15		Byte #2	Bit #16	0.	...	...		Byte #3	Bit #31	
Byte #0	Bit #0	1.																																												
(LSB)	(LSB)																																													
	Bit #1	0.																																												
	Bit #2	1 (reserved).																																												
	Bit #3	Position Acknowledge of x axis, 1 = OK.																																												
	Bit #4	Position Acknowledge of y axis, 1 = OK.																																												
	Bit #5	1 (reserved).																																												
	Bit #6	Temperature Status, 1 = OK.																																												
	Bit #7	Power Status, 1 = OK.																																												
Byte #1	Bit #8	Bit assignments as with Byte #0.																																												
	...																																													
	Bit #15																																													
Byte #2	Bit #16	0.																																												
...	...																																													
Byte #3	Bit #31																																													
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>get_head_status</b> is even available with the SL2-100 protocol, if the data type is not set to the 20-bit status word itself, see <a href="#">Section "Status Information Returned from the Scan System", page 172</a>. The status bits are returned by <b>get_head_status</b> in bits #2-7 and/or bits #10-15. Independently of the scan system's current state, bits #0 and 8 are returned by <b>get_head_status</b> as 1, while bits #1 and 9 are returned as 0.</li> <li>• If no scan system is currently connected or is not switched on, then the status of the most recently connected system is returned. <b>get_startstop_info</b> (Bit #17 and/or 25) can be used for distinguishing. If a scan system is still not connected after a reset (power-on or <b>load_program_file</b>) then <b>get_head_status</b> returns the value 0.</li> <li>• If a connected scan system is not switched on, then <b>get_head_status</b> returns a 0xFD byte ("everything OK"). The PowerOK signal is an electronically generated signal. Therefore, it cannot be used to check whether a connected scan head is switched on at all.</li> </ul>																																													



Ctrl Command	get_head_status
Comments (cont'd)	<ul style="list-style-type: none"> <li>With an XY2-100 converter the value 0 is returned, if no scan system is connected or not switched on.</li> <li>The Power Status and Temperature Status signals deliver combined status information of both axes.</li> <li>In any case also obey the status signal information described in the manual of your scan system.</li> <li>With iDRI<sup>E</sup> scan systems (see glossary entry on <a href="#">page 879</a>)             <ul style="list-style-type: none"> <li>Without the SL2-100 interface, <b>get_head_status</b> only returns meaningful return information if the XY2-100 status word was selected for return transmission.</li> <li>the Position Acknowledge signals of the x- and y axis are logically AND-connected and only returned as a common signal (for example, at Bit #3,4).</li> <li>after a reset or power-up of the scan system, it can take around 5 seconds for data to be returned from the scan system.</li> </ul> </li> <li>Status signals can also be queried by <a href="#">get_value</a>, <a href="#">get_values</a>, <a href="#">set_trigger</a> and <a href="#">set_trigger4</a>.</li> <li>See also <a href="#">Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 222</a> for information about using two scan heads.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: <ul style="list-style-type: none"> <li>The RTC6 allows the simultaneous reading of both scan head connectors' status words, and with iDRI<sup>E</sup> scan systems with SL2-100 interface even independently of the signal to be returned (set by <a href="#">control_command</a>).</li> <li>With iDRI<sup>E</sup> scan systems, Bit #0 and Bit #1 do not return information about the operational readiness of the scan system (see <a href="#">get_value</a>).</li> </ul>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_value</a> , <a href="#">get_values</a> , <a href="#">set_trigger</a> , <a href="#">set_trigger4</a> , <a href="#">get_waveform</a>



<b>Ctrl Command</b>	<b>get_hex_version</b>
<b>Function</b>	Returns the version number of the <b>DSP</b> program file <b>RTC6OUT.out</b> , which is currently loaded on the RTC6.
<b>Call</b>	<code>HexVersion = get_hex_version()</code>
<b>Result</b>	Version number. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The version numbers of program files are in the range 600...699. <b>get_hex_version</b> returns the following values:             <ul style="list-style-type: none"> <li>– if the <b>Option "3D"</b> is <i>not</i> enabled values in the range 2600...2699 (version number + 2000)</li> <li>– if the <b>Option "3D"</b> is <i>enabled</i> values in the range 3600...3699 (version number + 3000)</li> </ul> </li> <li>The file name extension for RTC6 <b>DSP</b> program files is <code>*.out</code>. See also <b>load_program_file</b>.</li> <li>The software version number can also be returned after an <b>RTC6_VERSION_MISMATCH</b> or <b>RTC6_ACCESS_DENIED</b> error. The return value is 0 if no program has yet been loaded. Here, an <b>RTC6_TIMEOUT</b> error is not generated.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_dll_version</a> , <a href="#">get_RTC_version</a>

<b>Ctrl Command</b>	<b>get_hi_data</b>								
<b>Function</b>	Returns the Home-In positions, last determined by <b>auto_cal</b> of the scan system attached to the first scan head connector.								
<b>Call</b>	<code>get_hi_data( &amp;X1, &amp;X2, &amp;Y1, &amp;Y2)</code>								
<b>Returned parameter values</b>	<table> <tr> <td>X1</td> <td>Coordinates of the currently stored Home-In positions. In bits.</td> </tr> <tr> <td>X2</td> <td>As pointers to signed 32-bit values.</td> </tr> <tr> <td>Y1</td> <td></td> </tr> <tr> <td>Y2</td> <td></td> </tr> </table>	X1	Coordinates of the currently stored Home-In positions. In bits.	X2	As pointers to signed 32-bit values.	Y1		Y2	
X1	Coordinates of the currently stored Home-In positions. In bits.								
X2	As pointers to signed 32-bit values.								
Y1									
Y2									
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>get_hi_data</b> is synonymous with <b>get_hi_pos</b> with <b>HeadNo</b> = 1 (see comments there).</li> </ul>								
RTC4→RTC6	Unchanged functionality. In addition: increased value range. The returned values are in the 20-bit value range.								
RTC5→RTC6	Unchanged functionality.								
Version info	Available as of version DLL 600, OUT 600, RBF 600.								
References	<a href="#">get_hi_pos</a> , <a href="#">write_hi_pos</a>								



<b>Ctrl Command</b>	<b>get_hi_pos</b>
<b>Function</b>	Returns the Home-In positions, last determined (by <b>auto_cal</b> ) of the scan system attached to the specified scan head connector.
<b>Call</b>	<code>get_hi_pos( HeadNo, &amp;X1, &amp;X2, &amp;Y1, &amp;Y2 )</code>
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value.            Allowed values.            = 1:    First scan head connector.            = 2:    Second scan head connector.</p>
<b>Returned parameter values</b>	<p>X1      Coordinates of the currently stored (last determined) Home-In positions.            In bits.            Y1      As pointers to signed 32-bit values.</p> <p>X2            Y2</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For information on using <b>get_hi_pos</b>, see <a href="#">Section "Customer-Specific Calibration", page 264</a>.</li> <li>• Make sure that the scan system currently attached to the specified scan head connector is the same scan system which has been used to determine the returned Home-In positions. For determination of Home-In position values, this scan system should be equipped with an internal sensor system for automatic self-calibration (Home-In sensors).</li> <li>• The returned values are 0, if:               <ul style="list-style-type: none"> <li>– no scan system equipped with automatic self-calibration (Home-In sensors) is attached to the specified scan head connector</li> <li>– an error has occurred during determination of the Home-In values for such a system</li> </ul> </li> <li>• Directly after initialization (<b>init_rtc6_dll</b>), particularly prior to a first call of the command <b>auto_cal</b>( <b>Command</b> = 0, 1 or 3 ), the returned values are the Home-In reference values stored in the <b>Flash memory</b>. If such reference values have not been successfully determined at least once by <b>auto_cal</b>( <b>Command</b> = 0 ), <b>get_hi_pos</b> returns 0.</li> <li>• <b>get_hi_pos</b> is available even without (current) explicit access rights to a specific RTC6 board. However, the return values are 0 as long as no access to the addressed board has been successful at least once before.</li> <li>• If parameter values are invalid, then all returned coordinates are 0 (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#"><b>get_hi_data</b></a> , <a href="#"><b>auto_cal</b></a> , <a href="#"><b>set_hi</b></a> , <a href="#"><b>write_hi_pos</b></a>



<b>Ctrl Command</b>	<b>get_input_pointer</b>
<b>Function</b>	Returns the present absolute position of the input pointer.
<b>Call</b>	<code>InputPointer = get_input_pointer()</code>
<b>Result</b>	Position of the input pointer [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The position of the input pointer corresponds to the position in RTC6 list memory (also in the protected "List 3" area), where the next list command is stored. The number of still-available storage positions there can be queried by <a href="#">get_list_space</a>.</li> <li><b>get_input_pointer</b> returns the absolute list memory address (offset relative to the start of "List 1"). The relative position referenced to the start of the respective list area can be queried by <a href="#">get_list_pointer</a>.</li> <li>Before loading a non-indexed subroutine or character set, you should use <b>get_input_pointer</b> to obtain the start address if subsequent referencing is to be performed by <a href="#">set_sub_pointer</a> or <a href="#">set_char_pointer</a>.</li> <li>The absolute position of the output pointer can be queried by <a href="#">get_status</a> or <a href="#">get_out_pointer</a>.</li> <li>The board-specific error variables <code>LastErrorHandler</code>, see <a href="#">Section "Error Handling", page 120</a>, are neither generated nor altered by <b>get_input_pointer</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_list_pointer</a> , <a href="#">set_input_pointer</a> , <a href="#">get_list_space</a> , <a href="#">get_status</a> , <a href="#">get_out_pointer</a>

<b>Ctrl Command</b>	<b>get_io_status</b>
<b>Function</b>	Returns the current state of the 16-bit digital output port on the EXTENSION 1 socket connector.
<b>Call</b>	<code>IOStatus = get_io_status()</code>
<b>Result</b>	16-bit value (DIGITAL OUT0...DIGITAL OUT15). As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>get_io_status</b> is designed for use in combination with <a href="#">set_io_cond_list</a> and <a href="#">clear_io_cond_list</a>. See also <a href="#">Section "Example Code (PASCAL)", page 282</a>.</li> <li>See also <a href="#">Section "16-Bit Digital Input and 16-Bit Digital Output", page 67</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">write_io_port</a> , <a href="#">write_io_port_mask</a> , <a href="#">set_io_cond_list</a> , <a href="#">clear_io_cond_list</a>



<b>Ctrl Command</b>	<b>get_jump_table</b>
<b>Function</b>	Reads out the jump delay table which is currently stored on the board. Then copies the 1024 corresponding unsigned 16-bit values to the specified PC address.
<b>Call</b>	ErrorCode = get_jump_table( Addr )
<b>Parameters</b>	Addr      PC address for the 2048 byte memory area.
<b>Result</b>	Error code. As an unsigned 32-bit value. 0            No error. 11          RTC6 board driver error.
<b>Notes</b>	<ul style="list-style-type: none"> <li>Do not call <b>get_jump_table</b> during processing of a list.</li> <li>The data format is “1024 16-bit values” representing the delay values for a piecewise linear interpolation at the sampling points (=jump lengths) <math>N \times 1024</math> with <math>0 \leq N &lt; 1024</math>. The values can thus also be directly generated or modified by users.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
Reference	<b>set_jump_table</b> , <b>load_jump_table_offset</b>

<b>Ctrl Command</b>	<b>get_lap_time</b>
<b>Function</b>	Returns the <i>current</i> RTC6 timer value (without resetting it to zero).
<b>Call</b>	TimerValue = get_lap_time()
<b>Result</b>	RTC6 timer value in seconds since the last call of <b>save_and_restart_timer</b> . As a 64-bit IEEE floating point value.
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>get_lap_time</b> serves to query the elapsed time of a time consuming marking during processing.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_time</b> , <b>save_and_restart_timer</b>



<b>Ctrl Command</b>	<b>get_laser_pin_in</b>
<b>Function</b>	Returns the current status of the 2-Bit-Digital digital input port at the LASER connector, see also <a href="#">Section "2-Bit Digital Input Port", page 63.</a>
<b>Call</b>	LaserPinIn = get_laser_pin_in()
<b>Result</b>	As an unsigned 32-bit value. Bit #0      DIGITAL IN1. <a href="#">(LSB)</a> Bit #1      DIGITAL IN2. Bit #2      Reserved. ...           ... Bit 31      Reserved.
<b>Comments</b>	• –
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_laser_pin_out</a>

<b>Ctrl Command</b>	<b>get_last_error</b>
<b>Function</b>	Returns an error code listing any errors which occurred during execution of the most recent command.
<b>Call</b>	<code>LastError = get_last_error()</code>
<b>Result</b>	Error code. As an unsigned 32-bit value. If multiple errors occurred simultaneously, then multiple bits are set. The meanings of bit numbers, error types and error constants is identical to those for <a href="#">get_error</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>For error handling see <a href="#">Chapter 6.8 "Error Handling", page 120</a>.</li> <li><code>get_last_error</code> and <code>n_get_last_error</code> are available even without explicit access rights to a specific RTC6 board.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code>, see <a href="#">Chapter 6.8 "Error Handling", page 120</a>, are neither generated nor altered by <code>get_last_error</code>.</li> <li>Each <code>eth_get_last_error</code> error also leads to a <code>get_last_error</code> error <code>RTC6_ETH_ERROR</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">eth_get_last_error</a> , <a href="#">get_error</a> , <a href="#">reset_error</a> , <a href="#">set_verify</a>

<b>Ctrl Command</b>	<b>get_list_pointer</b>				
<b>Function</b>	Returns the current relative position of the input pointer as well as the list number.				
<b>Call</b>	<code>get_list_pointer( &amp;ListNo, &amp;Pos )</code>				
<b>Returned parameter values</b>	<table> <tr> <td>ListNo</td> <td>Number of the list in which the input pointer is currently located. As a pointer to an unsigned 32-bit value. [1...3].</td> </tr> <tr> <td>Pos</td> <td>Current position of the input pointer (offset relative to the start of the respective list). As a pointer to an unsigned 32-bit value.</td> </tr> </table>	ListNo	Number of the list in which the input pointer is currently located. As a pointer to an unsigned 32-bit value. [1...3].	Pos	Current position of the input pointer (offset relative to the start of the respective list). As a pointer to an unsigned 32-bit value.
ListNo	Number of the list in which the input pointer is currently located. As a pointer to an unsigned 32-bit value. [1...3].				
Pos	Current position of the input pointer (offset relative to the start of the respective list). As a pointer to an unsigned 32-bit value.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>The absolute list memory address (offset relative to the start of "List 1") of the input pointer can be queried by <code>get_input_pointer</code> (see also comments there).</li> <li>The number of list positions until the end of the respective list (from the input pointer) can be queried by <code>get_list_space</code>.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code>, see <a href="#">Chapter 6.8 "Error Handling", page 120</a>, are neither generated nor altered by <code>get_list_pointer</code>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">get_input_pointer</a> , <a href="#">get_list_space</a>				



<b>Ctrl Command</b>	<b>get_list_serial</b>
<b>Function</b>	Returns the number of the serial-number-set most recently selected by <b>select_serial_set_list</b> (or of serial-number-set 0 after <b>load_program_file</b> ) and the current serial number of this serial-number-set.
<b>Call</b>	LastMarkedSerialNo = get_list_serial( &Set )
<b>Result</b>	Serial number. As a 64-bit IEEE floating point value.
<b>Returned parameter values</b>	Set      Number of the selected serial-number-set. As a pointer to an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The serial number queried by <b>get_list_serial</b> is typically the one most recently marked by <b>mark_serial</b> or <b>mark_serial_abs</b>.</li> <li>For usage of <b>get_list_serial</b>, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 197</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">select_serial_set_list</a> , <a href="#">get_serial</a>

<b>Ctrl Command</b>	<b>get_list_space</b>
<b>Function</b>	Returns the amount of free list memory, hence the number of list commands that can still be loaded from the input pointer's current position to the last position in the respective list.
<b>Call</b>	ListSpace = get_list_space()
<b>Result</b>	Number of free list positions. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>If an indexed subroutine or indexed character set is currently being loaded into the protected list memory area "List 3", then <b>get_list_space</b> returns the amount of still-available protected memory (otherwise the input pointer is not located in the protected area).</li> </ul>
RTC4→RTC6	<b>get_list_space</b> was available on the RTC4 to support their circular queue mode and returned the distance between the input and output pointers. The RTC6 does not have a circular queue mode. The input pointer's position can be queried by <b>get_input_pointer</b> or <b>get_list_pointer</b> and the output pointer's position can be queried by <b>get_status</b> or <b>get_out_pointer</b> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_input_pointer</a> , <a href="#">get_status</a> , <a href="#">get_out_pointer</a> , <a href="#">get_list_pointer</a>



Ctrl Command	get_marking_info
Function	Returns information about any boundary exceedances during Processing-on-the-fly correction as well as improper encoder signals. <b>get_marking_info</b> also returns the error bits of automatic suppression of laser control signals.
Call	MarkingInfo = get_marking_info()
Result	<p>Error code. As an unsigned 32-bit value.</p> <p>Bit #0           = 1: Processing-on-the-fly underflow in x direction (<math>X &lt; -524,288</math>). <b>(LSB)</b></p> <p>Bit #1           = 1: Processing-on-the-fly overflow in x direction (<math>X &gt; +524,287</math>).</p> <p>Bit #2           = 1: Processing-on-the-fly underflow in y direction (<math>Y &lt; -524,288</math>).</p> <p>Bit #3           = 1: Processing-on-the-fly overflow in y direction (<math>Y &gt; +524,287</math>).</p> <p>Bit #4           = 1: Processing-on-the-fly underflow in x direction (<math>X &lt; X_{min}</math>).</p> <p>Bit #5           = 1: Processing-on-the-fly overflow in x direction (<math>X &gt; X_{max}</math>).</p> <p>Bit #6           = 1: Processing-on-the-fly underflow in y direction (<math>Y &lt; Y_{min}</math>).</p> <p>Bit #7           = 1: Processing-on-the-fly overflow in y direction (<math>Y &gt; Y_{max}</math>).</p> <p>Bit #8           = 1: TriggerError: an enabled external trigger or simulated trigger occurred during execution of a list.</p> <p>Bit #9           = 1: An error has occurred during activation of Processing-on-the-fly correction by <b>activate_fly_2d</b>, <b>activate_fly_2d_encoder</b>, <b>activate_fly_xy</b> or <b>activate_fly_xy_encoder</b> ("ActivateFlyError"). See also <a href="#">Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications"</a>, <a href="#">page 237</a>, <a href="#">Chapter 8.6.8 "Encoder Resets"</a>, <a href="#">page 239</a> and <a href="#">Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections"</a>, <a href="#">page 240</a>.</p> <p>Bit #10          PosAck error bit of head A's x axis.</p> <p>Bit #11          TempOK error bit of head A's x axis.</p> <p>Bit #12          PowerOK error bit of head A's x axis.</p> <p>Bit #13          PosAck error bit of head A's y axis.</p> <p>Bit #14          TempOK error bit of head A's y axis.</p> <p>Bit #15          PowerOK error bit of head A's y axis.</p> <p>Bit #16          = 1: Signal 1 of encoder input ENCODER X has too-short spacing.</p> <p>Bit #17          = 1: Signal 2 of encoder input ENCODER X has too-short spacing.</p> <p>Bit #18          = 1: Signal 1 of encoder input ENCODER Y has too-short spacing.</p> <p>Bit #19          = 1: Signal 2 of encoder input ENCODER Y has too-short spacing.</p> <p>Bit #20          = 1: Improper signal sequence at encoder input ENCODER X.</p> <p>Bit #21          = 1: Improper signal sequence at encoder input ENCODER Y.</p> <p>Bit #22          = 1: Processing-on-the-fly underflow in z direction (<math>Z &lt; -524,288</math>).</p> <p>Bit #23          = 1: Processing-on-the-fly overflow in z direction (<math>Z &gt; +524,287</math>).</p> <p>Bit #24          = 1: Processing-on-the-fly underflow in z direction (<math>Z &lt; Z_{min}</math>).</p> <p>Bit #25          = 1: Processing-on-the-fly overflow in z direction (<math>Z &gt; Z_{max}</math>).</p>



Ctrl Command	get_marking_info
Result (cont'd)	<p>Bit #26      PosAck error bit of head B's x axis.</p> <p>Bit #27      TempOK error bit of head B's x axis.</p> <p>Bit #28      PowerOK error bit of head B's x axis.</p> <p>Bit #29      PosAck error bit of head B's y axis.</p> <p>Bit #30      TempOK error bit of head B's y axis.</p> <p>Bit #31      PowerOK error bit of head B's y axis.</p> <p>The remaining bits are reserved.</p>
Comments	<ul style="list-style-type: none"> <li>For usage of <code>get_marking_info</code> and of the error bits Bit #0...Bit #7, see <a href="#">Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections", page 240</a>.</li> <li>The boundary limits Xmin, Xmax, Ymin, Ymax for the customer-defined monitoring area (Bit #4...Bit #7) can be specified by <code>set_fly_limits</code>.</li> <li>Encoder-signal spacing could be too short if interfering signals are present, a rapid directional change occurs or the frequency is essentially too high.</li> <li>An improper encoder signal sequence occurs if both signals 1 and 2 change simultaneously, thus hindering determination of the counting direction.</li> <li>If no external encoder is used, the corresponding inputs on the MARKING ON THE FLY socket connector have undefined signal levels. Therefore signal errors may be permanently detected for these inputs (upper 16 bit of the return value), even if simulated encoders are used. If this disturbs the application, then this pins should be set to a defined signal level by pull-up resistors.</li> <li>The error bits Bit #10...Bit #15 and Bit #26...Bit #31 are only set in case of an error if automatic suppression of laser control signals has been activated, see <a href="#">Section "Automatic Suppression of Laser Control Signals", page 176</a>. Any time an error occurs, all error bits corresponding to an error-indicating status signal are (cumulatively) set. If applicable, even error bits are set, which have not been selected to be used for automatic suppression of laser control signals by <code>set_laser_control</code>. All error bits are reset by <code>get_marking_info</code>.</li> <li>The boundary limits Zmin, Zmax for the customer-defined monitoring area (Bit #24...Bit #25) can be specified by <code>set_fly_limits_z</code>.</li> <li>All error bits are reset during initialization (by <code>load_program_file</code>).</li> <li>Error bits Bit #22...Bit #23 are also reset by <code>get_marking_info</code>. Error bits Bit #24...Bit #25 are not reset by <code>get_marking_info</code>. Error bits Bit #24...Bit #25 get implicitly reset by the conditional commands. They can also be explicitly reset by <code>clear_fly_overflow</code>.</li> </ul>
RTC4→RTC6	Unchanged functionality for info bits that are also used on the RTC4.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>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_notActivated</code>



<b>Ctrl Command</b>	<b>get_master_slave</b>
<b>Function</b>	Returns the master/slave status of the addressed RTC6 board.
<b>Call</b>	<code>MasterSlaveStatus = get_master_slave()</code>
<b>Result</b>	<p>Master/slave status. As an unsigned 32-bit value. Information whether a further RTC6 board is connected to the Master or Slave connector of the addressed RTC6 board:</p> <ul style="list-style-type: none"> <li>Bit #0      = 1:    A RTC6 board is connected to the Slave connector. <b>(LSB)</b></li> <li>Bit #1      = 1:    A RTC6 board is connected to the Master connector.</li> <li>Bit #2      = 0.</li> <li>...           ...</li> <li>Bit #31     = 0.</li> </ul> <p>Information, whether the addressed board is operated as a master, slave or single board:</p> <ul style="list-style-type: none"> <li>= 0    Single board.</li> <li>= 1    Slave (without any further downstream slave board).</li> <li>= 2    Master.</li> <li>= 3    Slave (together with a downstream slave board).</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For usage of <b>get_master_slave</b>, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 114</a>.</li> <li>• <b>get_master_slave</b> only returns the status, if the addressed RTC6 board has been previously initialized by <a href="#">load_program_file</a>. Otherwise, 0 is returned.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">sync_slaves</a>

<b>Ctrl Command</b>	<b>get_mcbsp</b>
<b>Function</b>	Returns the most recent input value that was fully transferred by the <a href="#">McBSP interface</a> to the memory location for Processing-on-the-fly applications.
<b>Call</b>	<code>mcbsp_value = get_mcbsp()</code>
<b>Result</b>	Input value. As a signed 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>get_mcbsp</b> is equivalent to <a href="#">read_mcbsp(0)</a> (see notes there).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">read_mcbsp</a>



<b>Undelayed Short List Command</b>	<b>get_mcbsp_list</b>
Function	Without function.
Call	get_mcbsp_list()
Comments	• <b>get_mcbsp_list</b> has no effect on the user program.
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_mcbsp</b>



<b>Ctrl Command</b>	<b>get_out_pointer</b>
<b>Function</b>	Returns the current (or most recent) position of the output pointer as offset relative to the start of the respective list and the list number.
<b>Call</b>	<code>get_out_pointer( &amp;ListNo, &amp;Pos )</code>
<b>Returned parameter values</b>	ListNo      Number of the list ("List 1" or "List 2") in which the output pointer is currently located (or in which it most recently resided). As a pointer to an unsigned 32-bit value.
	Pos      Current (or most recent) position of the output pointer (relative memory address). As a pointer to an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>get_out_pointer</code> calls <code>get_status</code> (see the comments there, particularly with respect to "List 3") and uses the command's returned absolute output pointer position to determine the list number and the relative position within the list. The relative position and list number returned by <code>get_out_pointer</code> simplify comparing the output pointer position to the current input pointer position (particularly with respect to list consistency) during an alternative list change.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_status</a> , <a href="#">get_input_pointer</a> , <a href="#">get_list_pointer</a>

<b>Ctrl Command</b>	<b>get_overrun</b>
<b>Function</b>	Returns the number of overruns of the 10 µs clock period since the last call and resets the overrun counter.
<b>Call</b>	<code>NumberOfOverruns = get_overrun()</code>
<b>Result</b>	Number of overruns of the 10 µs clock period since the last call of <code>get_overrun</code> . As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">Section "Clock Overruns", page 171</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–



<b>Ctrl Command</b>	<b>get_RTC_mode</b>
<b>Function</b>	Returns the currently set operation mode of the <b>RTC6 DLL</b> .
<b>Call</b>	<code>DLLMode = get_RTC_mode()</code>
<b>Result</b>	<p>DLL operation mode as a 32-bit value.</p> <ul style="list-style-type: none"> <li>= 4: <b>RTC4 Compatibility Mode</b>.</li> <li>= 5: <b>RTC5 Compatibility Mode</b>.</li> <li>= 6: <b>RTC6 Standard Mode</b> (default setting).</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• The <b>RTC6 DLL</b> operation mode can be set by <b>set_RTC4_mode</b>, <b>set_RTC5_mode</b> or <b>set_RTC6_mode</b>. The default setting is <b>RTC6 Standard Mode</b>.</li> <li>• The <b>get_RTC_mode</b> command is available even without explicit access rights to a particular RTC6 board.</li> <li>• <b>get_RTC_mode</b> is not available as a multi-board command.</li> <li>• The board-specific error variables <code>LastError</code> and <code>AccError</code>, see <a href="#">Chapter 6.8 "Error Handling", page 120</a>, are neither generated nor altered by <b>get_RTC_mode</b>.</li> </ul>
<b>RTC4→RTC6</b>	New command.
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#"><b>set_RTC4_mode</b></a> , <a href="#"><b>set_RTC5_mode</b></a> , <a href="#"><b>set_RTC6_mode</b></a>



<b>Ctrl Command</b>	<b>get_RTC_version</b>																										
<b>Function</b>	Returns the version number of the RTC6 firmware ( <a href="#">RTC6RBF.rbf</a> ) and additional information about enabled options of the RTC6 board.																										
<b>Call</b>	<code>RTCVersion = get_RTC_version()</code>																										
<b>Result</b>	<p>RTC6 version number. As an unsigned 32-bit value.</p> <table> <tr> <td>Bit #0 (<a href="#">LSB</a>)</td> <td>Firmware version of the RTC6 board.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #7</td> <td></td> </tr> <tr> <td>Bit #8</td> <td>= 1: <a href="#">Option Processing-on-the-fly</a> is enabled. See also <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>.</td> </tr> <tr> <td>Bit #9</td> <td>= 1: <a href="#">Option "Second Scan Head Control"</a> is enabled. See also <a href="#">Section "2. SCANHEAD Socket Connector (Connector for Second Scan Head)", page 57</a>.</td> </tr> <tr> <td>Bit #10</td> <td>= 1: <a href="#">Option "3D"</a> is enabled. See also <a href="#">Chapter 8.5.2 "3D Scan Systems", page 222</a>.</td> </tr> <tr> <td>Bit #11</td> <td>Reserved.</td> </tr> <tr> <td>Bit #12</td> <td>= 1: <a href="#">Option "SCANa"</a> is enabled (as of DLL 605).</td> </tr> <tr> <td>Bit #13</td> <td>= 1: <a href="#">Option "UFPM"</a> is enabled (as of DLL 605).</td> </tr> <tr> <td>Bit #14</td> <td>= 1: <a href="#">Option "syncA"</a> is enabled (as of DLL 607).</td> </tr> <tr> <td>Bit #15</td> <td>Reserved.</td> </tr> <tr> <td>Bits #16...#23</td> <td><a href="#">DSP</a> version number.</td> </tr> <tr> <td>Bits #24...#31</td> <td>Firmware subversion of the RTC6 board.</td> </tr> </table>	Bit #0 ( <a href="#">LSB</a> )	Firmware version of the RTC6 board.	...		Bit #7		Bit #8	= 1: <a href="#">Option Processing-on-the-fly</a> is enabled. See also <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> .	Bit #9	= 1: <a href="#">Option "Second Scan Head Control"</a> is enabled. See also <a href="#">Section "2. SCANHEAD Socket Connector (Connector for Second Scan Head)", page 57</a> .	Bit #10	= 1: <a href="#">Option "3D"</a> is enabled. See also <a href="#">Chapter 8.5.2 "3D Scan Systems", page 222</a> .	Bit #11	Reserved.	Bit #12	= 1: <a href="#">Option "SCANa"</a> is enabled (as of DLL 605).	Bit #13	= 1: <a href="#">Option "UFPM"</a> is enabled (as of DLL 605).	Bit #14	= 1: <a href="#">Option "syncA"</a> is enabled (as of DLL 607).	Bit #15	Reserved.	Bits #16...#23	<a href="#">DSP</a> version number.	Bits #24...#31	Firmware subversion of the RTC6 board.
Bit #0 ( <a href="#">LSB</a> )	Firmware version of the RTC6 board.																										
...																											
Bit #7																											
Bit #8	= 1: <a href="#">Option Processing-on-the-fly</a> is enabled. See also <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> .																										
Bit #9	= 1: <a href="#">Option "Second Scan Head Control"</a> is enabled. See also <a href="#">Section "2. SCANHEAD Socket Connector (Connector for Second Scan Head)", page 57</a> .																										
Bit #10	= 1: <a href="#">Option "3D"</a> is enabled. See also <a href="#">Chapter 8.5.2 "3D Scan Systems", page 222</a> .																										
Bit #11	Reserved.																										
Bit #12	= 1: <a href="#">Option "SCANa"</a> is enabled (as of DLL 605).																										
Bit #13	= 1: <a href="#">Option "UFPM"</a> is enabled (as of DLL 605).																										
Bit #14	= 1: <a href="#">Option "syncA"</a> is enabled (as of DLL 607).																										
Bit #15	Reserved.																										
Bits #16...#23	<a href="#">DSP</a> version number.																										
Bits #24...#31	Firmware subversion of the RTC6 board.																										
<b>Comments</b>	<ul style="list-style-type: none"> <li>The RTC6 firmware version numbers are in the range 600...699.</li> <li><code>get_RTC_version( Bit #0...Bit #7 )</code> returns values in the range 0...99 (version number - 600).</li> <li>The current <a href="#">DSP</a> version number is 3.</li> <li>The current firmware subversion is 0.</li> <li>The firmware version can also be returned after an <a href="#">RTC6_VERSION_MISMATCH</a> or <a href="#">RTC6_ACCESS_DENIED</a> error. The return value is 0, if no program has yet been loaded. Here, an <a href="#">RTC6_TIMEOUT</a> error is not generated.</li> </ul>																										
RTC4→RTC6	Unchanged functionality.																										
RTC5→RTC6	Unchanged functionality.																										
Version info	Available as of version DLL 600, OUT 600, RBF 600.																										
References	<a href="#">get_hex_version</a> , <a href="#">get_dll_version</a>																										



<b>Ctrl Command</b>	<b>get_scanahead_params</b>
Comments	<p><i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i></p> <p>This command is described in the manual “excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards”.</p>



<b>Ctrl Command</b>	<b>get_serial</b>
<b>Function</b>	Returns the current serial number of the serial-number-set selected by <a href="#">select_serial_set</a> (or of serial-number-set 0 after <a href="#">load_program_file</a> ).
<b>Call</b>	<code>CurrentSerialNo = get_serial()</code>
<b>Result</b>	Serial number. As a 64-bit IEEE floating point value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <code>get_serial</code>, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 197</a>.</li> <li><code>get_serial</code> should not be confused with <code>get_serial_number</code>, which returns the product serial number of the RTC6 board.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">select_serial_set</a>

<b>Ctrl Command</b>	<b>get_serial_number</b>
<b>Function</b>	Returns the individual serial number of the active RTC6 board.
<b>Call</b>	<code>RTCSerialNumber = get_serial_number()</code>
<b>Result</b>	RTC6 serial number. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>Serial numbers of installed boards are ascertained by <a href="#">init_rtc6_dll</a> and cached in the <a href="#">RTC6 DLL</a>, from where you can query them by <code>get_serial_number</code>.</li> <li><code>get_serial_number</code> is helpful when using several RTC6 boards in one computer (see <a href="#">Chapter 6.6 "Using Several RTC6 PCIe Boards in One PC", page 113</a>). The associated multi-board command <code>n_get_serial_number</code> can be used for determining the relationship between the installed boards and the <a href="#">RTC6 DLL</a>-internal numbers assigned to them during initialization. The <a href="#">RTC6 DLL</a>-internal numbers are newly assigned during each initialization of a user program (see <a href="#">init_rtc6_dll</a>) and must be supplied for a variety of commands (in particular, all multi-board commands). The number of boards found during initialization can be queried by <a href="#">rtc6_count_cards</a>.</li> <li><code>get_serial_number</code> and <code>n_get_serial_number</code> is available even without (current) explicit access rights to a specific RTC6 board. However, the return values are 0 as long as no access to the addressed board has been successful at least once before.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <code>get_serial_number</code>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">rtc6_count_cards</a>



<b>Ctrl Command</b>	<b>get_standby</b>				
<b>Function</b>	Returns the currently set standby parameters.				
<b>Call</b>	<code>get_standby( &amp;HalfPeriod, &amp;PulseLength )</code>				
<b>Returned parameter values</b>	<table> <tr> <td>HalfPeriod</td> <td>Half of the currently set standby output period of the standby pulses. As a pointer to an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i></td> </tr> <tr> <td>PulseLength</td> <td>Currently set pulse length of the standby pulses. As a pointer to an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i></td> </tr> </table>	HalfPeriod	Half of the currently set standby output period of the standby pulses. As a pointer to an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i>	PulseLength	Currently set pulse length of the standby pulses. As a pointer to an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i>
HalfPeriod	Half of the currently set standby output period of the standby pulses. As a pointer to an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i>				
PulseLength	Currently set pulse length of the standby pulses. As a pointer to an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i>				
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>get_standby</b>, see <a href="#">Section "Signals for "Laser Standby" Operation", page 175</a>.</li> </ul>				
RTC4→RTC6	New command.  <a href="#">In RTC4 Compatibility Mode</a> , the RTC6 divides the parameter values by 8.				
RTC5→RTC6	Unchanged functionality.				
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.				
<b>References</b>	<a href="#">set_standby</a>				



Ctrl Command	get_startstop_info
Function	Provides information about internal and external starts and stops since the last time <b>get_startstop_info</b> has been called. Also provided are the current external start and stop levels, the status and signal level of the laser control signals, and possible transmission errors to and from the attached scan system.
Call	<code>StartStopInfo = get_startstop_info()</code>
Result	<p>Info signal. As an unsigned 32-bit value.</p> <p>Bit #0      = 1: An <i>internal start</i> has been executed (by <b>execute_list</b> or similar) since the last <b>get_startstop_info</b> has been called. (LSB)</p> <p>Bit #1      = 1: An <i>external start</i> has been executed (by /START, /START2, /Slave-START, <b>simulate_ext_start</b> or <b>simulate_ext_start_ctrl</b>) since the last <b>get_startstop_info</b> has been called.</p> <p>Bit #2      = 1: An <i>internal stop</i> has been executed (by <b>stop_execution</b>) since the last <b>get_startstop_info</b> has been called.</p> <p>Bit #3      = 1: An <i>external stop</i> has been executed (by /STOP, /STOP2, /Slave-STOP or <b>simulate_ext_stop</b>) since the last <b>get_startstop_info</b> has been called.</p> <p>Bit #4      Ext-stop status (= logical AND operation of the signals /STOP, /STOP2, /Slave-STOP and <b>simulate_ext_stop</b>, see <b>figure 63</b>): = 1: No stop signals are currently present at the inputs or the inputs are not connected. = 0: There is a stop signal at at least one input.</p> <p>Bit #5      Reserved.</p> <p>Bit #6      Reserved.</p> <p>Bit #7      Reserved.</p> <p>Bit #8      Reserved.</p> <p>Bit #9      = 1: The laser control signals are globally enabled, see <b>Chapter 7.4.1 "Enabling, Activating and Switching Laser Control Signals"</b>, <b>page 173</b>. See also <b>set_laser_control</b>.</p> <p>Bit #10     = 1: The TTL laser control signals at the LASER1 and LASER2 output ports are <i>active-LOW</i> (the signal level can be defined by <b>set_laser_control</b>).</p> <p>Bit #11     = 1: Since the last call of <b>get_startstop_info</b>, at least one external start has failed (more external starts were triggered than could be simultaneously held in the 8-start wait loop).</p> <p>Bit #12     Ext-Start status (= logical AND operation of the signals /START, /START2 and /Slave-START, see <b>figure 63</b>): = 1: No start signals are currently present at the inputs or the inputs are not connected. = 0: A start signal is present at at least one input.</p> <p>Bit #13     = 1: The TTL laser control signal at the LASERON output port is <i>active-LOW</i> (the signal level can be defined by <b>set_laser_control</b>).</p>



Ctrl Command	get_startstop_info
Result (cont'd)	<p>Bit #14 = 1: The laser control signals are enabled (<a href="#">enable_laser</a>). = 0: The laser control signals are disabled (<a href="#">disable_laser</a>).</p> <p>Bit #15 Reserved.</p> <p>Bit #16 Error bits. Get set when an error occurs during data transmission from the scan system: ... Bit #16...Bit #23 for the first scan head connector, Bit #24...Bit #31 for the second scan head connector.</p> <p>Bit #16, Bit #24: Incorrect number of frames within a data block.</p> <p>Bit #17, Bit #25: Incorrect pulse length of signal received from scan system, maybe no scan system is connected.</p> <p>Bit #18, Bit #26: Preamble sequence incorrect.</p> <p>Bit #19, Bit #27: Bit count within a subframe incorrect.</p> <p>Bit #20, Bit #28: Parity error when reading data received from scan system.</p> <p>Bit #21, Bit #29: The present data is invalid (old).</p> <p>Bit #22, Bit #30: Reserved.</p> <p>Bit #23, Bit #31: Reserved.</p>
Comments	<ul style="list-style-type: none"> <li>After <b>get_startstop_info</b> has been executed, reset are:           <ul style="list-style-type: none"> <li>– Info bit <b>Bit #0...Bit #3</b></li> <li>– Error bit <b>Bit #16...Bit #31</b></li> </ul> </li> <li>• See also <b>Section "External Stop", page 275</b> and <b>Section "External Start", page 276</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality for the info bits that are also used on the RTC4.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_counts</a> , <a href="#">get_status</a> , <a href="#">set_control_mode</a>



<b>Ctrl Command</b>	<b>get_status</b>																																	
<b>Function</b>	Returns the current list execution status ( <b>BUSY</b> , <b>INTERNAL-BUSY</b> and <b>PAUSED</b> ) and the current (or most recent) position of the output pointer.																																	
<b>Call</b>	get_status( &Status, &Pos )																																	
Returned parameter values	Status	<p>Status value. As a pointer to an unsigned 32-bit value.</p> <table> <tr> <td>Bit #0</td> <td>= 1: <b>BUSY</b> status set. (LSB)</td> </tr> <tr> <td>Bit #1</td> <td>Reserved.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #6</td> <td></td> </tr> <tr> <td>Bit #7</td> <td>= 1: <b>INTERNAL-BUSY</b> status set.</td> </tr> <tr> <td>Bit #8</td> <td>Reserved.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #14</td> <td></td> </tr> <tr> <td>Bit #15</td> <td>= 1: <b>PAUSED</b> status set.</td> </tr> <tr> <td>Bit #16</td> <td>Reserved.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #22</td> <td></td> </tr> <tr> <td>Bit #23</td> <td>= 1: HEAD_BUSY status set. Operating status for excelliSCAN, see manual "excelliSCAN scan heads – Functional Principle of SCANahead Servo Control and Operation by RTC6 Boards".</td> </tr> <tr> <td>Bit #24</td> <td>Reserved.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #31</td> <td></td> </tr> </table>	Bit #0	= 1: <b>BUSY</b> status set. (LSB)	Bit #1	Reserved.	...		Bit #6		Bit #7	= 1: <b>INTERNAL-BUSY</b> status set.	Bit #8	Reserved.	...		Bit #14		Bit #15	= 1: <b>PAUSED</b> status set.	Bit #16	Reserved.	...		Bit #22		Bit #23	= 1: HEAD_BUSY status set. Operating status for excelliSCAN, see manual "excelliSCAN scan heads – Functional Principle of SCANahead Servo Control and Operation by RTC6 Boards".	Bit #24	Reserved.	...		Bit #31	
Bit #0	= 1: <b>BUSY</b> status set. (LSB)																																	
Bit #1	Reserved.																																	
...																																		
Bit #6																																		
Bit #7	= 1: <b>INTERNAL-BUSY</b> status set.																																	
Bit #8	Reserved.																																	
...																																		
Bit #14																																		
Bit #15	= 1: <b>PAUSED</b> status set.																																	
Bit #16	Reserved.																																	
...																																		
Bit #22																																		
Bit #23	= 1: HEAD_BUSY status set. Operating status for excelliSCAN, see manual "excelliSCAN scan heads – Functional Principle of SCANahead Servo Control and Operation by RTC6 Boards".																																	
Bit #24	Reserved.																																	
...																																		
Bit #31																																		
	Pos	Current (or most recent) position of the output pointer (absolute memory address). As a pointer to an unsigned 32-bit value.																																
Comments	<ul style="list-style-type: none"> <li>For a description of when the <b>BUSY</b>, <b>INTERNAL-BUSY</b> or <b>PAUSED</b> status values are set or not set, see <a href="#">Chapter 6.4.3 "List Execution Status"</a>, page 98.</li> <li>(<b>BUSY</b> and <b>PAUSED</b> set) requires <b>restart_list</b> for continuation, (<b>BUSY</b> not set and <b>PAUSED</b> set) requires <b>release_wait</b> and (both <b>BUSY</b> and <b>PAUSED</b> not set) requires <b>execute_list_pos</b>. "Continuation" is not allowed with (<b>BUSY</b> set and <b>PAUSED</b> not set) and a currently running list. An improper continuation generates the <b>get_last_error</b> return code <b>RTC6_BUSY</b>. With (<b>INTERNAL-BUSY</b> set), <b>release_wait</b> and <b>execute_list_pos</b> are only executed with a delay (after <b>INTERNAL-BUSY</b> has been reset again).</li> <li>The output pointer points to the command (in "List 1" or "List 2") currently being executed or most recently executed. If, during processing of a subroutine in the protected list memory area "List 3", the output pointer's position <b>Pos</b> is queried, then the position is returned of the list command in the list memory area ("List 1" or "List 2") in which the output pointer most recently resided (typically from where the subroutine was called (for example, with <b>list_call</b>). <b>pause_list</b> and <b>set_wait</b> leave <b>Pos</b> unchanged.</li> </ul>																																	



Ctrl Command	get_status
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>get_status</b> returns the output pointer's position as an absolute memory address (offset relative to the start of "List 1"). The relative position referenced to the start of the respective list area can be queried by <b>get_out_pointer</b>.</li> <li>• The current input pointer position can be queried by <b>get_input_pointer</b> or <b>get_list_pointer</b>.</li> <li>• List status values for individual lists (see Chapter 6.4.2 "List Status", page 97) can be queried by <b>read_status</b>.</li> <li>• <b>get_status</b>, <b>get_input_pointer</b> and <b>read_status</b> can be used during loading of a list to ensure that no list is overwritten that has still not been processed (see also the <b>load_list</b> command).</li> <li>• As long as no program has been loaded (by <b>load_program_file</b>), <b>get_status</b> returns undefined values.</li> <li>• <b>get_head_status</b> is available for querying the status signals of the scan heads.</li> </ul>
RTC4→RTC6	<p>Basically unchanged functionality. However:</p> <p>The parameter <code>Status</code> returns additionally the <b>INTERNAL-BUSY</b> and <b>PAUSED</b> status with an RTC6.</p>
RTC5→RTC6	<p>Changed functionality.</p> <ul style="list-style-type: none"> <li>• The <code>HEAD_BUSY</code> status is returned by Bit #23.</li> </ul>
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>read_status</b> , <b>get_input_pointer</b> , <b>get_list_pointer</b> , <b>get_out_pointer</b>



<b>Ctrl Command</b>	<b>get stepper status</b>
<b>Function</b>	Returns the following status information for both stepper motor output ports: the current statuses of the stepper motor signals, the currently defined CLOCK pulse period, the Busy and Init statuses, and the current values of the internal position variables.
<b>Call</b>	get stepper status( &Status1, &Pos1, &Status2, &Pos2 )
<b>Returned parameter values</b>	<p>Status1    Current status of stepper motor output 1.                  As a pointer to an unsigned 32-bit value.</p> <p>Bit #0      ENABLE signal.                  (LSB)</p> <p>Bit #1      DIRECTION signal.</p> <p>Bit #2      CLOCK signal.</p> <p>Bit #3      SWITCH signal = limit switch signal.</p> <p>Bit #4      Status "Busy".</p> <p>Bit #5      Status "Init".</p> <p>Bit #6      Reserved.</p> <p>Bit #7      Reserved.</p> <p>Bit #8      CLOCK pulse period (24-bit value).</p> <p>...</p> <p>Bit #31</p> <p>Pos1      Current value of the internal position variable for stepper motor output port 1.                  As a pointer to a signed 32-bit value.</p> <p>Status2     Current status of stepper motor output port 1.                  Otherwise like &amp;Status1.</p> <p>Pos2      Like Pos1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For programming the stepper motor signals, see <a href="#">Chapter 9.1.5 "Controlling Stepper Motors", page 270</a>.</li> <li>If the SWITCH signal (limit switch signal) is set to (pin is LOW), no more CLOCK pulses are generated.</li> <li>The Busy status indicates that a previously initiated (by <code>stepper_abs</code>, <code>stepper_rel</code>, etc.) set-position movement has not yet completed.</li> <li>The Init status indicates that a previously initiated (by <code>stepper_init</code>) reference movement has not yet completed.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–



<b>Ctrl Command</b>	<b>get_sub_pointer</b>
<b>Function</b>	Returns the absolute start address of an indexed subroutine.
<b>Call</b>	SubPointer = get_sub_pointer( Index )
<b>Parameters</b>	Index      Index of the indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].
<b>Result</b>	Absolute start address. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The <b>get_sub_pointer</b> command reads from the internal management table the start address of the indexed subroutine with the specified index. Whether the read address resides in a protected or the unprotected list memory area "List 3" depends on whether the subroutine was loaded into the protected list memory area "List 3" or an unprotected subroutine was only subsequently referenced.</li> <li>If <code>Index &gt; 1023</code> or if no subroutine was referenced with the specified index, then <b>get_sub_pointer</b> returns the value "-1" (for example, <math>2^{32}-1</math>).</li> <li>This command is useful for checking if a subroutine has already been defined or for calling an indexed subroutine by an absolute memory address as if it were a non-indexed subroutine. Be aware, though, that a subsequent <b>save_disk/load_disk</b> might alter the absolute memory address.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_char_pointer, get_text_table_pointer</b>



<b>Ctrl Command</b>	<b>get_sync_status</b>
<b>Function</b>	Returns the master/slave synchronization information on the addressed RTC6 board.
<b>Call</b>	MasterSlaveSyncStatus = get_sync_status()
<b>Result</b>	<p>Master/slave synchronization information. As an unsigned 32-bit value.</p> <p>Bit #0      Master/slave synchronization status [0...640]. As unsigned 10 bit value. ... Bit #9      &lt; 4: The addressed board is synchronous to the master board (or to its preceding board in the master/slave chain). 640: The addressed board is operated as master.</p> <p>Bit #10     = 0: Bit #0...Bit #9 contain valid values. = 1: Bit #0...Bit #9 do not contain valid values. No external start has been received from the master board yet.</p> <p>Bit #11     Reserved.</p> <p>Bit #12     = 1: A master board has been detected. However, no /Slave-START or /Slave-STOP can be received. See <a href="#">master_slave_config</a>.</p> <p>Bit #13     = 1: A Slave board has been detected. However, no /Slave-START or /Slave-STOP can be received. See <a href="#">master_slave_config</a>.</p> <p>Bit #14     = 1: Since the last call of <b>get_sync_status</b>, the connection to the master has been lost.</p> <p>Bit #15     = 1: Since the last call of <b>get_sync_status</b>, the connection to the slave has been lost.</p> <p>Bit #16     Reserved. ... Bit #20</p> <p>Bit #21     Exact propagation time in 1/64 <math>\mu</math>s clock cycles between two RTC6 boards (outbound and return). As unsigned 10 bit value. ... Bit #30</p> <p>Bit #31     = 1: The propagation time Bit #21...Bit #30) could not be measured successfully.</p>



Ctrl Command	get_sync_status
Comments	<ul style="list-style-type: none"> <li>For usage of <code>get_sync_status</code>, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 114</a>.</li> <li>If the addressed board has not been initialized previously by <code>load_program_file</code>, the <code>get_sync_status</code> returns 0.</li> <li>Prior to <code>get_sync_status</code>, the boards of the master/slave chain should have been synchronized by <code>sync_slaves</code>.</li> <li>As of RBF 619: <ul style="list-style-type: none"> <li>Synchronization is automatic. <code>sync_slaves</code> is no longer required. The master/slave synchronization state is measured automatically.</li> <li>The exact propagation time is returned by Bit #21...Bit #30, if Bit #31 does not indicate an error.</li> </ul> </li> <li>Up to RBF 618: <ul style="list-style-type: none"> <li>To ensure that <code>get_sync_status</code> actually returns the current master/slave synchronization status, you should first have already triggered an external start for the master board by an external start signal or a command, see <a href="#">Section "External Start", page 276</a>. For all slave boards for which external starts and stops are not suppressed (see <code>master_slave_config</code>), this start then triggers a measurement of the time difference between the respective /Slave-START pulse and respective 10 µs clock period. This time difference gets stored on each board as the master/slave synchronization status (in units of 1/64 µs) and remains stored there until the a new external start is triggered for the master board. This gives you the flexibility to query the synchronization status by <code>get_sync_status</code> even at a later point in time.</li> <li>In the synchronized state, measured time differences are shorter than the transit time difference for synchronization between the boards themselves, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 114</a>: master/slave synchronization status &lt; 3.</li> <li>In a not-explicitly-synchronized state (prior to <code>sync_slaves</code>), the master/slave synchronization status might coincidentally be &lt; 3. If so, then the boards behave as if synchronized.</li> <li>If Bit #12 or Bit #13 is set, the affected cards cannot be synchronized with <code>sync_slaves</code>.</li> <li>If Bit #14 or Bit #15 is set, /Slave STARTs or /Slave STOPs may not have been forwarded. In this case it is recommended to adjust <code>master_slave_config</code> and to execute <code>sync_slaves</code> again.</li> <li>If one or more of Bit #12...Bit #15 are sporadically 1, this may indicate electromagnetic disturbances that interfere with communication between the boards.</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Basically unchanged functionality. However, more details are available.
Version info	<p>Available as of version DLL 600, OUT 600, RBF 600.</p> <p>Last change version DLL 614, OUT 614, RBF 619: internal improvements and additional results.</p>
References	<code>sync_slaves</code> , <code>get_master_slave</code> , <code>master_slave_config</code>



<b>Ctrl Command</b>	<b>get_table_para</b>
<b>Function</b>	Returns the value of the specified parameter from a currently loaded correction table.
<b>Call</b>	TablePara = get_table_para( TableNo, ParaNo )
<b>Parameters</b>	TableNo     Number of the currently loaded correction table. As an unsigned 32-bit value. Allowed values: [1...8].
	ParaNo     Number of the parameter. As an unsigned 32-bit value. Allowed values: 0...15. Assignment see <a href="#">Section "ct5 Correction File Header", page 167</a> .
<b>Result</b>	Parameter value, see <a href="#">Section "ct5 Correction File Header", page 167</a> . As a 64-bit IEEE floating point value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The parameter values can be read out by <b>get_table_para</b> from a currently loaded correction table and by <b>get_head_para</b> from an assigned correction table and thus directly incorporated into a user program, see <a href="#">Section "ct5 Correction File Header", page 167</a>.</li> <li>If the parameters <b>TableNo</b> and <b>ParaNo</b> are out of range, then the return value is 0 (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>If no correction table with the specified number has been loaded, then the parameter's value is undefined.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	–



<b>Ctrl Command</b>	<b>get_text_table_pointer</b>
<b>Function</b>	Returns the absolute start address of an indexed text string.
<b>Call</b>	TextTablePointer = get_text_table_pointer( Index )
<b>Parameters</b>	Index      Index of the indexed text string. As an unsigned 32-bit value. Allowed value range: [0...41].
<b>Result</b>	Absolute start address. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The <b>get_text_table_pointer</b> command reads from the internal management table the start address of the indexed text string with the specified index. Whether the read address resides in a protected or the protected list memory area "List 3" depends on whether the text string was loaded into the protected list memory area "List 3" or an unprotected subroutine was only subsequently referenced.</li> <li>If <b>Index &gt; 41</b> or if no text string was referenced with the specified index, then <b>get_text_table_pointer</b> returns the value "-1" (for example, <math>2^{32}-1</math>).</li> <li>This command is useful for checking if a text string has already been defined or for calling an indexed text string by an absolute memory address as if it were a non-indexed subroutine, for example, for conditional execution with <b>list_call_cond</b>. Be aware, though, that a subsequent <b>save_disk/load_disk</b> might alter the absolute memory address. And you should ensure that <b>get_text_table_pointer</b> does not return "-1"; otherwise <b>list_call_cond</b> is ignored.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_char_pointer, get_sub_pointer</b>

<b>Ctrl Command</b>	<b>get_time</b>
<b>Function</b>	Returns the RTC6 timer value (without resetting it to zero). It was stored during the most recent call of <b>save_and_restart_timer</b> .
<b>Call</b>	TimerValue = get_time()
<b>Result</b>	RTC6 timer value in seconds. As a 64-bit IEEE floating point value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <b>save_and_restart_timer</b>.</li> <li>The number of elapsed list-command clock cycles since the reset to zero can be queried by <b>get_lap_time</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_lap_time, save_and_restart_timer</b>



<b>Ctrl Command</b>	<b>get_transform</b>
<b>Function</b>	Transfers to the PC the position values that were recorded by <b>set_trigger</b> and stored on the RTC6, and applies backward transformation to these values.
<b>Call</b>	<code>get_transform( Number, Ptr1, Ptr2, Ptr, Code )</code>
<b>Parameters</b>	<p>Number      Number [1...2<sup>23</sup>] of to-be-backward-transformed position values. As an unsigned 32-bit value. The measured values with indices 0 to (Number-1) are backward transformed.</p> <p>Ptr1      Pointers (in C and C++ data type ULONG_PTR, an unsigned 32-bit or 64-bit value) to the two areas of PC main memory to which the backward transformed values should be transferred (see also <b>Code</b>).</p> <p>Ptr2      Pointers (in C and C++ data type ULONG_PTR, an unsigned 32-bit or 64-bit value) to the area of PC main memory to which the correction and transformation settings for backward transformation were previously transferred by <b>upload_transform</b>.</p> <p>Code      Controls aspects of backward transformation, particularly which partial transformations should be performed: If a partial transformation should <i>not</i> be performed, then its corresponding bit (#2...#5) must be set to 1. This parameter has the same meaning as in <b>transform</b> (Ptr1 corresponds to <b>Sig1</b> and Ptr2 to <b>Sig2</b>). As an unsigned 32-bit value.  For Bit #0 = 0, the measurement value pairs recorded by <b>set_trigger</b> are backward transformed as xy coordinates:  Bit #1      = 0:      Values recorded by measurement channel 1 (Signal1) are transferred to the PC using <b>Ptr1</b> and then backward transformed as x coordinates.                         Values recorded by measurement channel 2 (Signal2) are transferred to the PC using <b>Ptr2</b> and then backward transformed as y coordinates.                          = 1:      Values recorded by measurement channel 1 (Signal1) are transferred to the PC using <b>Ptr1</b> and then backward transformed as y coordinates.                         Values recorded by measurement channel 2 (Signal2) are transferred to the PC using <b>Ptr2</b> and then backward transformed as x coordinates.  Bit #2      = 0:      Gain/offset correction of automatic self-calibration is backward transformed.  Bit #3      = 0:      The image field correction is backward transformed.  Bit #4      = 0:      The offset of the defined coordinate transformation is backward transformed.  Bit #5      = 0:      The total matrix of the defined coordinate transformation is backward transformed.  Bit #6      Reserved. ... Bit #31</p>



Ctrl Command	get_transform																
Parameters (cont'd)	<p>Code      If Bit #0 = 1, then (only) the values recorded with <b>set_trigger</b> by one of the two measurement channels (either channel 1 or 2) are backward transformed as z coordinates:</p> <table> <tr> <td>Bit #1</td> <td>= 0: Values recorded by measurement channel 1 (Signal1) are transferred to the PC using <code>Ptr1</code> and then backward transformed as z coordinates. Values recorded by measurement channel 2 (Signal2) are transferred untransformed to the PC using <code>Ptr2</code>.</td> </tr> <tr> <td>= 1:</td> <td>Values recorded by measurement channel 2 (Signal2) are transferred to the PC using <code>Ptr1</code> and then backward transformed as z coordinates. Values recorded by measurement channel 1 (Signal1) are transferred untransformed to the PC using <code>Ptr2</code>.</td> </tr> <tr> <td>Bit #2</td> <td>= 0: The offset to the focal length defined by <b>set_defocus</b> or <b>set_defocus_list</b> is backward transformed.</td> </tr> <tr> <td>Bit #3</td> <td>= 0: The ABC correction is backward transformed.</td> </tr> <tr> <td>Bit #4</td> <td>= 0: The offset to the z coordinate defined by <b>set_offset_xyz</b> or <b>set_offset_xyz_list</b> is backward transformed.</td> </tr> <tr> <td>Bit #5</td> <td>Reserved.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #31</td> <td></td> </tr> </table>	Bit #1	= 0: Values recorded by measurement channel 1 (Signal1) are transferred to the PC using <code>Ptr1</code> and then backward transformed as z coordinates. Values recorded by measurement channel 2 (Signal2) are transferred untransformed to the PC using <code>Ptr2</code> .	= 1:	Values recorded by measurement channel 2 (Signal2) are transferred to the PC using <code>Ptr1</code> and then backward transformed as z coordinates. Values recorded by measurement channel 1 (Signal1) are transferred untransformed to the PC using <code>Ptr2</code> .	Bit #2	= 0: The offset to the focal length defined by <b>set_defocus</b> or <b>set_defocus_list</b> is backward transformed.	Bit #3	= 0: The ABC correction is backward transformed.	Bit #4	= 0: The offset to the z coordinate defined by <b>set_offset_xyz</b> or <b>set_offset_xyz_list</b> is backward transformed.	Bit #5	Reserved.	...		Bit #31	
Bit #1	= 0: Values recorded by measurement channel 1 (Signal1) are transferred to the PC using <code>Ptr1</code> and then backward transformed as z coordinates. Values recorded by measurement channel 2 (Signal2) are transferred untransformed to the PC using <code>Ptr2</code> .																
= 1:	Values recorded by measurement channel 2 (Signal2) are transferred to the PC using <code>Ptr1</code> and then backward transformed as z coordinates. Values recorded by measurement channel 1 (Signal1) are transferred untransformed to the PC using <code>Ptr2</code> .																
Bit #2	= 0: The offset to the focal length defined by <b>set_defocus</b> or <b>set_defocus_list</b> is backward transformed.																
Bit #3	= 0: The ABC correction is backward transformed.																
Bit #4	= 0: The offset to the z coordinate defined by <b>set_offset_xyz</b> or <b>set_offset_xyz_list</b> is backward transformed.																
Bit #5	Reserved.																
...																	
Bit #31																	
Comments	<ul style="list-style-type: none"> <li>For backward transformation of position values, see <a href="#">Chapter 8.1.3 "Monitoring the Positioning", page 201</a>.</li> <li><code>get_transform(Number, Ptr1, Ptr2, Ptr, Code)</code> transfers to the PC the data pairs recorded by <b>set_trigger</b> by executing <code>get_waveform(1,Number,Ptr1)</code> and <code>get_waveform(2,Number,Ptr2)</code> and overwrites the data pairwise by using <code>transform(Sig1,Sig2,Ptr,Code)</code>.</li> <li>Prior to calling <b>get_transform</b>, you must call <b>upload_transform</b>. Additionally, position values should have been recorded by <b>set_trigger</b>.</li> <li>Before calling <b>get_transform</b> (as with <b>get_waveform</b>), you can check by <b>measurement_status</b> whether a measurement session is currently running that was started with <b>set_trigger</b>. We recommend not to read any data when data recording is currently active. The number <code>Pos</code> for the last (or current) data pair of the measurement session can be queried by <b>measurement_status</b>. No more than <code>Pos+1</code> data elements should be read. Any further elements are from earlier recordings or the initialization.</li> <li>The PC main memory areas pointed to by <code>Ptr1</code> and <code>Ptr2</code> must have been sufficiently allocated by users (<code>Number × 4</code> bytes per channel).</li> <li>If only Z position values are to be backward transformed (Code Bit #0 = 1), then you can set the pointer (<code>Ptr1</code> or <code>Ptr2</code>) for the unused return channel to <b>NULL</b>. This channel does then not transfer and backward transform data to the PC. Ensure here that Code Bit #1 is appropriately set.</li> </ul>																



Ctrl Command	get_transform
Comments (cont'd)	<ul style="list-style-type: none"> <li>If the command call is made with <code>Ptr1 = NULL and Ptr2 = NULL</code>, then neither channel transfers data to the PC and likewise no backward transformations are performed (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>). The same applies for <code>Number = 0</code> or <code>Number &gt; 2<sup>23</sup></code>.</li> <li>If <code>Ptr = NULL</code>, then no backward transformation is performed. The data recorded by <code>set_trigger</code> is then transferred untransformed to the PC, as with <code>get_waveform(1, Number, Ptr1)</code> and <code>get_waveform(2, Number, Ptr2)</code>. The same applies for <code>Ptr ≠ NULL</code> if an error occurred during execution of <code>get_transform</code> (for example, when data referenced by <code>Ptr</code> are invalid or erroneous or when z axis inversion is not possible). Here, however, a <code>get_last_error</code> return code of <code>RTC6_PARAM_ERROR</code> is additionally generated.</li> <li>If needed, the values recorded with <code>set_trigger</code> and backward-transformed transferred to the PC by <code>get_transform</code> can additionally be untransformed transferred to the PC by <code>get_waveform</code>.</li> <li>If backward transformation of Z position values is requested (Code Bit #0 = 1), but only a 2D correction table was assigned at the timepoint of the prior successful call to <code>upload_transform</code>, then the offsets to the focal length and z coordinates are initialized with 0 and the values A, B, C with are initialized 0, 1, 0 (1-to-1 backward transformation).</li> <li>For backward transformation of xy position values (Code Bit #0 = 0), only the Z = 0 plane are transformed. xy stretching and Z defocus due to Z deviations (particularly with non-F-Theta systems) are not taken into account.</li> <li>PCI transmission errors generate the <code>get_last_error</code> return code <code>RTC6_SEND_ERROR</code>.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>Even in the RTC6's <b>RTC4 Compatibility Mode</b>, all coordinate values transferred to the PC are in the 20-bit range and must, when necessary, be divided by 16 by users.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>upload_transform</code> , <code>transform</code> , <code>set_trigger</code> , <code>get_waveform</code>



<b>Ctrl Command</b>	<b>get_value</b>
<b>Function</b>	Returns the current value of the specified signal.
<b>Call</b>	<code>Value = get_value( Signal )</code>
<b>Parameters</b>	<p>Signal      Desired signal type. As an unsigned 32-bit value.</p>
<b>Result</b>	<p>Current value of the specified signal. As a signed 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The selectable signal types are identical to those of the <b>set_trigger</b> command (refer to the comments there for the allowed value range, signal types and other information). If the value for <code>Signal</code> is unallowed, then <b>get_value</b> does not read out a signal and returns 0 (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>To observe the specified signal over a long time period, use <b>set_trigger</b> to start a corresponding measurement session.</li> <li>When using an <i>iDRIVE</i> scan system (see glossary entry on <a href="#">page 879</a>), after a reset or power-up of the scan system, it can take around 5 seconds before valid data starts being returned from the scan system. If the XY2-100 status word was selected for return transmission, then the operational readiness can be checked by monitoring bits 4 and 5: Bit #4 = 1 and Bit #5 = 0 (only) if returned-data transmission is valid.</li> <li>For <code>Signal = 0</code>, <b>get_value</b> returns the current laser status (LASERON signal) even when list execution has already been finished.</li> <li>When you query data returned as status signals from the scan system to the RTC6 (<code>Status&lt;AX...BY&gt;</code>), then be mindful of the returned data type's value range when evaluating it (see the command description of <b>control_command</b>): <ul style="list-style-type: none"> <li>Data types originally generated in the scan system as unsigned 16-bit values (for example, the XY2-100 status word or the serial number) and returned to the RTC6 as unsigned 20-bit values (whereby bits #0...3 = 0) contain the relevant information in bits #4...19. Here, only bits #4...19 should be evaluated (see code example below). For bits #20...31 of this data type, <b>get_value</b> returns to the PC not only zero, but (depending on Bit #19 of the underlying 16-bit status value) even the value one. So only evaluate bits #4...19.</li> <li>In contrast, data types returned to the RTC6 as signed 20-bit values (for example, actual positions or actual speeds) can be evaluated as a complete signed 32-bit value returned by <b>get_value</b> (see also code example below).</li> </ul> </li> </ul>



Ctrl Command	get_value
Example (C/C++)	<p>Querying diverse data types (first scan head connector, x axis):</p> <p>a) XY2-100 status word, PowerOK status</p> <pre>UINT statusword, powerOK; control_command (1, 1, 0x0500); // only applicable for iDRIVE systems statusword = (get_value(1) &amp; 0x000FFFF0) &gt;&gt; 4; powerOK = (statusword &amp; 0x00000080);</pre> <p>b) Serial number (only applicable for iDRIVE systems)</p> <pre>UINT SN_low, SN_high, SN; // the serial number's lower 16 bits are selected for return // and queried by get_value: control_command (1, 1, 0x051E); SN_low = (get_value(1) &amp; 0x000FFFF0)&gt;&gt;4; // the serial number's upper 16 bits are selected for return // and queried by get_value: control_command (1, 1, 0x051F); SN_high = (get_value(1) &amp; 0x000FFFF0)&gt;&gt;4; //Complete serial number: SN = (SN_high &lt;&lt; 16) + SN_low;</pre> <p>c) Actual position (only applicable for iDRIVE systems)</p> <pre>long real_position; control_command (1, 1, 0x0501); real_position = get_value(1);</pre>
RTC4→RTC6	<p>Basically unchanged functionality. However:</p> <p>Even in <b>RTC4 Compatibility Mode</b>, all returned values are in the RTC6's 20-bit range, but get transferred to the PC as 32-bit data (see above).</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_values</a> , <a href="#">set_trigger</a> , <a href="#">get_waveform</a> , <a href="#">get_head_status</a>



<b>Ctrl Command</b>	<b>get_values</b>
<b>Function</b>	Returns the current values of up to four specified signals.
<b>Call</b>	<code>get_values( SignalPtr, ResultPtr )</code>
<b>Parameters</b>	<p>SignalPtr    Pointer (in C and C++ data type <code>ULONG_PTR</code>, an unsigned 32-bit or 64-bit value) to an array of four unsigned 32-bit values, where the to-be-outputted signal types are specified.</p> <p>ResultPtr    Pointer (in C and C++ data type <code>ULONG_PTR</code>, an unsigned 32-bit or 64-bit value) to an array of four signed 32-bit values, where the current values of the up to four specified signals should be stored.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Up to four desired signals can be simultaneously queried. The selectable signal types are identical to those of the <a href="#">set_trigger</a> command (refer to the comments there for the allowed value range, signal types and other information). The to-be-outputted signal types must be specified by <code>SignalPtr</code>. The corresponding signal values are then stored by <code>ResultPtr</code>. For storage of each queried data set, the user program must make available (at the address specified by <code>ResultPtr</code>) <math>4 \times 4</math> bytes of PC memory.</li> <li><code>get_values</code> functions similarly to <a href="#">get_value</a> (see comments there). <code>get_values</code> returns 0 and performs no query on channels for which an invalid signal type was specified by <code>SignalPtr</code>. The <a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code> is only generated if all four specified signal types are invalid.</li> <li>If any of the pointer parameters are <code>NULL</code>, then <code>get_values</code> is not executed (all return values are 0) and a <a href="#">get_last_error</a> return code of <code>RTC6_PARAM_ERROR</code> is generated.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>Even in <a href="#">RTC4 Compatibility Mode</a>, the four returned values are in the RTC6's 20-bit range, but get transferred to the PC as 32-bit data (see comments for <a href="#">get_value</a>).</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_value</a> , <a href="#">set_trigger</a> , <a href="#">transform</a>



Ctrl Command	get_wait_status
Function	Returns the wait state of the RTC6 board.
Call	WaitStatus = get_wait_status()
Result	Wait state. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"><li>If list processing has been stopped at a break point ("wait marker"), then <b>get_wait_status</b> returns the corresponding number. See <b>set_wait</b>.</li><li>If no break point has been encountered, <b>get_wait_status</b> returns the value zero.</li><li>List processing is resumed by calling <b>release_wait</b>.</li></ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_wait, release_wait</b>



<b>Ctrl Command</b>	<b>get_waveform</b>
<b>Function</b>	Transfers to the PC the data that was measured and stored onto the RTC6 by <b>set_trigger</b> or <b>set_trigger4</b> .
<b>Call</b>	<b>get_waveform( Channel, Number, Ptr )</b>
<b>Parameters</b>	Channel      Measurement channel [1 or 2; if recordings started by <b>set_trigger4</b> , then also 3 or 4]; specified. As an unsigned 32-bit value.
	Number      Number [0...max. channel size, see <b>set_trigger4</b> ] of measured values to be transferred. As an unsigned 32-bit value. The measured values from position 0 to (Number-1) are transferred.
	Ptr      Pointer (data type ULONG_PTR in C and C++, an unsigned 32-bit or 64-bit value) to a location in the PC memory to where the measured values should be transferred.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• In the following cases, no data is transferred (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>):             <ul style="list-style-type: none"> <li>– For Number = 0.</li> <li>– For Number &gt; <math>2^{23} \times 4</math> and Channel = 1</li> <li>– For Number &gt; <math>2^{23} \times 2</math> and Channel = 2</li> <li>– For Number &gt; <math>2^{23} \times 3</math> and Channel = 3</li> <li>– For Number &gt; <math>2^{23} \times 1</math> and Channel = 4</li> <li>– For Ptr = <b>NULL</b>.</li> </ul> </li> <li>• PCI transmission errors generate the <b>get_last_error</b> return code <b>RTC6_SEND_ERROR</b>.</li> <li>• Before calling <b>get_waveform</b>, you can check by <b>measurement_status</b> whether a measurement session is currently running that was started with <b>set_trigger</b> or <b>set_trigger4</b>. We recommend not reading any data when data recording is currently active. The number <b>Pos</b> for the last (or current) data pair of the measurement session can be queried by <b>measurement_status</b>. No more than <b>Pos+1</b> data elements should be read. Any further elements are from earlier recordings or the initialization.</li> </ul>
RTC4→RTC6	<p>Basically unchanged functionality. However:</p> <p>Even in <b>RTC4 Compatibility Mode</b>, all values are in the RTC6's 20-bit range, but get transferred to the PC as 32-bit data (see comments for <b>get_value</b>).</p>
RTC5→RTC6	Basically unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_trigger</b> , <b>set_trigger4</b> , <b>get_value</b> , <b>get_values</b> , <b>get_waveform_offset</b>



<b>Ctrl Command</b>	<b>get_waveform_offset</b>
<b>Function</b>	Like <b>get_waveform</b> , but with parameter Offset.
<b>Call</b>	<b>get_waveform_offset( Channel, Offset, Number, Ptr )</b>
<b>Parameters</b>	<p>Channel      Measurement channel [1...4]. As an unsigned 32-bit value.</p> <p>Offset      Start position. As an unsigned 32-bit value. Allowed value range: [0...max. channel size, see <b>set_trigger4</b>].</p> <p>Number     Number of measured values to be transferred. As an unsigned 32-bit value. The measured values from position Offset to (Offset + (Number-1)) are transferred.</p> <p>Ptr        Pointer (data type ULONG_PTR in C and C++, an unsigned 32-bit or 64-bit value) to a location in the PC memory to where the measured values should be transferred.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>get_waveform</b> is synonymous with <b>get_waveform_offset( Channel, 0, Number, Ptr )</b>.</li> <li>• In the following cases, no data is transferred (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>): <ul style="list-style-type: none"> <li>– For Number = 0.</li> <li>– For Offset + Number &gt; max. channel size.</li> <li>– For Ptr = <b>NULL</b>.</li> </ul> </li> <li>• PCI transmission errors generate the <b>get_last_error</b> return code <b>RTC6_SEND_ERROR</b>.</li> <li>• See also <b>set_trigger/set_trigger4</b> (configuration of the channel).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 610, OUT 610, RBF 614.
References	<b>set_trigger</b> , <b>set_trigger4</b> , <b>get_value</b> , <b>get_values</b>



<b>Ctrl Command</b>	<b>get_z_distance</b>
<b>Function</b>	Returns the focus length value $l$ for the specified point within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> has not been enabled or if no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>get_z_distance</b> returns 0 and otherwise has no effect.
<b>Call</b>	$\text{ZDistance} = \text{get\_z\_distance}( \text{X}, \text{Y}, \text{Z} )$
<b>Parameters</b>	<p><b>X</b>      Absolute coordinates of the point <math>(x y z)</math> in the <b>3D image field</b>. In bits. As a signed 32-bit value. Allowed value range: For <math>x, y</math> and <math>z</math> <math>[-524,288\dots+524,287]</math>. Out-of-range values are clipped to the boundary values.</p> <p><b>Y</b>      Like <b>X</b> (analogously).</p> <p><b>Z</b>      Like <b>X</b> (analogously).</p>
<b>Result</b>	Focus length value. As a signed 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>get_z_distance</b> is only needed for re-calibrating the z axis in a 3-axis scan system, see <b>Section "Checking the z axis Calibration", page 159</b>. The (unit-free) focus length value <math>l</math> corresponds to the focus length difference between the specified point <math>(x y z)</math> and the point <math>(0 0 0)</math>. The focus length value can be positive or negative.</li> <li>• <b>get_z_distance</b> first performs a (virtual) jump to the point <math>(x y z)</math> and then returns the focus length value.</li> <li>• If a list is currently executed, then <b>get_z_distance</b> has no effect and returns 0 (<b>get_last_error</b> return code <b>RTC6_BUSY</b>).</li> <li>• <b>get_z_distance</b> is not executed and returns 0 (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>get_z_distance</b> is even executed, if: <ul style="list-style-type: none"> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• For 3D image field calibration, see <b>Chapter "3D Commands", page 223</b>.</li> </ul>
<b>RTC4→RTC6</b>	Unchanged functionality. In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the values specified for $x, y$ and $z$ by 16. The allowed value range decreases accordingly.
<b>RTC5→RTC6</b>	Unchanged functionality. In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the value specified for $z$ by 16. The allowed value range decreases accordingly.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>load_z_table</b>



<b>Ctrl Command</b>	<b>goto_xy</b>				
<b>Function</b>	Moves the output point (of the laser focus) along a 2D vector at jump speed from the current position to the specified position (absolute coordinate values) within a 2D image field.				
<b>Call</b>	<code>goto_xy( X, Y )</code>				
<b>Parameters</b>	<table border="1"> <tr> <td>X</td><td>Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td>Y</td><td>Like X (analogously).</td></tr> </table>	X	Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.	Y	Like X (analogously).
X	Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.				
Y	Like X (analogously).				
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the jump speed has not been previously explicitly set by <a href="#">set_jump_speed</a> or <a href="#">set_jump_speed_ctrl</a>, then the jump is executed at a predefined jump speed of 10000 bits/ms.</li> <li><b>goto_xy</b> (unlike the list commands <a href="#">jump_abs</a> and <a href="#">jump_rel</a>) has no effect on the laser control signals and also does not set a jump delay.</li> <li>Previously accumulated coordinate transformations become effective by <b>goto_xy</b>, see list item "With <code>at_once = 2 ...</code>", <a href="#">page 212</a>.</li> <li><b>goto_xy</b> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC6_BUSY</a>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li> <li>the <b>INTERNAL-BUSY</b> status of the board is set</li> <li>an external stop signal (/STOP, /STOP2 or /Slave STOP) is present.</li> </ul> It can also be generated by automatic monitoring, see <a href="#">set_laser_control</a> and <a href="#">range_checking</a>.</li> <li><b>goto_xy</b> is even executed, if: <ul style="list-style-type: none"> <li>a list has been paused by <a href="#">set_wait</a> (<b>PAUSED</b> status set)</li> </ul> </li> <li>The <b>INTERNAL-BUSY</b> status is set while <b>goto_xy</b> is executed.</li> <li><b>goto_xy</b> only returns to the user program when the movement has been completed.</li> </ul>				
RTC4→RTC6	<ul style="list-style-type: none"> <li>Increased value range.</li> <li><b>goto_xy</b> returns only after movement has been executed (see comments above).</li> <li>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified values for X and Y by 16. The allowed value range decreases accordingly.</li> </ul>				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">set_jump_speed</a> , <a href="#">jump_abs</a> , <a href="#">jump_rel</a> , <a href="#">goto_xyz</a> , <a href="#">get_status</a>				



<b>Ctrl Command</b>	<b>goto_xyz</b>						
<b>Function</b>	Moves the output point (of the laser focus) along a 3D vector at jump speed from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> .						
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>goto_xyz</b> has the same effect as <b>goto_xy</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the <b>xy</b> plane.						
<b>Call</b>	<b>goto_xyz( X, Y, Z )</b>						
<b>Parameters</b>	<table> <tr> <td>X</td> <td>Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</td> </tr> <tr> <td>Y</td> <td>Like X (analogously).</td> </tr> <tr> <td>Z</td> <td>Like X (analogously).</td> </tr> </table>	X	Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.	Y	Like X (analogously).	Z	Like X (analogously).
X	Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.						
Y	Like X (analogously).						
Z	Like X (analogously).						
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>goto_xyz</b> functions similarly to <b>goto_xy</b> (see comments there).</li> <li>The <b>DirectMove3D</b> parameter of <b>set_delay_mode</b> determines the type of z axis motion (linear or with stepwise correction).</li> <li>During calculation of the Z output value to the scan system, any previously defined offset to the z coordinate and focal length is taken into account, see <b>Chapter 7.3.6 "Output Values to the Scan System"</b>, page 170.</li> </ul>						
<b>RTC4→RTC6</b>	<ul style="list-style-type: none"> <li>Increased value range.</li> <li><b>goto_xyz</b> returns only after the motion has completed (see comments for <b>goto_xy</b>).</li> <li><b>RTC4 Compatibility Mode</b>: see <b>goto_xy</b>.</li> </ul>						
<b>RTC5→RTC6</b>	Unchanged functionality.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the values specified for Z by 16. The allowed value range decreases accordingly.						
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.						
<b>References</b>	<b>goto_xy, jump_abs_3d, jump_rel_3d</b>						



<b>Ctrl Command</b>	<b>home_position</b>
<b>Function</b>	Activates the home jump mode (for the x and y axes) and defines the home position.
<b>Call</b>	home_position( XHome, YHome )
<b>Parameters</b>	XHome      Absolute x coordinate of the home position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Larger values are clipped.
	YHome      Like XHome (analogously).
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>home_position</b> defines the coordinates of a home jump to be executed, for example, upon reaching the end of a list. Analogously, a list starts with a home return to the most recently valid position. Home jump and home return are executed at jump speed.</li> <li>• <b>home_position</b> is intended for laser systems that do not allow fast switching of the laser. After calling <b>home_position</b>, the laser focus moves to the specified home position whenever no list is executing or when a list has been paused by <b>set_wait</b>. A home jump is also executed, if list execution is stopped by <b>stop_execution</b> or by an external stop signal. The home jump itself (in contrary to a home return) cannot be stopped.</li> <li>• While a home jump or a home return is executed, the <b>INTERNAL-BUSY</b> status is set.</li> <li>• A <i>beam dump</i> should be placed in the home position.</li> <li>• The home jump mode is deactivated by <b>home_position(0, 0)</b>, even if it was activated by <b>home_position_xyz</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for XHome and YHome by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">home_position_xyz</a>



<b>Ctrl Command</b>	<b>home_position_xyz</b>						
<b>Function</b>	Activates the home jump mode (for the x, y and z axis) and defines the home position.						
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>home_position_xyz</b> has the same effect as <b>home_position</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.						
<b>Call</b>	<code>home_position_xyz( XHome, YHome, ZHome )</code>						
<b>Parameters</b>	<table> <tr> <td>XHome</td> <td>Absolute x coordinate of the home position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</td> </tr> <tr> <td>YHome</td> <td>Like XHome (analogously).</td> </tr> <tr> <td>ZHome</td> <td>Like XHome (analogously).</td> </tr> </table>	XHome	Absolute x coordinate of the home position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.	YHome	Like XHome (analogously).	ZHome	Like XHome (analogously).
XHome	Absolute x coordinate of the home position. In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.						
YHome	Like XHome (analogously).						
ZHome	Like XHome (analogously).						
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>home_position_xyz</b> functions similarly to <b>home_position</b> (see comments there).</li> <li>The home jump mode is deactivated by: <ul style="list-style-type: none"> <li>- <code>home_position( 0, 0 )</code></li> <li>- <code>home_position_xyz( 0, 0, 0 )</code></li> </ul> </li> <li>The x and y axes can be controlled with 20-bit resolution, the z axis with 16-bit resolution. The RTC6 (in <b>RTC6 Standard Mode</b> as well as in <b>RTC4 Compatibility Mode</b>) automatically upscales z coordinate values internally to 20-bit values, so that the three dimensions of space can be handled equivalently in terms of the supplied jump speed value.</li> <li>The <b>DirectMove3D</b> parameter of <b>set_delay_mode</b> determines the type of z axis motion (linear or with stepwise correction).</li> </ul>						
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for XHome, YHome and ZHome by 16. The allowed value range decreases accordingly.						
RTC5→RTC6	Unchanged functionality.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the specified value for ZHome by 16. The allowed value range decreases accordingly.						
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.						
<b>References</b>	<b>home_position</b>						



<b>Undelayed Short List Command</b>	<b>if_cond</b>
<b>Function</b>	<i>Conditional command execution:</i> <b>if_cond</b> immediately executes the directly following list command, if the current IOvalue at the EXTENSION 1 socket connector's 16-bit digital input port meets the following condition: $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ (= if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0). Otherwise, the next list command is skipped.
<b>Call</b>	<b>if_cond( Mask1, Mask0 )</b>
<b>Parameters</b>	<p>Mask1      16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</p> <p>Mask0      See Mask1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">if_not_cond</a> , <a href="#">if_pin_cond</a> , <a href="#">if_not_pin_cond</a>

<b>Undelayed Short List Command</b>	<b>if_fly_x_overflow</b>
<b>Function</b>	<i>Conditional command execution for Processing-on-the-fly applications:</i> <b>if_fly_x_overflow</b> immediately executes the directly subsequent list command, if the condition in accordance with Mode for the x axis was fulfilled. Otherwise, the next list command is skipped.
<b>Call</b>	<b>if_fly_x_overflow( Mode )</b>
<b>Parameters</b>	<p>Mode      To-be-evaluated condition. As a signed 32-bit value.</p> <p>= 0:      Some kind of boundary exceedance occurred (error Bit #4 = 1 or error Bit #5 = 1).</p> <p>&gt; 0:      An overflow occurred (error Bit #5 = 1).</p> <p>&lt; 0:      An underflow occurred (error Bit #4 = 1).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For usage of <b>if_fly_x_overflow</b>, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>• The error bits queried in accordance with Mode (error Bit #4 and/or error Bit #5) are reset.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">if_not_fly_x_overflow</a>

<b>Undelayed Short List Command</b>	<b>if_fly_y_overflow</b>
<b>Function</b>	<i>Conditional command execution for Processing-on-the-fly applications:</i> if_fly_y_overflow immediately executes the directly subsequent list command, if the condition in accordance with Mode for the y axis was fulfilled. Otherwise, the next list command is skipped.
<b>Call</b>	if_fly_x_overflow( Mode )
<b>Parameters</b>	Mode      To-be-evaluated condition. As a signed 32-bit value.  = 0:    Some kind of boundary exceedance occurred (error Bit #6 = 1 or error Bit #7 = 1). > 0:    An overflow occurred (error Bit #7 = 1). < 0:    An underflow occurred (error Bit #6 = 1).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of if_fly_y_overflow, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>The error bits queried in accordance with Mode (error Bit #6 and/or error Bit #7) are reset.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">if_not_fly_y_overflow</a>

<b>Undelayed Short List Command</b>	<b>if_fly_z_overflow</b>
<b>Function</b>	<i>Conditional command execution for Processing-on-the-fly applications:</i> if_fly_z_overflow immediately executes the directly subsequent list command, if the condition in accordance with Mode for the z axis was fulfilled. Otherwise, the next list command is skipped.
<b>Call</b>	if_fly_z_overflow( Mode )
<b>Parameters</b>	Mode      To-be-evaluated condition. As a signed 32-bit value.  = 0:    Some kind of boundary exceedance occurred (error Bit #24 = 1 or error Bit #25 = 1). > 0:    An overflow occurred (error Bit #25 = 1). < 0:    An underflow occurred (error Bit #24 = 1).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of if_fly_z_overflow, see also <a href="#">Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections", page 240</a>.</li> <li>The error bits queried in accordance with Mode (error Bit #24 and/or error Bit #25) are reset.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits_z</a> , <a href="#">clear_fly_overflow</a> , <a href="#">if_not_fly_z_overflow</a>



<b>Undelayed Short List Command</b>	<b>if_not_activated</b>
<b>Function</b>	Conditional command execution due to an error bit from <b>activate_fly_xy/activate_fly_xy_encoder</b> or <b>activate_fly_2d/activate_fly_2d_encoder</b> : If the error bit is set, then the next list command executes immediately, otherwise it is skipped.
<b>Call</b>	<code>if_not_activated()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also comments at <b>activate_fly_2d/activate_fly_2d_encoder</b> and <b>activate_fly_xy/activate_fly_xy_encoder</b>.</li> <li>It is useful to insert a list jump or subroutine call in the list directly after <b>if_not_activated</b> that jumps to an error-handling sequence. Or you could simply insert a <b>set_end_of_list</b>.</li> <li><b>if_not_activated</b> resets the error bit from <b>activate_fly_xy/activate_fly_xy_encoder</b> or <b>activate_fly_2d/activate_fly_2d_encoder</b>. If you still need it for subsequent querying by <code>get_marking_info(Bit #9)</code>, then you can include a renewed call of <b>activate_fly_2d/activate_fly_2d_encoder</b> or <b>activate_fly_xy</b> in your error-handling sequence to set the error bit again (provided that the error situation still exists).</li> <li>Successful activation by <b>activate_fly_2d/activate_fly_2d_encoder</b> or <b>activate_fly_xy/activate_fly_xy_encoder</b> does not reset any already-set error bit. It remains set for <b>get_marking_info</b> until <b>get_marking_info</b> is called.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>activate_fly_2d, activate_fly_2d_encoder, activate_fly_xy, activate_fly_xy_encoder, get_marking_info</b>



<b>Undelayed Short List Command</b>	<b>if_not_cond</b>
Function	<p><i>Conditional command execution:</i> <b>if_not_cond</b> immediately executes the directly following list command, if the current IOvalue at the EXTENSION 1 socket connector's 16-bit digital input port <i>does not meet</i> the following condition:</p> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ <p>(= if the bits specified in Mask1 are <i>not 1</i> or the bits specified in Mask0 are <i>not 0</i>). Otherwise, the next list command is skipped.</p>
Call	<code>if_not_cond( Mask1, Mask0 )</code>
Parameters	<p>Mask1      16-bit mask.  As an unsigned 32-bit value.  Only the lower 16 bits are evaluated.</p> <p>Mask0      See Mask1.</p>
Comments	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">if_cond</a> , <a href="#">if_pin_cond</a> , <a href="#">if_not_pin_cond</a>



<b>Undelayed Short List Command</b>	<b>if_not_fly_x_overflow</b>
<b>Function</b>	<i>Conditional command execution for Processing-on-the-fly applications:</i> <b>if_not_fly_x_overflow</b> immediately executes the directly subsequent list command, if the condition in accordance with <code>Mode</code> for the x axis is <i>not</i> fulfilled. Otherwise, the next list command is skipped.
<b>Call</b>	<code>if_not_fly_x_overflow( Mode )</code>
<b>Parameters</b>	<code>Mode</code> To-be-evaluated condition. As a signed 32-bit value. = 0: Some kind of boundary exceedance occurred (error Bit #4 = 1 or error Bit #5 = 1). > 0: An overflow occurred (error Bit #5 = 1). < 0: An underflow occurred (error Bit #4 = 1).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>if_not_fly_x_overflow</b>, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>The error bits queried in accordance with <code>Mode</code> (error Bit #4 and/or error Bit #5) are reset.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">if_fly_x_overflow</a>

<b>Undelayed Short List Command</b>	<b>if_not_fly_y_overflow</b>
<b>Function</b>	<i>Conditional command execution for Processing-on-the-fly applications:</i> <b>if_not_fly_y_overflow</b> immediately executes the directly subsequent list command, if the condition in accordance with <code>Mode</code> for the y axis is <i>not</i> fulfilled. Otherwise, the next list command is skipped.
<b>Call</b>	<code>if_not_fly_y_overflow( Mode )</code>
<b>Parameters</b>	<code>Mode</code> To-be-evaluated condition. As a signed 32-bit value. = 0: Some kind of boundary exceedance occurred (error Bit #6 = 1 or error Bit #7 = 1). > 0: An overflow occurred (error Bit #7 = 1). < 0: An underflow occurred (error Bit #6 = 1).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>if_not_fly_y_overflow</b>, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>The error bits queried in accordance with <code>Mode</code> (error Bit #6 and/or error Bit #7) are reset.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">if_fly_y_overflow</a>



<b>Undelayed Short List Command</b>	<b>if_not_fly_z_overflow</b>
<b>Function</b>	<i>Conditional command execution for Processing-on-the-fly applications:</i> <b>if_not_fly_z_overflow</b> immediately executes the directly subsequent list command, if the condition in accordance with <b>Mode</b> for the z axis was <b>not</b> fulfilled. Otherwise, the next list command is skipped.
<b>Call</b>	<b>if_not_fly_z_overflow( Mode )</b>
<b>Parameters</b>	<b>Mode</b> To-be-evaluated condition. As a signed 32-bit value. = 0: Some kind of boundary exceedance occurred (error Bit #24 = 1 or error Bit #25 = 1). > 0: An overflow occurred (error Bit #25 = 1). < 0: An underflow occurred (error Bit #24 = 1).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>if_not_fly_z_overflow</b>, see also <a href="#">Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections"</a>, page 240.</li> <li>The error bits queried in accordance with <b>Mode</b> (error Bit #24 and/or error Bit #25) are reset.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits_z</a> , <a href="#">clear_fly_overflow</a> , <a href="#">if_fly_z_overflow</a>

<b>Undelayed Short List Command</b>	<b>if_not_pin_cond</b>
<b>Function</b>	<i>Conditional command execution:</i> <b>if_not_pin_cond</b> immediately executes the directly following list command, if the current <b>I0value</b> at the LASER connector's 2-bit digital input port <i>does not meet</i> the following condition:  $((I0value \text{ AND } Mask1) = Mask1) \text{ AND } (((\text{not } I0value) \text{ AND } Mask0) = Mask0)$ (= if the bits specified in <b>Mask1</b> are <b>not 1</b> or the bits specified in <b>Mask0</b> are <b>not 0</b> ). Otherwise, the next list command is skipped.
<b>Call</b>	<b>if_not_pin_cond( Mask1, Mask0 )</b>
<b>Parameters</b>	<b>Mask1</b> 2-bit mask. As an unsigned 32-bit value. Only the lower 2 bits are evaluated.  <b>Mask0</b> See <b>Mask1</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 9.3.2 "Conditional Command Execution"</a>, page 281.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">if_pin_cond</a> , <a href="#">if_cond</a> , <a href="#">if_not_cond</a>



<b>Undelayed Short List Command</b>	<b>if_pin_cond</b>
Function	<p><i>Conditional command execution:</i> <b>if_pin_cond</b> immediately executes the directly following list command, if the current <code>IOvalue</code> at the LASER connector's 2-bit digital input port meets the following condition:</p> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ <p>(= if the bits specified in <code>Mask1</code> are 1 and the bits specified in <code>Mask0</code> are 0). Otherwise, the next list command is skipped.</p>
Call	<code>if_pin_cond( Mask1, Mask0 )</code>
Parameters	<p><code>Mask1</code>      2-bit mask.            As an unsigned 32-bit value.            Only the lower 2 bits are evaluated.</p> <p><code>Mask0</code>      See <code>Mask1</code>.</p>
Comments	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">if_not_pin_cond</a> , <a href="#">if_cond</a> , <a href="#">if_not_cond</a>



<b>Ctrl Command</b>	<b>init_fly_2d</b>
<b>Function</b>	Initializes the encoder reference values for 2D encoder compensation of a subsequent <b>set_fly_2d</b> Processing-on-the-fly application and resets both encoder values.
<b>Call</b>	<code>init_fly_2d( OffsetX, OffsetY, No )</code>
<b>Parameters</b>	<p>OffsetX      Reference value of the x axis encoder. As a signed 32-bit value. Allowed value range: depends on the compensation table loaded with <b>load_fly_2d_table</b>. Default values (after <b>load_program_file</b>): = 0.</p> <p>OffsetY      Like OffsetX (analogously).</p> <p>No            Number of the 2D compensation table to be used. See also <a href="#">page 235</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The final encoder value for the 2D correction value + reference value must not exceed the allowed range. Otherwise clipping occurs (see also <b>load_fly_2d_table</b>).</li> <li>No = 1: use table 1.</li> <li>No = 2: use table 2.</li> <li>No = 0: use the table that has been used up to now</li> <li>No &gt; 2: do not use a table.</li> <li>The tables to be used must have been validly loaded by <b>load_fly_2d_table</b> beforehand.</li> <li>With No &gt; 2, the table remains validly loaded.</li> <li>By <b>load_fly_2d_table( Name = NULL )</b>, the table becomes invalid.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Changed functionality. <ul style="list-style-type: none"> <li>Parameter No.</li> </ul>
<b>Version info</b>	Available as of version DLL 614, OUT 614, RBF 619.
<b>References</b>	<b>set_fly_2d</b> , <b>load_fly_2d_table</b> , <b>get_fly_2d_offset</b>



<b>Ctrl Command</b>	<b>init_rtc6_dll</b>
<b>Function</b>	Initializes control of the installed RTC6 boards for a user program.
<b>Call</b>	<code>InitErrorNo = init_rtc6_dll()</code>
<b>Result</b>	Error code. As an unsigned 32-bit value. If multiple errors occurred simultaneously, then multiple bits are set. The assignment between bit numbers, error types and error constants is identical to those for <a href="#">get_error</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>init_rtc6_dll</code> must be called by each user program at the beginning, so that an RTC6 board can be addressed by the user program at all.</li> <li>• <code>init_rtc6_dll</code> searches for all present RTC6 PCIe Boards (not for RTC6 Ethernet Boards! – see <a href="#">Chapter 15.5.2 "About Searching RTC6 Ethernet Boards", page 855</a> on this topic) and establishes the RTC6 board management. If no RTC6 board is found (for example, when only RTC6 Ethernet Boards are operated), then error code <code>RTC6_NO_PCIE_CARD_FOUND</code> is returned (see <a href="#">get_error</a>).</li> <li>• If an error occurs when reading the PCI configuration register (<code>get_last_error</code> return code <code>RTC6_CONFIG_ERROR</code>), then an <code>RTC6_ACCESS_DENIED</code> error is also generated. The board then remains inaccessible until the error gets cleared by a PC restart.</li> <li>• <code>init_rtc6_dll</code> automatically assigns the user program access rights to the found boards (as by <a href="#">acquire_rtc</a>), if access rights are not already assigned to another user program (any number of boards and applications may be used, but each particular board cannot be used simultaneously by multiple applications). The first initialization acquires all found boards for itself. Subsequent initializations started by other applications result in return of an <code>RTC6_ACCESS_DENIED</code> error code by <code>init_rtc6_dll</code>. The boards are then only accessible by these applications through RTC6 DLL-internal functions that require no access rights – for example, <a href="#">get_error</a>, <a href="#">get_last_error</a> or <a href="#">select_rtc</a> (most multi-board commands, too, are only callable for boards with existing access rights). If a user program has access rights for a board, then the user program must first explicitly release its access rights with <a href="#">release_rtc</a> or <a href="#">free_rtc6_dll</a> before the board can be used by another user program by <code>init_rtc6_dll</code> or <a href="#">acquire_rtc</a>. An <code>RTC6_ACCESS_DENIED</code> error code is returned if access was denied by at least one of the found boards. Which board(s) denied access can be determined by <code>n_get_error( CardNo )</code> (CardNo from 1 to the number of found boards) or directly after <code>init_rtc6_dll</code> (before the next command) by <code>n_get_last_error( CardNo )</code>.</li> <li>• If a board is acquired by <code>init_rtc6_dll</code> (as by <a href="#">acquire_rtc</a>), then a version compatibility check is performed. If a version error is detected, then access to the board is denied (return code <code>RTC6_ACCESS_DENIED RTC6_VERSION_MISMATCH</code>).</li> <li>• Only one user program can perform initialization at any one time. Subsequent initializations started by other applications wait until the current initialization is complete.</li> </ul>



Ctrl Command	init_RTC6_dll
Comments (cont'd)	<ul style="list-style-type: none"> <li>If a user program calls <code>init_RTC6_dll</code> multiple times, then the RTC6 board management created by the prior call is deleted for this user program and the originally-assigned access rights canceled. The RTC6 board management is then newly created and access rights are newly granted again.</li> <li>Each initialization of a user program by <code>init_RTC6_dll</code> causes the <b>RTC6 DLL</b>-internal numbers for all found RTC6 boards to be newly reassigned. The relationship between the <b>RTC6 DLL</b>-internal numbers and the installed boards can be determined by <code>get_serial_number</code>. When the accessible board with the smallest <b>RTC6 DLL</b>-internal number is initialized, it then simultaneously becomes the active board and the target of non-multi-board commands. <code>select_RTC</code> can be called at anytime to change the active board. If no access rights exist for any board, then the board with the highest <b>RTC6 DLL</b>-internal number (see <code>rtc6_count_cards</code>) is the active board, in which case only <b>RTC6 DLL</b>-internal commands that require no access rights can be used.</li> <li>Also observe <b>Chapter 6.7.1 "Notes on Board Acquisition by a User Program"</b>, page 118.</li> <li><code>init_RTC6_dll</code> is available even without explicit access rights to any particular RTC6 board.</li> <li><code>init_RTC6_dll</code> is not available as a multi-board command.</li> <li>After initialization of the <b>RTC6 DLL</b> with <code>init_RTC6_dll</code>, the <b>RTC6 Standard Mode</b> is set by default. After that, a different <b>RTC6 DLL</b> operational mode can be set, see <code>set_RTC4_mode</code> and <code>set_RTC5_mode</code>.</li> <li><code>init_RTC6_dll</code> does not trigger an initialization of RTC6 boards. Only <code>load_program_file</code> performs an initialization of RTC6 boards.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command. The functionalities of <code>init_RTC6_dll</code> and the RTC5 command <code>init_RTC5_dll</code> are identical.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>set_RTC4_mode</code> , <code>set_RTC5_mode</code>



<b>Normal List Command</b>	<b>jump_abs</b>
<b>Function</b>	Moves the output point (of the laser focus) along a 2D vector at jump speed from the current position to the specified position (absolute coordinate values) within a 2D image field.
<b>Call</b>	<code>jump_abs( X, Y )</code>
<b>Parameters</b>	<p>X      Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the jump speed has not been previously explicitly set by <a href="#">set_jump_speed</a> or <a href="#">set_jump_speed_ctrl</a>, then the jump is executed at a predefined jump speed of 10000 bits/ms.</li> <li>The signals for "laser active" operation are switched off before the jump and remain off during the jump.</li> <li>After a jump command, a (variable) jump delay is inserted. Exception: a zero-length jump vector's subsequent (variable) jump delay is not executed. However, <b>jump_abs</b> itself still requires a 10 µs clock period for execution.</li> <li>Previously accumulated coordinate transformations become effective by <b>jump_abs</b>, see list item "With <code>at_once = 2 ...</code>", <a href="#">page 212</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the values specified for X and Y by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_jump_speed</a> , <a href="#">set_scanner_delays</a> , <a href="#">jump_rel</a> , <a href="#">timed_jump_abs</a> , <a href="#">goto_xy</a>

<b>Normal List Command</b>	<b>jump_abs_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) along a 3D vector at jump speed from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>jump_abs_3d</b> has the same effect as <b>jump_abs</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
<b>Call</b>	<b>jump_abs_3d( X, Y, Z )</b>
<b>Parameters</b>	<p>X      Absolute x coordinate of the jump vector end point. In bits.  As a signed 32-bit value.  Allowed value range: [-268,435,456...+268,435,455].  Out-of-range values are clipped to the boundary values.  The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p> <p>Z      Like X (analogously), except  Allowed value range: [-524,288...+524,287].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>jump_abs_3d</b> functions similarly to <b>jump_abs</b> (see comments there).</li> <li>The <b>DirectMove3D</b> parameter of <b>set_delay_mode</b> determines the type of z axis motion (linear or with stepwise correction).</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the values specified for X, Y and Z by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality. In addition: increased value range.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the value specified for Z by 16. The allowed value range decreases accordingly.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>jump_abs, jump_rel_3d, timed_jump_abs_3d</b>



<b>Normal List Command</b>	<b>jump_rel</b>
<b>Function</b>	Moves the output point (of the laser focus) along a 2D vector at jump speed from the current position to the specified position (relative coordinate values) within a 2D image field.
<b>Call</b>	<code>jump_rel( dx, dy )</code>
<b>Parameters</b>	<p><code>dx</code>      Relative coordinates of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field, for example, for (enabled) Processing-on-the-fly applications.</p> <p><code>dy</code>      Like <code>dx</code> (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <code>jump_rel</code> is identical to <code>jump_abs</code> (see comments there).</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for <code>dx</code> and <code>dy</code> by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">jump_abs</a> , <a href="#">jump_rel_3d</a> , <a href="#">timed_jump_rel</a>



<b>Normal List Command</b>	<b>jump_rel_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) along a 3D vector at jump speed from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>jump_rel_3d</b> has the same effect as <b>jump_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
<b>Call</b>	<b>jump_rel_3d( dx, dy, dz )</b>
<b>Parameters</b>	<p><b>dx</b>      Relative x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>dy</b>      Like <b>dx</b> (analogously).</p> <p><b>dz</b>      Like <b>dx</b> (analogously), except               Allowed value range: [-524,288...+524,287].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>jump_rel_3d</b> is identical to <b>jump_abs_3d</b> (see comments there).</li> </ul>
<b>RTC4→RTC6</b>	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for <b>dx</b> , <b>dy</b> and <b>dz</b> by 16. The allowed value range decreases accordingly.
<b>RTC5→RTC6</b>	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>jump_abs_3d</b> , <b>jump_abs</b> , <b>jump_rel</b> , <b>timed_jump_rel_3d</b>



<b>Variable List Command</b>	<b>laser_on_list</b>
<b>Function</b>	Turns on the “laser active” laser control signals for a specified time interval.
<b>Call</b>	<code>laser_on_list( Period )</code>
<b>Parameters</b>	<p>Period      Time interval. In bits.                    As an unsigned 32-bit value.  <i>1 bit equals 10 µs.</i>                    Allowed value range: <math>0 \leq \text{Period} \leq (2^{31}-1)</math>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Bit #31 of Period is ignored (see also “Comments” and “Version info” at <a href="#">laser_on_pulses_list</a>).</li> <li>The signals for “laser active” operation must first be selected with <a href="#">set_laser_mode</a>, defined by further commands, and enabled by <a href="#">set_laser_control</a> or <a href="#">enable_laser</a> before they can be switched on with <a href="#">laser_on_list</a>, see <a href="#">Chapter 7.4 “Laser Control”, page 173</a>.</li> <li>While the laser control signals are turned on, the set position of the scanners is not changed. The next list command is executed when the programmed time interval has passed.</li> <li>The currently set LaserOn delay is applied at the beginning of the programmed time interval: The laser control signals turn on after a LaserOn delay.       <ul style="list-style-type: none"> <li>As with other <a href="#">[*]mark[*]</a> commands (for example, <a href="#">mark_abs</a>), the laser control signals do <i>not</i> explicitly turn off at the end of the time interval, but instead only turn off with a subsequent list command (for example, <a href="#">jump_abs</a> or <a href="#">list_nop</a>). The currently set LaserOff delay is applied.</li> </ul> </li> <li><a href="#">laser_on_list</a> is useful for marking single points, see <a href="#">Chapter 7.1.3 “Marking Single Points”, page 130</a>.</li> <li>The Wobbel Mode, see <a href="#">Chapter 8.4 “Wobbel Mode”, page 218</a>, is retained but ignored.</li> <li>For <code>Period = 0</code> the laser control signals do not turn on. <a href="#">laser_on_list</a> is a short list command then.</li> <li><a href="#">laser_on_list</a> does not trigger a scanner delay. To wait until the laser (after a LaserOff delay) is actually off before a jump, then explicitly a <a href="#">long_delay</a> should be inserted.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">laser_on_pulses_list</a> , <a href="#">long_delay</a> , <a href="#">para_laser_on_pulses_list</a>



<b>Variable List Command</b>	<b>laser_on_pulses_list</b>				
<b>Function</b>	Turns on the LASERON laser control signal for the specified number of external signal pulses, but for no longer than the specified time interval. For Pulses > 65,535 the function of <b>laser_on_pulses_list</b> is identical to <b>laser_on_list</b> .				
<b>Call</b>	<code>laser_on_pulses_list( Period, Pulses )</code>				
<b>Parameters</b>	<table> <tr> <td>Period</td> <td>Time interval. In bits. As an unsigned 32-bit value. <i>1 bit equals 10 µs.</i> Allowed value range: <math>0 \leq \text{Period} \leq (2^{32}-1)</math>.</td> </tr> <tr> <td>Pulses</td> <td>Number of external signal pulses. As an unsigned 32-bit value. Allowed value range: <math>0 \leq \text{Pulses} \leq 65,535</math> or larger (see comments below).</td> </tr> </table>	Period	Time interval. In bits. As an unsigned 32-bit value. <i>1 bit equals 10 µs.</i> Allowed value range: $0 \leq \text{Period} \leq (2^{32}-1)$ .	Pulses	Number of external signal pulses. As an unsigned 32-bit value. Allowed value range: $0 \leq \text{Pulses} \leq 65,535$ or larger (see comments below).
Period	Time interval. In bits. As an unsigned 32-bit value. <i>1 bit equals 10 µs.</i> Allowed value range: $0 \leq \text{Period} \leq (2^{32}-1)$ .				
Pulses	Number of external signal pulses. As an unsigned 32-bit value. Allowed value range: $0 \leq \text{Pulses} \leq 65,535$ or larger (see comments below).				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>laser_on_pulses_list</b> is useful for marking separate points in Laser Mode 6 (see <a href="#">Chapter 7.4.6 "Laser Mode 6", page 183</a>), but also effective in the other laser modes.</li> <li>• The external pulses must be supplied as TTL pulses at the LASER connector's DIGITAL IN1 digital input (see <a href="#">Section "2-Bit Digital Input Port", page 63</a>). By <b>set_laser_control(Bit #5)</b>, you can specify whether signal pulses should be counted at rising or falling edges.</li> <li>• If <code>Period = 0</code>, then <b>laser_on_pulses_list</b> has no effect. Then <b>laser_on_pulses_list</b> is a short list command.</li> <li>• If <math>0 &lt; \text{Period} \leq (2^{31}-1)</math>, then the command's duration is always <code>Period</code> clocks (that is, <code>Period × 10 µs</code>), even if the specified number of external signal pulses expire in a shorter time interval.</li> <li>• If <math>2^{31} \leq \text{Period} \leq (2^{32}-1)</math>, the command's maximum duration is <math>(\text{Period} - 2^{31})</math> clocks (<math>((\text{Period} - 2^{31}) \times 10 \mu s)</math>). Here, however, the command terminates as soon as the specified number of external signal pulses has been detected.</li> <li>• If <code>Pulses &gt; 65,535</code>, then <b>laser_on_pulses_list</b>'s function is identical to <b>laser_on_list</b> (external signal pulses are not taken into account, see comments there). Otherwise (for <math>0 \leq \text{Pulses} \leq 65,535</math>) <b>laser_on_pulses_list</b> (in contrast to <b>laser_on_list</b>) <i>do not</i> toggle the laser control signals between "laser active" and "laser standby" operation, but instead only switches the LASERON signal, whereby laser delays are not taken into account. Likewise during this command, unexpired laser delays activated by prior commands have no effect (though their effect resume when the LASERON signal switches off again after the final pulse or after <code>Period</code>).</li> <li>• If <math>1 \leq \text{Pulses} \leq 65,535</math>, then the LASERON signal does switch on upon the edge (according the polarity set by <b>set_laser_control(Bit #5)</b>) of the first external pulse (unless it is already on due to an unexpired LaserOff delay) and remains on for the specified number of pulses, but no longer than until the end of the number of <math>10 \mu s</math> periods specified by <code>Period</code>. Users should ensure to define <code>Period</code> large enough for processing <code>Pulses</code> number of signal pulses in this time interval. If the DIGITAL IN1 input does not receive any pulses, then <b>laser_on_pulses_list</b> does not alter the LASERON signal. If more signal pulses than specified by <code>Pulses</code> are received during the time interval defined by <code>Period</code>, then the surplus pulses are ignored.</li> </ul>				



<b>Variable List Command</b>	<b>laser_on_pulses_list</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If Pulses = 0, then <b>laser_on_pulses_list</b> has no effect (the LASERON signal is not switched on). Then <b>laser_on_pulses_list</b> becomes a short list command.</li> <li>The LASERON signal must first be defined and enabled by <b>set_laser_control</b> or <b>enable_laser</b> before it can be switched on with <b>laser_on_pulses_list</b>, see <a href="#">Chapter 7.4 "Laser Control", page 173</a>.</li> <li>While the command is executed, the set position of the scanners is not changed. The next list command is executed when the programmed time interval <b>Period</b> has passed.</li> <li>The Wobbel mode, see <a href="#">Chapter 8.4 "Wobbel Mode", page 218</a>, is retained but ignored.</li> <li>Any softstarts that were defined are not executed.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">laser_on_list</a> , <a href="#">para_laser_on_pulses_list</a>

<b>Ctrl Command</b>	<b>laser_signal_off</b>
Function	Turns off the signals for "laser active" operation immediately.
Call	<code>laser_signal_off()</code>
Comments	<ul style="list-style-type: none"> <li><b>laser_signal_off</b> is intended for direct laser control in combination with <b>laser_signal_on</b>.</li> <li><b>laser_signal_off</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> </ul> </li> <li><b>laser_signal_off</b> is even executed, if: <ul style="list-style-type: none"> <li>a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> <li>the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">laser_signal_on</a>

<b>Normal List Command</b>	<b>laser_signal_off_list</b>
Function	Like <a href="#">laser_signal_off</a> , but a list command.
Call	<code>laser_signal_off_list()</code>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">laser_signal_off</a>



<b>Ctrl Command</b>	<b><a href="#">laser_signal_on</a></b>
<b>Function</b>	Turns on the “laser active” laser control signals immediately.
<b>Call</b>	<code>laser_signal_on()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <i>Caution! Check the beam path before turning on the laser!</i></li> <li>• The signals for “laser active” operation must first be selected by <code>set_laser_mode</code>, defined by further commands, and enabled by <code>set_laser_control</code> or <code>enable_laser</code> before they can be switched on with <code>laser_signal_on</code>, see <a href="#">Chapter 7.4 “Laser Control”, page 173</a>.</li> <li>• <code>laser_signal_on</code> is intended for turning on the laser directly, for example, for alignment purposes.</li> <li>• The signals for “laser active” operation must be turned off by <code>laser_signal_off</code>.</li> <li>• <code>laser_signal_on</code> is not executed (<code>get_last_error</code> return code <code>RTC6_BUSY</code>), if: <ul style="list-style-type: none"> <li>– the <code>BUSY</code> status of the board is set (list is being processed or has been halted by <code>pause_list</code>)</li> <li>– the <code>INTERNAL-BUSY</code> status of the board is set</li> </ul> </li> <li>• <code>laser_signal_on</code> is even executed, if: <ul style="list-style-type: none"> <li>– a list has been paused by <code>set_wait</code> (<code>PAUSED</code> status set)</li> </ul> </li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">laser_signal_off</a>

<b>Normal List Command</b>	<b><a href="#">laser_signal_on_list</a></b>
<b>Function</b>	Like <code>laser_signal_on</code> , but a list command and never ignored.
<b>Call</b>	<code>laser_signal_on_list()</code>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">laser_signal_on</a>



<b>Undelayed Short List Command</b>	<b>list_call</b>
Function	Causes an unconditional jump to a subroutine that starts at the specified absolute address (in any desired location within list memory).
Call	list_call( Pos )
Parameters	Pos      Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li>The first command of a subroutine called by <b>list_call</b> is executed (possibly after a <b>list_continue</b>) immediately and without delay. Nested or recursive calls are also possible, up to a depth of 63, see also <a href="#">Chapter 6.5.1 "Subroutines", page 102</a>.</li> <li>Each subroutine must be terminated by <b>list_return</b> so that after the subroutine (including the terminating <b>list_return</b>) has been processed, execution continues with the command that follows the subroutine-call command. This, too, executes (after a possible <b>list_continue</b>) immediately and without delay. If <b>set_end_of_list</b> is encountered instead of the expected <b>list_return</b>, then list execution terminates or – if previously activated – an automatic list change takes place (for the latter, the current list status is a decisive factor as described below). Under no circumstances does program flow then return again to the calling location, even in the case of nested subroutine calls. Any not-yet-completed <b>mark_text</b> does not execute to completion. If the end of a list memory area ("List 1" or "List 2") is reached without having encountered a <b>list_return</b> or <b>set_end_of_list</b>, then execution continues at the start of the current list. If such a situation occurs in the protected list memory area "List 3", then a compulsory <b>list_return</b> command is inserted and executed.</li> <li>The <b>list_call</b> command is replaced by a <b>list_nop</b> if <b>Pos &gt; (2<sup>23</sup>-1)</b> or if <b>Pos</b> is also the current address (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>If a called subroutine executes a further <b>list_call</b> to the address of the calling <b>list_call</b> command (recursive call), then the resulting endless loop is terminated as soon as the 63-nested-call upper limit is reached. Further <b>list_call</b> commands are then ignored and the next command is instead executed.</li> <li>If the subroutine starts directly at the address which follows <b>list_call</b>, then the subroutine is executed once again after <b>list_return</b> (see also comments on a missing corresponding function call in the <b>list_return</b> command description). The next processed command is the one which follows after <b>list_return</b> (see also <b>list_repeat...list_until</b>). This bypasses the possibly unwanted list processing.</li> </ul>



<b>Undelayed Short List Command</b>	<b>list_call</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The <b>BUSY</b> list status readable by <b>read_status</b> is altered by <b>list_call</b> if the called address is in the list memory area ("List 1" or "List 2"). If this address is instead in protected list memory area "List 3", then the calling location's list status is retained because the protected area does not have its own list status. During execution of a subroutine in protected memory, <b>get_status</b> too returns the calling location's list execution status, and <b>get_out_pointer</b> returns the position of the calling <b>list_call</b> as the output pointer's position.</li> <li>Absolute vector and arc commands execute absolutely after <b>list_call</b> is called. If the subroutine needs to execute at various locations within the image field, then either the subroutine can only contain relative mark, arc or jump commands or <b>list_call_abs</b> must be used instead.</li> <li>If <b>list_call</b> is at the last possible memory position in a list ("List 1" or "List 2"), then – even if automatic list changing was previously enabled – execution continues at the start of the same list after processing of the called subroutine.</li> <li><b>list_call( Pos )</b> is synonymous with <b>list_call_repeat( Pos, 1 )</b>.</li> </ul>
RTC4→RTC6	Basically unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>list_continue, list_repeat, list_return, list_until, list_call_abs, list_call_cond, list_call_repeat, sub_call</b>



<b>Undelayed Short List Command</b>	<b>list_call_abs</b>
Function	Causes an unconditional jump to a subroutine that starts at the specified absolute address (in any desired area of list memory). In the called subroutine, any absolute vector and arc commands receive an offset (based on the current coordinates at the time of the call).
Call	list_call_abs( Pos )
Parameters	Pos            Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li>• <b>list_call_abs</b> is basically identical to <b>list_call</b> (see comments there). However, <b>list_call_abs</b> results in a different execution of absolute vector and arc commands within the called subroutine. When <b>list_call_abs</b> executes, an offset is established for the called subroutine and set according to the current position. Thereby, the current position is automatically considered when absolute vector and arc commands of the called subroutine are subsequently executed. Nested calls are taken into account when the offset is determined (with <b>list_return</b>, the previous offset values are re-established). Subroutines can thus contain absolute vector and arc commands even if they are intended to be repeated in different parts of the image field.</li> <li>• If the called subroutine contains no absolute commands, then there is no difference between <b>list_call_abs</b> and <b>list_call</b>.</li> <li>• <b>list_call_abs( Pos )</b> is synonymous with <b>list_call_abs_repeat( Pos, 1 )</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_call</a> , <a href="#">list_call_abs_cond</a> , <a href="#">list_call_abs_repeat</a>



<b>Undelayed Short List Command</b>	<b>list_call_abs_cond</b>
Function	<p><i>Conditional subroutine call (AbsCall):</i> <b>list_call_abs_cond</b> executes the command <b>list_call_abs( Pos )</b>, if the current IOvalue at the EXTENSION 1 socket connector's 16-bit digital input port meets the following condition:</p> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ <p>(= if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0). Otherwise, the directly following list command is immediately executed.</p>
Call	<code>list_call_abs_cond( Mask1, Mask0, Pos )</code>
Parameters	<p>Mask1      16-bit mask. As an unsigned 32-bit value. Only the least significant 16 bits are evaluated.</p> <p>Mask0      See Mask1.</p> <p>Pos        Absolute jump address [0...(2<sup>23</sup>-1)]. As an unsigned 32-bit value.</p>
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">list_call_abs</a>.</li> <li>• See also Chapter 9.3.2 "Conditional Command Execution", page 281.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_call_abs</a>

<b>Undelayed Short List Command</b>	<b>list_call_abs_repeat</b>
Function	Causes an unconditional jump to a subroutine that starts at the specified absolute address (in any desired area of list memory) and executes its body several times.
Call	<code>list_call_abs_repeat( Pos, Number )</code>
Parameters	<p>Pos        Absolute jump address [0...(2<sup>23</sup>-1)]. As with <a href="#">list_call_abs</a>: As an unsigned 32-bit value.</p> <p>Number     Number of repetitions. As an unsigned 32-bit value. Number = 0 is treated as Number = 1.</p>
Comments	<ul style="list-style-type: none"> <li>• <a href="#">list_call_abs( Pos )</a> is synonymous with <code>list_call_abs_repeat( Pos, 1 )</code>.</li> <li>• <a href="#">list_call_abs_repeat</a> avoids an empty cycle at the repetition, which otherwise inevitably occurs with <a href="#">list_call_abs...list_call_abs</a> or <a href="#">list_repeat...list_call_abs...list_until</a> constructions.</li> <li>• See <a href="#">list_call_repeat</a> and <a href="#">list_call_abs</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_call</a> , <a href="#">list_call_abs</a> , <a href="#">list_call_abs_cond</a> , <a href="#">list_call_repeat</a> , <a href="#">list_repeat</a> , <a href="#">list_until</a>



<b>Undelayed Short List Command</b>	<b>list_call_cond</b>						
Function	<p><i>Conditional subroutine call:</i> <b>list_call_abs_cond</b> executes <a href="#">list_call</a>( Pos ), if the current IOvalue at the EXTENSION 1 socket connector's 16-bit digital input port meets the following condition:</p> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ <p>(= if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0). Otherwise, the directly following list command is immediately executed.</p>						
Call	<code>list_call_cond( Mask1, Mask0, Pos )</code>						
Parameters	<table> <tr> <td>Mask1</td> <td>16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</td></tr> <tr> <td>Mask0</td> <td>See Mask1.</td></tr> <tr> <td>Pos</td> <td>Absolute jump address [0...(2<sup>23</sup>-1)]. As an unsigned 32-bit value.</td></tr> </table>	Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.	Mask0	See Mask1.	Pos	Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.						
Mask0	See Mask1.						
Pos	Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.						
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">list_call</a>.</li> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>.</li> </ul>						
RTC4→RTC6	Basically unchanged functionality (see <a href="#">list_call</a> ).						
RTC5→RTC6	Unchanged functionality.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">list_call</a>						



<b>Undelayed Short List Command</b>	<b>list_call_repeat</b>				
<b>Function</b>	Causes an unconditional jump to a subroutine that starts at the specified absolute address (in any desired area of list memory) and executes its body several times.				
<b>Call</b>	<code>list_call_repeat( Pos, Number )</code>				
<b>Parameters</b>	<table> <tr> <td>Pos</td> <td>Absolute jump address [0...(2<sup>23</sup>-1)]. As with <a href="#">list_call</a>: As an unsigned 32-bit value.</td> </tr> <tr> <td>Number</td> <td>Number of repetitions. As an unsigned 32-bit value. Number = 0 is treated as Number = 1.</td> </tr> </table>	Pos	Absolute jump address [0...(2 <sup>23</sup> -1)]. As with <a href="#">list_call</a> : As an unsigned 32-bit value.	Number	Number of repetitions. As an unsigned 32-bit value. Number = 0 is treated as Number = 1.
Pos	Absolute jump address [0...(2 <sup>23</sup> -1)]. As with <a href="#">list_call</a> : As an unsigned 32-bit value.				
Number	Number of repetitions. As an unsigned 32-bit value. Number = 0 is treated as Number = 1.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <a href="#">list_call( Pos )</a> is synonymous with <code>list_call_repeat( Pos, 1 )</code>.</li> <li>• <a href="#">list_call_repeat</a> avoids an empty cycle at the repetition, which otherwise inevitably occurs with <a href="#">list_call...list_call</a> or <a href="#">list_repeat...list_call...list_until</a> constructions.</li> <li>• By <a href="#">list_call_repeat</a>, for example, trajectories (see Glossary entry on <a href="#">page 880</a>) from micro vector commands for runup curves and coast down curves can be seamlessly joined together with shapes from subroutines.</li> <li>• An empty cycle must be inserted if at <a href="#">list_call_repeat</a> another subroutine call follows immediately (nested calls). This can be avoided, if there is for example, at least one microvector command between two subroutine calls.</li> <li>• The subroutine start <code>Pos</code> can be located in any part of the list memory area. This applies in particular directly after the command <a href="#">list_call_repeat</a>. Thus, the complete list memory can continuously be used for list commands without reserving a special area for protected subroutines.</li> <li>• After a <a href="#">list_return</a> the next processed command is the one which follows directly after <a href="#">list_call_repeat</a>. However, if the subroutine start follows directly after <a href="#">list_call_repeat</a>, then the next processed command is the one which follows directly after <a href="#">list_return</a>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">list_call</a> , <a href="#">list_call_abs</a> , <a href="#">list_call_abs_repeat</a> , <a href="#">list_repeat</a> , <a href="#">list_return</a> , <a href="#">list_until</a> , <a href="#">micro_vector_abs</a> , <a href="#">micro_vector_abs_3d</a> , <a href="#">micro_vector_rel</a> , <a href="#">micro_vector_rel_3d</a>				



<b>Normal List Command</b>	<b>list_continue</b>
<b>Function</b>	Inserts a null operation (no operation) into the list memory.
<b>Call</b>	<code>list_continue()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Execution of <b>list_continue</b> (as does <b>list_nop</b>) requires 10 µs.</li> <li>When <b>list_continue</b> immediately follows a short list command, it ensures (as does <b>list_nop</b>) that the subsequent list command only executes in the next 10 µs clock period (the short list command's "effective" execution time is then 10 µs). Unlike <b>list_nop</b>, however, <b>list_continue</b> neither modifies laser signals nor pauses for scanner delays. In contrast to <b>list_nop</b>, therefore, <b>list_continue</b> allows postponing short list commands to the next 10 µs clock period without interrupting <b>Polylines</b> by switching off the laser.</li> <li>With exceedance of the maximum allowed number of directly consecutive short list commands, an empty cycle (which exactly corresponds <b>list_continue</b>) is automatically inserted.</li> <li>You can also use <b>list_continue</b> to separate from each other several outputs to the same output port. Without this command, for example, multiple directly consecutive <b>write_da_x_list</b> commands (short list commands) would occur in a single 10 µs clock period, whereby some values to the analog output port would never get outputted.</li> <li>The RTC6 never uses <b>list_continue</b> as a placeholder for other (rejected) list commands, but instead always uses <b>list_nop</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">list_nop</a>



<b>Undelayed Short List Command</b>	<b>list_jump_cond</b>
Function	<p><i>Conditional (absolute) list jump:</i> <b>list_jump_cond</b> executes <b>list_jump_pos( Pos )</b>, if the current IOvalue at the EXTENSION 1 socket connector's 16-bit digital input port meets the following condition:</p> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ <p>(= if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0). Otherwise, the directly following list command is immediately executed.</p>
Call	<code>list_jump_cond( Mask1, Mask0, Pos )</code>
Parameters	<p>Mask1      16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</p> <p>Mask0      See Mask1.</p> <p>Pos      Absolute jump address [0...(2<sup>23</sup>-1)]. As an unsigned 32-bit value.</p>
Comments	<ul style="list-style-type: none"> <li>• <b>list_jump_cond</b> is synonymous with <b>list_jump_pos_cond</b> (see comments there).</li> </ul>
RTC4→RTC6	<p>Basically unchanged functionality. However:</p> <ul style="list-style-type: none"> <li>• Jumps into or out from the protected list memory area "List 3" are not allowed (illegal jump commands are ignored during processing, see also <b>list_jump_pos</b>).</li> </ul>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>list_jump_pos, list_jump_pos_cond</b>



<b>Undelayed Short List Command</b>	<b>list_jump_pos</b>
Function	Execution produces an unconditional jump to the specified list-memory address. The next command there is executed immediately without delay.
Call	<code>list_jump_pos( Pos )</code>
Parameters	Pos      Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li>For <code>Pos</code>, an absolute address can be specified within the configured list memory ("List 1" and "List 2"), but not within the protected "List 3" area or completely outside of the list memory area.</li> <li>Jumps into or out from the protected list memory area "List 3" are not allowed.</li> <li>Illegal jump commands are transmitted unaltered to the RTC6, but are ignored during processing. Instead, the next command is executed. Hence, the user program does probably no longer perform as expected. Therefore, jump commands must be carefully programmed (see also <a href="#">Chapter 6.5.3 "Jumps", page 110</a>).</li> <li>Jump commands initiating a jump to themselves (<code>Pos</code> = list position of the jump command) are also ignored at runtime to prevent an infinite loop that excludes further activities (and that could only be halted by <a href="#">stop_execution</a> or an external stop).</li> <li>Decisive are the runtime conditions. When reconfiguring list memory or converting a subroutine, an originally valid jump address might become invalid due to new list boundaries or a relocated subroutine storage position.</li> <li>After a jump to another list area, the status information might under some circumstances not be correct (see also <a href="#">Chapter 6.4.2 "List Status", page 97</a>).</li> <li>The <b>BUSY</b> list status of the two lists is alternatingly set by <code>list_jump_pos</code>, the <b>USED</b> list status of the two lists remains unchanged (see <a href="#">read_status</a>).</li> <li><code>list_jump_pos</code> is synonymous with <a href="#"><code>set_list_jump</code></a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#"><code>set_list_jump</code></a> , <a href="#"><code>list_jump_rel</code></a> , <a href="#"><code>list_jump_pos_cond</code></a>



<b>Undelayed Short List Command</b>	<b>list_jump_pos_cond</b>						
Function	<p><i>Conditional (absolute) list jump:</i> <b>list_jump_pos_cond</b> executes <a href="#">list_jump_pos</a>(<i>Pos</i>), if the current <i>IOvalue</i> at the EXTENSION 1 socket connector's 16-bit digital input port meets the following condition:</p> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ <p>(= if the bits specified in <i>Mask1</i> are 1 and the bits specified in <i>Mask0</i> are 0). Otherwise, the directly following list command is immediately executed.</p>						
Call	<code>list_jump_pos_cond( Mask1, Mask0, Pos )</code>						
Parameters	<table> <tr> <td>Mask1</td><td>16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</td></tr> <tr> <td>Mask0</td><td>See <i>Mask1</i>.</td></tr> <tr> <td>Pos</td><td>Absolute jump address [0...(2<sup>23</sup>-1)]. As an unsigned 32-bit value.</td></tr> </table>	Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.	Mask0	See <i>Mask1</i> .	Pos	Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.						
Mask0	See <i>Mask1</i> .						
Pos	Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.						
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">list_jump_pos</a>.</li> <li>• Unlike the rules for preventing endless loops (see <a href="#">list_jump_pos</a>), jumps by <b>list_jump_pos_cond</b> are allowed even if they are to their own address (<i>Pos</i> = list position of the jump command), for example, to wait for confirmation of a signal.</li> <li>• <b>list_jump_pos_cond</b> is synonymous with <a href="#">list_jump_cond</a>.</li> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>.</li> </ul>						
Examples (Pascal)	<ul style="list-style-type: none"> <li>• wait until Bit #3 of the input port turns HIGH (= loop while the bit is <i>LOW</i>): <code>list_jump_pos_cond(0, \$0008, get_input_pointer);</code></li> <li>• skip the next two list commands if the state of the input port is xxxx xxxx xxxx 0110 : <code>list_jump_pos_cond(6, 9, get_input_pointer + 3);</code></li> <li>• See also <a href="#">Section "Example Code (PASCAL)", page 282</a>.</li> </ul>						
RTC4→RTC6	New command.						
RTC5→RTC6	Unchanged functionality.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">list_jump_pos</a>						



<b>Undelayed Short List Command</b>	<b>list_jump_rel</b>
Function	Execution produces an unconditional jump to the specified address within the current list. The next command there is executed immediately without delay.
Call	list_jump_rel( Pos )
Parameters	Pos      Jump distance [(-2 <sup>23</sup> +1)...(2 <sup>23</sup> -1)]. As a signed 32-bit value.
Comments	<ul style="list-style-type: none"> <li>• <b>list_jump_rel</b> enables implementation of branching (for example, "if-then-else") independently of the command's list position, in particular also coded independently of the list number because of relative addressing.</li> <li>• The current input list pointer can be queried by <b>get_list_pointer</b>.</li> <li>• <b>list_jump_rel</b> is usable in all list areas, including the protected list memory "List 3".</li> <li>• When specifying a jump distance within "List 1" or "List 2" or for non-indexed subroutines within "List 3" be sure that the jump does not exceed the boundaries of the corresponding memory area.</li> <li>• If the command is used in an indexed subroutine or character set, then also be sure that the jump does not exceed the boundaries of the subroutine or character set.</li> <li>• Illegal jump commands are transmitted unaltered to the RTC6, but are ignored during processing. Instead, the next command is executed. Hence, the user program does probably no longer perform as expected. Therefore, jump commands must be carefully programmed (see also <b>Chapter 6.5.3 "Jumps", page 110</b>).</li> <li>• Jump commands initiating a jump to themselves (Pos = 0) is also ignored at runtime to prevent an infinite loop that excludes further activities (and that could only be halted by <b>stop_execution</b> or an external stop).</li> <li>• Decisive are the runtime conditions. When reconfiguring list memory or converting a subroutine, an originally valid jump address might become invalid due to new list boundaries or a relocated subroutine storage position.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>list_jump_pos, list_jump_rel_cond</b>



<b>Undelayed Short List Command</b>	<b>list_jump_rel_cond</b>						
Function	<p><i>Conditional (relative) list jump:</i> <b>list_jump_rel_cond</b> executes <a href="#">list_jump_rel</a>(<i>Pos</i>), if the current <i>IOvalue</i> at the EXTENSION 1 socket connector's 16-bit digital input port meets the following condition:</p> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ <p>(= if the bits specified in <i>Mask1</i> are 1 and the bits specified in <i>Mask0</i> are 0). Otherwise, the directly following list command is immediately executed.</p>						
Call	<code>list_jump_rel_cond( Mask1, Mask0, Pos )</code>						
Parameters	<table> <tr> <td>Mask1</td><td>16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</td></tr> <tr> <td>Mask0</td><td>See <i>Mask1</i>.</td></tr> <tr> <td>Pos</td><td>Jump distance <math>[(-2^{23}+1) \dots (2^{23}-1)]</math>. As a signed 32-bit value.</td></tr> </table>	Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.	Mask0	See <i>Mask1</i> .	Pos	Jump distance $[(-2^{23}+1) \dots (2^{23}-1)]$ . As a signed 32-bit value.
Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.						
Mask0	See <i>Mask1</i> .						
Pos	Jump distance $[(-2^{23}+1) \dots (2^{23}-1)]$ . As a signed 32-bit value.						
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">list_jump_rel</a>.</li> <li>• Unlike the rules for preventing endless loops (see <a href="#">list_jump_rel</a>), jumps by <b>list_jump_rel_cond</b> are allowed even if they are to their own address (<i>Pos</i> = 0), for example, to wait for confirmation of a signal.</li> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>.</li> </ul>						
Examples (Pascal)	<ul style="list-style-type: none"> <li>• Wait until Bit #3 of the input port turns HIGH (= loop while the bit is <i>LOW</i>): <code>list_jump_rel_cond(0, \$0008, 0);</code></li> <li>• Skip the next two list commands if the state of the input port is xxxx xxxx xxxx 0110 : <code>list_jump_rel_cond(6, 9, 3);</code></li> <li>• See also <a href="#">Section "Example Code (PASCAL)", page 282</a>.</li> </ul>						
RTC4→RTC6	New command.						
RTC5→RTC6	Unchanged functionality.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">list_jump_rel</a>						



<b>Undelayed Short List Command</b>	<b>list_next</b>
Function	Executes the next list command.
Call	list_next()
Comments	<ul style="list-style-type: none"> <li>• <b>list_next</b> reads the next list command and executes it immediately, as long as the maximum allowed number of short list commands has not been exceeded. Otherwise, it is executed within the next 10 µs clock period.</li> <li>• <b>list_next</b> is a proper place holder for another command in the list, because it prevents an extra clock, which is unavoidable with other place holders such as <b>list_nop</b> or <b>list_continue</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>list_nop, list_continue</b>

<b>Normal List Command</b>	<b>list_nop</b>
Function	Inserts a null operation (no operation) into the list memory.
Call	list_nop()
Comments	<ul style="list-style-type: none"> <li>• <b>list_nop</b> has the same effect as <b>long_delay( 1 )</b>. It switches off the signals for “laser active” operation after a LaserOff delay and waits for a possible scanner delay. Even if no delays need to be waited for, execution of the command requires 10 µs.</li> <li>• <b>list_nop</b> serves as a placeholder for rejected list commands (<b>get_last_error</b> return code <b>RTC6_IGNORED</b>).</li> <li>• When <b>list_nop</b> immediately follows a short list command, it ensures that the subsequent list command only executes in the next 10 µs clock period (the short list command’s “effective” execution time is then 10 µs).</li> </ul>
RTC4→RTC6	Basically unchanged functionality. But the RTC6 command switches the signals for “laser active” operation off.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>long_delay, list_continue</b>



<b>Undelayed Short List Command</b>	<b>list_repeat</b>
Function	Initiates repetition of a group of list commands.
Call	<code>list_repeat()</code>
Comments	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 6.5.5 "Loops", page 111</a>.</li> <li>Any still-pending delayed short list command executes first.</li> <li>All list commands between this command and the next <a href="#">list_until</a> command is possibly repeated multiple times.</li> <li>Nesting up to 8 <a href="#">list_repeat/list_until</a> loops deep is allowed. Each further <a href="#">list_repeat</a> command beyond that limit is ignored as long as no <a href="#">list_until</a> has terminated a loop.</li> <li>If a list terminates (by <a href="#">set_end_of_list</a> or <a href="#">stop_execution</a> or /STOP), then all not-yet-fully-processed <a href="#">list_repeat/list_until</a> loops are deleted. However, this does not occur if an automatic list change (by <a href="#">auto_change</a>, <a href="#">auto_change_pos</a> or <a href="#">start_loop</a>) is active.</li> <li>Complete <a href="#">list_repeat/list_until</a> loops can reside within a list or subroutine. Changing between lists and subroutines or between different subroutines is not permitted. Changing between lists is permitted as long as the list change occurs without explicit list termination (by an automatic list change or by an explicit list jump to another list with <a href="#">list_jump_pos</a>).</li> <li>List jumps into a <a href="#">list_repeat/list_until</a> loop's body or from inside a loop to a loop-external location should be avoided. Careless use could compromise loop management integrity so severely that loops do not execute as expected. But subroutine calls from inside a loop are always reliably possible.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_until</a>



<b>Undelayed Short List Command</b>	<b>list_return</b>
Function	Terminates a previously called subroutine, jumps to the calling location and executes the next list command (possibly after a <b>list_continue</b> ) immediately and without delay.
Call	list_return()
Comments	<ul style="list-style-type: none"> <li>• While loading an indexed subroutine, character or text string in the protected list memory area "List 3", the <b>list_return</b> command additionally triggers entries into the corresponding internal management table of data such as the starting address.</li> <li>• In contrast, if <b>list_return</b> directly follows a <b>load_sub</b>, <b>load_char</b> or <b>load_text_table</b> command, then the affected subroutine, character or text string are deleted from the corresponding internal management table.</li> <li>• If, during loading into the protected list memory area "List 3", indexed subroutines, characters or text strings are <i>not</i> terminated with <b>list_return</b> before the input pointer is explicitly set to another position, then they are not stored and are thereafter unavailable.</li> <li>• During loading of a <b>list_return</b> command, a flush of the buffered list input is triggered, see <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a>.</li> <li>• If <b>list_return</b> follows <b>load_sub</b>, <b>load_char</b> or <b>load_text_table</b>, then the input pointer after <b>list_return</b> is invalid. That is, further commands can not be loaded without a prior request for a new input pointer positioning.</li> <li>• If, during processing, a <b>list_return</b> is encountered without the prior explicit calling of a subroutine, character or text string, then processing continues at the absolute address 0 (starting address of "List 1"). With nested calls the integrity of the subroutine structure is destroyed.</li> </ul>
RTC4→RTC6	No changes to the previous functionality (termination of a subroutine). New: impact during loading into the protected list memory area "List 3".
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_call</a> , <a href="#">sub_call</a> , <a href="#">load_char</a> , <a href="#">load_text_table</a> , <a href="#">load_sub</a> , <a href="#">list_continue</a>



<b>Undelayed Short List Command</b>	<b>list_until</b>
Function	Terminates repetition of a group of list commands.
Call	<code>list_until( Number )</code>
Parameters	Number      Number of repetitions. As an unsigned 32-bit value. Number = 0 is treated like Number = 1.
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">Chapter 6.5.5 "Loops", page 111</a>.</li> <li>• Any still-pending delayed short list command executes first.</li> <li>• All list commands between this command and the most recent preceding <a href="#">list_repeat</a> command is executed Number number of times.</li> <li>• Nesting up to 8 <a href="#">list_repeat/list_until</a> loops deep is allowed. Each further <a href="#">list_until</a> command for which there was no preceding <a href="#">list_repeat</a> command is ignored.</li> <li>• An empty loop consisting of <a href="#">list_repeat</a> directly followed by <a href="#">list_until</a> terminates immediately and is not repeated.</li> <li>• If a list terminates (by <a href="#">set_end_of_list</a> or <a href="#">stop_execution</a> or /STOP), then all not-yet-fully-processed <a href="#">list_repeat/list_until</a> loops are deleted. However, this does not occur if an automatic list change (by <a href="#">auto_change</a>, <a href="#">auto_change_pos</a> or <a href="#">start_loop</a>) is active.</li> <li>• Complete <a href="#">list_repeat/list_until</a> loops can reside within a list or subroutine. Changing between lists and subroutines or between different subroutines is not permitted. Changing between lists is permitted as long as the list change occurs without explicit list termination (by an automatic list change or by an explicit list jump to another list with <a href="#">list_jump_pos</a>).</li> <li>• List jumps into a <a href="#">list_repeat/list_until</a> loop's body or from inside a loop to a loop-external location should be avoided. Careless use could compromise loop management integrity so severely that loops do not execute as expected. But subroutine calls from inside a loop are always reliably possible.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_repeat</a>



<b>Ctrl Command</b>	<b>load_auto_laser_control</b>																				
<b>Function</b>	Loads a table with data points from an ASCII text file and determines – by linear interpolation – the non-linearity curve for position- and/or speed-dependent laser control, see <a href="#">Section "Loading and Determining the Nonlinearity Curve", page 193.</a>																				
<b>Call</b>	NoOfDataPoints = load_auto_laser_control( Name, No )																				
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.</td> </tr> <tr> <td>No</td> <td>Determines which table in the text file is to be loaded. The parameter corresponds to the extension &lt;No&gt; of [AutoLaserCtrlTable&lt;No&gt;] at the beginning of the desired table. As an unsigned 32-bit value.</td> </tr> </table>	Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.	No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of [AutoLaserCtrlTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.																
Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.																				
No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of [AutoLaserCtrlTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.																				
<b>Result</b>	<p>A positive error code in case of an error, the negative number of found data points in case of success. As a signed 32-bit value.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>-1...- 50</td> <td>Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see also <a href="#">Section "Loading and Determining the Nonlinearity Curve", page 193.</a></td> </tr> <tr> <td>-1024</td> <td>For Name = <b>NULL</b> (see also comments).</td> </tr> <tr> <td>1</td> <td>No valid data points found (though Table No found).</td> </tr> <tr> <td>3</td> <td>File not found.</td> </tr> <tr> <td>4</td> <td>DSP memory error.</td> </tr> <tr> <td>5</td> <td><b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b>, no download (<b>get_last_error</b> return code <b>RTC6_BUSY</b>).</td> </tr> <tr> <td>8</td> <td>Board is locked by another user program (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>).</td> </tr> <tr> <td>11</td> <td>PCI error (<b>get_last_error</b> return code <b>RTC6_SEND_ERROR</b>), verify error (<b>get_last_error</b> return code <b>RTC6_VERIFY_ERROR</b>).</td> </tr> <tr> <td>13</td> <td>The specified table number was not found in the file.</td> </tr> </tbody> </table>	Value	Description	-1...- 50	Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see also <a href="#">Section "Loading and Determining the Nonlinearity Curve", page 193.</a>	-1024	For Name = <b>NULL</b> (see also comments).	1	No valid data points found (though Table No found).	3	File not found.	4	DSP memory error.	5	<b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <b>get_last_error</b> return code <b>RTC6_BUSY</b> ).	8	Board is locked by another user program ( <b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b> ).	11	PCI error ( <b>get_last_error</b> return code <b>RTC6_SEND_ERROR</b> ), verify error ( <b>get_last_error</b> return code <b>RTC6_VERIFY_ERROR</b> ).	13	The specified table number was not found in the file.
Value	Description																				
-1...- 50	Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see also <a href="#">Section "Loading and Determining the Nonlinearity Curve", page 193.</a>																				
-1024	For Name = <b>NULL</b> (see also comments).																				
1	No valid data points found (though Table No found).																				
3	File not found.																				
4	DSP memory error.																				
5	<b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <b>get_last_error</b> return code <b>RTC6_BUSY</b> ).																				
8	Board is locked by another user program ( <b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b> ).																				
11	PCI error ( <b>get_last_error</b> return code <b>RTC6_SEND_ERROR</b> ), verify error ( <b>get_last_error</b> return code <b>RTC6_VERIFY_ERROR</b> ).																				
13	The specified table number was not found in the file.																				
<b>Comments</b>	<ul style="list-style-type: none"> <li>The format requirements for the text file's table entries with data points for the nonlinearity curve are described in <a href="#">Section "Loading and Determining the Nonlinearity Curve", page 193.</a> When loading the table, the RTC6 determines suitable values for the entire range of percent values.</li> <li><b>load_auto_laser_control</b> overwrites any previously loaded nonlinearity curve.</li> <li>For Name = <b>NULL</b> (as during initialization by <b>load_program_file</b>), the function <b>Scale(Percent)=1.0</b> is loaded for the complete percent range (no nonlinearity).</li> </ul>																				



Ctrl Command	load_auto_laser_control
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>load_auto_laser_control</b> is not executed (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>load_auto_laser_control</b> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <a href="#">set_wait</a> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• During the runtime of <b>load_auto_laser_control</b>, external starts are suppressed.</li> <li>• Before loading a table, <b>load_auto_laser_control</b> performs a <b>DSP</b> memory check that produces error code 4 in case of error.</li> <li>• The table can be saved by <a href="#">create_dat_file</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_auto_laser_control</a> , <a href="#">create_dat_file</a>



<b>Ctrl Command</b>	<b>load_char</b>
<b>Function</b>	Assigns a desired index to a character defined by subsequent list commands, and loads the character into the protected list memory area "List 3".
<b>Call</b>	<code>load_char( Char )</code>
<b>Parameters</b>	Char      Index of the indexed character. As an unsigned 32-bit value. Allowed range [0...1023].
<b>Comments</b>	<ul style="list-style-type: none"> <li>Up to 1024 indexed characters, hence 4 character sets with 256 indexed characters per set, can be stored. For defining character sets, the following applies: <math>\text{Char} = \text{character set number} * 256 + \text{ASCII number of the character}</math> (character sets are numbered 0 to 3). If <math>\text{Char} &gt; 1023</math> then <b>load_char</b> is ignored (<a href="#">get_last_error</a> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>The addresses in the protected list memory area "List 3" where the character definitions are to be stored are automatically determined and internally managed. This management is independent of that for indexed subroutines (see <a href="#">load_sub</a>) and text string definitions (see <a href="#">load_text_table</a>).</li> <li>Indexed character definitions must be terminated with a <a href="#">list_return</a> command. This is a prerequisite for actual storage of the commands, entry of the start address and other information (for example, the number of commands) into the internal management table, and initiating a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a>. Otherwise (the input pointer is altered without a preceding <a href="#">list_return</a> command) the character definition with this index is not available.</li> <li>An indexed character definition is not stored if the protected list memory area "List 3" was not previously configured for a sufficient size beyond "List 1" and "List 2".</li> <li>If <a href="#">list_return</a> is the next command after <b>load_char</b>, then the corresponding character definition is deleted from the internal management table.</li> <li>Observe all notes in <a href="#">Chapter 6.5.2 "Character Sets and Text Strings", page 108</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_return</a> , <a href="#">load_sub</a> , <a href="#">load_text_table</a>



<b>Ctrl Command</b>	<b>load_correction_file</b>
<b>Function</b>	Loads the specified image field correction file into RTC6 memory (as correction table #1...#8). Then calls automatically <b>select_cor_table</b> with the most recently used parameter values (or the default parameter values).
<b>Call</b>	ErrorNo = load_correction_file( Name, No, Dim )
<b>Parameters</b>	<p>Name      Name of the correction file. As a pointer to a \0-terminated ANSI string.</p> <p>No      Determines under which number #1...#8 the correction table is to be stored. Allowed values: [1...8]. As an unsigned 32-bit value.</p> <p>Dim      Determines whether the correction file content is stored as a 2D correction table or (if possible) as a 3D correction table (see also comments). As an unsigned 32-bit value. = 2:    2D and 3D correction files are stored as a 2D correction table ( downgrade, if necessary ). = 3:    If the <b>Option "3D"</b> is enabled, 2D and 3D correction files are stored as a 3D correction table ( upgrade, if necessary ). If the <b>Option "3D"</b> is not enabled, then <b>load_correction_file</b> has the same effect as with <b>Dim</b> = 2. Others: <b>Dim</b> has no effect. A 2D correction file is stored as a 2D correction table. A 3D correction file is stored as a 3D correction table, if the <b>Option "3D"</b> is enabled, otherwise as a 2D correction table ( downgrade ).</p>
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <p>0      Success.</p> <p>1      File error (file corrupt or incomplete).</p> <p>2      Memory error (<b>RTC6 DLL</b>-internal, Windows system memory).</p> <p>3      File-open error (empty string submitted for <b>Name</b> parameter, file not found, etc.).</p> <p>4      <b>DSP</b> memory error.</p> <p>5      PCI download error (RTC6 board driver error), Ethernet download error.</p> <p>8      RTC6 board driver not found (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>).</p> <p>10     Parameter error (incorrect <b>No</b>).</p>

Ctrl Command	load_correction_file	
Result (cont'd)	11	Access error: board reserved for another user program ( <code>get_last_error</code> return code <code>RTC6_ACCESS_DENIED</code> ) or version check has detected an error (OUT or RTC version not compatible to the current RTC6 DLL version, <code>get_last_error</code> return code <code>RTC6_ACCESS_DENIED   RTC6_VERSION_MISMATCH</code> ).
	12	Warning: 3D correction table or Dim = 3 selected, but the Option "3D" is not enabled. The system subsequently operates as an ordinary 2D system (this warning is only returned, if no other error has occurred).
	13	Busy error: no download, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b> ( <code>get_last_error</code> return code <code>RTC6_BUSY</code> ).
	14	PCI upload error (RTC6 board driver error, only applicable for download verification)
	15	Verify error (only applicable for download verification).
Comments	<p>On loading correction tables:</p> <ul style="list-style-type: none"> <li>The RTC6 can store 8 different correction tables at the same time, for example, for use in a multiple scan head configuration.</li> <li>Before storing a correction file, <code>load_correction_file</code> performs a <b>DSP</b> memory check that produces error code 4 in case of error.</li> <li>If the parameter <code>No</code> is out of range, then no correction table is loaded (return value 10).</li> <li>The name of the to-be-loaded correction file must be passed to <code>load_correction_file</code> as a pointer to a \0-terminated ANSI string. If <code>Name</code> is passed as a null pointer (0), the corresponding table is replaced by a 1-to-1 table (for <code>Dim</code> = 3 and enabled Option "3D" with a 1-to-1 3D correction table, otherwise with a 1-to-1 2D table). Most of the above-mentioned parameters are then set to 0. An empty string ("") for <code>Name</code> result in an error return code of 3.</li> <li>If the Option "3D" is not enabled (default), then 2D and 3D correction files are stored as 2D correction tables – regardless of the value specified for <code>Dim</code>. The 3D data sections of 3D correction files are then ignored.</li> <li>If, on the other hand, the Option "3D" is enabled, then both 2D and 3D correction tables can be loaded. <ul style="list-style-type: none"> <li>For <code>Dim</code> = 2, a 2D table is always stored (the 3D data section of 3D correction files are ignored) and, accordingly, only 2D corrections are calculated (the z axis thereby remains unchanged, as if no Option "3D" was enabled).</li> <li>For <code>Dim</code> = 3, if the Option "3D" is enabled then both 2D and 3D correction files are stored as 3D correction tables. 2D correction tables are thereby automatically expanded to incorporate a linear Z correction. The actually suitable Z correction can subsequently be loaded by <code>load_z_table</code> or <code>load_z_table_no</code>, see Chapter 7.3.4 "3D Image Field", page 159.</li> <li>All other values for <code>Dim</code> do not change the type of the correction file.</li> </ul> </li> </ul>	



Ctrl Command	load_correction_file
Comments (cont'd)	<p>On assigning correction tables by <code>select_cor_table</code>:</p> <ul style="list-style-type: none"> <li>• Use the <code>select_cor_table</code> or <code>select_cor_table_list</code> command to assign one (or two) correction table(s) stored on the RTC6 to the scan head (or to both scan head connectors).</li> <li>• <code>load_correction_file</code> automatically calls <code>select_cor_table</code> after loading of a correction table. However, if you call <code>load_correction_file</code> before loading the program file (by <code>load_program_file</code>), then the automatic call of <code>select_cor_table</code> has no effect. If you call <code>load_correction_file</code> after <code>load_program_file</code>, then the <code>select_cor_table</code> call uses the parameter values (<code>HeadA = 1, HeadB = 0</code>) or the values most recently used (after <code>load_program_file</code>) when having called <code>select_cor_table</code>. <code>load_correction_file</code> does not return to the user program until the <code>select_cor_table</code>-induced jump to the corrected galvanometer scanner position has been completed.</li> </ul> <p>Others:</p> <ul style="list-style-type: none"> <li>• RTC5/RTC6 correction tables contain parameters that with an RTC4/RTC3/RTC2 must be looked-up in a supplied ReadMe file first, and then must be entered manually into the user program. With the RTC6 these parameters can be read from the currently loaded correction tables (by <code>get_table_para</code>) or from the assigned correction tables (by <code>get_head_para</code>). Thus, they are directly integrated into a user program.</li> <li>• During the runtime of <code>load_correction_file</code>, external starts are suppressed.</li> <li>• <code>load_correction_file</code> is not executed (<code>get_last_error</code> return code <code>RTC6_BUSY</code>), if: <ul style="list-style-type: none"> <li>– the <code>BUSY</code> status of the board is set (list is being processed or has been halted by <code>pause_list</code>)</li> <li>– the <code>INTERNAL-BUSY</code> status of the board is set</li> </ul> </li> <li>• <code>load_correction_file</code> is even executed, if: <ul style="list-style-type: none"> <li>– a list has been paused by <code>set_wait</code> (<code>PAUSED</code> status set)</li> </ul> </li> <li>• If an <code>RTC6_VERSION_MISMATCH</code> error occurs (return value 11), a RTC6 DLL version appropriate for the program file must be chosen and the board must be made currentless (to unload the program software) or alternatively program files appropriate for the RTC6 DLL version must be reloaded by <code>load_program_file</code> (after <code>RTC6_ACCESS_DENIED</code>, the single-board command <code>load_program_file</code> does not get access rights for the board). Only afterward (and after the board has been reacquired by <code>acquire_RTC</code> or <code>select_RTC</code>) <code>load_correction_file</code> can be normally executed again.</li> </ul>



Ctrl Command	<b>load_correction_file</b>
RTC4→RTC6	<p>Except its basic function and the parameters <code>Name</code> and <code>No</code>, the functionality of the command has substantially changed:</p> <ul style="list-style-type: none"> <li>• <code>load_correction_file</code> now uses the new parameter <code>Dim</code>. This allows, for instance, storage of 3D correction files as 2D correction tables and storage of 2D correction files as 3D correction tables.</li> <li>• Now, <i>up to 8</i> 3D correction tables can be stored in RTC6 memory (in the 3D-enabled version). However, two 3D correction tables can <i>not</i> be simultaneously assigned to the scan head connector (see <code>select_cor_table</code>).</li> <li>• The parameters <code>k</code>, <code>Phi</code> and <code>Offset</code> no longer exist. Therefore, table transformations (scaling, rotation, translation) are no longer possible during loading of the correction file. Such coordinate transformations can now be specified by <code>set_scale</code>, <code>set_angle</code> and <code>set_offset</code> in addition to <code>set_matrix</code>.</li> <li>• <code>select_cor_table</code> is automatically called, see comments above and <a href="#">Section "Notes", page 165</a>.</li> <li>• <code>select_cor_table</code> does not return to the user program until the correction motion has been completed (see comments above).</li> </ul>
RTC5→RTC6	<p>Changed functionality.</p> <ul style="list-style-type: none"> <li>• <code>load_correction_file</code> lets you load 8 correction tables simultaneously into the RTC6 memory.</li> <li>• Overlaps no longer exist between memory areas used elsewhere.</li> </ul>
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	—



<b>Ctrl Command</b>	<b>load_disk</b>
<b>Function</b>	Loads into the protected list memory area "List 3" the indexed characters, text strings and subroutines previously stored in a binary file by <b>save_disk</b> and returns the number of actually loaded list commands.
<b>Call</b>	NoOfLoadedCommands = load_disk( Name, Mode )
<b>Parameters</b>	<p>Name            File name. As a pointer to a \0-terminated ANSI string.</p> <p>Mode          Determines how the loading procedure is executed. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>= 0: The internal management tables for indexed characters, text strings and subroutines are initialized (all old references are thereby lost) and the input pointer is set to the beginning of "List 3" (the resulting loading process overwrites list commands stored there).</li> <li>&gt; 0: Internal management tables are not initialized (all old references are initially retained, but then replaced or supplemented by other references during the loading process, depending on the file's content) and the input pointer is set to the position after the last stored (by <b>load_char</b>, <b>load_text_table</b> or <b>load_sub</b>) indexed character, text string or subroutine (the resulting loading process does <i>not</i> overwrite the list commands of old indexed characters, text strings and subroutines).</li> </ul>
<b>Result</b>	The number of commands loaded by <b>load_disk</b> . As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For <code>Name = NULL</code>, only the internal management tables are initialized as with <code>Mode = 0</code> (no loading occurs, no "empty" binary file must be provided). Otherwise, <b>load_disk</b> can only be used for reading files that were stored with <b>save_disk</b>.</li> <li>• For all indexed characters, text strings and subroutines in the specified file, <b>load_disk</b> executes a corresponding <b>load_char</b>, <b>load_text_table</b> or <b>load_sub</b> command. The corresponding list commands are thereby written to the protected list memory area "List 3" and appropriate entries are made in the internal management tables in accordance with the index assignment stored in the file. If there is no character, text string or subroutine for a particular index in the specified file, then no list commands are loaded for this index, and if <code>Mode &gt; 0</code> then any already existing entries in the internal management table remains unaltered. Thus, supplementary loading of indexed characters, text strings and subroutines from various files is possible.</li> </ul>

Ctrl Command	load_disk
Comments (cont'd)	<ul style="list-style-type: none"> <li>Together with <b>save_disk</b>, <b>load_disk</b> can be used, for example, to defragment the protected list memory area "List 3" and to subsequently protect subroutines, see <b>Section "Subsequent Protection and Conversion of Non-Indexed Subroutines"</b>, page 106 and <b>Section "Index Management and Defragmentation"</b>, page 105. If the characters, text strings and subroutines come from various files, then defragmentation can be achieved if the first file is loaded in <code>Mode = 0</code> and subsequent files are loaded with <code>Mode &gt; 0</code>, provided that no indices are used simultaneously in different files. Memory gaps in the protected area caused by dereferencing of indexed characters, text strings or subroutines are thereby closed.</li> <li>If, during loading, the end of the protected list memory area "List 3" is reached before the end of the file (EOF), then all further list commands in the file are ignored. Likewise, incomplete characters, text strings and subroutines (as with individual <b>load_char</b>, <b>load_text_table</b> or <b>load_sub</b> commands) are not stored. Before each <b>load_disk</b>, be sure that the memory configuration provides a sufficiently large protected list memory area "List 3" above the list areas ("List 1" and "List 2"). <b>save_disk</b> returns the number of list commands stored in the file. If a board changes ownership, it is the user's responsibility to ensure that the memory configuration data is consistent (<code>Mem1</code> and <code>Mem2</code> are queried from the <b>RTC6 DLL</b>, not from the board – see also <b>get_config_list</b> and <b>Chapter 6.7.1 "Notes on Board Acquisition by a User Program"</b>, page 118).</li> <li>If the specified file is corrupt and cannot be read to the end, then only the characters, text strings and subroutines fully readable to that point are stored.</li> <li><b>load_disk</b> is not executed (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>), if: <ul style="list-style-type: none"> <li>"List 3" has no memory area assigned (<math>Mem1 + Mem2 = 2^{23}</math>)</li> </ul> </li> <li><b>load_disk</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li><b>load_disk</b> is even executed, if: <ul style="list-style-type: none"> <li>a list has only been paused by <b>set_wait</b> (<b>PAUSED</b> status set) But users are responsible for ensuring that no still-needed commands are overwritten, for example, if <b>set_wait</b> has been called from an indexed subroutine.</li> </ul> </li> <li>During the runtime of <b>load_disk</b>, no other commands can be executed.</li> <li>During the runtime of <b>load_disk</b>, external starts are suppressed.</li> <li><b>load_disk</b> checks the version info saved by <b>save_disk</b> and compares it with the current runtime version. Both must match. Otherwise, <b>load_disk</b> is not executed (<b>get_last_error</b> return code <b>RTC6_VERSION_MISMATCH</b>).</li> </ul>



<b>Ctrl Command</b>	<b>load_disk</b>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 615: version check.
References	<a href="#">save_disk</a>



<b>Ctrl Command</b>	<b>load_fly_2d_table</b>				
<b>Function</b>	Loads a 2D table from an ASCII text file for a <b>set_fly_2d</b> -Processing-on-the-fly application with 2D encoder compensation for xy positioning stages, see <a href="#">Section "2D Encoder Compensation for xy Positioning Stages", page 234</a> .				
<b>Call</b>	NoOfDataPoints = load_fly_2d_table( Name, No )				
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.</td> </tr> <tr> <td>No</td> <td>Determines which table in the text file is to be loaded. The parameter corresponds to the extension &lt;No&gt; of [Fly2DTable&lt;No&gt;] at the beginning of the desired table. As an unsigned 32-bit value.</td> </tr> </table>	Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.	No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of [Fly2DTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.
Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.				
No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of [Fly2DTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.				
<b>Result</b>	<p>A positive error code in case of an error, the negative number of found data points in case of success. As a signed 32-bit value.</p> <ul style="list-style-type: none"> <li>&lt; 0 Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see <a href="#">Section "For the 2D compensation tables, the following rules apply:", page 235</a>.</li> <li>0 For Name = <b>NULL</b> (see comments).</li> <li>1 No valid data points found (though Table No found).</li> <li>2 Out of Memory (not enough Windows system memory).</li> <li>3 File not found.</li> <li>4 DSP memory error.</li> <li>5 <b>BUSY</b> error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b>, no download (<b>get_last_error</b> return code <b>RTC6_BUSY</b>).</li> <li>8 Board is locked by another user program (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>).</li> <li>11 PCI error (<b>get_last_error</b> return code <b>RTC6_SEND_ERROR</b>), verify error (<b>get_last_error</b> return code <b>RTC6_VERIFY_ERROR</b>).</li> <li>13 The specified table number was not found in the file.</li> </ul>				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• No = &lt;No&gt; loads table number 1; No = &lt;No&gt; + 65.536 loads table number 2.</li> <li>• The text file's data format requirements for reference points of the 2D encoder compensation table are described in <a href="#">Section "For the 2D compensation tables, the following rules apply:", page 235</a>. The largest of these reference points should not exceed the range -524,288...+524,287 (otherwise precision may be lost). During runtime, the current encoder values (including reference values) must not exceed the largest values specified in the table. Otherwise clipping occurs.</li> <li>• <b>load_fly_2d_table</b> overwrites a previously loaded table for 2D encoder compensation.</li> <li>• If Name = <b>NULL</b>, then a 0-correction table for 2D encoder compensation is loaded and the table is marked as invalid (see <b>init_fly_2d</b>).</li> </ul>				



<b>Ctrl Command</b>	<b>load_fly_2d_table</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>load_fly_2d_table</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>load_fly_2d_table</b> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• During the runtime of <b>load_fly_2d_table</b>, external starts are suppressed.</li> <li>• Before loading a table, <b>load_fly_2d_table</b> performs a <b>DSP</b> memory check that produces error code 4 in case of error.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_fly_2d</b> , <b>init_fly_2d</b> , <b>get_fly_2d_offset</b>

<b>Ctrl Command</b>	<b>load_jump_table</b>												
Function	Loads a value table with jump delay data points from an ASCII text file (or alternatively performs automatic determination on the attached scan system) and uses linear interpolation to create the internal jump delay table for 2D jumps executable in jump mode.												
Call	NoOfDataPoints = load_jump_table( Name, No, PosAck, MinDelay, MaxDelay, ListPos )												
Parameters	<table> <tr> <td>Name</td> <td>See <b>load_jump_table_offset</b>.</td> </tr> <tr> <td>No</td> <td>See <b>load_jump_table_offset</b>.</td> </tr> <tr> <td>PosAck</td> <td>See <b>load_jump_table_offset</b>.</td> </tr> <tr> <td>MinDelay</td> <td>See <b>load_jump_table_offset</b>.</td> </tr> <tr> <td>MaxDelay</td> <td>See <b>load_jump_table_offset</b>.</td> </tr> <tr> <td>ListPos</td> <td>See <b>load_jump_table_offset</b>.</td> </tr> </table>	Name	See <b>load_jump_table_offset</b> .	No	See <b>load_jump_table_offset</b> .	PosAck	See <b>load_jump_table_offset</b> .	MinDelay	See <b>load_jump_table_offset</b> .	MaxDelay	See <b>load_jump_table_offset</b> .	ListPos	See <b>load_jump_table_offset</b> .
Name	See <b>load_jump_table_offset</b> .												
No	See <b>load_jump_table_offset</b> .												
PosAck	See <b>load_jump_table_offset</b> .												
MinDelay	See <b>load_jump_table_offset</b> .												
MaxDelay	See <b>load_jump_table_offset</b> .												
ListPos	See <b>load_jump_table_offset</b> .												
Result	See <b>load_jump_table_offset</b> .												
Comments	<ul style="list-style-type: none"> <li>• <b>load_jump_table</b> is identical to <b>load_jump_table_offset</b> with <b>Offset = 0</b>.</li> </ul>												
RTC4→RTC6	New command.												
RTC5→RTC6	Unchanged functionality.												
Version info	Available as of version DLL 600, OUT 600, RBF 600.												
Reference	<b>load_jump_table_offset</b>												

<b>Ctrl Command</b>	<b>load_jump_table_offset</b>
<b>Function</b>	Loads a value table with jump delay data points from an ASCII text file (or alternatively performs automatic determination on the attached scan system) and uses linear interpolation to create the internal jump delay table for 2D jumps executable in jump mode.
<b>Call</b>	NoOfDataPoints = load_jump_table_offset( Name, No, PosAck, Offset, MinDelay, MaxDelay, ListPos )
<b>Parameters</b>	<p>Name      Name of the text file or <b>NULL</b> for automatic determination. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.</p> <p>No        Specifies: <ul style="list-style-type: none"> <li>For <b>Name</b> = <b>Filename</b>, which table of the text file should be loaded. The parameter corresponds to the extension <b>&lt;No&gt;</b> of <b>[JumpTable&lt;No&gt;]</b> at the beginning of the desired table.</li> <li>For <b>Name</b> = <b>NULL</b>, which scan system (or which scan head connector) should be used for automatic determination. Allowed values: [1, 2].</li> </ul> As an unsigned 32-bit value. </p> <p>PosAck     Position tolerance value in [bits] (only relevant for automatic determination). As an unsigned 32-bit value.</p> <p>Offset      Offset in [10 µs] as a signed 32-bit number, which is added to all automatically determined delay values (only relevant for automatic determination). As a signed 32-bit value.</p> <p>MinDelay    Minimum jump delay in [10 µs] (only relevant for automatic determination). As an unsigned 32-bit value.</p> <p>MaxDelay    Maximum jump delay in [10 µs] (only relevant for automatic determination). As an unsigned 32-bit value.</p> <p>ListPos     List position (in the area of list 1 or 2) for six list commands that can be overwritten (only relevant for automatic determination). As an unsigned 32-bit value.</p>
<b>Result</b>	<p>Error code. As a signed 32-bit value.</p> <ul style="list-style-type: none"> <li>-1...-50   Success for <b>Name</b> = <b>filename</b>. The absolute value of the return value equals the number of valid data points found in the table. Invalid entry values are ignored, see also <a href="#">Section "Notes on Loading Determined Jump Delay Values", page 205</a>.</li> <li>-1024      Success for <b>Name</b> = <b>NULL</b>: Table was automatically determined.</li> <li>1            No valid data points found (but Table <b>No</b> found).</li> <li>3            File not found.</li> <li>4            Verify error: <b>DSP</b> memory error.</li> <li>5            Busy error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b>, no download (<b>get_last_error</b> return code <b>RTC6_BUSY</b>).</li> <li>8            Access error: board reserved for another user program (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>).</li> <li>10          only if <b>Name</b> = <b>NULL</b>: Param error: HeadNo or ListPos invalid.</li> <li>11          PCI error during download (<b>get_last_error</b> return code <b>RTC6_SEND_ERROR</b>).</li> <li>13          The supplied table number could not be found in the file.</li> </ul>



Ctrl Command	<code>load_jump_table_offset</code>
Comments	<ul style="list-style-type: none"> <li>For information on command usage, see <a href="#">Section "Jump-Length-Dependent Jump Delays", page 204</a>.</li> <li>For jump mode information, see <a href="#">Chapter 8.1.5 "Jump Mode", page 203</a>.</li> <li>Format requirements for placing the table with jump delay data points into the text file are described in the <a href="#">Section "Notes on Loading Determined Jump Delay Values", page 205</a>. When loading the table, the RTC6 uses linear interpolation to establish appropriate values for the complete jump length range.</li> <li><code>load_jump_table_offset</code> overwrites any previously loaded jump delay tables for jump mode.</li> <li>If <code>Name = filename</code>, then the table number <code>No</code> is loaded. The other parameters are not used.</li> <li>If <code>Name = NULL</code>, then the jump delay table for head <code>No</code> is automatically determined and loaded (if jump mode has been previously enabled and activated successfully by <code>set_jump_mode</code> (<code>Flag = 1</code>)). Here, the parameters <code>PosAck</code>, <code>Offset</code>, <code>MinDelay</code>, <code>MaxDelay</code> and <code>ListPos</code> are used (see <a href="#">Section "Automatic Determination of the Jump Delay Table", page 207</a>).</li> <li>Jump mode takes effect upon the next 2D jump only if it was activated and enabled by <code>set_jump_mode</code> or activated by <code>set_jump_mode_list</code>.</li> <li><code>load_jump_table_offset</code> is not executed (<code>get_last_error</code> return code <code>RTC6_BUSY</code>), if: <ul style="list-style-type: none"> <li>the <code>BUSY</code> status of the board is set (list is being processed or has been paused by <code>pause_list</code>)</li> <li>the <code>INTERNAL-BUSY</code> status of the board is set</li> </ul> </li> <li><code>load_jump_table_offset</code> is even executed, if: <ul style="list-style-type: none"> <li>a list has been paused by <code>set_wait</code> (<code>PAUSED</code> status set)</li> </ul> </li> <li>During the runtime of <code>load_jump_table_offset</code>, external starts are suppressed.</li> <li>Before loading a table, <code>load_jump_table_offset</code> performs a <code>DSP</code> memory check that produces error code 4 in case of error.</li> <li>The table can be saved by <code>create_dat_file</code>.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>There is no <a href="#">RTC4 Compatibility Mode</a> for this command.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
Reference	<a href="#">load_jump_table</a> , <a href="#">set_jump_mode</a> , <a href="#">set_jump_mode_list</a> , <a href="#">get_jump_table</a> , <a href="#">set_jump_table</a> , <a href="#">create_dat_file</a>



Ctrl Command	load_list
Function	Opens the list memory for writing with list commands and sets the input pointer to the specified (relative) position within the desired list ("List 1" or "List 2"), but only if the list is not currently being processed ( <b>BUSY</b> status not set) or has already been processed ( <b>USED</b> status set). In case of success, the next list command is stored at this address and all further commands at subsequent addresses in the selected list.
Call	NoOfOpenedList = load_list( ListNo, Pos )
Parameters	<p>ListNo      Number of the list in which the input pointer should be set. As an unsigned 32-bit value. Allowed values: [0...3]. Only the two least significant bits are evaluated.</p> <ul style="list-style-type: none"> <li>= 0: The list that is currently not <b>BUSY</b> is opened. If both lists are not <b>BUSY</b>, then "List 1" is opened.</li> <li>= 1: "List 1" is opened if not <b>BUSY</b> (<b>BUSY1</b> status not set).</li> <li>= 2: "List 2" is opened if not <b>BUSY</b> (<b>BUSY2</b> status not set).</li> <li>= 3: The list that is currently not <b>BUSY</b> but <b>USED</b> (<b>USED</b> status set) is opened. If both lists are not <b>BUSY</b> and both are <b>USED</b>, then that list is opened, which would be executed by a following automatic list change.</li> </ul> <p>For Mem2 = 0 (see <b>config_list</b>) "List 1" is opened if not <b>BUSY</b> (ListNo = 0...2) or if not <b>BUSY</b> but <b>USED</b> (ListNo = 3).</p> <p>Pos      Position of the input pointer (offset relative to the start of the respective list). As an unsigned 32-bit value. Allowed value range: [0...(2<sup>23</sup>-1)].</p>
Result	Number of the opened list [1 or 2] if successful, otherwise 0. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li>• <b>load_list</b> (ListNo = 3) is useful in scenarios such as alternating list changes, where you want to wait specifically for a list to be processed (see <a href="#">Section "Alternating List Changes", page 101</a>) without needing to separately query the list status. Unintentional overwriting of not-yet-executed commands is thereby automatically avoided. To instead perform <i>unconditional</i> loading, you can use commands such as <b>set_start_list_pos</b>.</li> <li>• After a successful check of the list number (<b>BUSY</b> status not set and, if applicable, <b>USED</b> status set), <b>load_list</b> behaves like <b>set_start_list_pos</b> (see comments there).</li> <li>• If <b>load_list</b> was not successful (return code 0), then no list is opened and the input pointer is set to an invalid position. Then no further list commands can be input until the input pointer is correctly set (for example, by repeating <b>load_list</b> with a positive result or by the <b>set_start_list_pos</b> command etc.). It is up to users to react to a return code of 0. <b>load_list</b> produces no wait loop and <i>does not</i> block execution of subsequent control commands.</li> </ul>



Ctrl Command	load_list
Comments (cont'd)	<ul style="list-style-type: none"> <li>If <code>ListNo = 0...2</code> when using <code>load_list</code>, then note that a list's <b>BUSY</b> status can change between opening of the list with <code>load_list</code> and the actual loading of list commands, for example, if in the meantime an automatic list change has occurred that was previously defined by <code>auto_change</code> or <code>start_loop</code>. In cases such as this, where loading of a list might occur simultaneously with processing of the same list, users must always ensure that the output pointer does not overtake the input pointer.</li> <li>In contrast, for <code>ListNo = 3</code> <code>load_list</code> ensures that a list is actually opened for loading only directly after processing (and after any successful automatic list changes). Particularly, if no list is <b>BUSY</b> and both lists are USED (for example, after initialization, after <code>stop_execution</code> or after an external stop), the opened list number is synchronized with the following automatic list change.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_start_list_pos</a>



<b>Ctrl Command</b>	<b>load_position_control</b>																				
<b>Function</b>	Loads a table with data points from an ASCII text file and determines – by linear interpolation – the scaling function for position-dependent laser control (radial correction, see <a href="#">Section "Position-Dependent Laser Control", page 188</a> ).																				
<b>Call</b>	NoOfDataPoints = load_position_control( Name, No )																				
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.</td> </tr> <tr> <td>No</td> <td>Determines which table in the text file is to be loaded. The parameter corresponds to the extension &lt;No&gt; of [PositionCtrlTable&lt;No&gt;] at the beginning of the desired table. As an unsigned 32-bit value.</td> </tr> </table>	Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.	No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of [PositionCtrlTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.																
Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.																				
No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of [PositionCtrlTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.																				
<b>Result</b>	<p>A positive error code in case of an error, the negative number of found data points in case of success. As a signed 32-bit value.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>-1...- 50</td> <td>Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see also <a href="#">Section "Notes on Loading a Scaling Function", page 188</a>.</td> </tr> <tr> <td>-256</td> <td>For Name = <b>NULL</b> (see also comments).</td> </tr> <tr> <td>1</td> <td>No valid data points found (though Table No found).</td> </tr> <tr> <td>3</td> <td>File not found.</td> </tr> <tr> <td>4</td> <td>DSP memory error.</td> </tr> <tr> <td>5</td> <td><b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b>, no download (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>).</td> </tr> <tr> <td>8</td> <td>Board is locked by another user program (<a href="#">get_last_error</a> return code <b>RTC6_ACCESS_DENIED</b>).</td> </tr> <tr> <td>11</td> <td>PCI error (<a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b>), verify error (<a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b>).</td> </tr> <tr> <td>13</td> <td>The specified table number was not found in the file.</td> </tr> </tbody> </table>	Value	Description	-1...- 50	Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see also <a href="#">Section "Notes on Loading a Scaling Function", page 188</a> .	-256	For Name = <b>NULL</b> (see also comments).	1	No valid data points found (though Table No found).	3	File not found.	4	DSP memory error.	5	<b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).	8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <b>RTC6_ACCESS_DENIED</b> ).	11	PCI error ( <a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b> ), verify error ( <a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b> ).	13	The specified table number was not found in the file.
Value	Description																				
-1...- 50	Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see also <a href="#">Section "Notes on Loading a Scaling Function", page 188</a> .																				
-256	For Name = <b>NULL</b> (see also comments).																				
1	No valid data points found (though Table No found).																				
3	File not found.																				
4	DSP memory error.																				
5	<b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).																				
8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <b>RTC6_ACCESS_DENIED</b> ).																				
11	PCI error ( <a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b> ), verify error ( <a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b> ).																				
13	The specified table number was not found in the file.																				
<b>Comments</b>	<ul style="list-style-type: none"> <li>The format requirements for the text file's table entries with data points for position-dependent laser control are described in <a href="#">Section "Notes on Loading a Scaling Function", page 188</a>. When loading the table, the RTC6 determines suitable values for the entire range of control values.</li> <li><b>load_position_control</b> overwrites any previously loaded scaling function for position-dependent laser control.</li> <li>For Name = <b>NULL</b> (as during initialization by <b>load_program_file</b>), the scaling function <math>Scale(Position) = 1.0</math> is loaded for the complete position range so that no position-dependent correction takes place.</li> <li>Position-dependent laser control only takes effect during subsequent <b>[*]mark[*]</b> commands or arc commands if it was initialized by <b>set_auto_laser_control</b>. Position-dependent laser control is deactivated by <b>set_auto_laser_control</b> (Ctrl = 0) or by loading <math>Scale(Position) = 1.0</math>. See also <a href="#">Section "Position-Dependent Laser Control", page 188</a>.</li> </ul>																				



Ctrl Command	load_position_control
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>load_position_control</b> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC6_BUSY</a>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>load_position_control</b> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <a href="#">set_wait</a> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• During the runtime of <b>load_position_control</b>, external starts are suppressed.</li> <li>• Before loading a table, <b>load_position_control</b> performs a <b>DSP</b> memory check that produces error code 4 in case of error.</li> <li>• The table can be saved by <a href="#">create_dat_file</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_auto_laser_control</a> , <a href="#">create_dat_file</a>



<b>Ctrl Command</b>	<b>load_program_file</b>																																								
<b>Function</b>	Executes a board reset, performs a <b>DSP</b> memory check, loads into RTC6 memory the program file <b>RTC6OUT.out</b> along with the files <b>RTC6RBF.rbf</b> and <b>RTC6DAT.dat</b> from the specified directory, performs a version check and starts the <b>DSP</b> .																																								
<b>Call</b>	ErrorNo = load_program_file( Path )																																								
<b>Parameters</b>	<p>Path      Path name of the directory, where the files <b>RTC6OUT.out</b>, <b>RTC6RBF.rbf</b> and <b>RTC6DAT.dat</b> are stored. As a pointer to a \0-terminated string.</p>																																								
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Success.</td> </tr> <tr> <td>1</td> <td>Not used.</td> </tr> <tr> <td>2</td> <td>The board is not running. If a renewed call does not bring success, then a power cycle is necessary.</td> </tr> <tr> <td>3</td> <td><b>RTC6DAT.dat</b> file or <b>RTC6RBF.rbf</b> file not found.</td> </tr> <tr> <td>4</td> <td>Not used.</td> </tr> <tr> <td>5</td> <td>Not enough Windows memory.</td> </tr> <tr> <td>6</td> <td>Access error: the board is reserved for another user program.</td> </tr> <tr> <td>7</td> <td>Version error: <b>RTC6 DLL</b> version, RTC version (firmware file <b>RTC6RBF.rbf</b>) and OUT version (<b>DSP</b> program file <b>RTC6OUT.out</b>) are incompatible with each other.</td> </tr> <tr> <td>8</td> <td>RTC6 board driver not found.</td> </tr> <tr> <td>9</td> <td>Loading of <b>RTC6OUT.out</b> or <b>RTC6ETH.out</b> failed or has incorrect format or other error.</td> </tr> <tr> <td>10</td> <td>Not used.</td> </tr> <tr> <td>11</td> <td>Firmware error: loading of <b>RTC6RBF.rbf</b> file failed.</td> </tr> <tr> <td>12</td> <td>Error opening/reading file <b>RTC6DAT.dat</b>.</td> </tr> <tr> <td>13</td> <td>Not used.</td> </tr> <tr> <td>14</td> <td><b>DSP</b> memory error (external).</td> </tr> <tr> <td>15</td> <td>Verify memory error.</td> </tr> <tr> <td>16</td> <td><b>DSP</b> memory error (internal).</td> </tr> <tr> <td>17</td> <td>Ethernet error. See <b>eth_get_last_error</b> and <b>eth_get_error</b>.</td> </tr> <tr> <td>18</td> <td>Only RTC6 Ethernet Board: NAND memory error.</td> </tr> </table>	Value	Description	0	Success.	1	Not used.	2	The board is not running. If a renewed call does not bring success, then a power cycle is necessary.	3	<b>RTC6DAT.dat</b> file or <b>RTC6RBF.rbf</b> file not found.	4	Not used.	5	Not enough Windows memory.	6	Access error: the board is reserved for another user program.	7	Version error: <b>RTC6 DLL</b> version, RTC version (firmware file <b>RTC6RBF.rbf</b> ) and OUT version ( <b>DSP</b> program file <b>RTC6OUT.out</b> ) are incompatible with each other.	8	RTC6 board driver not found.	9	Loading of <b>RTC6OUT.out</b> or <b>RTC6ETH.out</b> failed or has incorrect format or other error.	10	Not used.	11	Firmware error: loading of <b>RTC6RBF.rbf</b> file failed.	12	Error opening/reading file <b>RTC6DAT.dat</b> .	13	Not used.	14	<b>DSP</b> memory error (external).	15	Verify memory error.	16	<b>DSP</b> memory error (internal).	17	Ethernet error. See <b>eth_get_last_error</b> and <b>eth_get_error</b> .	18	Only RTC6 Ethernet Board: NAND memory error.
Value	Description																																								
0	Success.																																								
1	Not used.																																								
2	The board is not running. If a renewed call does not bring success, then a power cycle is necessary.																																								
3	<b>RTC6DAT.dat</b> file or <b>RTC6RBF.rbf</b> file not found.																																								
4	Not used.																																								
5	Not enough Windows memory.																																								
6	Access error: the board is reserved for another user program.																																								
7	Version error: <b>RTC6 DLL</b> version, RTC version (firmware file <b>RTC6RBF.rbf</b> ) and OUT version ( <b>DSP</b> program file <b>RTC6OUT.out</b> ) are incompatible with each other.																																								
8	RTC6 board driver not found.																																								
9	Loading of <b>RTC6OUT.out</b> or <b>RTC6ETH.out</b> failed or has incorrect format or other error.																																								
10	Not used.																																								
11	Firmware error: loading of <b>RTC6RBF.rbf</b> file failed.																																								
12	Error opening/reading file <b>RTC6DAT.dat</b> .																																								
13	Not used.																																								
14	<b>DSP</b> memory error (external).																																								
15	Verify memory error.																																								
16	<b>DSP</b> memory error (internal).																																								
17	Ethernet error. See <b>eth_get_last_error</b> and <b>eth_get_error</b> .																																								
18	Only RTC6 Ethernet Board: NAND memory error.																																								



Ctrl Command	load_program_file
Comments	<ul style="list-style-type: none"> <li>If <code>Path = NULL</code>, then the path of the user program's current working directory is used. Caution: The user program's current working directory is not always the directory from which the user program was launched. The current working directory can change, for example, when a file from another directory is selected by the Windows Explorer (unless the "NoChangeDir" flag was set when incorporating the Explorer window into the user program).</li> <li>After each hardware reset (powerup), the first user program must begin by issuing a <code>load_program_file</code> command during initialization of the RTC6 board, see <a href="#">Initializing the Board, page 88</a>. <code>load_program_file</code> should also be executed (for example, if another user program acquires the board – see <a href="#">acquire_rtc</a>) when the board needs to be returned to the default state.</li> <li>If multiple RTC6 boards are connected as master and slave, then <code>load_program_file</code> must have been called on all boards prior to initializing and operating the individual boards with further commands, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 114</a>.</li> <li><code>load_program_file</code> resets the RTC6, initializes the list memory, performs a <b>DSP</b> memory check, loads the firmware (<code>RTC6RBF.rbf</code>), the program file <code>RTC6OUT.out</code> and a binary auxiliary file (<code>RTC6DAT.dat</code>) and starts the <b>DSP</b> (with RTC6 Ethernet Boards, <code>RTC6ETH.out</code> is loaded instead of <code>RTC6OUT.out</code>). After execution of the <code>load_program_file</code> command, the laser focus is positioned in the center of the image field at the point (0 0) and the laser control is <b>deactivated</b>.</li> <li>The <code>load_program_file</code> command does <i>not</i> load correction tables. Even 1-to-1 tables therefore need to be explicitly requested, see <a href="#">load_correction_file</a>. Already-loaded correction tables remain loaded after <code>load_program_file</code>.</li> <li><code>load_correction_file</code> assigns a correction table by <code>select_cor_table( 1, 0 )</code> but does not execute a galvanometer scanner motion to the corrected output position.</li> <li><code>load_program_file</code> only returns to the calling user program, when <b>DSP</b> initialization has been completed.</li> <li><code>load_program_file</code> automatically executes <code>stop_execution</code>, if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li> <li>the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>The files <code>RTC6OUT.out</code>, <code>RTC6ETH.out</code>, <code>RTC6RBF.rbf</code> and <code>RTC6DAT.dat</code> are included in the RTC6 software package. For easy identifying and archiving of different software versions, the files are also delivered zipped (the <code>zip</code> file names <code>RTC6&lt;...&gt;_&lt;Version&gt;.zip</code> include the version numbers). Copy or unzip the three files (of desired version) to the harddisk of your PC.</li> </ul>



Ctrl Command	load_program_file
Comments (cont'd)	<ul style="list-style-type: none"> <li>Assorted versions of the <b>RTC6 DLL</b> and the files <b>RTC6OUT.out</b>, <b>RTC6ETH.out</b>, <b>RTC6RBF.rbf</b> and <b>RTC6DAT.dat</b> cannot be arbitrarily combined with another (each zip file in the RTC6 software includes a text file with version information). <b>load_program_file</b> performs a version compatibility check. If there is a version error, then the loaded programs remain in RTC6 memory, but the board is released by <b>release_RTC</b> directly after the version check and therefore is not available for further commands other than those not requiring access rights (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>   <b>RTC6_VERSION_MISMATCH</b>). To then load a correct program version, <b>load_program_file</b> can be called. Hereby, temporary access rights are requested and released after the download (if the board has not been reserved by another user program; <b>load_program_file</b> does not perform an <b>acquire_RTC</b> command).</li> </ul>
RTC4→RTC6	<ul style="list-style-type: none"> <li>The command parameter specifies a directory name with RTC6 (in contrast a file name with the RTC4).</li> <li><b>load_program_file</b> loads three files, with fixed formats and names (<b>RTC6OUT.out</b>, <b>RTC6RBF.rbf</b>, <b>RTC6DAT.dat</b>) (see above).</li> <li>After execution of the command, the laser control is <i>deactivated</i>.</li> </ul>
RTC5→RTC6	<p>Changed functionality.</p> <ul style="list-style-type: none"> <li>Notes on migrating the source code of RTC5 user programs, etc.: <b>load_program_file</b> is downward compatible with the RTC5 and can continue to be used without modification. Though you do not need to change your user program, keep in mind the following: <ul style="list-style-type: none"> <li>There is no checking of the <b>BUSY</b> status (hence error code 13 is never be outputted).</li> <li>Data transfer by the <b>McBSP interface</b> is deactivated without warning.</li> <li>A running list is terminated without warning, similarly to the control command <b>stop_execution</b>. Observe notes there about mirror positions etc.</li> <li>Other used error codes are provided as listed in the table above.</li> </ul> </li> </ul>
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–



<b>Ctrl Command</b>	<b>load_stretch_table</b>																							
<b>Function</b>	Loads a table with data pairs from an ASCII text file for enhanced 3D correction, see <a href="#">Section "Enhanced 3D Correction", page 225</a> .																							
<b>Call</b>	NoOfDataPairs = load_stretch_table( Name, No, TableNo )																							
<b>Parameters</b>	<p>Name              Name of the text file or <b>NULL</b>. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.</p> <p>No              As a signed 32-bit value.           <ul style="list-style-type: none"> <li>• For <math>No \geq 0</math>, this parameter specifies which table in the text file is to be loaded. The parameter corresponds to the extension <math>&lt;No&gt;</math> of [StretchTable&lt;No&gt;] at the beginning of the desired table.</li> <li>• <math>No &lt; 0</math>: Reserved.</li> </ul> </p> <p>TableNo        The already loaded 3D correction table to which the extended correction is assigned. As a signed 32-bit value. Allowed value range: 1...8.</p>																							
<b>Result</b>	<p>Signed 32-bit value (a positive error code in case of an error, the negative number of found data pairs in case of success).</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt; 0</td> <td>Success. The absolute value of the return value is equal to the number of valid data pairs found in the table.</td> </tr> <tr> <td>0</td> <td>Reserved.</td> </tr> <tr> <td>2</td> <td>Out of Memory (not enough Windows memory).</td> </tr> <tr> <td>3</td> <td>File not found.</td> </tr> <tr> <td>4</td> <td>DSP memory error.</td> </tr> <tr> <td>5</td> <td><b>BUSY</b> error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b>, no download (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>).</td> </tr> <tr> <td>6</td> <td>Data error: data pairs missing.</td> </tr> <tr> <td>11</td> <td>PCI download error (<a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b>).</td> </tr> <tr> <td>13</td> <td>The specified table number was not found in the file.</td> </tr> <tr> <td>15</td> <td>Verify error (<a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b>, only possible with active download verification, see <a href="#">set_verify</a>).</td> </tr> </tbody> </table>		Value	Description	< 0	Success. The absolute value of the return value is equal to the number of valid data pairs found in the table.	0	Reserved.	2	Out of Memory (not enough Windows memory).	3	File not found.	4	DSP memory error.	5	<b>BUSY</b> error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).	6	Data error: data pairs missing.	11	PCI download error ( <a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b> ).	13	The specified table number was not found in the file.	15	Verify error ( <a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b> , only possible with active download verification, see <a href="#">set_verify</a> ).
Value	Description																							
< 0	Success. The absolute value of the return value is equal to the number of valid data pairs found in the table.																							
0	Reserved.																							
2	Out of Memory (not enough Windows memory).																							
3	File not found.																							
4	DSP memory error.																							
5	<b>BUSY</b> error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).																							
6	Data error: data pairs missing.																							
11	PCI download error ( <a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b> ).																							
13	The specified table number was not found in the file.																							
15	Verify error ( <a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b> , only possible with active download verification, see <a href="#">set_verify</a> ).																							
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For details about enhanced 3D correction, see <a href="#">Section "Enhanced 3D Correction", page 225</a>.</li> <li>• The enhanced 3D correction is also switched when <a href="#">select_cor_table</a> is called.</li> <li>• A successfully loaded table activates the new enhanced 3D correction. Here, <a href="#">load_stretch_table</a> overwrites a previously loaded table of the same TableNo.</li> <li>• If <b>Name</b> is not <b>NULL</b>, but no table was successfully read, then <a href="#">load_stretch_table</a> returns an error code, but otherwise has no effect (that is, any previous successfully downloaded table remains valid).</li> </ul>																							



Ctrl Command	<b>load_stretch_table</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If <code>Name</code> is <b>NULL</b>, then <code>load_stretch_table</code> disables any enhanced 3D correction enabled by a previous <code>load_stretch_table</code>.</li> <li><code>load_stretch_table</code> is not executed (<code>get_last_error</code> return code <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <code>pause_list</code>)</li> <li>the <b>INTERNAL-BUSY</b> status of the board status is set</li> </ul> </li> <li><code>load_stretch_table</code> is even executed, if: <ul style="list-style-type: none"> <li>a list has been paused by <code>set_wait</code> (<b>PAUSED</b> status set)</li> </ul> </li> <li>During execution of <code>load_stretch_table</code>, external starts are suppressed.</li> <li>Before loading a table, <code>load_stretch_table</code> performs a <b>DSP</b> memory check that produces error code 4 in case of error.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Changed functionality. <ul style="list-style-type: none"> <li>Additional <code>TableNo</code> parameter. This allows that each 3D correction table can be assigned its own extended 3D correction.</li> </ul>
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>select_cor_table</b>



<b>Ctrl Command</b>	<b>load_sub</b>
<b>Function</b>	Assigns the specified index to a subroutine defined by subsequent list commands and loads the subroutine into the protected list memory area "List 3".
<b>Call</b>	<code>load_sub( Index )</code>
<b>Parameters</b>	<p>Index      Index of the indexed subroutine. As an unsigned 32-bit value. Allowed range [0...1023].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Up to 1024 indexed subroutines can be stored. If <code>Index &gt; 1023</code> then the <b>load_sub</b> command is ignored (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>The address in the protected list memory area "List 3" where the subroutine should be stored is automatically determined and internally managed.</li> <li>Indexed subroutines must be terminated with a <b>list_return</b> command. This is a prerequisite for actual storage of the commands, entry of the start address and other information (for example, the number of commands) into the internal management table, and initiating a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a>. Otherwise (the input pointer is altered without a preceding <b>list_return</b> command) the subroutine with this index is not available.</li> <li>An indexed subroutine is not stored if the protected list memory area "List 3" was not previously configured for a sufficient size beyond "List 1" and "List 2".</li> <li>If <b>list_return</b> is the next command after <b>load_sub</b>, then the corresponding subroutine is deleted from the internal management table.</li> <li>Indexed subroutines can be called by the <b>sub_call</b> command along with the corresponding index (see <a href="#">Section "General Information on Calling Subroutines", page 104</a>).</li> <li>Observe all notes in <a href="#">Section "Indexed Subroutines", page 103</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">list_return</a> , <a href="#">sub_call</a> , <a href="#">load_char</a> , <a href="#">load_text_table</a>



<b>Ctrl Command</b>	<b>load_text_table</b>
<b>Function</b>	Assigns the specified index to a text string defined by subsequent list commands and loads the text string into the protected list memory area "List 3".
<b>Call</b>	<code>load_text_table( Index )</code>
<b>Parameters</b>	<p>Index      Index of the indexed text string. As an unsigned 32-bit value. Allowed range [0...41].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Up to 42 indexed text strings can be stored (for marking times, dates and serial numbers by other commands). The following ordering applies: <ul style="list-style-type: none"> <li>Index = 0...9: digits for marking the time and date [0...9]</li> <li>Index = 10...21: months [January...December]</li> <li>Index = 22...28: days-of-the-week [Sunday...Saturday]</li> <li>Index = 29: blank character for marking serial numbers</li> <li>Index = 30...39: digits for marking serial numbers [0...9]</li> <li>Index = 40: text for "a.m."</li> <li>Index = 41: text for "p.m."</li> </ul> </li> <li>If Index &gt; 41 then <b>load_text_table</b> is ignored (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>Even if digits are defined by <b>mark_text</b> instead of as individual characters with <b>mark_char</b>, the character set can still be subsequently switched for this purpose (see <b>select_char_set</b>).</li> <li>The addresses in the protected list memory area "List 3" where the text string definitions are stored are automatically determined and internally managed. Management is independent of that for indexed subroutines (see <b>load_sub</b>) and character definitions (see <b>load_char</b>).</li> <li>Indexed text string definitions must be terminated with a <b>list_return</b> command. This is a prerequisite for actual storage of the commands, entry of the start address and other information (for example, the number of commands) into the internal management table, and initiating a flush of the buffered list input, see <b>Chapter 6.4.1 "Loading Lists", page 95</b>. Otherwise (the input pointer is altered without a preceding <b>list_return</b> command) the text string with this index is not available.</li> <li>An indexed text string definition is not stored if the protected list memory area "List 3" was not previously configured for a sufficient size beyond "List 1" and "List 2".</li> <li>If <b>list_return</b> is the next command after <b>load_text_table</b>, then the corresponding text string definition is deleted from the internal management table.</li> <li>Also observe all notes in the <b>Chapter 6.5.2 "Character Sets and Text Strings", page 108</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>list_return</b> , <b>load_sub</b> , <b>load_char</b>



<b>Ctrl Command</b>	<b>load_varpolydelay</b>																				
<b>Function</b>	Loads a table with data points from an ASCII text file for the scaling function of the variable polygon delay, see <a href="#">Section "Loading User-defined Variable Polygon Delays", page 142</a> .																				
<b>Call</b>	NoOfDataPoints = load_varpolydelay( Name, No )																				
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.</td> </tr> <tr> <td>No</td> <td>Table in the text file which is to be loaded. The parameter corresponds to the extension &lt;No&gt; of [VarPolyTable&lt;No&gt;] at the beginning of the desired table. As an unsigned 32-bit value.</td> </tr> </table>	Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.	No	Table in the text file which is to be loaded. The parameter corresponds to the extension <No> of [VarPolyTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.																
Name	Name of the text file. The text file may contain one or more tables. As a pointer to a \0-terminated ANSI string.																				
No	Table in the text file which is to be loaded. The parameter corresponds to the extension <No> of [VarPolyTable<No>] at the beginning of the desired table. As an unsigned 32-bit value.																				
<b>Result</b>	<p>Signed 32-bit value (a positive error code in case of an error, the negative number of found data points in case of success).</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>-1...- 50</td> <td>Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see <a href="#">Section "Loading User-defined Variable Polygon Delays", page 142</a>.</td> </tr> <tr> <td>-1024</td> <td>For Name = <b>NULL</b>: the table initialized according to <math>1 - \cos(\phi)</math> has been internally loaded (as with program start), see <a href="#">figure 36</a>.</td> </tr> <tr> <td>1</td> <td>No valid data points found (though Table No found).</td> </tr> <tr> <td>3</td> <td>File not found.</td> </tr> <tr> <td>4</td> <td>DSP memory error.</td> </tr> <tr> <td>5</td> <td><b>BUSY</b> error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b>, no download (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>).</td> </tr> <tr> <td>8</td> <td>The board is locked by another user program (<a href="#">get_last_error</a> return code <b>RTC6_ACCESS_DENIED</b>).</td> </tr> <tr> <td>11</td> <td>PCI error (<a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b>), verify error (<a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b>).</td> </tr> <tr> <td>13</td> <td>The specified table number has not been found in the file.</td> </tr> </tbody> </table>	Value	Description	-1...- 50	Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see <a href="#">Section "Loading User-defined Variable Polygon Delays", page 142</a> .	-1024	For Name = <b>NULL</b> : the table initialized according to $1 - \cos(\phi)$ has been internally loaded (as with program start), see <a href="#">figure 36</a> .	1	No valid data points found (though Table No found).	3	File not found.	4	DSP memory error.	5	<b>BUSY</b> error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).	8	The board is locked by another user program ( <a href="#">get_last_error</a> return code <b>RTC6_ACCESS_DENIED</b> ).	11	PCI error ( <a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b> ), verify error ( <a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b> ).	13	The specified table number has not been found in the file.
Value	Description																				
-1...- 50	Success. The absolute value of the return value is equal to the number of valid data points found in the table. Invalid entries are ignored, see <a href="#">Section "Loading User-defined Variable Polygon Delays", page 142</a> .																				
-1024	For Name = <b>NULL</b> : the table initialized according to $1 - \cos(\phi)$ has been internally loaded (as with program start), see <a href="#">figure 36</a> .																				
1	No valid data points found (though Table No found).																				
3	File not found.																				
4	DSP memory error.																				
5	<b>BUSY</b> error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b> , no download ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).																				
8	The board is locked by another user program ( <a href="#">get_last_error</a> return code <b>RTC6_ACCESS_DENIED</b> ).																				
11	PCI error ( <a href="#">get_last_error</a> return code <b>RTC6_SEND_ERROR</b> ), verify error ( <a href="#">get_last_error</a> return code <b>RTC6_VERIFY_ERROR</b> ).																				
13	The specified table number has not been found in the file.																				
<b>Comments</b>	<ul style="list-style-type: none"> <li>The format requirements for text file's table entries with data points for the user-defined variable polygon delay" are described in <a href="#">Section "Loading User-defined Variable Polygon Delays", page 142</a>. When loading the table, the RTC6 determines suitable values for the entire range of angles by linear interpolation.</li> <li><b>load_varpolydelay</b> overwrites any previously loaded table for the "Variable polygon delay".</li> <li>For Name = <b>NULL</b> (as during initialization by <b>load_program_file</b>), the internal (default) table for the "Variable polygon delay" (<math>1 - \cos(\phi)</math>, see <a href="#">figure 36</a>) is loaded.</li> </ul>																				



Ctrl Command	load_varpolydelay
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>load_varpolydelay</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (a list is being processed or has been halted by <b>pause_list</b>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>load_varpolydelay</b> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• During the runtime of <b>load_varpolydelay</b>, external starts are suppressed.</li> <li>• Before loading a table, <b>load_varpolydelay</b> performs a <b>DSP</b> memory check that produces error code 4 in case of error.</li> <li>• The table can be saved by <b>create_dat_file</b>.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: <ul style="list-style-type: none"> <li>• To return to the internal standard polygon delay table, a reset or renewed program loading by <b>load_program_file</b> is no longer necessary (see <b>Name = NULL</b> above).</li> <li>• The ASCII text file can have any filename extension.</li> </ul>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>load_program_file, set_delay_mode, create_dat_file</b>



<b>Ctrl Command</b>	<b>load_z_table</b>						
<b>Function</b>	Loads coefficients <i>A</i> , <i>B</i> and <i>C</i> into the currently assigned 3D correction table.						
<b>Restriction</b>	Like <a href="#">load_z_table_no</a> .						
<b>Call</b>	ErrorNo = load_z_table( <i>A</i> , <i>B</i> , <i>C</i> )						
<b>Parameters</b>	<table border="0"> <tr> <td><i>A</i></td> <td>Like <a href="#">load_z_table_no</a>.</td> </tr> <tr> <td><i>B</i></td> <td>Like <a href="#">load_z_table_no</a>.</td> </tr> <tr> <td><i>C</i></td> <td>Like <a href="#">load_z_table_no</a>.</td> </tr> </table>	<i>A</i>	Like <a href="#">load_z_table_no</a> .	<i>B</i>	Like <a href="#">load_z_table_no</a> .	<i>C</i>	Like <a href="#">load_z_table_no</a> .
<i>A</i>	Like <a href="#">load_z_table_no</a> .						
<i>B</i>	Like <a href="#">load_z_table_no</a> .						
<i>C</i>	Like <a href="#">load_z_table_no</a> .						
<b>Result</b>	Like <a href="#">load_z_table_no</a> .						
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>load_z_table</b> is not executed (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>load_z_table</b> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <a href="#">set_wait</a> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• <b>load_z_table</b> should always be used after <a href="#">load_correction_file</a>, since <a href="#">load_correction_file</a> sets the three coefficients to the default values of the loaded correction table.</li> <li>• The <b>ABC</b> values are lost by <a href="#">select_cor_table</a>.</li> <li>• <b>load_z_table( <i>A</i>, <i>B</i>, <i>C</i> )</b> is synonymous with <b>load_z_table_no( <i>A</i>, <i>B</i>, <i>C</i>, 0 )</b>.</li> <li>• See also other comments at <a href="#">load_z_table_no</a>.</li> </ul>						
<b>RTC4→RTC6</b>	Unchanged functionality. In addition: changed value ranges and error codes. If the correct calibration factors are used (see <a href="#">Chapter "3D Commands", page 223</a> ), then the same ABC coefficients can be used on the same 3-axis scan system with the RTC4 and RTC6 boards.						
<b>RTC5→RTC6</b>	Unchanged functionality.						
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.						
<b>References</b>	<a href="#">get_z_distance</a> , <a href="#">read_abc_from_file</a> , <a href="#">write_abc_to_file</a> , <a href="#">load_z_table_no</a> , <a href="#">select_cor_table</a>						



<b>Ctrl Command</b>	<b>load_z_table_no</b>																												
<b>Function</b>	Loads coefficients <i>A</i> , <i>B</i> and <i>C</i> and then assigns them to the 3D correction table <i>No.</i>																												
<b>Restriction</b>	If the <b>Option "3D"</b> has not been enabled or a 3D correction table has not been assigned (see <b>select_cor_table</b> ), then <b>load_z_table_no</b> returns the error code 12 or 13 and otherwise has no effect.																												
<b>Call</b>	ErrorNo = load_z_table_no( <i>A</i> , <i>B</i> , <i>C</i> , <i>No</i> )																												
<b>Parameters</b>	<table> <tr> <td><i>A</i></td><td>Coefficient <i>A</i> of the parabolic function <math>z_{out} = A + Bi + Ci^2</math> which is used for calculating the Z output values (<i>i</i> in the RTC4 compatibility range [-32,768...+32,767]). As a 64-bit IEEE floating point value. Allowed value range: [-67108864.0...67108864.0]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td><i>B</i></td><td>Like <i>A</i> (analogously). Allowed value range: [-2048.0...2048.0].</td></tr> <tr> <td><i>C</i></td><td>Like <i>A</i> (analogously). Allowed value range: [-16.0...16.0].</td></tr> <tr> <td><i>No</i></td><td>Number of the 3D correction table to which the three coefficients <i>A</i>, <i>B</i>, <i>C</i> are to be assigned.</td></tr> </table>	<i>A</i>	Coefficient <i>A</i> of the parabolic function $z_{out} = A + Bi + Ci^2$ which is used for calculating the Z output values ( <i>i</i> in the RTC4 compatibility range [-32,768...+32,767]). As a 64-bit IEEE floating point value. Allowed value range: [-67108864.0...67108864.0]. Out-of-range values are clipped to the boundary values.	<i>B</i>	Like <i>A</i> (analogously). Allowed value range: [-2048.0...2048.0].	<i>C</i>	Like <i>A</i> (analogously). Allowed value range: [-16.0...16.0].	<i>No</i>	Number of the 3D correction table to which the three coefficients <i>A</i> , <i>B</i> , <i>C</i> are to be assigned.																				
<i>A</i>	Coefficient <i>A</i> of the parabolic function $z_{out} = A + Bi + Ci^2$ which is used for calculating the Z output values ( <i>i</i> in the RTC4 compatibility range [-32,768...+32,767]). As a 64-bit IEEE floating point value. Allowed value range: [-67108864.0...67108864.0]. Out-of-range values are clipped to the boundary values.																												
<i>B</i>	Like <i>A</i> (analogously). Allowed value range: [-2048.0...2048.0].																												
<i>C</i>	Like <i>A</i> (analogously). Allowed value range: [-16.0...16.0].																												
<i>No</i>	Number of the 3D correction table to which the three coefficients <i>A</i> , <i>B</i> , <i>C</i> are to be assigned.																												
<b>Result</b>	<p>Error code. As an unsigned 32-bit value. Error bits with values 1...64 can also occur combined, but not in conjunction with error codes 11...17, which only occur separately. Warnings 12 and 13 are only returned if no other errors exist.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error.</td> </tr> <tr> <td>1</td> <td><i>A</i> exceeded the maximum allowed value.</td> </tr> <tr> <td>2</td> <td><i>A</i> undercut the minimum allowed value.</td> </tr> <tr> <td>4</td> <td><i>B</i> exceeded the maximum allowed value.</td> </tr> <tr> <td>8</td> <td><i>B</i> undercut the minimum allowed value.</td> </tr> <tr> <td>16</td> <td><i>C</i> exceeded the maximum allowed value.</td> </tr> <tr> <td>32</td> <td><i>C</i> undercut the minimum allowed value.</td> </tr> <tr> <td>64</td> <td>Execution denied (possibly a <b>BUSY</b> or <b>INTERNAL-BUSY</b> error; for exact reason see <b>get_last_error</b>).</td> </tr> <tr> <td>11</td> <td>Access denied.</td> </tr> <tr> <td>12</td> <td><b>Option "3D"</b> is not enabled.</td> </tr> <tr> <td>13</td> <td>No 3D correction table is currently assigned.</td> </tr> <tr> <td>14</td> <td>RTC6 board driver not found.</td> </tr> <tr> <td>15</td> <td>Invalid table number (&gt; 8).</td> </tr> </tbody> </table>	Value	Description	0	No error.	1	<i>A</i> exceeded the maximum allowed value.	2	<i>A</i> undercut the minimum allowed value.	4	<i>B</i> exceeded the maximum allowed value.	8	<i>B</i> undercut the minimum allowed value.	16	<i>C</i> exceeded the maximum allowed value.	32	<i>C</i> undercut the minimum allowed value.	64	Execution denied (possibly a <b>BUSY</b> or <b>INTERNAL-BUSY</b> error; for exact reason see <b>get_last_error</b> ).	11	Access denied.	12	<b>Option "3D"</b> is not enabled.	13	No 3D correction table is currently assigned.	14	RTC6 board driver not found.	15	Invalid table number (> 8).
Value	Description																												
0	No error.																												
1	<i>A</i> exceeded the maximum allowed value.																												
2	<i>A</i> undercut the minimum allowed value.																												
4	<i>B</i> exceeded the maximum allowed value.																												
8	<i>B</i> undercut the minimum allowed value.																												
16	<i>C</i> exceeded the maximum allowed value.																												
32	<i>C</i> undercut the minimum allowed value.																												
64	Execution denied (possibly a <b>BUSY</b> or <b>INTERNAL-BUSY</b> error; for exact reason see <b>get_last_error</b> ).																												
11	Access denied.																												
12	<b>Option "3D"</b> is not enabled.																												
13	No 3D correction table is currently assigned.																												
14	RTC6 board driver not found.																												
15	Invalid table number (> 8).																												



Ctrl Command	load_z_table_no
Comments	<ul style="list-style-type: none"> <li>• <b>load_z_table_no</b> is only needed for re-calibrating the z axis in a 3-axis scan system. For adjusting corresponding coefficients see <a href="#">Section "Checking the z axis Calibration", page 159</a>. Both positive and negative coefficients can be specified. The coefficients should preferably be chosen so that all z output values <math>z_{out} = A + Bz + Cz^2</math> lie within the range [-32,768...+32,767].</li> <li>• Prior to the next list command that directly follows <b>load_z_table_no</b>, a smooth transition from the last Z position to the changed position is performed at jump speed. You can also immediately force this by <a href="#"><b>select_cor_table</b></a>. This way, time delays can be avoided during an external start.</li> <li>• Coefficients A, B and C can be queried from the loaded 3D correction table by <a href="#"><b>get_table_para</b></a> (and from the currently assigned 3D correction table by <a href="#"><b>get_head_para</b></a>).</li> <li>• By <b>load_z_table_no</b>, the three coefficients A, B, C can be assigned to the 3D correction table No. When <a href="#"><b>select_cor_table</b></a> is executed, they are switched as well.</li> <li>• If No= currently assigned table number, the current values are overwritten as well and thus immediately makes them available, given no list is being processed. Otherwise, they are only saved and are only available after a <a href="#"><b>select_cor_table</b></a> or <a href="#"><b>select_cor_table_list</b></a>.</li> <li>• If no list is being processed, No = 0 merely overwrites the current values. <code>load_z_table_no( A, B, C, 0 )</code> is synonymous with <code>load_z_table( A, B, C )</code>. The values are lost after a <a href="#"><b>select_cor_table</b></a> or <a href="#"><b>select_cor_table_list</b></a>.</li> <li>• For No = 0, see also other comments at <a href="#"><b>load_z_table</b></a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 611, OUT 611, RBF 616.
References	<a href="#"><b>load_z_table</b></a> , <a href="#"><b>get_z_distance</b></a> , <a href="#"><b>read_abc_from_file</b></a> , <a href="#"><b>write_abc_to_file</b></a> , <a href="#"><b>select_cor_table</b></a>



<b>Normal List Command</b>	<b>long_delay</b>
<b>Function</b>	Pauses further processing of the list for the specified time.
<b>Call</b>	<code>long_delay( Delay )</code>
<b>Parameters</b>	<p>Delay      Delay time. In bits.            As an unsigned 32-bit value.  <i>1 bit equals 10 µs.</i>            Allowed value range: <math>0 \leq \text{delay} \leq (2^{32}-1)</math>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>long_delay</b> (if applicable) switches off the signals for “laser active” operation after a LaserOff delay, waits for a possible scanner delay and pauses further processing of the list for the specified time.</li> <li>• <b>long_delay</b> should always be called after changing the lamp current of a YAG laser to obtain a constant laser power.</li> <li>• <b>list_nop</b> corresponds to <code>long_delay( 1 )</code>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	–



<b>Normal List Command</b>	<b>mark_abs</b>
<b>Function</b>	Moves the laser focus at mark speed along a 2D vector from the current position to the specified position (absolute coordinate values) within a 2D image field.
<b>Call</b>	<code>mark_abs( X, Y )</code>
<b>Parameters</b>	<p>X      Absolute x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the mark speed has not been previously explicitly set by <a href="#">set_mark_speed</a> or <a href="#">set_mark_speed_ctrl</a>, then the marking is executed at a predefined mark speed of 1,000 bits/ms.</li> <li>The “laser active” laser control signals are automatically turned on at the beginning of the command (or remain on after a directly preceding mark or arc command). The defined scanner and laser delays are thereby taken into account, see <a href="#">Chapter 7.2 “Delay Settings – Coordinating Scan Head Control and Laser Control”, page 134</a>. Note that other delays are executed in Sky Writing mode, see <a href="#">Chapter 7.2.4 “Sky Writing”, page 149</a>. Exception: zero-length [*]mark[*] commands, see <a href="#">Section “Notes”, page 138</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the values specified for X and Y by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mark_speed</a> , <a href="#">set_scanner_delays</a> , <a href="#">mark_rel</a> , <a href="#">arc_abs</a> , <a href="#">timed_mark_abs</a>



<b>Normal List Command</b>	<b>mark_abs_3d</b>
<b>Function</b>	Moves the laser focus at mark speed along a 3D vector from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>mark_abs_3d</b> has the same effect as <b>mark_abs</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
<b>Call</b>	<b>mark_abs_3d( X, Y, Z )</b>
<b>Parameters</b>	<p>X      Absolute x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p> <p>Z      Like X (analogously), except Allowed value range: [-524,288...+524,287].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>mark_abs_3d</b> functions similarly to the <b>mark_abs</b> command (see comments there).</li> </ul>
<b>RTC4→RTC6</b>	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the values specified for X, Y and Z by 16. The allowed value range decreases accordingly.
<b>RTC5→RTC6</b>	Unchanged functionality. In addition: increased value range.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the value specified for Z by 16. The allowed value range decreases accordingly.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>mark_abs</b> , <b>mark_rel_3d</b> , <b>timed_mark_abs_3d</b>



<b>Undelayed Short List Command</b>	<b>mark_char</b>
Function	Marks an indexed character.
Call	<code>mark_char( Char )</code>
Parameters	<p>Char      Index of the indexed character to be marked.            As an unsigned 32-bit value.            Allowed value range: [0...1023]).            The following applies: Char = character set number × 256 + ASCII number of the character (character sets are numbered 0...3).</p>
Comments	<ul style="list-style-type: none"> <li>The <b>mark_char</b> command reads the indexed character's starting address from the internal management table based on the supplied index and then calls the command <b>list_call</b> (see also the comments there), which then starts the corresponding command list.</li> <li>The <b>mark_char</b> command starts indexed characters in protected memory (that were loaded and/or referenced by <b>load_char</b>, <b>load_disk</b> or <b>copy_dst_src</b>) as well as indexed subroutines in the unprotected list area (that were referenced as characters by <b>set_char_pointer</b> or <b>copy_dst_src</b>).</li> <li>If no character is referenced for the supplied index, then the jump is suppressed and execution continues at the command located after the calling position. In some circumstances, a <b>list_continue</b> might be executed, see <a href="#">Section "Normal, Short, Variable and Multiple List Commands", page 288</a>.</li> <li><b>get_char_pointer( Char )</b> can be used to determine whether a character has been referenced for a particular index. If no character has been referenced, this command returns the value “-1” (d.h. <math>2^{32}-1</math>).</li> <li>If <code>Char &gt; 1023</code>, then <b>mark_char</b> is, already during loading, replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>Absolute vector and arc commands execute absolutely after being called with <b>mark_char</b>. If the character needs to execute at various locations within the image field, then either the command list can only contain relative mark, arc or jump commands or <b>mark_char_abs</b> must be used instead.</li> <li>The called character should not contain <b>mark_text</b> commands that also contain this character. Such text is <i>not</i> marked. The called character itself then might not be complete.</li> <li>See also <a href="#">Chapter 6.5.2 "Character Sets and Text Strings", page 108</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>mark_char_abs</b> , <b>mark_text</b> , <b>set_char_pointer</b> , <b>get_char_pointer</b>



<b>Undelayed Short List Command</b>	<b>mark_char_abs</b>
Function	Marks an indexed character. In the called command list (see below), any absolute vector and arc commands receive an offset (based on the current coordinates at the time of the call).
Call	<code>mark_char_abs( Char )</code>
Parameters	Char      Index of the indexed character to be marked. As an unsigned 32-bit value. Allowed value range: [0...1023]). The following applies: Char = character set number * 256 + ASCII number of the character (character sets are numbered 0...3).
Comments	<ul style="list-style-type: none"> <li>• <b>mark_char_abs</b> reads the indexed character's starting address from the internal management table based on the supplied index and then calls the command <b>list_call_abs</b> (see also the comments there), which then starts the corresponding command list.</li> <li>• If the command list of the called character contains no absolute commands, then there is no difference between <b>mark_char_abs</b> and <b>mark_char</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>mark_char</b>



<b>Normal List Command</b>	<b>mark_date</b>
<b>Function</b>	Marks a part of the date previously stored by <b>time_fix</b> , <b>time_fix_f</b> or <b>time_fix_f_off</b> in the selected format at the current position.
<b>Call</b>	<code>mark_date( Part, Mode )</code>
<b>Parameters</b>	<p><b>Part</b>      Specifies which part of the date should be marked.</p> <ul style="list-style-type: none"> <li>= 0: Year (only the last two digits).</li> <li>= 1: Month (in customer specific notation).</li> <li>= 2: Day (corresponding to the normal Gregorian date).</li> <li>= 3: Day-of-the-week (in customer specific notation).</li> <li>= 4: Julian day (see also <b>time_fix_f_off</b>).</li> <li>= 5: Year (four digits).</li> <li>= 6: Month as numerals (January = 1, ...).</li> <li>= 7: Day-of-the-week as numerals (Sunday = 1, ...).</li> </ul> <p>As an unsigned 32-bit value. Allowed value range: [0...7].</p> <p><b>Mode</b>      Selection of the format.  As an unsigned 32-bit value.  Allowed value range: [0...3].</p> <p>a) Only affects Part = 2, 4, 6 or 7:  The number of leading zeros in the day number.</p> <p>Bit #0 = 0: Leading zeros are suppressed.</p> <p>Bit #0 = 1: Day of the month (Part = 2), always two digits.  Day of the year (Part = 4), always three digits.  Month (Part = 6), always two digits.  Day-of-the-week (Part = 7), always two digits.</p> <p>b) Only affects Part = 0, 2 or 4...7:  Desired character set for marking digits.</p> <p>Bit #1 = 0: The indexed text string for digits [0...9], as defined by <b>load_text_table</b> or <b>set_text_table_pointer</b>, is marked.</p> <p>Bit #1 = 1: The indexed characters for digits [0...9], as defined by <b>load_char</b> or <b>set_char_pointer</b>, are marked. The desired character set can be specified with <b>select_char_set</b>.</p> <p>For Part = 1 or 3, days of the month or days of the week are marked by indexed text strings (Index = 10...28) defined with <b>load_text_table</b> or <b>set_text_table_pointer</b> (here, the parameter Mode is ignored). For marking as numerals, also Part = 6 or 7 can be used (then Mode is considered).</p>



<b>Normal List Command</b>	<b>mark_date</b>
Comments	<ul style="list-style-type: none"> <li>Before marking dates (after every boot-up), the RTC6 and PC times should be synchronized (see <a href="#">time_update</a>) and the current (to be marked) date value should be stored with <a href="#">time_fix</a>, <a href="#">time_fix_f</a> or <a href="#">time_fix_f_off</a> (see <a href="#">Chapter 7.5 "Marking Dates, Times and Serial Numbers", page 197</a>).</li> <li>The complete date can be marked by multiple calls of the <b>mark_date</b> command.</li> <li>The <b>mark_date</b> command reads (according to the stored date and according to the selected date part) the starting address of the corresponding indexed text string or character from the internal management table and then calls the command <a href="#">list_call</a> (see also the comments there) an appropriate number of times, which then starts the corresponding command list. The command lists must contain marking instructions for digits 0...9 and for the month and day-of-the-week designations (see <a href="#">Section "Defining Indexed Text Strings for Time, Date and Serial Number", page 109</a>). Non-defined text strings or characters are ignored (that is, not marked). The called indexed text strings can also contain calls to indexed characters (<a href="#">mark_char</a> or <a href="#">mark_char_abs</a>) and complete texts (<a href="#">mark_text</a> or <a href="#">mark_text_abs</a>). In the latter case, the character set can be switched when needed (before marking by <b>mark_date</b>) with <a href="#">select_char_set</a> (see also <a href="#">Chapter 6.5.2 "Character Sets and Text Strings", page 108</a>).</li> <li>If <code>Part &gt; 7</code> and/or <code>Mode &gt; 3</code>, then <b>mark_date</b> is, already during loading, replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</li> <li>Absolute vector and arc commands execute absolutely after being called with <b>mark_date</b>. If date markings need to execute at various locations within the image field, then the corresponding indexed text strings (or characters) can only contain relative mark, arc or jump commands or <a href="#">mark_date_abs</a> must be used instead.</li> <li>When marking Gregorian dates or Julian days, the transition to the next day occurs at 24:00 o'clock. Leap years are represented in both date styles.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.  <b>mark_date</b> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">time_fix</a> , <a href="#">time_fix_f</a> , <a href="#">time_fix_f_off</a> , <a href="#">load_text_table</a> , <a href="#">set_text_table_pointer</a> , <a href="#">mark_date_abs</a>



<b>Normal List Command</b>	<b>mark_date_abs</b>
<b>Function</b>	Marks a part of the date previously stored by <b>time_fix</b> , <b>time_fix_f</b> or <b>time_fix_f_off</b> in the selected format at the current position. In the called indexed text strings or characters (see below), any absolute vector and arc commands receive an offset (based on the current coordinates at the time of the call).
<b>Call</b>	<code>mark_date_abs( Part, Mode )</code>
<b>Parameters</b>	Part      See <a href="#">mark_date</a> .
	Mode     See <a href="#">mark_date</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>mark_date_abs</b> has the same effect as <b>mark_date</b>; however, internal calling of the indexed text strings (or characters) is by <b>list_call_abs</b> instead of <b>list_call</b>. If the command lists of the called indexed text strings (or characters) contain no absolute commands, then there is no difference between <b>mark_date_abs</b> and <b>mark_date</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">mark_date</a>



<b>Normal List Command</b>	<b>mark_ellipse_abs</b>				
<b>Function</b>	Moves the laser focus at mark speed along an elliptical arc around the specified midpoint (absolute coordinate values) within a 2D image field.				
<b>Call</b>	<code>mark_ellipse_abs( X, Y, Alpha )</code>				
<b>Parameters</b>	<table> <tr> <td>X</td><td>Absolute coordinates of the ellipse midpoint. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td>Alpha</td><td>Angle in degrees between elliptical half-axis a (defined by <a href="#">set_ellipse</a>) and the x axis (the angle is referenced to the positive x direction, positive angle values correspond to counterclockwise angles). As a 64-bit IEEE floating point value. Alpha gets normalized to the value range [0...&lt;360°].</td></tr> </table>	X	Absolute coordinates of the ellipse midpoint. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.	Alpha	Angle in degrees between elliptical half-axis a (defined by <a href="#">set_ellipse</a> ) and the x axis (the angle is referenced to the positive x direction, positive angle values correspond to counterclockwise angles). As a 64-bit IEEE floating point value. Alpha gets normalized to the value range [0...<360°].
X	Absolute coordinates of the ellipse midpoint. In bits. As signed 32-bit values. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.				
Alpha	Angle in degrees between elliptical half-axis a (defined by <a href="#">set_ellipse</a> ) and the x axis (the angle is referenced to the positive x direction, positive angle values correspond to counterclockwise angles). As a 64-bit IEEE floating point value. Alpha gets normalized to the value range [0...<360°].				
<b>Comments</b>	<ul style="list-style-type: none"> <li>The parameters for <b>mark_ellipse_abs</b> only determine the position and orientation of the to-be-executed arc. Before execution of <b>mark_ellipse_abs</b>, its shape must have been specified by <a href="#">set_ellipse</a>. For descriptions of the individual parameters, see also <a href="#">Section "Ellipse Commands", page 127</a>.</li> <li>If the arc starting point defined by <b>mark_ellipse_abs</b> and <a href="#">set_ellipse</a> does not equal the current position, then a "Hard jump" to the starting point is executed prior to marking, see also notes in <a href="#">Section "Ellipse Commands", page 127</a>.</li> <li>See also all comments for <a href="#">arc_abs</a>.</li> </ul>				
RTC4→RTC6	New command.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified ellipse midpoint coordinate values by 16. The allowed value range decreases accordingly.				
RTC5→RTC6	Unchanged functionality. In addition: increased value range.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">set_ellipse</a> , <a href="#">mark_ellipse_rel</a> , <a href="#">set_mark_speed</a> , <a href="#">set_scanner_delays</a> , <a href="#">arc_abs</a>				



<b>Normal List Command</b>	<b>mark_ellipse_rel</b>
<b>Function</b>	Moves the laser focus at mark speed along an elliptical arc around the specified midpoint (relative coordinate values) within a 2D image field.
<b>Call</b>	<code>mark_ellipse_rel( dx, dy, Alpha )</code>
<b>Parameters</b>	<p>dx      <i>Relative coordinates of the ellipse midpoint. In bits.</i>            dy      <i>As signed 32-bit values.</i>  <i>Allowed value range: [-268,435,456...+268,435,455].</i>  <i>Out-of-range values are clipped to the boundary values.</i></p> <p>Alpha    Angle in degrees between elliptical half-axis a (defined by <a href="#">set_ellipse</a>) and the x axis (see <a href="#">mark_ellipse_abs</a>).  <i>As a 64-bit IEEE floating point value.</i></p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the ellipse midpoint are to be specified as relative coordinates with respect to the current position. Otherwise, <b>mark_ellipse_rel</b> is identical to <b>mark_ellipse_abs</b> (see comments there).</li> </ul>
RTC4→RTC6	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified ellipse midpoint coordinate values by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">mark_ellipse_abs</a> , <a href="#">set_ellipse</a>



<b>Normal List Command</b>	<b>mark_rel</b>
<b>Function</b>	Moves the laser focus at mark speed along a 2D vector from the current position to the specified position (relative coordinate values) within a 2D image field.
<b>Call</b>	<code>mark_rel( dx, dy )</code>
<b>Parameters</b>	<p><code>dx</code>      Relative x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><code>dy</code>      Like <code>dx</code> (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>mark_rel</b> is identical to <b>mark_abs</b> (see comments there).</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the values specified for <code>dx</code> and <code>dy</code> by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">mark_abs</a> , <a href="#">mark_rel_3d</a> , <a href="#">timed_mark_rel</a>



<b>Normal List Command</b>	<b>mark_rel_3d</b>
<b>Function</b>	Moves the laser focus at mark speed along a 3D vector from the current position to the specified position (relative coordinate values) within a 3D process volume.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>mark_rel_3d</b> has the same effect as <b>mark_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
<b>Call</b>	<b>mark_rel_3d( dx, dy, dz )</b>
<b>Parameters</b>	<p><b>dx</b>      Relative x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>dy</b>      Like <b>dx</b> (analogously).</p> <p><b>dz</b>      Like <b>dx</b> (analogously), except Allowed value range: [-524,288...+524,287].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>mark_rel_3d</b> is identical to <b>mark_abs_3d</b> (see comments there).</li> </ul>
<b>RTC4→RTC6</b>	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the values specified for <b>dx</b> , <b>dy</b> and <b>dz</b> by 16. The allowed value range decreases accordingly.
<b>RTC5→RTC6</b>	Unchanged functionality. In addition: increased value range.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the value specified for <b>dz</b> by 16. The allowed value range decreases accordingly.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>mark_abs_3d</b> , <b>mark_abs</b> , <b>mark_rel</b> , <b>timed_mark_rel_3d</b>



<b>Normal List Command</b>	<b>mark_serial</b>
<b>Function</b>	Marks the current serial number of the serial-number-set most recently selected by <b>select_serial_set_list</b> (or of serial-number-set 0 after <b>load_program_file</b> ) in the selected format at the current position. Afterward the serial number is (optionally) automatically incremented.
<b>Call</b>	<code>mark_serial( Mode, Digits )</code>
<b>Parameters</b>	<p><b>Mode</b> Selection of the serial number format and (de)activation of automatic serial number incrementing. As an unsigned 32-bit value.  <math>\text{Mode} = 20 \times M_1 + 10 \times M_2 + M_3</math>          Allowed range for <math>M_1</math>: [0, 1], for <math>M_2</math>: [0, 1], for <math>M_3</math>: [0...2].</p> <p>a) Selection of the character set for digit marking.</p> <ul style="list-style-type: none"> <li><math>M_1 = 0</math>: The indexed text string for digits [30...39], as defined by <b>load_text_table</b> or <b>set_text_table_pointer</b>, is marked.</li> <li><math>M_1 = 1</math>: The indexed characters for digits [0...9], as defined by <b>load_char</b> or <b>set_char_pointer</b>, are marked. The desired character set can be selected (previously) with <b>select_char_set</b>.</li> </ul> <p>b) Incrementing of the serial number after marking.</p> <ul style="list-style-type: none"> <li><math>M_2 = 0</math>: The serial number is incremented after marking.</li> <li><math>M_2 = 1</math>: The serial number is <i>not</i> incremented after marking.</li> </ul> <p>c) Marking of leading zeros.</p> <ul style="list-style-type: none"> <li><math>M_3 = 0</math>: Leading zeros are marked as zeros. Dependent on <math>M_1</math> the corresponding indexed text string definition (Index = 30) or the indexed character definition '0' is used.</li> <li><math>M_3 = 1</math>: Leading zeros are suppressed (left-justified marking).</li> <li><math>M_3 = 2</math>: Leading zeros are marked as blank characters (right-justified marking). Dependent on <math>M_1</math> the corresponding indexed text string definition (Index = 29) or the indexed character definition '' is used.</li> </ul> <p><b>Digits</b> Number [0-12] of to-be-marked digits. As an unsigned 32-bit value.          Allowed value range: [0-12]. Larger values are clipped.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The first serial number to be marked must have been previously specified by <b>set_serial</b>, <b>set_serial_step</b> or <b>set_serial_step_list</b>; otherwise, the starting serial number is 0. The starting serial number can have a maximum length of 10 digits.</li> <li>With every call of <b>mark_serial</b>, the serial number is formatted in accordance with <math>M_3</math> and when <math>M_2 = 0</math> it is automatically incremented before the actual marking. Here, the increment size is 1 unless otherwise specified by <b>set_serial_step</b> or <b>set_serial_step_list</b>.</li> <li>The current serial number can be queried with <b>get_list_serial</b>, for example, after an aborted list to determine if the current number was incremented or not.</li> <li>If the incremented serial number exceeds <math>10^{\text{digits}}</math>, then marking begins again at 0. The control command <b>set_max_counts</b> allows specification of the maximum number of external starts and thus the maximum number of markings. Here, all markings of all serial-number-sets contribute jointly to the count.</li> </ul>



<b>Normal List Command</b>	<b>mark_serial</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If digits = 0, then a "markless" marking is executed. If M<sub>2</sub> = 0, then the serial number is incremented by 1 (any increment size defined by <b>set_serial_step</b> or <b>set_serial_step_list</b> are not used in this case!). This can be useful, if a single serial number is to be omitted (repeat n times if necessary; n = increment), but can also be used to indirectly increase the serial number by an additional increment.</li> <li>For each to-be-marked serial number digit, the <b>mark_serial</b> command reads the starting address of the corresponding indexed text string (or – for M<sub>1</sub> = 1 – of the corresponding indexed character) from the internal management table and then calls the command <b>list_call</b> (see also the comments there) an appropriate number of times, which starts the corresponding command list. The command lists must contain marking instructions for digits 0...9 (see <a href="#">Section "Defining Indexed Text Strings for Time, Date and Serial Number", page 109</a>). Non-defined text strings or characters are ignored (that is, not marked). The called indexed text strings can also contain calls to indexed characters (<b>mark_char</b> or <b>mark_char_abs</b>) and complete texts (<b>mark_text</b> or <b>mark_text_abs</b>). In the latter case, the character set can be switched if needed (before marking by <b>mark_serial</b>) with <b>select_char_set</b> (see also <a href="#">Chapter 6.5.2 "Character Sets and Text Strings", page 108</a>).</li> <li>For invalid Mode values, the <b>mark_serial</b> is, already during loading, replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>Absolute vector and arc commands execute absolutely after being called with <b>mark_serial</b>. If serial number markings need to execute at various locations within the image field, then the corresponding indexed text strings (or characters) can only contain relative mark, arc or jump commands or <b>mark_serial_abs</b> must be used instead.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. <b>mark_serial</b> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#"><b>set_serial</b></a> , <a href="#"><b>set_serial_step</b></a> , <a href="#"><b>set_serial_step_list</b></a> , <a href="#"><b>get_list_serial</b></a> , <a href="#"><b>set_max_counts</b></a> , <a href="#"><b>get_counts</b></a> , <a href="#"><b>load_text_table</b></a> , <a href="#"><b>set_text_table_pointer</b></a> , <a href="#"><b>mark_serial_abs</b></a>



<b>Normal List Command</b>	<b>mark_serial_abs</b>				
<b>Function</b>	Marks the current serial number of the serial-number-set most recently selected by <b>select_serial_set_list</b> (or of serial-number-set 0 after <b>load_program_file</b> ) in the selected format at the current position. In the called indexed text strings or characters (see below), any absolute vector and arc commands receive an offset (based on the current coordinates at the time of the call). Afterward the serial number is (optionally) automatically incremented.				
<b>Call</b>	<code>mark_serial_abs( Mode, Digits )</code>				
<b>Parameters</b>	<table> <tr> <td>Mode</td> <td>See <b>mark_serial</b>.</td> </tr> <tr> <td>Digits</td> <td>See <b>mark_serial</b>.</td> </tr> </table>	Mode	See <b>mark_serial</b> .	Digits	See <b>mark_serial</b> .
Mode	See <b>mark_serial</b> .				
Digits	See <b>mark_serial</b> .				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>mark_serial_abs</b> has the same effect as <b>mark_serial</b>; however, internal calling of the indexed text strings (or characters) is by <b>list_call_abs</b> instead of <b>list_call</b>. If the command lists of the called indexed text strings (or characters) contain no absolute commands, then there is no difference between <b>mark_serial_abs</b> and <b>mark_serial</b>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<b>mark_serial</b>				



<b>Variable List Command</b>	<b>mark_text</b>
<b>Function</b>	Marks a \0-terminated string.
<b>Call</b>	<code>mark_text( Text )</code>
<b>Parameters</b>	<p>Text      PC memory address of the first character (byte) of the to-be-marked text string. As a pointer to a \0-terminated string.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The to-be-marked text (character sequence, byte array, \0-terminated string) must be terminated with a \0 character (0 byte, <b>NULL</b>). The \0 character itself is not marked.</li> <li>When a <b>mark_text</b> is loaded, the to-be-marked text (if more than 12 characters in length, \0 not included) is split into blocks of 12 characters, with each block receiving its own <b>mark_text</b> command in the list memory (make sure that no undesired memory overflow of the respective memory area occurs). During processing of the individual <b>mark_text</b> commands, the corresponding <b>mark_char</b> commands (indexed characters) are executed in accordance with the selected character set.</li> <li>The desired character set can be selected prior to <b>mark_text</b> by <b>select_char_set</b>. For the default setting, character set 0 is used. If the <b>select_char_set</b> is used within a called indexed character, then all subsequently called indexed characters are marked using this character set.</li> <li>If the end of a list ("List 1" or "List 2") is reached during loading of a <b>mark_text</b>, then loading continues at the start of the corresponding list. In contrast, loading in the protected area (as part of an indexed subroutine) is aborted (<b>get_last_error</b> return code <b>RTC6_REJECTED</b>) and the indexed subroutine is not stored.</li> <li>Absolute vector and arc commands execute absolutely after being called with <b>mark_text</b>. If the text string needs to execute at various locations within the image field, then either the indexed character definitions can only contain relative mark, arc or jump commands or <b>mark_text_abs</b> must be used instead.</li> <li><b>mark_text</b> should not be used within an indexed character definition. The corresponding text is <i>not</i> marked and the indexed character is therefore not fully processed.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>mark_text_abs</b>



<b>Variable List Command</b>	<b>mark_text_abs</b>
Function	Marks a \0-terminated string. In the called indexed characters (see below), any absolute vector and arc commands receive an offset (based on the current coordinates at the time of the call).
Call	<code>mark_text_abs( Text )</code>
Parameters	Text      PC memory address of the first character (byte) of the to-be-marked text string. As a pointer to a \0-terminated string.
Comments	<ul style="list-style-type: none"> <li>During processing of the individual <b>mark_text_abs</b> commands, the corresponding <b>mark_char_abs</b> commands (indexed characters) are executed in accordance with the selected character set.</li> <li>If the command list of the called indexed character contains no absolute commands, then there is no difference between <b>mark_text_abs</b> and <b>mark_text</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>mark_text</b>



<b>Normal List Command</b>	<b>mark_time</b>
<b>Function</b>	Marks a part of the time previously stored by <b>time_fix</b> , <b>time_fix_f</b> or <b>time_fix_f_off</b> in the specified format at the current position.
<b>Call</b>	<code>mark_time( Part, Mode )</code>
<b>Parameters</b>	<p><b>Part</b>      Specifies which part of the time to mark.</p> <ul style="list-style-type: none"> <li>= 0: Hours (24-h time, no a.m./p.m.)</li> <li>= 1: Minutes</li> <li>= 2: Seconds</li> <li>= 3: Hours (12-h time, no a.m./p.m.)</li> <li>= 4: a.m./p.m. (automatically switched in accordance with 24-h time)</li> </ul> <p>As an unsigned 32-bit value. Allowed value range: [0...4].</p> <p><b>Mode</b>      Format choice.</p> <p>As an unsigned 32-bit value.</p> <p>Allowed value range: [0...3], only affects Part = 0...3).</p> <p>a) Number of leading zeros:</p> <p>Bit #0 = 0: Leading zeros are suppressed.</p> <p>Bit #0 = 1: Always two digits.</p> <p>b) Character set choice for marking of digits:</p> <p>Bit #1 = 0: The indexed text strings for digits [0...9], as defined by <b>load_text_table</b> or <b>set_text_table_pointer</b>, are marked.</p> <p>Bit #1 = 1: The indexed characters for digits [0...9], as defined by <b>load_char</b> or <b>set_char_pointer</b>, are marked.</p> <p>The desired character set can be specified with <b>select_char_set</b>.</p> <p>a.m./p.m. (Part = 4) can only be marked in accordance with the indexed text strings (Index = 40, 41) defined by <b>load_text_table</b> or <b>set_text_table_pointer</b>. Here, the parameter Mode is ignored.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Before marking times (after every boot-up), the RTC6 and PC times should be synchronized (see <b>time_update</b>) and the current (to be marked) time should be stored with <b>time_fix</b>, <b>time_fix_f</b> or <b>time_fix_f_off</b>, see Chapter 7.5 "Marking Dates, Times and Serial Numbers", page 197.</li> <li>• The complete time can be marked by multiple calls of the <b>mark_time</b> command.</li> </ul>



<b>Normal List Command</b>	<b>mark_time</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>mark_time</b> reads (according to the stored time and according to the selected time part) the starting address of the corresponding indexed text string (or – for Part = 0...3 and Mode = 2 or 3 – of the corresponding indexed character) from the internal management table and then calls the command <b>list_call</b> (see also the comments there) an appropriate number of times, which starts the corresponding command list. The command lists must contain marking instructions for digits 0...9 and for a.m./p.m., see <a href="#">Section "Defining Indexed Text Strings for Time, Date and Serial Number", page 109</a>. Non-defined text strings or characters are ignored (that is, not marked). The called indexed text strings can also contain calls to indexed characters (<b>mark_char</b> or <b>mark_char_abs</b>) and complete texts (<b>mark_text</b> or <b>mark_text_abs</b>). In the latter case, the character set can be switched, if needed (before marking with <b>mark_time</b>), by <b>select_char_set</b>, see also <a href="#">Chapter 6.5.2 "Character Sets and Text Strings", page 108</a>.</li> <li>• If Part &gt; 4 and/or Mode &gt; 3, then <b>mark_time</b> is, already during loading, replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>• Absolute vector and arc commands execute absolutely after being called with <b>mark_time</b>. If time markings need to execute at various locations within the image field, then the corresponding indexed text strings (or characters) can only contain relative mark, arc or jump commands or <b>mark_time_abs</b> must be used instead.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. <b>mark_time</b> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">time_fix</a> , <a href="#">time_fix_f</a> , <a href="#">time_fix_f_off</a> , <a href="#">load_text_table</a> , <a href="#">set_text_table_pointer</a> , <a href="#">mark_time_abs</a>



<b>Normal List Command</b>	<b>mark_time_abs</b>				
<b>Function</b>	Marks a part of the time previously stored by <b>time_fix</b> , <b>time_fix_f</b> or <b>time_fix_f_off</b> in the specified format at the current position. In the called indexed text strings or characters (see below), any absolute vector and arc commands receive an offset (based on the current coordinates at the time of the call).				
<b>Call</b>	<code>mark_time_abs( Part, Mode )</code>				
<b>Parameters</b>	<table> <tr> <td>Part</td> <td>See <a href="#">mark_time</a>.</td> </tr> <tr> <td>Mode</td> <td>See <a href="#">mark_time</a>.</td> </tr> </table>	Part	See <a href="#">mark_time</a> .	Mode	See <a href="#">mark_time</a> .
Part	See <a href="#">mark_time</a> .				
Mode	See <a href="#">mark_time</a> .				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>mark_time_abs</b> has the same effect as <b>mark_time</b>; however, internal calling of the indexed text strings (or characters) is by <b>list_call_abs</b> instead of <b>list_call</b>. If the command lists of the called indexed text strings (or characters) contain no absolute commands, then there is no difference between <b>mark_time_abs</b> and <b>mark_time</b>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">mark_time</a>				



<b>Ctrl Command</b>	<b>master_slave_config</b>
<b>Function</b>	Sets settings for the master-slave interface of an RTC6 board.
<b>Call</b>	master_slave_config( Config )
<b>Parameters</b>	<p>Config As an unsigned 32-bit value.</p> <p>Bit #0 Suppression of /Slave-START signals and /Slave-STOP signals.</p> <ul style="list-style-type: none"> <li>= 0: /Slave-START and /Slave-STOP signals via the master-slave interface are received and processed.</li> <li>= 1: /Slave-START and /Slave-STOP signals received by this board via the master-slave interface are ignored on this board, but are forwarded to other boards of the master-slave interface.</li> </ul> <p>Bit #1 /STOP in case of master-slave faults.</p> <ul style="list-style-type: none"> <li>= 0: The board does not react explicitly to a fault in the master-slave connection. Any effects are undefined.</li> <li>= 1: Triggers a /STOP, if the connection to the master card is interrupted. This occurs, for example, if <b>load_program_file</b> is executed on a card of the already synchronized master-slave chain, the master-slave cable is removed or the signal of the master-slave interface is electromagnetically disturbed.</li> </ul> <p>Bit #2 Forwarding /STOP in case of master-slave faults.</p> <ul style="list-style-type: none"> <li>= 0: A /STOP is not forwarded to other boards.</li> <li>= 1: If an error of the master-slave connection is detected and a /STOP is triggered (see Bit #1), this /STOP is forwarded to all still connected cards of the master-slave chain (also upwards to a master card).</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For usage of <b>master_slave_config</b>, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 114</a>.</li> <li>• The master-slave interface of a board should normally only be configured after successful synchronization.</li> <li>• If Bit #0 is already set before <b>sync_slaves</b>, this board cannot be synchronized to a master board with <b>sync_slaves</b>.</li> <li>• In certain other malfunction scenarios, fault-free operation may no longer be possible. In this case, a /STOP is always performed on the affected board, even if Bit #1 = 0.</li> <li>• If the connection to a master-slave synchronized board is disturbed, this can also result in further boards in the chain being disturbed, and thus execute a /STOP, even if Bit #2 is not set.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 609, OUT 609, RBF 613.
References	<a href="#">sync_slaves</a>



<b>Ctrl Command</b>	<b>mcbsp_init</b>				
<b>Function</b>	Defines the data delays for transmitting and receiving data by the <b>McBSP interface</b> , see also <b>Section "McBSP Interface", page 73.</b>				
<b>Call</b>	<code>mcbsp_init( XDelay, RDelay )</code>				
<b>Parameters</b>	<table> <tr> <td>XDelay</td> <td>Transmission delay. As an unsigned 32-bit value. Allowed value range: [0...2].</td> </tr> <tr> <td>RDelay</td> <td>Receiving delay. As an unsigned 32-bit value. Allowed value range: [0...2].</td> </tr> </table>	XDelay	Transmission delay. As an unsigned 32-bit value. Allowed value range: [0...2].	RDelay	Receiving delay. As an unsigned 32-bit value. Allowed value range: [0...2].
XDelay	Transmission delay. As an unsigned 32-bit value. Allowed value range: [0...2].				
RDelay	Receiving delay. As an unsigned 32-bit value. Allowed value range: [0...2].				
<b>Comments</b>	<ul style="list-style-type: none"> <li>For invalid parameter values, <b>mcbsp_init</b> is not executed (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>The initialized values (after program start) are <b>XDelay = RDelay = 1</b>.</li> <li>The signals and operating conditions of the <b>McBSP interface</b> are presented in <b>Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73.</b></li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<b>read_mcbsp, set_mcbsp_out, set_mcbsp_out_ptr, set_mcbsp_freq, mcbsp_init_spi</b>				



<b>Ctrl Command</b>	<b>mcbsp_init_spi</b>
<b>Function</b>	Has no effect in the RTC6 command set.
<b>Call</b>	<code>mcbsp_init_spi( ClockLevel, ClockDelay )</code>
<b>Parameters</b>	<p><code>ClockLevel</code> = 0: inactive low.                           &gt; 0: inactive high.                           As an unsigned 32-bit value.</p> <p><code>ClockDelay</code> = 0: Clock signal and data bits at the same time.                           &gt; 0: Clock signal is delayed a half period.                           As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73.</a></li> </ul>
<b>RTC4→RTC6</b>	New command.
<b>RTC5→RTC6</b>	<p>Changed functionality.</p> <ul style="list-style-type: none"> <li>• The RTC6 command set contains <code>mcbsp_init_spi</code>, but it has no effect. Execution of it is refused and the <code>get_last_error</code> return code is <code>RTC6_REJECTED</code>.</li> <li>• Notes on migrating the source code of RTC5 user programs, etc.: For hardware reasons, RTC6 boards are no longer configurable for SPI (Serial Peripheral Interface) functionality. You need to appropriately modify any such source code sections.</li> </ul>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#"><b>mcbsp_init, set_mcbsp_freq</b></a>



<b>Ctrl Command</b>	<b>measurement_status</b>	
<b>Function</b>	Returns the status of a measurement session started by <a href="#">set_trigger</a> or <a href="#">set_trigger4</a> and the current position of the measurement pointer.	
<b>Call</b>	<code>measurement_status( &amp;Busy, &amp;Pos )</code>	
<b>Returned parameter values</b>	Busy	Measurement status. As a pointer to an unsigned 32-bit value. > 0: A measurement session is currently in progress. = 0: No measurement session is currently in progress.
	Pos	Current position of the measurement pointer (within the RTC6 measurement data memory) [0...max. channel size, see <a href="#">set_trigger4</a> ]. As a pointer to an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>If a measurement session started with <a href="#">set_trigger</a> or <a href="#">set_trigger4</a> is no longer active, then <code>Pos+1</code> indicates the number of recorded data pairs up to termination ([0...max. channel size, see <a href="#">set_trigger4</a>]).</li> <li><code>Pos = 2<sup>32</sup>-1</code> indicates that data recording still has not occurred after <a href="#">load_program_file</a>.</li> <li>Stored data can be queried with <a href="#">get_waveform</a>.</li> <li>The status of a measurement session is reset by <a href="#">set_trigger( Period = 0 )</a>, <a href="#">set_trigger4( Period = 0 )</a>, <a href="#">stop_trigger</a> or <a href="#">stop_execution</a> (see <a href="#">set_trigger</a> comments).</li> </ul>	
RTC4→RTC6	Unchanged functionality.	
RTC5→RTC6	Unchanged functionality.	
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.	
<b>References</b>	<a href="#">set_trigger</a> , <a href="#">set_trigger4</a>	



<b>Normal List Command</b>	<b>micro_vector_abs</b>
<b>Function</b>	Moves the output point (of the laser focus) by a “ <b>Hard jump</b> ” (without split-up into microsteps) directly from the current position to the specified position (absolute coordinate values) within the 2D image field.
<b>Call</b>	<code>micro_vector_abs( X, Y, LasOn, LasOff )</code>
<b>Parameters</b>	<p>X            Absolute x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y            Like X (analogously).</p> <p>LasOn        LaserOn delay. <i>1 bit equals 1/64 µs</i>. As a signed 32-bit value. Allowed value range: [-1...+(2<sup>15</sup>-1)]. ≥ 0:        Delay is newly set. Values over (2<sup>15</sup>-1) are clipped. &lt; 0:        The previously set delay continues unaffected.</p> <p>LasOff      LaserOff delay. <i>1 bit equals 1/64 µs</i>. See <code>LasOn</code>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 8.8 “Microvector Commands”, page 259</a>.</li> <li>• Wobble is not taken into account, see <a href="#">2</a> in <a href="#">Chapter 7.3.6 “Output Values to the Scan System”, page 170</a>.</li> <li>• The microvector is always executed as a “<b>Hard jump</b>”.</li> <li>• By <code>LasOn ≥ 0</code> and <code>LasOff ≥ 0</code>, you can set a new LaserOn or LaserOff delay for each individual microvector. Each delay thereby gets set at the end of the clock period in which the new position actually gets outputted (this output clock cycle is delayed by a preceding scanner delay).</li> <li>• Negative values (<code>LasOn &lt; 0</code> and <code>LasOff &lt; 0</code>) do not affect laser delays. Hereby, the laser can remain on or off across multiple clock periods (mark and jump simulation).</li> <li>• Delays set with <code>LasOn</code> and <code>LasOff</code> only apply to the execution of microvectors. For execution of normal mark and arc commands (such as <code>mark_abs</code>), only the laser delays defined by <code>set_laser_delays</code> apply. <code>LasOn</code> and <code>LasOff</code> do not overwrite the laser delay parameter from <code>set_laser_delays</code>.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC6 multiplies the specified values for X and Y by 16, those for <code>LasOn</code> and <code>LasOff</code> by 64. The allowed value ranges decrease accordingly.</p>
RTC5→RTC6	<p>Unchanged functionality.</p> <p>In <a href="#">RTC5 Compatibility Mode</a>, the RTC6 multiplies the specified values for <code>LasOn</code> and <code>LasOff</code> by 32. The allowed value ranges decrease accordingly.</p>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">micro_vector_rel</a> , <a href="#">micro_vector_abs_3d</a>



<b>Normal List Command</b>	<b>micro_vector_abs_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) by a "Hard jump" (without split-up into microsteps) directly from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>micro_vector_abs_3d</b> has the same effect as <b>micro_vector_abs</b> .
<b>Call</b>	<b>micro_vector_abs_3d( X, Y, Z, LasOn, LasOff )</b>
<b>Parameters</b>	<p><b>X</b>      Absolute x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>Y</b>      Like <b>X</b> (analogously).</p> <p><b>Z</b>      Like <b>X</b> (analogously), except Allowed value range: [-524,288...+524,287].</p> <p><b>LasOn</b>    LaserOn delay. <i>1 bit equals 1/64 µs</i>. As a signed 32-bit value. Allowed value range: [-1...+(2<sup>15</sup>-1)]. ≥ 0:      Delay is newly set. Values over (2<sup>15</sup>-1) are clipped. &lt; 0:      The previously set delay continues unaffected.</p> <p><b>LasOff</b>    LaserOff delay. <i>1 bit equals 1/64 µs</i>. See <b>LasOn</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>micro_vector_abs_3d</b> functions similarly to <b>micro_vector_abs</b> (see comments there).</li> </ul>
<b>RTC4→RTC6</b>	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified values for <b>X</b>, <b>Y</b> and <b>Z</b> by 16, those for <b>LasOn</b> and <b>LasOff</b> by 64. The allowed value ranges decrease accordingly.</p>
<b>RTC5→RTC6</b>	<p>Unchanged functionality.</p> <p>In <b>RTC5 Compatibility Mode</b>, the RTC6 multiplies the specified value for <b>Z</b> by 16, those for <b>LasOn</b> and <b>LasOff</b> by 32. The allowed value ranges decrease accordingly.</p>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>micro_vector_abs</b> , <b>micro_vector_rel_3d</b>



<b>Normal List Command</b>	<b>micro_vector_rel</b>
<b>Function</b>	Moves the output point (of the laser focus) by a "Hard jump" (without split-up into microsteps) directly from the current position to the specified position (relative coordinate values) within the 2D image field.
<b>Call</b>	<code>micro_vector_rel( dX, dY, LasOn, LasOff )</code>
<b>Parameters</b>	<p><code>dX</code> Relative x coordinate of the micro vector end point. In bits. Otherwise, like <code>x</code> from <a href="#">micro_vector_abs</a>.</p> <p><code>dY</code> Relative y coordinate of the micro vector end point. In bits. Otherwise, like <code>y</code> from <a href="#">micro_vector_abs</a>.</p> <p><code>LasOn</code> Like <code>LasOn</code> from <a href="#">micro_vector_abs</a>.</p> <p><code>LasOff</code> Like <code>LasOff</code> from <a href="#">micro_vector_abs</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>micro_vector_rel</b> is identical to <a href="#">micro_vector_abs</a> (see comments there).</li> <li>The microvector is always executed as a "Hard jump".</li> </ul>
RTC4→RTC6	New command. <a href="#">RTC4 Compatibility Mode</a> : see <a href="#">micro_vector_abs</a> .
RTC5→RTC6	Unchanged functionality. <a href="#">RTC5 Compatibility Mode</a> : see <a href="#">micro_vector_abs</a> .
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">micro_vector_abs</a> , <a href="#">micro_vector_rel_3d</a>



<b>Normal List Command</b>	<b>micro_vector_rel_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) by a "Hard jump" (without split-up into microsteps) directly from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>micro_vector_rel_3d</b> has the same effect as <b>micro_vector_rel</b> .
<b>Call</b>	<code>micro_vector_rel_3d( dx, dy, dz, LasOn, LasOff )</code>
<b>Parameters</b>	<p><b>dx</b>      Relative x coordinate of the micro vector end point. In bits. Otherwise, like wie <b>x</b> von <b>micro_vector_abs_3d</b>.</p> <p><b>dy</b>      Relative y coordinate of the micro vector end point. In bits. Otherwise, like wie <b>y</b> von <b>micro_vector_abs_3d</b>.</p> <p><b>dz</b>      Relative z coordinate of the micro vector end point. In bits. Otherwise, like wie <b>z</b> von <b>micro_vector_abs_3d</b>.</p> <p><b>LasOn</b>    Like <b>LasOn</b> from <b>micro_vector_abs_3d</b>.</p> <p><b>LasOff</b>    Like <b>LasOff</b> from <b>micro_vector_abs_3d</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>micro_vector_rel_3d</b> is identical to <b>micro_vector_abs_3d</b> (see comments there).</li> <li>The microvector always executes as a "Hard jump".</li> </ul>
RTC4→RTC6	New command. <b>RTC4 Compatibility Mode:</b> see <b>micro_vector_abs_3d</b> .
RTC5→RTC6	Unchanged functionality. <b>RTC5 Compatibility Mode:</b> see <b>micro_vector_abs_3d</b> .
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>micro_vector_abs_3d</b> , <b>micro_vector_rel</b>

<b>Ctrl Command</b>	<b>move_to</b>
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>move_to</b> is described in the manual "Installation and Operation RTC Step Motor Extension for the RTC4 and RTC5 PC interface boards" (available in English only).</li> <li><b>move_to</b> has been introduced for the extension board "RTC4 STEP MOTOR EXTENSION" (#112097).</li> <li><b>move_to</b> is <i>not</i> supported by the extension board RTC5/6 varioSCAN 40 FLEX Extension (#128683).</li> </ul>



<b>Ctrl Command</b>	<b>number_of_correction_tables</b>
<b>Function</b>	Defines the maximum number of allowed correction tables.
<b>Call</b>	<code>number_of_correction_tables( Number )</code>
<b>Parameters</b>	<p>Number      Maximum number of allowed correction tables.            As an unsigned 32-bit value.            Allowed value range: [1...8].            Default after <a href="#">load_program_file</a>: 4.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <code>Number</code> outside the allowed value range, <code>number_of_correction_tables</code> is ignored (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</li> <li><code>number_of_correction_tables</code> serves to protect other commands (for example, such as <a href="#">load_correction_file</a> and <a href="#">select_cor_table</a>) from unwanted table numbers.</li> <li>Existing user programs do not have to be changed. The exception is, if user input is to be rejected (using explicit RTC6 error messages) in the future.</li> <li><code>number_of_correction_tables</code> refers only to subsequent command executions. Existing assignments of correction tables cannot be corrected automatically. This is particularly important, if RTC6 boards that have been initialized by other user programs are subsequently acquired.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Changed impacts (the RTC5-command has not been documented due to the limited number of users).
Version info	Available as of version DLL 609, OUT 609, RBF 613.
References	<a href="#">load_correction_file</a> , <a href="#">select_cor_table</a> , <a href="#">select_cor_table_list</a>



<b>Normal List Command</b>	<b>para_jump_abs</b>
<b>Function</b>	Moves the output point for the laser focus along a 2D vector at jump speed from the current position to the specified position (absolute coordinate values) within a 2D image field. Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Call</b>	<code>para_jump_abs( X, Y, P )</code>
<b>Parameters</b>	<p>X            Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y            Like X (analogously).</p> <p>P            End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If ""vector-controlled laser control"" has not been previously activated by <b>set_vector_control</b>, then <b>para_jump_abs</b> behaves like <b>jump_abs</b> (see comments there). The parameter P is then ignored.</li> <li>If ""vector-controlled laser control"" is activated, then simultaneously with the motion of the output point of the laser focus the signal parameter selected by <b>set_vector_control</b> is linearly varied from the last valid value to P (see <a href="#">Section "Vector-Defined Laser Control", page 195</a>).</li> <li>There is no abs mechanism for P. P is, if necessary, clipped to the maximum allowed value.</li> <li>If <b>para_jump_abs</b> is used along with position- and/or speed-dependent laser control for the same control parameter, then the current value of P is used as the basis of the 100% value for laser control, see <a href="#">Section "Vector-Defined Laser Control", page 195</a>.</li> </ul>
<b>RTC4→RTC6</b>	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for X and Y by 16. The allowed value ranges decrease accordingly. There is no <b>RTC4 Compatibility Mode</b> for the parameter P. The original RTC6 units must be used. Exception is Ctrl = 7 (Defocus): In <b>RTC4 Compatibility Mode</b> , with Ctrl = 7, the RTC6 multiplies the specified values for Value by 16. The allowed value ranges decrease accordingly.
<b>RTC5→RTC6</b>	Unchanged functionality.  Exception is Ctrl = 7 (Defocus): In <b>RTC5 Compatibility Mode</b> , with Ctrl = 7, the RTC6 multiplies the specified values for Value by 16. The allowed value ranges decrease accordingly.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">jump_abs</a> , <a href="#">set_vector_control</a> , <a href="#">para_jump_abs_3d</a> , <a href="#">para_jump_rel</a>



<b>Normal List Command</b>	<b>para_jump_abs_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) along a 3D vector at jump speed from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> . Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>para_jump_abs_3d</b> has the same effect as <b>para_jump_abs</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
<b>Call</b>	<b>para_jump_abs_3d( X, Y, Z, P )</b>
<b>Parameters</b>	<p>X      Absolute x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p> <p>Z      Like X (analogously), except Allowed value range: [-524,288...+524,287].</p> <p>P      End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>para_jump_abs_3d</b> functions similarly to the <b>para_jump_abs</b> command (see comments there).</li> <li>Further comments see <b>jump_abs_3d</b>.</li> </ul>
<b>RTC4→RTC6</b>	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified values for X, Y and Z coordinates by 16. The allowed value ranges decrease accordingly.</p> <p>There is no <b>RTC4 Compatibility Mode</b> for the parameter P. The original RTC6 units must be used. Exception is <b>Ctrl</b> = 7 (Defocus): In <b>RTC4 Compatibility Mode</b>, with <b>Ctrl</b> = 7, the RTC6 multiplies the specified values for <b>Value</b> by 16. The allowed value ranges decrease accordingly.</p>
<b>RTC5→RTC6</b>	<p>Unchanged functionality.</p> <p>Exception is <b>Ctrl</b> = 7 (Defocus): In <b>RTC5 Compatibility Mode</b>, with <b>Ctrl</b> = 7, the RTC6 multiplies the specified values for <b>Value</b> by 16. The allowed value ranges decrease accordingly.</p>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>jump_abs_3d, set_vector_control, para_jump_abs, para_jump_rel_3d</b>



<b>Normal List Command</b>	<b>para_jump_rel</b>
<b>Function</b>	Moves the output point (of the laser focus) along a 2D vector at jump speed from the current position to the specified position (relative coordinate values) within a 2D image field. Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Call</b>	<code>para_jump_rel( dx, dy, p )</code>
<b>Parameters</b>	<p><b>dx</b>      Relative x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>dy</b>      Like <b>dx</b> (analogously).</p> <p><b>p</b>      End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>para_jump_rel</b> is identical to <b>para_jump_abs</b> (see comments there).</li> <li><b>p</b> is not treated on a relative basis.</li> </ul>
<b>RTC4→RTC6</b>	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified coordinate values by 16. The allowed value range decreases accordingly. For the parameter <b>p</b> , see <b>para_jump_abs</b> .
<b>RTC5→RTC6</b>	Unchanged functionality.  For the parameter <b>p</b> , see <b>para_jump_abs</b> .
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>jump_rel</b> , <b>set_vector_control</b> , <b>para_jump_abs</b> , <b>para_jump_rel_3d</b>



<b>Normal List Command</b>	<b>para_jump_rel_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) along a 3D vector at jump speed from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> . Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>para_jump_rel_3d</b> has the same effect as <b>para_jump_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
<b>Call</b>	<b>para_jump_rel_3d( dx, dy, dz, P )</b>
<b>Parameters</b>	<p><b>dx</b> Relative x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>dy</b> Like <b>dx</b> (analogously).</p> <p><b>dz</b> Like <b>dx</b> (analogously), except Allowed value range: [-524,288...+524,287].</p> <p><b>P</b> End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>para_jump_rel_3d</b> is identical to <b>para_jump_abs_3d</b> (see comments there).</li> <li><b>P</b> is not treated on a relative basis.</li> </ul>
<b>RTC4→RTC6</b>	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified value for the x and y coordinates by 16. The allowed value range decreases accordingly. For the parameter <b>P</b> , see <b>para_jump_abs</b> .
<b>RTC5→RTC6</b>	Unchanged functionality.  For the parameter <b>P</b> , see <b>para_jump_abs</b> .
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>jump_rel_3d, set_vector_control, para_jump_abs_3d, para_jump_rel</b>

<b>Variable List Command</b>	<b>para_laser_on_pulses_list</b>
<b>Function</b>	Turns on the LASERON laser control signal for the specified number of external signal pulses (but for no longer than the specified time interval). Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Call</b>	<code>para_laser_on_pulses_list( Period, Pulses, P )</code>
<b>Parameters</b>	<p>Period      time interval. In bits. As an unsigned 32-bit value.  <i>1 bit equals 10 µs.</i>          Allowed value range: <math>0 \leq \text{Period} \leq (2^{32}-1)</math>.</p> <p>Pulses      Number of external signal pulses. As an unsigned 32-bit value.          Allowed value range: <math>0 \leq \text{Pulses} \leq 65,535</math> or larger (see comments below).</p> <p>P      End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>para_laser_on_pulses_list</b> is useful for marking separate points, see <a href="#">Chapter 7.1.3 "Marking Single Points", page 130</a>.</li> <li>• If “vector-controlled laser control” has not been previously activated by <b>set_vector_control</b>, then <b>para_laser_on_pulses_list</b> behaves like <b>laser_on_pulses_list</b> and – if <math>\text{Pulses} &gt; 65,535</math> – like <b>laser_on_list</b> (see comments there). The parameter <b>P</b> is then ignored.</li> <li>• If “vector-controlled laser control” is activated, then the signal parameter selected by <b>set_vector_control</b> is linearly varied from the last valid value to <b>P</b> within the command’s duration (<math>\text{Period} \times 10 \mu\text{s}</math>), see <a href="#">Section “Vector-Defined Laser Control”, page 195</a>.</li> <li>• There is no abs mechanism for <b>P</b>. <b>P</b> is, if necessary, clipped to the maximum allowed value. This maximum value is <math>(2^{31}-1)</math> or a lower value depending on what was selected with <b>set_vector_control</b> (<b>Ctrl</b> parameter).</li> <li>• If <b>para_laser_on_pulses_list</b> is used along with position- and/or speed-dependent laser control for the same control parameter, then the current value of <b>P</b> is used as the basis of the 100% value for laser control, see <a href="#">Section “Vector-Defined Laser Control”, page 195</a>.</li> <li>• If <math>\text{Period} = 0</math>, <b>para_laser_on_pulses_list</b> has no effect. Then <b>para_laser_on_pulses_list</b> is a short list command.</li> <li>• If <math>0 &lt; \text{Period} \leq (2^{31}-1)</math>, then the <b>para_laser_on_pulses_list</b> duration is always <b>Period</b> clocks (that is, <math>\text{Period} \times 10 \mu\text{s}</math>), even if the specified number of external signal pulses expires in a shorter time interval.</li> <li>• If <math>2^{31} \leq \text{Period} \leq (2^{32}-1)</math>, the <b>para_laser_on_pulses_list</b> maximum duration is <math>(\text{Period} - 2^{31})</math> clocks (that is, <math>(\text{Period} - 2^{31}) \times 10 \mu\text{s}</math>). Here, however, <b>para_laser_on_pulses_list</b> terminates as soon as the specified number of external signal pulses has been detected.</li> </ul>



<b>Variable List Command</b>	<b>para_laser_on_pulses_list</b>
RTC4→RTC6	New command. For the parameter $P$ , see <a href="#">para_jump_abs</a> .
RTC5→RTC6	Unchanged functionality. For the parameter $P$ , see <a href="#">para_jump_abs</a> .
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">laser_on_pulses_list</a> , <a href="#">laser_on_list</a>



<b>Normal List Command</b>	<b>para_mark_abs</b>
<b>Function</b>	Moves the laser focus at mark speed along a 2D vector from the current position to the specified position (absolute coordinate values) within a 2D image field. Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Call</b>	<b>para_mark_abs( X, Y, P )</b>
<b>Parameters</b>	<p>X      Absolute x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p> <p>P      End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If "vector-controlled laser control" has not been previously activated by <b>set_vector_control</b>, then <b>para_mark_abs</b> behaves like <b>mark_abs</b> (see comments there). The parameter P is then ignored.</li> <li>If "vector-controlled laser control" is activated, then simultaneously with the laser focus' motion the signal parameter selected by <b>set_vector_control</b> is linearly varied from the last valid value to P, see <b>Section "Vector-Defined Laser Control", page 195</b>.</li> <li>There is no abs mechanism for P. P is, if necessary, clipped to the maximum allowed value.</li> <li>If <b>para_mark_abs</b> is used along with position- and/or speed-dependent laser control for the same control parameter, then the current value of P is used as the basis of the 100% value for laser control (see <b>Section "Vector-Defined Laser Control", page 195</b>).</li> <li>[*]<b>para_mark</b>[*] commands generally do not take Sky Writing into account.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified coordinate values by 16. The allowed value range decreases accordingly. For the parameter P, see <b>para_jump_abs</b>.</p>
RTC5→RTC6	<p>Unchanged functionality.</p> <p>For the parameter P, see <b>para_jump_abs</b>.</p>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>mark_abs</b> , <b>set_vector_control</b> , <b>para_mark_abs_3d</b> , <b>para_mark_rel</b>



<b>Normal List Command</b>	<a href="#">para_mark_abs_3d</a>
<b>Function</b>	Moves the laser focus at mark speed along a 3D vector from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> . Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>para_mark_abs_3d</b> has the same effect as <b>para_mark_abs</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
<b>Call</b>	<code>para_mark_abs_3d( X, Y, Z, P )</code>
<b>Parameters</b>	<p>X      Absolute x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p> <p>Z      Like X (analogously), except Allowed value range: [-524,288...+524,287].</p> <p>P      End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, this command functions similarly to the <b>para_mark_abs</b> command (see comments there).</li> <li>Further comments see <b>mark_abs_3d</b>.</li> <li>[*]<b>para_mark</b>[*] commands generally do not take Sky Writing into account.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified value for the x and y coordinates by 16. The allowed value range decreases accordingly.</p> <p>For the parameter P, see <b>para_jump_abs</b>.</p>
RTC5→RTC6	<p>Unchanged functionality.</p> <p>For the parameter P, see <b>para_jump_abs</b>.</p>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">mark_abs_3d</a> , <a href="#">set_vector_control</a> , <a href="#">para_mark_abs</a> , <a href="#">para_mark_rel_3d</a>



<b>Normal List Command</b>	<b>para_mark_rel</b>
<b>Function</b>	Moves the laser focus at mark speed along a 2D vector from the current position to the specified position (relative coordinate values) within a 2D image field. Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Call</b>	<code>para_mark_rel( dX, dY, P )</code>
<b>Parameters</b>	<p><b>dX</b>      Relative x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>dY</b>      Like <b>dX</b> (analogously).</p> <p><b>P</b>      End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>para_mark_rel</b> is identical to <b>para_mark_abs</b> (see comments there).</li> <li><b>P</b> is not treated on a relative basis.</li> <li>[*]<b>para_mark</b>[*] commands generally do not take Sky Writing into account.</li> </ul>
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified coordinate values by 16. The allowed value range decreases accordingly. For the parameter <b>P</b> , see <b>para_jump_abs</b> .
RTC5→RTC6	Unchanged functionality.  For the parameter <b>P</b> , see <b>para_jump_abs</b> .
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>mark_rel</b> , <b>set_vector_control</b> , <b>para_mark_abs</b> , <b>para_mark_rel_3d</b>



<b>Normal List Command</b>	<b>para_mark_rel_3d</b>
<b>Function</b>	Moves the laser focus at mark speed along a 3D vector from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> . Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>para_mark_rel_3d</b> has the same effect as <b>para_mark_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
<b>Call</b>	<b>para_mark_rel_3d( dx, dy, dz, P )</b>
<b>Parameters</b>	<p><b>dx</b>      Relative x coordinate of the mark vector end point. In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual image field for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>dy</b>      Like <b>x</b> (analogously).</p> <p><b>dz</b>      Like <b>x</b> (analogously), except Allowed value range: [-524,288...+524,287].</p> <p><b>P</b>      End value of the signal parameter. As an unsigned 32-bit value. Allowed values: dependent on the selection made by <b>set_vector_control</b> (<b>Ctrl</b> parameter), identical with <b>set_vector_control</b> (<b>Value</b> parameter).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>para_mark_rel_3d</b> is identical to <b>para_mark_abs_3d</b> (see comments there).</li> <li><b>P</b> is not treated on a relative basis.</li> <li>[*]<b>para_mark</b>[*] commands generally do not take Sky Writing into account.</li> </ul>
<b>RTC4→RTC6</b>	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified value for the x and y coordinates by 16. The allowed value range decreases accordingly. For the parameter <b>P</b> , see <b>para_jump_abs</b> .
<b>RTC5→RTC6</b>	Unchanged functionality.  For the parameter <b>P</b> , see <b>para_jump_abs</b> .
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>mark_rel_3d</b> , <b>set_vector_control</b> , <b>para_mark_abs_3d</b> , <b>para_mark_rel</b>

<b>Variable List Command</b>	<b>park_position</b>
<b>Function</b>	For temporary parking, this command moves the output point (of the laser focus) along a 2D vector at jump speed from the current position to the specified position (absolute coordinate values) within the 2D image field.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>park_position</b> functions like a normal <b>jump_abs</b> .
<b>Call</b>	<code>park_position( Mode, X, Y )</code>
<b>Parameters</b>	<p>Mode      As an unsigned 32-bit value.</p> <p>= 0:    X and Y are interpreted as park position coordinates in the real image field. Allowed value range: [-524,288...+524,287].  The current Processing-on-the-fly mode is switched off and the laser focus moved at the currently set jump speed to the specified park position in the real image field. During a subsequent list interruption (by <b>wait_for_encoder</b> etc.), the galvanometer scanners remains stationary (even for a Processing-on-the-fly application initiated by <b>set_fly_2d</b>).</p> <p>&gt; 0:    X and Y are interpreted as park position coordinates in the virtual image field. Allowed value range: [-268,435,456...+268,435,455].  The current Processing-on-the-fly mode remains switched on and the laser focus moved at the currently set jump speed to the specified park position in the virtual image field (subject to Processing-on-the-fly correction and clipped to the real image field's boundaries). During a subsequent list interruption (by <b>wait_for_encoder</b> etc.), the galvanometer scanners' positions are continuously Processing-on-the-fly-corrected in accordance with the current encoder values (even for a Processing-on-the-fly application initiated by <b>set_fly_x/set_fly_y</b>) and clipped to the real image field's boundaries.</p> <p>X      Absolute coordinates of the jump vector end point. In bits.  Y      As signed 32-bit values.  Allowed value range: see above. Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>park_position</b> is intended for encoder-based <b>set_fly_2d</b> or <b>set_fly_x/set_fly_y</b> Processing-on-the-fly applications where the laser focus needs to be moved to a safe parking area during an forward motion (prior to list interruption by <b>wait_for_encoder</b> etc.), see <b>Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications"</b>, page 237. For the return jump (away from the park position), use the <b>park_return</b> command (refer to all comments there).</li> <li>• If another (or no) Processing-on-the-fly application is active, then <b>park_position</b> functions like a normal jump command, as does the return jump by <b>park_return</b> (but Processing-on-the-fly remains switched off).</li> </ul>



Variable List Command	<b>park_position</b>
Comments (cont'd)	<ul style="list-style-type: none"><li>If the laser focus was already previously moved to a safe park position, then <b>park_position</b> is a short list command with no effect.</li><li><b>park_position</b> switches off the signals for "laser active" operation after a LaserOff delay, but does not activate a scanner delay afterward.</li></ul>
RTC4→RTC6	New command. In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified coordinate values by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">park_return</a>



<b>Variable List Command</b>	<b>park_position_1_axis</b>
<b>Function</b>	"Fly Extension" Command: Moves the output position for 1 <b>Axis</b> for intermediate parking with jump speed from the current to the specified absolute position in the 2D image field.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>park_position_1_axis</b> only carries-out the jump to the specified new output position.
<b>Call</b>	<code>park_position_1_axis( Mode, Axis, ParkPos )</code>
<b>Parameters</b>	<p>Mode           = 0: Intermediate parking position. Absolute coordinate in the real image field. Processing-on-the-fly is switched off at the specified <b>Axis</b>.            &gt; 0: Intermediate parking position. Absolute coordinate in the virtual image field. Processing-on-the-fly remains switched on at the specified <b>Axis</b>.            As an unsigned 32-bit value.</p> <p>Axis           <b>Axis</b> from <b>Table 3, page 245</b>.            As an unsigned 32-bit value.            Allowed values: 1...2.</p> <p>ParkPos       Absolute axis coordinate of the new output position. In bits.            As a signed 32-bit value.            Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <b>Chapter 8.6 "Processing-on-the-fly"</b>, page 227 and <b>Section ""Fly Extension" Commands"</b>, page 245.</li> <li>See also comments on <b>park_position</b>.</li> <li>To jump back, <b>park_return_1_axis</b> can be used.</li> <li>With an unallowed parameter value, <b>park_position_1_axis</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
<b>Version info</b>	Available as of version DLL 617, OUT 617, RBF 623.
<b>References</b>	<b>park_position_2_axes</b>



<b>Variable List Command</b>	<b>park_position_2_axes</b>
<b>Function</b>	"Fly Extension" Command: Moves the output positions for 2 Axes for intermediate parking with jump speed from the current to the specified absolute position in the 2D image field.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>park_position_2_axes</b> only carries-out the jump to the specified new output position.
<b>Call</b>	<code>park_position_2_axes( Mode, ParkPosX, ParkPosY )</code>
<b>Parameters</b>	<p>Mode = 0: Intermediate parking position. Absolute coordinate in the real image field. Processing-on-the-fly is switched off at the specified Axes.            &gt; 0: Intermediate parking position. Absolute coordinate in the virtual image field. Processing-on-the-fly remains switched on at the specified Axes.            As an unsigned 32-bit value.</p> <p>ParkPosX Absolute axis coordinate of the new output position. In bits.            As a signed 32-bit value.            Out-of-range values are clipped to the boundary values.</p> <p>ParkPosY Like ParkPosX.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See Chapter 8.6 "Processing-on-the-fly", page 227 and Section ""Fly Extension" Commands", page 245.</li> <li>See also comments on <b>park_position</b>.</li> <li>To jump back, <b>park_return_2_axes</b> can be used.</li> <li>With an unallowed parameter value, <b>park_position_2_axes</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>The following command calls are executed in the same way:           <pre>- park_position_2_axes( Mode, ParkPosX, ParkPosY ) =           {             park_position_1_axis( Mode, 1, ParkPosX );             park_position_1_axis( Mode, 2, ParkPosY );           }</pre> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
<b>Version info</b>	Available as of version DLL 617, OUT 617, RBF 623.
<b>References</b>	<b>park_position_1_axis</b>



<b>Variable List Command</b>	<b>park_return</b>
<b>Function</b>	Moves the output point (of the laser focus) away from a park position along a 2D vector at jump speed to the specified position (absolute coordinate values) within the 2D image field.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>park_return</b> functions as a normal jump command. If the laser focus is not currently in a park position, then <b>park_return</b> is a short list command with no effect (see below).
<b>Call</b>	<code>park_return( Mode, X, Y )</code>
<b>Parameters</b>	<p>Mode      As an unsigned 32-bit value.                    = 0:    X and Y are ignored (see comments).                    &gt; 0:    X and Y are interpreted as return jump coordinates (see comments).</p> <p>X      Absolute coordinates of the jump vector end point in the virtual image field.                In bits.</p> <p>Y      As signed 32-bit values.                Allowed value range: [-268,435,456...+268,435,455].                Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>park_return</b> is intended for encoder-based <b>set_fly_2d</b> or <b>set_fly_x/set_fly_y</b> Processing-on-the-fly applications where the laser focus should leave a safe parking area previously reached by <b>park_position</b> after an intermediate forwarding motion (following list interruption by <b>wait_for_encoder</b> etc.), see <a href="#">Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237</a>. See also comments at <b>park_position</b>.</li> <li>• If a <b>set_fly_2d</b> or <b>set_fly_x/set_fly_y</b> Processing-on-the-fly mode was switched off by a prior <b>park_position</b>( Mode = 0 ) call, then <b>park_return</b> switches the same Processing-on-the-fly mode back on. Other Processing-on-the-fly modes are not switched back on.</li> <li>• If the park position was attained by <b>park_position</b>, then <b>park_return</b>( Mode = 0 ) returns the galvanometer scanners to the most recent valid position before <b>park_position</b> was called, whereas <b>park_return</b>( Mode = 1 ) moves them to the position specified with the command. When calculating the new position, the RTC6 takes the current Processing-on-the-fly correction into account. But if clipping to the boundaries of the virtual image field occurs (for example, due to a too long forwarding motion during a list interruption by <b>wait_for_encoder</b>), then the Processing-on-the-fly mode is not reactivated (see also <b>activate_fly_2d</b> and <b>activate_fly_xy</b>) and the jump executes without Processing-on-the-fly correction.</li> </ul>



Variable List Command	park_return
Comments (cont'd)	<ul style="list-style-type: none"> <li>If the laser focus was <i>not</i> previously moved to a safe area by <b>park_position</b>, then <b>park_return</b> is a short list command with no further effect.</li> <li>If no Processing-on-the-fly mode was previously active or if any Processing-on-the-fly mode is currently active, then <b>park_return</b> performs a normal jump to the specified location.</li> <li>If a coordinate transformation in the virtual image field is active, then the specified or most recent valid position is subsequently appropriately transformed (see <a href="#">Section "Coordinate Transformations in the Virtual Image Field", page 158</a>).</li> <li>A scanner jump delay is activated after the <b>park_return</b> command.</li> <li><b>park_return</b> switches off the "laser active" laser control signals after a LaserOff delay.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC6 multiplies the specified coordinate values by 16. The allowed value range decreases accordingly.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">park_position</a>



<b>Variable List Command</b>	<b>park_return_1_axis</b>
<b>Function</b>	"Fly Extension" Command: Moves the output position for 1 Axis with jump speed from the intermediate parking position to the specified absolute position in the 2D image field.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then park_return_1_axis only carries-out the jump to the specified new output position.
<b>Call</b>	park_return_1_axis( Mode, Axis, RetPos )
<b>Parameters</b>	<p>Mode = 0: Return position is ignored.          &gt; 0: Return position. Absolute coordinate in the virtual image field.          As an unsigned 32-bit value.</p> <p>Axis <b>Axis from Table 3, page 245.</b>          As an unsigned 32-bit value.          Allowed values: 1...2.</p> <p>RetPos Absolute axis coordinate of the new output position. In bits.          As a signed 32-bit value.          Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See Chapter 8.6 "Processing-on-the-fly", page 227 and Section ""Fly Extension" Commands", page 245.</li> <li>See also comments on <b>park_return</b>.</li> <li><b>park_return_1_axis</b> can be used to jump back to <b>park_position_1_axis</b>.</li> <li>If a Processing-on-the-fly correction has been previously deactivated at the specified axis by <b>park_position_1_axis( Mode = 0 )</b> or <b>park_position_2_axes( Mode = 0 )</b> then <b>park_return_1_axis</b> switches it back on again.</li> <li>With an unallowed parameter value, <b>park_return_1_axis</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<b>park_return_2_axes</b>



<b>Variable List Command</b>	<b>park_return_2_axes</b>
<b>Function</b>	"Fly Extension" Command: Moves the output position for 2 Axes with jump speed from the intermediate parking position to the specified absolute position in the 2D image field.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>park_return_2_axes</b> only carries-out the jump to the specified new output positions.
<b>Call</b>	<code>park_return_2_axes( Mode, RetPosX, RetPosY )</code>
<b>Parameters</b>	<p>Mode = 0: Return positions are ignored.          &gt; 0: Return positions. Absolute coordinates in the virtual image field.          As an unsigned 32-bit value.</p> <p>RetPosX Absolute axes coordinates of the new output positions. In bits.          As a signed 32-bit value.          Out-of-range values are clipped to the boundary values.</p> <p>RetPosY Like RetPosX.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See Chapter 8.6 "Processing-on-the-fly", page 227 and Section ""Fly Extension" Commands", page 245.</li> <li>See also comments on <b>park_return</b>.</li> <li><b>park_return_2_axes</b> can be used to jump back to <b>park_position_2_axes</b>.</li> <li>If a Processing-on-the-fly correction has been previously deactivated at the specified axis by <code>park_position_2_axes( Mode = 0 )</code> then <b>park_position_2_axes</b> switches it back on again.</li> <li>With an unallowed parameter value, <b>park_return_2_axes</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>The following command calls are executed in the same way:             <pre>- park_return_2_axes( Mode, RetPosX, RetPosY ) = {     park_return_1_axis( Mode, 1, RetPosX );     park_return_1_axis( Mode, 2, RetPosY ); }</pre> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<b>park_return_1_axis</b>



<b>Ctrl Command</b>	<b>pause_list</b>
<b>Function</b>	Pauses execution of the list and disables the "laser active" laser control signals.
<b>Call</b>	<code>pause_list()</code>
<b>Parameters</b>	-
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>pause_list</b> is synonymous with <b>stop_list</b>. <b>stop_list</b> is often confused with <b>stop_execution</b>. Therefore, it is preferable to use <b>pause_list</b>.</li> <li>• If <b>pause_list</b> is called during execution of a list, then the signals for "laser active" operation are suppressed (the signals are set to their respective "Off" level) and keeps the scan system in the most recently defined state – even if in the middle of a split-up into microsteps. The <b>PAUSED</b> status (queryable by <b>get_status</b>) is set, but the <b>BUSY</b> status is left unchanged. Continuation by <b>execute_list_pos</b> or <b>release_wait</b> or by an external start is not possible. However, <b>stop_execution</b> or an external stop is possible. <b>restart_list</b>, <b>stop_execution</b> or an external stop ends suppression of the start.</li> <li>• If processing of a list should be continued, then <b>restart_list must</b> be used. After a subsequent <b>restart_list</b>, the scan system resumes the planned movements (of the current command) and the laser control signals are released again (in general, an interrupted marking cannot be continued without a disruption in the marking result). The <b>PAUSED</b> status is then reset (here too, <b>BUSY</b> remains unchanged).</li> <li>• If, during calling of <b>pause_list</b>, no list is currently executing (<b>BUSY</b> status not set) or a list has already been halted by <b>pause_list</b> or <b>set_wait</b> (<b>PAUSED</b> status set), then <b>pause_list</b> is ignored (<b>get_last_error</b> return code <b>RTC6_BUSY</b>; the laser is then already off).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>restart_list</b> , <b>set_wait</b>



<b>Ctrl Command</b>	<b>periodic_toggle</b>												
<b>Function</b>	Like <a href="#">periodic_toggle_list</a> , but a control command.												
<b>Call</b>	<code>periodic_toggle( Port, Mask, P1, P2, Count, Start )</code>												
<b>Parameters</b>	<table> <tr> <td>Port</td> <td>Like <a href="#">periodic_toggle_list</a>.</td> </tr> <tr> <td>Mask</td> <td>Like <a href="#">periodic_toggle_list</a>.</td> </tr> <tr> <td>P1</td> <td>Like <a href="#">periodic_toggle_list</a>.</td> </tr> <tr> <td>P2</td> <td>Like <a href="#">periodic_toggle_list</a>.</td> </tr> <tr> <td>Count</td> <td>Like <a href="#">periodic_toggle_list</a>.</td> </tr> <tr> <td>Start</td> <td>Like <a href="#">periodic_toggle_list</a>.</td> </tr> </table>	Port	Like <a href="#">periodic_toggle_list</a> .	Mask	Like <a href="#">periodic_toggle_list</a> .	P1	Like <a href="#">periodic_toggle_list</a> .	P2	Like <a href="#">periodic_toggle_list</a> .	Count	Like <a href="#">periodic_toggle_list</a> .	Start	Like <a href="#">periodic_toggle_list</a> .
Port	Like <a href="#">periodic_toggle_list</a> .												
Mask	Like <a href="#">periodic_toggle_list</a> .												
P1	Like <a href="#">periodic_toggle_list</a> .												
P2	Like <a href="#">periodic_toggle_list</a> .												
Count	Like <a href="#">periodic_toggle_list</a> .												
Start	Like <a href="#">periodic_toggle_list</a> .												
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">periodic_toggle_list</a>.</li> <li>• If an unallowed parameter value is supplied for <code>Port</code> or 0 for <code>P1</code> or <code>P2</code>, then <code>periodic_toggle</code> is not sent and a <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code> is generated.</li> <li>• See also <a href="#">Chapter 9.4 "Periodical I/O Signals", page 287</a>.</li> </ul>												
RTC4→RTC6	New command.												
RTC5→RTC6	Unchanged functionality.												
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 605, OUT 605: endless toggling with <code>Count</code> = $2^{32}-1$ .												
References	<a href="#">periodic_toggle_list</a>												



<b>Undelayed Short List Command</b>	<b>periodic_toggle_list</b>												
<b>Function</b>	Generates and periodically toggles a signal at an adjustable output port.												
<b>Call</b>	<code>periodic_toggle_list( Port, Mask, P1, P2, Count, Start )</code>												
<b>Parameters</b>	<table> <tr> <td>Port</td> <td>Output port (0–4, as with <a href="#">set_port_default</a>). As an unsigned 32-bit value.</td> </tr> <tr> <td>Mask</td> <td>Defines the bits to be toggled, (the maximum value depends on the port). 1 = bit is toggled. 0 = bit remains unchanged. As an unsigned 32-bit value.</td> </tr> <tr> <td>P1</td> <td>Duration of the toggled signal in [10 µs] (&gt;0, 16 bit max.). As an unsigned 32-bit value.</td> </tr> <tr> <td>P2</td> <td>Duration of the toggled signal in [10 µs] (&gt;0, 16 bit max.). As an unsigned 32-bit value.</td> </tr> <tr> <td>Count</td> <td>Number of P1–P2 period repetitions. As an unsigned 32-bit value.</td> </tr> <tr> <td>Start</td> <td>Duration until the first toggling starts in [10 µs]. As an unsigned 32-bit value.</td> </tr> </table>	Port	Output port (0–4, as with <a href="#">set_port_default</a> ). As an unsigned 32-bit value.	Mask	Defines the bits to be toggled, (the maximum value depends on the port). 1 = bit is toggled. 0 = bit remains unchanged. As an unsigned 32-bit value.	P1	Duration of the toggled signal in [10 µs] (>0, 16 bit max.). As an unsigned 32-bit value.	P2	Duration of the toggled signal in [10 µs] (>0, 16 bit max.). As an unsigned 32-bit value.	Count	Number of P1–P2 period repetitions. As an unsigned 32-bit value.	Start	Duration until the first toggling starts in [10 µs]. As an unsigned 32-bit value.
Port	Output port (0–4, as with <a href="#">set_port_default</a> ). As an unsigned 32-bit value.												
Mask	Defines the bits to be toggled, (the maximum value depends on the port). 1 = bit is toggled. 0 = bit remains unchanged. As an unsigned 32-bit value.												
P1	Duration of the toggled signal in [10 µs] (>0, 16 bit max.). As an unsigned 32-bit value.												
P2	Duration of the toggled signal in [10 µs] (>0, 16 bit max.). As an unsigned 32-bit value.												
Count	Number of P1–P2 period repetitions. As an unsigned 32-bit value.												
Start	Duration until the first toggling starts in [10 µs]. As an unsigned 32-bit value.												
<b>Comments</b>	<ul style="list-style-type: none"> <li>If an unallowed parameter value is supplied for <code>Port</code> or 0 for <code>P1</code> or <code>P2</code>, then <code>periodic_toggle_list</code> is replaced by a <code>list_nop</code> and a <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code> is generated.</li> <li>Mask defines the bits to be toggled. Bits which are set to 1 in <code>Mask</code> are toggled and bits which are set to 0 remain unchanged. <code>Mask</code> is limited to the maximum allowed value of the selected port (see also <a href="#">set_port_default</a>). Extra bits are ignored.</li> <li>The selected bits are toggled. This state is maintained for <math>P1 \times 10 \mu s</math> clock periods. Then they are toggled once again and kept stable for <math>P2 \times 10 \mu s</math> clock periods. Users define the toggle bit start values ("off" state; "on" signal is active-HIGH or active-LOW) by other standard commands (<a href="#">write_da_x</a>, <a href="#">write_8bit_port</a>, <a href="#">write_io_port</a>, etc.). These – and (should the situation arise) other queued delayed short list commands – are executed before the command <code>periodic_toggle_list</code>.</li> <li>The first toggling occurs after <code>Start</code> <math>\times 10 \mu s</math> clock periods. Toggling is repeated <code>Count</code>-times. <code>Count</code> = 0 immediately stops an output in progress. The remaining parameters are then not relevant. If the stop happens in the <code>P1</code> period ("On"), a toggle occurs to restore the initial state ("Off").</li> <li>The parameters <code>P1</code> and <code>P2</code> are clipped to the value range [0...65,535]. <code>P1</code> and <code>P2</code> must not be 0 at the same time.</li> <li>The selected port should not concurrently be used for other outputs such as "Automatic Laser Control".</li> <li>At a <code>set_end_of_list</code>, <code>stop_execution</code> or external /STOP the periodical signals continue.</li> <li><code>periodic_toggle_list</code> toggles endless with <code>Count</code> = <math>2^{32}-1</math>.</li> <li>See also <a href="#">Chapter 9.4 "Periodical I/O Signals", page 287</a>.</li> </ul>												



<b>Undelayed Short List Command</b>	<b>periodic_toggle_list</b>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 605, OUT 605: endless toggling with Count = $2^{32}-1$ .
References	<a href="#">periodic_toggle</a> , <a href="#">set_port_default</a>

<b>Ctrl Command</b>	<b>quit_loop</b>
Function	Stops the repeating automatic list change started with <a href="#">start_loop</a> .
Call	<code>quit_loop()</code>
Comments	<ul style="list-style-type: none"> <li>Before list execution is stopped, the current list is fully executed until the next <a href="#">set_end_of_list</a> is encountered.</li> <li><a href="#">start_loop</a> must be called prior to <a href="#">quit_loop</a>. Otherwise, <a href="#">quit_loop</a> has no effect.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">start_loop</a>



<b>Undelayed Short List Command</b>	<b>range_checking</b>	
<b>Function</b>	Defines an emergency action, if a galvanometer exceeds its position limit.	
<b>Call</b>	range_checking( HeadNo, Mode, Data )	
<b>Parameters</b>	HeadNo	<p>The scan system to be monitored. As an unsigned 32-bit value.</p> <p>0: No monitoring (default after <a href="#">load_program_file</a>). 1: First scan head is monitored. 2: Second scan head is monitored. 3: Both scan heads are monitored.</p> <p>Only the 2 least significant bits are evaluated.</p>
	Mode	<p>Action to take. As an unsigned 32-bit value.</p> <p>0: The signals for “laser active” operation are suppressed immediately and list execution continues. 1: The signals for “laser active” operation are switched-off immediately and list execution is stopped immediately (as with <a href="#">stop_execution</a> or <a href="#">/STOP</a>). 2: A <a href="#">simulate_ext_stop</a> is forwarded to all slave boards.</p>
	Data	<p>Data type chosen to be monitored. As an unsigned 32-bit value.</p> <p>0: Sample values (Processing-on-the-fly corrected and Wobbel corrected, as with <a href="#">set_trigger</a> signal types 7...9). 1: Transformed control values (with scan head-specific transformations, as with <a href="#">set_trigger</a> signal types 25–30). 2: Corrected control values (with correction file, as with <a href="#">set_trigger</a> signal types 10...15). 3: Actual output values (with “Offset” and “Gain”, as with <a href="#">set_trigger</a> signal types 20...23). 4: Real position of galvanometer (only with iDRIVE systems, varioSCAN is internally connected to an x axis, as with <a href="#">set_trigger</a> signal types 1, 2 and 3). 5: Real position of galvanometer (only with iDRIVE systems, varioSCAN is internally connected to an y axis, as with <a href="#">set_trigger</a> signal types 1, 2 and 4).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>No monitoring takes place if the specified scan head does not have a correction table assigned (see <a href="#">select_cor_table</a>).</li> <li>The used position limits (that is, if exceeded the emergency action is executed) are the parameters of a customer-specific Processing-on-the-fly application monitoring (see <a href="#">set_fly_limits</a>, <a href="#">set_fly_limits_z</a>). Exceeding these limits only cause error messages in case of Processing-on-the-fly applications. The error messages can be queried with <a href="#">get_marking_info</a> or the respective <a href="#">if_fly_x_overflow</a>/<a href="#">if_fly_y_overflow</a>/<a href="#">if_fly_z_overflow</a> and <a href="#">if_not_fly_x_overflow</a>/<a href="#">if_not_fly_y_overflow</a>/<a href="#">if_not_fly_z_overflow</a>. They do not trigger any activities.</li> </ul>	



<b>Undelayed Short List Command</b>	<b>range_checking</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Processing-on-the-fly applications use the data type Data = 0 (sample values) for customer-specific range limits. Inconsistent error messages and switch-offs may occur if used simultaneously with <b>range_checking</b> data types Data &gt; 0.</li> <li>If the laser has been switched-off with Mode = 0 and the fault condition is gone then the laser is <i>not</i> switched on until the next <b>Mark command</b>. The laser is <i>not</i> switched-on right in the middle of a currently executing <b>Mark command</b>, not even in the middle of a <b>Polyline</b>.</li> <li>The <b>range_checking</b> monitoring is active without Processing-on-the-fly application.</li> <li>Data &gt; 5 causes a <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>. In this case the command is sent to the RTC6 board as <b>list_nop</b>.</li> <li>Data=2 and Data=3 have the identical effect for the z axis.</li> <li>For Data = 4 and Data = 5 users must define the set position as the to be returned data type. A simultaneous use of a speed-dependent laser control with Mode = 2 (see <b>set_auto_laser_control</b>) is not possible.</li> <li>Data = 4 and Data = 5 only differ in regards to the axis onto an optional varioSCAN is connected. For 2D systems without varioSCAN both selections have the identical effect.</li> <li>Data = 4 and Data = 5 may produce nonsensical switch-offs if an intelliSCAN is connected but is either not switched-on or a real position has not been set as feedback.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 615: Mode = 2.
References	<b>set_fly_limits, set_fly_limits_z</b>



<b>Ctrl Command</b>	<b>read_abc_from_file</b>	
<b>Function</b>	Reads the ABC values directly from a specified correction file on the PC.	
<b>Call</b>	ErrorNo = read_abc_from_file( Name, &A, &B, &C )	
<b>Result</b>	ErrorNo	Error code. As an unsigned 32-bit value.
	0	No error.
	1	File error (corrupt or incomplete).
	2	Memory error ( <a href="#">RTC6 DLL</a> -internal, Windows working memory).
	3	File-open error (empty string, file not found etc.).
<b>Parameters</b>	Name	Name of the correction file. As a pointer to a \0-terminated ANSI string.
<b>Returned parameter values</b>	A B C	Coefficients of the parabolic function $z_{out} = A + BI + CI^2$ which is used for calculating the Z output values (I in the RTC4 compatibility range [-32,768...+32,767]). As 64-bit IEEE floating point values.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>read_abc_from_file</b> is available even without explicit access rights to a specific RTC6 board.</li> <li>• <b>read_abc_from_file</b> is not available as a multi-board command.</li> <li>• The board-specific error variables <code>LastError</code> and <code>AccError</code>, see <a href="#">Chapter 6.8 "Error Handling", page 120</a>, are neither generated nor altered by <b>read_abc_from_file</b>.</li> </ul>	
RTC4→RTC6	New command.	
RTC5→RTC6	Unchanged functionality.	
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.	
<b>References</b>	<a href="#">write_abc_to_file</a>	



<b>Ctrl Command</b>	<b>read_analog_in</b>																												
<b>Function</b>	Reads the analog input values (ANALOG IN0 and ANALOG IN1) at the "McBSP/ANALOG" socket connector of the RTC6 PCIe Board and returns them as 12-bit digital values.																												
<b>Call</b>	AnalogValue = read_analog_in()																												
<b>Result</b>	<p>As an unsigned 32-bit value.</p> <table> <tr> <td>Bit #0</td> <td>ANALOG IN0 input value as 12-bit digital value.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #11</td> <td></td> </tr> <tr> <td>Bit #12</td> <td>= 0. Channel number.</td> </tr> <tr> <td>Bit #13</td> <td>= 0.</td> </tr> <tr> <td>Bit #14</td> <td>= 0.</td> </tr> <tr> <td>Bit #15</td> <td>Old bit for ANALOG IN0.</td> </tr> <tr> <td>Bit #16</td> <td>ANALOG IN1 input value as 12-bit digital value.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #27</td> <td></td> </tr> <tr> <td>Bit #28</td> <td>= 1. Channel number.</td> </tr> <tr> <td>Bit #29</td> <td>= 0.</td> </tr> <tr> <td>Bit #30</td> <td>= 0.</td> </tr> <tr> <td>Bit #31</td> <td>Old bit for ANALOG IN1.</td> </tr> </table>	Bit #0	ANALOG IN0 input value as 12-bit digital value.	...		Bit #11		Bit #12	= 0. Channel number.	Bit #13	= 0.	Bit #14	= 0.	Bit #15	Old bit for ANALOG IN0.	Bit #16	ANALOG IN1 input value as 12-bit digital value.	...		Bit #27		Bit #28	= 1. Channel number.	Bit #29	= 0.	Bit #30	= 0.	Bit #31	Old bit for ANALOG IN1.
Bit #0	ANALOG IN0 input value as 12-bit digital value.																												
...																													
Bit #11																													
Bit #12	= 0. Channel number.																												
Bit #13	= 0.																												
Bit #14	= 0.																												
Bit #15	Old bit for ANALOG IN0.																												
Bit #16	ANALOG IN1 input value as 12-bit digital value.																												
...																													
Bit #27																													
Bit #28	= 1. Channel number.																												
Bit #29	= 0.																												
Bit #30	= 0.																												
Bit #31	Old bit for ANALOG IN1.																												
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>read_analog_in</b> sets the old bits (<b>Bit #15</b> and <b>Bit #31</b>) to 1 after reading. As soon new data are available at the corresponding analog input they are automatically set to 0.</li> <li>• The analog input values (ANALOG IN0 and ANALOG IN1) can be recorded by <b>set_trigger/set_trigger4</b> (signal 54). However, the old bits (<b>Bit #15</b> and <b>Bit #31</b>) are not recorded. The format of the data is ((ANALOG IN1 &lt;&lt; 16) + ANALOG IN0).</li> </ul>																												
RTC4→RTC6	New command.																												
RTC5→RTC6	Unchanged functionality.																												
Version info	Available as of version DLL 600, OUT 600, RBF 600.																												
References	<b>set_trigger, set_trigger4</b>																												



<b>Ctrl Command</b>	<b>read_encoder</b>
<b>Function</b>	Returns the counts of the two RTC6 encoder counters that were stored by <a href="#">store_encoder</a> .
<b>Call</b>	<code>read_encoder( &amp;Encoder0_0, &amp;Encoder1_0, &amp;Encoder0_1, &amp;Encoder1_1 )</code>
<b>Returned parameter values</b>	<p>Encoder0_0      Counts.      Encoder1_0      As pointers to signed 32-bit values.      Encoder0_1      For parameter "Encodern_m":      Encoder1_1            <ul style="list-style-type: none"> <li>• <i>n</i> is the number of the encoder counter ("Encoder0", "Encoder1").</li> <li>• <i>m</i> is the number of the buffer position (parameter <code>Pos</code> of <a href="#">store_encoder</a>).</li> </ul> </p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 284</a>.</li> <li>• If incremental encoders are used to detect the motion of the parts to be processed, encoder counter "Encoder0" is triggered by the signals at encoder input ENCODER X and encoder counter "Encoder1" by the signals at encoder input ENCODER Y. In contrast, if an encoder simulation has been started by <a href="#">simulate_encoder( 3 )</a>, both encoder counters are triggered by an internal periodic 1 MHz clock signal.</li> <li>• For buffer positions in which counts were not previously stored by <a href="#">store_encoder</a>, the value 0 is returned (initialized value).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">store_encoder</a> , <a href="#">get_encoder</a> , <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">set_fly_rot</a> , <a href="#">wait_for_encoder</a>



<b>Ctrl Command</b>	<b>read_image_eth</b>																							
<b>Function</b>	<b>Standalone Functionality:</b> Reads out data for automatic booting from the <b>NAND memory</b> and saves it as a binary file on the PC.																							
<b>Prerequisite</b>	Minimum requirement: RTC6 Software Package V1.7.0 and <b>RTC6BIOSETH_26</b> .																							
<b>Call</b>	Result = <b>read_image_eth( Name )</b>																							
<b>Parameters</b>	<p>Name            Name of the binary file.  As a pointer to a \0-terminated ANSI string.</p>																							
<b>Result</b>	Result	<p>Error code.  As an unsigned 32-bit value.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error.</td> </tr> <tr> <td>1</td> <td>No RTC6 Ethernet Board.</td> </tr> <tr> <td>2</td> <td>Empty string has been specified as file name.</td> </tr> <tr> <td>3</td> <td>Access denied.</td> </tr> <tr> <td>4</td> <td>RTC6 Ethernet Board is <b>BUSY</b>.</td> </tr> <tr> <td>5</td> <td>TimeOut error (RTC6 Ethernet Board does not respond).</td> </tr> <tr> <td>6</td> <td>File cannot be opened.</td> </tr> <tr> <td>7</td> <td>Ethernet error, see <b>eth_get_last_error</b>.</td> </tr> <tr> <td>8</td> <td><b>NAND memory</b> error.</td> </tr> <tr> <td>9</td> <td>Aborted, see <b>eth_get_last_error</b>.</td> </tr> </tbody> </table>	Value	Description	0	No error.	1	No RTC6 Ethernet Board.	2	Empty string has been specified as file name.	3	Access denied.	4	RTC6 Ethernet Board is <b>BUSY</b> .	5	TimeOut error (RTC6 Ethernet Board does not respond).	6	File cannot be opened.	7	Ethernet error, see <b>eth_get_last_error</b> .	8	<b>NAND memory</b> error.	9	Aborted, see <b>eth_get_last_error</b> .
Value	Description																							
0	No error.																							
1	No RTC6 Ethernet Board.																							
2	Empty string has been specified as file name.																							
3	Access denied.																							
4	RTC6 Ethernet Board is <b>BUSY</b> .																							
5	TimeOut error (RTC6 Ethernet Board does not respond).																							
6	File cannot be opened.																							
7	Ethernet error, see <b>eth_get_last_error</b> .																							
8	<b>NAND memory</b> error.																							
9	Aborted, see <b>eth_get_last_error</b> .																							
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>read_image_eth</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>the <b>INTERNAL-BUSY</b> status of the board status is set</li> </ul> </li> <li>If the <b>Name</b> cannot be opened, a <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b> is generated.</li> <li>The content of an already existing binary file is deleted.</li> <li>During the execution of <b>read_image_eth</b> the 10 µs clock period of the <b>DSP</b> is interrupted for up to 2 minutes.</li> <li><b>read_image_eth</b> is only allowed with RTC6 Ethernet Boards.  Otherwise, a <b>get_last_error</b> return code <b>RTC6_TYPE_REJECTED</b> is generated.</li> <li>See <b>Chapter 15.7 "Standalone Functionality", page 859</b>.</li> </ul>																							
RTC4→RTC6	New command.																							
RTC5→RTC6	New command.																							
Version info	Available as of version DLL 618, OUT 618, RBF 623.																							
References	<b>write_image_eth, store_program</b>																							



Ctrl Command	read_io_port
Function	Returns the current state of the 16-bit digital input port on the EXTENSION 1 socket connector.
Call	IOPort = read_io_port()
Result	16-bit value (DIGITAL IN0...DIGITAL IN15). As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"><li>• See <a href="#">Chapter 9.2.1 "16-Bit Digital Input Port", page 274.</a></li></ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">write_io_port</a> , <a href="#">write_io_port_mask</a> , <a href="#">set_io_cond_list</a> , <a href="#">clear_io_cond_list</a> , <a href="#">read_io_port_buffer</a> , <a href="#">read_io_port_list</a>



<b>Ctrl Command</b>	<b>read_io_port_buffer</b>
<b>Function</b>	Returns the data previously stored in the IOPort buffer by <b>read_io_port_list</b> (input value read at the EXTENSION 1 socket connector's 16-bit digital input; the galvanometer set position and time value that was current at the time of reading).
<b>Call</b>	currentIndex = read_io_port_buffer( Index, &Value, &XPos, &YPos, &Time )
<b>Parameters</b>	<p>Index      Number of the to-be-returned entry in the IOPort buffer. As an unsigned 32-bit value. Only the 12 least significant bits are evaluated (the IOPort buffer only has 4,096 entries). Therefore, the user program can use a continuous counter for the index.</p>
<b>Returned parameter values</b>	<p>Value      The 16-bit value stored in the IOPort buffer under <code>Index</code>. As a pointer to an unsigned 32-bit value.</p>
	<p>XPos      The galvanometer scanner set positions (x and y values) stored in the IOPort buffer under <code>Index</code>. (x value). As a pointer to a signed 32-bit value.</p>
	<p>YPos      The galvanometer scanner set positions (x and y values) stored in the IOPort buffer under <code>Index</code>. (y value). As a pointer to a signed 32-bit value.</p>
	<p>Time      The time in seconds stored in the IOPort buffer under <code>Index</code>. As a pointer to an unsigned 32-bit value.</p>
<b>Result</b>	The index to which data is written upon the next <b>read_io_port_list</b> . As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also comments for <b>read_io_port_list</b>.</li> <li>If no <b>read_io_port_list</b> command has been called before <b>read_io_port_buffer</b>, then the initialization values (default: 0) are returned.</li> </ul>
<b>Example (C/C++)</b>	<p>Wait until the data under <code>Index</code> gets newly written:</p> <pre>while (Index == read_io_port_buffer( Index, &amp;Value, &amp;XPos, &amp;YPos, &amp;Time ));</pre> <p>Read all data from <code>Index</code> to the current (not yet newly written) read position:</p> <pre>while (Index != read_io_port_buffer( Index, &amp;Value, &amp;XPos, &amp;YPos, &amp;Time )) { Index = (Index+1) &amp; 0xf; ...}</pre>
<b>RTC4→RTC6</b>	New command.
<b>RTC5→RTC6</b>	Unchanged functionality. Larger IOPort buffer.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>read_io_port_list</b>



<b>Undelayed Short List Command</b>	<b>read_io_port_list</b>
Function	Writes into the internal IOPort buffer the current state (DIGITAL IN0...DIGITAL IN15) of the EXTENSION 1 socket connector's 16-bit digital input. At the same time, the current xy set positions and (relative) time in seconds are stored.
Call	read_io_port_list()
Comments	<ul style="list-style-type: none"> <li>• <b>read_io_port_list</b> does not return values. However, the values can be subsequently queried from the IOPort buffer by the <b>read_io_port_buffer</b> command.</li> <li>• The IOPort buffer holds 4,096 entries and is circularly organized. It always stores the 4,096 most recent entries and overwrites older entries without warning. The incremental Index starts after a reset (program start) at Index 0, and after passing 4,095 does rollover to 0.</li> <li>• The stored time is the current value of an internal seconds counter that is set to 0 at program start (<b>load_program_file</b>) or by <b>time_update</b>.</li> <li>• See also <b>Chapter 9.2.1 "16-Bit Digital Input Port", page 274</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality. Larger IOPort buffer.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>read_io_port_buffer</b> , <b>read_io_port</b>



<b>Ctrl Command</b>	<b>read_mcbsp</b>
<b>Function</b>	Returns the most recent input value that was fully copied to an internal memory location by the <b>McBSP interface</b> .
<b>Call</b>	<code>mcbsp_value = read_mcbsp( No )</code>
<b>Parameters</b>	<p>No                  Number of the internal memory location whose input value should be queried. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>0    Memory location for Processing-on-the-fly applications and for <b>set_mcbsp_in</b> input with coding Bit #31 = 0.</li> <li>1    Memory locations for <b>Online Positioning</b> and for <b>set_mcbsp_in</b> input.</li> <li>2    Wie 1.</li> <li>3    Memory location for <b>set_mcbsp_in</b> input with coding Bit #31 = 1.</li> </ul> <p>For <b>set_multi_mcbsp_in</b>, the following applies: memory locations 0 through 3 are continuously written to as a ring buffer. Only the two least significant bits are evaluated.</p>
<b>Result</b>	Input value. As a signed 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For information on the <b>McBSP interface</b> and the related memory locations, see also <a href="#">Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73</a>.</li> <li>• It is up to users whether to interpret the returned value as one signed or unsigned 32-bit data word, as two signed 16-bit data words or as two signed 15-bit data words: <ul style="list-style-type: none"> <li>– For Processing-on-the-fly applications (<math>No = 0</math>) see <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Chapter 9.3.4 "Synchronization and Online Positioning by McBSP Signals", page 286</a>.</li> <li>– For "<b>Local Online Positioning</b>" (<math>No = 1\dots 2</math>) see <a href="#">Chapter 8.3.1 ""Local Online Positioning"", page 214</a>.</li> <li>– For "<b>Global Online Positioning</b>" (<math>No = 1\dots 2</math>) see <a href="#">Chapter 8.3.2 ""Global Online Positioning"", page 217</a>.</li> <li>– For <b>set_mcbsp_in</b> input (<math>No = 0\dots 3</math>) see <a href="#">Section "Correction via McBSP Interface with Additional McBSP Input", page 231</a> and <a href="#">Section "Correction via McBSP Interface with Additional McBSP Input", page 233</a>.</li> <li>– For <b>set_multi_mcbsp_in</b>, see <a href="#">page 670</a>.</li> </ul> </li> <li>• After <b>load_program_file</b>, the input values at the <b>McBSP interface</b> are transferred to location 0 (default) even if no Processing-on-the-fly correction is activated.</li> <li>• The <b>McBSP interface</b> always ignores the first FrameSync signal that follows a <b>load_program_file</b> or <b>mcbsp_init</b>, so any available data is not transferred (see <a href="#">page 75</a>).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mcbsp_out</a> , <a href="#">mcbsp_init</a> , <a href="#">set_fly_x_pos</a> , <a href="#">set_fly_y_pos</a> , <a href="#">set_fly_rot_pos</a> , <a href="#">set_mcbsp_x</a> , <a href="#">set_mcbsp_y</a> , <a href="#">set_mcbsp_rot</a> , <a href="#">set_mcbsp_matrix</a> , <a href="#">apply_mcbsp</a> , <a href="#">set_mcbsp_global_x</a> , <a href="#">set_mcbsp_global_y</a> , <a href="#">set_mcbsp_global_rot</a> , <a href="#">set_mcbsp_global_matrix</a>



<b>Ctrl Command</b>	<b>read_multi_mcbsp</b>
<b>Function</b>	Queries the <b>McBSP</b> multi transmission input value that was most recently type-sorted and stored.
<b>Call</b>	<code>mcbsp_value = read_multi_mcbsp( No )</code>
<b>Parameters</b>	<p>No              Number of the type-sorted internal memory location. <code>No</code> corresponds to coding bits #0...2 of the <b>McBSP</b> multi input.  As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>0    Memory location for the x coordinate of the Processing-on-the-fly application.</li> <li>1    Memory location for the y coordinate of the Processing-on-the-fly application.</li> <li>2    Memory location for the z coordinate of the Processing-on-the-fly application.</li> <li>3    Memory location for the laser power P.</li> <li>4    Memory location for extra parameter E1.</li> <li>5    Memory location for extra parameter E2.</li> <li>6    Memory location for extra parameter E3.</li> <li>7    Memory location for extra parameter E4.</li> </ul> <p>Only the three least significant bits are evaluated.</p>
<b>Result</b>	Memory value. As a signed 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• You must have previously activated and used <b>McBSP</b> multi transmission by <b>set_multi_mcbsp_in</b> or <b>set_multi_mcbsp_in_list</b>. Otherwise, default or old values are returned.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>set_multi_mcbsp_in</b> , <b>set_multi_mcbsp_in_list</b>



<b>Ctrl Command</b>	<b>read_status</b>																				
<b>Function</b>	Returns the RTC6 list status, see <a href="#">Chapter 6.4.2 "List Status", page 97</a> .																				
<b>Call</b>	Status = read_status()																				
<b>Result</b>	<p>List status. As an unsigned 32-bit value.</p> <table> <thead> <tr> <th>Bit #</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Bit #0 (<b>LSB</b>)</td> <td>LOAD1</td> <td>= 1: indicates that the input pointer is currently in "List 1", that is, that all following list commands are stored in "List 1". LOAD1 is set if the input pointer is set into "List 1" (for example, by <a href="#">set_start_list_pos</a>). LOAD1 is reset if a <a href="#">set_end_of_list</a> is written to "List 1" (READY1 set) or if LOAD2 is set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #1</td> <td>LOAD2</td> <td>= 1: indicates that the input pointer is currently in "List 2", that is, that all following list commands are stored in "List 2". LOAD2 is set if the input pointer is set into "List 2" (for example, by <a href="#">set_start_list_pos</a>). LOAD2 is reset if a <a href="#">set_end_of_list</a> is written to "List 2" (READY2 set) or if LOAD1 is set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #2</td> <td>READY1</td> <td>= 1: indicates that during the loading procedure a <a href="#">set_end_of_list</a> was written to "List 1". READY1 is reset if LOAD1 is newly set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #3</td> <td>READY2</td> <td>= 1: indicates that during the loading procedure a <a href="#">set_end_of_list</a> was written to "List 2". READY2 is reset if LOAD2 is newly set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #4</td> <td>BUSY1</td> <td>= 1: indicates that "List 1" is executing at the moment (or more precisely: that the output pointer currently resides in "List 1" after execution of "List 1" or "List 2" was started). BUSY1 is set by starting execution of "List 1" (for example, by <a href="#">execute_list_pos</a>) or by a list change to "List 1" (automatic list change or jump). BUSY1 is reset if <a href="#">set_end_of_list</a> is executed in "List 1", if a jump into "List 2" is executed (for example, <a href="#">list_jump_pos</a>), if "List 2" is started (for example, by <a href="#">execute_list_pos</a>) or if <a href="#">stop_execution</a> is executed.</td> </tr> </tbody> </table>			Bit #	Name	Description	Bit #0 ( <b>LSB</b> )	LOAD1	= 1: indicates that the input pointer is currently in "List 1", that is, that all following list commands are stored in "List 1". LOAD1 is set if the input pointer is set into "List 1" (for example, by <a href="#">set_start_list_pos</a> ). LOAD1 is reset if a <a href="#">set_end_of_list</a> is written to "List 1" (READY1 set) or if LOAD2 is set (for example, by <a href="#">set_start_list_pos</a> ).	Bit #1	LOAD2	= 1: indicates that the input pointer is currently in "List 2", that is, that all following list commands are stored in "List 2". LOAD2 is set if the input pointer is set into "List 2" (for example, by <a href="#">set_start_list_pos</a> ). LOAD2 is reset if a <a href="#">set_end_of_list</a> is written to "List 2" (READY2 set) or if LOAD1 is set (for example, by <a href="#">set_start_list_pos</a> ).	Bit #2	READY1	= 1: indicates that during the loading procedure a <a href="#">set_end_of_list</a> was written to "List 1". READY1 is reset if LOAD1 is newly set (for example, by <a href="#">set_start_list_pos</a> ).	Bit #3	READY2	= 1: indicates that during the loading procedure a <a href="#">set_end_of_list</a> was written to "List 2". READY2 is reset if LOAD2 is newly set (for example, by <a href="#">set_start_list_pos</a> ).	Bit #4	BUSY1	= 1: indicates that "List 1" is executing at the moment (or more precisely: that the output pointer currently resides in "List 1" after execution of "List 1" or "List 2" was started). BUSY1 is set by starting execution of "List 1" (for example, by <a href="#">execute_list_pos</a> ) or by a list change to "List 1" (automatic list change or jump). BUSY1 is reset if <a href="#">set_end_of_list</a> is executed in "List 1", if a jump into "List 2" is executed (for example, <a href="#">list_jump_pos</a> ), if "List 2" is started (for example, by <a href="#">execute_list_pos</a> ) or if <a href="#">stop_execution</a> is executed.
Bit #	Name	Description																			
Bit #0 ( <b>LSB</b> )	LOAD1	= 1: indicates that the input pointer is currently in "List 1", that is, that all following list commands are stored in "List 1". LOAD1 is set if the input pointer is set into "List 1" (for example, by <a href="#">set_start_list_pos</a> ). LOAD1 is reset if a <a href="#">set_end_of_list</a> is written to "List 1" (READY1 set) or if LOAD2 is set (for example, by <a href="#">set_start_list_pos</a> ).																			
Bit #1	LOAD2	= 1: indicates that the input pointer is currently in "List 2", that is, that all following list commands are stored in "List 2". LOAD2 is set if the input pointer is set into "List 2" (for example, by <a href="#">set_start_list_pos</a> ). LOAD2 is reset if a <a href="#">set_end_of_list</a> is written to "List 2" (READY2 set) or if LOAD1 is set (for example, by <a href="#">set_start_list_pos</a> ).																			
Bit #2	READY1	= 1: indicates that during the loading procedure a <a href="#">set_end_of_list</a> was written to "List 1". READY1 is reset if LOAD1 is newly set (for example, by <a href="#">set_start_list_pos</a> ).																			
Bit #3	READY2	= 1: indicates that during the loading procedure a <a href="#">set_end_of_list</a> was written to "List 2". READY2 is reset if LOAD2 is newly set (for example, by <a href="#">set_start_list_pos</a> ).																			
Bit #4	BUSY1	= 1: indicates that "List 1" is executing at the moment (or more precisely: that the output pointer currently resides in "List 1" after execution of "List 1" or "List 2" was started). BUSY1 is set by starting execution of "List 1" (for example, by <a href="#">execute_list_pos</a> ) or by a list change to "List 1" (automatic list change or jump). BUSY1 is reset if <a href="#">set_end_of_list</a> is executed in "List 1", if a jump into "List 2" is executed (for example, <a href="#">list_jump_pos</a> ), if "List 2" is started (for example, by <a href="#">execute_list_pos</a> ) or if <a href="#">stop_execution</a> is executed.																			



Ctrl Command	read_status			
Result (cont'd)	Bit #5	BUSY2	= 1:	indicates that "List 2" is executing at the moment (or more precisely: that the output pointer currently resides in "List 2" after execution of "List 1" or "List 2" was started). BUSY2 is set by starting execution of "List 2" (for example, by <a href="#">execute_list_pos</a> ) or by a list change to "List 2" (automatic list change or jump). BUSY2 is reset if <a href="#">set_end_of_list</a> is executed in "List 2", if a jump into "List 1" is executed (for example, <a href="#">list_jump_pos</a> ), if "List 1" is started (for example, by <a href="#">execute_list_pos</a> ) or if <a href="#">stop_execution</a> is executed.
	Bit #6	USED1	= 1:	indicates that a <a href="#">set_end_of_list</a> was reached during processing of "List 1". USED1 is reset when LOAD1 is set (for example, by <a href="#">set_start_list_pos</a> ).
	Bit #7	USED2	= 1:	indicates that a <a href="#">set_end_of_list</a> was reached during processing of "List 2". USED2 is reset when LOAD2 is set (for example, by <a href="#">set_start_list_pos</a> ).
	Bit #8		1	
	...			
Comments	<ul style="list-style-type: none"> <li>When interpreting the status values returned by <a href="#">read_status</a>, always take into account the programmed loading or execution processes of the lists.</li> </ul> <p>Under some circumstances, the status values can be misleading, as illustrated by the following examples:</p> <ul style="list-style-type: none"> <li>– Even during a loading process, a list's LOAD status can already have been reset and its READY status set if – after loading of a <a href="#">set_end_of_list</a> into a list – further list commands are loaded into the same list.</li> <li>– The status values remain unchanged if a <a href="#">set_end_of_list</a> is overwritten with another command (READY is not reset).</li> <li>– If – during a loading process – a <a href="#">set_end_of_list</a> is processed at the same time in the same list, then the list is regarded as already processed (USED status set), even though it is still newly loaded (USED was then initially reset).</li> <li>– Even if a completely loaded command list (incl. <a href="#">set_end_of_list</a>) has been stored, a list's READY status can be reset if the input pointer was newly set (for example, by <a href="#">set_input_pointer</a>) into the list. Then a list's USED status can be reset too, even though a completely loaded and already processed command list is stored.</li> <li>– For jumps from one list area to another ("List 1" &lt;-&gt; "List 2", for example, by <a href="#">list_jump_pos</a>) during execution, the USED status values of both lists (unlike their BUSY status values) remain unchanged and are therefore not meaningful.</li> </ul>			



Ctrl Command	<a href="#">read_status</a>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If the list status is queried during processing of a subroutine in the protected list memory area "List 3", then the status is returned of the list ("List 1" or "List 2") in which the output pointer most recently resided (typically from where the subroutine was originally called).</li> <li>If list execution is interrupted (by <a href="#">pause_list</a>, <a href="#">stop_list</a> or <a href="#">set_wait</a>), then the above-mentioned status values remains unchanged.</li> <li>The list execution statuses <a href="#">BUSY</a> and <a href="#">PAUSED</a> (see Chapter 6.4.3 "List Execution Status", page 98) can be queried by <a href="#">get_status</a>.</li> <li>To read the status signals from the scan heads, use <a href="#">get_head_status</a>.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: Additional USED status.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_status</a> , <a href="#">get_head_status</a>

Ctrl Command	<a href="#">read_user_data</a>
Function	Queries the status information currently returned from the scan system by the user data bit.
Comments	<ul style="list-style-type: none"> <li>Up to now, this command has no effect.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">send_user_data</a>



<b>Ctrl Command</b>	<b>release_rtc</b>
<b>Function</b>	Releases the specified RTC6 for use by other user programs.
<b>Call</b>	NoOfReleasedCard = release_rtc( CardNo )
<b>Parameters</b>	CardNo <b>RTC6 DLL</b> -internal number of the RTC6 board. As an unsigned 32-bit value.
<b>Result</b>	The returned value is CardNo if the user program was still in possession of access rights for this board. Otherwise, 0 is returned. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• After the user program releases a board with <b>release_rtc</b>, it no longer has access rights for this board. The board can then be subsequently acquired by the same or another user program by <b>acquire_rtc</b> or <b>init_rtc6_dll</b>.</li> <li>• If a board released by <b>release_rtc</b> was the active board, then (other than multi-board commands) only those non-multi-board commands not requiring explicit access rights are subsequently available. Another board is not automatically selected as the active board. Activation of another board (for which access rights are assigned to the user program) can be achieved by the <b>select_rtc</b> command.</li> <li>• The <b>release_rtc</b> command is available even without explicit access rights for a particular RTC6 board, but the command then has no effect (return value 0).</li> <li>• The <b>release_rtc</b> command also has no effect (return value 0), if: <ul style="list-style-type: none"> <li>– CardNo exceeds the number of RTC6 boards found during initialization (see <b>rtc6_count_cards</b>) and</li> <li>– No RTC6 Ethernet Board is entered at CardNo</li> <li>– CardNo = 0 (real boards begin at 1)</li> </ul> </li> <li>• <b>release_rtc</b> is not available as a multi-board command.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>acquire_rtc, init_rtc6_dll</b>



<b>Ctrl Command</b>	<b>release_wait</b>
<b>Function</b>	Resumes processing of a list that has been interrupted by <a href="#">set_wait</a> .
<b>Call</b>	<code>release_wait()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>release_wait</b> is only executed if the RTC6 is actually in a wait state (hence if a break point has been previously reached and list processing has been interrupted; the <b>PAUSED</b> status is then set, the <b>BUSY</b> status is <i>not</i> set). Otherwise, <b>release_wait</b> is ignored (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>).</li> <li>• By <b>release_wait</b>, the <b>PAUSED</b> status (queriable with <a href="#">get_status</a>) is reset and the <b>BUSY</b> status is newly set, see also <a href="#">Chapter 6.4.3 "List Execution Status", page 98</a>.</li> <li>• <b>release_wait</b> resets the <a href="#">WaitWord</a> that receives the break point number during an interrupt to zero.</li> <li>• If a home jump (defined by <a href="#">home_position</a> or <a href="#">home_position_xyz</a>) has been executed by <a href="#">set_wait</a>, then <b>release_wait</b> leads to a corresponding home return (the <b>INTERNAL-BUSY</b> status is set while the home return is executed).</li> <li>• The wait state can be queried by <a href="#">get_wait_status</a>.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: Additional <b>PAUSED</b> status. It is set by <a href="#">set_wait</a> and reset with <a href="#">release_wait</a> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_wait</a> , <a href="#">get_wait_status</a> , <a href="#">restart_list</a>



<b>Ctrl Command</b>	<b>reset_error</b>
<b>Function</b>	Resets the cumulative error code.
<b>Call</b>	<code>reset_error( Code )</code>
<b>Parameters</b>	Code      OR connection of the error codes = sum by means of $2^{\text{Bitnumber}}$ of all bits to be reset. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>For error handling see <a href="#">Chapter 6.8 "Error Handling", page 120</a>.</li> <li>The cumulative error code can be reset: <ul style="list-style-type: none"> <li>– Bitwise (individually for each error type, for example, Bit #5 and Bit #6 by <code>Code = 2^5   2^6</code> or by <code>Code = RTC6_BUSY   RTC6_REJECTED</code>)</li> <li>– Completely (by <code>Code = "-1"</code>, therefore by <code>Code = 2^32 - 1</code>)</li> </ul> The meanings of bit numbers, error types and error constants is described at <a href="#">get_error</a>.</li> <li><code>reset_error</code> does not delete the error code of another user program currently assigned access rights to the board.</li> <li><code>reset_error</code> and <code>n_reset_error</code> are available even without explicit access rights to a specific RTC6 board.</li> <li>The board-specific error variable <code>LastError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) is neither generated nor altered by <code>reset_error</code>. In contrast, <code>AccError</code> is altered as specified by <code>Code</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_error</a> , <a href="#">get_last_error</a>



<b>Ctrl Command</b>	<b>restart_list</b>
<b>Function</b>	Reenables "laser active" laser control signals and resumes execution of a list that was interrupted by <b>pause_list</b> or <b>stop_list</b> .
<b>Call</b>	<code>restart_list()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The <b>restart_list</b> command is only executed if a list was previously halted by <b>pause_list</b> or <b>stop_list</b> and if the <b>BUSY</b> and <b>PAUSED</b> statuses (queryable with <b>get_status</b>) are set. Otherwise (for example, if a list was halted with <b>set_wait</b>), the command is ignored (<b>get_last_error</b> return code <b>RTC6_BUSY</b>).</li> <li><b>restart_list</b> resets the <b>PAUSED</b> status. The <b>BUSY</b> status is left unchanged.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: Additional <b>PAUSED</b> status that is set with <b>pause_list</b> and <b>stop_list</b> and reset with <b>restart_list</b> .
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>pause_list</b> , <b>stop_list</b> , <b>release_wait</b>

<b>Ctrl Command</b>	<b>rs232_config</b>
<b>Function</b>	Configures the RS-232 interface for the specified baud rate.
<b>Call</b>	<code>rs232_config( BaudRate )</code>
<b>Parameters</b>	<p><b>BaudRate</b>    Baud rate.            As an unsigned 32-bit value.            Allowed value range: [160 Bd...+2.8 MBd].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <b>Chapter 4.6.5 "RS232 Socket Connector"</b>, page 72.</li> <li><b>rs232_config</b> is synonymous with <b>uart_config</b>. However, <b>rs232_config</b> has no result.</li> <li>Up to DLL 611: value range [300 Bd... +115.200 Bd].</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600. As of DLL 611: extended value range.
<b>References</b>	<b>rs232_write_data</b> , <b>rs232_read_data</b> , <b>uart_config</b>



<b>Ctrl Command</b>	<b>rs232_read_data</b>
<b>Function</b>	Reads a value from the input buffer of the RS-232 interface, see <a href="#">Chapter 9.2.3 "RS-232 Interface", page 274.</a>
<b>Call</b>	<code>RS232Data = rs232_read_data()</code>
<b>Result</b>	<p>As an unsigned 32-bit value.</p> <p>Bit #0            Next (not yet read by the user program) value of the input buffer. (LSB)</p> <p>...</p> <p>Bit #7</p> <p>Bit #8            "New" bit: = 1:    The value is new (was not previously read). = 0:    The value is old (was already read once by <code>rs232_read_data</code>).</p> <p>Bit #9            0.</p> <p>...</p> <p>Bit #15</p> <p>Bit #16          Number of further (not yet read) characters.</p> <p>...</p> <p>Bit #23</p> <p>Bit #24          Number of buffer overruns.</p> <p>...</p> <p>Bit #31</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The RS-232 interface is internally read in internally asynchronously (one character at a time). If this character is new, then it is stored in a 256-character ring buffer. From there, it can be transferred to the user program by <code>rs232_read_data</code> (asynchronously to reading).</li> <li>Byte #0 (<a href="#">Bit #0...Bit #7</a>) returns only one character from the current reading position.</li> <li>Byte #1 (<a href="#">Bit #8</a>) indicates if the character has already been read by the user program.</li> <li>Byte #2 (<a href="#">Bit #16...Bit #23</a>) indicates the number of characters in the input buffer that still have not been read by the user program. If no unread characters are present (byte #2 = 0), then the most recently read character is transferred with "new bit" = 0.</li> <li>Byte #3 (<a href="#">Bit #24...Bit #31</a>) indicates the number of overruns of the input buffer (a corresponding number of characters were overwritten and therefore irretrievably lost). To reset the overrun counter, call <code>rs232_read_data</code> that many times.</li> <li>Example: return value 459098 = 0x0007015A = (0, 7, 1, 90) means: character 90 ('Z') was read and is new (1), 7 additional characters remain to be read, the buffer was never overrun (0).</li> </ul>
<a href="#">RTC4→RTC6</a>	New command.
<a href="#">RTC5→RTC6</a>	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">rs232_write_data</a> , <a href="#">uart_config</a>



<b>Ctrl Command</b>	<b>rs232_write_data</b>
<b>Function</b>	Sends a data word (byte) to the RS-232 interface.
<b>Call</b>	<code>rs232_write_data( Data )</code>
<b>Parameters</b>	<p>Data      Data word. As an unsigned 32-bit value. Only the least significant byte is transferred to the RS-232 interface.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The complete transmission of any previous data words is waited for. An overrun at the interface is not possible.</li> <li>See also <a href="#">Chapter 9.1.6 "RS-232 Interface", page 273</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">rs232_write_text</a> , <a href="#">rs232_read_data</a> , <a href="#">uart_config</a> , <a href="#">rs232_write_text_list</a>

<b>Ctrl Command</b>	<b>rs232_write_text</b>
<b>Function</b>	Sends a text string (character-by-character) to the RS-232 interface.
<b>Call</b>	<code>rs232_write_text( pData )</code>
<b>Parameters</b>	<p>pData      PC memory address of the first character (byte) of the to-be-sent text string. As a pointer to a \0-terminated string.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li><b>rs232_write_text</b> is split into an appropriate number of <a href="#">rs232_write_data</a>.</li> <li>During execution of <b>rs232_write_text</b> no further control commands can be executed, but the list execution on the board is not affected. If an executing list itself contains <a href="#">rs232_write_text_list</a>, <b>rs232_write_text</b> should preferably not be used so as to avoid conflicts (race conditions).</li> <li>See also <a href="#">Chapter 9.1.6 "RS-232 Interface", page 273</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">rs232_write_data</a> , <a href="#">rs232_write_text_list</a> , <a href="#">uart_config</a>



<b>Variable List Command</b>	<b>rs232_write_text_list</b>
Function	Sends a text string (character-by-character) to the RS-232 interface.
Call	<code>rs232_write_text_list( pData )</code>
Parameters	<code>pData</code> PC memory address of the first character (byte) of the to-be-sent text string. As a pointer to a \0-terminated string.
Comments	<ul style="list-style-type: none"> <li>When an <b>rs232_write_text_list</b> is loaded, the to-be-sent text (if more than 12 characters in length, \0 not included) is split into blocks of 12 characters, with each block receiving its own <b>rs232_write_text_list</b> in the list memory (keep this in mind to prevent unintended overflow of the corresponding buffer area). Processing of the individual <b>rs232_write_text_list</b> is similar to that of the <b>rs232_write_text</b> command (the 12-character block is split into individual characters and sent sequentially to the RS-232 interface).</li> <li><b>rs232_write_text_list</b> takes some time for execution, depending on the baud rate and text length.</li> <li>See also <a href="#">Chapter 9.1.6 "RS-232 Interface", page 273</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">rs232_write_text</a> , <a href="#">rs232_read_data</a> , <a href="#">uart_config</a>



<b>Ctrl Command</b>	<b>rtc6_count_cards</b>
<b>Function</b>	Returns the number of RTC6 PCIe Boards detected during initialization.
<b>Call</b>	NoOfCards = rtc6_count_cards()
<b>Result</b>	Number of RTC6 PCIe Boards. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>Initialization of the installed RTC6 PCIe Boards is to be made separately for each user program by calling <a href="#">init_RTC6_dll</a>.</li> <li><b>rtc6_count_cards</b> is available even without explicit access rights to a specific RTC6 PCIe Board.</li> <li><b>rtc6_count_cards</b> is not available as a multi-board command.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <b>rtc6_count_cards</b>.</li> </ul>
<b>RTC4→RTC6</b>	New command. Functionality is similar to <b>rtc4_count_cards</b> of the RTC4 command set.
<b>RTC5→RTC6</b>	New command. The functionalities of <b>rtc6_count_cards</b> and the RTC5 command <b>rtc5_count_cards</b> are identical.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">init_RTC6_dll</a>



<b>Delayed Short List Command</b>	<b>save_and_restart_timer</b>
Function	Stores the current value of the RTC6 timer and resets it to zero.
Call	<code>save_and_restart_timer()</code>
Comments	<ul style="list-style-type: none"> <li>The stored RTC6 timer value can be read by <a href="#">get_time</a>.</li> <li><code>save_and_restart_timer</code> is useful for measuring the execution time of a marking process, see <a href="#">Chapter 8.12 "Time Measurements", page 267</a>.</li> <li>The RTC6 timer only counts list-command clock cycles. Counting is paused during interruptions by <code>set_wait</code> or <code>pause_list</code>.</li> <li>By <a href="#">get_lap_time</a> the number of elapsed list-command clock cycles since the reset to zero can be queried.</li> <li>To compare RTC6-internal <code>save_and_restart_timer</code> time measurements to external time measurements by the <a href="#">BUSY pin</a>, you should insert a <code>list_nop</code> between <code>save_and_restart_timer</code> and <code>set_end_of_list</code>. This ensures that any scanner delay completes before <code>set_end_of_list</code>. Without <code>list_nop</code>, <code>save_and_restart_timer</code> includes the scanner delay in its measurement even though it completes only after <code>set_end_of_list</code> (and therefore the <a href="#">BUSY pin</a> is already low).</li> <li>A time stamp counter is available in addition to the RTC6 timer. It starts counting at 0 with <a href="#">load_program_file</a> and counts uninterruptible all 10 µs clock periods. See also <a href="#">Chapter 8.12 "Time Measurements", page 267</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_lap_time</a> , <a href="#">get_time</a>



<b>Ctrl Command</b>	<b>save_disk</b>				
<b>Function</b>	Stores all indexed characters, text strings and/or subroutines to a binary file on a PC data medium, ordered by index, and returns the number of thereby stored list commands.				
<b>Call</b>	NoOfSavedCommands = save_disk( Name, Mode )				
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>File name. As a pointer to a \0-terminated ANSI string.</td> </tr> <tr> <td>Mode</td> <td>Specifies what is to be stored:  Bit #0 = 1: All indexed characters and text strings are stored. Bit #1 = 1: All indexed subroutines are stored. Bits #2...31: Are not evaluated.  As an unsigned 32-bit value.</td> </tr> </table>	Name	File name. As a pointer to a \0-terminated ANSI string.	Mode	Specifies what is to be stored:  Bit #0 = 1: All indexed characters and text strings are stored. Bit #1 = 1: All indexed subroutines are stored. Bits #2...31: Are not evaluated.  As an unsigned 32-bit value.
Name	File name. As a pointer to a \0-terminated ANSI string.				
Mode	Specifies what is to be stored:  Bit #0 = 1: All indexed characters and text strings are stored. Bit #1 = 1: All indexed subroutines are stored. Bits #2...31: Are not evaluated.  As an unsigned 32-bit value.				
<b>Result</b>	The number of list commands saved by <b>save_disk</b> . As an unsigned 32-bit value.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>save_disk</b> can be used together with <b>load_disk</b>, for example, to perform defragmentation or to apply subsequent protection to subroutines, see <b>Section "Subsequent Protection and Conversion of Non-Indexed Subroutines"</b>, page 106 and <b>Section "Index Management and Defragmentation"</b>, page 105.</li> <li>• By <b>save_disk</b>, always the complete sets (no individual characters, text strings or subroutines!) are stored in the specified file. Indexed characters, text strings or subroutines that are referenced multiple times with <b>copy_dst_src</b> are correspondingly duplicated by <b>save_disk</b>, that is, also stored multiple times. The <b>save_disk</b> command ignores unreferenced non-indexed (not subsequently referenced by <b>set_char_pointer</b>,...) characters, text strings and subroutines.</li> <li>• <b>save_disk</b> stores to the binary file not only the list commands for characters, text strings and/or subroutines, but also their indexes and the number of commands.</li> <li>• The number of list commands stored with <b>save_disk</b> can differ considerably from the number of the list commands stored in the protected-area "List 3" (<math>= 2^{23} - \text{Mem1} - \text{Mem2} - \text{get_list_space}</math>), due to possible multiple referencing of an indexed character, text string or subroutine or referencing of an indexed character, text string or subroutine in the unprotected list memory area "List 1" or "List 2". Prior to a subsequent <b>load_disk</b>, the returned number must be compared to the size of the protected list memory area "List 3" (<math>= 2^{23} - \text{Mem1} - \text{Mem2}</math>).</li> <li>• No-longer-needed characters (or text strings or subroutines) should be dereferenced by <b>load_char</b> (or <b>load_text_table</b> or <b>load_sub</b>) directly followed by <b>list_return</b> previously to <b>save_disk</b> (see <b>Section "Index Management and Defragmentation"</b>, page 105).</li> </ul>				



<b>Ctrl Command</b>	<b>save_disk</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>save_disk</b> always stores all characters, text strings and/or subroutines from the referenced address to the next possible <b>list_return</b>. Jump commands (also branches to various <b>list_return</b> commands) are thereby neither evaluated nor executed.</li> <li>• The <b>save_disk</b> command automatically replaces unallowed commands (for example, <b>set_end_of_list</b>) with <b>list_nop</b> commands; missing commands (for example, <b>list_return</b> upon reaching the last memory position of "List 3") are added.</li> <li>• <b>save_disk</b> is not executed (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>), if: <ul style="list-style-type: none"> <li>– Mode = 0</li> <li>– Name = <b>NULL</b></li> </ul> </li> <li>• <b>save_disk</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>save_disk</b> is even executed, if: <ul style="list-style-type: none"> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• During the runtime of <b>save_disk</b>, no further commands can be executed.</li> <li>• During the runtime of <b>save_disk</b>, external starts are suppressed.</li> <li>• <b>save_disk</b> also stores version information which is checked by <b>load_disk</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 615: version info.
References	<b>load_disk</b>



<b>Undelayed Short List Command</b>	<b>select_char_set</b>
Function	Selects one of the available character sets (exclusively) for the execution of subsequent <b>mark_text</b> and <b>mark_text_abs</b> .
Call	<code>select_char_set( No )</code>
Parameters	No      Number of the desired character set. As an unsigned 32-bit value. Allowed value range: [0...3].
Comments	<ul style="list-style-type: none"> <li>The selection remains valid until another is encountered.</li> <li>The default value (after initialization) is No = 0.</li> <li>If No &gt; 3, then <b>select_char_set</b> is replaced with a <b>list_nop</b>. The current character set then remains valid (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>A character set change <i>within</i> a <b>mark_text</b> or <b>mark_text_abs</b> is only possible if a <b>select_char_set</b> is located within a (called) indexed character. The selection encountered there then applies to all subsequent characters.</li> <li>If the selected character set is not (fully) defined, then missing characters are not marked (with <b>mark_text</b> or <b>mark_text_abs</b>).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>mark_text</b> , <b>mark_text_abs</b>



<b>Ctrl Command</b>	<b>select_cor_table</b>
<b>Function</b>	Assigns the previously loaded correction tables to the scan head connector and activates image field correction.
<b>Call</b>	<code>select_cor_table( HeadA, HeadB )</code>
<b>Parameters</b>	<p>HeadA = 0: Turns off the signals for scan head A (first scan head connector)  = 1...8: Assigns correction table number HeadA to scan head A.  As an unsigned 32-bit value. See also <a href="#">number_of_correction_tables</a>.</p> <p>HeadB = 0: Turns off the signals for scan head B (second scan head connector).  = 1...8: Assigns correction table number HeadB to scan head B.  Requires <a href="#">Option "Second Scan Head Control"</a>.  As an unsigned 32-bit value. See also <a href="#">number_of_correction_tables</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <a href="#">select_cor_table</a> or <a href="#">select_cor_table_list</a> should be called directly after loading the desired correction table(s) by <a href="#">load_correction_file</a> and/or <a href="#">load_z_table/load_z_table_no</a> (or after a subsequent <a href="#">load_program_file</a> – see below) in order to assign the correction table(s) to the corresponding scan head connector (<a href="#">select_cor_table</a> is automatically called by <a href="#">load_correction_file</a>; see <a href="#">Section "Notes", page 165</a>). <a href="#">select_cor_table</a> and <a href="#">select_cor_table_list</a> issue a jump with jump speed (no “<a href="#">Hard jump</a>”) to the corrected galvanometer scanner position, so that image field correction is immediately applied to the currently set galvanometer scanner position. <a href="#">select_cor_table</a> does this automatically and immediately, whereas <a href="#">select_cor_table_list</a> inserts the jump in front of the next list command. Depending on the table contents and galvanometer scanner position, this can take a few clock cycles (and at least one clock cycle).</li> <li>• Correction tables should be loaded and assigned prior to first-time starting of a list or issuance of a control command (for example, <a href="#">goto_xy</a>) that sets the scan system’s galvanometer scanners in motion, and <a href="#">select_cor_table</a> or <a href="#">select_cor_table_list</a> should only be started <i>after</i> loading of the desired correction table(s). Otherwise, the galvanometers can, in some circumstances, be driven to an unexpected position and the laser beam deflected in an unintended direction.</li> <li>• Correction tables, loaded (by <a href="#">load_correction_file</a>) <i>prior</i> to <a href="#">load_program_file</a>, are not fully effective before <a href="#">select_cor_table</a> or <a href="#">select_cor_table_list</a> is called.</li> <li>• <a href="#">select_cor_table</a> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC6_BUSY</a>), if: <ul style="list-style-type: none"> <li>– the <a href="#">BUSY</a> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li> </ul> The list command <a href="#">select_cor_table_list</a> can be used without restrictions. </li> <li>• <a href="#">select_cor_table</a> is even executed, if: <ul style="list-style-type: none"> <li>– a list has been paused by <a href="#">set_wait</a> (<a href="#">PAUSED</a> status set)</li> </ul> </li> <li>• If the <a href="#">INTERNAL-BUSY</a> status of the board is set, <a href="#">select_cor_table</a> is only executed with a delay (after <a href="#">INTERNAL-BUSY</a> has been reset again).</li> <li>• The <a href="#">INTERNAL-BUSY</a> status is set while the jump to the corrected galvanometer scanner position is executed.</li> </ul>

Ctrl Command	select_cor_table
Comments (cont'd)	<ul style="list-style-type: none"> <li>If the values for parameters HeadA or HeadB are invalid (values &gt; <b>number_of_correction_tables</b>) or 0, then <b>select_cor_table</b> turns off the corresponding scan head signals. The galvanometer scanners then remain in their last position.</li> <li>If the <b>Option "Second Scan Head Control"</b> is not enabled, <b>select_cor_table</b> does not assign a correction table to the second scan head connector.</li> <li>The default setting (after <b>load_program_file</b>) is (1,0), that is, correction table #1 is used for scan head A, whereas the output signals for scan head B are turned off (this corresponds to parameter values HeadA = 1 and HeadB = 0). Initially however, after <b>load_program_file</b> and until <b>select_cor_table</b> is called or the galvanometer scanners are explicitly moved, the galvanometer scanners stay in the position (0 0), even if the correction table content is different.</li> <li>If the <b>Option "Second Scan Head Control"</b> is not enabled, then signals of the second scan head remain permanently turned off; even so, multiple correction tables can still be loaded and used at any time by the first scan head. Even with one scan head, you can thereby quickly switch back and forth between several correction tables, for example, one for a pointer laser and one for the main laser with a different wavelength, see also <a href="#">Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 222</a>.</li> <li>In a double scan head system, table #1 is typically used for scan head A, and table #2 is used for scan head B: <code>select_cor_table(1,2)</code>. But you can make a different assignment at any time.</li> <li>For 3D systems, the <b>Option "3D"</b> must be enabled. Provided the RTC6's <b>Option "3D"</b> is enabled, you can load several (2D or 3D) correction tables. If the is not enabled, then the xy axis must be connected to the first scan head connector, the z axis to the X or Y channel of the second scan head connector. If a 3D correction table is assigned to the first scan head connector, corrected signals for an xy scan head are then transmitted by the first scan head connector and corrected signals for the z axis are transmitted by both channels of the second scan head connector. If multiple RTC6 boards with enabled <b>Option "3D"</b> are installed in a PC, then that many 3-axis systems can be simultaneously controlled.</li> <li>Provided the <b>Option "3D"</b> and the <b>Option "Second Scan Head Control"</b> are enabled, users specify which signals (xy or z) are to be outputted by which connector when assigning the correction table (see also <a href="#">Chapter 4.5.1 "Scan Head Connectors and Transfer Protocol", page 56</a> and <a href="#">Section "2D and 3D Correction Files", page 163</a>). 2D correction tables should and 3D correction tables can be thereby assigned only to the xy axes and <i>not</i> to the z axis (for example, by <code>select_cor_table(0,1)</code> or <code>select_cor_table(0,2)</code> for xy at the scan head B connector and z at the scan head A connector). Because unexpected system behavior might otherwise occur and the system would not know where the xy axes and z axis are connected, the signals of both connectors are turned off if both connectors have been assigned a correction table and (at least) one of them is a 3D correction table (for example, for <code>select_cor_table(1,1)</code>).</li> </ul>



<b>Ctrl Command</b>	<b>select_cor_table</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Parameters from currently loaded correction tables can be read by <a href="#">get_table_para</a>, or from currently assigned correction tables by <a href="#">get_head_para</a> (see also <a href="#">load_correction_file</a>).</li> </ul>
RTC4→RTC6	<p>Unchanged functionality.</p> <p>Exception: now up to 8 correction tables in total can be stored in RTC6 memory. However, two 3D correction tables can <i>not</i> be simultaneously assigned to the scan head connectors.</p>
RTC5→RTC6	<p>Changed functionality.</p> <ul style="list-style-type: none"> <li>In the RTC6 command set, <b>select_cor_table</b> automatically switches on Zoom and/or Stretch, if a table was previously loaded by <a href="#">load_stretch_table</a> or <a href="#">load_zoom_correction_file</a> to this table number.</li> </ul>
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">select_cor_table_list</a> , <a href="#">load_correction_file</a> , <a href="#">load_program_file</a> , <a href="#">load_stretch_table</a> , <a href="#">number_of_correction_tables</a>

<b>Variable List Command</b>	<b>select_cor_table_list</b>
Function	Like <a href="#">select_cor_table</a> , but a list command.
Call	<code>select_cor_table_list( HeadA, HeadB )</code>
Parameters	<p>HeadA = 0: Turns off the signals for scan head A (first scan head connector). = 1...8: Assigns correction table number HeadA to scan head A. As an unsigned 32-bit value. See also <a href="#">number_of_correction_tables</a>.</p> <p>HeadB = 0: Turns off the signals for scan head B (second scan head connector) = 1...8: Assigns correction table number HeadB to scan head B. Requires <a href="#">Option "Second Scan Head Control"</a>. As an unsigned 32-bit value. See also <a href="#">number_of_correction_tables</a>.</p>
Comments	<ul style="list-style-type: none"> <li>See <a href="#">select_cor_table</a>.</li> <li>Even though <b>select_cor_table_list</b> is a short list command, execution of the directly following list command is delayed by a few clock cycles due to the intermediate jump to the corrected galvanometer scanner position. The extent of this delay depends on the assigned correction table's content for the current galvanometer scanner position (for example, (0 0) after <a href="#">load_program_file</a>); but it is at least 10 µs, even if the correction table does not specify a correction.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">select_cor_table</a>

<b>Ctrl Command</b>	<b>select_rtc</b>
<b>Function</b>	Defines, in a multi-board system, the active RTC6 board for a user program. See <a href="#">Chapter 6.6 "Using Several RTC6 PCIe Boards in One PC", page 113</a> .
<b>Call</b>	NoOfSelectedCard = select_rtc( CardNo )
<b>Parameters</b>	CardNo <b>RTC6 DLL</b> -internal number of the RTC6 board. As an unsigned 32-bit value.
<b>Result</b>	<p>The returned value is:</p> <ul style="list-style-type: none"> <li>CardNo if access rights exist for the specified board or if the specified board is not allocated to another user program (in this latter case, access rights are acquired through <b>select_rtc</b> – as by <b>acquire_rtc</b>, see below).</li> <li>The number of the active board if CardNo exceeds the number of RTC6 boards found during initialization, no RTC6 Ethernet Board is entered there or if CardNo = 0.</li> <li>0 otherwise (particularly when the specified board is reserved by another user program or if a version compatibility error is detected and activation of the board therefore cannot succeed) (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b> and possibly <b>RTC6_VERSION_MISMATCH</b>).</li> </ul> <p>As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Activation of a board for a user program already occurs when the <b>RTC6 DLL</b> for this user program is initialized (see <b>init_rtc6_dll</b>). The <b>select_rtc</b> command offers the possibility of changing the active board at any time. All user program commands subsequent to <b>select_rtc</b> (except for multi-board commands) are forwarded to the corresponding active board.</li> <li>If the specified board is reserved for another user program, <b>select_rtc</b> has no effect (return value 0, <b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>).</li> <li>If the specified board has no access rights granted and is not acquired by another user program, an attempt is made to acquire it (as by <b>acquire_rtc</b>). If a board is acquired (as by <b>acquire_rtc</b>), a version compatibility check is performed. If a version error is detected, then access to the board is denied and <b>select_rtc</b> has no effect (return code 0, <b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>   <b>RTC6_VERSION_MISMATCH</b>).</li> <li>If the specified board is already the active board for this user program, <b>select_rtc</b> has no effect (return value: CardNo).</li> <li>The <b>select_rtc</b> command also has no effect if CardNo exceeds the number of RTC6 boards found during initialization (see <b>rtc6_count_cards</b>), no RTC6 Ethernet Board is entered there or if CardNo = 0 (real boards begin at 1). The return value is then the number of the active board (see above). Therefore, <b>select_rtc( 0 )</b> can be used to determine the number of the active board at any time.</li> <li>The <b>select_rtc</b> command is available even without explicit access rights to a particular RTC6 board.</li> <li><b>select_rtc</b> is not available as a multi-board command.</li> </ul>



Ctrl Command	select_rtc
RTC4→RTC6	With the RTC4, <b>select_rtc</b> only activates the specified board (unconditionally). With the RTC6, <b>select_rtc</b> additionally delivers a return value and tries to get access to the specified board granted (as with <b>acquire_rtc</b> ) or remains ineffective.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–



<b>Ctrl Command</b>	<b>select_serial_set</b>
<b>Function</b>	Selects a serial-number-set for serial number control commands.
<b>Call</b>	<code>select_serial_set( No )</code>
<b>Parameters</b>	No Number of the desired serial-number-set. As an unsigned 32-bit value. Allowed value range: [0...3]. Only the two least significant bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>After initialization with <code>load_program_file</code>, serial-number-set 0 is selected.</li> <li><code>select_serial_set</code> does <i>not</i> set the serial-number-set for serial number marking. The list command <code>select_serial_set_list</code> is intended for that purpose.</li> <li>For usage of <code>select_serial_set</code>, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 197</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">select_serial_set_list</a> , <a href="#">set_serial_step</a> , <a href="#">get_serial</a>

<b>Undelayed Short List Command</b>	<b>select_serial_set_list</b>
<b>Function</b>	Selects a serial-number-set for serial number list commands (as well as for serial number marking).
<b>Call</b>	<code>select_serial_set_list( No )</code>
<b>Parameters</b>	No Number of the desired serial-number-set. As an unsigned 32-bit value. Allowed value range: [0...3]. Only the two least significant bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>After initialization with <code>load_program_file</code>, serial-number-set 0 is selected.</li> <li>For usage of <code>select_serial_set_list</code>, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 197</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">select_serial_set</a> , <a href="#">set_serial_step_list</a> , <a href="#">get_list_serial</a> , <a href="#">mark_serial</a> , <a href="#">mark_serial_abs</a>



<b>Ctrl Command</b>	<b>send_user_data</b>
Function	Sends data values by the user data bit to the specified scan system.
Comments	<ul style="list-style-type: none"><li>• To date, this command has no effect.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#"><b>read_user_data</b></a>



<b>Ctrl Command</b>	<b>set_angle</b>
<b>Function</b>	Uses a specified rotation angle to define the rotation matrix $M_R$ for all subsequent coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 210</a> .
<b>Call</b>	<code>set_angle( HeadNo, Angle, at_once )</code>
<b>Parameters</b>	<p><b>HeadNo</b> Number of the scan head connector. As an unsigned 32-bit value.            = 1: The definition only affects the <i>first</i> scan head connector.            = 2: The definition only affects the <i>second</i> scan head connector.            = 0, 3: The definition affects <i>both</i> scan head connectors.            = 4: The definition affects the virtual image field (see also comments). Only the two least significant bits are evaluated.</p> <p><b>Angle</b> Rotation angle. In degrees.            As a 64-bit IEEE floating point value.            Positive angles result in a counterclockwise rotation around the centerpoint of the image field.</p> <p><b>at_once</b> Determines when the defined transformation becomes effective.            As an unsigned 32-bit value.</p> <p>For <code>HeadNo = 0...3</code>, the following applies:</p> <ul style="list-style-type: none"> <li>= 0: The new total transformation (total matrix and offset) is only calculated and applied to the current position when the next list command is executed.</li> <li>= 1: The new total transformation is calculated immediately (or before the next list command if a list is currently <b>BUSY</b> or the board is <b>INTERNAL-BUSY</b>) and applied to the current position. The signals for "laser active" operation are switched off in advance if necessary.</li> <li>= 2: The new total transformation (total matrix and offset) is only calculated and applied to the current position when the next <b>jump_abs</b>, <b>jump_rel</b>, <b>goto_xy</b> or <b>goto_xyz</b> is executed.</li> <li>= 3: Like <code>at_once = 1</code>, but the laser control signals remain unaffected.</li> <li>&gt; 3: Like <code>at_once = 2</code>.</li> </ul> <p>For <code>HeadNo = 4</code>, the following applies:</p> <ul style="list-style-type: none"> <li>= 0: Only saved. Is applied at the next <b>Processing-on-the-fly session</b> start to the current position.</li> <li>= 1: Is applied immediately to the current position if currently no list is <b>BUSY</b> or no board is <b>INTERNAL-BUSY</b>. Otherwise like 0.</li> <li>= 2, 3: Like 0.</li> </ul>



<b>Ctrl Command</b>	<b>set_angle</b>
Comments	<ul style="list-style-type: none"><li>• See also <a href="#">Chapter 8.2 "Coordinate Transformations", page 210.</a></li><li>• As of DLL 617, OUT 617 the following applies for HeadNo = 4: The global coordinate transformations are generally available, even outside a <a href="#">Processing-on-the-fly session</a>.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Last change version DLL 617, OUT 617: HeadNo = 4.
References	<a href="#">set_angle_list</a> , <a href="#">set_matrix</a> , <a href="#">set_offset</a> , <a href="#">set_scale</a>



<b>Variable List Command</b>	<b>set_angle_list</b>
<b>Function</b>	Like <a href="#">set_angle</a> , but a list command.
<b>Call</b>	<code>set_angle_list( HeadNo, Angle, at_once )</code>
<b>Parameters</b>	<p>HeadNo      See <a href="#">set_angle</a>.</p> <p>Angle      See <a href="#">set_angle</a>.</p> <p>at_once      Determines when the defined transformation becomes effective. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>= 0:      The transformation settings are only accumulated and intermediately stored, but the transformation is not processed as long as this is not activated by another coordinate transformation (for example, by a list command with <code>at_once = 1</code> or a corresponding control command).</li> <li>= 1:      The transformation is immediately calculated (including all transformation settings that were accumulated until then) and processed prior to the next list command. The signals for "laser active" operation are switched off in advance if necessary.</li> <li>= 2:      The transformation settings are only accumulated and intermediately stored (as with <code>at_once = 0</code>). However, The transformation is immediately calculated (including all transformation settings that were accumulated and intermediately stored until then) and applied to the current position when the next <a href="#">jump_abs</a>, <a href="#">jump_rel</a>, <a href="#">goto_xy</a> or <a href="#">goto_xyz</a> is executed.</li> <li>= 3:      Like <code>at_once = 1</code>, but the laser control signals remain unaffected.</li> <li>&gt; 3:      Like <code>at_once = 2</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_angle</a>



<b>Ctrl Command</b>	<b>set_auto_laser_control</b>	
<b>Function</b>	Initializes or deactivates "Automatic Laser Control", see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a> , page 186.	
<b>Call</b>	ErrorCode = set_auto_laser_control( Ctrl, Value, Mode, MinValue, MaxValue )	
<b>Parameters</b>	<p><b>Ctrl</b></p> <p>Control parameter. As an unsigned 32-bit value.</p> <p>= 1...7: Defines which signal parameter is corrected by the "Automatic Laser Control".</p> <ul style="list-style-type: none"> <li>= 1: 12-bit output value at the ANALOG OUT1 output port.</li> <li>= 2: 12-bit output value at the ANALOG OUT2 output port.</li> <li>= 3: Output value at the 8-bit digital output port.</li> <li>= 4: Pulse length (<code>PulseLength</code>) of the laser signals LASER1 and LASER2.</li> <li>= 5: Output period (<code>HalfPeriod</code>) of the laser signals LASER1 and LASER2.</li> <li>= 6: Output value at the 16-bit digital output.</li> <li>= 7: As 5. However, the pulse distance is geometrically constant.</li> </ul> <p>= 0 or &gt; 7: Deactivates "Automatic Laser Control" (for Ctrl &gt; 7: <a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</p>	
<b>Value</b>	<p>Defines the 100% value for the parameter selected by Ctrl. As an unsigned 32-bit value. Allowed values:</p> <ul style="list-style-type: none"> <li>For Ctrl = 1/2: 12-bit values [0...4095]. Higher bits are ignored.</li> <li>For Ctrl = 3: 8-bit values [0...255]. Higher bits are ignored.</li> <li>For Ctrl = 4: [0...(2<sup>32</sup>-1)]. <i>1 bit equals 1/64 µs.</i></li> <li>For Ctrl = 5: [0...(2<sup>32</sup>-1)]. <i>1 bit equals 1/64 µs.</i></li> <li>For Ctrl = 6: 16-bit values [0...(2<sup>16</sup>-1)]. Higher bits are ignored.</li> </ul>	
<b>Mode</b>	<p>Speed-dependent laser control (Mode = 1...5) lets you select which data is to be used for calculating speed-dependent correction. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>=1: Set speed.</li> <li>=2: Actual speed (as well as extensions, see below).</li> <li>=3: Reserved.</li> <li>=4: Reserved.</li> <li>=5: Encoder speed (counter pulse rate).</li> </ul>	

Ctrl Command	set_auto_laser_control															
Parameters (cont'd)	Mode (cont'd)	<p>For <code>Mode</code> = 2 above, the following extensions are available in any combination:</p> <ul style="list-style-type: none"> <li>• <math>M = +4</math> encoder speed-dependent correction</li> <li>• <math>M = +16</math> for SCANAhead scan systems (instead of intelliSCAN)</li> <li>• <math>M = +32</math> The angle-synchronous galvanometer scanner speed is converted into an image field synchronous speed with the help of the assigned correction table.</li> </ul> <p>Then the following applies: <math>Mode = 2 + \text{sum of extensions } M</math>.</p> <p>For <code>Mode</code> = 0 or none of the above described ones:</p> <p>Disables the speed-dependent oder encoder-speed-dependent laser control (<a href="#">get_last_error</a>-Returncode: <code>RTC6_PARAM_ERROR</code>). The position-dependent laser control always remains effective with a valid value for <code>Ctrl</code>. See also <a href="#">Section "Position-Dependent Laser Control", page 188</a>.</p>														
	MinValue	<p>Defines the <i>lower</i> range limit of the parameter selected by <code>Ctrl</code> for automatic correction and that cannot be undercut.</p> <p>As an unsigned 32-bit value. Allowed values: see <code>Value</code>.</p>														
	MaxValue	<p>Defines the <i>upper</i> range limit of the parameter selected by <code>Ctrl</code> for automatic correction and that cannot be exceeded.</p> <p>As an unsigned 32-bit value. Allowed values: see <code>Value</code>.</p>														
Result	<p><code>ErrorCode</code>. As an unsigned 32-bit value.</p> <table> <tr> <td>0</td><td>No error.</td></tr> <tr> <td>1</td><td>No first scan head active.</td></tr> <tr> <td>2</td><td>No iDRIVE scan system (see glossary entry on <a href="#">page 879</a>) active.</td></tr> <tr> <td>3</td><td>Invalid <code>Ctrl</code> value.</td></tr> <tr> <td>4</td><td>Invalid <code>Mode</code> value.</td></tr> <tr> <td>5</td><td>Access denied (board locked, <a href="#">get_last_error</a> return code <code>RTC6_ACCESS_DENIED</code>).</td></tr> <tr> <td>6</td><td>Mode extension <math>M = +16</math>: No SCANAhead system connected or not active.</td></tr> </table>		0	No error.	1	No first scan head active.	2	No iDRIVE scan system (see glossary entry on <a href="#">page 879</a> ) active.	3	Invalid <code>Ctrl</code> value.	4	Invalid <code>Mode</code> value.	5	Access denied (board locked, <a href="#">get_last_error</a> return code <code>RTC6_ACCESS_DENIED</code> ).	6	Mode extension $M = +16$ : No SCANAhead system connected or not active.
0	No error.															
1	No first scan head active.															
2	No iDRIVE scan system (see glossary entry on <a href="#">page 879</a> ) active.															
3	Invalid <code>Ctrl</code> value.															
4	Invalid <code>Mode</code> value.															
5	Access denied (board locked, <a href="#">get_last_error</a> return code <code>RTC6_ACCESS_DENIED</code> ).															
6	Mode extension $M = +16$ : No SCANAhead system connected or not active.															
Comments	<ul style="list-style-type: none"> <li>• For usage of <code>set_auto_laser_control</code>, see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a>, <a href="#">page 186</a>.</li> <li>• <code>MinValue</code> and <code>MaxValue</code> are automatically exchanged if <code>MinValue &gt; MaxValue</code>; if necessary, <code>MinValue</code> is clipped to <math>\leq Value</math> and <code>MaxValue</code> clipped to <math>\geq Value</math>.</li> <li>• The control data for position- and/or speed-dependent laser control is exclusively derived from the scan system data at the first scan head connector and then also applied for the second scan head connector (if that connector has been activated and a scan system is attached). At the time the command is called, the first scan head connector must already have been assigned a correction table, otherwise error code 1 is returned and <code>Ctrl</code> is set to 0. If only the second scan head connector is being used and/or the scan system at the first scan head connector is shut off, then speed-dependent laser control does not function.</li> </ul>															



Ctrl Command	set_auto_laser_control
Comments (cont'd)	<ul style="list-style-type: none"> <li>For Mode = 2, a functioning iDRIVE scan system must be attached to the first scan head connector. As a result, <b>control_command</b> is unavailable for the first scan head connector as long as this mode selection is in effect. For Mode = 0 or 1, <b>control_command</b> is available without any restriction. If no functioning iDRIVE scan system is attached, then error code 2 is returned and Ctrl set to 0.</li> <li>Encoder-speed-dependent laser control (Mode = 5) functions even if no scan heads are attached to the scan-head connectors at runtime. The <b>Option Processing-on-the-fly</b> does not need to be activated here either.</li> <li>With the +4 extension, encoder speeds are converted to galvanometer scanner units (bits/ms) by using the scaling factors from <b>set_fly_x</b> and <b>set_fly_y</b> or <b>set_fly_2d</b>. The result is then added to the current galvanometer scanner speeds. This requires an enabled <b>Option Processing-on-the-fly</b> (in contrast to Mode = 5). The current mark speed is used as reference speed. The <b>set_encoder_speed</b> command (which is used for Mode = 5) is not taken into account with the +4 extension. If an axis is not activated for Processing-on-the-fly at runtime, its encoder speed is not taken into account. If both axes are not activated for Processing-on-the-fly, +4 extension is not effective. <b>set_fly_rot</b>, <b>set_fly_rot_pos</b>, <b>set_fly_x_pos</b> and <b>set_fly_y_pos</b> cannot be combined with the galvanometer scanner speed.</li> <li>For Ctrl = 1...6, the selected signal parameter is updated every 10 µs. For Ctrl = 5 (output period) it is always only effective after an already started laser signal period has elapsed. For Ctrl = 7, however, the output period is continuously adjusted (from pulse to pulse).</li> <li>Ctrl = 7 ("Spot Distance Control") is only available for excelliSCAN scan heads. For Ctrl = 7 to actually take effect the desired (geometric) pulse distance must be specified. As long as the pulse distance is 0 (default value) this control is ineffective, but not deactivated. The actual controlled variable ("HalfPeriod") cannot be recorded with <b>set_trigger/set_trigger4</b> and signal 24. The command parameters MinValue, MaxValue and Value are ignored. Therefore, the actual controlled variable ("HalfPeriod") cannot be limited by MinValue and MaxValue. It is automatically controlled to the current mark speed. See also <b>Section ""Spot Distance Control""</b>, page 192.</li> <li>Activating the angle-to-image field transformation of the galvanometer scanner speed (+32 extension) is particularly useful, if the correction file has to compensate for strong distortions.</li> <li>If the values for Ctrl or Mode are invalid, then <b>set_auto_laser_control</b> is not executed (return value 3 or 4, <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>"Automatic Laser Control" should not be combined with the variable laser power of <b>set_multi_mcbsp_in</b>.</li> <li>Only excelliSCAN scan heads: If an excelliSCAN scan head is connected and active, make sure that it is no longer busy before calling <b>set_auto_laser_control</b> (see <b>get_status</b> and state <b>HEAD_BUSY</b>). Otherwise, the not yet processed rest of the marking could be compromised.</li> </ul>



Ctrl Command	set_auto_laser_control
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> for Ctrl = 1/2, the specified values for Value, MinValue and MaxValue are internally multiplied by 4. For Ctrl = 4/5, the parameter values for Value, MinValue and MaxValue are multiplied by 8. The allowed value ranges decrease accordingly.
RTC5→RTC6	Changed functionality.  More laser control modes can be set.
Version info	Available as of version DLL 600, OUT 600, RBF 600.  Latest changed version DLL 609, OUT 609, RBF 613: Ctrl = 7.
References	<a href="#">load_position_control</a> , <a href="#">load_auto_laser_control</a> , <a href="#">set_auto_laser_params</a> , <a href="#">set_auto_laser_params_list</a> , <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">set_fly_2d</a> , <a href="#">set_encoder_speed</a> , <a href="#">spot_distance</a> , <a href="#">spot_distance_ctrl</a>



<b>Ctrl Command</b>	<b>set_auto_laser_params</b>		
<b>Function</b>	Specifies the to-be-controlled signal parameter of the "Automatic Laser Control", the 100% value and its corresponding limit values.		
<b>Call</b>	set_auto_laser_params( Ctrl, Value, MinValue, MaxValue )		
<b>Parameters</b>	Ctrl	The to-be-controlled signal parameter (see <a href="#">set_auto_laser_control</a> ). Allowed value range: [1...7].	
	Value	100% value. See <a href="#">set_auto_laser_control</a> .	
	MinValue	Lower range limit. See <a href="#">set_auto_laser_control</a> .	
	MaxValue	Upper range limit. See <a href="#">set_auto_laser_control</a> .	
<b>Comments</b>	<ul style="list-style-type: none"> <li>For information on using <a href="#">set_auto_laser_params</a>, see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a>, page 186.</li> <li>The meaning of the parameters is identical to that of <a href="#">set_auto_laser_control</a> (see also comments there). However, <a href="#">set_auto_laser_params</a> can neither start nor terminate "Automatic Laser Control".</li> <li>If the value for Ctrl is invalid (0 or &gt; 7), then <a href="#">set_auto_laser_params</a> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</li> </ul>		
RTC4→RTC6	New command.  <a href="#">RTC4 Compatibility Mode</a> : see <a href="#">set_auto_laser_control</a> .		
RTC5→RTC6	Unchanged functionality.		
Version info	Available as of version DLL 600, OUT 600, RBF 600.  Latest changed version DLL 609, OUT 609, RBF 613: Ctrl = 7.		
References	<a href="#">set_auto_laser_control</a> , <a href="#">set_auto_laser_params_list</a>		

<b>Delayed Short List Command</b>	<b>set_auto_laser_params_list</b>				
<b>Function</b>	As <a href="#">set_auto_laser_params</a> , but a delayed short list command.				
<b>Call</b>	set_auto_laser_params_list( Ctrl, Value, MinValue, MaxValue )				
<b>Parameters</b>	Ctrl	See <a href="#">set_auto_laser_params</a> .			
	Value	See <a href="#">set_auto_laser_params</a> .			
	MinValue	See <a href="#">set_auto_laser_params</a> .			
	MaxValue	See <a href="#">set_auto_laser_params</a> .			
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the value for Ctrl is invalid (0 or &gt; 7), then <a href="#">set_auto_laser_params_list</a> is replaced with a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</li> </ul>				
RTC4→RTC6	New command.  <a href="#">RTC4 Compatibility Mode</a> : see <a href="#">set_auto_laser_control</a> .				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.  Latest changed version DLL 609, OUT 609, RBF 613: Ctrl = 7.				
References	<a href="#">set_auto_laser_params</a> , <a href="#">set_auto_laser_control</a>				



<b>Ctrl Command</b>	<b>set_char_pointer</b>
<b>Function</b>	Stores the absolute start address of a command list in the internal management table for indexed characters.
<b>Call</b>	set_char_pointer( Char, Pos )
<b>Parameters</b>	Char      Index of the indexed character whose starting address <code>Pos</code> should be entered in the management table. As an unsigned 32-bit value. Allowed value range: [0...1023]. The same applies as for <b>load_char</b> : Char = character set number * 256 + ASCII number of the character (character sets are numbered 0 to 3).
	Pos      Absolute start address. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>23</sup> -1)].
<b>Comments</b>	<ul style="list-style-type: none"> <li>If <code>Char &gt; 1023</code> and/or <code>Pos &gt; (2<sup>23</sup>-1)</code>, then <b>set_char_pointer</b> is <i>not</i> executed (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>The <b>set_char_pointer</b> command can be used for referencing an indexed character. It thereby becomes an indexed character that is protectable by <b>save_disk/load_disk</b> and/or callable by the index.</li> <li><b>set_char_pointer</b> can also be used to reference anew an indexed subroutine, character or text string so that it can also be called by a second index. Here, it is preferable to use the <b>copy_dst_src</b> command for index management.</li> <li>The start addresses of command lists that are to be referenced with <b>set_char_pointer</b> can be queried by <b>get_input_pointer</b> before loading the command lists.</li> <li><b>set_char_pointer</b> only stores <i>starting addresses</i> in the internal management table. An indexed character only gains protection by a subsequent <b>save_disk/load_disk</b> command.</li> <li><code>Pos</code> should not be an arbitrary address within a list. Instead, it should be the starting address of an actually existing subroutine that was finalized by <b>list_return</b> and does not contain <b>set_end_of_list</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>load_char</b>



<b>Ctrl Command</b>	<b>set_char_table</b>
<b>Function</b>	Stores the absolute start address of a command list in the internal management table for indexed text strings.
<b>Call</b>	set_char_table( Index, Pos )
<b>Parameters</b>	<p>Index      Index of the indexed text string whose starting address <code>Pos</code> should be entered in the management table. As an unsigned 32-bit value. Allowed value range: [0...41]). The same ordering applies as for the <a href="#">load_text_table</a> command:</p> <ul style="list-style-type: none"> <li>= 0...9:      Digits for marking the time and date [0...9].</li> <li>= 10...21:      Months [January...December].</li> <li>= 22...28:      Days-of-the-week [Sunday...Saturday].</li> <li>= 29:      Blank character for marking serial numbers.</li> <li>= 30...39:      Digits for marking serial numbers [0...9].</li> <li>= 40:      Text for "a.m.". </li> <li>= 41:      Text for "p.m.". </li> </ul> <p>Pos      Absolute start address. As an unsigned 32-bit value. Allowed value range: [0...(2<sup>23</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>set_char_table</code> is synonymous with <a href="#">set_text_table_pointer</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality. <code>set_char_table</code> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_text_table_pointer</a>



Ctrl Command	set_control_mode		
Function	Enables or disables the external control input and resets the counter for external starts to zero.		
Call	set_control_mode( Mode )		
Parameters	Mode	As an unsigned 32-bit value.	
	Bit #	Value	Description
	Bit #0 (LSB)	= 1:	The external start input (by /START, /START2 or /Slave-START) is enabled.
		= 0:	The external start input is disabled.
	Bit #1	= 1:	An external stop (/STOP, /STOP2, /Slave-STOP or <b>simulate_ext_stop</b> ) causes explicit cancellation of the external start queue's entries (/START, /START2, /Slave-START or <b>simulate_ext_start</b> ).
		= 0:	No effect.
	Bit #2	= 1:	The track delay (defined by <b>simulate_ext_start</b> , <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b> ) that postpones execution of the start relative to the triggering input signal or <b>simulate_ext_start</b> or <b>simulate_ext_start_ctrl</b> command (see Section "External Start", page 276) is deactivated.
		= 0:	No effect. To define and activate the track delay (for example, for Processing-on-the-fly applications), use <b>simulate_ext_start</b> , <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b> .
	Bit #3	= 1:	The external start input is <i>not</i> disabled by an external stop request.
		= 0:	The external start input <i>is</i> disabled by an external stop request.
	Bit #4		Disables <b>simulate_ext_start_ctrl</b> .
	Bit #5		Reserved.
	Bit #6		Reserved.
	Bit #7		Reserved.
	Bit #8		Reserved.
	Bit #9	= 1:	Encoder resets of the two internal encoder counters occur with an external start signal or <b>simulate_ext_start</b> or <b>simulate_ext_start_ctrl</b> , postponed by a track delay defined by <b>simulate_ext_start</b> , <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b> , see also Bit #2).
		= 0:	Encoder resets occur immediately with each initiating Processing-on-the-fly command.



Ctrl Command	set_control_mode	
Parameters (cont'd)	Bits #10 = 1: Bits #10 = 0: Bit #12 ... Bit #31	<p>Track delay configured by <b>simulate_ext_start</b>, <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b> is counted beginning with the most recent externally (but not with <b>execute_list_pos</b> etc.) triggered or simulated external start. The interval between subsequent external starts (in encoder pulses) is thus constant, see also <b>Section "Regular (Periodic) External Starts", page 279</b>. For <b>stop_execution</b> or an external stop signal, Bit #10 gets reset to "0".</p> <p>Track delay configured by <b>simulate_ext_start</b>, <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b> is counted beginning with the time point an external start was requested (that is, with the corresponding <b>simulate_ext_start</b> or <b>simulate_ext_start_ctrl</b> command or external start signal). The interval between subsequent external starts (in encoder pulses) can thus vary.</p> <p>Reserved.</p> <p>...</p> <p>Reserved.</p>
Comments	<ul style="list-style-type: none"> <li>If execution is aborted by <b>stop_execution</b>, then <b>Bit #0</b> and <b>Bits #10</b> gets reset to zero, thus deactivating external start inputs and the counting of track delays with respect to a the triggering event.</li> <li>If <b>Bit #9</b> = 0, then there is generally a small (random) time offset (10 µs jitter) between the start signal at /START, /START2 and the actual start. If <b>Bit #9</b> = 1, then this 10 µs jitter is not present because the encoder reset then occurs synchronously with the start signal.</li> <li><b>Bit #9</b> = 0 is useful, when several separate <b>Processing-on-the-fly sessions</b> are to be started within a list.</li> <li>See also <b>Section "External Stop", page 275</b> and <b>Section "External Start", page 276</b>.</li> </ul>	
RTC4→RTC6	<p><b>Bit #8</b> cannot not be used to lock or unlock the upper 8 bits of the 16-bit digital output port for I/O commands. The RTC6 automatically reserves these bits for varioSCAN FLEX control by <b>move_to</b>. It is not possible to explicitly release these bits (release automatically occurs by <b>load_program_file</b>).</p> <p>Otherwise: unchanged functionality for the bits that are also used on the RTC4.</p>	
RTC5→RTC6	Unchanged functionality.	
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 605, OUT 605: Bit #4.	
References	<a href="#">set_control_mode_list</a> , <a href="#">set_extstartpos</a> , <a href="#">get_counts</a> , <a href="#">set_max_counts</a> , <a href="#">get_startstop_info</a> , <a href="#">move_to</a>	



<b>Normal List Command</b>	<b>set_control_mode_list</b>
<b>Function</b>	Like <b>set_control_mode</b> , but a list command.
<b>Call</b>	<code>set_control_mode_list( Mode )</code>
<b>Parameters</b>	Mode As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The counter for external starts is <i>not</i> reset by <b>set_control_mode_list</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality for the bits that are also used on the RTC4.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 605, OUT 605: Bit #4.
<b>References</b>	<b>set_control_mode</b>

<b>Ctrl Command</b>	<b>set_default_pixel</b>
<b>Function</b>	Defines the pulse length default value for the default pixel that terminates pixel output mode.
<b>Call</b>	<code>set_default_pixel( PulseLength )</code>
<b>Parameters</b>	PulseLength Default pixel pulse length. As an unsigned 32-bit value. <i>1 bit equals 1/64 μs.</i> Allowed value range: [0...(2 <sup>32</sup> –1)].
<b>Comments</b>	<ul style="list-style-type: none"> <li>In pixel output mode, the pixel pulse length at the end of an image line (at the beginning of the default pixel) gets set to the specified default value, see <a href="#">Chapter 8.7.4 "Synchronization", page 254</a>.</li> <li>When initializing (by <b>load_program_file</b>), the PulseLength default value is set to 0.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>set_port_default</b>



<b>Undelayed Short List Command</b>	<b>set_default_pixel_list</b>
Function	Like <b>set_default_pixel</b> , but a list command.
Call	<code>set_default_pixel_list( PulseLength )</code>
Parameters	PulseLength     Default pixel pulse length. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...(2 <sup>32</sup> -1)].
Comments	<ul style="list-style-type: none"><li>• See <b>set_default_pixel</b>.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_default_pixel, set_port_default</b>



<b>Ctrl Command</b>	<b>set_defocus</b>
<b>Function</b>	Determines a focus shift for 3D outputs.
<b>Restriction</b>	If the <b>Option "3D"</b> has not been enabled or if no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>set_defocus</b> has no effect. Nevertheless, the supplied focus shift value is stored internally and takes effect as soon as a 3D correction table is assigned.
<b>Call</b>	<code>set_defocus( Shift )</code>
<b>Parameters</b>	Shift      Focus shift in <i>bits</i> (as a signed 32-bit value). Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.
<b>Comments</b>	<ul style="list-style-type: none"> <li>A focus shift causes a defocusing of the laser focus relative to the working plane. A positive value increases the focal length of the <b>Dynamic focusing unit</b> and shifts the focus position, for example, for the control value <code>(0 0 0)</code> by about <math>d = \text{Shift} / K</math> to the plane <math>z = -d</math> [mm]. For the calibration factor <math>K</math>, see <a href="#">Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156</a>; furthermore, 8 in <a href="#">Chapter 7.3.6 "Output Values to the Scan System", page 170</a>.</li> <li>If the resulting total Z output value is too large, the z axis moves to the limit stop. Make sure that this is avoided as far as possible.</li> <li>If the command is called during output of a vector, then it is only executed directly before the next list command. To avoid "<b>Hard jumps</b>", a jump to the changed z output is performed at jump speed. This might take a few clock cycles, depending on the jump speed setting and desired focus shift. If no list is currently <b>BUSY</b>, then the jump executes immediately, whereby no delay occurs at the next list start.</li> <li>If the <b>INTERNAL-BUSY</b> status of the board is set, <b>set_defocus</b> is only executed with a delay (after <b>INTERNAL-BUSY</b> has been reset again).</li> <li><b>set_defocus</b> sets the <b>INTERNAL-BUSY</b> status while the jump to the changed z output is executed.</li> <li>After "<b>vector-controlled laser control</b>" is activated by <b>set_vector_control( Ctrl = 7 )</b>, the focus shift changes with parameterized mark or jump commands, too.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. But the RTC6 command avoids " <b>Hard jumps</b> ".
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_defocus_list</a> , <a href="#">set_offset_xyz</a> , <a href="#">set_defocus_offset</a>



<b>Variable List Command</b>	<b>set_defocus_list</b>
Function	Like <a href="#">set_defocus</a> , but a list command.
Restriction	See <a href="#">set_defocus</a> .
Call	set_defocus_list( Shift )
Parameters	Shift      See <a href="#">set_defocus</a> .
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">set_defocus</a>.</li> <li>• Even though <b>set_defocus_list</b> is a short list command, execution of the directly following list command is delayed by a few clock cycles due to the intermediate jump to the changed z output. The extent of this delay depends on the size of the specified focus shift; but it is at least 10 µs, even if Shift = 0.</li> </ul>
RTC4→RTC6	See <a href="#">set_defocus</a> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_defocus</a>

<b>Ctrl Command</b>	<b>set_defocus_offset</b>
Function	Defines an offset in addition to the focus shift (see <a href="#">set_defocus</a> ) for all 3D outputs.
Restriction	Like <a href="#">set_defocus</a> .
Call	set_defocus_offset( Shift )
Parameters	Shift      See <a href="#">set_defocus</a> .
Comments	<ul style="list-style-type: none"> <li>• <b>set_defocus_offset</b> has the same effect as <a href="#">set_defocus</a>.</li> <li>• <b>set_defocus_offset</b> enables setting a global defocus shift that is not overwritten by parameterized commands such as <a href="#">para_mark_abs</a>, see <a href="#">set_vector_control</a>(Ctrl = 7).</li> <li>• The <b>set_defocus_offset</b> shift, <ul style="list-style-type: none"> <li>– works additively to <a href="#">set_defocus</a>,</li> <li>– cannot be used as a control parameter for “vector-controlled laser control”,</li> <li>– cannot be recorded via Signal 32 with <a href="#">set_trigger</a>/<a href="#">set_trigger4</a>,</li> <li>– is taken into account by <a href="#">get_z_distance</a>, <a href="#">get_galvo_controls</a> and the back transformation (<a href="#">transform</a>, <a href="#">get_transform</a>).</li> </ul> </li> </ul>
RTC4→RTC6	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified value for Shift by 16.
RTC5→RTC6	Unchanged functionality. In <a href="#">RTC5 Compatibility Mode</a> , the RTC6 multiplies the specified value for Shift by 16.
Version info	Available as of version DLL 609, OUT 609, RBF 614.
References	<a href="#">set_defocus</a> , <a href="#">set_defocus_offset_list</a>



<b>Variable List Command</b>	<b>set_defocus_offset_list</b>
<b>Function</b>	Like <a href="#">set_defocus_offset</a> , but a list command.
<b>Restriction</b>	See <a href="#">set_defocus</a> .
<b>Call</b>	set_defocus_offset_list( Shift )
<b>Parameters</b>	Shift      See <a href="#">set_defocus</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">set_defocus_offset</a>.</li> <li>• See <a href="#">set_defocus_list</a>.</li> <li>• Even though <b>set_defocus_offset_list</b> is a short list command, execution of the directly following list command is delayed by a few clock cycles due to the intermediate jump to the changed z position. The extent of this delay depends on the size of the specified focus shift; but it is at least 10 µs, even if Shift= 0.</li> </ul>
<b>RTC4→RTC6</b>	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified value for Shift by 16.
<b>RTC5→RTC6</b>	Unchanged functionality. In <a href="#">RTC5 Compatibility Mode</a> , the RTC6 multiplies the specified value for Shift by 16.
<b>Version info</b>	Available as of version DLL 609, OUT 609, RBF 614.
<b>References</b>	<a href="#">set_defocus_offset</a> , <a href="#">set_defocus_list</a>



Ctrl Command	set_delay_mode		
Function	Turns the “Variable polygon delay mode” and the “Variable jump delay mode” on or off and sets some special scanner-delay related parameters as well as the 3D Z-Move mode.		
Call	<code>set_delay_mode( VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit )</code>		
Parameters	Name	Allowed Values	Description
	VarPoly	> 0  = 0	Enables the variable polygon delay mode, see <a href="#">Section “Variable Polygon Delay”, page 140</a> .  Disables the variable polygon delay mode (default setting).
	DirectMove3D	> 0  = 0	As an unsigned 32-bit value.  This parameter effects only 3D-applications.  The Z value is changed directly (linearly) to its end value during a jump.  The Z value is changed to its end-value in such a way that the focus is kept in one plane during the entire jump.
	EdgeLevel	0...(2 <sup>32</sup> -1). <i>1 bit equals 10 µs.</i>	As an unsigned 32-bit value.  This parameter defines a maximum “laser on” time for the corners of a <a href="#">Polyline</a> . If the polygon delay is longer than or equal to this value (because the angle $\phi$ is close to 180°, for instance), the laser is turned off (after a <a href="#">LaserOff</a> delay) and a new <a href="#">Polyline</a> is started. This can be useful for preventing burn-in effects. The <a href="#">EdgeLevel</a> value must be smaller than twice the set value for the polygon delay, otherwise it has no effect. See also <a href="#">figure 36</a> .  Note: To disable this feature, the <a href="#">EdgeLevel</a> value must be set to (2 <sup>32</sup> -1) (default value).
	MinJumpDelay	0...(2 <sup>32</sup> -1). <i>1 bit equals 10 µs.</i>	As an unsigned 32-bit value.  Minimum jump delay that cannot be undercut for jumps shorter than <a href="#">JumpLengthLimit</a> (even for jump vectors of zero length, see <a href="#">figure 32</a> ). For jumps longer than <a href="#">JumpLengthLimit</a> , the <a href="#">MinJumpDelay</a> has no relevance. To avoid anomalies in the range of <a href="#">MinJumpDelay</a> , define a value for <a href="#">MinJumpDelay</a> that is not larger than the jump delay.
			As an unsigned 32-bit value.



Ctrl Command	set_delay_mode
	<p>JumpLengthLimit 0...(2<sup>32</sup>-1)</p> <p>Jump length limit. In bits. If the jump vector is <i>longer</i> than this value, then the fixed jump delay (see <a href="#">set_scanner_delays</a>) is inserted. For all shorter jump lengths, a linearly interpolated Variable Jump Delay between MinJumpDelay and the jump delay is calculated and inserted, see <a href="#">figure 32</a>.</p> <p>JumpLengthLimit = 0 disables the Variable Jump Delay mode.</p> <p>As an unsigned 32-bit value.</p>
RTC4→RTC6	<p>Unchanged functionality. In addition: extended value range.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC6 multiplies the specified value for JumpLengthLimit by 16. The allowed value range decreases accordingly.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_scanner_delays</a> , <a href="#">load_varpolydelay</a> , <a href="#">set_delay_mode_list</a>

Multiple List Command	set_delay_mode_list
Function	Like <a href="#">set_delay_mode</a> , but a list command.
Call	set_delay_mode_list( VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit )
Parameters	<p>VarPoly      Like <a href="#">set_delay_mode</a>.</p> <p>DirectMove3D      Like <a href="#">set_delay_mode</a>.</p> <p>EdgeLevel      Like <a href="#">set_delay_mode</a>.</p> <p>MinJumpDelay      Like <a href="#">set_delay_mode</a>.</p> <p>JumpLengthLimit      Like <a href="#">set_delay_mode</a>.</p>
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">set_delay_mode</a>.</li> <li>• <a href="#">set_delay_mode_list</a> requires two list storage positions.</li> <li>• <a href="#">set_delay_mode_list</a> is executed as two undelayed short list commands.</li> <li>• Any still-pending delayed short list command is executed first.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p><a href="#">RTC4 Compatibility Mode</a>: see <a href="#">set_delay_mode</a>.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_delay_mode</a>



<b>Ctrl Command</b>	<b>set_dsp_mode</b>
<b>Function</b>	Sets a <b>DSP</b> mode for short-command count.
<b>Call</b>	<code>dsp_mode = set_dsp_mode( Mode )</code>
<b>Parameters</b>	<p>Mode      Desired <b>DSP</b> mode. As an unsigned 32-bit value.</p> <p>= 0: Corresponds to 500 MHz RTC5 boards (older type series).</p> <p>= 1: Reserved.</p> <p>= 2: Corresponds to 720 MHz RTC5 boards (newer type series).</p> <p>= 3: Corresponds to RTC6 boards. Default setting.</p>
<b>Result</b>	Set <b>DSP</b> mode prior to <b>set_dsp_mode</b> execution. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_dsp_mode</b> is used to switch the faster processing of short vectors in standard mode (<b>DSP</b> mode 3) to a RTC5-compatible processing which is slower, see <a href="#">Section "Automatic Delay Adjustments", page 144</a>.</li> <li>• <b>set_dsp_mode</b> is required, if: <ul style="list-style-type: none"> <li>– several RTC6 boards with different <b>DSP</b> modes are to be operated synchronously in a master/slave chain</li> <li>– AND if the adaptation of the short command count is not carried out by <a href="#">sync_slaves</a></li> </ul> </li> <li>• <b>DSP</b> mode &gt; 3 cannot be set. In this case, <code>Mode</code> is clipped to 3.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Basically unchanged functionality. However: Standard mode ( <b>DSP</b> mode 3) is available.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">sync_slaves</a> , <a href="#">get_RTC_version</a>



<b>Undelayed Short List Command</b>	<b>set_ellipse</b>
Function	Defines the shape of an elliptical arc that can subsequently be marked by <b>mark_ellipse_abs</b> or <b>mark_ellipse_rel</b> .
Call	<code>set_ellipse( a, b, Phi0, Phi )</code>
Parameters	<p>a      Length of the elliptical half-axis. In bits.             As an unsigned 32-bit value.             Allowed value range: [1...8,388,607].             Out-of-range positive values are clipped to the boundary values.</p>
	b      See a.
	<p>Phi0      Beginning phase angle in degrees (the arc starting point's position relative to the end point of half-axis a).             As a 64-bit IEEE floating point value.             A positive sign means "clockwise".             Phi0 gets normalized to the value range [0...&lt;360°].</p>
	<p>Phi      Arc angle in degrees (to-be-marked ellipse section).             As a 64-bit IEEE floating point value.             A positive sign means "clockwise".             Allowed value range: [-3,600.0°...+3,600.0°] (<math>\pm 10</math> full ellipses).             Out-of-range values are clipped to the boundary values.</p>
Comments	<ul style="list-style-type: none"> <li>Specify the to-be-marked elliptical arc's position and orientation in the 2D image field by <b>mark_ellipse_abs</b> or <b>mark_ellipse_rel</b>. For descriptions of the individual parameters, see <a href="#">Section "Ellipse Commands", page 127</a>.</li> <li>For <math>a &lt; 1</math> or/and <math>b &lt; 1</math>, <b>set_ellipse</b> is replaced with a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the length values specified for the elliptical half-axes by 16. The allowed value range decreases accordingly.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>mark_ellipse_abs, mark_ellipse_rel</b>



<b>Delayed Short List Command</b>	<b>set_encoder_speed</b>
<b>Function</b>	Defines the target encoder speed and further parameters for encoder-speed-dependent "Automatic Laser Control".
<b>Call</b>	<code>set_encoder_speed( EncoderNo, Speed, Smooth )</code>
<b>Parameters</b>	<p>EncoderNo      Number of the encoder counter to be used for speed measurement. As an unsigned 32-bit value. Allowed values: = 0:            Encoder counter "Encoder0". = 1:            Encoder counter "Encoder1". = 2 and 3: Vectorial encoder velocity: a vectorial velocity is calculated from both encoder speeds (by the pulse rates of both encoder counters) for use with encoder-speed-dependent automatic laser control in <code>Mode = 5</code>. Higher bits are ignored.</p> <p>Speed           Target encoder speed (counter pulse rate) in [counter pulses/ms]. As a 64-bit IEEE floating point value. Allowed value range: [100.0...16000.0]. For <code>Speed &gt; 16000.0</code>, <code>Speed</code> is clipped to 16000.0, for <code>0.0 &lt; Speed &lt; 100.0</code> to 100.0.</p> <p>Smooth          Smoothing factor for a 2-stage low-pass filter. As a 64-bit IEEE floating point value. Allowed value range: [0.0...1.0]. Larger values are clipped.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If <code>Smooth = 0.0</code>, then only the current encoder speed <code>CurrentSpeed</code> (based on counter pulses received in the most recent 10 µs) is used; if <code>Smooth = 1.0</code>, then the speed from the previous clock cycle <code>PreviousSpeed</code>. In general, the used speed is: <math display="block">\text{Speed} = \text{PreviousSpeed} \times \text{Smooth} + \text{CurrentSpeed} \times (1.0 - \text{Smooth})</math>.</li> <li>If <code>Speed ≤ 0.0</code> or <code>Smooth &lt; 0.0</code>, then <code>set_encoder_speed</code> is, already during loading, replaced by a <code>list_nop</code> (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>One encoder increment results in four counter pulses.</li> <li>The maximum value for <code>Speed</code> (16000.0) corresponds to a counting rate of 16 MHz. The minimum value for <code>Speed</code> (100.0) corresponds to a counting rate of 1/(10 µs), that is, one counter pulse per output period. Beware of the low resolution of encoder-speed-dependent laser control for low speed values! Encoder-speed-dependent correction is only recommended if substantially more than one counter pulse per output period (10 µs) are received.</li> <li>See also <a href="#">Section "Encoder-Speed-Dependent Laser Control", page 192</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_encoder_speed_ctrl</a> , <a href="#">set_auto_laser_control</a>



<b>Ctrl Command</b>	<code>set_encoder_speed_ctrl</code>
<b>Function</b>	Like <code>set_encoder_speed</code> , but a control command.
<b>Call</b>	<code>set_encoder_speed_ctrl( EncoderNo, Speed, Smooth )</code>
<b>Parameters</b>	EncoderNo      Like <code>set_encoder_speed</code> .
	Speed            Like <code>set_encoder_speed</code> .
	Smooth          Like <code>set_encoder_speed</code> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>set_encoder_speed_ctrl</code> is not executed (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>), if:           <ul style="list-style-type: none"> <li>– Speed ≤ 0.0</li> <li>– Smooth &lt; 0.0</li> </ul> </li> <li>• <code>set_encoder_speed_ctrl</code> is not executed (<code>get_last_error</code> return code <code>RTC6_BUSY</code>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <code>pause_list</code>)</li> </ul> </li> <li>• <code>set_encoder_speed_ctrl</code> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <code>set_wait</code> (<b>PAUSED</b> status set)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>set_encoder_speed</code> , <code>set_auto_laser_control</code>



<b>Normal List Command</b>	<b>set_end_of_list</b>
<b>Function</b>	Ends execution of a list.
<b>Call</b>	<code>set_end_of_list()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If, during processing of a list, the <code>set_end_of_list</code> is encountered and no automatic list change has been previously activated, see <a href="#">Chapter 6.4.6 "Changing Lists Automatically", page 100</a>, then list execution ends. The signals for "laser active" operation are then switched off and a home jump, if defined by <code>home_position</code> or <code>home_position_xyz</code>, is executed (the <code>INTERNAL-BUSY</code> status is set while the home jump is executed).</li> <li>In contrast, upon reaching a <code>set_end_of_list</code>, execution continues at the other list if an automatic list change was previously activated. The other list can also be "List 1" if "List 2" was not configured (<code>Mem2 = 0</code>, see <a href="#">config_list</a>).</li> <li>Upon processing of the <code>set_end_of_list</code>, the USED status of the respective list (USED1 or USED2) is always set and the list's BUSY status (BUSY1 or BUSY2) reset (see also <a href="#">Chapter 6.4.2 "List Status", page 97</a>). The BUSY list execution status, on the other hand, is only reset if no automatic list change was previously activated.</li> <li>An automatic list change of the <code>input</code> pointer never occurs during loading of <code>set_end_of_list</code> (in contrast to an automatic list change of the <code>output</code> pointer during execution of <code>set_end_of_list</code>, if previously activated by <a href="#">auto_change_pos</a> or <a href="#">start_loop</a>).</li> <li>Upon loading <code>set_end_of_list</code>, the READY status of the list (READY1 or READY2) is set and the LOAD status of the list (LOAD1 or LOAD2) is reset. Additionally, flushing of the buffered list input is triggered, see <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a>.</li> <li><code>set_end_of_list</code> is ignored during loading and execution, that is, replaced with a <code>list_nop</code> if an indexed subroutine is currently being loaded or executed (<a href="#">get_last_error</a> return code <code>RTC6_IGNORED</code>).</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: Additional USED status. This is reset when loading <code>set_end_of_list</code> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_start_list</a>



<b>Ctrl Command</b>	<b>set_eth_boot_control</b>
<b>Function</b>	<b>Standalone Functionality:</b> Activates or deactivates automatic booting.
<b>Prerequisite</b>	Minimum requirement: RTC6 Software Package V1.7.0 and RTC6BIOSETH_26.
<b>Call</b>	set_eth_boot_control( Ctrl )
<b>Parameters</b>	Ctrl = 0: Deactivates automatic booting. > 0: Activates automatic booting. As an unsigned 32-bit value.
<b>Result</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_eth_boot_control</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• During the execution of <b>set_eth_boot_control</b> the 10 µs clock period of the <b>DSP</b> is interrupted for several ms.</li> <li>• <b>set_eth_boot_control</b> is only allowed with RTC6 Ethernet Boards. Otherwise, a <b>get_last_error</b> return code <b>RTC6_TYPE_REJECTED</b> is generated.</li> <li>• See <b>Chapter 15.7 "Standalone Functionality"</b>, page 859.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 618, OUT 618, RBF 623.
References	<b>eth_boot_dcmsg</b>

<b>Ctrl Command</b>	<b>set_eth_boot_timeout</b>
<b>Function</b>	<b>Standalone Functionality:</b> Sets the waiting time for the automatic transition from <b>Boot Phase 1</b> to <b>Boot Phase 2</b> .
<b>Prerequisite</b>	Minimum requirement: RTC6 Software Package V1.7.0 and RTC6BIOSETH_26.
<b>Call</b>	eth_set_search_cards_timeout( TimeOut )
<b>Parameters</b>	TimeOut Waiting time. In s. As an unsigned 32-bit value.
<b>Result</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_eth_boot_timeout</b> is only allowed with RTC6 Ethernet Boards. Otherwise, a <b>get_last_error</b> return code <b>RTC6_TYPE_REJECTED</b> is generated.</li> <li>• TimeOut = 0 deactivates the waiting time. The boot process waits endlessly for an /START.</li> <li>• <b>Boot Phase 2</b> starts immediately when an /START is triggered within the waiting time.</li> <li>• See <b>Chapter 15.7 "Standalone Functionality"</b>, page 859.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 618, OUT 618, RBF 623.
References	<b>eth_boot_dcmsg</b>



<b>Ctrl Command</b>	<b>set_extstartpos</b>
Function	Defines the start address (within the range of "List 1" or "List 2") where execution should continue upon external starts.
Call	set_extstartpos( Pos )
Parameters	Pos      Absolute address of the first list command to be executed. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>23</sup> -1)].
Comments	<ul style="list-style-type: none"> <li>By default, an external start results in a continuation or start at the beginning of "List 1" (Pos = 0).</li> <li>Pos must be within the range of "List 1" or "List 2". Otherwise, Pos = 0 is set.</li> <li>The specified start address is used for all external starts until a new address is specified by <b>set_extstartpos</b> or <b>set_extstartpos_list</b>. An address range validity check is performed on Pos before each external start; Pos might therefore become set to 0 at a later point (and remain at this value) if the configuration was correspondingly changed in the meantime (see <b>config_list</b>).</li> <li>See also <b>Section "External Start", page 276</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. <b>set_extstartpos</b> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_extstartpos_list, set_control_mode</b>

<b>Undelayed Short List Command</b>	<b>set_extstartpos_list</b>
Function	Like <b>set_extstartpos</b> , but a list command.
Call	set_extstartpos_list( Pos )
Parameters	Pos      Absolute address of the first list command to be executed. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>23</sup> 609 -1)].
Comments	<ul style="list-style-type: none"> <li><b>set_extstartpos_list</b> can be used within a list, to "link" it to the list that follows.</li> <li>See <b>set_extstartpos</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_extstartpos</b>



<b>Ctrl Command</b>	<b>set_ext_start_delay</b>
<b>Function</b>	Sets a track delay for external starts, so that the lists are started with a delay relative to the triggering input signal or <b>simulate_ext_start</b> or <b>simulate_ext_start_ctrl</b> command.
<b>Call</b>	<code>set_ext_start_delay( Delay, EncoderNo )</code>
<b>Parameters</b>	<p>Delay      Track delay (counter steps of the selected encoder counter <code>EncoderNo</code>) as a signed 32-bit value. Allowed value range: <math>[-2^{31} \dots + (2^{31}-1)]</math></p> <p>EncoderNo    Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values:</p> <ul style="list-style-type: none"> <li>= 0:Encoder counter "Encoder0".</li> <li>= 1:Encoder counter "Encoder1".</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For external starts, see <a href="#">Section "External Start", page 276</a>.</li> <li>• The track delay is specified in (relative) counting units of the selected encoder counter (the RTC6's encoder counter is triggered by an external or simulated encoder signal, see <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 284</a>).</li> <li>• Ensure that the sign of the track delay (<code>Delay</code> parameter) is appropriate for the selected encoder's counting direction (for external triggering, this corresponds to the workpiece's direction of motion).</li> <li>• For <code>Delay = 0</code>, the track delay is deactivated.</li> <li>• If a track delay is specified, that causes a start trigger (initiated by <b>simulate_ext_start</b> or an external start signal) to occur when the <b>BUSY</b> status or <b>INTERNAL-BUSY</b> status is set (for example, when outputting a list or during <b>goto_xy</b>), then no starts are get triggered by this start trigger (in this case, Bit #11 of the <b>get_startstop_info</b> return value is set).</li> <li>• Track delays can also be set with <b>simulate_ext_start</b>. Track delays are deactivated by initialization (with <b>load_program_file</b>), by external stops and by <b>stop_execution</b>. They can also be deactivated with <b>set_control_mode</b> (Bit #2).</li> <li>• <b>set_ext_start_delay</b> cancels already externally triggered starts that have not yet executed and are still being held in a queue that accommodates up to 8 starts.</li> <li>• If <code>EncoderNo &gt; 1</code>, then <b>set_ext_start_delay</b> is ignored (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> </ul>
RTC4→RTC6	<p>Unchanged functionality. In addition: increased value range.</p> <p>The RTC6 allows this command to be used not only together with a real encoder (hardware encoder), but also with an encoder simulation started by <b>simulate_encoder</b>.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified value for <code>Delay</code> by 16. The allowed value range decreases accordingly.</p>
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>simulate_ext_start</b> , <b>set_ext_start_delay_list</b> , <b>set_extstartpos</b> , <b>set_extstartpos_list</b> , <b>set_control_mode</b>



<b>Normal List Command</b>	<b>set_ext_start_delay_list</b>
<b>Function</b>	Like <b>set_ext_start_delay</b> , but a list command.
<b>Call</b>	<code>set_ext_start_delay_list( Delay, EncoderNo )</code>
<b>Parameters</b>	<p>Delay      See <b>set_ext_start_delay</b>.</p> <p>EncoderNo    See <b>set_ext_start_delay</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If <code>EncoderNo &gt; 1</code>, then <b>set_ext_start_delay_list</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> </ul>
RTC4→RTC6	See <b>set_ext_start_delay</b> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_ext_start_delay</b>

<b>Ctrl Command</b>	<b>set_firstpulse_killer</b>
<b>Function</b>	Sets the length of the FirstPulseKiller signal for a YAG laser.
<b>Call</b>	<code>set_firstpulse_killer( Length )</code>
<b>Parameters</b>	<p>Length      Length of the FirstPulseKiller signal. As an unsigned 32-bit value. <i>1 bit equals 1/64 μs.</i> Allowed value range: [0...+(2<sup>26</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Values over (2<sup>26</sup>-1) are clipped.</li> <li>The clock frequency connected to the FirstPulseKiller signal is always 64 MHz. <i>1 bit equals 1/64 μs.</i></li> <li>The signal level is set by <b>set_laser_control</b>.</li> <li>For YAG Mode 2, the Q-Switch delay is also correspondingly altered, see <b>Section "Differences Between the YAG Modes"</b>, page 179.</li> <li>In CO<sub>2</sub> Mode, <b>set_firstpulse_killer</b> has no effect.</li> <li>The laser mode is set by <b>set_laser_mode</b>, see <b>Chapter 7.4 "Laser Control"</b>, page 173.</li> <li>Q-Switch pulse length and period can be set by <b>set_laser_pulses_ctrl</b>, <b>set_laser_pulses</b> or <b>set_laser_timing</b>.</li> </ul>
RTC4→RTC6	Essentially similar functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified value by 8. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_laser_mode</b> , <b>set_laser_timing</b>



<b>Undelayed Short List Command</b>	<b>set_firstpulse_killer_list</b>
Function	Like <b>set_firstpulse_killer</b> , but a list command.
Call	<code>set_firstpulse_killer_list( Length )</code>
Parameters	Length      Length of the FirstPulseKiller signal. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...+(2 <sup>26</sup> -1)].
RTC4→RTC6	As <b>set_firstpulse_killer</b> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_firstpulse_killer</b>



<b>Normal List Command</b>	<b>set_fly_1_axis</b>																
<b>Function</b>	"Fly Extension" Command: Activates a 1 <b>Axis</b> -Processing-on-the-fly application.																
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>set_fly_1_axis</b> terminates the Processing-on-the-fly process (even though it could not have been activated).																
<b>Call</b>	<b>set_fly_1_axis( Axis, Mode, Scale )</b>																
<b>Parameters</b>	<p>Axis      <b>Axis from Table 3, page 245.</b>                  As an unsigned 32-bit value.</p> <p>Mode     <b>Mode from Table 4, page 247.</b>                  As an unsigned 32-bit value.</p> <p>Scale    Scaling factor.                  As a 64-bit IEEE floating point value.                  Allowed value range:</p> <ul style="list-style-type: none"> <li>• <math>1/256 \leq  \text{Scale}  \leq 16.000,0</math> with linear axis (1, 2 or 3)</li> <li>• <math> \text{Scale}  &gt; 100,0</math> with <b>Rotary axis</b> (4)</li> </ul> <p>Scale can be + or -. Only the absolute value is restricted.</p>																
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <b>Chapter 8.6 "Processing-on-the-fly", page 227</b> and <b>Section "Fly Extension" Commands</b>, page 245.</li> <li>• With <b>Mode( 1...4 )</b>, <b>set_fly_1_axis</b> requires two <math>10\ \mu\text{s}</math> clock cycles for execution.</li> <li>• <b>set_fly_1_axis</b> overwrites a previous definition for the same <b>Axis</b>. As a linear axis, <b>set_fly_1_axis</b> can be combined with other linear axes, but not with a <b>Rotary axis</b>. Any still-active rotation-fly application (<b>set_fly_1_axis( Axis =4 )</b>) is automatically deactivated and vice versa. It is recommended to explicitly switch off incompatible Processing-on-the-fly corrections beforehand, for example, see <b>fly_return_1_axis</b> and <b>fly_return_2_axes</b>.</li> <li>• With an unallowed parameter value, <b>set_fly_1_axis</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>• The following command calls are executed in the same way:       <table style="width: 100%; border-collapse: collapse;"> <tbody> <tr> <td style="width: 50%;">– <b>set_fly_1_axis( 1, 1, ScaleX )</b></td> <td>= <b>set_fly_x( ScaleX )</b></td> </tr> <tr> <td>– <b>set_fly_1_axis( 2, 2, ScaleY )</b></td> <td>= <b>set_fly_y( ScaleY )</b></td> </tr> <tr> <td>– <b>set_fly_1_axis( 3, EncoderNo+3, ScaleZ )</b></td> <td>= <b>set_fly_z( ScaleZ, EncoderNo )</b></td> </tr> <tr> <td>– <b>set_fly_1_axis( 4, 1, Resolution )</b></td> <td>= <b>set_fly_rot( Resolution )</b></td> </tr> <tr> <td>– <b>set_fly_1_axis( 1, 6, ScaleX )</b></td> <td>= <b>set_fly_x_pos( ScaleX )</b></td> </tr> <tr> <td>– <b>set_fly_1_axis( 2, 6, ScaleY )</b></td> <td>= <b>set_fly_y_pos( ScaleY )</b></td> </tr> <tr> <td>– <b>set_fly_1_axis( 4, 6, Resolution )</b></td> <td>= <b>set_fly_rot_pos( Resolution )</b></td> </tr> <tr> <td>– <b>set_fly_1_axis( 1, 7, Scale )</b></td> <td>corresponds to <b>set_mcbsp_in( 1, Scale )</b></td> </tr> </tbody> </table> </li> </ul>	– <b>set_fly_1_axis( 1, 1, ScaleX )</b>	= <b>set_fly_x( ScaleX )</b>	– <b>set_fly_1_axis( 2, 2, ScaleY )</b>	= <b>set_fly_y( ScaleY )</b>	– <b>set_fly_1_axis( 3, EncoderNo+3, ScaleZ )</b>	= <b>set_fly_z( ScaleZ, EncoderNo )</b>	– <b>set_fly_1_axis( 4, 1, Resolution )</b>	= <b>set_fly_rot( Resolution )</b>	– <b>set_fly_1_axis( 1, 6, ScaleX )</b>	= <b>set_fly_x_pos( ScaleX )</b>	– <b>set_fly_1_axis( 2, 6, ScaleY )</b>	= <b>set_fly_y_pos( ScaleY )</b>	– <b>set_fly_1_axis( 4, 6, Resolution )</b>	= <b>set_fly_rot_pos( Resolution )</b>	– <b>set_fly_1_axis( 1, 7, Scale )</b>	corresponds to <b>set_mcbsp_in( 1, Scale )</b>
– <b>set_fly_1_axis( 1, 1, ScaleX )</b>	= <b>set_fly_x( ScaleX )</b>																
– <b>set_fly_1_axis( 2, 2, ScaleY )</b>	= <b>set_fly_y( ScaleY )</b>																
– <b>set_fly_1_axis( 3, EncoderNo+3, ScaleZ )</b>	= <b>set_fly_z( ScaleZ, EncoderNo )</b>																
– <b>set_fly_1_axis( 4, 1, Resolution )</b>	= <b>set_fly_rot( Resolution )</b>																
– <b>set_fly_1_axis( 1, 6, ScaleX )</b>	= <b>set_fly_x_pos( ScaleX )</b>																
– <b>set_fly_1_axis( 2, 6, ScaleY )</b>	= <b>set_fly_y_pos( ScaleY )</b>																
– <b>set_fly_1_axis( 4, 6, Resolution )</b>	= <b>set_fly_rot_pos( Resolution )</b>																
– <b>set_fly_1_axis( 1, 7, Scale )</b>	corresponds to <b>set_mcbsp_in( 1, Scale )</b>																
RTC4→RTC6	New command.																
RTC5→RTC6	New command.																
Version info	Available as of version DLL 617, OUT 617, RBF 623.																
References	<b>set_fly_2_axes, set_fly_3_axes</b>																



<b>Normal List Command</b>	<b>set_fly_2_axes</b>
<b>Function</b>	"Fly Extension" Command: Activates a 2-Axes-Processing-on-the-fly application.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>set_fly_2_axes</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>set_fly_2_axes( Axis1, Mode1, Scale1, Axis2, Mode2, Scale2 )</code>
<b>Parameters</b>	<p>Axis1      <b>Axis from Table 3, page 245.</b>            Allowed values: 1...3 (linear axis only).            As an unsigned 32-bit value.</p> <p>Mode1      <b>Mode from Table 4, page 247.</b>            As an unsigned 32-bit value.</p> <p>Scale1      Scaling factor.            As a 64-bit IEEE floating point value.            Allowed value range:            • <math>1/256 \leq  Scale  \leq 16.000,0</math> with linear axis (1, 2 or 3)            Scale can be + or -. Only the absolute value is restricted.</p> <p>Axis2      Like Axis1.</p> <p>Mode2      Like Mode1.</p> <p>Scale2      Like Scale1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See Chapter 8.6 "Processing-on-the-fly", page 227 and Section ""Fly Extension" Commands", page 245.</li> <li>With <code>Mode( 1...4 )</code>, <b>set_fly_2_axes</b> requires two <math>10 \mu s</math> clock cycles for execution.</li> <li><b>set_fly_2_axes</b> overwrites a previous definition for the same <b>Axes</b> (even individually). As 2 linear axes, <b>set_fly_2_axes</b> can be combined with a third linear axis, but not with a <b>Rotary axis</b>. Any still-active rotation-fly application (<code>set_fly_1_axis( Axis =4 )</code>) is automatically deactivated and vice versa. It is recommended to explicitly switch off incompatible Processing-on-the-fly corrections beforehand, for example, see <b>fly_return_2_axes</b>.</li> <li>Axis1 and Axis2 need to be different linear axes and must not be rotary axes.</li> <li>Mode1 and Mode2 can be the same. If they use the same <b>Encoder</b>, a different scaling factor can be specified by <code>Mode1/Mode2 = 1 / 3 or 2 / 4</code>.</li> <li>With an unallowed parameter value, <b>set_fly_2_axes</b> is replaced by a <b>list_nop</b> (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> </ul>



<b>Normal List Command</b>	<b>set_fly_2_axes</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The following command calls are executed in the same way:           <ul style="list-style-type: none"> <li>- <code>set_fly_2_axes( 1, 1, ScaleX, 2, 2, ScaleY )</code> = <code>set_fly_2d( ScaleX, ScaleY )</code></li> <li>- <code>set_fly_2_axes( 1, 1, ScaleX, 2, 2, ScaleY )</code> =               <pre> {   set_fly_1_axis( 1, 1, ScaleX );   set_fly_1_axis( 2, 2, ScaleY ); }</pre> </li> </ul> </li> </ul> <p><b>Encoder</b> resets occur:</p> <ul style="list-style-type: none"> <li>With a single <code>set_fly_2_axes</code> call in the same 10 <math>\mu</math>s clock period</li> <li>With two <code>set_fly_1_axis</code> calls in two different 10 <math>\mu</math>s clock periods</li> </ul> <p>- <code>set_fly_2_axes( 1, 11, Scale, 2, 15, Scale )</code> corresponds to  <code>set_mcbsp_in_list( 3, Scale )</code></p>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">set_fly_1_axis</a> , <a href="#">set_fly_3_axes</a>

<b>Normal List Command</b>	<b>set_fly_2d</b>				
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of a linear workpiece-movement in 2 dimensions (based on the encoder values transferred to the RTC6 by encoder counters "Encoder0" and "Encoder1"). Sets the corresponding scaling factors.				
<b>Restriction</b>	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <b>set_fly_2d</b> terminates the Processing-on-the-fly process (even though it could not have been activated).				
<b>Call</b>	<code>set_fly_2d( ScaleX, ScaleY )</code>				
<b>Parameters</b>	<table> <tr> <td>ScaleX</td> <td>Scaling factor for the x direction (encoder counter "Encoder0"). In bits/count. As a 64-bit IEEE floating point value. Allowed value range: <math>1/256 \leq  \text{ScaleX}  \leq 16000.0</math>.</td> </tr> <tr> <td>ScaleY</td> <td>Scaling factor for the y direction (encoder counter "Encoder1"). In bits/count. As a 64-bit IEEE floating point value. Allowed value range: <math>1/256 \leq  \text{ScaleY}  \leq 16000.0</math>.</td> </tr> </table>	ScaleX	Scaling factor for the x direction (encoder counter "Encoder0"). In bits/count. As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleX}  \leq 16000.0$ .	ScaleY	Scaling factor for the y direction (encoder counter "Encoder1"). In bits/count. As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleY}  \leq 16000.0$ .
ScaleX	Scaling factor for the x direction (encoder counter "Encoder0"). In bits/count. As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleX}  \leq 16000.0$ .				
ScaleY	Scaling factor for the y direction (encoder counter "Encoder1"). In bits/count. As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleY}  \leq 16000.0$ .				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• ScaleX and ScaleY can be negative depending on the motion direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>• For Processing-on-the-fly correction (for example, determination of the scaling factor or deactivating Processing-on-the-fly correction), see the <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>. For <b>set_fly_2d</b> usage, see the <a href="#">Chapter 8.6.4 "Compensating 2D Motions", page 234</a>.</li> <li>• If unallowed parameter values are supplied (for example, for ScaleX = 0), then <b>set_fly_2d</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_2d</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "<b>set_fly_2d</b>" Processing-on-the-fly correction). However, Processing-on-the-fly correction successfully activated by <b>set_fly_2d</b> switches off any other Processing-on-the-fly correction and does itself get switched off by any other Processing-on-the-fly command, even if that other command contains unallowed parameters, see <a href="#">Section "Overview", page 227</a>.</li> <li>• If an encoder compensation has been set by <b>load_fly_2d_table</b> and <b>init_fly_2d</b>, the current encoder values are added to the latest reference values of the 2D encoder compensation and then the sums are saved as new reference values. The encoder counters are then reset to 0.</li> <li>• It should <i>not</i> be set by <b>set_control_mode( Bit #9 )</b> that the encoder counters are only reset after the subsequent external start trigger. Otherwise, the reference values of 2D encoder compensation are lost. See also <a href="#">Section "2D Encoder Compensation for xy Positioning Stages", page 234</a>.</li> <li>• Do not intermediately call <b>set_fly_x</b> or <b>set_fly_y</b> to switch on the Processing-on-the-fly application if you intend to use <b>set_fly_2d</b> in conjunction with 2D encoder compensation for an xy positioning stage, because here too the reference values are lost.</li> </ul>				



Normal List Command	<code>set_fly_2d</code>
Comments (cont'd)	<ul style="list-style-type: none"><li>If no correction table for 2D encoder compensation has yet been loaded onto the board (see <a href="#">load_fly_2d_table</a>), then the encoder values are used without correction.</li><li>You can also use <a href="#">activate_fly_2d/activate_fly_2d_encoder</a> to switch on <code>set_fly_2d</code> Processing-on-the-fly correction.</li></ul>
RTC4→RTC6	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified scaling factors by 16. The allowed value ranges change accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">init_fly_2d</a> , <a href="#">load_fly_2d_table</a> , <a href="#">get_fly_2d_offset</a> , <a href="#">activate_fly_2d</a> , <a href="#">activate_fly_xy</a>



<b>Multiple List Command</b>	<b>set_fly_3_axes</b>
<b>Function</b>	"Fly Extension" Command: Activates a 3-Axes-Processing-on-the-fly application.
<b>Restriction</b>	If the Option Processing-on-the-fly is not enabled, then <b>set_fly_3_axes</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>set_fly_3_axes( ModeX, ScaleX, ModeY, ScaleY, ModeZ, ScaleZ )</code>
<b>Parameters</b>	<p>ModeX      <b>Mode from Table 4, page 247.</b>                    As an unsigned 32-bit value.</p> <p>ScaleX     Scaling factor.                    As a 64-bit IEEE floating point value.                    Allowed value range:                    • <math>1/256 \leq  \text{Scale}  \leq 16.000,0</math> with linear axis (1, 2 or 3)                    Scale can be + or -. Only the absolute value is restricted.</p> <p>ModeY      Like ModeX.</p> <p>ScaleY     Like ScaleX.</p> <p>ModeZ      Like ModeX.</p> <p>ScaleZ     Like ScaleX.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See Chapter 8.6 "Processing-on-the-fly", page 227 and Section ""Fly Extension" Commands", page 245.</li> <li>With <code>Mode( 1...4 )</code>, <b>set_fly_3_axes</b> requires two <math>10\ \mu\text{s}</math> clock cycles for execution.</li> <li><b>set_fly_3_axes</b> requires two list memory positions. The first part is executed as an undelayed short list command prior to the second part, which executes as a normal list command. Any pending delayed short list commands execute first.</li> <li><b>set_fly_3_axes</b> overwrites a previous definition for the same <b>Axes</b> (even individually). Any still-active rotation-fly application (<code>set_fly_1_axis( Axis =4 )</code>) is automatically deactivated and vice versa. It is recommended to explicitly switch off incompatible Processing-on-the-fly corrections beforehand, for example, see <b>fly_return_3_axes</b>.</li> <li>The <b>Axes</b> are automatically set to 1, 2, 3 and therefore, do not need to be specified explicitly.</li> <li>ModeX, ModeY, ModeZ can be same.                    If they use the same <b>Encoder</b>, a different scaling factor can be specified by <code>ModeX/ModeY/ModeZ = 1 / 3 or 2 / 4</code>.</li> <li>With an unallowed parameter value, <b>set_fly_3_axes</b> is replaced by a <b>list_nop</b> (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> </ul>



<b>Multiple List Command</b>	<b>set_fly_3_axes</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The following command calls are executed in the same way:           <pre>- set_fly_3_axes( 1, ScaleX, 2, ScaleY, 3 or 4, ScaleZ ) =             {               set_fly_1_axis( 1, 1, Scale X );               set_fly_1_axis( 2, 2, Scale Y );               set_fly_1_axis( 3, 3 or 4, Scale Z );             }  - set_fly_3_axes( 1, ScaleX, 2, ScaleY, 3 or 4, ScaleZ ) =             {               set_fly_2_axes( 1, 1, Scale1 X, 2, 2, Scale2 X );               set_fly_1_axis( 3, 3 or 4, Scale Z );             }</pre> </li> </ul> <p><b>Encoder</b> resets occur:</p> <ul style="list-style-type: none"> <li>With a single <b>set_fly_3_axes</b> call in the same 10 µs clock cycle</li> <li>With three <b>set_fly_1_axis</b> calls in three different 10 µs clock cycles</li> <li>With one <b>set_fly_2_axes</b> call and one <b>set_fly_1_axis</b> call in two different 10 µs clock cycles</li> </ul> <p>- <b>set_fly_3_axes( 8, 1, 0, 12, 1, 0, 16, 1, 0 ) corresponds to</b>  <b>set_multi_mcbsp_in_list( Ctrl, P, Mode )</b></p>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<b>set_fly_1_axis, set_fly_2_axes</b>



<b>Undelayed Short List Command</b>	<b>set_fly_limits</b>
Function	Defines the boundaries of the customer-defined monitoring area for Processing-on-the-fly applications.
Call	<code>set_fly_limits( Xmin, Xmax, Ymin, Ymax )</code>
Parameters	<p>Xmin      Range boundary.              As a signed 32-bit value.              Allowed value range: [-524,288...+524,287].              Out-of-range values are clipped to the boundary values.</p> <p>Xmax      Like Xmin (analogously).</p> <p>Ymin      Like Xmin (analogously).</p> <p>Ymax      Like Xmin (analogously).</p>
Comments	<ul style="list-style-type: none"> <li>For usage of <b>set_fly_limits</b>, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>During initialization (with <b>load_program_file</b>), the boundary limits get set to the following default values:           <ul style="list-style-type: none"> <li>- Xmin = Ymin = -524,288</li> <li>- Xmax = Ymax = +524,287</li> </ul> </li> <li>Range boundaries specified using the parameter value 0 are set to the above-mentioned default values.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_marking_info</a>



<b>Undelayed Short List Command</b>	<b>set_fly_limits_z</b>
<b>Function</b>	Defines the boundaries of the customer-defined monitoring range for z axis Processing-on-the-fly applications.
<b>Call</b>	<code>set_fly_limits_z( Zmin, Zmax )</code>
<b>Parameters</b>	Zmin      Range boundary. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.
	Zmax      Like Zmin (analogously).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>set_fly_limits_z</b>, see also <a href="#">Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections"</a>, page 240.</li> <li>During initialization (with <b>load_program_file</b>), the boundary limits get set to the following default values:             <ul style="list-style-type: none"> <li>- Zmin = -524,288</li> <li>- Zmax = +524,287</li> </ul> </li> <li>Boundary limits specified using the parameter value 0 also get set to the above-mentioned default values.</li> </ul>
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified values for Zmin and Zmax by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.  In <b>RTC5 Compatibility Mode</b> , the RTC6 multiplies the specified values for Zmin and Zmax by 16. The allowed value range decreases accordingly.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_fly_limits</a> , <a href="#">get_marking_info</a>



<b>Normal List Command</b>	<b>set_fly_rot</b>
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of workpiece rotary movement (based on angular position values transferred to the RTC6 by encoder counter "Encoder0") and sets the corresponding <b>Resolution</b> parameter.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>set_fly_rot</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<b>set_fly_rot( Resolution )</b>
<b>Parameters</b>	Resolution    Number of steps per revolution. As a 64-bit IEEE floating point value. Allowed value range:  Resolution  > 100.0.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Resolution can be negative depending on the rotation direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>• For Processing-on-the-fly correction and determining the <b>Resolution</b> parameter, see <b>Chapter 8.6 "Processing-on-the-fly"</b>, page 227.</li> <li>• Before executing the command <b>set_fly_rot</b>, you should define the rotation center for Processing-on-the-fly correction by <b>set_rot_center</b> or <b>set_rot_center_list</b>. Otherwise, the default value (0 0) is applied.</li> <li>• If unallowed parameter values are supplied (for example, for Resolution = 0), then <b>set_fly_rot</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_rot</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "<b>set_fly_rot</b>" Processing-on-the-fly correction).</li> <li>• The various Processing-on-the-fly corrections cannot be arbitrarily combined, see <b>Section "Overview"</b>, page 227.</li> <li>• For deactivating Processing-on-the-fly correction, see <b>Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction"</b>, page 236.</li> <li>• By <b>set_control_mode( Bit #9 )</b>, it can be set in advance whether encoder counter "Encoder0" is reset by <b>set_fly_rot</b>: <ul style="list-style-type: none"> <li>– Immediately</li> <li>– Only after the subsequent external start signal</li> <li>– <b>simulate_ext_start</b></li> <li>– <b>simulate_ext_start_ctrl</b>, postponed by a track delay set by <b>simulate_ext_start</b>, <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b>, see also <b>set_control_mode</b>, Bit #2</li> </ul> </li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_rot_center</b> , <b>set_rot_center_list</b> , <b>fly_return</b> , <b>get_encoder</b> , <b>set_fly_rot_pos</b>



<b>Normal List Command</b>	<b>set_fly_rot_pos</b>
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of workpiece or scan system rotary movement (based on angular position values transferred to the RTC6 by the <b>McBSP interface</b> ). It thereby sets the corresponding <b>Resolution</b> parameter.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>set_fly_rot_pos</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>set_fly_rot_pos( Resolution )</code>
<b>Parameters</b>	Resolution    Number of steps per revolution. As a 64-bit IEEE floating point value. Allowed value range: $  \text{Resolution}   > 100.0$ .
<b>Comments</b>	<ul style="list-style-type: none"> <li>Resolution can be negative depending on the rotation direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>For Processing-on-the-fly correction and determining the <b>Resolution</b> parameter, see <b>Chapter 8.6 "Processing-on-the-fly"</b>, page 227.</li> <li>Before executing the command <b>set_fly_rot_pos</b>, you should define the rotation center for Processing-on-the-fly correction by <b>set_rot_center</b> or <b>set_rot_center_list</b>. Otherwise, the default value (0 0) is used.</li> <li>If unallowed parameter values are supplied (for example, for <b>Resolution</b> = 0), then <b>set_fly_rot_pos</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_rot_pos</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "<b>set_fly_rot_pos</b>" Processing-on-the-fly correction).</li> <li>The various Processing-on-the-fly corrections cannot be arbitrarily combined, see <b>Section "Overview"</b>, page 227.</li> <li>For deactivating Processing-on-the-fly correction, see <b>Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction"</b>, page 236.</li> <li>The <b>McBSP interface</b> cannot be simultaneously used for both Processing-on-the-fly applications and <b>Online Positioning</b> See also <b>Section "Notes"</b>, page 216.</li> <li>The <b>McBSP interface</b> always ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>, so available data is not transmitted, see <b>page 75</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_rot_center</a> , <a href="#">set_rot_center_list</a> , <a href="#">fly_return</a> , <a href="#">read_mcbsp</a> , <a href="#">set_fly_rot</a>



<b>Ctrl Command</b>	<b>set_fly_tracking_error</b>				
<b>Function</b>	Activates or deactivates tracking error compensation of encoder values for Processing-on-the-fly applications.				
<b>Call</b>	<code>set_fly_tracking_error( TrackingErrorX, TrackingErrorY )</code>				
<b>Parameters</b>	<table><tr><td>TrackingErrorX</td><td>Tracking error for the x axis. In units of 10 <math>\mu</math>s. As a signed 32-bit value. Allowed value range: [0...65,535]. Excessive bits are ignored.</td></tr><tr><td>TrackingErrorY</td><td>Tracking error for the y axis. Otherwise, like TrackingErrorX.</td></tr></table>	TrackingErrorX	Tracking error for the x axis. In units of 10 $\mu$ s. As a signed 32-bit value. Allowed value range: [0...65,535]. Excessive bits are ignored.	TrackingErrorY	Tracking error for the y axis. Otherwise, like TrackingErrorX.
TrackingErrorX	Tracking error for the x axis. In units of 10 $\mu$ s. As a signed 32-bit value. Allowed value range: [0...65,535]. Excessive bits are ignored.				
TrackingErrorY	Tracking error for the y axis. Otherwise, like TrackingErrorX.				
<b>Comments</b>	<ul style="list-style-type: none"><li>To date, this command has no effect.</li></ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Changed functionality. To date, this command has no effect.				
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.				
<b>References</b>	–				



<b>Normal List Command</b>	<b>set_fly_x</b>
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of a linear workpiece-movement in the X direction (based on position values transferred to the RTC6 by encoder counter "Encoder0") and sets the corresponding scaling factor.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>set_fly_x</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<b>set_fly_x( ScaleX )</b>
<b>Parameters</b>	ScaleX     Scaling factor for the x direction (encoder counter "Encoder0") in <i>bits/count</i> . As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleX}  \leq 16000.0$ .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• ScaleX can be negative depending on the motion direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>• For Processing-on-the-fly correction and determination of the scaling factor, see <b>Chapter 8.6 "Processing-on-the-fly", page 227</b>.</li> <li>• If unallowed parameter values are supplied (for example, for ScaleX = 0), then <b>set_fly_x</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_x</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "set_fly_x" Processing-on-the-fly correction).</li> <li>• The various Processing-on-the-fly corrections cannot be arbitrarily combined, see <b>Section "Overview", page 227</b>.</li> <li>• For deactivating Processing-on-the-fly correction, see <b>Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236</b>.</li> <li>• By <b>set_control_mode( Bit #9 )</b>, it can be set in advance whether encoder counter "Encoder0" is reset by <b>set_fly_x</b>: <ul style="list-style-type: none"> <li>– Immediately</li> <li>– Only after the external start signal</li> <li>– <b>simulate_ext_start</b></li> <li>– <b>simulate_ext_start_ctrl</b>, postponed by a track delay set by <b>simulate_ext_start</b>, <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b>, see also <b>set_control_mode</b>, Bit #2</li> </ul> </li> <li>• You can also switch on <b>set_fly_x/set_fly_y</b> Processing-on-the-fly correction by <b>activate_fly_xy/activate_fly_xy_encoder</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: changed value range. In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified scaling factor by 16. The allowed value range changes accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>fly_return</b> , <b>set_fly_y</b> , <b>get_encoder</b> , <b>set_fly_x_pos</b> , <b>set_fly_y_pos</b> , <b>activate_fly_xy</b> , <b>set_fly_2d</b>



<b>Normal List Command</b>	<b>set_fly_x_pos</b>
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of a linear workpiece or scan system movement in the X direction (based on position values transferred to the RTC6 by the <a href="#">McBSP interface</a> ); thereby sets the corresponding scaling factor.
<b>Restriction</b>	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <b>set_fly_x_pos</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>set_fly_x_pos( ScaleX )</code>
<b>Parameters</b>	ScaleX     Scaling factor for the x direction in (RTC6) <i>bits</i> /(McBSP) <i>bit</i> . As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleX}  \leq 16000.0$ .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• ScaleX can be negative depending on the motion direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>• For Processing-on-the-fly correction and determination of the scaling factor, see <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>.</li> <li>• If unallowed parameter values are supplied (for example, for ScaleX = 0), then <b>set_fly_x_pos</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_x_pos</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "set_fly_x_pos" Processing-on-the-fly correction).</li> <li>• The various Processing-on-the-fly corrections cannot be arbitrarily combined, see <a href="#">Section "Overview", page 227</a>.</li> <li>• For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236</a>.</li> <li>• For 1D correction (when only <b>set_fly_x_pos</b> is used), the <a href="#">McBSP interface</a> provides a (signed) 32-bit value. In contrast, only a (signed) 16-bit value per axis is supplied for 2D correction (<b>set_fly_x_pos</b> and <b>set_fly_y_pos</b>). Here, <b>set_fly_x_pos</b> uses the <a href="#">McBSP interface</a>'s lower 16 bits for the x value and <b>set_fly_y_pos</b> uses its upper 16-bits for the y value.</li> <li>• The <a href="#">McBSP interface</a> cannot be simultaneously used for both Processing-on-the-fly applications and <a href="#">Online Positioning</a>. See also <a href="#">Section "Notes", page 216</a>.</li> <li>• The <a href="#">McBSP interface</a> always ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>, so available data is not transmitted, see <a href="#">page 75</a>.</li> </ul>
RTC4→RTC6	New command.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified scaling factor by 16. The allowed value range changes accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">fly_return</a> , <a href="#">set_fly_y_pos</a> , <a href="#">read_mcbsp</a> , <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a>

<b>Normal List Command</b>	<b>set_fly_y</b>
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of a linear workpiece-movement in the Y direction (based on position values transferred to the RTC6 by encoder counter "Encoder1") and sets the corresponding scaling factor.
<b>Restriction</b>	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <b>set_fly_y</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<b>set_fly_y( ScaleY )</b>
<b>Parameters</b>	ScaleY     Scaling factor for the y direction (encoder counter "Encoder1") in <i>bits/count</i> . As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleY}  \leq 16000.0$ .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• ScaleY can be negative depending on the motion direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>• For Processing-on-the-fly correction and determination of the scaling factor, see <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>.</li> <li>• If unallowed parameter values are supplied (for example, for ScaleY = 0), then <b>set_fly_y</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_y</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "set_fly_y" Processing-on-the-fly correction).</li> <li>• The various Processing-on-the-fly corrections cannot be arbitrarily combined, see <a href="#">Section "Overview", page 227</a>.</li> <li>• For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236</a>.</li> <li>• By <b>set_control_mode( Bit #9 )</b>, it can be set in advance whether encoder counter "Encoder0" is reset by <b>set_fly_y</b>: <ul style="list-style-type: none"> <li>– Immediately</li> <li>– Only after the subsequent external start signal</li> <li>– <a href="#">simulate_ext_start</a></li> <li>– <a href="#">simulate_ext_start_ctrl</a>, postponed by a track delay set by <a href="#">simulate_ext_start</a>, <a href="#">set_ext_start_delay</a> or <a href="#">set_ext_start_delay_list</a>, see also <b>set_control_mode</b>, Bit #2</li> </ul> </li> <li>• You can also switch on <b>set_fly_x/set_fly_y</b> Processing-on-the-fly correction by <a href="#">activate_fly_xy/activate_fly_xy_encoder</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: changed value range. In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified scaling factor by 16. The allowed value range changes accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">fly_return</a> , <a href="#">set_fly_x</a> , <a href="#">get_encoder</a> , <a href="#">set_fly_x_pos</a> , <a href="#">set_fly_y_pos</a> , <a href="#">activate_fly_xy</a> , <a href="#">set_fly_2d</a>



<b>Normal List Command</b>	<b>set_fly_y_pos</b>
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of a linear workpiece or scan system movement in the Y direction (based on position values transferred to the RTC6 by the <a href="#">McBSP interface</a> ); thereby sets the corresponding scaling factor.
<b>Restriction</b>	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <b>set_fly_y_pos</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>set_fly_y_pos( ScaleY )</code>
<b>Parameters</b>	ScaleY     Scaling factor for the y direction in (RTC6) <i>bits</i> /(McBSP) <i>bit</i> . As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleY}  \leq 16000.0$ .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• ScaleY can be negative depending on the motion direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>• For Processing-on-the-fly correction and determination of the scaling factor, see <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>.</li> <li>• If unallowed parameter values are supplied (for example, for ScaleY = 0), then <b>set_fly_y_pos</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_y_pos</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "set_fly_y_pos" Processing-on-the-fly correction).</li> <li>• The various Processing-on-the-fly corrections cannot be arbitrarily combined, see <a href="#">Section "Overview", page 227</a>.</li> <li>• For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236</a>.</li> <li>• For 1D correction (when only <b>set_fly_y_pos</b> is used), the <a href="#">McBSP interface</a> provides a (signed) 32-bit value. In contrast, only a (signed) 16-bit value per axis is supplied for 2D correction (<b>set_fly_x_pos</b> and <b>set_fly_y_pos</b>). Here, <b>set_fly_x_pos</b> uses the <a href="#">McBSP interface</a>'s lower 16 bits for the x value and <b>set_fly_y_pos</b> uses its upper 16-bits for the y value.</li> <li>• The <a href="#">McBSP interface</a> cannot be simultaneously used for both Processing-on-the-fly applications and <a href="#">Online Positioning</a>. See also <a href="#">Section "Notes", page 216</a>.</li> <li>• The <a href="#">McBSP interface</a> always ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>, so available data is not transmitted, see <a href="#">page 75</a>.</li> </ul>
RTC4→RTC6	New command.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified scaling factor by 16. The allowed value range changes accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">fly_return</a> , <a href="#">set_fly_x_pos</a> , <a href="#">read_mcbsp</a> , <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a>

<b>Normal List Command</b>	<b>set_fly_z</b>				
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of a linear workpiece-movement in the Z direction (based on position values transferred to the RTC6 by the specified encoder counter) and sets the corresponding scaling factor.				
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>set_fly_z</b> terminates the Processing-on-the-fly process (even though it could not have been activated).				
<b>Call</b>	<b>set_fly_z( ScaleZ, EncoderNo )</b>				
<b>Parameters</b>	<table> <tr> <td>ScaleZ</td> <td>Scaling factor for the z direction. In bits/count. As a 64-bit IEEE floating point value. Allowed value range: <math>1/256 \leq  \text{ScaleZ}  \leq 16,000.0</math>.</td> </tr> <tr> <td>EncoderNo</td> <td>Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values: = 0: Encoder counter "Encoder0". = 1: Encoder counter "Encoder1".</td> </tr> </table>	ScaleZ	Scaling factor for the z direction. In bits/count. As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleZ}  \leq 16,000.0$ .	EncoderNo	Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values: = 0: Encoder counter "Encoder0". = 1: Encoder counter "Encoder1".
ScaleZ	Scaling factor for the z direction. In bits/count. As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleZ}  \leq 16,000.0$ .				
EncoderNo	Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values: = 0: Encoder counter "Encoder0". = 1: Encoder counter "Encoder1".				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• ScaleZ can be negative depending on the motion direction of the workpiece. The restricted value range applies only to the absolute value.</li> <li>• For Processing-on-the-fly correction and determination of the scaling factor, see <b>Chapter 8.6 "Processing-on-the-fly", page 227</b>.</li> <li>• If unallowed ScaleZ parameter values are supplied (for example, for ScaleZ = 0), then <b>set_fly_z</b> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <b>set_fly_z</b> (but does not deactivate any other Processing-on-the-fly correction). The latter case leads to a jump (at jump speed) to the endpoint of the most recently executed vector or arc command (without "set_fly_z" Processing-on-the-fly correction).</li> <li>• For deactivating Processing-on-the-fly correction, see <b>Section "Notes on Usage", page 243</b>.</li> <li>• By <b>set_control_mode( Bit #9 )</b>, it can be set in advance whether encoder counter "Encoder0" is reset by <b>set_fly_z</b>: <ul style="list-style-type: none"> <li>– Immediately</li> <li>– Only after the subsequent external start signal</li> <li>– <b>simulate_ext_start</b></li> <li>– <b>simulate_ext_start_ctrl</b>, postponed by a track delay set by <b>simulate_ext_start</b>, <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</b>, see also <b>set_control_mode</b>, Bit #2</li> </ul> </li> <li>• If EncoderNo &gt; 1, <b>set_fly_z</b> is replaced by a <b>list_nop (get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> </ul>				
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified scaling factor by 16. The allowed value range decreases accordingly.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<b>fly_return_z</b>				



<b>Ctrl Command</b>	<b>set_free_variable</b>
<b>Function</b>	Sets a free variable to the desired value.
<b>Call</b>	<code>set_free_variable( No, Value )</code>
<b>Parameters</b>	No      Number of the free variable to be set. As an unsigned 32-bit value. Allowed value range: [0...7]. Only the 3 least significant bits are evaluated.
	Value      Desired variable value. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 6.9.1 "Free Variables", page 124</a>.</li> <li><b>Standalone Functionality:</b> <code>set_free_variable</code> is a control command allowed for automatic booting (although the corresponding list command <code>set_free_variable_list</code> exists), see <a href="#">Table 8, page 864 in Chapter 15.7 "Standalone Functionality", page 859</a>. By <code>set_free_variable</code>, any version code can be assigned to the <b>Boot Image</b>, which in <b>PC operation</b> can be read and checked before an /START.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.  Last change version DLL 618: control command allowed for automatic booting, siehe <a href="#">page 863</a> .
References	<a href="#">set_free_variable_list</a> , <a href="#">get_free_variable</a> , <a href="#">set_trigger</a> , <a href="#">eth_boot_dcmsg</a>

<b>Undelayed Short List Command</b>	<b>set_free_variable_list</b>
<b>Function</b>	Like <code>set_free_variable</code> , but a list command.
<b>Call</b>	<code>set_free_variable_list( No, Value )</code>
<b>Parameters</b>	No      See <code>set_free_variable</code> .
	Value      See <code>set_free_variable</code> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <code>set_free_variable</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_free_variable</a>



<b>Ctrl Command</b>	<b>set_hi</b>
<b>Function</b>	Defines gain and offset values for the galvanometer scanners of the scan system attached to the specified scan head connector.
<b>Call</b>	<code>set_hi( HeadNo, GalvoGainX, GalvoGainY, GalvoOffsetX, GalvoOffsetY )</code>
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value. Allowed values: = 1:      First scan head connector. = 2:      Second scan head connector.</p> <p>GalvoGainX      Gain values. GalvoGainY      As 64-bit IEEE floating point values. Allowed value range: [0.01...100].</p> <p>GalvoOffsetX      Offset values. In bits. GalvoOffsetY      As 64-bit IEEE floating point values. Allowed value range: [-524,288...+524,287].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>set_hi</b>, see <a href="#">Section "Customer-Specific Calibration", page 264</a>.</li> <li>The specified gain and offset values overwrite the values that were set by <a href="#">auto_cal</a> and can themselves be overwritten by a subsequent call of <a href="#">auto_cal</a>.</li> <li>With changed gain and offset values, the transition is automatically performed at the predefined jump speed (see <a href="#">set_jump_speed</a>).</li> <li><b>set_hi</b> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>), if: <ul style="list-style-type: none"> <li>a parameter value is invalid</li> </ul> </li> <li><b>set_hi</b> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC6_BUSY</a>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li> <li>the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li><b>set_hi</b> is even executed, if: <ul style="list-style-type: none"> <li>a list has been paused by <a href="#">set_wait</a> (<b>PAUSED</b> status set)</li> </ul> </li> <li>For <a href="#">RTC6_PARAM_ERROR</a>, the <b>BUSY</b> status is not checked. Therefore the return codes <a href="#">RTC6_BUSY</a> and <a href="#">RTC6_PARAM_ERROR</a> do not occur simultaneously.</li> <li>If the <a href="#">Option "Second Scan Head Control"</a> has not been enabled, values specified for the second scan head control have no effect.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">auto_cal</a> , <a href="#">get_hi_pos</a> , <a href="#">write_hi_pos</a>



Ctrl Command	<b>set_input_pointer</b>
Function	Opens the list memory for writing with list commands and sets the input pointer to the specified (absolute) address in list memory ("List 1" or "List 2"). The next list command is stored at this address and all further list commands at the subsequent addresses in the selected list.
Call	<code>set_input_pointer( Pos )</code>
Parameters	Pos      Position (absolute memory address) of the input pointer [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li>• <b>set_input_pointer</b> performs basically like the command <b>set_start_list_pos</b> (see comments there). But <b>set_input_pointer</b> sets the input pointer based on a specified <i>absolute</i> memory address, whereas <b>set_start_list_pos</b> uses a specified list number and a <i>relative</i> memory address.</li> <li>• For <math>\text{Pos} \geq \text{Mem1} + \text{Mem2}</math> (see <b>config_list</b>), <b>Pos</b> is set to 0.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_start_list_pos, get_input_pointer</b>



<b>Undelayed Short List Command</b>	<b>set_io_cond_list</b>
Function	Sets the bits of the 16-bit digital output port on the EXTENSION 1 socket connector that are set in the parameter MaskSet, if the current IOvalue at the 16-bit digital input port on the EXTENSION 1 socket connector meets the following condition:  $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ (= if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0).
Call	set_io_cond_list( Mask1, Mask0, MaskSet )
Parameters	<p>Mask1      16-bit mask.  As an unsigned 32-bit value.  Only the lower 16 bits are evaluated.</p> <p>Mask0      Like Mask1.</p> <p>MaskSet    Like Mask1.</p>
Comments	<ul style="list-style-type: none"> <li>• <b>set_io_cond_list</b> sets only those bits of the digital output port that are set in the parameter MaskSet and leaves the other bits unchanged.</li> <li>• See <a href="#">Section "16-Bit Digital Input and 16-Bit Digital Output", page 67</a> and <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>.</li> </ul>
Examples (Pascal)	<ul style="list-style-type: none"> <li>• Set Bit #4 of the output port (DIGITAL OUT4), if Bit #0 of the input port (DIGITAL IN0) is set and Bit #1...Bit #3 (DIGITAL IN1...3) of the input port are not set:  <code>set_io_cond_list(\$0001, \$000E, \$0010)</code></li> <li>• Always set Bit #15 of the output port (and leave the other bits unchanged):  <code>set_io_cond_list(0, 0, \$8000)</code></li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">clear_io_cond_list</a> , <a href="#">write_io_port</a> , <a href="#">write_io_port_mask</a> , <a href="#">get_io_status</a> , <a href="#">read_io_port</a>



<b>Ctrl Command</b>	<b>set_jump_mode</b>
<b>Function</b>	Enables and activates or disables and deactivates jump mode for 2D jumps and sets the related parameters.
<b>Requirements</b>	Enabling is only possible if a jump-tuning-equipped intelliSCAN (with firmware version 2078 or higher) is attached to at least one of the two scan head connectors. Otherwise, <b>set_jump_mode</b> has no effect.
<b>Call</b>	ErrorCode = set_jump_mode( Flag, Length, VA1, VA2, VB1, VB2, JA1, JA2, JB1, JB2 )
<b>Parameters</b>	<p>Flag            Switching flag. As a signed 32-bit value.            Allowed values:            = -1: Jump mode is disabled. Afterward, switching the Flag by <b>set_jump_mode_list</b> is no longer possible.            = 0: Jump mode is enabled but deactivated.            Afterwards it is also possible to switch the flag by <b>set_jump_mode_list</b>.            = 1: Jump mode is enabled and activated.            Afterwards it is also possible to switch the flag by <b>set_jump_mode_list</b>.</p> <p>Length        Jump length limit (per axis) under which 2D jumps – even with activated jump mode – are performed in vector mode.            As an unsigned 32-bit value.</p> <p>VA1            Numbers of the tunings that should be used for jump execution in jump mode:            VA2            V: Vector tuning that is set at the end of a 2D jump.            VB1            J: Jump tuning that is set at the beginning of a 2D jump.            VB2            A: First scan head connector.            JA1            B: Second scan head connector.            JA2            1: x axis (STATUS channel, galvanometer scanner 2).            JA2            2: y axis (STATUS1 channel, galvanometer scanner 1).            JB1            Allowed values:            JB2            = -1: Tuning is neither checked nor used.            = 0...3: After passing a check, tuning is used for jump mode.            The allowed value range also depends on the number of tunings with which the attached scan system is equipped.</p> <p>As a signed 32-bit value.</p>



Ctrl Command	set_jump_mode
Result	<p>Error code. As a signed 32-bit value.</p> <p>0            No error: Flag successfully switched to 0 (jump mode deactivated, vector mode activated).</p> <p>1            No error: Flag successfully switched to 1 (jump mode activated, vector mode deactivated).</p> <p>-1          Flag successfully switched to -1 (jump mode deactivated and disabled).</p> <p>-2          Busy error: board <b>BUSY</b> or <b>INTERNAL-BUSY</b> (<b>get_last_error</b> return code <b>RTC6_BUSY</b>).</p> <p>-3          Board not responding: possibly no program loaded or PCI error (<b>get_last_error</b> return code <b>RTC6_TIMEOUT</b>).</p> <p>-4          Access error: board reserved for another user program (<b>get_last_error</b> return code <b>RTC6_ACCESS_DENIED</b>).</p> <p>&gt; 1         Did not pass the check (see also notes). Flag is set to -1 (jump mode deactivated and disabled).</p> <p>The following is returned:</p> <p>Byte #0 = 255. Byte #1 = Error code for first scan head connector. Byte #2 = Error code for second scan head connector. Byte #3 = 0.</p> <p>Whereby error code:</p> <ul style="list-style-type: none"> <li>=1: x axis (galvanometer scanner 2) not responding or no <b>intelliSCAN</b> (with firmware version 2078 or higher) attached.</li> <li>=2: y axis (galvanometer scanner 1) not responding or no <b>intelliSCAN</b> (with firmware version 2078 or higher) attached.</li> <li>=4: no correction table assigned.</li> <li>=8: incorrect tuning number(s): incorrect type or unsuitable for rapid switching.</li> </ul>



Ctrl Command	<code>set_jump_mode</code>
Comments	<ul style="list-style-type: none"> <li>For usage of <code>set_jump_mode</code>, see <a href="#">Chapter 8.1.5 "Jump Mode", page 203</a>.</li> <li>A check (see also <a href="#">Section "Requirements and Activation", page 204</a>) is only performed if <code>Flag</code> = -1 (the initialization state) prior to the <code>set_jump_mode</code> call and/or if the supplied tuning numbers do not match those stored on the board. Otherwise, only the flag is switched.</li> </ul> <p>For the check, the board must not be <b>BUSY</b> or <b>INTERNAL-BUSY</b>, because meanwhile the to-be-returned data type changes and "Automatic Laser Control" is deactivated (both get restored at the end of the command).</p> <p>Depending on results of the check, different error codes are returned (see above). In case of error, <code>Flag</code> gets set to -1 (jump mode deactivated and disabled).</p> <p>If the check is successful, then you can afterward (even by <code>set_jump_mode_list</code> during processing of a list) switch freely between the states <code>Flag</code> = 1 (jump mode activated, vector mode deactivated) and <code>Flag</code> = 0 (jump mode deactivated, vector mode activated) without another check having to be performed.</p> <ul style="list-style-type: none"> <li>Use -1 as the tuning number if certain tunings should not be checked (for example, because no intelliSCAN scan system is attached or specific tunings are not available – vector tuning, for example, is not needed in pure drilling applications) or if, after switching to jump tuning, it is not desirable to return to vector tuning. As a result, such tunings are neither checked nor switched on.</li> </ul> <p>If the <a href="#">Option "Second Scan Head Control"</a> is not enabled, then the tuning numbers for the second scan head connector (B) is automatically set to -1 (even if others were supplied).</p> <p>If all tuning numbers are -1, then <code>Flag</code> is set to -1 (return value -1).</p> <ul style="list-style-type: none"> <li>Even after successful activation of jump mode (<code>Flag</code> = 1), the first servo switching only occurs after the first subsequent 2D jump (see <a href="#">Section "Functional Principle", page 203</a>). If the currently set tuning then does not match the jump or vector tuning specified by <code>set_jump_mode</code>, then the first switching can take somewhat longer (approx. 250 ms), depending on the currently set tuning. You can determine ahead of time whether this is so by calling <code>set_jump_mode</code> using the currently set tuning as a parameter. If true (return value &gt; 1, Error code = 2), then you can achieve the desired operational sequence by calling <a href="#">control_command</a> before the first 2D jump to set the tuning to one that was supplied by <code>set_jump_mode</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_jump_mode_list</a> , <a href="#">load_jump_table_offset</a> , <a href="#">set_jump_table</a>



<b>Normal List Command</b>	<b>set_jump_mode_list</b>
<b>Function</b>	Activates or deactivates and disables jump mode for 2D jumps.
<b>Requirements</b>	See <a href="#">set_jump_mode</a> .
<b>Call</b>	<code>set_jump_mode_list( Flag )</code>
<b>Parameters</b>	Flag      See <a href="#">set_jump_mode</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For usage of <code>set_jump_mode_list</code>, see <a href="#">Chapter 8.1.5 "Jump Mode", page 203</a>.</li> <li>• <code>set_jump_mode_list</code> functions like the control command <code>set_jump_mode</code> (see notes there) but, as a list command, has the following differences: <ul style="list-style-type: none"> <li>– No tunings or jump length limits can be supplied by <code>set_jump_mode_list</code>.</li> <li>– <code>set_jump_mode_list</code> does not perform a check. <code>Flag</code> must have previously been successfully set to 0 or 1 by <code>set_jump_mode</code>.</li> <li>– Though jump mode can be deactivated and disabled by setting <code>Flag</code> to -1 by <code>set_jump_mode</code> or <code>set_jump_mode_list</code>, it can only be reactivated again by <code>set_jump_mode</code> (if the check is successful). If <code>Flag</code> was -1 prior to calling <code>set_jump_mode_list</code>, then <code>set_jump_mode_list</code> has no effect.</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
Reference	<a href="#">set_jump_mode</a>



<b>Undelayed Short List Command</b>	<b>set_jump_speed</b>
Function	Defines the jump speed for vector commands.
Call	<code>set_jump_speed( Speed )</code>
Parameters	Speed      Jump speed. In bits/ms. As a 64-bit IEEE floating point value. Allowed value range: [1.6...800000.0].
Comments	<ul style="list-style-type: none"> <li>By default a jump speed of 10000 bits/ms is preset.</li> <li>The specified jump speed is used for all jump commands until a new value is specified.</li> <li>The actual jump speed <math>v_{jump}</math> in the image plane in m/s is derived from the specified Speed value [bits/ms] and the calibration factor <math>K</math> [Bits/mm] as follows:</li> </ul> $v_{jump} = \text{Speed} / K$ <p>The calibration factor <math>K</math> can be queried from the correction table by <a href="#">get_table_para</a> or <a href="#">get_head_para</a>.</p> <ul style="list-style-type: none"> <li><code>set_jump_speed</code> is also available as control command <a href="#">set_jump_speed_ctrl</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified Speed value by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">jump_abs</a> , <a href="#">jump_rel</a> , <a href="#">set_mark_speed</a> , <a href="#">set_jump_speed_ctrl</a>



<b>Ctrl Command</b>	<b>set_jump_speed_ctrl</b>
<b>Function</b>	Like <b>set_jump_speed</b> , but a control command.
<b>Call</b>	<code>set_jump_speed_ctrl( Speed )</code>
<b>Parameters</b>	<p>Speed      Jump speed. In bits/ms.            As a 64-bit IEEE floating point value.            Allowed value range: [1.6...800000.0].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_jump_speed_ctrl</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:               <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> </ul> </li> <li>• <b>set_jump_speed_ctrl</b> is even executed, if:               <ul style="list-style-type: none"> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> </ul>
RTC4→RTC6	New command. In <b>RTC4 Compatibility Mode</b> : as <b>set_jump_speed</b> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_jump_speed</b> , <b>set_mark_speed</b> , <b>set_mark_speed_ctrl</b>



<b>Ctrl Command</b>	<b>set_jump_table</b>								
<b>Function</b>	Reads the jump delay table with 1024 unsigned 16-bit values stored at the supplied PC address and loads them onto the board as the jump delay table (see notes for <a href="#">get_jump_table</a> ).								
<b>Call</b>	ErrorCode = set_jump_table( Addr )								
<b>Parameters</b>	Addr      PC Address of a 2048-byte area of PC main memory.								
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <table> <tr> <td>0</td><td>No error.</td></tr> <tr> <td>1</td><td>Busy error: board <b>BUSY</b> or <b>INTERNAL-BUSY</b> (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>).</td></tr> <tr> <td>4</td><td>Verify error: <b>DSP</b> memory error.</td></tr> <tr> <td>11</td><td>RTC6 board driver error.</td></tr> </table>	0	No error.	1	Busy error: board <b>BUSY</b> or <b>INTERNAL-BUSY</b> ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).	4	Verify error: <b>DSP</b> memory error.	11	RTC6 board driver error.
0	No error.								
1	Busy error: board <b>BUSY</b> or <b>INTERNAL-BUSY</b> ( <a href="#">get_last_error</a> return code <b>RTC6_BUSY</b> ).								
4	Verify error: <b>DSP</b> memory error.								
11	RTC6 board driver error.								
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_jump_table</b> is not executed (<a href="#">get_last_error</a> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been paused by <a href="#">pause_list</a>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <b>set_jump_table</b> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <a href="#">set_wait</a> (<b>PAUSED</b> status set)</li> </ul> </li> </ul>								
RTC4→RTC6	New command.								
RTC5→RTC6	Unchanged functionality.								
Version info	Available as of version DLL 600, OUT 600, RBF 600.								
Reference	<a href="#">get_jump_table</a> , <a href="#">load_jump_table_offset</a>								



<b>Ctrl Command</b>	<b>set_laser_control</b>
<b>Function</b>	Defines and enables or disables the laser control signals.
<b>Call</b>	<code>set_laser_control( Ctrl )</code>
<b>Parameters</b>	<p>Ctrl      As an unsigned 32-bit value.</p> <p>Bit #0    Pulse Switch Setting (does not concern Laser Mode 4 or Laser Mode 6).  <b>(LSB)</b>   The setting only affects those laser control signals (more precisely: those LASER1 or LASER2 "laser active" modulation pulses in CO<sub>2</sub> Mode or LASER1 Q-Switch pulses in the YAG modes) that are not yet fully processed at completion of the LASERON signal, see <a href="#">figure 48</a> and <a href="#">figure 49</a>.  = 0:   The signals are cut off at the end of the LASERON signal.  = 1:   The final pulse fully executes despite completion of the LASERON signal. See "<a href="#">Pulse Completion</a>", page 175.</p> <p>Bit #1    Phase shift of the laser control signals (does not concern Laser Mode 4 or Laser Mode 6).  = 0:   No phase shift.  = 1:   CO<sub>2</sub> Mode: The LASER1 signal is exchanged with the LASER2 signal.  YAG modes: The LASER1 is shifted back 180° (half a signal period).</p> <p>Bit #2    Enabling or disabling of laser control signals for "Laser active" operation.  = 0:   The "Laser active" laser control signals are enabled.  = 1:   The "Laser active" laser control signals are disabled  (the signals are set to their respective "Off" level).</p> <p>Bit #3    LASERON signal level.  = 0:   The signal at the LASERON port is set to active-HIGH.  = 1:   The signal at the LASERON port is set to active-LOW.</p> <p>Bit #4    LASER1/LASER2 signal level.  = 0:   The signals at the LASER1 and LASER2 output ports are set to active-HIGH.  = 1:   The signals at the LASER1 and LASER2 output ports are set to active-LOW.</p> <p>Bit #5    Determines for <a href="#">laser_on_pulses_list</a> whether external signal pulses (at the LASER connector's DIGITAL IN1 digital input) are to be counted at rising or falling edges:  = 0:   At the falling edge.  = 1:   At the rising edge.</p> <p>Bit #6    = 0:   Synchronization is switched off (default setting).  See <a href="#">Chapter 7.4.10 "Synchronization of the RTC6 Clock Cycle and an External Clock Signal"</a>, page 196.  = 1:   Synchronization is switched on.</p>



<b>Ctrl Command</b>	<b>set_laser_control</b>
<b>Parameters (cont'd)</b>	<p>Bit #7      = 0: The “constant pulse length” mode is switched off (default setting).</p> <p>                = 1: The “constant pulse length” mode is switched on. See <a href="#">Chapter 7.4.8 “Pulse Picking Laser Mode”, page 185</a> and <a href="#">set_pulse_picking_length</a>.</p> <p>Bit #8      Reserved.</p> <p>...</p> <p>Bit #15     Reserved.</p> <p>Bit #16     For the automatic suppression of laser control signals is used: PowerOK of the head A, x axis.</p> <p>Bit #17     Like Bit #16, but: TempOK of the head A, x axis.</p> <p>Bit #18     Like Bit #16, but: PosAck of the head A, x axis.</p> <p>Bit #19     Like Bit #16, but: PowerOK of the head A, y axis.</p> <p>Bit #20     Like Bit #16, but: TempOK of the head A, y axis.</p> <p>Bit #21     Like Bit #16, but: PosAck of the head A, y axis.</p> <p>Bit #22     Like Bit #16, but: PowerOK of the head B, x axis.</p> <p>Bit #23     Like Bit #16, but: TempOK of the head B, x axis.</p> <p>Bit #24     Like Bit #16, but: PosAck of the head B, x axis.</p> <p>Bit #25     Like Bit #16, but: PowerOK of the head B, y axis.</p> <p>Bit #26     Like Bit #16, but: TempOK of the head B, y axis.</p> <p>Bit #27     Like Bit #16, but: PosAck of the head B, y axis.</p> <p>Bit #28     = 1: In case of error, automatic monitoring (automatic suppression of laser control signals) automatically generates a /STOP signal (list stops, laser control signals get permanently switched off).</p> <p>Bit #29     = 1: In case of error according to Bit #28, the <a href="#">stop_execution</a> is forwarded as /Master-STOP (see <a href="#">figure 63</a>) to all Master/Slave-connected RTC6 boards. See <a href="#">master_slave_config</a>.</p> <p>Bit #30     Reserved.</p> <p>Bit #31     Reserved.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>In the default setting (after <a href="#">load_program_file</a>), all bits are set to 0. After a hardware reset, however, the settings become effective following the first-time call of <a href="#">set_laser_control</a>. Prior to this, all laser signal outputs (LASERON, LASER1 and LASER2) are in the high-impedance mode. TTL states (LOW or HIGH) only become available when <a href="#">set_laser_control</a> is called to define the desired TTL level, see also <a href="#">Chapter 7.4 “Laser Control”, page 173</a>. After <a href="#">load_program_file</a>, which deactivates the laser control signals, <a href="#">set_laser_control</a> must be called for first-time activation.</li> </ul>



Ctrl Command	<a href="#">set_laser_control</a>
Comments (cont'd)	<ul style="list-style-type: none"> <li>For the RTC5/RTC6's predecessors, the laser signal levels are defined by solder jumpers. The RTC5/RTC6 lets users control them completely by software. <a href="#">get_startstop_info</a> queries the current status of the laser control signals (<a href="#">Bit #9</a>) and whether the signals are set to active-HIGH or active-LOW (<a href="#">Bit #13</a>).</li> <li>Enabling and disabling of laser control signals can also be achieved by <a href="#">enable_laser</a> or <a href="#">disable_laser</a>.</li> <li>Even if the laser control signals were enabled with <a href="#">set_laser_control</a> or <a href="#">enable_laser</a>, they are not outputted without further commands, see <a href="#">Chapter 7.4 "Laser Control", page 173</a>.</li> <li>The phase shift of the laser control signals (<a href="#">Bit #1</a> = 1) can be set for better synchronization of an analog output, for example, in softstart mode, see <a href="#">Chapter 7.4.7 "Softstart Mode (not yet implemented)", page 184</a> or the pixel output mode, see <a href="#">Chapter 8.7 "Pixel Output Mode – Marking Pixel Images (Bitmaps)", page 249</a>.</li> <li>For usage of the <a href="#">Bit #16...Bit #29</a> for automatic suppression of laser control signals, see <a href="#">Section "Automatic Suppression of Laser Control Signals", page 176</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 614, OUT 614, RBF 619: <a href="#">Bit #29</a> .
References	<a href="#">set_laser_mode</a>



<b>Undelayed Short List Command</b>	<b>set_laser_delays</b>				
Function	Sets the LaserOn delay and the LaserOff delay.				
Call	<code>set_laser_delays( LaserOnDelay, LaserOffDelay )</code>				
Parameters	<table> <tr> <td>LaserOnDelay</td> <td>LaserOn delay. As a signed 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [-2<sup>31</sup> ... +(2<sup>21</sup>-1)].</td> </tr> <tr> <td>LaserOffDelay</td> <td>LaserOff delay. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...+(2<sup>21</sup>-1)].</td> </tr> </table>	LaserOnDelay	LaserOn delay. As a signed 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [-2 <sup>31</sup> ... +(2 <sup>21</sup> -1)].	LaserOffDelay	LaserOff delay. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...+(2 <sup>21</sup> -1)].
LaserOnDelay	LaserOn delay. As a signed 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [-2 <sup>31</sup> ... +(2 <sup>21</sup> -1)].				
LaserOffDelay	LaserOff delay. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...+(2 <sup>21</sup> -1)].				
Comments	<ul style="list-style-type: none"> <li>Values over (2<sup>21</sup>-1) are clipped.</li> <li>The delays can be freely chosen within the allowed ranges. If <code>LaserOffDelay &lt; LaserOnDelay</code>, overlaps of LaserOn and LaserOff are automatically prevented during processing of short vectors, see <a href="#">Section "Automatic Delay Adjustments", page 144</a>.</li> <li>Observe the notes in <a href="#">Chapter 7.2.1 "Laser Delays", page 134</a>.</li> <li>A negative <code>LaserOnDelay</code> value extends the total marking time.</li> <li>The XY2-100 converter, see <a href="#">Chapter 4.5.2 "XY2-100 Converter (Accessory)", page 58</a>, introduces a 10 µs runtime latency to scan system control. This runtime latency can be compensated by increasing the LaserOn delay and LaserOff delay by 10 µs each.</li> <li>The default setting after <code>load_program_file</code> corresponds to <code>set_laser_delays( 20, 20 )</code>.</li> </ul>				
RTC4→RTC6	Essentially similar functionality. In addition: increased value range.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified delays by 64. The allowed value ranges decrease accordingly.				
RTC5→RTC6	Unchanged functionality.  In <a href="#">RTC5 Compatibility Mode</a> , the RTC6 multiplies the specified delays by 32. The allowed value ranges decrease accordingly.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">set_scanner_delays</a>				



<b>Ctrl Command</b>	<b>set_laser_mode</b>
<b>Function</b>	Sets the laser mode of the RTC6.
<b>Call</b>	<code>set_laser_mode( Mode )</code>
<b>Parameters</b>	<p>Mode = 0: CO<sub>2</sub> Mode.  = 1: YAG Mode 1.  = 2: YAG Mode 2.  = 3: YAG Mode 3.  = 4: Laser Mode 4.  = 5: YAG Mode 5.  = 6: Laser Mode 6.</p> <p>As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The available laser signals depend on the set laser mode, see also <a href="#">Chapter 7.4 "Laser Control", page 173</a>.</li> </ul>
RTC4→RTC6	Additional laser modes, for example, YAG Mode 5, Laser Mode 6 and Pulse Picking. Otherwise unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_laser_control</a> , <a href="#">set_laser_pulses_ctrl</a> , <a href="#">set_laser_pulses</a> , <a href="#">set_laser_timing</a> , <a href="#">set_firstpulse_killer</a> , <a href="#">set_firstpulse_killer_list</a> , <a href="#">set_standby</a> , <a href="#">set_standby_list</a> , <a href="#">set_qswitch_delay</a> , <a href="#">set_qswitch_delay_list</a>

<b>Ctrl Command</b>	<b>set_laser_off_default</b>
<b>Function</b>	Sets the default output value for the ANALOG OUT1 and ANALOG OUT2 ports, as well as for the 8-bit digital output port of the RTC6.
<b>Call</b>	<code>set_laser_off_default( AnalogOut1, AnalogOut2, DigitalOut )</code>
<b>Parameters</b>	<p>AnalogOut1 12-bit value for analog output port ANALOG OUT1, see also <a href="#">Section "12-Bit Analog Output Port 1 and 2", page 63</a>. As an unsigned 32-bit value. Higher bits are ignored (exception: AnalogOut1/AnalogOut2 = "-1", see comments for <a href="#">set_port_default</a>).</p> <p>AnalogOut2 12-bit value for analog output port ANALOG OUT2. See AnalogOut1.</p> <p>DigitalOut 8-bit value for the 8-bit digital output port, see also <a href="#">Section "8-Bit Digital Output Port", page 70</a>. As an unsigned 32-bit value. Higher bits are ignored (exception: DigitalOut = "-1", see comments for <a href="#">set_port_default</a>).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The default values can also be defined by <a href="#">set_port_default</a>. See also all comments there.</li> </ul>
RTC4→RTC6	New command.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified values for AnalogOut1 and AnalogOut2 by 4.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_port_default</a>



<b>Ctrl Command</b>	<a href="#">set_laser_pin_out</a>
<b>Function</b>	Sends a value to the two digital outputs of the LASER connector.
<b>Call</b>	<code>set_laser_pin_out( Pins )</code>
<b>Parameters</b>	<p>Pins      Output value (DIGITAL OUT1 and DIGITAL OUT2).  As an unsigned 32-bit value.</p> <p>Bit # 0: DIGITAL OUT1.  Bit # 1: DIGITAL OUT2.  Bit # 2: Reserved.  ...  Bit #31:Reserved.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 9.1.3 "2 Bit Digital Output Port", page 269.</a></li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_laser_pin_out_list</a> , <a href="#">get_laser_pin_in</a>

<b>Undelayed Short List Command</b>	<a href="#">set_laser_pin_out_list</a>
<b>Function</b>	Like <a href="#">set_laser_pin_out</a> , but a list command.
<b>Call</b>	<code>set_laser_pin_out_list( Pins )</code>
<b>Parameters</b>	Pins      Like <a href="#">set_laser_pin_out</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">set_laser_pin_out</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_laser_pin_out</a>



<b>Undelayed Short List Command</b>	<b>set_laser_power</b>
Function	Synchronizes laser power outputs at the output ports (ANALOG OUT1, ANALOG OUT2, 8-bit digital output port, 16-bit digital output port) and laser delays.
Call	<code>set_laser_power( Port, Power )</code>
Parameters	<p>Port                                     Output port for which a laser power value <code>Power</code> is to be defined.               As an unsigned 32-bit value.               Allowed values:               = 0: ANALOG OUT1 output port.               = 1: ANALOG OUT2 output port.               = 2: 8-bit digital output port (EXTENSION 2).               = 3: 16-bit digital output port (EXTENSION 1).               For values &gt;3 <b>set_laser_power</b> is not executed               (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</p> <p>Power                                     Desired laser power value.               As an unsigned 32-bit value.               Allowed values:               For Port = 0: 12-bit values [0...4095].               For Port = 1: 12-bit values [0...4095].               For Port = 2: 8-bit values [0...255].               For Port = 3: 16-bit values [0...(2<sup>16</sup>-1)].               Out-of-range values are clipped to the boundary values.               <b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>.</p>
Comments	<ul style="list-style-type: none"> <li>In particular, the <b>set_laser_power</b> command is to be used with excelliSCAN scan heads, if laser power needs to be changed within a <b>Polyline</b>.</li> <li>The <b>set_laser_power</b> command is also necessary, for example, if <i>short vectors</i> need unequal laser power and laser delays extend beyond the next vector (in particular in <b>DSP mode 3</b>).</li> </ul>
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified <code>Power</code> value for <code>Port = 0</code> and <code>Port = 1</code> by 4. The allowed value range decreases accordingly.
RTC5→RTC6	New command.
Version info	Available as of version DLL 602, OUT 602, RBF 602.
References	<b>get_last_error</b> , <b>set_dsp_mode</b> , <b>set_RTC4_mode</b> , <b>write_da_x</b>



<b>Delayed Short List Command</b>	<b>set_laser_pulses</b>
<b>Function</b>	Defines the output period and the pulse lengths for the laser signals LASER1 and LASER2 for "laser active" operation.
<b>Call</b>	<code>set_laser_pulses( HalfPeriod, PulseLength )</code>
<b>Parameters</b>	<p>HalfPeriod      <i>Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i></p> <p>PulseLength     <i>Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i></p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>By the <code>HalfPeriod</code> parameter, <i>half</i> the period duration is specified, see <a href="#">figure 48</a> and <a href="#">figure 49</a>.</li> <li>If <code>HalfPeriod = 0</code> and/or <code>PulseLength = 0</code>, no laser signals are outputted.</li> <li>With <code>PulseLength ≥ (2 × HalfPeriod)</code>, the laser remains on all the time.</li> <li>The signal level is defined by <a href="#">set_laser_control</a>.</li> <li><a href="#">set_laser_pulses</a> is also available as the control command <a href="#">set_laser_pulses_ctrl</a>.</li> <li><a href="#">set_laser_pulses</a> is largely identical to the RTC4 command <a href="#">set_laser_timing</a>, but has less parameters.</li> </ul>
RTC4→RTC6	New command.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified Power values for <code>HalfPeriod</code> and <code>PulseLength</code> by 8. The allowed value ranges decrease accordingly.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_laser_pulses_ctrl</a> , <a href="#">set_laser_timing</a>



<b>Ctrl Command</b>	<b>set_laser_pulses_ctrl</b>				
<b>Function</b>	Like <b>set_laser_pulses</b> , but a control command.				
<b>Call</b>	<code>set_laser_pulses_ctrl( HalfPeriod, PulseLength )</code>				
<b>Parameters</b>	<table> <tr> <td>HalfPeriod</td> <td><i>Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i></td> </tr> <tr> <td>PulseLength</td> <td>Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i></td> </tr> </table>	HalfPeriod	<i>Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i>	PulseLength	Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i>
HalfPeriod	<i>Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i>				
PulseLength	Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs. Allowed value range: [0...(2<sup>32</sup>-1)].</i>				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <b>set_laser_pulses</b>.</li> </ul>				
RTC4→RTC6	New command.  <b>In RTC4 Compatibility Mode:</b> as <b>set_laser_pulses</b> .				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<b>set_laser_pulses</b> , <b>set_laser_timing</b>				



<b>Delayed Short List Command</b>	<b>set_laser_timing</b>	
<b>Function</b>	Defines the output period and the pulse lengths for the laser signals LASER1 and LASER2 for "laser active" operation.	
<b>Call</b>	set_laser_timing( HalfPeriod, PulseLength1, (PulseLength2), TimeBase )	
<b>Parameters</b>	HalfPeriod	<p>Half of the output period. In bits. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>In <b>RTC6 Standard Mode</b>: <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...(2<sup>32</sup>-1)].</li> <li>In <b>RTC4 Compatibility Mode</b>: <i>1 bit equals 1/8 µs or 1 µs</i>, depending on the selected clock frequency. With respect to the specified TimeBase, the value is converted to an integer-multiple of 1/64 µs. The allowed range is correspondingly smaller.</li> </ul>
	PulseLength1	<p>Pulse lengths of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>In <b>RTC6 Standard Mode</b>: <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...(2<sup>32</sup>-1)].</li> <li>In <b>RTC4 Compatibility Mode</b>: <i>1 bit equals 1/8 µs or 1 µs</i>, depending on the selected clock frequency. With respect to the specified TimeBase, the value is converted to an integer-multiple of 1/64 µs. The allowed range is correspondingly smaller.</li> </ul>
	(PulseLength2)	<p>Value is not used. (As an unsigned 32-bit value.)</p> <p>With the RTC6, the pulse lengths of laser signals LASER1 and LASER2 are always identical and are defined by PulseLength1.</p>
	TimeBase	<p>As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>In <b>RTC6 Standard Mode</b>, the value is ignored and the clock frequency is fixed at 64 MHz. <i>1 bit equals 1/64 µs.</i></li> <li>In <b>RTC4 Compatibility Mode</b>, the TimeBase value is handled as follows: = 0: sets the clock frequency to 1 MHz. <i>1 bit equals 1 µs.</i> ≠ 0: sets the clock frequency to 8 MHz. <i>1 bit equals 1/8 µs.</i></li> </ul>



<b>Delayed Short List Command</b>	<b>set_laser_timing</b>
Comments	<ul style="list-style-type: none"> <li>In <b>RTC6 Standard Mode</b>, <b>set_laser_timing</b> is synonymous with <b>set_laser_pulses</b>. See comments there (on <code>HalfPeriod</code>, <code>PulseLength</code>).</li> <li>The clock frequency settings apply <i>only</i> for the parameters of <b>set_laser_timing</b>.</li> <li>In <b>RTC4 Compatibility Mode</b>, SCANLAB generally recommends setting the clock frequency to 8 MHz. A clock frequency of 1 MHz should only be set, if absolutely necessary.</li> <li>Observe also the notes in <b>Chapter 7.4 "Laser Control"</b>, page 173.</li> <li>In <b>RTC4 Compatibility Mode</b>, in Laser Mode 4 and Laser Mode 6, the time base for signals LASER1 and LASER2 is independently of <code>TimeBase</code> always <math>1/8 \mu\text{s}</math>.</li> </ul>
RTC4→RTC6	<ul style="list-style-type: none"> <li>With the RTC6, the pulse lengths of laser signals LASER1 and LASER2 are always identical.</li> <li>Clock frequencies: <ul style="list-style-type: none"> <li>In <b>RTC6 Standard Mode</b>: 64 MHz, fixed</li> <li>In <b>RTC4 Compatibility Mode</b>: 1 MHz or 8 MHz</li> </ul> </li> </ul>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_laser_pulses_ctrl</b> , <b>set_laser_pulses</b> , <b>set_laser_mode</b> , <b>set_firstpulse_killer</b> , <b>set_firstpulse_killer_list</b> , <b>set_standby</b> , <b>set_standby_list</b>

<b>Undelayed Short List Command</b>	<b>set_list_jump</b>
Function	Produces an unconditional jump to the specified address within the list memory upon execution. The next command there is executed immediately without delay.
Call	<code>set_list_jump( Pos )</code>
Parameters	Pos      Absolute jump address [0...(2 <sup>23</sup> -1)]. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li><b>set_list_jump</b> is synonymous with <b>list_jump_pos</b>. See the comments there.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>list_jump_pos</b>



<b>Delayed Short List Command</b>	<b>set_mark_speed</b>
Function	Sets the mark speed for the vector commands and arc commands.
Call	<code>set_mark_speed( Speed )</code>
Parameters	Speed      Marking speed. In bits/ms. As a 64-bit IEEE floating point value. Allowed value range: [1.6...800000.0].
Comments	<ul style="list-style-type: none"> <li>By default a mark speed of 1,000 bits/ms is preset.</li> <li>The specified mark speed is used for all mark commands and arc commands until a new value is specified.</li> <li>The actual mark speed <math>v_{mark}</math> in the image plane in m/s is derived from the specified Speed value [bits/ms] and the calibration factor <math>K</math> [Bits/mm] as follows:</li> </ul> $v_{mark} = \text{Speed} / K$ <p>The calibration factor <math>K</math> can be queried from the correction table by <a href="#">get_table_para</a> or <a href="#">get_head_para</a>.</p> <ul style="list-style-type: none"> <li><code>set_mark_speed</code> is also available as control command <a href="#">set_mark_speed_ctrl</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified Speed value by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">mark_abs</a> , <a href="#">mark_rel</a> , <a href="#">set_jump_speed</a> , <a href="#">set_mark_speed_ctrl</a>



<b>Ctrl Command</b>	<b>set_mark_speed_ctrl</b>
<b>Function</b>	Like <b>set_mark_speed</b> , but a control command.
<b>Call</b>	<code>set_mark_speed_ctrl( Speed )</code>
<b>Parameters</b>	<p>Speed      Marking speed. In bits/ms.            As a 64-bit IEEE floating point value.            Allowed value range: [1.6...800,000.0].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_mark_speed_ctrl</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:               <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> </ul> </li> <li>• <b>set_mark_speed_ctrl</b> is even executed, if:               <ul style="list-style-type: none"> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> </ul>
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> : as <b>set_mark_speed</b> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_mark_speed</b> , <b>set_jump_speed</b> , <b>set_jump_speed_ctrl</b>



Ctrl Command	set_matrix
Function	<p>Defines the coefficients of the</p> <ul style="list-style-type: none"> <li>General transformation matrix <math>M_T</math> for coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 210</a></li> <li>Global transformation matrix in virtual image field, see <a href="#">Section "Coordinate Transformations in the Virtual Image Field", page 158</a></li> </ul>
Call	set_matrix( HeadNo, M11, M12, M21, M22, at_once )
Parameters	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>= 1: The definition only affects the <i>first</i> scan head connector.</li> <li>= 2: The definition only affects the <i>second</i> scan head connector.</li> <li>= 0, 3: The definition affects <i>both</i> scan head connectors.</li> <li>= 4: The definition affects the virtual image field (see also comments). Only the three least significant bits are evaluated.</li> </ul> <p>M11      Matrix coefficient of the general transformation matrix <math>M_T</math>. As a 64-bit IEEE floating point value. Allowed value range: [-50...+50] in the real image field and [-2.0...+2.0] in the virtual image field. With invalid value, <b>set_matrix</b> is ignored.</p> <p>M12      Like M11 (analogously).</p> <p>M21      Like M11 (analogously).</p> <p>M22      Like M11 (analogously).</p> <p>at_once      Defines when the defined transformation becomes effective. As an unsigned 32-bit value.</p> <p>For HeadNo = 0...3, the following applies:</p> <ul style="list-style-type: none"> <li>= 0: The new total transformation (total matrix and offset) is only calculated when the next list command is executed and applied to the position which is current at that time.</li> <li>= 1: The new total transformation is calculated immediately (or prior the next list command, if a list is currently <b>BUSY</b> or the board is <b>INTERNAL-BUSY</b>) and applied to the current position. The "laser active" laser control signals are switched off in advance.</li> <li>= 2: The new total transformation (total matrix and offset) is only calculated and applied to the current position when the next <b>jump_abs</b>, <b>jump_rel</b>, <b>goto_xy</b> or <b>goto_xyz</b> is executed.</li> <li>= 3: Like at_once = 1, but the laser control signals remain unaffected.</li> <li>&gt; 3: Like at_once = 2.</li> </ul> <p>For HeadNo = 4, the following applies:</p> <ul style="list-style-type: none"> <li>= 0: Only saved. Is applied at the next <b>Processing-on-the-fly session</b> start to the current position.</li> <li>= 1: Is applied immediately to the current position if currently no list is <b>BUSY</b> or no board is <b>INTERNAL-BUSY</b>. Otherwise like 0.</li> <li>= 2, 3: Like 0.</li> </ul>



Ctrl Command	<a href="#">set_matrix</a>
Comments	<ul style="list-style-type: none"> <li>Up to DLL 613, OUT 613 the following applies for HeadNo = 4:           <ul style="list-style-type: none"> <li>The global coordinate transformations are only available during a <a href="#">Processing-on-the-fly session</a> that has been started by <a href="#">set_fly_2d</a>.</li> <li><code>at_once</code> is ignored</li> </ul> </li> <li>As of DLL 614, OUT 614 the following applies for HeadNo = 4:           <ul style="list-style-type: none"> <li>The global coordinate transformations are generally available, even outside a <a href="#">Processing-on-the-fly session</a>.</li> <li>Coordinate transformations that have been merely saved, are applied at the next <a href="#">Processing-on-the-fly session</a> start.</li> </ul> </li> </ul>
RTC4→RTC6	Unchanged first functionality (transformation matrix definition), however: <ul style="list-style-type: none"> <li>The parameters HeadNo and <code>at_once</code> are new.</li> <li>Reduced value range for coefficients.</li> <li>See also <a href="#">Section "Notes for RTC4 Users", page 213</a>.</li> </ul>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 614, OUT 614: HeadNo = 4 generally available.
References	<a href="#">set_matrix_list</a> , <a href="#">set_angle</a> , <a href="#">set_offset</a> , <a href="#">set_scale</a>



<b>Variable List Command</b>	<b>set_matrix_list</b>
<b>Function</b>	Sets <i>one</i> of the four coefficients of the general transformation matrix $M_T$ during execution of a list, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 210</a> .
<b>Call</b>	<code>set_matrix_list( HeadNo, Ind1, Ind2, Mij, at_once )</code>
<b>Parameters</b>	<p>HeadNo      Like <a href="#">set_matrix</a>.</p> <p>Ind1        Row index and column index of the matrix coefficient to be changed. As an unsigned 32-bit value.</p> <p>Ind2        Allowed values: [Index uneven: 1, Index even: 2].</p> <p>Mij         Matrix coefficient. As a 64-bit IEEE floating point value. Allowed value range: [-50...+50]. If the parameter is set to an invalid value, <a href="#">set_matrix_list</a> is replaced by a <a href="#">list_nop</a>.</p> <p>at_once      Determines when the defined transformation becomes effective. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>= 0: The transformation settings are only accumulated and intermediately stored, but the transformation is not processed as long as this is not activated by another coordinate transformation (for example, by a list command with <code>at_once = 1</code> or a corresponding control command).</li> <li>= 1: The transformation is immediately calculated (including all transformation settings that were accumulated until then) and processed prior to the next list command. The "laser active" laser control signals are switched off in advance if necessary.</li> <li>= 2: The transformation settings are only accumulated and intermediately stored (as with <code>at_once = 0</code>). However, The transformation is immediately calculated (including all transformation settings that were accumulated and intermediately stored until then) and applied to the current position when the next <a href="#">jump_abs</a>, <a href="#">jump_rel</a>, <a href="#">goto_xy</a> or <a href="#">goto_xyz</a> is executed.</li> <li>= 3: As with <code>at_once = 1</code>, but the laser control signals remain unaffected.</li> <li>&gt; 3: See <code>at_once = 2</code>.</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <a href="#">set_matrix_list</a> only allows changing <i>one</i> of the four coefficients at a time. To change several coefficients during execution of a list, <a href="#">set_matrix_list</a> has to be called repeatedly. Here, we recommend making the first calls with <code>at_once = 0</code> and only the last call with <code>at_once = 1</code>.</li> <li>• See <a href="#">Chapter 8.2 "Coordinate Transformations", page 210</a>.</li> </ul>
RTC4→RTC6	See <a href="#">set_matrix</a> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_matrix</a>

<b>Ctrl Command</b>	<b>set_max_counts</b>
<b>Function</b>	Defines the maximum number of external starts.
<b>Call</b>	<code>set_max_counts( Counts )</code>
<b>Parameters</b>	Counts      Maximum number of external starts. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].
<b>Comments</b>	<ul style="list-style-type: none"> <li>When the specified number of external starts has been reached, the external start input is disabled (see <a href="#">set_control_mode</a>, Bit #0 = 0).</li> <li>If Counts = 0, the number of external starts is unlimited.</li> <li>When the RTC6 is initialized (by <a href="#">load_program_file</a>), Counts is set to 0.</li> <li>The current number of external starts can be read from the corresponding internal counter by <a href="#">get_counts</a>. The counter can be reset by <a href="#">set_control_mode</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_counts</a> , <a href="#">set_control_mode</a>

<b>Ctrl Command</b>	<b>set_mcbsp_freq</b>
<b>Function</b>	Sets the transmission frequency of the <a href="#">McBSP interface</a> .
<b>Call</b>	<code>mcbsp_freq = set_mcbsp_freq( Freq )</code>
<b>Parameters</b>	Freq      Desired transmission frequency of the <a href="#">McBSP interface</a> in Hz. As an unsigned 32-bit value. Allowed value range: [4000000...16000000] (4...16 MHz). Out-of-range values are clipped to the boundary values. Out-of-range values cause the <a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a> to be generated.
<b>Result</b>	The actually set frequency in Hz. As an unsigned 32-bit value. With overflowing values 0 is returned.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The default transmission frequency (after initialization) is 8 MHz (Freq = 8,000,000).</li> <li>Because not every arbitrary frequency can be implemented, <a href="#">set_mcbsp_freq</a> returns the actually set frequency. Example: <code>mcbsp_freq = set_mcbsp_freq(7,000,000);</code> returns <code>mcbsp_freq = 7,200,000</code>.</li> <li>The new transmission frequency only becomes effective when you re-initialize the <a href="#">McBSP interface</a> by <a href="#">mcbsp_init</a>.</li> <li>The signals and operating conditions of the <a href="#">McBSP interface</a> are presented in the <a href="#">Chapter 4.6.6 "McBSP/ANALOG Socket Connector"</a>, page 73.</li> <li>Note: The receiving frequency is exclusively determined by the incoming clock pulses and has a maximum limit of 16 MHz.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">mcbsp_init</a> , <a href="#">set_mcbsp_out</a> , <a href="#">set_mcbsp_out_ptr</a>



<b>Ctrl Command</b>	<b>set_mcbsp_global_matrix</b>
<b>Function</b>	Activates matrix correction for “Global Online Positioning” by the <b>McBSP interface</b> .
<b>Call</b>	<code>set_mcbsp_global_matrix()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 8.3.2 ““Global Online Positioning””, page 217</a>.</li> <li>• A matrix correction cannot be used in conjunction with offset and/or rotation corrections. Any such already-activated options gets deactivated by <b>set_mcbsp_global_matrix</b>. Subsequent activation of other options (by <b>set_mcbsp_global_x</b>, <b>set_mcbsp_global_y</b> or <b>set_mcbsp_global_rot</b>) deactivates the matrix option.</li> <li>• The following restrictions apply to the matrix coefficients transferred over the <b>McBSP interface</b> (as with <b>set_matrix</b>(<code>HeadNo = 4, ...</code>)): <ul style="list-style-type: none"> <li>– The allowed value range for matrix coefficients is <math>[-2.0 \dots +2.0]</math>.</li> <li>– Transferred coefficients exceeding this range are ignored.</li> </ul> </li> <li>• Users must individually supply as input value <math>M_{in}</math> to the <b>McBSP interface</b> each matrix coefficient <math>M_{ij}</math> of the transformation matrix <math>M_T</math> as a normalized integer with associated indices <math>i</math> and <math>j</math> as follows:  <math display="block">M_{in} = (\text{integer}( M_{ij} * 2^{28} ) &lt;&lt; 2) + (i &lt;&lt; 1) + j</math> with <math>M_T = \{ M00, M01, M10, M11 \} = \{ m_{11}, m_{12}, m_{21}, m_{22} \}</math>.  Conversely, the RTC6 determines a coefficient from the input value as follows:  <math display="block">M_T[ M_{in} \&amp; 0x3 ] = ( M_{in} &gt;&gt; 2 ) / 2^{28}</math>. </li> <li>• The coefficients are transferred to internal memory location 1 and can be checked there by querying with <code>read_mcbsp(1)</code>.</li> <li>• The <b>McBSP interface</b> cannot be simultaneously used for an <b>Online Positioning</b> and Processing-on-the-fly applications.  See also <a href="#">Section “Notes”, page 216</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#"><b>set_mcbsp_global_matrix_list</b></a> , <a href="#"><b>set_mcbsp_matrix</b></a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_global_matrix_list</b>
<b>Function</b>	Same as <a href="#"><b>set_mcbsp_global_matrix</b></a> , but a list command.
<b>Call</b>	<code>set_mcbsp_global_matrix_list()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#"><b>set_mcbsp_global_matrix</b></a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#"><b>set_mcbsp_global_matrix</b></a>



<b>Ctrl Command</b>	<b>set_mcbsp_global_rot</b>
<b>Function</b>	Activates or deactivates rotation correction for “ <a href="#">Global Online Positioning</a> ” by the <a href="#">McBSP interface</a> .
<b>Call</b>	<code>set_mcbsp_global_rot( Resolution )</code>
<b>Parameters</b>	<p>Resolution As a 64-bit IEEE floating point value.  <math>2^{-26} &lt; \text{Resolution} &lt; 2^{26}</math>: scaling factor (correction is activated), otherwise: correction is deactivated.</p> <p>Resolution = McBSP bits per full circle.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 8.3.2 ““Global Online Positioning””, page 217</a>.</li> <li>With an McBSP rotation correction input value of <math>\text{Rot}_{\text{in}}</math>, <a href="#">set_mcbsp_global_rot</a> functions like <code>set_angle(4, Angle × 360°, ...)</code>, whereby Angle (in full circles) = <math>\text{Rot}_{\text{in}} / \text{Resolution}</math>. Only Angle values in the range [0.0...+20.0 full circles] are allowed.</li> <li>The <a href="#">McBSP interface</a> cannot be simultaneously used for an <a href="#">Online Positioning</a> and Processing-on-the-fly applications.</li> </ul> <p>See also <a href="#">Section “Notes”, page 216</a>.</p>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">set_mcbsp_global_rot_list</a> , <a href="#">set_mcbsp_rot</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_global_rot_list</b>
<b>Function</b>	Same as <a href="#">set_mcbsp_global_rot</a> , but a list command.
<b>Call</b>	<code>set_mcbsp_global_rot_list( Resolution )</code>
<b>Parameters</b>	Resolution See <a href="#">set_mcbsp_global_rot</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_mcbsp_global_rot</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">set_mcbsp_global_rot</a>



<b>Ctrl Command</b>	<b>set_mcbsp_global_x</b>
<b>Function</b>	Activates or deactivates X offset correction for “Global Online Positioning” by the <b>McBSP interface</b> .
<b>Call</b>	set_mcbsp_global_x( Scale )
<b>Parameters</b>	Scale      As a 64-bit IEEE floating point value. $2^{-26} < \text{Scale} < 2^{26}$ : scaling factor (correction is activated), otherwise: correction is deactivated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 8.3.2 “Global Online Positioning”</a>, page 217.</li> <li>With an McBSP input value of <math>X_{in}</math> for the X offset correction, set_mcbsp_global_x functions like set_offset_xyz(4, XOffset, ... ), whereby <math>X_{Offset}</math> (in bits) = Scale <math>\times X_{in}</math>.  XOffset values outside of [-524,288...+524,287] are clipped to the boundary value as long as Scale <math>\times X_{in}</math> does not exceed the value range <math>\pm 2^{31}</math>.</li> <li>The <b>McBSP interface</b> cannot be simultaneously used for an <b>Online Positioning</b> and Processing-on-the-fly applications.  See also <a href="#">Section “Notes”</a>, page 216.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">set_mcbsp_global_x_list</a> , <a href="#">set_mcbsp_x</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_global_x_list</b>
<b>Function</b>	Same as <a href="#">set_mcbsp_global_x</a> , but a list command.
<b>Call</b>	set_mcbsp_global_x_list( Scale )
<b>Parameters</b>	Scale      See <a href="#">set_mcbsp_global_x</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_mcbsp_global_x</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">set_mcbsp_global_x</a>



<b>Ctrl Command</b>	<b>set_mcbsp_global_y</b>
<b>Function</b>	Activates or deactivates Y offset correction for “Global Online Positioning” by the McBSP interface.
<b>Call</b>	set_mcbsp_global_y( Scale )
<b>Parameters</b>	Scale      As a 64-bit IEEE floating point value. $2^{-26} < \text{Scale} < 2^{26}$ : scaling factor (correction is activated), otherwise: correction is deactivated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 8.3.2 “Global Online Positioning”</a>, page 217.</li> <li>With an McBSP input value of <math>Y_{in}</math> for the Y offset correction, set_mcbsp_global_y functions like set_offset_xyz(4, ..., YOffset, ...), whereby <math>Y_{Offset}</math> (in bits) = Scale <math>\times Y_{in}</math>.  <math>Y_{Offset}</math> values outside of <math>[-524,288...+524,287]</math> are clipped to the boundary value as long as Scale <math>\times Y_{in}</math> does not exceed the value range <math>\pm 2^{31}</math>.</li> <li>The <a href="#">McBSP interface</a> cannot be simultaneously used for an <a href="#">Online Positioning</a> and Processing-on-the-fly applications.  See also <a href="#">Section “Notes”</a>, page 216.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">set_mcbsp_global_y_list</a> , <a href="#">set_mcbsp_y</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_global_y_list</b>
<b>Function</b>	Same as <a href="#">set_mcbsp_global_y</a> , but a list command.
<b>Call</b>	set_mcbsp_global_y_list( Scale )
<b>Parameters</b>	Scale      See <a href="#">set_mcbsp_global_y</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_mcbsp_global_y</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">set_mcbsp_global_y</a>

<b>Ctrl Command</b>	<b>set_mcbsp_in</b>
<b>Function</b>	Activates Processing-on-the-fly correction for compensation of a workpiece or scan system movement (based on position values transferred to the RTC6 via the <b>McBSP interface</b> ). The <b>McBSP interface</b> can also be used for inputting other desired signals.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>set_mcbsp_in</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>set_mcbsp_in( Mode, Scale )</code>
<b>Parameters</b>	<p>Mode      As an unsigned 32-bit value.          Allowed values:            = 0:    Processing-on-the-fly correction is switched off.            = 1:    Compensation of linear movement in the X direction.            = 2:    Compensation of linear movement in the Y direction.            = 3:    Compensation of linear movement in the X and Y directions.            = 4:    Compensation of rotary movement.            = 5:    Processing-on-the-fly correction is switched off.                  • Mode = 0...5: All <b>McBSP</b> input values are alternatingly copied to internal memory locations 1 and 2.                  • Mode = 1...5 (but not for Mode = 0): <b>McBSP</b> input values coded with "Bit #31 = 0" is additionally copied to internal memory location 0 and those with "Bit #31 = 1" to internal memory location 3.                  • Mode = 1...4: Values copied to internal memory location 0 are applied for Processing-on-the-fly correction.</p> <p>Scale      Scaling factor or rotation resolution.          As a 64-bit IEEE floating point value.                  • Mode = 1...3: scaling factor in (RTC6)bits/(McBSP)bit.          Allowed value range: <math>1/256 \leq  \text{Scale}  \leq 16000.0</math>          (if Mode = 3, Scale applies to both axes).                  • Mode = 4: number of steps (counts) per revolution.          Allowed value range: <math> \text{Scale}  &gt; 100.0</math>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>You can query the internal memory locations at any time by <b>read_mcbsp</b>.</li> <li>For Processing-on-the-fly correction and determination of the scaling factor, see <b>Chapter 8.6 "Processing-on-the-fly", page 227</b>.</li> <li>The various Processing-on-the-fly corrections cannot be arbitrarily combined, see <b>Section "Overview", page 227</b>.</li> </ul>



<b>Ctrl Command</b>	<b>set_mcbsp_in</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Correction", page 236</a>.</li> <li>15 bits per axis (with sign) are effectively available for 2D correction (<code>Mode = 3</code>), whereby the x value is in the lower 16 bits of the "Bit #31 = 0"-coded <b>McBSP</b> input value and the y value in the upper 16 bits.</li> <li>The <b>McBSP interface</b> cannot be simultaneously used for both Processing-on-the-fly applications and <b>Online Positioning</b>. See also <a href="#">Section "Notes", page 216</a>.</li> <li>The <b>McBSP interface</b> always ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>, so available data is not transmitted (see <a href="#">page 75</a>).</li> <li>If <code>Mode &gt; 5</code>, <b>set_mcbsp_in</b> is not executed (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>If an unallowed <code>Scale</code> parameter value is supplied (for example, <code>Scale = 0</code>), <b>set_mcbsp_in</b> behaves as with <code>Mode = 0</code>.</li> <li>Processing-on-the-fly corrections are compatible with the settings from <a href="#">Chapter 8.6.12 ""Fly Extension" Commands", page 245</a> as of RTC6 Software Package V1.6.1. They can also be subsequently overwritten, for example, with encoder-based Processing-on-the-fly corrections. The <b>McBSP</b> values are then not used for Processing-on-the-fly corrections. However, the <b>McBSP</b> memory transfer described above cannot be changed. If <b>McBSP</b>-based Processing-on-the-fly corrections other than those set according to the <code>Mode</code> are set, the user is responsible for transferring the <b>McBSP</b> data in the appropriate format. An encoder-based Processing-on-the-fly correction activated for the z axis is retained.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Last change version DLL 617, OUT 617: compatibility with <a href="#">"Fly Extension" Commands</a> .
References	<a href="#">set_mcbsp_in_list</a>



<b>Normal List Command</b>	<b>set_mcbsp_in_list</b>				
<b>Function</b>	Like <a href="#">set_mcbsp_in</a> , but a list command.				
<b>Restriction</b>	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <b>set_mcbsp_in_list</b> terminates the Processing-on-the-fly process (even though it could not have been activated).				
<b>Call</b>	<code>set_mcbsp_in_list( Mode, Scale )</code>				
<b>Parameters</b>	<table> <tr> <td>Mode</td> <td>See <a href="#">set_mcbsp_in</a>.</td> </tr> <tr> <td>Scale</td> <td>See <a href="#">set_mcbsp_in</a>.</td> </tr> </table>	Mode	See <a href="#">set_mcbsp_in</a> .	Scale	See <a href="#">set_mcbsp_in</a> .
Mode	See <a href="#">set_mcbsp_in</a> .				
Scale	See <a href="#">set_mcbsp_in</a> .				
<b>Comments</b>	<ul style="list-style-type: none"> <li>If <code>Mode &gt; 5</code>, then <b>set_mcbsp_in_list</b> is replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>See <a href="#">set_mcbsp_in</a>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Last change version DLL 617, OUT 617: compatibility with " <a href="#">Fly Extension</a> " Commands.				
References	<a href="#">set_mcbsp_in</a>				



<b>Ctrl Command</b>	<b>set_mcbsp_matrix</b>
<b>Function</b>	Activates matrix correction for “Local Online Positioning” by the McBSP interface.
<b>Call</b>	<code>set_mcbsp_matrix()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For “Local Online Positioning”, see Chapter 8.3.1 ““Local Online Positioning””, page 214.</li> <li>Matrix corrections cannot be used in conjunction with offset and/or rotation corrections. Any such already-activated options gets deactivated by <code>set_mcbsp_matrix</code>. Subsequent activation of other options (by <code>set_mcbsp_x</code>, <code>set_mcbsp_y</code> or <code>set_mcbsp_rot</code>) deactivates the matrix correction.</li> <li>The following restrictions apply to the matrix coefficients transferred over the <b>McBSP interface</b> (as with <code>set_matrix</code>):           <p>The allowed value range for matrix coefficients is [-50...+50]. Transferred coefficients exceeding this range are ignored.</p> </li> <li>You must individually supply as input value <math>M_{in}</math> to the <b>McBSP interface</b> each matrix coefficient <math>M_{ij}</math> of the transformation matrix <math>M_T</math> as a normalized integer with associated indices <math>i</math> and <math>j</math> as follows:</li> <p><math>M_{in} = (\text{integer}(M_{ij} * 2^{24}) &lt;&lt; 2) + (i &lt;&lt; 1) + j</math> with <math>M_T = \{ M_{00}, M_{01}, M_{10}, M_{11} \} = \{ m_{11}, m_{12}, m_{21}, m_{22} \}</math>.</p> <p>Conversely, the RTC6 determines a coefficient from the input value as follows: <math>M_T[M_{in} \&amp; 0x3] = (M_{in} &gt;&gt; 2) / 2^{24}</math>.</p> <li>You must separately fetch each transferred matrix coefficient by <code>apply_mcbsp</code> or <code>apply_mcbsp_list</code>. We recommend fetching the first coefficient with <code>at_once = 0</code> and only the last one with <code>at_once &gt; 0</code>.</li> <li>The coefficients get transferred to internal memory location 1 and can be checked there by querying with <code>read_mcbsp(1)</code>.</li> <li>The <b>McBSP interface</b> cannot be simultaneously used for an <b>Online Positioning</b> and Processing-on-the-fly applications.</li> </ul> <p>See also Section “Notes”, page 216.</p>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mcbsp_matrix_list</a>



<b>Undelayed Short List Command</b>	<b>set_mcbsp_matrix_list</b>
Function	Like <b>set_mcbsp_matrix</b> , but a list command.
Call	set_mcbsp_matrix_list()
Comments	• See <b>set_mcbsp_matrix</b> .
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_mcbsp_matrix</b>



<b>Undelayed Short List Command</b>	<b>set_mcbsp_out</b>
<b>Function</b>	Defines two signal types for output at the <b>McBSP interface</b> , see also <b>Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73.</b>
<b>Call</b>	<code>set_mcbsp_out( Signal1, Signal2 )</code>
<b>Parameters</b>	Signal1      To-be-outputted signal type. As an unsigned 32-bit value.
	Signal2      To-be-outputted signal type. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The selectable signal types are identical to those of the <b>set_trigger</b> command (refer to the comments there for the allowed value range, signal types and other information). If the value for Signal1/Signal2 is unallowed, then <b>set_mcbsp_out</b> is replaced by <b>list_nop (get_last_error return code RTC6_PARAM_ERROR)</b>.</li> <li>Both selected data signals are continuously transmitted (once per <math>10 \mu\text{s}</math> cycle).</li> <li>Only 16-bit portions of the selected data signals are packed into a common 32-bit data word for output. Signal1 is the lower half and Signal2 the upper half of this 32-bit data word. <ul style="list-style-type: none"> <li>Only RTC4-compatible Bit #4...Bit #19 of the sample values and status values returned by the scan system (Signal1, Signal2 = 1...23, 25...30) are outputted. The least significant 4 bits and any exceeding bits (in the virtual image field) are ignored.</li> <li>For the other data types (Signal1, Signal2 = 0, 24, 31...57), the least significant 16 bits are outputted and the upper bits are ignored.</li> <li>McBSP_Output = (( Out2 &amp; 0x0000FFFF ) &lt;&lt; 16 )   ( Out1 &amp; 0x0000FFFF );</li> </ul> </li> <li>Transmitted values are those of the preceding clock cycle: data is processed at the end of a cycle and transmitted at the beginning of the next clock cycle at the set transmission frequency (see <b>set_mcbsp_freq</b>).</li> <li>The signals and operating conditions of the <b>McBSP interface</b> are presented in <b>Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73.</b></li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>read_mcbsp, mcbsp_init, set_mcbsp_freq, set_mcbsp_out_ptr, set_trigger</b>



<b>Ctrl Command</b>	<b>set_mcbsp_out_ptr</b>
<b>Function</b>	Defines a list of up to 8 signal types for output at the <b>McBSP interface</b> , see also <b>Chapter 4.6.6 "McBSP/ANALOG Socket Connector", page 73.</b>
<b>Call</b>	set_mcbsp_out_ptr( Number, SignalPtr )
<b>Parameters</b>	<p>Number      As an unsigned 32-bit value.            Allowed value range: [0...8].            = 1...8: Number of signal types to be outputted.            = 0      Output at the <b>McBSP interface</b> occurs in accordance with the pre-defined settings or as specified by a prior <b>set_mcbsp_out</b>.</p> <p>SignalPtr    Pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit or 64-bit value) to an array of Number unsigned 32-bit values, where the to-be-outputted Number signal type numbers are specified.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The memory area for the SignalPtr array must be provided by the user program.</li> <li>If Number &gt; 8 and/or SignalPtr = <b>NULL</b>, <b>set_mcbsp_out_ptr</b> is not executed (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>The selectable signal types are identical to those of the <b>set_trigger</b> command (refer to the comments there for the allowed value range, signal types and other information).</li> <li>The up to 8 selected data types are outputted sequentially (one data type per 10 µs clock period). Each individual signal belongs to a different clock cycle. Transmitted values are always from the previous clock cycle. The list is repeated until the output is switched off or replaced by another list.</li> <li>When outputting, each signal value is supplemented by the corresponding signal type number: the signal type number gets inserted into the lowest byte of the 32-bit data word. The actual signal value therefore gets shifted 8 bits to the left, whereby all bits above the 24th get truncated (overflow, no clipping, relevant only for signal types 24, 31 and 37...57):            32-bit output value = (signal value &lt;&lt; 8)   (signal type number &amp; 0xFF).</li> <li>The signals and operating conditions of the <b>McBSP interface</b> are presented in the <b>Section "McBSP Interface", page 73.</b></li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_mcbsp_out, set_trigger</b>



<b>Ctrl Command</b>	<b>set_mcbsp_rot</b>
<b>Function</b>	Activates or deactivates rotation correction for “ <a href="#">Local Online Positioning</a> ” by the <a href="#">McBSP interface</a> .
<b>Call</b>	<code>set_mcbsp_rot( Resolution )</code>
<b>Parameters</b>	<p>Resolution As a 64-bit IEEE floating point value.</p> $2^{-26} < \text{Resolution} < 2^{26}$ : scaling factor (correction is activated), otherwise: correction is deactivated. <p>Resolution = <a href="#">McBSP</a> bits per full circle.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For “<a href="#">Local Online Positioning</a>”, see <a href="#">Chapter 8.3.1 ““Local Online Positioning””, page 214</a>.</li> <li>For an <a href="#">McBSP</a> rotation correction input value of <math>\text{Rot}_{\text{in}}</math>, the command <a href="#">apply_mcbsp</a> functions like <code>set_angle( ..., Angle × 360°, ... )</code>, whereby (FC = full circle)</li> </ul> $\text{Angle (in FC)} = \text{Rot}_{\text{in}} / \text{Resolution}$ <p>Only <a href="#">Angle</a> values in the range [0.0...+20.0 FC] are allowed.</p> <ul style="list-style-type: none"> <li>The <a href="#">McBSP interface</a> cannot be simultaneously used for an <a href="#">Online Positioning</a> and Processing-on-the-fly applications.</li> </ul> <p>See also <a href="#">Section “Notes”, page 216</a>.</p>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_angle</a> , <a href="#">set_mcbsp_rot_list</a> , <a href="#">set_mcbsp_x</a> , <a href="#">set_mcbsp_y</a> , <a href="#">apply_mcbsp</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_rot_list</b>
<b>Function</b>	Like <a href="#">set_mcbsp_rot</a> , but a list command.
<b>Call</b>	<code>set_mcbsp_rot_list( Resolution )</code>
<b>Parameters</b>	Resolution See <a href="#">set_mcbsp_rot</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_mcbsp_rot</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mcbsp_rot</a>



<b>Ctrl Command</b>	<b>set_mcbsp_x</b>
<b>Function</b>	Activates or deactivates X offset correction for “Local Online Positioning” by the <b>McBSP interface</b> .
<b>Call</b>	<code>set_mcbsp_x( Scale )</code>
<b>Parameters</b>	Scale      As a 64-bit IEEE floating point value. $2^{-26} < \text{Scale} < 2^{26}$ : scaling factor (correction is activated). Otherwise: correction is deactivated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>For “Local Online Positioning”, see <a href="#">Chapter 8.3.1 ““Local Online Positioning””, page 214</a>.</li> <li>With an <b>McBSP</b> input value of <math>X_{in}</math> for the X offset correction, the command <b>apply_mcbsp</b> functions like <code>set_offset( ..., XOffset,...)</code>, whereby  <math display="block">XOffset \text{ (in bits)} = \text{Scale} \times X_{in}</math> <math display="block">XOffset \text{ values outside of } [-524,288 \dots +524,287] \text{ are clipped to the boundaries as long as } \text{Scale} \times X_{in} \text{ does not exceed the value range } \pm 2^{31}.</math> </li> <li>The <b>McBSP interface</b> cannot be simultaneously used for an <b>Online Positioning</b> and Processing-on-the-fly applications.  See also <a href="#">Section “Notes”, page 216</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mcbsp_x_list</a> , <a href="#">set_mcbsp_y</a> , <a href="#">set_mcbsp_rot</a> , <a href="#">apply_mcbsp</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_x_list</b>
<b>Function</b>	Like <a href="#">set_mcbsp_x</a> , but a list command.
<b>Call</b>	<code>set_mcbsp_x_list( Scale )</code>
<b>Parameters</b>	Scale      See <a href="#">set_mcbsp_x</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_mcbsp_x</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mcbsp_x</a>



<b>Ctrl Command</b>	<b>set_mcbsp_y</b>
<b>Function</b>	Activates or deactivates Y offset correction for “Local Online Positioning” by the <b>McBSP interface</b> .
<b>Call</b>	<code>set_mcbsp_y( Scale )</code>
<b>Parameters</b>	Scale      As a 64-bit IEEE floating point value. $2^{-26} < \text{Scale} < 2^{26}$ : scaling factor (correction is activated). Otherwise: correction is deactivated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>For “Local Online Positioning”, see <a href="#">Chapter 8.3.1 ““Local Online Positioning””, page 214</a>.</li> <li>With an <b>McBSP</b> input value of <math>Y_{in}</math> for the Y offset correction, the command <b>apply_mcbsp</b> functions like <code>set_offset( ..., YOffset,...)</code>, whereby  <math display="block">YOffset \text{ (in bits)} = \text{Scale} \times Y_{in}</math> <math display="block">YOffset \text{ values outside of } [-524,288 \dots +524,287] \text{ are clipped to the boundaries as long as } \text{Scale} \times Y_{in} \text{ does not exceed the value range } \pm 2^{31}.</math> </li> <li>The <b>McBSP interface</b> cannot be simultaneously used for an <b>Online Positioning</b> and Processing-on-the-fly applications.  See also <a href="#">Section “Notes”, page 216</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mcbsp_y_list</a> , <a href="#">set_mcbsp_x</a> , <a href="#">set_mcbsp_rot</a> , <a href="#">apply_mcbsp</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_y_list</b>
<b>Function</b>	Like <a href="#">set_mcbsp_y</a> , but a list command.
<b>Call</b>	<code>set_mcbsp_y_list( Scale )</code>
<b>Parameters</b>	Scale      See <a href="#">set_mcbsp_y</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_mcbsp_y</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_mcbsp_y</a>



<b>Ctrl Command</b>	<b>set_multi_mcbsp_in</b>
<b>Function</b>	Activates a Processing-on-the-fly application and <b>McBSP interface</b> multi transmission with up to 8 different data types.
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>set_multi_mcbsp_in</b> switches off the Processing-on-the-fly process (even if it was never switched on).
<b>Call</b>	<code>set_multi_mcbsp_in( Ctrl, P, Mode )</code>
<b>Parameters</b>	<p>Ctrl      Control parameter for initializing or deactivating laser power variation. As an unsigned 32-bit value. = 1...6: defines which signal parameter to vary: for a description, see <b>set_auto_laser_control</b>. = 0 or &gt; 6: deactivates laser power variation (for Ctrl &gt; 6: <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</p> <p>P      Initialization value for laser power. As an unsigned 32-bit value. Allowed value range: see <b>set_auto_laser_control</b>.</p> <p>Mode      As an unsigned 32-bit value. = 0: The transmitted value is used directly. = 1: The transmitted value is multiplied by <math>P/16384</math> and then put out.</p>
	Ctrl, P and Mode are only relevant for Type 3 (= laser power) of the transmitted data word, see <b>read_multi_mcbsp</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>Up to V1.6.0, the following applies: Any other already activated <b>McBSP</b> transmission for <b>Online Positioning</b> and any other activated Processing-on-the-fly application with encoder signals or positional values is terminated first. As of V1.6.1, the following applies: Processing-on-the-fly corrections are compatible with the <b>"Fly Extension"</b> Commands. The Processing-on-the-fly corrections of the 3 axes are correspondingly overwritten. They can also be overwritten later by other corrections, for example, encoder-based Processing-on-the-fly corrections. Data types 0...2 are then not used for Processing-on-the-fly corrections. Hence, transmission of extra parameters can be combined even with encoder-based Processing-on-the-fly corrections. However, the <b>McBSP</b> memory transfer cannot be changed.</li> <li><b>set_multi_mcbsp_in</b> enables inputting by the <b>McBSP interface</b> of up to 8 different types for asynchronous transmission every 10 µs. Each 10 µs, the data is copied in accordance with its type code to separate memory, from where it can also be queried by <b>read_multi_mcbsp</b>.</li> <li>To avoid faulty sorting, no active <b>McBSP</b> transmission should be occurring when you issue <b>set_multi_mcbsp_in</b>.</li> <li>The <b>McBSP interface</b> always ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>, so available data is not transmitted (see <b>page 75</b>).</li> </ul>

Ctrl Command	set_multi_mcbsp_in
Comments (cont'd)	<ul style="list-style-type: none"> <li>For the transmission, the type assignment must be coded into the 3 least significant bits of the data word according to:           <ul style="list-style-type: none"> <li>McBSPValue = ( Value &lt;&lt; 3 )   Type</li> </ul>           The RTC6 board stores the data word according to:           <ul style="list-style-type: none"> <li>Memory[ McBSPValue &amp; 0x7 ] = (long) McBSPValue &gt;&gt; 3</li> </ul> </li> <li>For more on type assignments, see <a href="#">read_multi_mcbsp</a>.</li> <li>The remaining (most significant) 29 bits are available for the data word itself. There are no further restrictions other than the type-dependent value ranges themselves. The transmitted values are not checked with respect to their ranges. Clipping or data overflow may occur.</li> <li>The four extra parameters are not the same as the free variables, see <a href="#">Chapter 6.9.1 "Free Variables", page 124</a>, although they can be used for similar purposes. Upon program start, they are initialized with 0 and this command does not further modify them. The transmitted values merely get copied into type-sorted memory.</li> <li>If more than four <b>McBSP</b> transfers complete within a <math>10 \mu\text{s}</math> clock period, then prior values might get overwritten.</li> <li>If the <a href="#">Option Processing-on-the-fly</a> is enabled, then <b>set_multi_mcbsp_in</b> activates a Processing-on-the-fly application with positional values for the three coordinate directions x, y and z. The memory values of their respective types are initialized with 0.</li> <li>Additionally, laser power can be varied by outputting the transmitted type 3 value at the port assigned by the <code>Ctrl</code> parameter. The initialization value <code>P</code> gets put out immediately.</li> <li>For <code>Mode</code> = 0, subsequent type-3 transfers are outputted directly at the port assigned by <code>Ctrl</code>. For <code>Mode</code> = 1, the transferred value is handled as a multiplication factor in accordance with Normalization <math>1.0 = 16384</math> (14 bits). Thus, the following is put out: <math>(P \times \text{the transmitted value} / 16384)</math>. This mode is an alternative to laser power variation by "freely definable wobble shapes".</li> <li>These laser power variation method can be combined with the "vector-controlled laser control" (with parameterized commands). It cannot be combined with other "Automatic Laser Control" methods. It overwrites other variations sharing the same <code>Ctrl</code> parameter. Though identical <code>Ctrl</code> parameters are allowed, they serve no practical purpose.</li> <li>If <code>Ctrl</code> = 0, then laser power variation is switched off. The values transmitted by <b>McBSP</b> continue to be copied into type-sorted memory, but are not put out.</li> <li>Special case: if a "freely definable wobble shape" is active, then <code>P</code> is always regarded (irrespective of the <code>Mode</code> parameter) as relative laser power and multiplicatively coupled with the wobble-shape's laser power (see <a href="#">set_wobble_vector</a>). Because this wobble shape defines its own laser power (see <a href="#">set_wobble_control</a>), the command's <code>Ctrl</code> parameter has no relevance. It should be set to an invalid value (for example, with <code>Ctrl</code> = 0) to avoid doubled or meaningless output.</li> </ul>



<b>Ctrl Command</b>	<b>set_multi_mcbsp_in</b>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 617, OUT 617: compatibility with "Fly Extension" Commands.
Comments	<b>set_multi_mcbsp_in_list, read_multi_mcbsp, set_wobbel_control, set_wobbel_vector</b>



<b>Normal List Command</b>	<b>set_multi_mcbsp_in_list</b>
<b>Function</b>	Like <a href="#">set_multi_mcbsp_in</a> , but a list command.
<b>Restriction</b>	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <b>set_multi_mcbsp_in_list</b> switches off the Processing-on-the-fly process (even if it was never switched on).
<b>Call</b>	<code>set_multi_mcbsp_in_list( Ctrl, P, Mode )</code>
<b>Parameters</b>	<p>Ctrl      Like <a href="#">set_multi_mcbsp_in</a>.</p> <p>P        Like <a href="#">set_multi_mcbsp_in</a>.</p> <p>Mode     Like <a href="#">set_multi_mcbsp_in</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">set_multi_mcbsp_in</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 617, OUT 617: compatibility with <a href="#">"Fly Extension" Commands</a> .
<b>References</b>	<a href="#">set_multi_mcbsp_in</a> , <a href="#">read_multi_mcbsp</a> , <a href="#">set_wobbel_control</a> , <a href="#">set_wobbel_vector</a>



<b>Variable List Command</b>	<b>set_n_pixel</b>
<b>Function</b>	In pixel output mode, executes PortOutValue1 and PortOutValue2 assigned to the <b>set_pixel</b> command Number times in immediate succession.
<b>Call</b>	set_n_pixel( PortOutValue1, PortOutValue2, Number )
<b>Parameters</b>	PortOutValue1 Like <b>set_pixel</b> .
	PortOutValue2 Like <b>set_pixel</b> .
	Number Number of pixels. As an unsigned 32-bit value. Allowed value range: [1...(2 <sup>32</sup> -1)]. 0 is automatically set to 1.
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>set_n_pixel</b>, see <a href="#">Chapter 8.7 "Pixel Output Mode – Marking Pixel Images (Bitmaps)", page 249</a>.</li> <li>Before the first <b>set_n_pixel</b> command of a line, <b>set_pixel_line</b> must have been called.</li> <li><b>set_n_pixel</b> defines the parameters for the following Number (identical) <b>set_pixel</b> commands of an image line. For usage, see comments on <b>set_pixel</b>.</li> <li>If only an individual pixel is to be defined, then <b>set_pixel</b> can be used as an alternative to <b>set_n_pixel</b>. <b>set_pixel</b> is synonymous with <b>set_n_pixel( Number = 1 )</b>.</li> <li>Outside pixel output mode (if <b>set_n_pixel</b> is not directly preceded by <b>set_pixel_line</b>, <b>set_pixel</b> or <b>set_n_pixel</b>), <b>set_n_pixel</b> is a short list command and otherwise ignored. Under some circumstances, a <b>list_continue</b> might be inserted, see <a href="#">Section "Normal, Short, Variable and Multiple List Commands", page 288</a>.</li> </ul>
RTC4→RTC6	New command.  For <a href="#">RTC4 Compatibility Mode</a> : see <b>set_pixel</b> .
RTC5→RTC6	Unchanged functionality.  See <b>set_pixel</b> .
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>set_pixel</b> , <b>set_pixel_line</b> , <b>set_pixel_line_3d</b>



<b>Ctrl Command</b>	<b>set_offset</b>								
Function	<p>Defines:</p> <ul style="list-style-type: none"> <li>The offset for coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 210.</a></li> <li>The global offset in virtual image field, see <a href="#">Section "Coordinate Transformations in the Virtual Image Field", page 158</a></li> </ul>								
Call	<code>set_offset( HeadNo, XOffset, YOffset, at_once )</code>								
Parameters	<table> <tr> <td>HeadNo</td> <td>See <a href="#">set_offset_xyz</a>.</td> </tr> <tr> <td>XOffset</td> <td>See <a href="#">set_offset_xyz</a>.</td> </tr> <tr> <td>YOffset</td> <td>See <a href="#">set_offset_xyz</a>.</td> </tr> <tr> <td>at_once</td> <td>See <a href="#">set_offset_xyz</a>.</td> </tr> </table>	HeadNo	See <a href="#">set_offset_xyz</a> .	XOffset	See <a href="#">set_offset_xyz</a> .	YOffset	See <a href="#">set_offset_xyz</a> .	at_once	See <a href="#">set_offset_xyz</a> .
HeadNo	See <a href="#">set_offset_xyz</a> .								
XOffset	See <a href="#">set_offset_xyz</a> .								
YOffset	See <a href="#">set_offset_xyz</a> .								
at_once	See <a href="#">set_offset_xyz</a> .								
Comments	<ul style="list-style-type: none"> <li><b>set_offset</b> has the same effect as <a href="#">set_offset_xyz</a>, but leaves <code>ZOffset</code> unchanged.</li> </ul>								
RTC4→RTC6	<p>Unchanged first functionality (offset definition).</p> <p>See <a href="#">set_offset_xyz</a>.</p>								
RTC5→RTC6	<p>Unchanged functionality.</p> <p>See <a href="#">set_offset_xyz</a>.</p>								
Version info	Available as of version DLL 600, OUT 600, RBF 600.								
References	<a href="#">set_offset_list</a> , <a href="#">set_offset_xyz</a> , <a href="#">set_angle</a> , <a href="#">set_matrix</a> , <a href="#">set_scale</a>								

<b>Variable List Command</b>	<b>set_offset_list</b>								
Function	Like <a href="#">set_offset_xyz</a> , but a list command.								
Call	<code>set_offset_list( HeadNo, XOffset, YOffset, at_once )</code>								
Parameters	<table> <tr> <td>HeadNo</td> <td>See <a href="#">set_offset_xyz</a>.</td> </tr> <tr> <td>XOffset</td> <td>See <a href="#">set_offset_xyz</a>.</td> </tr> <tr> <td>YOffset</td> <td>See <a href="#">set_offset_xyz</a>.</td> </tr> <tr> <td>at_once</td> <td>See <a href="#">set_offset_xyz_list</a>.</td> </tr> </table>	HeadNo	See <a href="#">set_offset_xyz</a> .	XOffset	See <a href="#">set_offset_xyz</a> .	YOffset	See <a href="#">set_offset_xyz</a> .	at_once	See <a href="#">set_offset_xyz_list</a> .
HeadNo	See <a href="#">set_offset_xyz</a> .								
XOffset	See <a href="#">set_offset_xyz</a> .								
YOffset	See <a href="#">set_offset_xyz</a> .								
at_once	See <a href="#">set_offset_xyz_list</a> .								
RTC4→RTC6	See <a href="#">set_offset_xyz</a> .								
RTC5→RTC6	Unchanged functionality.								
Version info	Available as of version DLL 600, OUT 600, RBF 600.								
References	<a href="#">set_offset_xyz</a>								



<b>Ctrl Command</b>	<b>set_offset_xyz</b>
<b>Function</b>	<p>Defines:</p> <ul style="list-style-type: none"> <li>The offset for coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 210.</a></li> <li>The global offset in virtual image field, see <a href="#">Section "Coordinate Transformations in the Virtual Image Field", page 158</a></li> </ul>
<b>Call</b>	<code>set_offset_xyz( HeadNo, XOffset, YOffset, ZOffset, at_once )</code>
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value. = 1: The definition only affects the <i>first</i> scan head connector. = 2: The definition only affects the <i>second</i> scan head connector. = 0, 3: The definition affects <i>both</i> scan head connectors. = 4: The definition affects the virtual image field. Only the three least significant bits are evaluated.</p> <p>XOffset     Offsets for the x direction (coordinate translation relative to the origin of the Cartesian coordinate system). In bits. As a signed 32-bit value. Allowed value range: [-268,435,456...+268,435,455]. Out-of-range values are clipped to the boundary values.</p> <p>YOffset     Like <code>XOffset</code> (analogously).</p> <p>ZOffset     Offset for the z directions (coordinate translation relative to the origin of the Cartesian coordinate system). In bits. As a signed 32-bit value. Allowed value range: [-524,288...+524,287]. Out-of-range values are clipped to the boundary values.</p> <p>at_once    Like <a href="#">set_matrix</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li><code>ZOffset</code> is stored only once, applying jointly for both scan head connectors (<code>HeadNo</code> is therefore ignored).</li> <li><code>ZOffset ≠ 0</code> causes a shift of the working plane (opposite to the direction of the laser beam). A positive value increases the z coordinate. For a hypothetical control value <code>of(0 0 0)</code>, this would be by <math>d = ZOffset / K</math> to the plane <math>z = +d</math>. On the calibration factor <math>K</math>, see <a href="#">Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156</a>; furthermore, also 3 und 6 in <a href="#">Chapter 7.3.6 "Output Values to the Scan System", page 170</a>.</li> <li>If <code>HeadNo = 4</code>, then <code>ZOffset</code> is ignored.</li> <li>For <code>at_once</code>, see <a href="#">set_matrix</a>.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC6 multiplies the specified values for <code>XOffset</code>, <code>YOffset</code>, <code>ZOffset</code> by 16. The allowed value ranges decrease accordingly.</p>
RTC5→RTC6	<p>Unchanged functionality.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC6 multiplies the specified value for <code>ZOffset</code> by 16. The allowed value range decreases accordingly.</p>
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_offset_xyz_list</a> , <a href="#">set_offset</a> , <a href="#">set_angle</a> , <a href="#">set_matrix</a> , <a href="#">set_scale</a> , <a href="#">set_defocus</a>



<b>Variable List Command</b>	<b>set_offset_xyz_list</b>
<b>Function</b>	Like <a href="#">set_offset_xyz</a> , but a list command.
<b>Call</b>	<code>set_offset_xyz_list( HeadNo, XOffset, YOffset, ZOffset, at_once )</code>
<b>Parameters</b>	HeadNo      See <a href="#">set_offset_xyz</a> .
	XOffset      See <a href="#">set_offset_xyz</a> .
	YOffset      See <a href="#">set_offset_xyz</a> .
	ZOffset      See <a href="#">set_offset_xyz</a> .
	at_once      Like <a href="#">set_matrix_list</a> .
RTC4→RTC6	See <a href="#">set_offset_xyz</a> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_offset_xyz</a> , <a href="#">set_offset_list</a> , <a href="#">set_defocus_list</a>



<b>Ctrl Command</b>	<b>set_pause_list_cond</b>
<b>Function</b>	Defines the condition at the 16-bit digital input port of the EXTENSION 1 socket connector under which a <b>pause_list</b> automatically is executed.
<b>Call</b>	set_pause_list_cond( Mask1, Mask0 )
<b>Parameters</b>	<p>Mask1      16-bit mask. As an unsigned 32-bit value. Only the least 16 bits are evaluated.</p> <p>Mask0      As Mask1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If a list is currently executed and the following condition is met, see also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 281</a>:  <math>((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})</math>          (= if the bits in IOValue specified in Mask1 are 1 and the bits specified in Mask0 are 0), then automatically a <b>pause_list</b> is executed.          The condition is checked once per <math>10 \mu\text{s}</math> clock period.</li> <li>The paused list can only be continued by <b>restart_list</b>.</li> <li>Mask1 = Mask0 = 0 disables the condition.</li> <li>If the condition is disabled or no list is currently executed, nothing else happens.</li> <li>With <b>set_pause_list_cond</b> in the case of an error the currently executed list can be paused immediately by a hardware circuit. This avoids using a time critical call of a command via the operating system.</li> <li>A conditional <b>pause_list</b> takes precedence over a simultaneously present /STOP signal.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 609, OUT 609, RBF 614.
References	<a href="#">pause_list</a> , <a href="#">restart_list</a> , <a href="#">set_pause_list_not_cond</a>



<b>Ctrl Command</b>	<b>set_pause_list_not_cond</b>
<b>Function</b>	Defines the “NOT” condition at the 16-bit digital input port of the EXTENSION 1 socket connector under which a <b>pause_list</b> automatically is executed.
<b>Call</b>	set_pause_list_not_cond( Mask1, Mask0 )
<b>Parameters</b>	<p>Mask1      16-bit mask. As an unsigned 32-bit value. Only the least 16 bits are evaluated.</p> <p>Mask0      As Mask1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If a list is currently executed and the following condition is <i>not</i> met conditional commands, see also <b>Chapter 9.3.2 “Conditional Command Execution”, page 281:</b>  <math>((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})</math>          (= if the bits in IOValue specified in Mask1 are 1 and the bits specified in Mask0 are 0), then automatically a <b>pause_list</b> is executed.          The condition is checked once per <math>10 \mu\text{s}</math> clock period.</li> <li>The paused list can only be continued by <b>restart_list</b>.</li> <li>Mask1 = Mask0 = 0 disables the condition.</li> <li>If the condition is disabled or no list is currently executed, nothing else happens.</li> <li>With <b>set_pause_list_not_cond</b> in the case of an error the currently executed list can be paused immediately by a hardware circuit. This avoids using a time critical call of a command via the operating system.</li> <li>A conditional <b>pause_list</b> takes precedence over a simultaneously present /STOP signal.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 610, OUT 610, RBF 615.
References	<b>pause_list, restart_list, set_pause_list_cond</b>



<b>Variable List Command</b>	<b>set_pixel</b>
<b>Function</b>	In the pixel output mode, defines the laser control parameters (in “Classic Mode” pulse length and analog voltage level) for <b>one</b> pixel in an image line.
<b>Call</b>	<code>set_pixel( PortOutValue1, PortOutValue2 )</code>
<b>Parameters</b>	<p>PortOutValue1    For the meaning, see <a href="#">Chapter 8.7.3 “Laser Control”, page 251</a>. As an unsigned 32-bit value.</p> <p>PortOutValue2    Like PortOutValue1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_pixel</b> is synonymous with <b>set_n_pixel</b> with <b>Number</b> = 1 (see comments there).</li> </ul>
RTC4→RTC6	<ul style="list-style-type: none"> <li>The RTC6 does <i>not</i> support querying of analog voltage levels (see the RTC4’s <b>ADChannel</b> parameter and <b>read_pixel_ad</b> command).</li> <li>Pixelmode 0 is no longer supported.</li> <li>“Classic Mode” (compatible to RTC4 pixel output mode 1): In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified value for <b>PulseLength</b> (PortOutValue1) by 8 and the one for <b>AnalogOut</b> (PortOutValue2) by 4. The allowed value ranges decrease accordingly. The other pixel output modes are not RTC4 compatible.</li> </ul>
RTC5→RTC6	Unchanged functionality. Only applies to “Classic Mode”, since the other pixel output modes are not supported by the RTC5.
<b>Version info</b>	Last change version DLL 604, OUT 604, RBF 609.
<b>References</b>	<a href="#">set_n_pixel</a> , <a href="#">set_pixel_line</a> , <a href="#">set_pixel_line_3d</a>



<b>Normal List Command</b>	<b>set_pixel_line</b>
<b>Function</b>	Activates a pixel output mode and defines various pixel output parameters.
<b>Call</b>	<code>set_pixel_line( Channel, HalfPeriod, dX, dY )</code>
<b>Parameters</b>	<p>Channel      Defines the pixel output mode as well as the output ports, where the pixel contents are to be outputted. The pixel contents by themselves are defined in <b>set_pixel</b> oder <b>set_n_pixel</b> commands. These must directly follow <b>set_pixel_line</b>.  The following applies: Channel = Mode + Port.  As an unsigned 32-bit value.</p> <p>Allowed are:</p> <ul style="list-style-type: none"> <li>Mode = 0:      Output of <code>PortOutValue1</code> to the pixel duration (formerly parameter <code>PulseLength</code>) and output of <code>PortOutValue2</code> to the analog output port Port. The galvanometer scanner movement is not continuous, see <a href="#">Section "RTC5→RTC6", page 683</a>. "Classic Mode".</li> <li>Mode = 16:      Two outputs at Port. One 32-bit pixel in each output. "Extended Mode".</li> <li>Mode = 32:      Two outputs at Port. Two 16-bit pixels in each output. "Fast Mode".</li> <li>Mode = 64:      Two outputs at Port. Four 8-bit pixels in each output. "Ultra Fast Mode".</li> <li>Mode = 256:      Like Mode = 0. However, the galvanometer scanner movement is continuous, see <a href="#">Section "RTC5→RTC6", page 683</a>.</li> <li>Mode + 512:      Like Mode = 0. However, the galvanometer scanner movement with Sky Writing (position-accurate pixel line). Cannot be combined with Mode = 256. See also <a href="#">Chapter 8.7.4 "Synchronization", page 254</a> and <a href="#">figure 60</a>.</li> <li>Port = 1:      12-bit analog output port 1 (LASER connector). See comment, <a href="#">page 682</a>.</li> <li>Port = 2:      12-bit analog output port 2 (LASER connector. With the RTC6 PCIe Board also the MARKING ON THE FLY socket connector). See comment, <a href="#">page 682</a>.</li> <li>Port = 3:      8-bit digital output port (EXTENSION 2 socket connector).</li> <li>Port = 4:      16-bit digital output port (EXTENSION 1 socket connector).</li> <li>Port = 5:      Output of <code>PortOutValue1</code> and <code>PortOutValue2</code> of <b>set_pixel</b> and of <b>set_n_pixel</b> to the pixel duration (formerly parameter <code>PulseLength</code>). Not allowed with Mode = 0 and Mode = 256.</li> </ul>

<b>Normal List Command</b>	<b>set_pixel_line</b>
Parameters (cont'd)	<p>HalfPeriod     Half pixel output period. In bits.  <i>1 bit equals 1/64 µs.</i>  As an unsigned 32-bit value.  Allowed value range: [Min...(2<sup>32</sup>-1)] with:  Min = 80 for Mode = 0 and 256. Maximum frequency: 0.4 MHz.  Min = 40 for Mode = 16. Maximum frequency: 0.8 MHz.  Min = 20 for Mode = 32. Maximum frequency: 1.2 MHz.  For RTC6 Boards without <b>Option "UFP"</b> the following applies:  the value is clipped at 40.  Min = 10 for Mode = 64. Maximum frequency: 3.2 MHz.  For RTC6 Boards <b>Option "UFP"</b> the following applies:  the value is clipped at 40.</p>
	<p>dx     Distance in the X direction between adjacent pixels. In bits.  As a 64-bit IEEE floating point value.</p>
	<p>dy     Distance in the Y direction between adjacent pixels. In bits.  As a 64-bit IEEE floating point value.</p>
Comments	<ul style="list-style-type: none"> <li>Each image line of a pixel image must be started by <b>set_pixel_line</b>. <b>set_pixel_line</b> should be preceded by a jump command or <b>[*]mark[*]</b> command to the start point of the image line.</li> <li>Directly after <b>set_pixel_line</b>, the required number of <b>set_pixel</b> and <b>set_n_pixel</b> commands must follow. These transmit the <b>PortOutValue1</b> and <b>PortOutValue2</b> parameters. Their meaning depend from Mode and Port (see above, parameter Channel). Siehe auch <b>Chapter 8.7.3 "Laser Control"</b>, page 251.</li> <li>The first list command after <b>set_pixel_line</b> that is not a <b>set_pixel</b> or <b>set_n_pixel</b> command turns off the pixel output mode. <b>set_pixel_line</b>, too, ends the pixel output mode before starting it again. In the process, a default pixel is inserted.</li> <li>The default pixel for Port 1...4 is defined by <b>set_port_default</b>. But beware: Port numbers of <b>set_port_default</b> and <b>set_pixel_line</b> do not match! The default pixel for Port 5 (<b>PulseLength</b>) is defined by <b>set_default_pixel</b>.</li> <li>With unallowed <b>Channel</b> values (example: 5, for Mode = 0 and Port = 5) <b>set_pixel_line</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>). The pixel output mode is <i>not</i> activated in these cases.</li> <li>Note that <i>half</i> the period length must be specified for <b>HalfPeriod</b>. <math>2 \times \text{HalfPeriod}</math> is thus the chronological distance between individual pixels, see <b>Chapter 8.7 "Pixel Output Mode – Marking Pixel Images (Bitmaps)"</b>, page 249. <b>HalfPeriod</b> must not be smaller than Min (is automatically clipped).</li> <li>For outputs at the 12-bit analog output port 1 and 12-bit analog output port 2 the following must be obeyed: only for pixel output frequencies up to around 100 kHz (that is, for a <b>HalfPeriod</b> &lt; approx. 320) digital-to-analog conversion is always fully completed. With such pixel output frequencies users must carefully verify whether the results are as expected. With pixel output frequencies &gt; 100 kHz, usage of the UFP Extension Board, see <b>Chapter 16 "Appendix B: The UFP Extension Board"</b>, page 872, is recommended.</li> </ul>



<b>Normal List Command</b>	<code>set_pixel_line</code>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The pixel output mode can be combined with Processing-on-the-fly. See also <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>.</li> <li>The pixel output mode should <i>not</i> be used in conjunction with "Automatic Laser Control" (see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a>, page 186), if there is a readjustment of the port output, pulse length (<code>PulseLength</code>) or output period (<code>HalfPeriod</code>) of the laser signals LASER1 and LASER2.</li> <li>The pixel output mode is incompatible with the softstart mode. See also <a href="#">Chapter 7.4.7 "Softstart Mode (not yet implemented)", page 184</a>.</li> <li>The pixel output mode <i>cannot</i> be combined with Sky Writing. However, a Sky Writing mode 1-like movement can be set by Mode + 512. This requires Sky Writing to be switched on. In case of a <a href="#">SCANAhead system</a>, the automatic delay calculation must be switched on in addition. This allows pixel lines to be positioned accurately. This Sky Writing movement cannot be combined with Sky Writing mode 2 movements. See also: <ul style="list-style-type: none"> <li><a href="#">Chapter 8.7.4 "Synchronization", page 254</a> and figure 60</li> <li><a href="#">Chapter 7.2.4 "Sky Writing", page 149</a></li> </ul> </li> <li>The pixel output mode <i>cannot</i> be combined with Wobbel. See also <a href="#">Chapter 8.4 "Wobbel Mode", page 218</a>.</li> <li>See also <a href="#">Chapter 8.7 "Pixel Output Mode – Marking Pixel Images (Bitmaps)", page 249</a>.</li> </ul>
RTC4→RTC6	<ul style="list-style-type: none"> <li>For differences to the pixel output mode of the RTC4, see <a href="#">Section "Special Functions", page 46</a>.</li> <li>In <a href="#">RTC4 Compatibility Mode</a>, the RTC6 multiplies the specified value for <code>HalfPeriod</code> by 8 and those for <code>dX</code> and <code>dY</code> by 16. The allowed value ranges decrease accordingly.</li> </ul>
RTC5→RTC6	In <a href="#">"Classic Mode"</a> (RTC5 compatible), frequencies up to 400 kHz are possible. Furthermore, there are additional pixel output modes.
Version info	Last change version DLL 604, OUT 604, RBF 609. Additional pixel output modes: 16, 32, 64 and +512.
References	<a href="#">set_default_pixel</a> , <a href="#">set_default_pixel_list</a> , <a href="#">set_pixel</a> , <a href="#">set_n_pixel</a> , <a href="#">set_pixel_line_3d</a> , <a href="#">set_port_default</a>



<b>Multiple List Command</b>	<b>set_pixel_line_3d</b>
<b>Function</b>	Activates the pixel output mode and defines various pixel output parameters.
<b>Call</b>	<code>set_pixel_line_3d( Channel, HalfPeriod, dX, dY, dZ )</code>
<b>Parameters</b>	Channel      See <a href="#">set_pixel_line</a> (number of the analog output port).
	HalfPeriod    See <a href="#">set_pixel_line</a> ( <i>half</i> pixel output period).
	dX            See <a href="#">set_pixel_line</a> : Distance in the X direction between adjacent pixels. In bits. As a 64-bit IEEE floating point value.
	dY            See <a href="#">set_pixel_line</a> : Distance in the Y direction between adjacent pixels. In bits. As a 64-bit IEEE floating point value.
	dZ            Distance in the Z direction between adjacent pixels. In bits. As a 64-bit IEEE floating point value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>set_pixel_line_3d</code> lets you define the pixel spacing (between two adjacent image points on a line) by a 3D vector.</li> <li>• <code>set_pixel_line_3d</code> additionally requires two list-memory positions, if parameter <code>dZ</code> is non-zero. The initial component executes as a short list command before the principal part (a normal list command). Any still-pending delayed short list command gets executed first.</li> <li>• In other respects, <code>set_pixel_line_3d</code> behaves similarly to <a href="#">set_pixel_line</a> (see comments there).</li> </ul>
RTC4→RTC6	New command. <a href="#">RTC4 Compatibility Mode</a> : see <a href="#">set_pixel_line</a> . In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 does <i>not</i> multiply the specified value for <code>dZ</code> by 16.
RTC5→RTC6	Basically unchanged functionality. <a href="#">RTC5 Compatibility Mode</a> : see <a href="#">set_pixel_line</a> .
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600. Last change version: see <a href="#">set_pixel_line</a> .
<b>References</b>	<a href="#">set_pixel_line</a> , <a href="#">set_pixel</a> , <a href="#">set_n_pixel</a>



<b>Ctrl Command</b>	<b>set_port_default</b>
<b>Function</b>	Defines the default output value for the selected output port.
<b>Call</b>	<code>set_port_default( Port, Value )</code>
<b>Parameters</b>	<p><b>Port</b> Selection of the output port for which a default value is to be defined. As an unsigned 32-bit value. Allowed values: = 0: ANALOG OUT1 output port. See also <a href="#">Section "12-Bit Analog Output Port 1 and 2", page 63</a>. = 1: ANALOG OUT2 output port. See also <a href="#">Section "12-Bit Analog Output Port 1 and 2", page 63</a>. = 2: 8-bit digital output port. See also <a href="#">Section "8-Bit Digital Output Port", page 70</a>. = 3: 16-bit digital output port. See also <a href="#">Section "16-Bit Digital Input and 16-Bit Digital Output", page 67</a>. = 4: 2-bit digital output port. See also <a href="#">Section "2-Bit Digital Output Port", page 63</a>.</p>
<b>Value</b>	<p>Default value. As an unsigned 32-bit value. Allowed values: For Port = 0/1: 12-bit values[0...4095]. Higher bits are ignored. For Port = 2: 8-bit values [0...255]. Higher bits are ignored. For Port = 3: 16-bit values [0...(2<sup>16</sup>-1)]. Higher bits are ignored. For Port = 4: 2-bit values [0...3]. Higher bits are ignored. For Value = "-1" (= 2<sup>32</sup>-1 = 0xFFFFFFFF), the corresponding default output functionality is switched off (see comment below).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>During initialization of the RTC6 (by <a href="#">load_program_file</a>), the default values of all output ports are set to "-1" (= 2<sup>32</sup>-1).</li> <li>If a default value ≠ "-1" has been specified for an output port, the port is set to this default value as soon as processing of a list has ended with <a href="#">stop_execution</a> or by an external stop. For a default value of "-1", the corresponding output port is <i>not</i> addressed (any existing high-impedance state of the digital output ports is retained).</li> <li>Default values (≠ "-1") at the output ports 0...3 are also set at the end of pixel output mode, see <a href="#">Chapter 8.7 "Pixel Output Mode – Marking Pixel Images (Bitmaps)", page 249</a>.</li> <li>After initialization of position- and/or speed-dependent laser control by <a href="#">set_auto_laser_control</a>, the corresponding default value (for output port 0, 1, 2 or 3, depending on the selected laser signal parameter Ctrl), is also outputted when the laser is switched off after marking or when position- and/or speed-dependent laser control is set to another Ctrl parameter by <a href="#">set_auto_laser_control</a> or deactivated by <a href="#">set_auto_laser_control</a> (Ctrl = 0), see <a href="#">Chapter 7.4.1 "Enabling, Activating and Switching Laser Control Signals", page 173</a> and <a href="#">Section "General Notes", page 187</a>. If the default value is set to "-1", then the maximum allowed value (4095, 4095, 255 or 65,535) is outputted.</li> </ul>



<b>Ctrl Command</b>	<b>set_port_default</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If the value for <code>Port</code> is invalid, then <code>set_port_default</code> is not executed (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>The default values for the output ports 0...2 can also be defined by <code>set_laser_off_default</code>.</li> </ul>
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies Value for Port = 0/1 by 4.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_laser_off_default</a> , <a href="#">set_default_pixel</a>

<b>Undelayed Short List Command</b>	<b>set_port_default_list</b>				
Function	Like <code>set_port_default</code> , but a list command.				
Call	<code>set_port_default_list( Port, Value )</code>				
Parameters	<table> <tr> <td>Port</td> <td>See <code>set_port_default</code>.</td> </tr> <tr> <td>Value</td> <td>See <code>set_port_default</code>.</td> </tr> </table>	Port	See <code>set_port_default</code> .	Value	See <code>set_port_default</code> .
Port	See <code>set_port_default</code> .				
Value	See <code>set_port_default</code> .				
Comments	<ul style="list-style-type: none"> <li>See <code>set_port_default</code>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 609, OUT 609, RBF 614.				
References	<a href="#">set_port_default</a>				



<b>Ctrl Command</b>	<b>set_pulse_picking</b>
<b>Function</b>	Switches on pulse picking laser mode.
<b>Call</b>	<code>set_pulse_picking( No )</code>
<b>Parameters</b>	No      As an unsigned 32-bit value. Allowed value range: [0...63]. = 0: LASER2 puts out the LASERON signal. = 1...63: LASER2 puts out each $No^{th}$ LASER1 pulse. $No > 63$ : Clipped to 63 ( <a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a> ).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For pulse picking laser mode, see <a href="#">Chapter 7.4.8 "Pulse Picking Laser Mode", page 185</a>.</li> <li>The pulse picking signals are outputted until a different laser mode gets set by <a href="#">set_laser_mode</a> (the pulse picking laser mode gets switched off if any other laser mode switches on by <a href="#">set_laser_mode</a>).</li> <li>You can <i>not</i> switch on pulse picking laser mode by <a href="#">set_laser_mode</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_laser_mode</a> , <a href="#">set_pulse_picking_list</a>

<b>Ctrl Command</b>	<b>set_pulse_picking_length</b>
<b>Function</b>	Defines a constant pulse length for the LASER2 signal in pulse picking laser mode.
<b>Call</b>	<code>set_pulse_picking_length( Length )</code>
<b>Parameters</b>	Length      Pulse length. As an unsigned 32-bit value. Allowed value range: [0...65,535]. Higher bits are ignored. <i>1 bit equals 1/64 μs</i> . The default value after <a href="#">load_program_file</a> is 0.
<b>Comments</b>	<ul style="list-style-type: none"> <li>For the value to take effect, the following must have been activated:             <ul style="list-style-type: none"> <li>Pulse picking laser mode by <a href="#">set_pulse_picking</a></li> <li>The "constant pulse length" mode by <a href="#">set_laser_control</a>( Bit #7 = 1 ).</li> </ul>             The value then takes immediate effect, even if a marking is carried out.         </li> <li>If pulse picking laser mode has been activated, but not constant pulse length mode (<a href="#">set_laser_control</a>( Bit #7 = 0 )), then the pulse-picking signal uses the pulse length of the LASER1 signal, see <a href="#">Chapter 7.4.8 "Pulse Picking Laser Mode", page 185</a>.</li> <li>If neither pulse picking laser mode nor constant pulse length mode has been activated, then the value <code>Length</code> is irrelevant (<a href="#">set_pulse_picking</a>, <a href="#">set_laser_control</a> and <a href="#">set_pulse_picking_length</a> can be activated in any desired order).</li> <li>After <a href="#">set_pulse_picking</a>( 0 ), LASER2 is continuously output the LASERON signal, but no constant pulse length signal.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–



<b>Undelayed Short List Command</b>	<b>set_pulse_picking_list</b>
Function	Like <b>set_pulse_picking</b> , but a list command.
Call	<code>set_pulse_picking_list( No )</code>
Parameters	No      Like <b>set_pulse_picking</b> .
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_pulse_picking</b>

<b>Ctrl Command</b>	<b>set_qswitch_delay</b>
Function	In the YAG modes, defines the delay length of the first Q-Switch pulse with reference to the FirstPulseKiller signal, see also <b>figure 49</b> .
Call	<code>set_qswitch_delay( Delay )</code>
Parameters	Delay      Q-Switch Delay. As an unsigned 32-bit value. 1 bit equals 1/64 µs. Allowed value range: [0...(2 <sup>26</sup> -1)].
Comments	<ul style="list-style-type: none"> <li>Values over (2<sup>26</sup>-1) are clipped.</li> <li>The YAG modes are selectable by <b>set_laser_mode</b> ([1, 2, 3 or 5]).</li> <li>Also for YAG modes defined with <b>set_laser_mode</b>([1-3]), the length of the Q-Switch delay can be subsequently changed by this command; and for YAG mode 2 also with <b>set_firstpulse_killer</b> or <b>set_firstpulse_killer_list</b>.</li> </ul>
RTC4→RTC6	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified value for <code>Delay</code> by 8. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_qswitch_delay_list</b> , <b>set_laser_control</b> , <b>set_laser_pulses_ctrl</b> , <b>set_laser_pulses</b> , <b>set_laser_timing</b> , <b>set_firstpulse_killer</b> , <b>set_firstpulse_killer_list</b>



<b>Undelayed Short List Command</b>	<b>set_qswitch_delay_list</b>
Function	Like <a href="#">set_qswitch_delay</a> , but a list command.
Call	<code>set_qswitch_delay_list( Delay )</code>
Parameters	Delay      Q-Switch Delay. As an unsigned 32-bit value. <i>1 bit equals 1/64 µs.</i> Allowed value range: [0...(2 <sup>26</sup> -1)].
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_qswitch_delay</a>



<b>Ctrl Command</b>	<b>set_rot_center</b>
<b>Function</b>	Sets the rotation center of a Processing-on-the-fly rotation correction (see <a href="#">set_fly_rot</a> and <a href="#">set_fly_rot_pos</a> ).
<b>Call</b>	<code>set_rot_center( X, Y )</code>
<b>Parameters</b>	X Position of the rotation center referenced to the zero point (0 0) of the image field as signed 32-bit values. Allowed value range: [-2 <sup>24</sup> ... (2 <sup>24</sup> -1)]
	Y Like X (analogously).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For Processing-on-the-fly correction, see <a href="#">Chapter 8.6.3 "Compensation of Rotary Movements", page 232</a>.</li> <li>The position of the rotation center should be defined by <a href="#">set_rot_center</a> or <a href="#">set_rot_center_list</a> before the Processing-on-the-fly correction is activated by <a href="#">set_fly_rot</a> or <a href="#">set_fly_rot_pos</a>.</li> <li>The rotation center can also lie outside the image field. The allowed area is equivalent to 32x the image field.</li> <li>Usage of a second scan head is only practical if it is set up for exactly the same rotational center.</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified coordinate values by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_fly_rot</a> , <a href="#">set_fly_rot_pos</a>

<b>Delayed Short List Command</b>	<b>set_rot_center_list</b>
<b>Function</b>	Like <a href="#">set_rot_center</a> , but a list command.
<b>Call</b>	<code>set_rot_center_list( X, Y )</code>
<b>Parameters</b>	X Like <a href="#">set_rot_center</a> .
	Y Like <a href="#">set_rot_center</a> .
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_rot_center</a>



<b>Ctrl Command</b>	<b>set_RTC4_mode</b>
<b>Function</b>	Sets <b>RTC4 Compatibility Mode</b> as the current DLL operation mode.
<b>Call</b>	<code>set_RTC4_mode()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>RTC4 Compatibility Mode</b> is an optional operation mode of the <b>RTC6 DLL</b>. It has been made available so that user programs written for the RTC4 can also be processed by the RTC6 (to a large extent) without needing to modify the programming code. However, a prerequisite here is that the program can only contain RTC4 commands that also exist with unchanged functionality as RTC6 commands. In the command descriptions of this user manual such changes are noted in the "RTC4→RTC6" row.</li> <li>• <b>RTC6 Standard Mode</b> is pre-defined as the default DLL operation mode and can also be specified subsequently by <b>set_RTC6_mode</b>.</li> <li>• The current DLL operation mode can be queried by <b>get_RTC_mode</b>.</li> <li>• <b>set_RTC4_mode</b> is available even without explicit access rights to a particular board.</li> <li>• <b>set_RTC4_mode</b> is not available as a multi-board command.</li> <li>• The scope of <b>set_RTC4_mode</b> is not board-specific, but rather global to the <b>RTC6 DLL</b> and all RTC6 boards to which the user program has access rights.</li> <li>• The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <b>Chapter 6.8 "Error Handling"</b>, page 120) are neither generated nor altered by <b>set_RTC4_mode</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_RTC_mode</b> , <b>set_RTC5_mode</b> , <b>set_RTC6_mode</b>



<b>Ctrl Command</b>	<b>set_RTC5_mode</b>
<b>Function</b>	Sets <b>RTC5 Compatibility Mode</b> as the current DLL operation mode.
<b>Call</b>	<code>set_RTC5_mode()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>RTC5 Compatibility Mode</b> is an optional operation mode of the <b>RTC6 DLL</b>. It has been made available so that user programs written for the RTC5 can also be processed by the RTC6 (to a large extent) without needing to modify the programming code. However, a prerequisite here is that the program can only contain RTC5 commands that also exist with unchanged functionality as RTC6 commands. In the command descriptions of this user manual such changes are noted in the "RTC5→RTC6" row. See also <b>Chapter 2.9.2 "Adapting RTC5 Source Code for the RTC6 PCIe Board"</b>, step 3, page 50.</li> <li>• In <b>RTC5 Compatibility Mode</b> as DLL operation mode Z coordinates and defocus values (for example, parameter <code>Shift</code> of <b>set_defocus</b>) must be specified with a parameter resolution of 16 bits. These values are automatically multiplied by 16. The allowed value ranges decrease accordingly.</li> <li>• In <b>RTC5 Compatibility Mode</b> as DLL operation mode the laser delay values (parameter <code>LaserOnDelay</code> and <code>LaserOffDelay</code> of <b>set_laser_delays</b>) must be specified with a parameter resolution of 1/2 µs. These values are automatically multiplied by 32. The allowed value ranges decrease accordingly.</li> <li>• <b>RTC6 Standard Mode</b> is pre-defined as the default DLL operation mode and can also be specified subsequently by <b>set_RTC6_mode</b>.</li> <li>• The current DLL operation mode can be queried by <b>get_RTC_mode</b>.</li> <li>• <b>set_RTC5_mode</b> is available even without explicit access rights to a particular RTC6 board.</li> <li>• <b>set_RTC5_mode</b> is not available as a multi-board command.</li> <li>• The scope of <b>set_RTC5_mode</b> is not board-specific, but rather global to the <b>RTC6 DLL</b> and all RTC6 boards to which the user program has access rights.</li> <li>• The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <b>Chapter 6.8 "Error Handling"</b>, page 120) are neither generated nor altered by <b>set_RTC5_mode</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>get_RTC_mode</b> , <b>set_RTC4_mode</b> , <b>set_RTC6_mode</b>



<b>Ctrl Command</b>	<b>set_RTC6_mode</b>
<b>Function</b>	Sets <b>RTC6 Standard Mode</b> as the current DLL operation mode (default setting).
<b>Call</b>	<code>set_RTC6_mode()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>In <b>RTC6 Standard Mode</b> as DLL operation mode, Z coordinate values and defocus values must be specified with a parameter resolution of 20 bits.</li> <li>In <b>RTC6 Standard Mode</b> as DLL operation mode, laser delay values must be specified with a parameter resolution of <math>1/64 \mu\text{s}</math>.</li> <li>The current DLL operation mode can be queried by <b>get_RTC_mode</b>.</li> <li><b>set_RTC6_mode</b> is available even without explicit access rights to a particular RTC6 board.</li> <li><b>set_RTC6_mode</b> is not available as a multi-board command.</li> <li>The scope of <b>set_RTC6_mode</b> is not board-specific, but rather global to the <b>RTC6 DLL</b> and therefore, to all RTC6 boards to which the user program has access rights.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <b>Chapter 6.8 "Error Handling"</b>, page 120) are neither generated nor altered by <b>set_RTC6_mode</b>.</li> </ul>
<b>RTC4→RTC6</b>	New command.
<b>RTC5→RTC6</b>	New command.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>get_RTC_mode</b> , <b>set_RTC4_mode</b> , <b>set_RTC5_mode</b>



<b>Ctrl Command</b>	<b>set_scale</b>
<b>Function</b>	Uses a specified scaling factor common to the x and y axes to define the scaling matrix $M_S$ for all subsequent coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 210</a> .
<b>Call</b>	<code>set_scale( HeadNo, Scale, at_once )</code>
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>= 1:      The definition only affects the <i>first</i> scan head connector.</li> <li>= 2:      The definition only affects the <i>second</i> scan head connector.</li> <li>= 0, 3:      The definition affects <i>both</i> scan head connectors. Only the two least significant bits are evaluated.</li> </ul> <p>Scale      Scaling factor. As a 64-bit IEEE floating point value. Allowed value range: [-16...+16]. If the parameter is set to an invalid value, it is set to 1. Negative values additionally produce a mirroring around both axes (corresponding to a 180° rotation).</p> <p>at_once      Determines when the defined transformation becomes effective. Like <code>set_matrix( HeadNo = 0...3 )</code>. As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">Chapter 8.2 "Coordinate Transformations", page 210</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_scale_list</a> , <a href="#">set_angle</a> , <a href="#">set_matrix</a> , <a href="#">set_offset</a>

<b>Variable List Command</b>	<b>set_scale_list</b>
<b>Function</b>	Like <a href="#">set_scale</a> , but a list command.
<b>Call</b>	<code>set_scale_list( HeadNo, Scale, at_once )</code>
<b>Parameters</b>	<p>HeadNo      Like <a href="#">set_scale</a>.</p> <p>Scale      Like <a href="#">set_scale</a>.</p> <p>at_once      Determines when the defined transformation becomes effective. Like <a href="#">set_matrix_list</a>. As an unsigned 32-bit value.</p>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_scale</a>



<b>Ctrl Command</b>	<code>set_scanahead_laser_shifts</code>
Comments	<p><i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i></p> <p>This command is described in the manual “excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards”.</p>

<b>Undelayed Short List Command</b>	<code>set_scanahead_laser_shifts_list</code>
Comments	<p><i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i></p> <p>This command is described in the manual “excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards”.</p>

<b>Ctrl Command</b>	<code>set_scanahead_line_params</code>
Comments	<p><i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i></p> <p>This command is described in the manual “excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards”.</p>

<b>Undelayed Short List Command</b>	<code>set_scanahead_line_params_list</code>
Comments	<p><i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i></p> <p>This command is described in the manual “excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards”.</p>

<b>Ctrl Command</b>	<code>set_scanahead_params</code>
Comments	<p><i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i></p> <p>This command is described in the manual “excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards”.</p>

<b>Ctrl Command</b>	<code>set_scanahead_speed_control</code>
Comments	<p><i>Only for scan systems with SCANAhead technology, for example, the excelliSCAN.</i></p> <p>This command is described in the manual “excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards”.</p>



<b>Delayed Short List Command</b>	<b>set_scanner_delays</b>
<b>Function</b>	Sets the scanner delays.
<b>Call</b>	<code>set_scanner_delays( Jump, Mark, Polygon )</code>
<b>Parameters</b>	<p><b>Jump</b>      Scanner delay of type: jump delay. As an unsigned 32-bit value.  <i>1 bit equals 10 µs.</i>          Allowed value range: [0...(2<sup>32</sup>-1)].</p> <p><b>Mark</b>      Scanner delay of type: mark delay. As an unsigned 32-bit value.  <i>1 bit equals 10 µs.</i>          Allowed value range: [0...(2<sup>32</sup>-1)].</p> <p><b>Polygon</b>      Scanner delay of type: polygon delay. As an unsigned 32-bit value.  <i>1 bit equals 10 µs.</i>          Allowed value range: [0...(2<sup>32</sup>-1)].</p>
	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 7.2.2 "Scanner Delays", page 136</a>.</li> <li>• Variable Delays for jumps and <a href="#">Polylines</a> can be set by <a href="#">set_delay_mode</a>.</li> <li>• The specified delays are automatically adjusted by the RTC6 to avoid laser control errors, see <a href="#">Section "Automatic Delay Adjustments", page 144</a>.</li> <li>• The default setting after <a href="#">load_program_file</a> corresponds to <code>set_scanner_delays( 9, 6, 3 )</code>.</li> </ul>
	RTC4→RTC6      Unchanged functionality. In addition: increased value range.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_delay_mode</a> , <a href="#">set_laser_delays</a>



<b>Ctrl Command</b>	<b>set_serial</b>
<b>Function</b>	Sets the starting serial number of the serial-number-set most recently selected by <b>select_serial_set</b> (= 0 after <b>load_program_file</b> ) and sets the increment size for this serial-number-set to 1.
<b>Call</b>	<code>set_serial( No )</code>
<b>Parameters</b>	No      Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_serial</b> is synonymous with <b>set_serial_step</b> with Step = 1 (see comments there).</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. <b>set_serial</b> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_serial_step</b> , <b>select_serial_set</b>

<b>Ctrl Command</b>	<b>set_serial_step</b>				
<b>Function</b>	Sets the starting serial number and the increment size for the serial-number-set most recently selected by <b>select_serial_set</b> (= 0 after <b>load_program_file</b> ).				
<b>Call</b>	<code>set_serial_step( No, Step )</code>				
<b>Parameters</b>	<table border="1"> <tr> <td>No</td> <td>Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2<sup>32</sup>-1)].</td> </tr> <tr> <td>Step</td> <td>Increment size. As an unsigned 32-bit value. Allowed value range: [0...9999]; only the last 4 decimal digits are used.</td> </tr> </table>	No	Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].	Step	Increment size. As an unsigned 32-bit value. Allowed value range: [0...9999]; only the last 4 decimal digits are used.
No	Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].				
Step	Increment size. As an unsigned 32-bit value. Allowed value range: [0...9999]; only the last 4 decimal digits are used.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>load_program_file</b> sets the starting serial number to 0 and the increment size to 1.</li> <li>• If <b>mark_serial</b> or <b>mark_serial_abs</b> was called with Mode M<sub>2</sub> = 1 (that is, automatic serial-number incrementing was deactivated), then the increment size setting has no effect.</li> <li>• If Step = 0, then incrementing does not occur, except in the case of markless marking (see <b>mark_serial</b> or <b>mark_serial_abs</b>: digits = 0), which always increments serial numbers by 1.</li> <li>• For usage of <b>set_serial_step</b>, see <b>Chapter 7.5.2 "Marking Serial Numbers", page 197</b>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<b>mark_serial</b> , <b>mark_serial_abs</b> , <b>set_serial</b> , <b>select_serial_set</b> , <b>select_serial_set_list</b>				



<b>Normal List Command</b>	<b>set_serial_step_list</b>				
<b>Function</b>	Sets the starting serial number and the increment size for the serial-number-set most recently selected by <a href="#">select_serial_set_list</a> (= 0 after <a href="#">load_program_file</a> ).				
<b>Call</b>	<code>set_serial_step_list( No, Step )</code>				
<b>Parameters</b>	<table> <tr> <td>No</td><td>Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2<sup>32</sup>-1)]</td></tr> <tr> <td>Step</td><td>Increment size. As an unsigned 32-bit value. Allowed value range: [0...9999]. Only the last 4 decimal digits are used.</td></tr> </table>	No	Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)]	Step	Increment size. As an unsigned 32-bit value. Allowed value range: [0...9999]. Only the last 4 decimal digits are used.
No	Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)]				
Step	Increment size. As an unsigned 32-bit value. Allowed value range: [0...9999]. Only the last 4 decimal digits are used.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_serial_step</a>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">set_serial_step</a> , <a href="#">select_serial_set_list</a> , <a href="#">get_list_serial</a> , <a href="#">mark_serial</a> , <a href="#">mark_serial_abs</a>				

<b>Ctrl Command</b>	<b>set_sky_writing</b>				
<b>Function</b>	Activates Sky Writing mode 1 and sets the corresponding parameters or switches off Sky Writing.				
<b>Call</b>	<code>set_sky_writing( Timelag, LaserOnShift )</code>				
<b>Parameters</b>	<table> <tr> <td>Timelag</td><td>See <a href="#">set_sky_writing_para</a>.</td></tr> <tr> <td>LaserOnShift</td><td>See <a href="#">set_sky_writing_para</a>.</td></tr> </table>	Timelag	See <a href="#">set_sky_writing_para</a> .	LaserOnShift	See <a href="#">set_sky_writing_para</a> .
Timelag	See <a href="#">set_sky_writing_para</a> .				
LaserOnShift	See <a href="#">set_sky_writing_para</a> .				
<b>Comments</b>	<ul style="list-style-type: none"> <li><code>set_sky_writing</code> is identical to <code>set_sky_writing_para( Timelag, LaserOnShift, Nprev, Npost )</code> with <code>Nprev</code> = approx. (0.15 × <code>Timelag</code>) and <code>Npost</code> = approx. (0.1 × <code>Timelag</code>).</li> <li>See also information about <a href="#">set_sky_writing_para</a> and <a href="#">Chapter 7.2.4 "Sky Writing", page 149</a>.</li> </ul>				
RTC4→RTC6	New command.				
RTC5→RTC6	Unchanged functionality.				
Version info	Available as of version DLL 600, OUT 600, RBF 600.				
References	<a href="#">set_sky_writing_para</a> , <a href="#">set_sky_writing_list</a>				



<b>Ctrl Command</b>	<b>set_sky_writing_limit</b>
<b>Function</b>	Defines the limit for Sky Writing switching in mode 3.
<b>Call</b>	<code>set_sky_writing_limit( Limit )</code>
<b>Parameters</b>	<p>Limit      Limit value. As a 64-bit IEEE floating point value. Allowed value range: [-1.0...+1.0]. Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <code>set_sky_writing_limit</code>, see <a href="#">Chapter 7.2.4 "Sky Writing"</a>, page 149.</li> <li>Limit is the cosine of the angular limit (for angular change between consecutive vectors or arcs within a <a href="#">Polyline</a>) for which a Sky Writing motion should be performed.</li> <li>The initialized value (after program start) is Limit = 0 (angular limit = 90°).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_sky_writing_limit_list</a>

<b>Undelayed Short List Command</b>	<b>set_sky_writing_limit_list</b>
<b>Function</b>	Like <a href="#">set_sky_writing_limit</a> , but a list command.
<b>Call</b>	<code>set_sky_writing_limit_list( Limit )</code>
<b>Parameters</b>	Limit      See <a href="#">set_sky_writing_limit</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_sky_writing_limit</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_sky_writing_limit</a>

<b>Normal List Command</b>	<b>set_sky_writing_list</b>
<b>Function</b>	Like <a href="#">set_sky_writing</a> , but a list command.
<b>Call</b>	<code>set_sky_writing_list( Timelag, LaserOnShift )</code>
<b>Parameters</b>	<p>Timelag      See <a href="#">set_sky_writing_para</a>. LaserOnShift      See <a href="#">set_sky_writing_para</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_sky_writing</a>, <a href="#">set_sky_writing_para</a> and <a href="#">set_sky_writing_para_list</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_sky_writing</a> , <a href="#">set_sky_writing_para</a> , <a href="#">set_sky_writing_para_list</a> .



<b>Ctrl Command</b>	<b>set_sky_writing_mode</b>
<b>Function</b>	Switches the Sky Writing mode.
<b>Call</b>	<code>set_sky_writing_mode( Mode )</code>
<b>Parameters</b>	<p>Mode                    Sky Writing mode.                            As an unsigned 32-bit value.                            Allowed value range: [0...(2<sup>32</sup>-1)].                            = 0:     Sky Writing is deactivated.                            = 1:     Sky Writing mode 1 is activated.                            = 2:     Sky Writing mode 2 is activated.                            &gt; 2:    Sky Writing mode 3 is activated.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Sky Writing mode 2 and Sky Writing mode 3 can only be activated (by Mode &gt; 1), if:           <ul style="list-style-type: none"> <li>– Sky Writing mode 1 has been activated by <code>set_sky_writing_para</code> (<code>Timelag</code> <math>\geq</math> 1/64) before</li> <li>– AND is still active</li> </ul> </li> <li>• After <code>set_sky_writing_mode( Mode =0 )</code>, subsequent Sky Writing reactivation:           <ul style="list-style-type: none"> <li>– Is possible by <code>set_sky_writing_mode( Mode &gt; 0 )</code></li> <li>– Not possible after <code>set_sky_writing_para</code> (<code>Timelag</code> <math>&lt;</math> 1/64)</li> </ul> </li> <li>• Each mode switch by <code>set_sky_writing_mode</code> becomes effective as of the next list command.</li> <li>• If you reactivate Sky Writing mode 1 (switching from Mode = 0 to 1) during execution of a list, then an already-begun <b>Mark command</b> still executes to completion <i>without</i> Sky Writing.</li> <li>• If you deactivate Sky Writing mode 1 (switching from Mode = 1 to 0) during execution of a list, then a <b>Mark command</b> already begun in Sky Writing mode 1 still executes to completion in Sky Writing mode 1 and then is appended with a mark delay.</li> <li>• Activation or deactivation of Sky Writing mode 2 or 3 (switching to or from Mode = 2 or 3) is not possible if the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <code>pause_list</code>). Then <code>set_sky_writing_mode</code> is ignored (<code>get_last_error</code> return code <code>RTC6_BUSY</code>). In contrast, <code>set_sky_writing_mode</code> is executed when a list has been paused by <code>set_wait</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_sky_writing_mode_list</a> , <a href="#">set_sky_writing_para</a>



<b>Normal List Command</b>	<b>set_sky_writing_mode_list</b>
<b>Function</b>	Like <b>set_sky_writing_mode</b> , but a list command.
<b>Call</b>	set_sky_writing_mode_list( Mode )
<b>Parameters</b>	Mode      Like <b>set_sky_writing_mode</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>By <b>set_sky_writing_mode_list</b>, Sky Writing mode 2 and Sky Writing mode 3 can be activated or deactivated even within a list (switching to or from Mode = 2 or 3). Here, an already-begun <b>Mark command</b> is finished with Sky Writing mode 2 and the next is started with it.</li> <li>Deactivation of Sky Writing mode 1 by <b>set_sky_writing_mode_list</b> (switching from Mode = 1 to 0) results in the addition of a mark delay defined prior to activation of Sky Writing – provided that no other delay is in effect (for example, a jump delay).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_sky_writing_mode</b>



<b>Ctrl Command</b>	<b>set_sky_writing_para</b>
<b>Function</b>	Activates Sky Writing mode 1 and sets the corresponding parameters or switches Sky Writing off.
<b>Call</b>	<code>set_sky_writing_para( Timelag, LaserOnShift, Nprev, Npost )</code>
<b>Parameters</b>	<p><b>Timelag</b>      Sky Writing parameter. <i>1.0 equals 1 μs.</i>      The <code>Timelag</code> value is used with an accuracy of <math>1/64 \mu s</math>.      As a 64-bit IEEE floating point value.  <math>\geq 1/4</math>: Sky Writing mode 1 is activated. From <math>1/8</math> to <math>1/4</math>, Sky Writing is activated, but <code>Timelag = 0</code> is internally applied.  <math>&lt; 1/8</math>: Sky Writing is deactivated.</p> <p><b>LaserOnShift</b>      Shift (for positive values: delay) of the point of time, where the laser control signals are switched on (see <a href="#">figure 42</a>).      As a signed 32-bit value.  <i>1 bit equals <math>1/64 \mu s</math>.</i>      Negative values smaller than the complete run-in phase are clipped at runtime, therefore the following applies automatically:  <b>LaserOnShiftModus</b>  <math>\geq \text{ca. } (-40) \times N_{\text{prev1}}</math>  <math>\geq \text{ca. } (-20) \times N_{\text{prev2 oder 3}}</math></p> <p><b>Nprev</b>      Defines the duration of the run-in phase (see <a href="#">figure 42</a>).  <i>1 bit equals <math>10 \mu s</math>.</i>      As an unsigned 32-bit value.      Allowed value range: <math>0 \leq N_{\text{prev}} \leq (2^{32}-1)</math>.      Run-in duration [<math>\mu s</math>]      Modus  <math>20 \times N_{\text{prev}}</math>      1  <math>10 \times N_{\text{prev}}</math>      2 oder 3</p> <p><b>Npost</b>      Defines the duration of the run-out phase (see <a href="#">figure 42</a>).  <i>1 bit equals <math>10 \mu s</math>.</i>      As an unsigned 32-bit value.      Allowed value range: <math>0 \leq N_{\text{post}} \leq (2^{32}-1)</math>.      Run-out duration [<math>\mu s</math>]      Modus  <math>20 \times N_{\text{post}}</math>      1  <math>10 \times N_{\text{post}}</math>      2 oder 3</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For information on Sky Writing mode and a description of the parameters, see <a href="#">Chapter 7.2.4 "Sky Writing", page 149</a>.</li> <li>If <math>N_{\text{prev}} \geq 65,535</math>, then <code>set_sky_writing_para</code> behaves similarly to <code>set_sky_writing</code>: <code>Nprev</code> is set to a value of approx. <math>(0.15 \times \text{Timelag})</math>.</li> <li>If <math>N_{\text{post}} \geq 65,535</math>, then <code>set_sky_writing_para</code> behaves similarly to <code>set_sky_writing</code>: <code>Npost</code> is set to a value of approx. <math>(0.1 \times \text{Timelag})</math>.</li> <li><code>set_sky_writing_para</code> cannot be used to switch back and forth between Sky Writing modes 1, 2 and 3 (the proper command for this is <code>set_sky_writing_mode</code>).</li> </ul>



Ctrl Command	<a href="#">set_sky_writing_para</a>
Comments (cont'd)	<ul style="list-style-type: none"> <li>When <code>set_sky_writing_para</code> is called and no list is running or a list has been halted by <code>set_wait</code>, the following applies:           <ul style="list-style-type: none"> <li>With Timelag <math>\geq 1/8</math>, Sky Writing mode 1 is activated, if Sky Writing has not been active before. Sky Writing mode 2 or Sky Writing mode 3 remain, but the next <b>Mark command</b> is always started in Sky Writing mode 1</li> <li>With Timelag <math>&lt; 1/8</math>, Sky Writing is deactivated.</li> </ul> </li> <li>When <code>set_sky_writing_para</code> is called and a list is running or a list has been halted by <code>pause_list</code>, the following applies:           <ul style="list-style-type: none"> <li>If Sky Writing mode 2 or Sky Writing mode 3 is active, then <code>set_sky_writing_para</code> is not executed (<code>get_last_error</code> return code <code>RTC6_BUSY</code>)</li> <li>If no Sky Writing or Sky Writing mode 0 or Sky Writing mode 1 is active, <code>set_sky_writing_para</code> parameters are going to be effective upon the next list command.               <ul style="list-style-type: none"> <li>In Sky Writing mode 1, an already started <b>Mark command</b> is executed with the previous parameters and ended with Sky Writing mode 1.</li> <li>In Sky Writing mode 0 or Sky Writing mode 1, the next <b>Mark command</b> is started in Sky Writing mode 1, if Timelag <math>\geq 1/8</math>. Otherwise, it is terminated and a mark delay is inserted in Sky Writing mode 1.</li> </ul> </li> <li>For Sky Writing mode 2 or 3, <code>Nprev = 0</code> and/or <code>Npost = 0</code> automatically get(s) internally corrected to 1 (this is not treated as an error). If you subsequently activate Sky Writing mode 1 (by <code>set_sky_writing_mode</code>), then <code>Nprev = 0</code> and/or <code>Npost = 0</code> is/are reused.</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_sky_writing</a> , <a href="#">set_sky_writing_para_list</a>



<b>Normal List Command</b>	<b>set_sky_writing_para_list</b>
<b>Function</b>	Like <a href="#">set_sky_writing_para</a> , but a list command.
<b>Call</b>	set_sky_writing_para_list( Timelag, LaserOnShift, Nprev, Npost )
<b>Parameters</b>	Timelag      See <a href="#">set_sky_writing_para</a> .
	LaserOnShift      See <a href="#">set_sky_writing_para</a> .
	Nprev      See <a href="#">set_sky_writing_para</a> .
	Npost      See <a href="#">set_sky_writing_para</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_sky_writing_para_list</b> can be executed within a list, even if Sky Writing mode 2 or 3 is active. Here, an already-begun <b>Mark command</b> is finished with Sky Writing mode 2 and the next is started with it.</li> <li>• By <b>set_sky_writing_para_list</b>, you cannot switch from Sky Writing mode 2 or 3 to Sky Writing mode 1 permanently. For this, use <a href="#">set_sky_writing_mode_list</a>.</li> <li>• Deactivation of Sky Writing mode 1, 2 or 3 by <b>set_sky_writing_para_list</b> results in the addition of a mark delay defined prior to activation of Sky Writing – provided that no other delay is in effect (for example, a jump delay).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_sky_writing_para</a>



<b>Ctrl Command</b>	<b>set_softstart_level</b>
<b>Function</b>	Without function.
<b>Call</b>	set_softstart_level( Index, Level )
<b>Parameters</b>	Index As an unsigned 32-bit value. Level As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_softstart_level</b> has no effect on the user program.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	–

<b>Normal List Command</b>	<b>set_softstart_level_list</b>
<b>Function</b>	Without function.
<b>Call</b>	set_softstart_level_list( Index, Level1, Level2, Level3 )
<b>Parameters</b>	Index Like <b>set_softstart_level</b> . Level1 Like <b>set_softstart_level</b> . Level2 Like <b>set_softstart_level</b> . Level3 Like <b>set_softstart_level</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <b>set_softstart_level</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	–



<b>Ctrl Command</b>	<b>set_softstart_mode</b>
<b>Function</b>	Without function.
<b>Call</b>	set_softstart_mode( Mode, Number, Delay )
<b>Parameters</b>	<p>Mode            As an unsigned 32-bit value.</p> <p>Number        As an unsigned 32-bit value.</p> <p>Delay         As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_softstart_mode</b> has no effect on the user program.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–

<b>Variable List Command</b>	<b>set_softstart_mode_list</b>
<b>Function</b>	Without function.
<b>Call</b>	set_softstart_mode_list( Mode, Number, Delay )
<b>Parameters</b>	<p>Mode            Like <b>set_softstart_mode</b>.</p> <p>Number        Like <b>set_softstart_mode</b>.</p> <p>Delay         Like <b>set_softstart_mode</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <b>set_softstart_mode</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–



<b>Ctrl Command</b>	<b>set_standby</b>
<b>Function</b>	Defines the output period and the pulse length of the standby pulses for "laser standby" operation or – in Laser Mode 4 and Laser Mode 6 – the continuously-running laser signals for "laser active" and "laser standby" operation.
<b>Call</b>	<code>set_standby( HalfPeriod, PulseLength )</code>
<b>Parameters</b>	<p>HalfPeriod      <i>half of the standby output period.</i>                      As an unsigned 32-bit value.                      <i>1 bit equals 1/64 µs.</i>                      Allowed value range: [0...+(2<sup>32</sup>-1)].</p> <p>PulseLength     Pulse length of the standby pulses.                      As an unsigned 32-bit value.                      <i>1 bit equals 1/64 µs.</i>                      Allowed value range: [0...(2<sup>32</sup>-1)].</p>
	<ul style="list-style-type: none"> <li>After a hardware reset, (and after <a href="#">load_program_file</a>), the LASER1, LASER2 and LASERON ports must be first-time-activated with <a href="#">set_laser_control</a> before standby pulses can be activated by <a href="#">set_standby</a> or <a href="#">set_standby_list</a>, see <a href="#">Chapter 7.4 "Laser Control", page 173</a>. At the same time, the signal level of the standby pulses are set by <a href="#">set_laser_control</a>.</li> <li>The standby pulses are available in <i>all</i> laser modes (CO<sub>2</sub> Mode, YAG Mode 1, YAG Mode 2, YAG Mode 3, YAG Mode 5, Laser Mode 4 and Laser Mode 6).</li> <li>The standby pulses can be deactivated (turned off) by setting the standby pulse length and/or the standby output period to zero (default).</li> <li>With <code>HalfPeriod</code>, <i>half</i> the standby output period must be specified, see <a href="#">figure 48</a> and <a href="#">figure 50</a>.</li> <li>If <code>PulseLength</code> is larger than the output period (2 × <code>HalfPeriod</code>), the laser is permanently on.</li> <li>The laser mode is set with <a href="#">set_laser_mode</a>, see also <a href="#">Chapter 7.4 "Laser Control", page 173</a>. To set the active output period and pulse length for the "laser active" laser control signals (beside for Laser Mode 4 and Laser Mode 6), are set by <a href="#">set_laser_pulses_ctrl</a>, <a href="#">set_laser_pulses</a> or <a href="#">set_laser_timing</a>.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. In addition: increased value range.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified values by 8. The allowed value ranges decrease accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_standby_list</a> , <a href="#">get_standby</a>



<b>Delayed Short List Command</b>	<b>set_standby_list</b>
Function	Like <a href="#">set_standby</a> , but a list command.
Call	<code>set_standby_list( HalfPeriod, PulseLength )</code>
Parameters	<p>HalfPeriod      <i>Half of the stand-by output period.</i>            As an unsigned 32-bit value.  <i>1 bit equals 1/64 µs.</i>            Allowed value range: [0...+(2<sup>32</sup>-1)].</p> <p>PulseLength      <i>Pulse length of the stand-by pulses.</i>            As an unsigned 32-bit value.  <i>1 bit equals 1/64 µs.</i>            Allowed value range: [0...(2<sup>32</sup>-1)].</p>
RTC4→RTC6	See <a href="#">set_standby</a> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_standby</a>

<b>Ctrl Command</b>	<b>set_start_list</b>
Function	Opens the list memory for writing of list commands and sets the input pointer to the start of the specified list ("List 1" or "List 2"). The next list command is stored at this address and all further list commands at the subsequent addresses in the selected list.
Call	<code>set_start_list( ListNo )</code>
Parameters	<p>ListNo      Number of the list in which the input pointer should be set.            As an unsigned 32-bit value.            Allowed values: [uneven: "List 1", even: "List 2"].</p>
Comments	<ul style="list-style-type: none"> <li>• <code>set_start_list</code> is synonymous with <code>set_start_list_pos</code> for <code>Pos = 0</code>.</li> <li>• Alternatively, <code>set_start_list_1</code> and <code>set_start_list_2</code> (with no parameter) can be used.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">execute_list</a> , <a href="#">read_status</a> , <a href="#">set_start_list_pos</a>



<b>Ctrl Command</b>	<b>set_start_list_1</b>
<b>Function</b>	See <a href="#">set_start_list</a> .
<b>Call</b>	set_start_list_1()
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_start_list</a>

<b>Ctrl Command</b>	<b>set_start_list_2</b>
<b>Function</b>	See <a href="#">set_start_list</a> .
<b>Call</b>	set_start_list_2()
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">set_start_list</a>



<b>Ctrl Command</b>	<b>set_start_list_pos</b>
<b>Function</b>	Opens the list memory for writing of list commands and sets the input pointer to the specified (relative) position in the desired list ("List 1" or "List 2"). The next list command is stored at this address and all further list commands at the subsequent addresses in the selected list.
<b>Call</b>	<code>set_start_list_pos( ListNo, Pos )</code>
<b>Parameters</b>	<p>ListNo      Number of the list for which the input pointer should be set. As an unsigned 32-bit value. Allowed values: [uneven: "List 1", even: "List 2"].</p> <p>Pos          Position of the input pointer (offset relative to the start of the respective list). As an unsigned 32-bit value. Allowed value range: [0...(2<sup>23</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The selected list is unconditionally opened for loading. There is no checking of whether the list is currently being processed, see also <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a> and <a href="#">load_list</a>.</li> <li>The input pointer <i>cannot</i> be set to the protected list memory area "List 3". If the protected area is to be written to (at the full risk of users) with <code>set_start_list_pos</code>, then this area can be temporarily allocated to "List 2" by <a href="#">config_list( Mem1, -1 )</a>. After writing, configuration of the area's protection should be restored: <a href="#">config_list( Mem1, Mem2 )</a>.</li> <li>For uneven ListNo values, "List 1" is opened, otherwise "List 2". This facilitates continuous automatic list changing through incrementing counts.</li> <li>If "List 2" has not been assigned memory (<code>Mem2 = 0</code>, see <a href="#">config_list</a>) then "List 1" is opened.</li> <li>If Pos is specified as being larger than the memory area of the respective list (<code>Pos &gt; Mem1</code> or <code>Pos &gt; Mem2</code>), then Pos is set to 0.</li> <li>The status values of the selected list (see also <a href="#">read_status</a>) are set as follows: LOAD set, READY reset, USED reset. The LOAD status of the other corresponding list is reset.</li> <li>The <code>set_start_list_pos</code> command triggers a flush of the buffered list input (see <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a>).</li> <li><code>set_start_list_pos</code> also covers the specialized variants <a href="#">set_start_list_1</a>, <a href="#">set_start_list_2</a>, <a href="#">set_start_list</a> and <a href="#">set_input_pointer</a>.</li> <li><i>CAUTION: If the end of the respective list area is reached, the list input pointer is automatically reset to the start of the same list area.</i> <i>Make sure not to overwrite any commands still needed by your user program.</i></li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">execute_list_pos</a> , <a href="#">read_status</a>



<b>Ctrl Command</b>	<b>set_sub_pointer</b>
<b>Function</b>	Stores the absolute start address of a command list in the internal management table for indexed subroutines.
<b>Call</b>	<code>set_sub_pointer( Index, Pos )</code>
<b>Parameters</b>	Index      Index of the indexed subroutine whose starting address <code>Pos</code> should be entered in the management table. As an unsigned 32-bit value. Allowed value range: [0...1023]).
	Pos      Absolute start address. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>23</sup> -1)].
<b>Comments</b>	<ul style="list-style-type: none"> <li>If <code>Index &gt; 1023</code> and/or <code>Pos &gt; (2<sup>23</sup>-1)</code>, then <b>set_sub_pointer</b> is not executed (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>The <b>set_sub_pointer</b> command can be used for referencing a nonindexed subroutine, which thereby becomes an indexed subroutine that is protectable by <b>save_disk/load_disk</b> and/or callable by the index.</li> <li><b>set_sub_pointer</b> can also be used to reference anew an indexed subroutine, character or text string so that it can also be called by a second index. Here, it is preferable to use the <b>copy_dst_src</b> command for index management.</li> <li>The start addresses of command lists that are to be referenced with <b>set_sub_pointer</b> can be queried by <b>get_input_pointer</b> before loading the command lists.</li> <li><b>set_sub_pointer</b> only stores <i>starting addresses</i> in the internal management table. An indexed subroutine only gains protection by a subsequent <b>save_disk/load_disk</b> command.</li> <li><code>Pos</code> should not be an arbitrary address within a list. Instead, it should be the starting address of an actually existing subroutine that was finalized by <b>list_return</b> and does not contain <b>set_end_of_list</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>load_sub</b>



<b>Ctrl Command</b>	<b>set_text_table_pointer</b>
<b>Function</b>	Stores the absolute start address of a command list in the internal management table for indexed text strings.
<b>Call</b>	<code>set_text_table_pointer( Index, Pos )</code>
<b>Parameters</b>	<p>Index      Index of the indexed text string whose starting address <code>Pos</code> should be entered in the management table. As an unsigned 32-bit value. Allowed value range: [0...41]. The same assignment applies as for <b>load_text_table</b>:</p> <ul style="list-style-type: none"> <li>= 0...9:      Digits for marking the time and date [0...9].</li> <li>= 10...21:      Months [January...December].</li> <li>= 22...28:      Days-of-the-week [Sunday...Saturday].</li> <li>= 29:      Blank character for marking serial numbers.</li> <li>= 30...39:      Digits for marking serial numbers [0...9].</li> <li>= 40:      Text for "a.m.". </li> <li>= 41:      Text for "p.m.". </li> </ul> <p>Pos      Absolute start address. As an unsigned 32-bit value. Allowed value range: [0...(2<sup>23</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Indexed text strings can be used for marking time, date or serial numbers, see <a href="#">Section "Calling Indexed Text Strings", page 110</a>.</li> <li>• If <code>Index &gt; 41</code> and/or <code>Pos &gt; (2<sup>23</sup>-1)</code>, then <b>set_text_table_pointer</b> is not executed (<b>get_last_error</b> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>• <b>set_text_table_pointer</b> can be used for referencing a nonindexed subroutine, which thereby becomes an indexed text string that is protectable by <b>save_disk/load_disk</b> and/or callable by the index.</li> <li>• <b>set_text_table_pointer</b> can also be used to reference anew an indexed subroutine, character or text string so that it can also be called by a second index. Here, it is preferable to use the <b>copy_dst_src</b> command for index management.</li> <li>• The start addresses of command lists that are to be referenced with <b>set_text_table_pointer</b> can be queried by <b>get_input_pointer</b> before loading the command lists.</li> <li>• <b>set_text_table_pointer</b> only stores <i>starting addresses</i> in the internal management table. An indexed text string only gains protection by a subsequent <b>save_disk/load_disk</b> command.</li> <li>• <code>Pos</code> should not be an arbitrary address within a list. Instead, it should be the starting address of an actually existing subroutine that was finalized by <b>list_return</b> and does not contain <b>set_end_of_list</b>.</li> </ul>
RTC4→RTC6	New command.  <code>set_text_table_pointer</code> is synonymous with <b>set_char_table</b> , which is available for the RTC4 SCANalone Board (standalone version of the RTC4 board).
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">load_text_table</a> , <a href="#">mark_date</a> , <a href="#">mark_serial</a> , <a href="#">mark_time</a>



<b>Ctrl Command</b>	<b>set_timelag_compensation</b>						
<b>Function</b>	Compensates the various tracking errors of the xy axes and z axis.						
<b>Call</b>	<code>set_timelag_compensation( HeadNoXY, TimelagXY, TimelagZ )</code>						
<b>Parameters</b>	<table> <tr> <td>HeadNoXY</td> <td>Number of the xy scan head connector. As an unsigned 32-bit value. Allowed range [1...2].</td> </tr> <tr> <td>TimelagXY</td> <td>Tracking error of the xy axis in [10 µs]. As an unsigned 32-bit value. Allowed range [0...255].</td> </tr> <tr> <td>TimelagZ</td> <td>Tracking error of the z axis in [10 µs]. As an unsigned 32-bit value. Allowed range [0...255].</td> </tr> </table>	HeadNoXY	Number of the xy scan head connector. As an unsigned 32-bit value. Allowed range [1...2].	TimelagXY	Tracking error of the xy axis in [10 µs]. As an unsigned 32-bit value. Allowed range [0...255].	TimelagZ	Tracking error of the z axis in [10 µs]. As an unsigned 32-bit value. Allowed range [0...255].
HeadNoXY	Number of the xy scan head connector. As an unsigned 32-bit value. Allowed range [1...2].						
TimelagXY	Tracking error of the xy axis in [10 µs]. As an unsigned 32-bit value. Allowed range [0...255].						
TimelagZ	Tracking error of the z axis in [10 µs]. As an unsigned 32-bit value. Allowed range [0...255].						
<b>Comments</b>	<ul style="list-style-type: none"> <li>Unallowed parameter values produce a <code>get_last_error</code> return code of <code>RTC6_PARAM_ERROR</code> and <code>set_timelag_compensation</code> is not executed.</li> <li>If list execution is currently active, then the <code>get_last_error</code> return code gets set to <code>RTC6_BUSY</code> (the control command <code>get_status</code> returns <code>BUSY</code> or <code>INTERNAL-BUSY</code> or <code>PAUSED</code>) and the command is not executed.</li> <li>Faster axes are held back towards the slower axes.</li> <li><code>set_timelag_compensation</code> can also be used with scan heads of the excelliSCAN series. If <code>set_scanahead_params</code> configured the RTC6 board for controlling an excelliSCAN, then <code>TimelagXY</code> is ignored. Instead, the <code>PreviewTime</code> parameter value from <code>set_scanahead_params</code> gets applied. <code>set_timelag_compensation</code> automatically waits until a <code>HEAD_BUSY</code> gets reset (see "excelliSCAN scan heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" manual), hence execution of the command can last up to <code>PreviewTime</code>.</li> <li>With <code>SCANAhead systems</code>, the parameter <code>PreviewTime</code> from <code>set_scanahead_params</code> is automatically used as "tracking error".</li> </ul>						
RTC5→RTC6	New command.						
RTC5→RTC6	New command.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<code>set_scanahead_params</code> , <code>get_last_error</code> , <code>get_status</code>						



<b>Delayed Short List Command</b>	<b>set_trigger</b>
<b>Function</b>	Starts measurement value recording of the specified signals.
<b>Call</b>	<code>set_trigger( Period, Signal1, Signal2 )</code>
<b>Parameters</b>	<p><b>Period</b>    Measurement period (desired duration).                  As an unsigned 32-bit value.  <i>1 bit equals 10 µs.</i>                  Allowed value range: [0...(2<sup>31</sup>-1)].                  Bit#31 = 0: Automatic stop at the max. channel size, see <a href="#">set_trigger4</a>.                  Bit#31 = 1: Endless recording (circular buffer).</p> <p><b>Signal1</b>    Signal type for measurement channels 1 and 2. As unsigned 32-bit values.                  Allowed value range: as listed below.</p> <ul style="list-style-type: none"> <li>= 0: LASERON signal.            1 = laser signal on, 0 = laser signal off.</li> <li>= 1: StatusAX.            Status signal of x axis, first scan head connector.</li> <li>= 2: StatusAY.            Status signal of y axis, first scan head connector.</li> <li>= 3: Reserved.</li> <li>= 4: StatusBX.            Status signal of x axis, second scan head connector.</li> <li>= 5: StatusBY.            Status signal of y axis, second scan head connector.</li> <li>= 6: Reserved.</li> <li>= 7: SampleX (up to 4 in <a href="#">7.3.6, page 170</a>).            Cartesian control value for the x axis.</li> <li>= 8: SampleY (up to 4 in <a href="#">7.3.6, page 170</a>).            Cartesian control value for the y axis.</li> <li>= 9: SampleZ (up to 4 in <a href="#">7.3.6, page 170</a>).            Cartesian control value for the z axis.</li> <li>= 10: SampleAX_Corr (up to 8 in <a href="#">7.3.6, page 170</a>).            Corrected x axis control value for the first scan head connector.</li> <li>= 11: SampleAY_Corr (up to 8 in <a href="#">7.3.6, page 170</a>).            Corrected y axis control value for the first scan head connector.</li> <li>= 12: SampleAZ_Corr (up to 8 in <a href="#">7.3.6, page 170</a>).            Corrected z axis control value, if xy are connected to the first scan head connector. Identical to the effective output value for the z axis.</li> </ul>



Delayed Short List Command	set_trigger
Parameters (cont'd)	<p>Signal1 = 13: SampleBX_Corr (up to 8 in <a href="#">7.3.6, page 170</a>).            Corrected control value for the x axis, second scan head connector.</p> <p>= 14: SampleBY_Corr (up to 8 in <a href="#">7.3.6, page 170</a>).            Corrected control value for the y axis, second scan head connector.</p> <p>= 15: SampleBZ_Corr (up to 8 in <a href="#">7.3.6, page 170</a>).            Corrected z axis control value, if xy are connected to the second scan head connector. Identical to the effective output value for the z axis.</p> <p>= 16: StatusAX+LASERON.            StatusAX for laser signal on, -524,288 for laser signal off.</p> <p>= 17: StatusAY+LASERON.            StatusAY for laser signal on, -524,288 for laser signal off.</p> <p>= 18: StatusBX+LASERON.            StatusBX for laser signal on, -524,288 for laser signal off.</p> <p>= 19: StatusBY+LASERON.            StatusBY for laser signal on, -524,288 for laser signal off.</p> <p>= 20: SampleAX_Out (up to 9 in <a href="#">7.3.6, page 170</a>).            Actual output value for the x axis, first scan head connector. If applicable incl. any scanner offset and gain compensation, see comments.            Not usable for z axis output values.</p> <p>= 21: SampleAY_Out (up to 9 in <a href="#">7.3.6, page 170</a>).            Actual output value for the y axis, first scan head connector.            Not usable for z axis output values.</p> <p>= 22: SampleBX_Out (up to 9 in <a href="#">7.3.6, page 170</a>).            Actual output value for the for the x axis, second scan head connector.            Not usable for z axis output values.</p> <p>= 23: SampleBY_Out (up to 9 in <a href="#">7.3.6, page 170</a>).            Actual output value for the y axis, second scan head connector.            Not usable for measuring z axis output values.</p> <p>= 24: Laser control parameter of "Automatic Laser Control".            See <a href="#">set_auto_laser_control</a>.</p>



<b>Delayed Short List Command</b>	<b>set_trigger</b>
Parameters (cont'd)	<p>Signal1 = 25: SampleAX_Trans (up to 7 in 7.3.6, page 170).      Transformed control value for the x axis, first scan head connector.</p> <p>= 26: SampleAY_Trans (up to 7 in 7.3.6, page 170).      Transformed control value for the y axis, first scan head connector.</p> <p>= 27: SampleAZ_Trans (up to 7 in 7.3.6, page 170).      Transformed z axis control value, if xy are connected to the first scan head connector.</p> <p>= 28: SampleBX_Trans (up to 7 in 7.3.6, page 170).      Transformed control value for the x axis, second scan head connector.</p> <p>= 29: SampleBY_Trans (up to 7 in 7.3.6, page 170).      Transformed control value for the y axis, second scan head connector.</p> <p>= 30: SampleBZ_Trans (up to 7 in 7.3.6, page 170).      Transformed z axis control value, if xy are connected to the second scan head connector.</p> <p>= 31: Laser control parameter of "vector-controlled laser control".      See <a href="#">set_vector_control</a>.</p> <p>= 32: Focus shift. See <a href="#">set_vector_control</a>, <a href="#">set_defocus</a>, <a href="#">set_defocus_list</a>.</p> <p>= 33: 12-bit output value at the ANALOG OUT1 output port.      See <a href="#">set_vector_control</a> and <a href="#">Chapter 9.1.4 "12-Bit Analog Output Ports"</a>.</p> <p>= 34: 12-bit output value at the ANALOG OUT2 output port.      See <a href="#">set_vector_control</a> and <a href="#">Chapter 9.1.4 "12-Bit Analog Output Ports"</a>.</p> <p>= 35: Output value at the 16-bit digital output port.      See <a href="#">set_vector_control</a> and <a href="#">Chapter 9.1.1 "16-Bit Digital Output Port"</a>.</p> <p>= 36: Output value at the 8-bit digital output port.      See <a href="#">set_vector_control</a> and <a href="#">Chapter 9.1.2 "8-Bit Digital Output Port"</a>.</p> <p>= 37: Pulse length (PulseLength) of the LASER1 and LASER2 laser signals.      See <a href="#">set_vector_control</a>.</p> <p>= 38: Output period (HalfPeriod) of the LASER1 and LASER2 laser signals.      See <a href="#">set_vector_control</a>.</p>



<b>Delayed Short List Command</b>	<b>set_trigger</b>
Parameters (cont'd)	<p>Signal1 = 39: FreeVariable0.</p> <p>(cont'd)</p> <p>= 40: FreeVariable1.</p> <p>= 41: FreeVariable2.</p> <p>= 42: FreeVariable3.</p> <p>= 43: Counter value of encoder counter "Encoder0".</p> <p>= 44: Counter value of encoder counter "Encoder1".</p> <p>= 45: Marking speed. From <a href="#">set_mark_speed</a>, <a href="#">set_mark_speed_ctrl</a>.</p> <p>= 46: 16-bit digital input (EXTENSION 1).</p> <p>= 47: Only for intelliWELD II: Zoom value.</p> <p>= 48: FreeVariable4.</p> <p>= 49: FreeVariable5.</p> <p>= 50: FreeVariable6.</p> <p>= 51: FreeVariable7.</p> <p>= 52: Time stamp counter. See <a href="#">Chapter 8.12 "Time Measurements", page 267</a>.</p> <p>= 53: Wobbel amplitude. See <a href="#">set_wobbel</a>, <a href="#">set_wobbel_mode</a> and <a href="#">Chapter 8.4 "Wobbel Mode", page 218</a>.</p> <p>= 54: ReadAnalogIn. See <a href="#">read_analog_in</a>.</p> <p>= 55: Scaled encoder value for X.</p> <p>= 56: Scaled encoder value for Y.</p> <p>= 57: Scaled encoder value for Z.</p> <p>= 58: RS-232.</p> <p>= 59: <a href="#">read_mcbsp( 0 )</a></p> <p>= 60: <a href="#">read_multi_mcbsp( 0 )</a></p> <p>= 61: <a href="#">read_multi_mcbsp( 1 )</a></p> <p>= 62: <a href="#">read_multi_mcbsp( 2 )</a></p>
Signal2	Like Signal1.

<b>Delayed Short List Command</b>	<b>set_trigger</b>
Comments	<p><i>General Comments</i></p> <ul style="list-style-type: none"> <li>If Signal1 and Signal2 are not from the above list, then <b>set_trigger</b> is replaced with a <b>list_nop</b>, even if Period = 0 (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>After a <b>set_trigger</b> with Period &gt; 0, a data pair is recorded immediately and subsequently at intervals defined by the specified measurement period.       <ul style="list-style-type: none"> <li>Period (Bit #31 = 0):           <ul style="list-style-type: none"> <li>The measurement value recording ends automatically with <math>2^{24}</math> data pairs (if Signal2 is invalid, even with <math>2^{25}</math> data entries, see also <b>set_trigger4</b>).</li> <li>It ends earlier, if one of the following occurs:               <ul style="list-style-type: none"> <li>- <b>set_trigger</b> (Period = 0) is called</li> <li>- <b>set_trigger4</b>(Period = 0) is called</li> </ul> </li> </ul> </li> <li>Period (Bit #31 = 1):           <ul style="list-style-type: none"> <li>Starts an endless data recording according Period = measurement period   0x80000000" with 0 &lt; measurement period &lt; <math>2^{31}-1</math>. In this case, the data recording does not end automatically. Instead, it starts from the beginning each time and overwrites previously recorded data (ring buffer). In the meantime, the data can be read out with <b>get_waveform_offset</b> from any location and in any packets.</li> <li><b>measurement_status</b> also provides information about the current status of the measurement value recording.</li> </ul> </li> </ul> </li> <li>A measurement value recording started by <b>set_trigger</b> occurs only during the execution of a list.</li> <li>Given the measurement value recording has not been terminated by <b>set_trigger</b> (Period = 0) or <b>set_trigger4</b>(Period = 0), it is automatically continued with the start of a new list (the status of the measurement value recording is not reset by <b>set_end_of_list</b>). During the execution of a list, the status is either reset by <b>set_trigger</b> (Period = 0) or <b>stop_execution</b>. By <b>stop_trigger</b> the status can be reset if no list is executed.</li> <li>Sample values and status values returned by the scan system and stored on the RTC6 by <b>set_trigger</b> (Signal1, Signal2 = 1...23, 25...30) are always in the RTC6's 20-bit range, even if the <b>RTC6 DLL</b> is set to <b>RTC4 Compatibility Mode</b>. The measured and stored values can be subsequently transferred to the PC by the <b>get_waveform</b> command for evaluation. We recommend explicitly ending the measurement session before reading the data. The measured values are transferred to the PC by <b>get_waveform</b> as 32-bit data and must be evaluated accordingly by users (see comments for <b>get_value</b>).</li> <li>The current status of a measurement value recording can be queried by <b>measurement_status</b>.</li> <li>For aborting a measurement value recording by <b>set_trigger</b> (Period = 0) or <b>set_trigger4</b>(Period = 0), the values of Signal1 and Signal2 are irrelevant.</li> </ul>

<b>Delayed Short List Command</b>	<b>set_trigger</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If you abort a measurement session with <b>set_trigger</b> (<code>Period = 0</code>) or <b>set_trigger4</b>(<code>Period = 0</code>), then previously recorded measurement values are <i>not</i> lost and the measurement pointer halts at its most recent value. This allows subsequent querying by <b>measurement_status</b> of the number of data entries. In contrast, if a measurement session is newly started with <b>set_trigger</b> (<code>Period &gt; 0</code>) or <b>set_trigger4</b>(<code>Period &gt; 0</code>), the measurement pointer is reset and the measurements obtained thus far are overwritten. It is not possible to resume an explicitly or automatically halted measurement session.</li> </ul> <p><i>Comments on the Data</i></p> <ul style="list-style-type: none"> <li>The type of scan system being used determines which status signals are generated and returned by the status channels. Specific information can be found in your scan system's operating manual. The <b>control_command</b> command can be used with iDRIVE scan systems (see glossary entry on <a href="#">page 879</a>) to specify which information are returned on the status channels.</li> <li>If the scan system has only one status channel, then only the X measurement signal contains meaningful data.</li> <li>With 3D scan systems, the output and status values of the z axis are transmitted over the scan head connector's channel to which the Z axis is attached (if correspondingly configured by <b>select_cor_table</b>). Example: If the Z axis is attached to the second scan head connector's X channel (<b>select_cor_table</b>([1...8], 0)), then its status signal can be queried by StatusBX (Signal1/Signal2 = 4).</li> <li>The signals 12 and 15 as well as 27 and 30 are identical, each: <code>SampleAZ_Corr = SampleBZ_Corr</code> and <code>SampleAZ_Trans = SampleBZ_Trans</code>.</li> <li>The status signals Status&lt;...&gt; lag the control signals Sample&lt;...&gt; by a few clock cycles. See also <a href="#">Section "Reading Out Data", page 200</a>.</li> <li>Signal1, Signal2 = 0, 24, 31...42 and 47...51 enables logging of values that were outputted during the previous clock cycle. During the current clock cycle, the corresponding signals may change after logging, either automatically or by control and list commands.</li> <li>Signal1, Signal2 = 24 enables logging of the signal parameter that is outputted by "Automatic Laser Control" (see <b>set_auto_laser_control</b>). Logging only occurs when "Automatic Laser Control" has been activated and the laser is on. For switched-off lasers, 0 is logged.</li> <li>Signal1, Signal2 = 31 enables logging of the signal parameter specified for "vector-controlled laser control" (see <b>set_vector_control</b>). Logging is also possible without executing <b>[*]para[*]</b> commands, but then the signal values remain unchanged.</li> <li>Pulse length and output period (Signal1, Signal2 = 37, 38) are only logged for "Laser active" laser control signals (not standby signals).</li> <li>For Signal1, Signal2 = 39...42 and 48...51, see <a href="#">Chapter 6.9.1 "Free Variables", page 124</a>.</li> </ul>



<b>Delayed Short List Command</b>	<b>set_trigger</b>
RTC4→RTC6	Unchanged basic functionality (measurement storage in 10 µs clock period), however: <ul style="list-style-type: none"> <li>The measurement session can be aborted by <code>Period = 0</code>.</li> <li>The signals <code>StatusAZ</code> and <code>StatusBZ</code> (<code>Signal1/Signal2 = 3, 6</code>) are no longer available.</li> <li>All coordinate values are referred to (0 0 0) (value range [-2<sup>19</sup>...(2<sup>19</sup>-1)]) and are no longer zero point shifted.</li> </ul>
RTC5→RTC6	Basically unchanged functionality. Increased memory area for data recordings by merging channels, see <a href="#">set_trigger4</a> .
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 610, OUT 610, RBF 615: endless recording.
References	<a href="#">get_waveform</a> , <a href="#">get_waveform_offset</a> , <a href="#">measurement_status</a> , <a href="#">control_command</a> , <a href="#">get_value</a> , <a href="#">get_values</a> , <a href="#">set_mcbsp_out</a> , <a href="#">set_mcbsp_out_ptr</a> , <a href="#">set_trigger4</a>



<b>Delayed Short List Command</b>	<b>set_trigger4</b>
<b>Function</b>	Starts the measurement value recording of the specified measurement signals (has the same effect as <b>set_trigger</b> , but with up to 4 simultaneous measurement signals).
<b>Call</b>	<code>set_trigger4( Period, Signal1, Signal2, Signal3, Signal4 )</code>
<b>Parameters</b>	Period      Like <b>set_trigger</b> .
	Signal1    Like <b>set_trigger</b> .
	Signal2    Like <b>set_trigger</b> .
	Signal3    Like <b>set_trigger</b> .
	Signal4    Like <b>set_trigger</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See comments for <b>set_trigger</b>.</li> <li><math>2^{23}</math> entries per measurement channel are available with <b>set_trigger4</b>, if all 4 signals are within the allowed range (see <b>set_trigger</b>). The measurement value recording ends automatically with <math>2^{23}</math> data quadruplets.</li> <li>If Signal 3 and Signal 4 are not allowed (see <b>set_trigger</b>), <b>set_trigger4</b> is synonymous with <b>set_trigger</b>. The memory areas of Signal 3 and Signal 4 are added to the memory areas of Signal 1 and Signal 2. This allows recording of <math>2^{24}</math> data pairs. The measured value recording ends automatically with <math>2^{24}</math> data pairs.</li> <li>If only Signal 1 is within the permissible range, the memory areas of all 4 channels are merged into a single one. With this up to <math>2^{25}</math> data values can be recorded. The measured value recording ends automatically at <math>2^{25}</math> data values.</li> <li><b>set_trigger</b>(<code>Period = 0</code>) and <b>set_trigger4</b>(<code>Period = 0</code>) both terminate active measurement value recording regardless of which command has started it.</li> <li>If <b>set_trigger</b>(<code>Period &gt; 0</code>) or <b>set_trigger4</b>(<code>Period &gt; 0</code>) is used to start a new measurement value recording, then the measurement value counter is reset. Existing data is overwritten with new measurement values. See also the corresponding comments for <b>set_trigger</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Basically unchanged functionality. Increased memory area for data recordings by merging unused channels, endless recording.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 610, OUT 610, RBF 615: see <b>set_trigger</b> .
<b>References</b>	<b>set_trigger</b> , <b>stop_trigger</b> , <b>get_waveform</b> , <b>get_waveform_offset</b>



<b>Undelayed Short List Command</b>	<b>set_vector_control</b>	
<b>Function</b>	Initializes or deactivates “vector-controlled laser control”. See <a href="#">Section “Vector-Defined Laser Control”, page 195.</a>	
<b>Call</b>	set_vector_control( <i>Ctrl</i> , <i>Value</i> )	
<b>Parameters</b>	<i>Ctrl</i>	<p>Control parameter for initializing or deactivating “vector-controlled laser control” (for <i>Ctrl</i> = 1...6: identical with <a href="#">set_auto_laser_control</a>). As an unsigned 32-bit value.</p> <p>= 1...7: Defines which signal parameter is to be varied by “vector-controlled laser control”.</p> <ul style="list-style-type: none"> <li>= 1: 12-bit output value at the ANALOG OUT1 output port.</li> <li>= 2: 12-bit output value at the ANALOG OUT2 output port.</li> <li>= 3: Output value at the 8-bit digital output port.</li> <li>= 4: Pulse length (<i>PulseLength</i>) of the laser signals LASER1 and LASER2.</li> <li>= 5: Output period (<i>HalfPeriod</i>) of the laser signals LASER1 and LASER2.</li> <li>= 6: Output value at the 16-bit digital output.</li> <li>= 7: Focus shift (“Defocus”).</li> </ul> <p>= 0 or &gt; 7: deactivates “vector-controlled laser control” (for <i>Ctrl</i> &gt; 7: <a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</p>
	<i>Value</i>	<p>Defines the starting value for the parameter selected by <i>Ctrl</i>. As an unsigned 32-bit value.</p> <p>Allowed values (out-of-range values are clipped to the boundary values):</p> <ul style="list-style-type: none"> <li>For <i>Ctrl</i> = 1/2: 12-bit values [0...4095].</li> <li>For <i>Ctrl</i> = 3: 8-bit values [0...255].</li> <li>For <i>Ctrl</i> = 4: [0...(2<sup>32</sup>-1)]. 1 bit equals 1/64 µs.</li> <li>For <i>Ctrl</i> = 5: [0...(2<sup>32</sup>-1)]. 1 bit equals 1/64 µs.</li> <li>For <i>Ctrl</i> = 6: 16-bit values [0...(2<sup>16</sup>-1)].</li> <li>For <i>Ctrl</i> = 7: [-524,288...+524,287], Value = focus shift + 524,288.</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• If <i>Ctrl</i> is an invalid value, then “vector-controlled laser control” is deactivated (<i>Ctrl</i> = 0, initialization state). Otherwise, <i>Value</i> is applied as the starting value of the next <a href="#">[*]para[*]</a> command. If <i>Value</i> is invalid, then it is clipped to the maximum allowed value.</li> <li>• <a href="#">set_vector_control</a> only affects <a href="#">[*]para[*]</a> commands.</li> <li>• If an “Automatic Laser Control” has been activated by <a href="#">set_auto_laser_control</a> with the same control parameter (<i>Ctrl</i> = 1...6), then <a href="#">set_vector_control</a> sets the 100% value of the “Automatic Laser Control” to value <i>Value</i>.</li> </ul>	



<b>Undelayed Short List Command</b>	<b>set_vector_control</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>For Ctrl = 7, the focus shift is linearly varied as with <b>set_defocus</b> and <b>set_defocus_list</b>, see comments there) with <b>[*]para[*]</b> commands (in order for the z outputs to be outputted, the Option "3D" must be enabled and a 3D correction file must be loaded and assigned as well) up to the specified end value (this requires enabling the Option "3D" as well as loading and assigning a 3D correction file). If, for the first <b>[*]para[*]</b> command, the currently set focus shift does not match the initial value (Value) specified by <b>set_vector_control</b>, then the command begins with a "Hard jump" (in the z output) to Value. The setting defined by <b>set_defocus</b> or <b>set_defocus_list</b> is lost.</li> <li>Unlike signed values in the range [-524,288...+524,287] for <b>set_defocus</b> and <b>set_defocus_list</b>, the focus shift (Value) specified for <b>set_vector_control</b> must be an unsigned number shifted upward by 524,288: Value = focus shift + 524,288.</li> <li>"Automatic Laser Control" should not be combined with the variable laser power of <b>set_multi_mcbsp_in</b> or the "freely definable wobble shape".</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>This command has no <b>RTC4 Compatibility Mode</b> for Value.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies with Ctrl = 7 the specified value for Value by 16. The allowed value range decreases accordingly.</p>
RTC5→RTC6	<p>Unchanged functionality.</p> <p>In <b>RTC5 Compatibility Mode</b>, the RTC6 multiplies with Ctrl = 7 the specified value for Value by 16. The allowed value range decreases accordingly.</p>
Version info	<p>Available as of version DLL 600, OUT 600, RBF 600.</p> <p>Last change version DLL 614:</p>
References	<b>set_auto_laser_control</b>



<b>Ctrl Command</b>	<b>set_verify</b>
Function	Activates or deactivates a download verification. See <a href="#">Chapter 6.8.1 "Download Verification", page 121.</a>
Call	OldVerify = set_verify( Verify )
Parameters	Verify      Setting parameter. As an unsigned 32-bit value. = 0:      Verification is deactivated. > 0:      Verification is activated.
Result	The Verify setting parameter that was active before calling <code>set_verify</code> . As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li>If verification is activated, the download times are extended.</li> <li>Verification of correction file downloads only works if the correction file contains a checksum (otherwise the <code>get_last_error</code> return code <code>RTC6_VERIFY_ERROR</code> is generated). To ensure that a checksum is present (which is not the case for older ct5 correction files), you should test your correction file prior to loading by using the control command <code>verify_checksum</code>. If the correction file does not contain a check sum, then <code>verify_checksum</code> enters it.</li> <li><code>set_verify</code> is available even without explicit access rights to a specific RTC6 board. These commands only change settings in the <code>RTC6 DLL</code> of the specified (or default) board. They have no effect on the board itself.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <code>set_verify</code>.</li> <li>Activation of the download verification by <code>set_verify</code> can generate the <code>get_last_error</code> return code <code>RTC6_OUT_OF_MEMORY</code>, if a <code>RTC6 DLL</code>-internal Windows memory request fails. In this case, the download verification remains deactivated.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_error</a> , <a href="#">get_last_error</a> , <a href="#">verify_checksum</a>



<b>Normal List Command</b>	<b>set_wait</b>
<b>Function</b>	Inserts a numbered break point ("wait marker") into the list.
<b>Call</b>	<code>set_wait( WaitWord )</code>
<b>Parameters</b>	<p>WaitWord      Number of the break point.            As an unsigned 32-bit value.            Allowed value range: [1...(2<sup>32</sup>-1)].            0 is corrected to 1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The list processing is interrupted at each break point and remains halted until continued by <code>release_wait</code>. This provides a way to implement synchronizations.</li> <li>The signals for "laser active" operation are switched off by <code>set_wait</code> and a home jump defined by <code>home_position</code> or <code>home_position_xyz</code> might be executed (the <code>INTERNAL-BUSY</code> status is set while the home jump is executed). Continuation by <code>execute_list_pos</code>, <code>restart_list</code> or an external start is consequently not possible. However, <code>stop_execution</code> or an external stop is possible. <code>release_wait</code>, <code>stop_execution</code> or an external stop removes suppression of the list start.</li> <li><code>set_wait</code> sets the <code>PAUSED</code> status and resets the <code>BUSY</code> status (both queryable with <code>get_status</code>). The opposite occurs with a subsequent <code>release_wait</code>, see also <a href="#">Chapter 6.4.3 "List Execution Status", page 98</a>.</li> <li>If processing has been stopped at a break point, then <code>get_wait_status</code> returns the number of this break point.</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: Additional <code>PAUSED</code> status which is set by <code>set_wait</code> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_wait_status</a> , <a href="#">release_wait</a> , <a href="#">pause_list</a> , <a href="#">stop_list</a>

<b>Delayed Short List Command</b>	<b>set_wobbel</b>
<b>Function</b>	Defines the parameters for an ellipse-shaped wobbel motion. "Wobbel" is a motion of the output position which is added to the regular marking movement. See <a href="#">Chapter 8.4 "Wobbel Mode", page 218</a> .
<b>Call</b>	<code>set_wobbel( Transversal, Longitudinal, Freq )</code>
<b>Parameters</b>	<p><b>Transversal</b> Amplitude of the elliptical <i>perpendicular</i> to the momentary direction of motion or to the one defined by <a href="#">set_wobbel_direction</a>. In bits.            As an unsigned 32-bit value.            Allowed value range: [0...131.071 (=<math>2^{17}</math>-1)].            Larger values are clipped.</p>
	<p><b>Longitudinal</b> Amplitude of the elliptical <i>parallel</i> to the momentary direction of motion or to the one defined by <a href="#">set_wobbel_direction</a>. In bits.            As an unsigned 32-bit value.            Allowed value range: [0...131.071 (=<math>2^{17}</math>-1)].            Larger values are clipped.</p>
	<p><b>Freq</b> Frequency of the wobbel movement in Hz (number of ellipses per second).            As a 64-bit IEEE floating point value.            Allowed value range: [-6000...6000].            Larger values are clipped.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The Wobbel mode can be used for marking lines with various line widths. An ellipse-shaped movement is added to the linear marking movement, resulting in a spiral movement of the laser focus in the image field. Alternatively, a figure-of-8 wobbel shape (horizontal or vertical to the direction of motion) can be activated by <a href="#">set_wobbel_mode</a>. The line width can be set by appropriate values for the amplitude and frequency (frequency and mark speed should be coordinated, see <a href="#">Chapter 8.4.1 "Wobbel Shapes – Important Notes on Choosing Appropriate Parameter Values", page 220</a>). For arcs, too, the wobbel motion follows the current marking direction (exception: see below). Therefore, independently of the Cartesian angle, the effective line width is always the same.</li> <li>The Wobbel mode is terminated by setting both amplitudes and/or the frequency to zero (more accurate for <math>-n \leq Freq \leq n</math> with <math>n = 50000/65536 = 0.7629\dots</math>).</li> <li>The frequency is signed. The wobbel vector rotates clockwise for positive values and counterclockwise for negative values. Thus, the inner and outer point densities of arcs can be individually set independently of their actual direction of rotation.</li> <li>At the beginning of a marking, (after <a href="#">set_wobbel</a> or a jump), the wobbel start point is generally set for the same value relative to the vector/arc startpoint; it is repeatedly continued, however, for <a href="#">Polylines</a> (including arcs).</li> </ul>



<b>Delayed Short List Command</b>	<b>set_wobbel</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>For identical amplitudes (Longitudinal = Transversal), the wobbel startpoint is permanently referenced to the coordinate system, that is, independent of the current direction of motion. By <b>set_wobbel_direction</b>, a fixed reference direction can be set which differs from it.</li> <li>For an angle, ellipse-shaped wobbel movements can result in more or less small jumps after reaching the corner, depending on the current wobbel phase. This does not occur (= "rounded corners") if the wobbel motion is exactly circular (Longitudinal = Transversal).</li> <li>Longitudinal = 0 produces a sine-shaped wobbel motion across the direction of movement.</li> <li>When defining the wobbel shape and its frequency take the dynamics of the scan head and laser into account. Otherwise, an overheating and even a permanent damage of the system may occur, see <b>Chapter 8.4.1 "Wobbel Shapes – Important Notes on Choosing Appropriate Parameter Values"</b>, page 220.</li> <li>The present wobbel amplitude can be recorded by <b>set_trigger/set_trigger4</b> (signal 53). The format of the data is ((transversal &lt;&lt; 16) + longitudinal).</li> </ul>
RTC4→RTC6	<p>Basically unchanged functionality.</p> <p>More wobbel shape: elliptical, direction dependent adjustable, see also <b>set_wobbel_mode</b>.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified value for Longitudinal and Transversal by 16. The allowed value ranges decrease accordingly.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_mark_speed, set_wobbel_mode, set_wobbel_direction</b>



<b>Undelayed Short List Command</b>	<b>set_wobbel_control</b>
<b>Function</b>	Specifies laser control parameters for laser power variation with "freely definable wobbel shapes".
<b>Call</b>	<code>set_wobbel_control( Ctrl, Value, MinValue, MaxValue )</code>
<b>Parameters</b>	<p><b>Ctrl</b> Control parameter for initializing or deactivating laser power variation. As an unsigned 32-bit value.</p> <p>= 1...6: Defines which signal parameter to vary. On the meaning, see <a href="#">set_auto_laser_control</a>.</p> <p>= 0 or &gt; 6: Deactivates laser power variation (for Ctrl &gt; 6: <a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</p> <p><b>Value</b> Nominal laser power P0 (100%). As an unsigned 32-bit value. Allowed value range: see <a href="#">set_auto_laser_control</a>; the maximum is [0...65,535] or 0xFFFFFFFF. Excessive values are clipped.</p> <p><b>MinValue</b> Limit that cannot be exceeded, see <a href="#">set_auto_laser_control</a>. As an unsigned 32-bit value. The allowed value range depends on the selected Ctrl parameter and on Value.</p> <p><b>MaxValue</b> See <a href="#">MinValue</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Any still-pending delayed short list command executes first.</li> <li>For Ctrl = 0 and Ctrl &gt; 6, laser power variation is switched off (initialization state after <a href="#">load_program_file</a>). The values for <a href="#">McBSP</a> multi transfers are then not used.</li> <li>The limits for Value, minValue, maxValue are the same as for "Automatic Laser Control" (see <a href="#">set_auto_laser_control</a>), but with a maximum of [0...65,535]. Initialized values are minValue = 0 and maxValue = 0xFFFFFFFF, that is, no restrictions.</li> <li>The laser power can be combined with the "vector-controlled laser control", if the corresponding Ctrl parameters have been chosen identically.</li> <li>If you want variable laser power along a wobbel shape, then <a href="#">set_wobbel_control</a> must execute before you activate the "freely definable wobbel shape" by <a href="#">set_wobbel_mode</a>. Otherwise, laser power is not varied, even if you make a change in the data sets.</li> </ul>



<b>Undelayed Short List Command</b>	<b>set_wobbel_control</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Special case: if <code>Value</code> = 0xFFFFFFFF, then the nominal laser power is derived from the port assigned by the signal parameter <code>Ctrl</code>, instead of from the command. This is then the current content at this timepoint set by other normal commands, for example, <a href="#">write_da_1_list</a> for <code>Ctrl</code> = 1. These – and (should the situation arise) other queued delayed short list commands – are executed prior to <code>set_wobbel_control</code>. But beware: with execution of a “freely definable wobbel shape”, the value at the port can change at any time. Subsequent calls of <code>set_wobbel_control</code> then return other values for the nominal laser power.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_wobbel_vector</a> , <a href="#">set_multi_mcbsp_in</a> , <a href="#">set_multi_mcbsp_in_list</a>



<b>Undelayed Short List Command</b>	<b>set_wobbel_direction</b>
Function	Defines a direction vector for wobbel motions.
Call	set_wobbel_direction( dx, dy )
Parameters	dx      x component of the direction vector. In bits. As a signed 32-bit value.
	dy      y component of the direction vector. In bits. As a signed 32-bit value.
Comments	<ul style="list-style-type: none"> <li>For non-zero direction vectors (dx and/or dy non-zero), wobbel motion (longitudinal and transversal) is relative to <i>this</i> direction vector instead of to the momentary direction vector. The direction vector's length is inconsequential. The RTC6 normalizes the direction vector.</li> <li>If dx = dy = 0, then the function is deactivated.</li> <li>The direction vector setting remains in effect even after the Wobbel mode is switched off by <b>set_wobbel</b> or <b>set_wobbel_mode</b>, and continues being used if you switch the Wobbel mode back on.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_wobbel</b> , <b>set_wobbel_direction</b> , <b>set_wobbel_mode</b> , <b>set_wobbel_offset</b>



<b>Delayed Short List Command</b>	<b>set_wobbel_mode</b>	
<b>Function</b>	Switches the Wobbel mode on and off and defines the parameters for an ellipse-shaped or figure-of-8 wobbel motion, see <a href="#">Chapter 8.4 "Wobbel Mode", page 218</a> .	
<b>Call</b>	set_wobbel_mode( Transversal, Longitudinal, Freq, Mode )	
<b>Parameters</b>	<p>Transversal      Amplitude of the elliptical or figure-8 movement <i>perpendicular</i> to the momentary direction of motion or to the one defined by <a href="#">set_wobbel_direction</a>. In bits.            As an unsigned 32-bit value.            Allowed value range: [0...131.071 (=<math>2^{17}</math>-1)].            Excessive values are clipped.</p> <p>Longitudinal      Amplitude of the elliptical or figure-8 movement <i>parallel</i> to the momentary direction of motion or to the one defined by <a href="#">set_wobbel_direction</a>. In bits.            As an unsigned 32-bit value.            Allowed value range: [0...131.071 (=<math>2^{17}</math>-1)].            Excessive values are clipped.</p> <p>Freq      Frequency of the wobbel movement in <i>Hz</i> (number of ellipses or figure-of-8s per second).            As a 64-bit IEEE floating point value.            Allowed value range: [-6000...6000].            Larger values are clipped.</p> <p>Mode      Defines the wobbel shape.            As a signed 32-bit value.            = 0:      Ellipse-shaped wobbel movement.            &lt; 0:      Figure-of-8 wobbel movement perpendicular to the motion direction (vertical 8).            = 1:      Figure-of-8 wobbel movement parallel to the motion direction (horizontal 8).            &gt; 1:      "Freely definable wobbel shape", see <a href="#">set_wobbel_vector</a>.</p>	
<b>Comments</b>	<ul style="list-style-type: none"> <li>If Mode = 0, <a href="#">set_wobbel_mode</a> functions identically to <a href="#">set_wobbel</a> (see comments there).</li> <li>If Mode ≠ 0, then a figure-of-8 motion is activated. With figure-of-8s, broader mid-line processing can be achieved by appropriate parameter values. If the specified transverse and longitudinal amplitudes are identical, then the wobbel shape remains stationary in space; otherwise the orientation of the wobbel shape follows the current direction of motion. If the command executes while a list contains a "freely definable wobbel shape" (see <a href="#">set_wobbel_vector</a> and <a href="#">Chapter 8.4 "Wobbel Mode", page 218</a>), then Mode &gt; 1 selects this shape, otherwise the horizontal figure-8 shape is selected similarly to Mode = 1. If you subsequently define a wobbel shape and the Wobbel mode has been activated with the parameter Mode &gt; 1, then a switch from the horizontal figure-8 to the wobbel shape automatically occurs. But beware: it is then not possible to vary the power along the wobble figure, see <a href="#">set_wobbel_control</a>.</li> </ul>	

<b>Delayed Short List Command</b>	<b>set_wobbel_mode</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The Wobbel mode is terminated by setting both amplitudes and/or the frequency to zero (more accurate for <math>-n \leq \text{Freq} \leq n</math> with <math>n = 50000/65536 = 0.7629\dots</math>).</li> <li>The frequency is signed. During the figure-of-8's first loop, the wobbel vector rotates clockwise for positive values and counterclockwise for negative values. Thus, (especially for ellipse-shaped wobbel motions), the inner and outer point densities of arcs can be individually set independently of their actual direction of rotation.</li> <li>At the beginning of a marking, (after <b>set_wobbel</b> or a jump), the wobbel start point is generally set for the same value relative to the vector/arc startpoint; it is repeatedly continued, however, for <b>Polyline</b> (including arcs). For identical amplitudes (<b>Longitudinal</b> = <b>Transversal</b>), the wobbel startpoint is permanently referenced to the coordinate system, that is, independent of the current direction of motion.</li> <li>For an angle, elliptical or figure-8 wobbel movements can result in more or less small jumps after reaching the corner, depending on the current wobbel phase. This does not occur (= "rounded corners") if the wobbel motion is exactly circular (<b>Longitudinal</b> = <b>Transversal</b>).</li> <li><b>Longitudinal</b> = 0 produces a sine-shaped wobbel motion across the direction of movement.</li> <li>For "freely definable wobbel shapes", the parameter <b>Freq</b> has no meaning. It must nevertheless lie within the valid range, because otherwise the Wobbel mode gets deactivated. This also applies to the parameters <b>Transversal</b> and <b>Longitudinal</b>.</li> <li>If the Wobbel mode gets deactivated, then any already-defined wobbel shape remains active and is used upon the next switch-on of the Wobbel mode with <b>Mode</b> &gt; 1. This also applies, if the wobble figure has been previously saved by <b>create_dat_file</b> and then loaded by <b>load_program_file</b>.</li> <li>When defining the wobbel shape and its frequency take the dynamics of the scan head and laser into account. Otherwise, an overheating and even a permanent damage of the system may occur, see <b>Chapter 8.4.1 "Wobbel Shapes – Important Notes on Choosing Appropriate Parameter Values"</b>, page 220.</li> <li>The present wobbel amplitude can be recorded by <b>set_trigger/set_trigger4</b> (signal 53). The format of the data is ((transversal &lt;&lt; 16) + longitudinal).</li> <li>The Wobbel mode cannot be combined with: <ul style="list-style-type: none"> <li>– Sky Writing</li> <li>– Pixel output mode</li> <li>– Jumps</li> <li>– <b>laser_on_list</b></li> </ul> </li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC6 multiplies the specified value for <b>Longitudinal</b> and <b>Transversal</b> by 16. The allowed value ranges decrease accordingly.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>set_wobbel</b> , <b>set_wobbel_control</b> , <b>set_wobbel_offset</b> , <b>set_wobbel_vector</b>



<b>Delayed Short List Command</b>	<b>set_wobbel_mode_phase</b>
<b>Function</b>	Switches a classical Wobbel mode on and off as with <b>set_wobbel_mode</b> . In addition, defines a start phase.
<b>Call</b>	<code>set_wobbel_mode_phase( Transversal, Longitudinal, Freq, Mode, Phase )</code>
<b>Parameters</b>	<p>Transversal      As with <b>set_wobbel_mode</b>.</p> <p>Longitudinal    As with <b>set_wobbel_mode</b>.</p> <p>Freq             As with <b>set_wobbel_mode</b>.</p> <p>Mode             As with <b>set_wobbel_mode</b>.</p> <p>Phase           Start phase. As a 64-bit IEEE floating point value. Allowed value range: [0.0°...360.0°]. Negative values are clipped to 0.0°. Too big positive values are reduced by the corresponding multiples of 360.0°.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Mode <math>\geq 2</math> (freely definable wobbel shape) is not allowed with <b>set_wobbel_mode_phase</b> (is clipped to 1).</li> <li>• The start phase is converted to an integer value with a 16-bit resolution for a full circle.</li> <li>• By <b>set_wobbel_mode_phase</b>, the wobbel shape continues also with: <ul style="list-style-type: none"> <li>– <b>jump</b> commands</li> <li>– <b>timed_jump</b> commands</li> <li>– <b>para_jump</b> commands</li> <li>– <b>list_nop</b></li> <li>– <b>long_delay</b></li> </ul> Here, the wobbel shape follows the last valid marking direction. </li> <li>• <b>set_wobbel_direction</b> and <b>set_wobbel_offset</b> are effective.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 609, OUT 609, RBF 614.
References	<b>set_wobbel_mode</b>



<b>Delayed Short List Command</b>	<b>set_wobbel_offset</b>
Function	Defines a wobbel shape shift in the direction of motion or perpendicular to the direction of motion.
Call	<code>set_wobbel_offset( OffsetTrans, OffsetLong )</code>
Parameters	<p>OffsetTrans    Transversal offset. As a signed 32-bit value.            Allowed value range: <math>\pm 32,767</math>. Larger values are clipped.            Initialization values after <b>load_program_file</b>: (0,0).</p> <p>OffsetLong    Longitudinal offset. Otherwise, like OffsetTrans.</p>
Comments	<ul style="list-style-type: none"> <li>Offsets can be defined for “classic” wobbel shapes (circle, ellipse, sine, figure-8) as well as for “freely definable wobbel shapes”, see <a href="#">Chapter 8.4 “Wobbel Mode”, page 218</a>.</li> <li>The summed up wobbel amplitude including offsets may never exceed <math>\pm(2^{17}-1)</math>.</li> <li>At the beginning of the wobbel marking offsets are set as “<b>Hard jumps</b>”. This applies also in case of switching-off or non-using the Wobbel mode, for example, when using a jump command (analogously to “classical” wobbel shapes longitudinal amplitudes).</li> </ul>
RTC4→RTC6	New command. In <b>RTC4 Compatibility Mode</b> the RTC6 multiplies the specified values for OffsetTrans and OffsetLong by 16. The allowed value ranges decrease accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_wobbel_mode</a> , <a href="#">set_wobbel_vector</a> , <a href="#">set_wobbel_direction</a>



<b>Undelayed Short List Command</b>	<b>set_wobbel_vector</b>
<b>Function</b>	Defines a linear section of a wobbel shape.
<b>Call</b>	<code>set_wobbel_vector( dTrans, dLong, Period, dPower )</code>
<b>Parameters</b>	<p>dTrans      Microstep of a linear wobbel shape section. In bits.      dLong      As a 64-bit IEEE floating point value.      Allowed value range: [-256.0...+255.0].      dTrans is the wobbel excursion perpendicular to the direction of motion (which is either the laser trajectory (see Glossary entry on <a href="#">page 880</a>) or the direction of motion defined by <a href="#">set_wobbel_direction</a>).      dLong is correspondingly longitudinal to it.</p> <p>Period      As an unsigned 32-bit value.      Allowed value range: [0...65,535].      = 1...65,535: number of microsteps.      = 0: The wobbel shape is switched off.</p> <p>dPower      Microstep of the relative laser power.      As a 64-bit IEEE floating point value.      Allowed value range: [-1.0...+1.0].</p> <p>Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Period = 0 is not allowed as a shape section. Period = 0 means explicitly switch off the “freely definable wobbel shape” (not the Wobbel mode itself, see <a href="#">set_wobbel_mode</a>). Subsequent calls of <a href="#">set_wobbel_vector</a> begin a new wobbel shape.</li> <li>Each call of <a href="#">set_wobbel_vector</a> adds a new section at the end of the previous section independently of when the call is performed.</li> <li>Up to 1023 wobbel shape sections can be defined. After 1023 sections, storage automatically wraps around to the first section and overwrites it. Each further call does the same to the next section.</li> <li>The wobbel shape automatically begins with wobbel vector (0,0), that is, directly on the marking itself (the “Hard jump” of “classic” wobbel shapes is eliminated if the longitudinal amplitude ≠ 0) and also ends there without requiring explicit declaration.</li> <li>Each wobbel shape section consists of a vector defined by microsteps and their number. The parameters dTrans, dLong get internally rounded to 7 bit decimal places. At runtime each microstep is executed within 10 µs. For large Period values, you should therefore take rounding error into account. Positive dTrans values cause that in respect to the direction of movement the start is to the right (which applies to circular wobbel shapes as well). Positive dLong values cause that in respect to the direction of movement the start is forward. The last section’s endpoint is also the first section’s start point. Independently of its respective position, it always ends at (0,0) and get executed as a “Hard jump”. With the last step the laser power is reset to the initial nominal laser power.</li> <li>The summed up wobbel amplitude may never exceed ±(2<sup>17</sup>-1).</li> </ul>

<b>Undelayed Short List Command</b>	<b>set_wobbel_vector</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>• If activated by <b>set_wobbel_control</b> and <b>set_wobbel_mode( Mode = 2 )</b>, the following applies:           <ul style="list-style-type: none"> <li>– The RTC6 varies the current laser power P within the wobbel shape section as per <math>P = P0 \times (1 + dPower \times n)</math>, whereby <math>0 \leq n</math>. n represents the current number of each microstep. You can define the nominal laser power P0 with the list command <b>set_wobbel_control</b>.</li> <li>– If activated by <b>set_multi_mcbsp_in</b> or <b>set_multi_mcbsp_in_list</b>, the nominal laser power P0 is multiplicatively varied by the laser power parameter P across multiple McBSP transfers: <math>P0_{McBSP} = P0 \times P / 16,384</math>.</li> <li>– Beware here that for each both corrections above a maximum laser power of only <math>P0 \times 4.0</math> is allowed, whereby the maximum range of the laser control parameter likewise must not be exceeded (see <b>set_wobbel_control</b>).</li> <li>– For laser power variation with Ctrl = 5 (half period), dPower needs to have the opposite sign. Unlike with "Automatic Laser Control", the multiplication factor cannot be inverted.</li> <li>– By using an "empty" wobbel shape <b>set_wobbel_vector(0.0, 0.0, 1, 0.0)</b>, you can also multiplicatively vary the nominal laser power P0 without needing to explicitly wobbel (for another alternative, see <b>set_multi_mcbsp_in</b>).</li> <li>– When you switch off the wobbel shape (not by deactivation by <b>set_wobbel_mode</b>), the nominal laser power P0 is emitted at the port assigned by Ctrl if <b>set_wobbel_control</b> has been last called with Value = 0xFFFFFFFF. Otherwise, the laser power P0 multiplied by the external factor from <b>set_multi_mcbsp_in</b> is emitted.</li> </ul> </li> <li>• If activated by <b>set_wobbel_control</b> and <b>set_wobbel_mode( Mode = 3 )</b>, the following applies:           <ul style="list-style-type: none"> <li>– The RTC6 varies the current laser power P within the wobbel shape section as per <math>P = P100 \times (\text{Factor} + dPower \times n)</math>.</li> <li>– P100 (as with Mode = 2) is the nominal laser power as it would be without the wobbel figure. It is set by <b>set_wobbel_control( Ctrl, Value, MinValue, MaxValue )</b>:               <ul style="list-style-type: none"> <li>• With Value = 0...65,535, the specified Value value is used</li> <li>• With Value = 0xFFFFFFFF, the value is taken over that has been last outputted to the port Ctrl.</li> </ul> </li> <li>– Factor is an unsigned 16-bit value with scaling 16,384 = one (default value). The initial Factor value can be set to nnnn by Value = 0xFF00nnnn and another <b>set_wobbel_control</b> call. The parameters Ctrl, MinValue and MaxValue are taken over from the latest <b>set_wobbel_control</b> call.</li> <li>– To change the 100% performance, <b>set_wobbel_control</b> can be called at any time. However, the initial Factor value only becomes effective when the Wobbel mode is switched on <i>newly</i>. At wobbel shape runtime, Factor cannot be changed by <b>set_wobbel_control</b>.</li> <li>– Factor is changed per microstep additively by the dPower value from <b>set_wobbel_vector( dTrans, dLong, Period, dPower )</b>. The maximum value is 65,535 (Factor 4) and cannot be exceeded.</li> <li>– Power modulation via McBSP (as with Mode = 2) is not possible.</li> </ul> </li> </ul>



<b>Undelayed Short List Command</b>	<b>set_wobbel_vector</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If the wobbel shape gets switched off while the Wobbel mode remains active, then the shape automatically switches to <code>Mode</code> = 1 (horizontal figure-8).</li> <li>The Wobbel mode with "freely definable wobbel shapes" also needs to be switched on by <code>set_wobbel_mode</code>.</li> <li>Variable laser power for wobbel shapes cannot be combined with variable laser power for automatic or vector-based laser control (parameterized [*]mark[*] commands). Wobbel variation overwrites other variations if the respective Ctrl parameters are identical. Though unidentical Ctrl parameters are allowed, they serve no practical purpose.</li> <li>When defining "freely definable wobbel shapes" make sure that the microsteps of each individual wobbel vector does not exceed the maximum positioning speed significantly. Otherwise, a galvanometer scanner overheating may occur, see also Chapter 8.4.1 "Wobbel Shapes – Important Notes on Choosing Appropriate Parameter Values", page 220.</li> <li><code>create_dat_file</code> saves a "freely definable wobbel shape", see comment on page 342.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>set_multi_mcbsp_in</code> , <code>set_multi_mcbsp_in_list</code> , <code>set_wobbel_control</code> , <code>set_wobbel_offset</code>



<b>Ctrl Command</b>	<b>simulate_encoder</b>
<b>Function</b>	Activates or deactivates encoder simulation for the specified encoder.
<b>Call</b>	<code>simulate_encoder( EncoderNo )</code>
<b>Parameters</b>	<p>EncoderNo    Encoder number as an unsigned 32-bit value.            Allowed values:</p> <ul style="list-style-type: none"> <li>= 1:    ENCODER X pulses are simulated and encoder counter "Encoder0" thereby incremented.</li> <li>= 2:    ENCODER Y pulses are simulated and encoder counter "Encoder1" thereby incremented.</li> <li>= 3:    Pulses for ENCODER X and ENCODER Y are simulated.</li> <li>= 0:    The encoder simulation is deactivated.</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• The encoder simulation is driven by an internal 1 MHz clock (see also <a href="#">Section "Encoder Simulation", page 285</a>).</li> <li>• <b>simulate_encoder</b> does not trigger a reset of the encoder counter.</li> <li>• If <code>EncoderNo &gt; 3</code>, then <b>simulate_encoder</b> is ignored (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_encoder</a> , <a href="#">store_encoder</a> , <a href="#">read_encoder</a> , <a href="#">wait_for_encoder</a> , <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">set_fly_rot</a>



<b>Normal List Command</b>	<b>simulate_ext_start</b>
<b>Function</b>	After the specified track delay, causes a simulated external start.
<b>Call</b>	<code>simulate_ext_start( Delay, EncoderNo )</code>
<b>Parameters</b>	<p>Delay      Track delay (in counter steps of the selected <code>EncoderNo</code> encoder counter) as a signed 32-bit value. Allowed value range: <math>[-2^{31} \dots + (2^{31}-1)]</math>.</p> <p>EncoderNo    Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values:</p> <ul style="list-style-type: none"> <li>= 0:    Encoder counter "Encoder0"</li> <li>= 1:    Encoder counter "Encoder1"</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• For external starts, see <a href="#">Section "External Start", page 276</a>.</li> <li>• The track delay is specified in (relative) counting units of the selected encoder counter (the RTC6 encoder counters are triggered by an external or simulated encoder signal, see <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 284</a>). The start trigger for the start occurs only after the internal encoder counter has reached the specified track delay. External starts initiated by <code>simulate_ext_start</code> or by an external start signal, but whose execution was postponed according to the specified track delay, are placed in a queue that accommodates up to 8 starts. <code>simulate_ext_start</code> cancels a previous queue and starts a new one.</li> <li>• A start trigger initiated by <code>simulate_ext_start</code> or an external start signal only triggers a start if it does not occur when the <b>BUSY</b> status is set (for example, when outputting a list), when the <b>INTERNAL-BUSY</b> status is set (for example, during <code>goto_xy</code>) or/and when the <b>PAUSED</b> status is set (after <code>pause_list</code>, <code>stop_list</code> or <code>set_wait</code>). Otherwise, Bit #11 of the <code>get_startstop_info</code> return value is set. Therefore, if an unsuitable track delay is specified (for example, <code>Delay = 0</code>), no start is triggered. If <code>simulate_ext_start</code> is the first command in a list, then <code>get_startstop_info</code> can be used for checking whether processing of this list can finish within the defined track delay.</li> <li>• Ensure that the sign of the track delay (<code>Delay</code> parameter) is appropriate for the selected encoder's counting direction (for external triggering, this corresponds to the workpiece's direction of motion).</li> <li>• Track delays can also be set with <code>set_ext_start_delay</code> or <code>set_ext_start_delay_list</code>. Track delays are deactivated by initialization (with <code>load_program_file</code>), by external stops and by <code>stop_execution</code>. They can also be deactivated with <code>set_control_mode</code> (Bit #2).</li> <li>• The <code>simulate_ext_start</code> command alone <i>does not</i> cause an encoder reset. But if accordingly set with <code>set_control_mode</code> (Bit #9), a start trigger initiated by <code>simulate_ext_start</code>, <code>simulate_ext_start_ctrl</code> or by an external start signal causes an encoder reset if the start trigger is preceded by one of the Processing-on-the-fly commands <code>set_fly_x</code>, <code>set_fly_y</code>, <code>set_fly_2d</code> or <code>set_fly_rot</code>.</li> <li>• If <code>EncoderNo &gt; 1</code>, then <code>simulate_ext_start</code> is replaced with a <code>list_nop</code> (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> </ul>



<b>Normal List Command</b>	<b>simulate_ext_start</b>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC6 multiplies the specified value for <b>Delay</b> by 16. The allowed value range decreases accordingly.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>simulate_ext_start_ctrl</b> , <b>set_ext_start_delay</b> , <b>set_ext_start_delay_list</b> , <b>set_extstartpos</b> , <b>set_extstartpos_list</b> , <b>set_control_mode</b> , <b>simulate_ext_stop</b>



<b>Ctrl Command</b>	<b>simulate_ext_start_ctrl</b>
<b>Function</b>	Causes a simulated external start.
<b>Call</b>	<code>simulate_ext_start_ctrl()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For external starts, see <a href="#">Section "External Start", page 276</a>.</li> <li>The start trigger for the start occurs only after a track delay previously set by <a href="#">set_ext_start_delay</a>, <a href="#">set_ext_start_delay_list</a> or <a href="#">simulate_ext_start</a>. Track delays are deactivated by initialization (with <a href="#">load_program_file</a>), by external stops and by <a href="#">stop_execution</a>. They can also be deactivated with <a href="#">set_control_mode</a> (Bit #2). External starts initiated by <a href="#">simulate_ext_start_ctrl</a> or by an external start signal, but whose execution was postponed according to the specified track delay, are placed in a queue that accommodates up to 8 starts. In contrast to <a href="#">simulate_ext_start</a>, <a href="#">simulate_ext_start_ctrl</a> does <i>not</i> cancel the previous queue.</li> <li>A start trigger initiated by <a href="#">simulate_ext_start_ctrl</a> or by an external start signal does only trigger a start if it does not coincide with the output of a list (otherwise, Bit #11 of the <a href="#">get_startstop_info</a> return value gets set).</li> <li>The <a href="#">simulate_ext_start_ctrl</a> command alone <i>does not</i> cause an encoder reset. But if accordingly set with <a href="#">set_control_mode</a>( Bit #9 ), a start trigger initiated by <a href="#">simulate_ext_start_ctrl</a>, <a href="#">simulate_ext_start</a> or by an external start signal causes an encoder reset if the start trigger is preceded by one of the Processing-on-the-fly commands <a href="#">set_fly_x</a>, <a href="#">set_fly_y</a>, <a href="#">set_fly_2d</a> or <a href="#">set_fly_rot</a>.</li> <li><a href="#">simulate_ext_start_ctrl</a> can be disabled by <a href="#">set_control_mode</a>( Bit #4 = 1 ) or <a href="#">set_control_mode_list</a>( Bit #4 = 1 ).</li> <li>As of version DLL 609, OUT 609, RBF 614, the following applies: provided that a start is allowed, <a href="#">simulate_ext_start_ctrl</a> waits 30 µs before it returns. This closes a potential timing gap (with <a href="#">get_status</a>) between command call and actual start.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 609, OUT 609, RBF 614: see above.
References	<a href="#">simulate_ext_start</a>



<b>Ctrl Command</b>	<b>simulate_ext_stop</b>
Function	Causes a simulated external stop.
Call	<code>simulate_ext_stop()</code>
Comments	<ul style="list-style-type: none"><li>For external stops, see <a href="#">Section "External Stop", page 275</a>.</li><li><code>simulate_ext_stop</code> simultaneously halts the master board and all slave boards.</li><li>In contrast, <a href="#"><code>stop_execution</code></a> only halts the master board.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#"><code>simulate_ext_start</code></a> , <a href="#"><code>simulate_ext_start_ctrl</code></a> , <a href="#"><code>stop_execution</code></a>



<b>Undelayed Short List Command</b>	<b>spot_distance</b>
Function	As <a href="#">spot_distance_ctrl</a> , but an undelayed short list command.
Call	spot_distance( Dist )
Parameters	Dist      See <a href="#">spot_distance_ctrl</a> .
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">spot_distance_ctrl</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 609, OUT 609, RBF 613.
References	<a href="#">spot_distance_ctrl</a> , <a href="#">set_auto_laser_control</a> , <a href="#">set_auto_laser_params</a> , <a href="#">set_auto_laser_params_list</a>

<b>Ctrl Command</b>	<b>spot_distance_ctrl</b>
Function	Defines the geometric pulse distance for the "Automatic Laser Control" with Ctrl = 7.
Call	spot_distance_ctrl( Dist )
Parameters	Dist      Pulse distance. In bits. As a 64-bit IEEE floating point value.
Comments	<ul style="list-style-type: none"> <li>• The pulse distance is resolved with an accuracy of 1/40 bit (image field coordinates).</li> <li>• Dist = 0 suppresses the "Automatic Laser Control" with Ctrl = 7, but does not switch it off.</li> <li>• Dist must not exceed the maximum value 26,214. A practical Dist value depends on mark speed, laser frequency and scan system.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 609, OUT 609, RBF 613.
References	<a href="#">spot_distance</a> , <a href="#">set_auto_laser_control</a> , <a href="#">set_auto_laser_params</a> , <a href="#">set_auto_laser_params_list</a>



<b>Ctrl Command</b>	<b>start_loop</b>
<b>Function</b>	Starts a repeating automatic list change.
<b>Call</b>	<code>start_loop()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>A list change or new list start activated with <b>start_loop</b> repeats until execution is ended by calling the command <b>quit_loop</b>.</li> <li><b>start_loop</b> can be called at any time but only has effect upon the next <b>set_end_of_list</b>.</li> <li>If the automatic list change is activated during processing of a list, then upon reaching <b>set_end_of_list</b> execution continues without delay at the other list. If there is only one list (<code>Mem2 = 0</code>, see <b>config_list</b>), then upon reaching <b>set_end_of_list</b> execution continues at <code>Pos = 0</code> (that is, at the list's beginning).</li> <li>During processing of a list, the other list (and also the current list) can be newly loaded, see <a href="#">Chapter 6.4.6 "Changing Lists Automatically", page 100</a>.</li> <li>So that <b>start_loop</b> can function at all, the already active list must absolutely be finalized by <b>set_end_of_list</b>; the new list should already be loaded and the input pointer should be sufficiently ahead of the output pointer (otherwise, "old" commands are executed). If, during list execution, the end of the list is reached without encountering a <b>set_end_of_list</b>, then execution automatically continues at the beginning of the current list, not at the beginning of the other list.</li> <li>If, during processing of a list, <b>start_loop</b> and <b>auto_change_pos( Pos &gt;0 )</b> are called, then upon the next <b>set_end_of_list</b> the command <b>auto_change_pos( Pos &gt;0 )</b> is executed; and at the next one <b>start_loop</b> is executed.</li> <li>The current list and list execution statuses can be queried by the commands <b>read_status</b> and <b>get_status</b>.</li> <li><b>start_loop</b> triggers a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 95</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>quit_loop</b>



<b>Ctrl Command</b>	<b>stepper_abs</b>						
<b>Function</b>	Triggers set-position movements to the specified absolute set positions by both stepper motor output ports.						
<b>Call</b>	<code>stepper_abs( Pos1, Pos2, WaitTime )</code>						
<b>Parameters</b>	<table> <tr> <td>Pos1</td> <td>Absolute set position in CLOCK pulse units for stepper motor output ports 1. As signed 32-bit values. Allowed value range: <math>[-2^{31} \dots + (2^{31}-1)]</math>.</td> </tr> <tr> <td>Pos2</td> <td>Like Pos1 (analogously).</td> </tr> <tr> <td>WaitTime</td> <td>Determines when <b>stepper_abs</b> returns at the latest. As an unsigned 32-bit value. <i>1 bit equals 1 s.</i> Allowed value range: <math>[0 \dots + (2^{32}-1)]</math>.</td> </tr> </table>	Pos1	Absolute set position in CLOCK pulse units for stepper motor output ports 1. As signed 32-bit values. Allowed value range: $[-2^{31} \dots + (2^{31}-1)]$ .	Pos2	Like Pos1 (analogously).	WaitTime	Determines when <b>stepper_abs</b> returns at the latest. As an unsigned 32-bit value. <i>1 bit equals 1 s.</i> Allowed value range: $[0 \dots + (2^{32}-1)]$ .
Pos1	Absolute set position in CLOCK pulse units for stepper motor output ports 1. As signed 32-bit values. Allowed value range: $[-2^{31} \dots + (2^{31}-1)]$ .						
Pos2	Like Pos1 (analogously).						
WaitTime	Determines when <b>stepper_abs</b> returns at the latest. As an unsigned 32-bit value. <i>1 bit equals 1 s.</i> Allowed value range: $[0 \dots + (2^{32}-1)]$ .						
<b>Comments</b>	<ul style="list-style-type: none"> <li>For programming the stepper motor signals, see <a href="#">Chapter 9.1.5 "Controlling Stepper Motors", page 270</a>.</li> <li><b>stepper_abs</b> sets the new set-position values even if a previously started set-position movement is still in progress: <ul style="list-style-type: none"> <li>If Pos1/Pos2 in the current direction of movement lies in front of the internal position variable's value, then the movement continues and the Busy status remains set.</li> <li>If Pos1/Pos2 equals the current value of the internal position variable, then the movement stops. The Busy status gets reset.</li> <li>If Pos1/Pos2 in the current direction of movement already lies past the internal position variable's value, then the corresponding stepper motor's direction of movement reverses, see <a href="#">Section "Notes", page 270</a>. The Busy status remains set.</li> </ul> </li> <li>If no set-position movement is in progress, then one starts and the Busy status gets set.</li> <li>During performance of a reference movement (Init status set, see <b>stepper_init</b>), the command <b>stepper_abs</b> does not execute (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>If the CLOCK pulse period was set to 0 by <b>stepper_init</b>, <b>stepper_control</b> or <b>stepper_control_list</b>, then no stepper motor movement occurs at the corresponding stepper motor output.</li> <li>If WaitTime = 0, then <b>stepper_abs</b> returns immediately so that system control is restored to the user program.</li> </ul>						
RTC4→RTC6	New command.						
RTC5→RTC6	Unchanged functionality.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">stepper_abs_no</a> , <a href="#">stepper_rel</a> , <a href="#">stepper_rel_no</a> , <a href="#">stepper_abs_list</a>						



<b>Undelayed Short List Command</b>	<b>stepper_abs_list</b>
Function	Like <b>stepper_abs</b> , but a list command and without <code>WaitTime</code> parameter.
Call	<code>stepper_abs_list( Pos1, Pos2 )</code>
Parameters	Pos1      Like <b>stepper_abs</b> .
	Pos2      Like <b>stepper_abs</b> .
Comments	<ul style="list-style-type: none"> <li>• See <b>stepper_abs</b>.</li> <li>• During performance of a reference movement (see <b>stepper_init</b>), execution of <b>stepper_abs_list</b> is delayed until the reference movement completes.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>stepper_abs</b>

<b>Ctrl Command</b>	<b>stepper_abs_no</b>
Function	Triggers a set-position movement to the specified absolute set position at the specified stepper motor output port.
Call	<code>stepper_abs_no( No, Pos, WaitTime )</code>
Parameters	No      Number of the stepper motor output port. As an unsigned 32-bit value. Allowed values: = 1: Stepper motor output port 1. = 2: Stepper motor output port 2.  If the value is invalid, then <b>stepper_abs_no</b> is not executed ( <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b> ).
	Pos      Absolute set position in CLOCK pulse units. As a signed 32-bit value. Allowed value range: [-2 <sup>31</sup> ...+(2 <sup>31</sup> -1)].
	WaitTime      Determines when, at the latest, the function returns. <i>1 bit equals 1 s.</i> As an unsigned 32-bit value. Allowed value range: [0...+(2 <sup>32</sup> -1)].
Comments	<ul style="list-style-type: none"> <li>• A set-position movement is only performed at the stepper motor output port specified by <code>No</code>. Otherwise the command is identical to <b>stepper_abs</b> (see comments there).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>stepper_abs</b> , <b>stepper_abs_no_list</b>



<b>Undelayed Short List Command</b>	<b>stepper_abs_no_list</b>
Function	Like <b>stepper_abs_no</b> , but a list command and without WaitTime parameter.
Call	<code>stepper_abs_no_list( No, Pos )</code>
Parameters	<p>No      Number of the stepper motor output.            As an unsigned 32-bit value.            Allowed values:            = 1: Stepper motor output 1.            = 2: Stepper motor output 2.            If the value is invalid, then <b>stepper_abs_no_list</b> is, already during loading,            replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</p> <p>Pos     Like <b>stepper_abs_no</b>.</p>
Comments	<ul style="list-style-type: none"> <li>• See <b>stepper_abs_no</b>.</li> <li>• During performance of a reference movement (see <b>stepper_init</b>), execution of <b>stepper_abs_no_list</b> is delayed until the reference movement completes.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>stepper_abs_no</b>



<b>Ctrl Command</b>	<b>stepper_control</b>
<b>Function</b>	Sets the CLOCK signal pulse periods for stepper motor control.
<b>Call</b>	<code>stepper_control( Period1, Period2 )</code>
<b>Parameters</b>	<p>Period1      Pulse period of the CLOCK signals for stepper motor output ports 1.  <i>1 bit equals 10 µs.</i>  As a signed 32-bit value.  Allowed values: [0...+(2<sup>24</sup>-1)] or &lt; 0. Larger values are clipped.</p> <p>&gt; 0:      The new period waits for an already-running CLOCK pulse period at the corresponding stepper motor output before starting.</p> <p>= 0:      An already-running CLOCK pulse period aborts at each stepper motor output (the corresponding stepper motor movement gets stopped, the Init- and/or Busy statuses for the respective stepper motor outputs get reset).</p> <p>&lt; 0:      the corresponding stepper motor control remains unchanged.</p> <p>Period2      Pulse period of the CLOCK signals for stepper motor output ports 2.  Otherwise, like Period1.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For programming the stepper motor signals, see <a href="#">Chapter 9.1.5 "Controlling Stepper Motors", page 270</a>.</li> <li>Period1 <b>or</b> Period2 = 0 can be used as an emergency stop, see also the <a href="#">Section "Terminating Infinite Movements", page 272</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">stepper_control_list</a>



<b>Undelayed Short List Command</b>	<b>stepper_control_list</b>
Function	Like <b>stepper_control</b> , but a list command.
Call	<code>stepper_control_list( Period1, Period2 )</code>
Parameters	Period1 Like <b>stepper_control</b> . Period2 Like <b>stepper_control</b> .
Comments	<ul style="list-style-type: none"> <li>• See <b>stepper_control</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>stepper_control</b>

<b>Ctrl Command</b>	<b>stepper_disable_switch</b>
Function	Controls the usage of the stepper motor control SWITCH signals.
Call	<code>stepper_disable_switch( Disable1, Disable2 )</code>
Parameters	Disable1 Instruction how the SWITCH signals from stepper motor input 1 are to be used. As a signed 32-bit value.. Allowed value range: $[-2^{32} \dots +(2^{32}-1)]$ . > 0: The SWITCH signal at the stepper motor input is not used. = 0: The SWITCH signal at the stepper motor input is used. < 0: The use of the stepper motor input signal remains unchanged. Disable2 Like Disable1 (analogously).
Comments	<ul style="list-style-type: none"> <li>• For programming the stepper motor signals, see <b>Chapter 9.1.5 "Controlling Stepper Motors", page 270</b>.</li> <li>• The limit switch can be ignored during normal forwarding motions. For example, this may make sense for continuously rotating axes.</li> <li>• The SWITCH signals are always used with forwarding motions initiated by <b>stepper_init</b>.</li> </ul>
Version info	Available as of version DLL 542, OUT 542.
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>stepper_init</b>



<b>Ctrl Command</b>	<b>stepper_enable</b>
<b>Function</b>	Sets the ENABLE signals of the stepper motor control.
<b>Call</b>	<code>stepper_enable( Enable1, Enable2 )</code>
<b>Parameters</b>	<p>Enable1     ENABLE signal for stepper motor output 1.  As a signed 32-bit value.  Allowed value range: <math>[-2^{32} \dots + (2^{32}-1)]</math>.  &gt; 0:     The ENABLE signal at the stepper motor output gets set.  = 0:     The ENABLE signal at the stepper motor output gets reset.  &lt; 0:     The stepper motor output signal remains unchanged.</p> <p>Enable2     Like <code>Enable1</code> (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For programming the stepper motor signals, see <a href="#">Chapter 9.1.5 "Controlling Stepper Motors", page 270</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">stepper_enable_list</a>

<b>Undelayed Short List Command</b>	<b>stepper_enable_list</b>
<b>Function</b>	Like <a href="#">stepper_enable</a> , but a list command.
<b>Call</b>	<code>stepper_enable_list( Enable1, Enable2 )</code>
<b>Parameters</b>	<p>Enable1     Like <a href="#">stepper_enable</a>.</p> <p>Enable2     Like <a href="#">stepper_enable</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">stepper_enable</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">stepper_enable</a>



<b>Ctrl Command</b>	<b>stepper_init</b>
<b>Function</b>	Performs a stepper motor initialization.
<b>Call</b>	<code>stepper_init( No, Period, Dir, Pos, Tol, Enable, WaitTime )</code>
<b>Parameters</b>	<p>No      Number of the stepper motor output. As an unsigned 32-bit value. Allowed values: = 1: Stepper motor output 1. = 2: Stepper motor output 2. If the value is invalid, then <b>stepper_init</b> is not executed (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</p> <p>Period    Pulse period of the CLOCK signal. As an unsigned 32-bit value. <i>1 bit equals 10 µs.</i> Allowed value range: [0...+(2<sup>24</sup>-1)]. Larger values are clipped. If Period = 0, then no reference movement is performed and the function immediately returns (see comments).</p> <p>Dir      Direction of stepper motor motion during the reference movement. As a signed 32-bit value. Allowed value range: [-2<sup>31</sup>...+(2<sup>31</sup>-1)]. &gt; 0: The DIRECTION signal gets set; during the reference movement the internal position variable increments. = 0: The DIRECTION signal gets reset; during the reference movement the internal position variable decrements. &lt; 0: The DIRECTION signal remains unchanged, no reference movement is performed and the function immediately returns.</p> <p>Pos      New position variable value. In CLOCK pulse units (see comments). As a signed 32-bit value. Allowed value range: [-2<sup>31</sup>...+(2<sup>31</sup>-1)].</p> <p>Tol      Tolerance for the reference movement (see comments). As an unsigned 32-bit value. Allowed value range: [0...+(2<sup>32</sup>-1)]. If Dir &lt; 0 and/or Period = 0, then the value Tol = 0 is irrelevant, otherwise Tol = 0 causes <b>stepper_init</b> not to be executed (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</p> <p>Enable    ENABLE signal. As an unsigned 32-bit value. Allowed value range: [0...+(2<sup>32</sup>-1)]. = 0: The ENABLE signal is reset. &gt; 0: The ENABLE signal is set.</p> <p>WaitTime   Defines when <b>stepper_init</b>, at the latest, returns (see comments). <i>1 bit equals 1 s.</i> As an unsigned 32-bit value. Allowed value range: [0...+(2<sup>32</sup>-1)].</p>

Ctrl Command	stepper_init
Comment	<ul style="list-style-type: none"> <li>For programming the stepper motor signals, see <a href="#">Chapter 9.1.5 "Controlling Stepper Motors", page 270</a>.</li> <li><b>stepper_init</b> immediately stops all previously started movements of the stepper motor specified by the <code>No</code> parameter.</li> <li>The <code>ENABLE</code> signal <code>Enable</code> is merely forwarded and always correspondingly set, but has no effect on internal operations.</li> <li>If <code>Period &gt; 0</code> and <code>Dir &gt;= 0</code>, then stepper motor <code>No</code> starts a reference movement with the supplied <code>CLOCK</code> pulse period in the defined direction and the <code>Init</code> status gets set. The first <code>Clock</code> pulse is only generated after a full <code>CLOCK</code> pulse period.             <ul style="list-style-type: none"> <li>If a limit switch is activated right from the beginning, then the controller attempts to seek a position within the <math>\pm \text{Tol}</math> range of the current position, initially opposite to the defined direction, with the limit switch deactivated. If this does not bring success, then the attempt terminates. In this case, the <code>SWITCH</code> status bit remains set (see <a href="#">get_steerer_status</a>).</li> <li>If a limit switch gets activated during a movement, then the reference movement stops there. Afterward, the limit switch position is crossed 4× to arrive at an averaged value for this position. Finally, the stepper motor is driven in the opposite direction by a normal set-position movement (<code>Init</code> status reset, <code>Busy</code> status set) within the tolerance value <code>Tol</code>. Here, the <code>DIRECTION</code> status signal changes, whereas it remains constant during seeking movements with multiple direction changes. The internal position variable (for the current position) gets set to the value defined by the <code>Pos</code> parameter. Thus, this value represents a positional offset by <code>Tol</code> with respect to the defined position. With <code>Pos = Tol</code>, the middle limit switch position corresponds to position 0.</li> <li>If no limit switch is found (for example, because no limit switch exists in the defined direction), then the stepper motor performs an infinite movement. <b>stepper_init</b> then returns after <code>WaitTime</code> seconds to restore system control to the user program. But the stepper motor's infinite movement continues until it is aborted by a new <b>stepper_init</b> command or <b>stepper_control</b> (<code>Period = 0</code>). For more on this, see the <a href="#">Section "Terminating Infinite Movements", page 272</a>.</li> </ul> </li> <li>If <code>Period = 0</code>, <code>Dir &lt; 0</code> and/or <code>WaitTime = 0</code>, then <b>stepper_init</b> returns straight away to immediately restore system control to the user program. You can then call <a href="#">get_steerer_status</a> to check whether the reference movement has completed. During the seeking movement the status "Init" (see <a href="#">page 414</a>) is set and during the terminating set-position movement to (limit switch + <code>Tol</code>) the status "Busy" (see <a href="#">page 414</a>) is set.</li> <li><code>Period = 0</code> and/or <code>Dir &lt; 0</code> can be used as an emergency stop. Then a previously started stepper motor movement gets aborted, but no reference movement is performed. Here, too, the position variable gets set to the value <code>Pos</code>. The <code>DIRECTION</code> signal remains unchanged.</li> <li>If <code>Period = 0</code>, then status "Init" (see <a href="#">page 414</a>) and status "Busy" (see <a href="#">page 414</a>) get reset. No further clock pulses are outputted until <code>Period</code> is again set to a positive value.</li> </ul>



<b>Ctrl Command</b>	<b>stepper_init</b>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–

<b>Ctrl Command</b>	<b>stepper_rel</b>
Function	Triggers set-position movements to the specified relative positions by both stepper motor output ports.
Call	<code>stepper_rel( dPos1, dPos2, WaitTime )</code>
Parameters	<p>dPos1      Relative set positions in CLOCK pulse units for stepper motor output port 1. As a signed 32-bit value. Allowed value range: [-2<sup>31</sup>...+(2<sup>31</sup>-1)].</p> <p>dPos2      Relative set positions in CLOCK pulse units for stepper motor output port 2. As a signed 32-bit value. Allowed value range: [-2<sup>31</sup>...+(2<sup>31</sup>-1)].</p> <p>WaitTime    Like <a href="#">stepper_abs</a>.</p>
Comments	<ul style="list-style-type: none"> <li>The set positions should be specified relative to the current position values. Otherwise, the command is identical to <a href="#">stepper_abs</a> (see comments there).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">stepper_abs</a> , <a href="#">stepper_rel_list</a>

<b>Undelayed Short List Command</b>	<b>stepper_rel_list</b>
Function	Like <a href="#">stepper_rel</a> , but a list command and without <code>WaitTime</code> parameter.
Call	<code>stepper_rel_list( dPos1, dPos2 )</code>
Parameters	<p>dPos1      Like <a href="#">stepper_rel</a>.</p> <p>dPos2      Like <a href="#">stepper_rel</a>.</p>
Comments	<ul style="list-style-type: none"> <li>See <a href="#">stepper_rel</a>.</li> <li>During performance of a reference movement (see <a href="#">stepper_init</a>), execution of <a href="#">stepper_rel_list</a> is delayed until the reference movement completes.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">stepper_rel</a>



<b>Ctrl Command</b>	<b>stepper_rel_no</b>
<b>Function</b>	Triggers a set-position movement to the specified relative position at the specified stepper motor output port.
<b>Call</b>	<code>stepper_rel_no( No, dPos, WaitTime )</code>
<b>Parameters</b>	<p>No            Like <a href="#">stepper_abs_no</a>.</p> <p>dPos        Relative position. In CLOCK pulse units. As a signed 32-bit value. Allowed value range: <math>[-2^{31} \dots + (2^{31}-1)]</math>.</p> <p>WaitTime    Like <a href="#">stepper_abs_no</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The set positions should be specified relative to the current position values (the - position value is correspondingly get newly set) and a set-position movement is only performed at the stepper motor output specified by <code>No</code>. Otherwise the command is identical to <a href="#">stepper_abs</a> (see comments there).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">stepper_abs_no</a> , <a href="#">stepper_abs</a> , <a href="#">stepper_rel_no_list</a>

<b>Undelayed Short List Command</b>	<b>stepper_rel_no_list</b>
<b>Function</b>	Like <a href="#">stepper_rel_no</a> , but a list command and without <code>WaitTime</code> parameter.
<b>Call</b>	<code>stepper_rel_no_list( No, dPos )</code>
<b>Parameters</b>	<p>No            Number of the stepper motor output port. As an unsigned 32-bit value. Allowed values: = 1: Stepper motor output port 1. = 2: Stepper motor output port 2.</p> <p>If the value is invalid, then <code>stepper_rel_no_list</code> is, already during loading, replaced by a <a href="#">list_nop</a> (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</p> <p>dPos        Like <a href="#">stepper_rel_no</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">stepper_rel_no</a>.</li> <li>During performance of a reference movement (see <a href="#">stepper_init</a>), execution of <code>stepper_rel_no_list</code> is delayed until the reference movement completes.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">stepper_rel_no</a>



<b>Normal List Command</b>	<b>stepper_wait</b>
<b>Function</b>	Interrupts further execution of a list until a previously started (at the specified stepper motor output) stepper motor movement completes.
<b>Call</b>	<code>stepper_wait( No )</code>
<b>Parameters</b>	No      Number of the stepper motor output. As an unsigned 32-bit value. Allowed values: = 1: Stepper motor output 1. = 2: Stepper motor output 2. = 0, 3: Both stepper motor output ports.  Only the two least-significant bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>For programming the stepper motor signals, see <a href="#">Chapter 9.1.5 "Controlling Stepper Motors", page 270</a>.</li> <li>If no stepper motor movement had been previously started at the specified stepper motor output port, then <b>stepper_wait</b> still needs 10 µs to execute, even though it otherwise has no effect.</li> <li><b>stepper_wait</b> does <i>not</i> influence: <ul style="list-style-type: none"> <li>the "laser active" laser control signals</li> <li>the list status</li> <li>the list execution status</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	–



<b>Ctrl Command</b>	<b>stop_execution</b>
<b>Function</b>	Stops execution of the list and deactivates the “laser active” laser control signals immediately.
<b>Call</b>	<code>stop_execution()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>stop_execution</b> deactivates the signals for “laser active” operation even if no list is active (here the command has no other effects; here too: <b>get_last_error</b> return code <b>RTC6_BUSY</b>).</li> <li>• With <b>stop_execution</b>, the galvanometer scanners stay in the current position, unless a home jump has been previously defined by <b>home_position</b> or <b>home_position_xyz</b> (a home jump is executed). Therefore, before a new list is loaded, the galvanometer scanners should be set to a defined position using the command <b>goto_xy</b>.</li> <li>• The external START inputs are disabled, see <b>Section “External Start”, page 276</b>.</li> <li>• The Processing-on-the-fly correction is switched off.</li> <li>• The <b>BUSY</b> list status values (see <b>read_status</b>) and the <b>BUSY</b> list execution status value (see <b>get_status</b>) are reset.</li> <li>• A list that was interrupted with <b>stop_execution</b> cannot be resumed and must instead be newly started (for example, by <b>execute_list_pos</b>). To only temporarily halt a list and later resume it, you can use <b>pause_list</b>.</li> <li>• <b>stop_execution</b> only affects the addressed RTC6 board. Even in a master/slave chain, it is not passed on to the slave boards. If all boards of a master/slave chain are to be synchronously stopped, then <b>simulate_ext_stop</b> or an external stop signal must be issued to any card of the master/slave chain, see also <b>Chapter 6.6.3 “Master/Slave Operation”, page 114</b>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality. However: Master/slave change.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>get_startstop_info</b>

<b>Ctrl Command</b>	<b>stop_list</b>
<b>Function</b>	Pauses execution of the list and deactivates the signals for “laser active” operation.
<b>Call</b>	<code>stop_list()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>stop_list</b> is synonymous with <b>pause_list</b> (see comments there).</li> </ul>
RTC4→RTC6	Basically unchanged functionality. However: Additional <b>PAUSED</b> status which is set by <b>stop_list</b> .
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>pause_list</b>



<b>Ctrl Command</b>	<b>stop_trigger</b>
<b>Function</b>	Resets (to <code>Busy = 0</code> ) the measurement session status that can be queried by <a href="#">measurement_status</a> .
<b>Call</b>	<code>stop_trigger()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>stop_trigger</code> is only needed if a measurement session has been started by <code>set_trigger</code> or <code>set_trigger4</code>, but not subsequently terminated by <code>set_trigger(Period = 0)</code> or <code>set_trigger4(Period = 0)</code>. Here, you can reset the measurement session's status even when no list is active (see comments at <code>set_trigger</code>).</li> <li>• <code>stop_trigger</code> is not executed (<code>get_last_error</code> return code <code>RTC6_BUSY</code>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <code>pause_list</code>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board is set</li> </ul> </li> <li>• <code>stop_trigger</code> is even executed, if: <ul style="list-style-type: none"> <li>– a list has been paused by <code>set_wait</code> (<b>PAUSED</b> status set)</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">measurement_status</a> , <a href="#">set_trigger</a> , <a href="#">set_trigger4</a>

<b>Undelayed Short List Command</b>	<b>store_encoder</b>
<b>Function</b>	Stores the current counts of the two RTC6 encoder counters in a cache on the RTC6.
<b>Call</b>	<code>store_encoder( Pos )</code>
<b>Parameters</b>	<p>Pos      Storage position.  As an unsigned 32-bit value.</p> <ul style="list-style-type: none"> <li>• Bit #0 = 0: The counter values are saved to storage position 0.</li> <li>• Bit #0 = 1: The counter values are saved to storage position 1.</li> </ul>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>store_encoder</code> can be used, for instance, to determine the exact number of encoder increments occurring within a specific list-processing period. Here, you could begin the command list by saving the counts in storage position 0 by <code>store_encoder(0)</code>. At the end of the command list, you could save the counts to storage position 1 by <code>store_encoder(1)</code> and subsequently retrieve by <code>read_encoder</code>.</li> <li>• See also <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 284</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">read_encoder</a> , <a href="#">get_encoder</a> , <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">set_fly_rot</a> , <a href="#">wait_for_encoder</a>



<b>Ctrl Command</b>	<b>store_program</b>													
<b>Function</b>	<b>Standalone Functionality:</b> Saves (deletes) data for automatic booting to (from) the <b>NAND memory</b> .													
<b>Prerequisite</b>	Minimum requirement: RTC6 Software Package V1.7.0 and RTC6BIOSETH_26.													
<b>Call</b>	Error = store_program( Mode )													
<b>Parameters</b>	<p>Mode           = 0:   Saves data for "<b>Standalone Basic State</b>" (see below).</p> <p>               = 1:   Erases the <b>NAND memory</b> content.</p> <p>               &gt; 1:   Like = 0, and in addition some files for "<b>Standalone Full State</b>" (see below).</p> <p>As an unsigned 32-bit value.</p>													
<b>Result</b>	<p>Error           Error code.</p> <p>As an unsigned 32-bit value.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error.</td> </tr> <tr> <td>1</td> <td><b>NAND memory</b> not addressable.</td> </tr> <tr> <td>2</td> <td><b>NAND memory</b> end reached early.</td> </tr> <tr> <td>3</td> <td>Data have not been or only partially stored in <b>NAND memory</b>.</td> </tr> <tr> <td>4</td> <td>Not an RTC6 Ethernet Board.</td> </tr> </tbody> </table>		Value	Description	0	No error.	1	<b>NAND memory</b> not addressable.	2	<b>NAND memory</b> end reached early.	3	Data have not been or only partially stored in <b>NAND memory</b> .	4	Not an RTC6 Ethernet Board.
Value	Description													
0	No error.													
1	<b>NAND memory</b> not addressable.													
2	<b>NAND memory</b> end reached early.													
3	Data have not been or only partially stored in <b>NAND memory</b> .													
4	Not an RTC6 Ethernet Board.													
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>store_program</b> is only allowed with RTC6 Ethernet Boards. Otherwise, a <b>get_last_error</b> return code <b>RTC6_TYPE_REJECTED</b> is generated.</li> <li>• <b>store_program</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status set)</li> </ul> </li> <li>• During the execution of <b>store_program</b> the 10 µs clock period of the <b>DSP</b> is interrupted: <ul style="list-style-type: none"> <li>– Mode = 0 approx. 10 s</li> <li>– Mode &gt; 1 approx. 60 s</li> </ul> </li> <li>• Data for "<b>Standalone Basic State</b>" are: <ul style="list-style-type: none"> <li>– <b>RTC6ETH.out</b>, <b>RTC6RBF.rbf</b> and <b>RTC6DAT.dat</b></li> </ul> </li> <li>• Data for "<b>Standalone Full State</b>" are: <ul style="list-style-type: none"> <li>– Data for "<b>Standalone Basic State</b>" (see bullet above)</li> <li>– The required control commands, list commands and correction files</li> </ul> </li> <li>• In case of an error, a <b>get_last_error</b> return code <b>RTC6_FLASH_ERROR</b> is generated.</li> <li>• See <b>Chapter 15.7 "Standalone Functionality", page 859</b>.</li> </ul>													
RTC4→RTC6	New command.													
RTC5→RTC6	New command.													
Version info	Available as of version DLL 618, OUT 618, RBF 623.													
References	<b>read_image_eth</b> , <b>write_image_eth</b>													



<b>Ctrl Command</b>	<b>store_timestamp_counter</b>
<b>Function</b>	Saves the current "time stamp counter" value.
<b>Call</b>	<code>store_timestamp_counter()</code>
<b>Parameters</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">Chapter 8.12 "Time Measurements", page 267</a>.</li> <li>• The current "time stamp counter" value is stored as time reference <code>TimeStampStorage</code> for <a href="#">wait_for_timestamp_counter</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
<b>Version info</b>	Available as of version DLL 617, OUT 617, RBF 623.
<b>References</b>	<a href="#">store_timestamp_counter_list</a> , <a href="#">wait_for_timestamp_counter</a>

<b>Undelayed Short List Command</b>	<b>store_timestamp_counter_list</b>
<b>Function</b>	Like <a href="#">store_timestamp_counter</a> , but a list command.
<b>Call</b>	<code>store_timestamp_counter_list()</code>
<b>Parameters</b>	None.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">store_timestamp_counter</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
<b>Version info</b>	Available as of version DLL 617, OUT 617, RBF 623.
<b>References</b>	<a href="#">store_timestamp_counter</a> , <a href="#">wait_for_timestamp_counter</a>



<b>Undelayed Short List Command</b>	<b>sub_call</b>
Function	Causes an unconditional jump to an indexed subroutine.
Call	sub_call( Index )
Parameters	Index      Index of the called indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].
Comments	<ul style="list-style-type: none"> <li>• <b>sub_call</b> reads the indexed subroutine's starting address from the internal management table based on the supplied index and then calls the command <b>list_call</b> (see also the comments there), which then triggers the jump to the subroutine.</li> <li>• <b>sub_call</b> starts indexed subroutines in protected memory (that were loaded and/or referenced by <b>load_sub</b>, <b>load_disk</b> or <b>copy_dst_src</b>) as well as indexed subroutines in the unprotected list area (that were referenced by <b>set_sub_pointer</b> or <b>copy_dst_src</b>).</li> <li>• If no subroutine is referenced for the supplied index, then the jump is suppressed and execution continues at the command located after the calling position. If applicable, a <b>list_continue</b> is executed.</li> </ul> <p><b>get_sub_pointer( Index )</b> can be used to determine whether a subroutine has been referenced for a particular index. If no subroutine has been referenced, this command returns the value "-1" (= <math>2^{32}-1</math>).</p> <ul style="list-style-type: none"> <li>• If <b>Index &gt; 1023</b>, then <b>sub_call</b> is, already during loading, replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>• Absolute vector and arc commands execute absolutely after being called with <b>sub_call</b>. If the subroutine needs to execute at various locations within the image field, then either the subroutine can only contain relative mark, arc or jump commands or <b>sub_call_abs</b> must be used instead.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>list_call</b> , <b>sub_call_abs</b> , <b>sub_call_cond</b>



<b>Undelayed Short List Command</b>	<b>sub_call_abs</b>
Function	Causes an unconditional jump to an indexed subroutine. In the called subroutine, any absolute vector commands and arc commands receive an offset (corresponding to the current coordinates at the time of the call).
Call	sub_call_abs( Index )
Parameters	Index      Index of the called indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].
Comments	<ul style="list-style-type: none"> <li>The <b>sub_call_abs</b> command reads the indexed subroutine's starting address from the internal management table based on the supplied index and then calls <b>list_call_abs</b> (see also the comments there). <b>list_call_abs</b> then triggers the jump to the subroutine.</li> <li>If the called subroutine contains no absolute commands, then there is no difference between <b>sub_call_abs</b> and <b>sub_call</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>sub_call, sub_call_abs_cond</b>

<b>Undelayed Short List Command</b>	<b>sub_call_abs_cond</b>						
Function	<i>Conditional call (AbsCall) of an indexed subroutine:</i> <b>sub_call_abs_cond</b> executes <b>sub_call_abs</b> ( Index ), if the current I0value at the 16-bit digital input port of the EXTENSION 1 socket connector meets the following condition:  $((\text{I0value AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not I0value}) \text{ AND Mask0}) = \text{Mask0})$ (= if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0). Otherwise, the directly following list command is immediately executed.						
Call	sub_call_abs_cond( Mask1, Mask0, Index )						
Parameters	<table> <tr> <td>Mask1</td> <td>16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</td> </tr> <tr> <td>Mask0</td> <td>See Mask1.</td> </tr> <tr> <td>Index</td> <td>Index of the called indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].</td> </tr> </table>	Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.	Mask0	See Mask1.	Index	Index of the called indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].
Mask1	16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.						
Mask0	See Mask1.						
Index	Index of the called indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].						
Comments	<ul style="list-style-type: none"> <li>See <b>sub_call_abs</b>.</li> <li>See also Chapter 9.3.2 "Conditional Command Execution", page 281.</li> </ul>						
RTC4→RTC6	New command.						
RTC5→RTC6	Unchanged functionality.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<b>sub_call_abs</b>						

<b>Undelayed Short List Command</b>	<b>sub_call_abs_repeat</b>
<b>Function</b>	Causes an unconditional jump to an indexed subroutine and executes its body several times.
<b>Call</b>	<code>sub_call_abs_repeat( Index, Number )</code>
<b>Parameters</b>	Index      Index of the to be called indexed subroutine (as with <a href="#">sub_call_abs</a> ). Number      Number of repetitions. 0 is treated as 1. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>sub_call_abs( Index )</code> is synonymous with <code>sub_call_abs_repeat( Index, 1 )</code>.</li> <li>• See <a href="#">sub_call_repeat</a> and <a href="#">sub_call_abs</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">sub_call_repeat</a> , <a href="#">sub_call_abs</a> , <a href="#">sub_call</a>

<b>Undelayed Short List Command</b>	<b>sub_call_cond</b>
<b>Function</b>	<i>Conditional call of an indexed subroutine:</i> <b>sub_call_cond</b> executes <a href="#">sub_call</a> ( Index ), if the current IValue at the 16-bit digital input port of the EXTENSION 1 socket connector meets the following condition: $((\text{IValue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IValue}) \text{ AND Mask0}) = \text{Mask0})$ (= if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0). Otherwise, the directly following list command is immediately executed.
<b>Call</b>	<code>sub_call_cond( Mask1, Mask0, Index )</code>
<b>Parameters</b>	Mask1      16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.  Mask0      See Mask1.  Index      Index of the to-be-called indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">sub_call</a>.</li> <li>• See also Chapter 9.3.2 "Conditional Command Execution", page 281.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">sub_call</a>



<b>Undelayed Short List Command</b>	<b>sub_call_repeat</b>
Function	Causes an unconditional jump to an indexed subroutine and executes its body several times.
Call	<code>sub_call_repeat( Index, Number )</code>
Parameters	<p>Index      Index of the to be called indexed subroutine (as with <a href="#">sub_call</a>).</p> <p>Number      Number of repetitions. As an unsigned 32-bit value. Number = 0 is treated like Number = 1.</p>
Comments	<ul style="list-style-type: none"> <li>• <a href="#">sub_call( Index )</a> is synonymous with <code>sub_call_repeat( Index, 1 )</code>.</li> <li>• <a href="#">sub_call_repeat</a> avoids an empty cycle at the repetition, which otherwise inevitably occurs with <a href="#">sub_call...sub_call</a> or <a href="#">list_repeat...sub_call...list_until</a> constructions.</li> <li>• By <a href="#">sub_call_repeat</a>, for example, trajectories (see Glossary entry on <a href="#">page 880</a>) from micro vector commands for runup curves and coast down curves can be seamlessly joined together with shapes from subroutines.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">sub_call_abs_repeat</a> , <a href="#">sub_call</a> , <a href="#">sub_call_abs</a> , <a href="#">micro_vector_abs</a> , <a href="#">micro_vector_abs_3d</a> , <a href="#">micro_vector_rel</a> , <a href="#">micro_vector_rel_3d</a>



<b>Undelayed Short List Command</b>	<b>switch_ioport</b>
<b>Function</b>	Executes a relative list jump <b>list_jump_rel</b> ( <i>Pos</i> ) whose jump distance <i>Pos</i> ( $> 1$ ) is determined at runtime by the current value ( <i>IOvalue</i> ) at the 16-bit digital input port of the EXTENSION 1 socket connector. You can specify which of the 16-bit digital input port's bits should be evaluated for this purpose.
<b>Call</b>	<code>switch_ioport( MaskBits, ShiftBits )</code>
<b>Parameters</b>	MaskBits      Number of contiguous bits of the 16-bit digital input port to be evaluated for determining the jump distance. As an unsigned 32-bit value. Allowed value range: [1...16].
	ShiftBits      Position of the least significant to-be-evaluated bit of the 16-bit digital input port. As an unsigned 32-bit value. Allowed value range: [0...15].
<b>Comments</b>	<ul style="list-style-type: none"> <li>With invalid values of MaskBits or ShiftBits and with <math>(\text{MaskBits} + \text{ShiftBits}) &gt; 16</math>, the command is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>The following applies: <math>\text{Mask} = ((1 &lt;&lt; \text{MaskBits}) - 1) &lt;&lt; \text{ShiftBits}</math> and <math>\text{SwitchNo} = (\text{Mask} \&amp; \text{IOvalue}) &gt;&gt; \text{ShiftBits}</math>. Here, a <b>list_jump_rel</b> ( <i>Pos</i> ) with <i>Pos</i> = (<i>SwitchNo</i> + 1) list positions are then executed.</li> <li>The jump distance is at least 1. This prevents infinite loops when no signal is present. Jumps to the same address (<i>Pos</i> = 0) are not possible with <b>switch_ioport</b>, but can be simulated by <b>list_jump_rel</b> ( -1 ) as the directly subsequent command.</li> <li>The maximum jump distance is <math>2^{16}</math> list positions.</li> <li>See also <b>list_jump_rel</b>.</li> <li>See also Section "16-Bit Digital Input and 16-Bit Digital Output", page 67 and Chapter 9.3.2 "Conditional Command Execution", page 281.</li> </ul>
<b>Example (Pascal)</b>	<ul style="list-style-type: none"> <li>It is assumed that the current value at the 16-bit digital input port is \$F152 at runtime: then <b>switch_ioport( \$0008, \$0004 )</b> executes <b>list_jump_rel</b> ( \$0016 ), that is, a relative list jump of length 22 list positions.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>list_jump_rel</b> , <b>list_jump_rel_cond</b>



Ctrl Command	sync_slaves
Function	<p><b>sync_slaves</b> is no longer necessary as of DLL 614, OUT 614, RBF 619. Synchronizes all slave boards (connected in a master/slave chain with the slave connection of the addressed RTC6 board) stably to the 10 µs clock period of the addressed RTC6 board.</p>
Call	sync_slaves()
Comments	<ul style="list-style-type: none"> <li>• For usage of <b>sync_slaves</b>, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 114</a>.</li> <li>• With RTC6 Software Package ≥ V1.5.2, the actions described below for RTC6 Software Packages &lt;V1.5.0 are no longer executed.</li> <li>• RTC6 Software Packages &lt;V1.5.0 (&lt; RBF 619): <ul style="list-style-type: none"> <li>– SCANLAB recommends executing synchronization immediately after all boards have been initialized by <a href="#">load_program_file</a> and <a href="#">load_correction_file</a>. Otherwise, all involved boards (that is, the master board and the downstream slave boards allocated to the user program) should already have been halted prior to the call of <b>sync_slaves</b>. To avoid irregularities during execution of <b>sync_slaves</b>, you should neither apply external stop signals to the boards nor trigger external starts (this is not automatically prevented).</li> <li>– During the course of <b>sync_slaves</b>, a <a href="#">simulate_ext_stop</a> is passed to the addressed board. This halts the addressed board and all downstream slave boards in the master/slave chain (including boards not allocated to the user program). Users themselves are responsible to ensure that any running processes are not disrupted by that.</li> <li>– After execution of <b>sync_slaves</b>, the scan system axes of all involved boards are in either the coordinate center position (0, 0 [,0]) or the HomeJump position (possibly shifted by an offset set by <a href="#">set_offset</a>, <a href="#">set_defocus</a> or <a href="#">set_hi</a>).</li> <li>– <b>sync_slaves</b> (relating to synchronization) only affects RTC6 boards connected in a master/slave chain to the Master connector of the addressed RTC6 board. It does not affect the addressed board itself or any boards connected to the Slave connector of the addressed board. Therefore, if all slave boards of a master/slave chain is to be synchronized with the master board, then <b>sync_slaves</b> must address the master board of the master/slave chain. <b>sync_slaves</b> has no effect, if no board is connected to the Master connector of the addressed board.</li> <li>– Synchronization of downstream slave boards by <b>sync_slaves</b> occurs even when the boards' <a href="#">BUSY</a> or <a href="#">INTERNAL-BUSY</a> status is set (they are automatically halted). Nevertheless, the only slave boards to get synchronized are those allocated to the user program (allocation is not requested automatically). If the user program possesses access rights for the addressed board but no further boards, then <b>sync_slaves</b> has no effect.</li> <li>– During the course of <b>sync_slaves</b>, all <a href="#">get_startstop_info</a> error bits are cleared on all boards (including upstream boards) allocated to the user program.</li> </ul> </li> </ul>



Ctrl Command	sync_slaves
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>sync_slaves</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the addressed board is currently set (list is being processed or has been paused by <b>pause_list</b>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the addressed board is set</li> </ul> </li> <li>• <b>sync_slaves</b> is even executed, if:           <ul style="list-style-type: none"> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED</b> status is set)</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality. However: Master/Slave functionality has been changed.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 615: <b>sync_slaves</b> has no function anymore.
References	<b>get_master_slave</b> , <b>get_sync_status</b> , <b>master_slave_config</b>



<b>Ctrl Command</b>	<b>time_control_eth</b>	
<b>Function</b>	Sets a parameter to fine-tune the accuracy of the real-time clock.	
<b>Call</b>	time_control_eth( PPM )	
<b>Parameters</b>	PPM	Deviation. As a 64-bit IEEE floating point value. Allowed value range: PPM / 4.34 = [-64...+63]
<b>Comments</b>	<ul style="list-style-type: none"><li>The execution of <b>time_control_eth</b> can take several 100 <math>\mu</math>s, see <a href="#">time_update</a>.</li><li><b>time_control_eth</b> is not executed with RTC6 PCIe Boards.</li><li>PPM means the deviation in parts per million. A positive value slows down the clock.</li></ul>	
RTC4→RTC6	New command.	
RTC5→RTC6	New command.	
<b>Version info</b>	Available as of version DLL 612, ETH 612, RBF 617.	
<b>References</b>	<a href="#">time_update</a>	



<b>Normal List Command</b>	<b>time_fix</b>
<b>Function</b>	Stores the current time and date of the RTC clock/calender in a cache for use with <b>mark_date</b> and <b>mark_time</b> .
<b>Call</b>	<code>time_fix()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>time_fix</b> is synonymous with <b>time_fix_f_off</b> with <code>FirstDay = 0</code> and <code>Offset = 0</code> (see comments there).</li> </ul>
<b>RTC4→RTC6</b>	<p>Unchanged functionality.</p> <p><b>time_fix</b> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.</p>
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>time_fix_f, time_fix_f_off</b>

<b>Normal List Command</b>	<b>time_fix_f</b>
<b>Function</b>	Stores the current time and date of the RTC clock/calender in a cache for use with <b>mark_date</b> and <b>mark_time</b> .
<b>Call</b>	<code>time_fix_f( FirstDay )</code>
<b>Parameters</b>	<code>FirstDay</code> Like <b>time_fix_f_off</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>time_fix_f</b> is synonymous with <b>time_fix_f_off</b> with <code>Offset = 0</code> (see comments there).</li> </ul>
<b>RTC4→RTC6</b>	New command.
<b>RTC5→RTC6</b>	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>time_fix_f, time_fix_f_off, time_update, mark_time, mark_date</b>



<b>Normal List Command</b>	<b>time_fix_f_off</b>
<b>Function</b>	Stores the current time and date of the RTC clock/calender or a forward-dated date and time for use with <b>mark_date</b> and <b>mark_time</b> in a cache.
<b>Call</b>	<code>time_fix_f_off( FirstDay, Offset )</code>
<b>Parameters</b>	FirstDay      Defines the starting number for determining the Julian calendar day from the current date of the RTC calendar: Counting proceeds from <code>FirstDay</code> to <code>FirstDay + 364</code> (+1 for leap years). As an unsigned 32-bit value.
	Offset      Forward dating in <i>seconds</i> . As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].
<b>Comments</b>	<ul style="list-style-type: none"> <li>Before calling <b>time_fix_f_off</b>, <b>time_fix_f</b> or <b>time_fix</b>, synchronization of the RTC6 and PC time should be performed (for RTC6 boards, at least once after each <b>load_program_file</b>) by <b>time_update</b>.</li> <li>The complete time can be marked through multiple calls of <b>mark_time</b> and the complete date through multiple calls of <b>mark_date</b>. <b>time_fix_f_off</b>, <b>time_fix_f</b> or <b>time_fix</b> must therefore be called <i>before</i> these marking commands so that the to-be-marked time or date do not change during marking.</li> <li>If <b>time_fix_f_off</b>, <b>time_fix_f</b> or <b>time_fix</b> are not called again before a time or date marking, then the last marked time is marked again. If <b>time_fix_f_off</b>, <b>time_fix_f</b> or <b>time_fix</b> are not called at all after a <b>load_program_file</b>, then a time of 00:00 or a date of January 1, 2000 is marked.</li> <li>If <code>Offset = 0</code>, then the current date and current time are fixed. One practical use of forward dating (<code>Offset &gt; 0</code>) is for setting a date of expiry based on the current date.</li> <li>Backdating (<code>Offset &lt; 0</code>) is not possible.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>time_fix</b> , <b>time_fix_f</b> , <b>time_update</b> , <b>mark_time</b> , <b>mark_date</b>



<b>Ctrl Command</b>	<b>time_update</b>
<b>Function</b>	Sets the 24-hour clock and calendar of the RTC6 board to the current PC time.
<b>Call</b>	<code>time_update()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>time_update</b> must be called after each <a href="#">load_program_file</a>, if 24-hour clock and calendar of the board are to be synchronized with the PC time.</li> <li>• The base value for internal time counting is set to January 1, 2000, 00:00 by <a href="#">load_program_file</a> whereas to the PC time by <b>time_update</b>. An internal seconds counter is set to 0 by <a href="#">load_program_file</a> or by <b>time_update</b>, but is always driven by the quartz-controlled 10 µs clock.</li> <li>• Before marking with <a href="#">mark_date</a> or <a href="#">mark_time</a>, you must call <a href="#">time_fix</a>, <a href="#">time_fix_f</a> or <a href="#">time_fix_f_off</a> so that the current time can be captured (as sum of the base value and the current value of the internal seconds counter) and formatted.</li> <li>• The RTC6 Ethernet Board is a real-time clock. Therefore, <b>time_update</b> must be called only once. <b>time_update</b> takes several hundred µs. During this time the 10 µs clock period of the RTC6 Ethernet Board is interrupted. Therefore, <b>time_update</b> should not be called during list processing. The clock continues to run even if the power supply is switched off (&gt; about 1 week). See also <a href="#">time_control_eth</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.  <b>time_update</b> is only available for the RTC4 SCANalone Board, which is the standalone version of the RTC4 board.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600. Last change version DLL 612, ETH 612, RBF 617: real-time clock of the RTC6 Ethernet Board.
References	<a href="#">time_fix</a> , <a href="#">time_fix_f</a> , <a href="#">time_fix_f_off</a> , <a href="#">mark_date</a> , <a href="#">mark_time</a> , <a href="#">time_control_eth</a>



<b>Normal List Command</b>	<b>timed_arc_abs</b>								
<b>Function</b>	Moves the laser focus for the specified marking duration from the current position along an arc with the specified angle and center point (absolute coordinate values) within a 2D image field.								
<b>Call</b>	<code>timed_arc_abs( X, Y, Angle, T )</code>								
<b>Parameters</b>	<table> <tr> <td>X</td><td>Like <a href="#">arc_abs</a>.</td></tr> <tr> <td>Y</td><td>Like <a href="#">arc_abs</a>.</td></tr> <tr> <td>Angle</td><td>Like <a href="#">arc_abs</a>.</td></tr> <tr> <td>T</td><td>           Duration of the complete arc marking process in <i>microseconds</i>.            As a 64-bit IEEE floating point value.            Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped.            If <math>T &lt; 5</math>, then <b>timed_arc_abs</b> behaves like <a href="#">arc_abs</a>.         </td></tr> </table>	X	Like <a href="#">arc_abs</a> .	Y	Like <a href="#">arc_abs</a> .	Angle	Like <a href="#">arc_abs</a> .	T	Duration of the complete arc marking process in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_arc_abs</b> behaves like <a href="#">arc_abs</a> .
X	Like <a href="#">arc_abs</a> .								
Y	Like <a href="#">arc_abs</a> .								
Angle	Like <a href="#">arc_abs</a> .								
T	Duration of the complete arc marking process in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_arc_abs</b> behaves like <a href="#">arc_abs</a> .								
<b>Comments</b>	<ul style="list-style-type: none"> <li>Unlike <a href="#">arc_abs</a>, <b>timed_arc_abs</b> does not execute the marking process with the specified (by <a href="#">set_mark_speed</a> or <a href="#">set_mark_speed_ctrl</a>) mark speed. Instead, the speed (that is, the number of microsteps) is adjusted so that the arc lasts as long as specified (see <a href="#">Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260</a>). The total marking time is (for <math>T \geq 5</math>) the sum of the specified (rounded) time and the set delays.</li> <li>See also comments on <a href="#">arc_abs</a>.</li> </ul>								
RTC4→RTC6	New command. See <a href="#">arc_abs</a> .								
RTC5→RTC6	Unchanged functionality. See <a href="#">arc_abs</a> .								
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.								
<b>References</b>	<a href="#">arc_abs</a> , <a href="#">timed_arc_rel</a>								



<b>Normal List Command</b>	<b>timed_arc_rel</b>
<b>Function</b>	Moves the laser focus for the specified marking duration from the current position along an arc with the specified angle and center point (relative coordinate values) within a 2D image field.
<b>Call</b>	<code>timed_arc_rel( dx, dy, Angle, T )</code>
<b>Parameters</b>	<p><code>dx</code> Like <a href="#">arc_rel</a>.</p> <p><code>dy</code> Like <a href="#">arc_rel</a>.</p> <p><code>Angle</code> Like <a href="#">arc_rel</a>.</p> <p><code>T</code> Duration of the complete arc marking process in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_arc_rel</b> behaves like <a href="#">arc_rel</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the arc center are to be supplied as relative coordinates with respect to the current position. Otherwise, the command is analogous to <a href="#">timed_arc_abs</a> (see the comments there).</li> </ul>
<b>RTC4→RTC6</b>	<p>New command.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC6 multiplies the specified values for <code>dx</code> and <code>dy</code> by 16. The allowed value ranges decrease accordingly.</p>
<b>RTC5→RTC6</b>	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">timed_arc_abs</a> , <a href="#">arc_rel</a>



<b>Normal List Command</b>	<b>timed_jump_abs</b>						
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 2D vector from the current position to the specified position (absolute coordinate values) within a 2D image field.						
<b>Call</b>	<code>timed_jump_abs( X, Y, T )</code>						
<b>Parameters</b>	<table> <tr> <td>X</td><td>Like <a href="#">jump_abs</a>.</td></tr> <tr> <td>Y</td><td>Like <a href="#">jump_abs</a>.</td></tr> <tr> <td>T</td><td> <p>Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_jump_abs</b> behaves like <a href="#">jump_abs</a>.</p> </td></tr> </table>	X	Like <a href="#">jump_abs</a> .	Y	Like <a href="#">jump_abs</a> .	T	<p>Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_jump_abs</b> behaves like <a href="#">jump_abs</a>.</p>
X	Like <a href="#">jump_abs</a> .						
Y	Like <a href="#">jump_abs</a> .						
T	<p>Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_jump_abs</b> behaves like <a href="#">jump_abs</a>.</p>						
<b>Comments</b>	<ul style="list-style-type: none"> <li>Unlike <a href="#">jump_abs</a>, the <b>timed_jump_abs</b> command does not execute the jump with the specified (by <a href="#">set_jump_speed</a> or <a href="#">set_jump_speed_ctrl</a>) jump speed. Instead, the speed (that is, the number of microsteps) is adjusted so that the vector lasts as long as specified, see <a href="#">Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260</a>. The total jump time is (for <math>T \geq 5</math>) the sum of the specified (rounded) time and the set delays.</li> <li>The “laser active” laser control signals are off during the jump (if necessary, they are switched off before the jump).</li> <li>After a jump command, a (variable) jump delay is inserted. Exception: a zero-length jump vector’s subsequent (variable) jump delay is not executed. However, the command itself still requires a 10 µs clock period for execution.</li> <li>During the conversion of specified coordinate values into scan system output values, any previously defined coordinate transformations, assigned correction tables etc. are taken into account after successful split-up into microsteps, see <a href="#">Chapter 7.3.6 "Output Values to the Scan System", page 170</a>.</li> <li>See also comments on <a href="#">jump_abs</a>.</li> </ul>						
RTC4→RTC6	Unchanged functionality. In addition: increased value range.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified values for X and Y by 16. The allowed value ranges decrease accordingly.						
RTC5→RTC6	Unchanged functionality. In addition: increased value range.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">jump_abs</a> , <a href="#">timed_jump_rel</a> , <a href="#">timed_jump_abs_3d</a>						



<b>Normal List Command</b>	<b>timed_jump_abs_3d</b>								
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 3D vector from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> .								
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_jump_abs_3d</b> has the same effect as <b>timed_jump_abs</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.								
<b>Call</b>	<code>timed_jump_abs_3d( X, Y, Z, T )</code>								
<b>Parameters</b>	<table> <tr> <td>X</td> <td>Like <b>jump_abs_3d</b>.</td> </tr> <tr> <td>Y</td> <td>Like <b>jump_abs_3d</b>.</td> </tr> <tr> <td>Z</td> <td>Like <b>jump_abs_3d</b>.</td> </tr> <tr> <td>T</td> <td>Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_jump_abs_3d</b> behaves like <b>jump_abs_3d</b>.</td> </tr> </table>	X	Like <b>jump_abs_3d</b> .	Y	Like <b>jump_abs_3d</b> .	Z	Like <b>jump_abs_3d</b> .	T	Duration of the complete jump vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_jump_abs_3d</b> behaves like <b>jump_abs_3d</b> .
X	Like <b>jump_abs_3d</b> .								
Y	Like <b>jump_abs_3d</b> .								
Z	Like <b>jump_abs_3d</b> .								
T	Duration of the complete jump vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_jump_abs_3d</b> behaves like <b>jump_abs_3d</b> .								
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>timed_jump_abs_3d</b> functions similarly to the <b>timed_jump_abs</b> command (see the comments there).</li> <li>See also comments on <b>jump_abs_3d</b>.</li> </ul>								
RTC4→RTC6	New command. See <b>jump_abs_3d</b> .								
RTC5→RTC6	Unchanged functionality. In addition: increased value range.								
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.								
<b>References</b>	<b>timed_jump_abs</b> , <b>jump_abs_3d</b> , <b>timed_jump_rel_3d</b>								



<b>Normal List Command</b>	<b>timed_jump_rel</b>
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 2D vector from the current position to the specified position (relative coordinate values) within a 2D image field.
<b>Call</b>	<code>timed_jump_rel( dx, dy, T )</code>
<b>Parameters</b>	<p><code>dx</code>      Like <a href="#">jump_rel</a>.</p> <p><code>dy</code>      Like <a href="#">jump_rel</a>.</p> <p><code>T</code>      Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <code>T &lt; 5</code>, then <b>timed_jump_rel</b> behaves like <a href="#">jump_rel</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, the command is identical to <a href="#">timed_jump_abs</a> (see the comments there).</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. See <a href="#">jump_rel</a> .
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">timed_jump_abs</a> , <a href="#">jump_rel</a> , <a href="#">timed_jump_rel_3d</a>



<b>Normal List Command</b>	<b>timed_jump_rel_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 3D vector from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_jump_rel_3d</b> has the same effect as <b>timed_jump_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
<b>Call</b>	<code>timed_jump_rel_3d( dx, dy, dz, T )</code>
<b>Parameters</b>	<p><b>dx</b>      Like <b>jump_rel_3d</b>.</p> <p><b>dy</b>      Like <b>jump_rel_3d</b>.</p> <p><b>dz</b>      Like <b>jump_rel_3d</b>.</p> <p><b>T</b>      Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_jump_rel_3d</b> behaves like <b>jump_rel_3d</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, the command is identical to <b>timed_jump_abs_3d</b> (see the comments there).</li> </ul>
RTC4→RTC6	New command. See <b>jump_rel_3d</b> .
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>timed_jump_abs_3d, jump_rel_3d, timed_jump_rel</b>



<b>Normal List Command</b>	<b>timed_mark_abs</b>						
<b>Function</b>	Moves the laser focus for the specified marking duration along a 2D vector from the current position to the specified position (absolute coordinate values) within a 2D image field.						
<b>Call</b>	<code>timed_mark_abs( X, Y, T )</code>						
<b>Parameters</b>	<table> <tr> <td>X</td><td>Like <a href="#">mark_abs</a>.</td></tr> <tr> <td>Y</td><td>Like <a href="#">mark_abs</a>.</td></tr> <tr> <td>T</td><td>Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_mark_abs</b> behaves like <a href="#">mark_abs</a>.</td></tr> </table>	X	Like <a href="#">mark_abs</a> .	Y	Like <a href="#">mark_abs</a> .	T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_mark_abs</b> behaves like <a href="#">mark_abs</a> .
X	Like <a href="#">mark_abs</a> .						
Y	Like <a href="#">mark_abs</a> .						
T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_mark_abs</b> behaves like <a href="#">mark_abs</a> .						
<b>Comments</b>	<ul style="list-style-type: none"> <li>Unlike <a href="#">mark_abs</a>, the <b>timed_mark_abs</b> command does not execute the marking process with the specified (by <a href="#">set_mark_speed</a> or <a href="#">set_mark_speed_ctrl</a>) mark speed. Instead, the speed (that is, the number of microsteps) is adjusted so that the vector lasts as long as specified (see <a href="#">Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260</a>). The total marking time is (for <math>T \geq 5</math>) the sum of the specified (rounded) time and the set delays.</li> <li>See also comments on <a href="#">mark_abs</a>.</li> </ul>						
RTC4→RTC6	Unchanged functionality. In addition: increased value range. See <a href="#">mark_abs</a> .						
RTC5→RTC6	Unchanged functionality. In addition: increased value range.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">mark_abs</a> , <a href="#">timed_mark_rel</a> , <a href="#">timed_mark_abs_3d</a>						



<b>Normal List Command</b>	<b>timed_mark_abs_3d</b>								
<b>Function</b>	Moves the laser focus for the specified marking duration along a 3D vector from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> .								
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_mark_abs_3d</b> has the same effect as <b>timed_mark_abs</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.								
<b>Call</b>	<code>timed_mark_abs_3d( X, Y, Z, T )</code>								
<b>Parameters</b>	<table> <tr> <td>X</td> <td>Like <b>mark_abs_3d</b>.</td> </tr> <tr> <td>Y</td> <td>Like <b>mark_abs_3d</b>.</td> </tr> <tr> <td>Z</td> <td>Like <b>mark_abs_3d</b>.</td> </tr> <tr> <td>T</td> <td>Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_mark_abs_3d</b> behaves like <b>mark_abs_3d</b>.</td> </tr> </table>	X	Like <b>mark_abs_3d</b> .	Y	Like <b>mark_abs_3d</b> .	Z	Like <b>mark_abs_3d</b> .	T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_mark_abs_3d</b> behaves like <b>mark_abs_3d</b> .
X	Like <b>mark_abs_3d</b> .								
Y	Like <b>mark_abs_3d</b> .								
Z	Like <b>mark_abs_3d</b> .								
T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_mark_abs_3d</b> behaves like <b>mark_abs_3d</b> .								
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>timed_mark_abs_3d</b> functions similarly to <b>timed_mark_abs</b> (see the comments there).</li> <li>See also comments on <b>mark_abs_3d</b>.</li> </ul>								
RTC4→RTC6	New command. See <b>mark_abs_3d</b> .								
RTC5→RTC6	Unchanged functionality. In addition: increased value range.								
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.								
<b>References</b>	<b>timed_mark_abs</b> , <b>mark_abs_3d</b> , <b>timed_mark_rel_3d</b>								



<b>Normal List Command</b>	<b>timed_mark_rel</b>
<b>Function</b>	Moves the laser focus for the specified marking duration along a 2D vector from the current position to the specified position (relative coordinate values) within a 2D image field.
<b>Call</b>	<code>timed_mark_rel( dX, dY, T )</code>
<b>Parameters</b>	<p><code>dX</code> Like <a href="#">mark_rel</a>.</p> <p><code>dY</code> Like <a href="#">mark_rel</a>.</p> <p><code>T</code> Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. (If <math>T &lt; 5</math>, then <b>timed_mark_rel</b> behaves like <a href="#">mark_rel</a>).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>timed_mark_rel</b> is analogous to <a href="#">timed_mark_abs</a> (see the comments there).</li> </ul>
RTC4→RTC6	Unchanged functionality. In addition: increased value range. See <a href="#">mark_rel</a> .
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">timed_mark_abs</a> , <a href="#">mark_rel</a> , <a href="#">timed_mark_rel_3d</a>



<b>Normal List Command</b>	<b>timed_mark_rel_3d</b>
<b>Function</b>	Moves the laser focus for the specified marking duration along a 3D vector from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> .
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_mark_rel_3d</b> has the same effect as <b>timed_mark_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
<b>Call</b>	<code>timed_mark_rel_3d( dx, dy, dz, T )</code>
<b>Parameters</b>	<p><b>dx</b>      Like <b>mark_rel_3d</b>.</p> <p><b>dy</b>      Like <b>mark_rel_3d</b>.</p> <p><b>dz</b>      Like <b>mark_rel_3d</b>.</p> <p><b>T</b>      Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_mark_rel_3d</b> behaves like <b>mark_rel_3d</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, the command is identical to <b>timed_mark_abs_3d</b> (see the comments there).</li> </ul>
RTC4→RTC6	New command.. See <b>mark_rel_3d</b> .
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<b>timed_mark_abs_3d, timed_mark_abs, mark_rel_3d, timed_mark_rel</b>



<b>Normal List Command</b>	<b>timed_para_jump_abs</b>								
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 2D vector from the current position to the specified position (absolute coordinate values) within a 2D image field and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_ctrl</b> to the specified value.								
<b>Call</b>	<code>timed_para_jump_abs( X, Y, P, T )</code>								
<b>Parameters</b>	<table> <tr> <td>X</td><td>Like <b>para_jump_abs</b>.</td></tr> <tr> <td>Y</td><td>Like <b>para_jump_abs</b>.</td></tr> <tr> <td>P</td><td>Like <b>para_jump_abs</b>.</td></tr> <tr> <td>T</td><td>Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_para_jump_abs</b> behaves like <b>para_jump_abs</b>.</td></tr> </table>	X	Like <b>para_jump_abs</b> .	Y	Like <b>para_jump_abs</b> .	P	Like <b>para_jump_abs</b> .	T	Duration of the complete jump vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_jump_abs</b> behaves like <b>para_jump_abs</b> .
X	Like <b>para_jump_abs</b> .								
Y	Like <b>para_jump_abs</b> .								
P	Like <b>para_jump_abs</b> .								
T	Duration of the complete jump vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_jump_abs</b> behaves like <b>para_jump_abs</b> .								
<b>Comments</b>	<ul style="list-style-type: none"> <li>Unlike <b>para_jump_abs</b>, the <b>timed_para_jump_abs</b> command does not execute the jump with the specified (by <b>set_jump_speed</b> or <b>set_jump_speed_ctrl</b>) jump speed. Instead, the speed (that is, the number of microsteps) is adjusted so that the vector lasts as long as specified (see <a href="#">Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260</a>). The total jump time is (for <math>T \geq 5</math>) the sum of the specified (rounded) time and the set delays.</li> <li><b>timed_para_jump_abs</b> requires two list entries. During runtime, the command's part on the first list entry is executed as a short list command and afterward the second part is executed as a normal list command. Thereby, both command parts are executed within the same <math>10\ \mu s</math> clock, unless further (previous) short list commands induce a <b>list_continue</b> between the two parts.</li> <li>See also comments on <b>para_jump_abs</b>.</li> </ul>								
RTC4→RTC6	New command. See <b>para_jump_abs</b> .								
RTC5→RTC6	Unchanged functionality. In addition: increased value range.								
Version info	Available as of version DLL 600, OUT 600, RBF 600.								
References	<b>para_jump_abs</b> , <b>timed_jump_abs</b> , <b>jump_abs</b> , <b>timed_para_jump_rel</b> , <b>timed_para_jump_abs_3d</b>								



<b>Multiple List Command</b>	<b>timed_para_jump_abs_3d</b>										
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 3D vector from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.										
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_para_jump_abs_3d</b> has the same effect as <b>timed_para_jump_abs</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.										
<b>Call</b>	<b>timed_para_jump_abs_3d( X, Y, Z, P, T )</b>										
<b>Parameters</b>	<table> <tr> <td>X</td> <td>Like <b>para_jump_abs_3d</b>.</td> </tr> <tr> <td>Y</td> <td>Like <b>para_jump_abs_3d</b>.</td> </tr> <tr> <td>Z</td> <td>Like <b>para_jump_abs_3d</b>.</td> </tr> <tr> <td>P</td> <td>Like <b>para_jump_abs_3d</b>.</td> </tr> <tr> <td>T</td> <td>Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_para_jump_abs_3d</b> behaves like <b>para_jump_abs_3d</b>.</td> </tr> </table>	X	Like <b>para_jump_abs_3d</b> .	Y	Like <b>para_jump_abs_3d</b> .	Z	Like <b>para_jump_abs_3d</b> .	P	Like <b>para_jump_abs_3d</b> .	T	Duration of the complete jump vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_jump_abs_3d</b> behaves like <b>para_jump_abs_3d</b> .
X	Like <b>para_jump_abs_3d</b> .										
Y	Like <b>para_jump_abs_3d</b> .										
Z	Like <b>para_jump_abs_3d</b> .										
P	Like <b>para_jump_abs_3d</b> .										
T	Duration of the complete jump vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_jump_abs_3d</b> behaves like <b>para_jump_abs_3d</b> .										
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>timed_para_jump_abs_3d</b> functions similarly to <b>timed_para_jump_abs</b> (see the comments there).</li> <li>For <math>T \geq 5</math>, <b>timed_para_jump_abs_3d</b> occupies two list storage positions. The initial component executes as an undelayed short list command before the principal part (a normal list command).</li> <li>See also comments on <b>para_jump_abs_3d</b>.</li> </ul>										
RTC4→RTC6	New command. See <b>para_jump_abs_3d</b> .										
RTC5→RTC6	Unchanged functionality. In addition: increased value range.										
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.										
<b>References</b>	<b>timed_para_jump_abs</b> , <b>para_jump_abs_3d</b> , <b>jump_abs_3d</b> , <b>timed_para_jump_rel_3d</b>										



<b>Normal List Command</b>	<b>timed_para_jump_rel</b>
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 2D vector from the current position to the specified position (relative coordinate values) within a 2D image field and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
<b>Call</b>	<code>timed_para_jump_rel( dx, dy, p, T )</code>
<b>Parameters</b>	<p>dx      Like <a href="#">para_jump_rel</a>.</p> <p>dy      Like <a href="#">para_jump_rel</a>.</p> <p>p      Like <a href="#">para_jump_rel</a>.</p> <p>T      Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. (If <math>T &lt; 5</math>, then <b>timed_para_jump_rel</b> behaves like <a href="#">para_jump_rel</a>).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, the command is analogous to <a href="#">timed_para_jump_abs</a> (see the comments there).</li> </ul>
RTC4→RTC6	New command. See <a href="#">para_jump_rel</a> .
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">timed_para_jump_abs</a> , <a href="#">para_jump_rel</a> , <a href="#">timed_jump_rel</a> , <a href="#">timed_para_jump_rel_3d</a>



<b>Multiple List Command</b>	<b>timed_para_jump_rel_3d</b>
<b>Function</b>	Moves the output point (of the laser focus) for the specified jump duration along a 3D vector from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_jump_rel_3d</b> has the same effect as <b>timed_para_jump_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
<b>Call</b>	<code>timed_para_jump_rel_3d( dx, dy, dz, P, T )</code>
<b>Parameters</b>	<p><b>dx</b>      Like <b>para_jump_rel_3d</b>.</p> <p><b>dy</b>      Like <b>para_jump_rel_3d</b>.</p> <p><b>dz</b>      Like <b>para_jump_rel_3d</b>.</p> <p><b>P</b>      Like <b>para_jump_rel_3d</b>.</p> <p><b>T</b>      Duration of the complete jump vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <b>T &lt; 5</b>, then <b>timed_para_jump_rel_3d</b> behaves like <b>para_jump_rel_3d</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the jump vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>timed_para_jump_rel_3d</b> is analogous to <b>timed_para_jump_abs_3d</b> (see comments there).</li> <li>For <b>T ≥ 5</b>, <b>timed_para_jump_rel_3d</b> occupies two list storage positions. The initial component executes as an undelayed short list command before the principal part (a normal list command).</li> </ul>
<b>RTC4→RTC6</b>	New command. See <b>para_jump_rel_3d</b> .
<b>RTC5→RTC6</b>	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>timed_para_jump_abs_3d</b> , <b>para_jump_rel_3d</b> , <b>timed_jump_rel_3d</b> , <b>timed_para_jump_rel</b>



<b>Normal List Command</b>	<b>timed_para_mark_abs</b>								
<b>Function</b>	Moves the laser focus for the specified marking duration along a 2D vector from the current position to the specified position (absolute coordinate values) within a 2D image field and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.								
<b>Call</b>	<code>timed_para_mark_abs( X, Y, P, T )</code>								
<b>Parameters</b>	<table> <tr> <td>X</td> <td>Like <b>para_mark_abs</b>.</td> </tr> <tr> <td>Y</td> <td>Like <b>para_mark_abs</b>.</td> </tr> <tr> <td>P</td> <td>Like <b>para_mark_abs</b>.</td> </tr> <tr> <td>T</td> <td>Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_para_mark_abs</b> behaves like <b>para_mark_abs</b>.</td> </tr> </table>	X	Like <b>para_mark_abs</b> .	Y	Like <b>para_mark_abs</b> .	P	Like <b>para_mark_abs</b> .	T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_mark_abs</b> behaves like <b>para_mark_abs</b> .
X	Like <b>para_mark_abs</b> .								
Y	Like <b>para_mark_abs</b> .								
P	Like <b>para_mark_abs</b> .								
T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_mark_abs</b> behaves like <b>para_mark_abs</b> .								
<b>Comments</b>	<ul style="list-style-type: none"> <li>Unlike <b>para_mark_abs</b>, the <b>timed_para_mark_abs</b> command does not execute the marking process with the specified (by <b>set_mark_speed</b> or <b>set_mark_speed_ctrl</b>) mark speed. Instead, the speed (that is, the number of microsteps) is adjusted so that the vector lasts as long as specified (see <b>Chapter 8.9 "Timed Vector Commands and Timed Arc Commands", page 260</b>). The total marking time is (for <math>T \geq 5</math>) the sum of the specified (rounded) time and the set delays.</li> <li><b>timed_para_mark_abs</b> requires two list entries. During runtime, the command's part on the first list entry is executed as a short list command and afterward the second part is executed as a normal list command. Thereby, both command parts are executed within the same <math>10\ \mu s</math> clock, unless further (previous) short list commands induce a <b>list_continue</b> between the two parts.</li> <li>If <math>T &lt; 5</math>, then <b>timed_para_mark_abs</b> behaves like <b>para_mark_abs</b>. Then it requires only one list entry.</li> <li>See also comments on <b>para_mark_abs</b>.</li> </ul>								
RTC4→RTC6	New command. See <b>para_mark_abs</b> .								
RTC5→RTC6	Unchanged functionality. In addition: increased value range.								
Version info	Available as of version DLL 600, OUT 600, RBF 600.								
References	<b>para_mark_abs</b> , <b>timed_mark_abs</b> , <b>mark_abs</b> , <b>timed_para_mark_rel</b> , <b>timed_para_mark_abs_3d</b>								



<b>Multiple List Command</b>	<b>timed_para_mark_abs_3d</b>										
<b>Function</b>	Moves the laser focus for the specified marking duration along a 3D vector from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.										
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_para_mark_abs_3d</b> has the same effect as <b>timed_para_mark_abs</b> . However, the split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.										
<b>Call</b>	<code>timed_para_mark_abs_3d( X, Y, Z, P, T )</code>										
<b>Parameters</b>	<table> <tr> <td>X</td> <td>Like <b>para_mark_abs_3d</b>.</td> </tr> <tr> <td>Y</td> <td>Like <b>para_mark_abs_3d</b>.</td> </tr> <tr> <td>Z</td> <td>Like <b>para_mark_abs_3d</b>.</td> </tr> <tr> <td>P</td> <td>Like <b>para_mark_abs_3d</b>.</td> </tr> <tr> <td>T</td> <td>Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_para_mark_abs_3d</b> behaves like <b>para_mark_abs_3d</b>.</td> </tr> </table>	X	Like <b>para_mark_abs_3d</b> .	Y	Like <b>para_mark_abs_3d</b> .	Z	Like <b>para_mark_abs_3d</b> .	P	Like <b>para_mark_abs_3d</b> .	T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_mark_abs_3d</b> behaves like <b>para_mark_abs_3d</b> .
X	Like <b>para_mark_abs_3d</b> .										
Y	Like <b>para_mark_abs_3d</b> .										
Z	Like <b>para_mark_abs_3d</b> .										
P	Like <b>para_mark_abs_3d</b> .										
T	Duration of the complete mark vector in <i>microseconds</i> . As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If $T < 5$ , then <b>timed_para_mark_abs_3d</b> behaves like <b>para_mark_abs_3d</b> .										
<b>Comments</b>	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>timed_para_mark_abs_3d</b> functions similarly to <b>timed_para_mark_abs</b> (see the comments there).</li> <li><b>timed_para_mark_abs_3d</b> occupies two list storage positions. The initial component executes as an undelayed short list command before the principal part (a normal list command).</li> <li>See also comments on <b>para_mark_abs_3d</b>.</li> </ul>										
RTC4→RTC6	New command. See <b>para_mark_abs_3d</b> .										
RTC5→RTC6	Unchanged functionality. In addition: increased value range.										
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.										
<b>References</b>	<b>timed_para_mark_abs</b> , <b>para_mark_abs_3d</b> , <b>mark_abs_3d</b> , <b>timed_para_mark_rel_3d</b>										



<b>Normal List Command</b>	<b>timed_para_mark_rel</b>
<b>Function</b>	Moves the laser focus for the specified marking duration along a 2D vector from the current position to the specified position (relative coordinate values) within a 2D image field and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
<b>Call</b>	<code>timed_para_mark_rel( dx, dy, P, T )</code>
<b>Parameters</b>	<p>dx      Like <a href="#">para_mark_rel</a>.</p> <p>dy      Like <a href="#">para_mark_rel</a>.</p> <p>P      Like <a href="#">para_mark_rel</a>.</p> <p>T      Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If T &lt; 5, then <b>timed_para_mark_rel</b> behaves like <a href="#">para_mark_rel</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, the command is analogous to <b>timed_para_mark_abs</b> (see the comments there).</li> <li>See also comments on <a href="#">para_mark_rel</a>.</li> </ul>
RTC4→RTC6	New command. See <a href="#">para_mark_rel</a> .
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">timed_para_mark_abs</a> , <a href="#">para_mark_rel</a> , <a href="#">timed_mark_rel</a> , <a href="#">timed_para_mark_rel_3d</a>



<b>Multiple List Command</b>	<b>timed_para_mark_rel_3d</b>
<b>Function</b>	Moves the laser focus for the specified marking duration along a 3D vector from the current position to the specified position (relative coordinate values) within the <b>3D image field</b> . Simultaneously varies the signal parameter selected by <b>set_vector_control</b> to the specified value.
<b>Restriction</b>	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <b>select_cor_table</b> ), then <b>timed_para_mark_rel_3d</b> has the same effect as <b>timed_para_mark_rel</b> . However, split-up into microsteps is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
<b>Call</b>	<code>timed_para_mark_rel_3d( dx, dy, dz, P, T )</code>
<b>Parameters</b>	<p>dx      Like <b>para_mark_rel_3d</b>.</p> <p>dy      Like <b>para_mark_rel_3d</b>.</p> <p>dz      Like <b>para_mark_rel_3d</b>.</p> <p>P      Like <b>para_mark_rel_3d</b>.</p> <p>T      Duration of the complete mark vector in <i>microseconds</i>. As a 64-bit IEEE floating point value. Allowed value range: [0...167,772,160]. The parameter is rounded to an integer-multiple of 10. Out-of-range values are clipped. If <math>T &lt; 5</math>, then <b>timed_para_mark_rel_3d</b> behaves like <b>para_mark_rel_3d</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The coordinates for the mark vector's end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>timed_para_mark_rel_3d</b> is analogous to <b>timed_para_mark_abs_3d</b> (see the comments there).</li> <li><b>timed_para_mark_rel_3d</b> occupies two list storage positions. The initial component executes as an undelayed short list command before the principal part (a normal list command).</li> <li>See also comments on <b>para_mark_rel_3d</b>.</li> </ul>
RTC4→RTC6	New command. See <b>para_mark_rel_3d</b> .
RTC5→RTC6	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<b>timed_para_mark_abs_3d</b> , <b>para_mark_rel_3d</b> , <b>timed_mark_rel_3d</b> , <b>timed_para_mark_rel</b>



<b>Ctrl Command</b>	<b>transform</b>																											
<b>Function</b>	Performs a backward transformation of individual position values.																											
<b>Call</b>	TransformErrorCode = transform( &Sig1, &Sig2, Ptr, Code )																											
<b>Parameters and Returned parameter values</b>	<p>Sig1      Parameters: to-be-transformed position values. As pointers to signed 32-bit values.</p> <p>Sig2      Returned parameter values: transformed position values. As signed 32-bit values (the input values are overwritten).</p>																											
<b>Parameters</b>	Ptr	Pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit or 64-bit value) to the area of PC main memory to which the correction and transformation settings for backward transformation were previously transferred by <a href="#">upload_transform</a> .																										
	Code	<p>Controls aspects of the backward transformation, particularly which partial transformations to perform: If a partial transformation is <i>not</i> to be performed, then its corresponding bit (#2...#5) should be set to 1. As an unsigned 32-bit value.</p> <p>The parameter's meaning is similar to that of <a href="#">get_transform</a> (Sig1 corresponds to Ptr1 and Sig2 to Ptr2).</p> <p>If Bit #0 = 0, then both supplied position values (Sig1 and Sig2) are backward transformed as xy coordinates:</p> <table> <tr> <td>Bit #1</td> <td>= 0:</td> <td>The value supplied by Sig1 is backward transformed as the x coordinate and the value supplied by Sig2 as the y coordinate.</td> </tr> <tr> <td></td> <td>= 1:</td> <td>The value supplied by Sig1 is backward transformed as the y coordinate and the value supplied by Sig2 as the x coordinate.</td> </tr> <tr> <td>Bit #2</td> <td>= 0:</td> <td>The gain/offset correction of automatic self-calibration is backward transformed.</td> </tr> <tr> <td>Bit #3</td> <td>= 0:</td> <td>The image field correction is backward transformed.</td> </tr> <tr> <td>Bit #4</td> <td>= 0:</td> <td>The offset of the defined coordinate transformation is backward transformed.</td> </tr> <tr> <td>Bit #5</td> <td>= 0:</td> <td>The total matrix of the defined coordinate transformation is backward transformed.</td> </tr> <tr> <td>Bit #6</td> <td>Reserved.</td> <td></td> </tr> <tr> <td></td> <td>...</td> <td></td> </tr> <tr> <td>Bit #31</td> <td></td> <td></td> </tr> </table>	Bit #1	= 0:	The value supplied by Sig1 is backward transformed as the x coordinate and the value supplied by Sig2 as the y coordinate.		= 1:	The value supplied by Sig1 is backward transformed as the y coordinate and the value supplied by Sig2 as the x coordinate.	Bit #2	= 0:	The gain/offset correction of automatic self-calibration is backward transformed.	Bit #3	= 0:	The image field correction is backward transformed.	Bit #4	= 0:	The offset of the defined coordinate transformation is backward transformed.	Bit #5	= 0:	The total matrix of the defined coordinate transformation is backward transformed.	Bit #6	Reserved.			...		Bit #31	
Bit #1	= 0:	The value supplied by Sig1 is backward transformed as the x coordinate and the value supplied by Sig2 as the y coordinate.																										
	= 1:	The value supplied by Sig1 is backward transformed as the y coordinate and the value supplied by Sig2 as the x coordinate.																										
Bit #2	= 0:	The gain/offset correction of automatic self-calibration is backward transformed.																										
Bit #3	= 0:	The image field correction is backward transformed.																										
Bit #4	= 0:	The offset of the defined coordinate transformation is backward transformed.																										
Bit #5	= 0:	The total matrix of the defined coordinate transformation is backward transformed.																										
Bit #6	Reserved.																											
	...																											
Bit #31																												



Ctrl Command	transform																
Parameters (cont'd)	<p>Code (cont'd) If Bit #0 = 1, then one of the two supplied position values (specifiable as Sig1 or Sig2) is backward transformed as the z coordinate:</p> <table> <tr> <td>Bit #1 = 0:</td><td>The value supplied by Sig1 is backward transformed as the z coordinate (Sig2 remains unchanged).</td></tr> <tr> <td>      = 1:</td><td>The value supplied by Sig2 is backward transformed as the z coordinate (Sig1 remains unchanged).</td></tr> <tr> <td>Bit #2 = 0:</td><td>The offset to the focal length defined by <a href="#">set_defocus</a> or <a href="#">set_defocus_list</a> is backward transformed.</td></tr> <tr> <td>Bit #3 = 0:</td><td>The ABC correction is backward transformed.</td></tr> <tr> <td>Bit #4 = 0:</td><td>The offset to the z coordinate defined by <a href="#">set_offset_xyz</a> or <a href="#">set_offset_xyz_list</a> is backward transformed.</td></tr> <tr> <td>Bit #5</td><td>Reserved.</td></tr> <tr> <td>...</td><td></td></tr> <tr> <td>Bit #31</td><td></td></tr> </table>	Bit #1 = 0:	The value supplied by Sig1 is backward transformed as the z coordinate (Sig2 remains unchanged).	= 1:	The value supplied by Sig2 is backward transformed as the z coordinate (Sig1 remains unchanged).	Bit #2 = 0:	The offset to the focal length defined by <a href="#">set_defocus</a> or <a href="#">set_defocus_list</a> is backward transformed.	Bit #3 = 0:	The ABC correction is backward transformed.	Bit #4 = 0:	The offset to the z coordinate defined by <a href="#">set_offset_xyz</a> or <a href="#">set_offset_xyz_list</a> is backward transformed.	Bit #5	Reserved.	...		Bit #31	
Bit #1 = 0:	The value supplied by Sig1 is backward transformed as the z coordinate (Sig2 remains unchanged).																
= 1:	The value supplied by Sig2 is backward transformed as the z coordinate (Sig1 remains unchanged).																
Bit #2 = 0:	The offset to the focal length defined by <a href="#">set_defocus</a> or <a href="#">set_defocus_list</a> is backward transformed.																
Bit #3 = 0:	The ABC correction is backward transformed.																
Bit #4 = 0:	The offset to the z coordinate defined by <a href="#">set_offset_xyz</a> or <a href="#">set_offset_xyz_list</a> is backward transformed.																
Bit #5	Reserved.																
...																	
Bit #31																	
Result	<p>Error code. As an unsigned 32-bit value.</p> <table> <tr> <td>Value</td><td>Description</td></tr> <tr> <td>0</td><td>Success.</td></tr> <tr> <td>1</td><td>Ptr = <a href="#">NULL</a> (no memory area specified).</td></tr> <tr> <td>2</td><td>No valid data at Ptr (<a href="#">upload_transform</a> did not execute).</td></tr> <tr> <td>3</td><td>Erroneous data at Ptr (a corresponding error indication has been stored by <a href="#">upload_transform</a>).</td></tr> <tr> <td>4</td><td>z axis inversion not possible.</td></tr> </table>	Value	Description	0	Success.	1	Ptr = <a href="#">NULL</a> (no memory area specified).	2	No valid data at Ptr ( <a href="#">upload_transform</a> did not execute).	3	Erroneous data at Ptr (a corresponding error indication has been stored by <a href="#">upload_transform</a> ).	4	z axis inversion not possible.				
Value	Description																
0	Success.																
1	Ptr = <a href="#">NULL</a> (no memory area specified).																
2	No valid data at Ptr ( <a href="#">upload_transform</a> did not execute).																
3	Erroneous data at Ptr (a corresponding error indication has been stored by <a href="#">upload_transform</a> ).																
4	z axis inversion not possible.																
Comments	<ul style="list-style-type: none"> <li>For backward transformation of position values see <a href="#">Chapter 8.1.3 "Monitoring the Positioning"</a>, page 201.</li> <li>The execution of <b>transform</b> must be preceded by a call to <a href="#">upload_transform</a>. Additionally, position values should have been requested by <a href="#">get_values</a>.</li> <li>If execution of <b>transform</b> results in an error (returned error code &gt; 0), then no transformation occurs (Sig1 and Sig2 then remain unchanged). Errors also include Ptr = <a href="#">NULL</a> (error code = 1) or errors resulting from prior, erroneous execution of <a href="#">upload_transform</a> (error code = 3).</li> <li>If backward transformation of Z values is requested (Code Bit #0 = 1), but only a 2D correction table was assigned at the timepoint of the prior successful call to <a href="#">upload_transform</a>, then the offsets to the focal length and z coordinates are initialized with 0 and the values A, B and C are initialized with 0, 1, 0 (1-to-1 backward transformation).</li> <li>For backward transformation of xy position values (Code Bit #0 = 0), only the Z = 0 plane is transformed. xy stretching and Z defocus resulting from Z deviations (particularly with non-F-Theta systems) are not taken into account.</li> </ul>																



Ctrl Command	transform
Comments (cont'd)	<ul style="list-style-type: none"> <li>Because the command <b>transform</b> does not access any RTC6 boards, calling it does not require explicit access rights to a specific board. If both the <a href="#">upload_transform</a> data and the queried data recorded by <a href="#">get_values</a> or <a href="#">get_waveform</a> have been binarily stored on the PC, then offline operation of <b>transform</b> is also possible (then <b>transform</b> does not require the presence of an RTC6 board on the PCIe bus).</li> <li><b>transform</b> is not available as a multi-board command.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <b>transform</b>.</li> </ul>
RTC4→RTC6	<p>New command.</p> <p>In the <a href="#">RTC4 Compatibility Mode</a>, all back transformed values (including z values) are in the RTC6 20-bit range.</p>
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">upload_transform</a> , <a href="#">get_transform</a> , <a href="#">get_values</a>



<b>Ctrl Command</b>	<b>uart_config</b>
<b>Function</b>	Configures the internal UART interface for the specified baud rate.
<b>Call</b>	<code>RealBaudRate = uart_config( BaudRate )</code>
<b>Parameters</b>	BaudRate    Baud rate. As an unsigned 32-bit value. Allowed value range: [160 Bd...+2.8 MBd].
<b>Result</b>	<code>RealBaudRate</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>uart_config</b> extends <b>rs232_config</b> with a higher value range for the baud rate.</li> <li>• <b>uart_config</b> is synonymous with <b>rs232_config</b>, but returns the nearest possible actually used baud rate.</li> <li>• The default value is 9,600 baud.</li> <li>• The other RS-232 interface parameters cannot be altered (data bits: 8, start bits: 1, stop bits: 1, parity: none).</li> <li>• See also <a href="#">Chapter 4.6.5 "RS232 Socket Connector", page 72</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 611, OUT 611, RBF 616.
References	<a href="#">rs232_config</a> , <a href="#">rs232_write_data</a> , <a href="#">rs232_read_data</a>



<b>Ctrl Command</b>	<b>upload_transform</b>																						
<b>Function</b>	Transfers from the RTC6 board to the PC all correction and transformation settings currently assigned to the scan system.																						
<b>Call</b>	<code>UploadErrorCode = upload_transform( HeadNo, Ptr )</code>																						
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector whose settings should be queried. As an unsigned 32-bit value. Allowed values: = 1: First scan head connector. = 2: Second scan head connector.</p> <p>Ptr          Pointer (in C and C++ data type <code>ULONG_PTR</code>, an unsigned 32-bit or 64-bit value) to the PC's area of memory that should receive the queried settings.</p>																						
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <table> <tr> <td>Bit #0</td> <td>=1: X gain (gain of automatic self-calibration for galvanometer 2) = 0.</td> </tr> <tr> <td>Bit #1</td> <td>=1: Y gain (gain of automatic self-calibration for galvanometer 1) = 0.</td> </tr> <tr> <td>Bit #2</td> <td>=1: The total matrix of the defined coordinate transformation is noninvertable.</td> </tr> <tr> <td>Bit #3</td> <td>=1: No correction table assigned.</td> </tr> <tr> <td>Bit #4</td> <td>=1: The ABC values (z axis) are noninvertable.</td> </tr> <tr> <td>Bit #5</td> <td>=1: Error querying correction table.</td> </tr> <tr> <td>Bit #6</td> <td>=1: Parameter error: invalid HeadNo or <code>Ptr</code> = 0.</td> </tr> <tr> <td>Bit #7</td> <td>=1: <b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b> (<code>get_last_error</code> return code <code>RTC6_BUSY</code>).</td> </tr> <tr> <td>Bit #8</td> <td>Reserved.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #31</td> <td></td> </tr> </table>	Bit #0	=1: X gain (gain of automatic self-calibration for galvanometer 2) = 0.	Bit #1	=1: Y gain (gain of automatic self-calibration for galvanometer 1) = 0.	Bit #2	=1: The total matrix of the defined coordinate transformation is noninvertable.	Bit #3	=1: No correction table assigned.	Bit #4	=1: The ABC values (z axis) are noninvertable.	Bit #5	=1: Error querying correction table.	Bit #6	=1: Parameter error: invalid HeadNo or <code>Ptr</code> = 0.	Bit #7	=1: <b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b> ( <code>get_last_error</code> return code <code>RTC6_BUSY</code> ).	Bit #8	Reserved.	...		Bit #31	
Bit #0	=1: X gain (gain of automatic self-calibration for galvanometer 2) = 0.																						
Bit #1	=1: Y gain (gain of automatic self-calibration for galvanometer 1) = 0.																						
Bit #2	=1: The total matrix of the defined coordinate transformation is noninvertable.																						
Bit #3	=1: No correction table assigned.																						
Bit #4	=1: The ABC values (z axis) are noninvertable.																						
Bit #5	=1: Error querying correction table.																						
Bit #6	=1: Parameter error: invalid HeadNo or <code>Ptr</code> = 0.																						
Bit #7	=1: <b>BUSY</b> error, board was <b>BUSY</b> or <b>INTERNAL-BUSY</b> ( <code>get_last_error</code> return code <code>RTC6_BUSY</code> ).																						
Bit #8	Reserved.																						
...																							
Bit #31																							
<b>Comments</b>	<ul style="list-style-type: none"> <li>The queried and transferred data can be used for backward transforming actual position values by <code>transform</code> or <code>get_transform</code> (see also <a href="#">Chapter 8.1.3 "Monitoring the Positioning", page 201</a>).</li> <li>For storage of each queried data set, the user program must provide (at an address specified by <code>Ptr</code>) an area of PC main memory equal to 528,520 bytes.</li> <li>In case of error (except for Bit #6 = 1), an error indication is stored at <code>Ptr</code> to indicate that the data are erroneous. <code>transform</code> and <code>get_transform</code> recognize this error information and ensure that the backward transformation is not executed (<code>transform</code> then generates a corresponding error code, <code>get_transform</code> generates a <code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> </ul>																						



Ctrl Command	upload_transform
Comments (cont'd)	<ul style="list-style-type: none"><li>• <b>upload_transform</b> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC6_BUSY</a>), if:<ul style="list-style-type: none"><li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <a href="#">pause_list</a>)</li><li>– the <b>INTERNAL-BUSY</b> status of the board is set</li></ul></li><li>• <b>upload_transform</b> is even executed, if:<ul style="list-style-type: none"><li>– a list has been paused by <a href="#">set_wait</a> (<b>PAUSED</b> status set)</li></ul></li><li>• During the runtime of <b>upload_transform</b>, external starts are suppressed.</li></ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">transform</a> , <a href="#">get_transform</a>



<b>Ctrl Command</b>	<b>verify_checksum</b>
Function	Tests for the presence of a checksum or creates a checksum for a correction file.
Call	<code>verify_checksum( Name )</code>
Parameters	Name      Name of the correction file. As a pointer to a \0-terminated ANSI string.
Result	Result of the test. As an unsigned 32-bit value.  = 0:    No error (the tested checksum is OK.). = 1:    A checksum was newly created (no info about file integrity). = 2:    The tested checksum is incorrect. = 3:    A checksum could not be determined (file error, etc.).
Comments	<ul style="list-style-type: none"> <li>Verification of correction file downloads only works for files that contain a checksum (see <a href="#">Section "Loading of correction files", page 121</a> and <a href="#">set_verify</a>).</li> <li>The <b>verify_checksum</b> command is available even without explicit access rights to a particular RTC6 board.</li> <li><b>verify_checksum</b> is not available as a multi-board command.</li> <li>The programs <code>CorrectionFileConverter.exe</code> (version 1.04) and <code>correXion5.exe</code> (version 1.01) together with <code>RTC5Base.dll</code> (version 1.0.0.4) already automatically create checksums for the output files.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <b>verify_checksum</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">set_verify</a>



<b>Normal List Command</b>	<code>wait_for_1_axis</code>
<b>Function</b>	" <b>Fly Extension</b> " Command: Waits until the value for the specified <b>Mode</b> has been exceeded or underrun.
<b>Restriction</b>	—
<b>Call</b>	<code>wait_for_1_axis( Value, Mode, WaitMode, LaserMode )</code>
<b>Parameters</b>	<p>Value      Value to be waited for. As a signed 32-bit value.</p> <p>Mode      <b>Mode</b> from <a href="#">Table 4, page 247</a>. As an unsigned 32-bit value.</p> <p>WaitMode    &lt; 0: Undercutting. = 0: Equal. &gt; 0: Overrun. The galvanometer scanner follow the object movement. <code>WaitMode+16</code>: The galvanometer scanner stop. As a signed 32-bit value.</p> <p>LaserMode   = 0: The laser remains unchanged. &gt; 0: The laser is switched off after a <b>LaserOff</b> delay. As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Section ""Fly Extension" Commands", page 245</a>.</li> <li>Depending on the set <b>Mode</b>, it can be waited for an encoder value or (any) <b>McBSP</b> value. In case of <b>McBSP</b>, a corresponding Processing-on-the-fly correction should be enabled. Make sure that the data is transferred in the correct format.</li> <li>With <code>wait_for_1_axis</code> and <code>wait_for_2_axes</code>, <b>Mode 1...4</b> must not be specified. Instead, <b>Mode 17...20</b> is to be used. <b>Mode 17...18</b> is to be used with an scan system with <b>SCANahead control</b>, if waiting is yet to occur within <b>PreviewTime</b>. Outside of this, <b>Mode 19...20</b> can be used. <b>Mode 17...18</b> and <b>Mode 19...20</b> are identical with <b>intelliSCAN</b> systems.</li> <li><b>LaserMode</b> = 0: like before.</li> <li><b>LaserMode</b> &gt; 0: Laser is switched off after a <b>LaserOff</b> delay.</li> <li><b>WaitMode</b> is like <b>Mode</b> of <a href="#">wait_for_encoder_mode</a>.</li> <li><b>WaitMode</b> and <code>WaitMode+16</code> differentiate the galvanometer scanner movement, not <a href="#">set_fly_2d</a> and <a href="#">set_fly_x/set_fly_y</a>.</li> <li><b>wait_for_encoder</b> with automatic positions-dependent selection of direction is not supported.</li> <li>With an unallowed parameter value, <code>wait_for_1_axis</code> is replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</li> </ul>



<b>Normal List Command</b>	<b>wait_for_1_axis</b>
<b>Comments (cont'd)</b>	<ul style="list-style-type: none"> <li>The following command calls are executed in the same way:           <ul style="list-style-type: none"> <li>- <code>wait_for_1_axis( Value, EncoderNo + 19, Mode, 0 ) = wait_for_encoder_mode( Value, EncoderNo, Mode )</code> and <b>set_fly_2d</b> session</li> <li>- <code>wait_for_1_axis( Value, EncoderNo + 19, Mode+16, 0 ) = wait_for_encoder_mode( Value, EncoderNo, Mode )</code> and <b>set_fly_x/set_fly_y</b> session</li> <li>- <code>wait_for_1_axis( Value, 6, Mode, 0 ) = wait_for_mcbsp( Axis, Value, Mode )</code> and <b>set_fly_x_pos</b> session</li> <li>- <code>wait_for_1_axis( Value, Axis×4+10, Mode+16, 0 ) = wait_for_mcbsp( Axis, Value, Mode )</code> and <b>set_fly_x_pos/set_fly_y_pos</b> session</li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<b>wait_for_2_axes</b>



<b>Multiple List Command</b>	<b>wait_for_2_axes</b>
<b>Function</b>	"Fly Extension" Command: Waits until the values for both modes (see <b>Mode</b> ) are within or outside the specified range.
<b>Restriction</b>	—
<b>Call</b>	<code>wait_for_2_axes( ModeX, MinValueX, MaxValueX, ModeY, MinValueY, MaxValueY, WaitMode, LaserMode )</code>
<b>Parameters</b>	<p>ModeX      <b>Mode</b> from <a href="#">Table 4, page 247</a>. As an unsigned 32-bit value.</p> <p>MinValueX    Lower limit for the x axis <b>Mode</b> value to be waited for. As a signed 32-bit value.</p> <p>MaxValueX    Upper limit for the x axis <b>Mode</b> value to be waited for. As a signed 32-bit value.</p> <p>ModeY      <b>Mode</b> from <a href="#">Table 4, page 247</a>. As an unsigned 32-bit value.</p> <p>MinValueY    Lower limit for the y axis <b>Mode</b> value to be waited for. As a signed 32-bit value.</p> <p>MaxValueY    Upper limit for the y axis <b>Mode</b> value to be waited for. As a signed 32-bit value.</p> <p>WaitMode     <math>\geq 0</math>: Within limit values. <math>&lt; 0</math>: Outside limit values. The galvanometer scanner follow the object movement. WaitMode+16: The galvanometer scanners stop. As a signed 32-bit value.</p> <p>LaserMode    = 0: The laser remains unchanged. <math>&gt; 0</math>: The laser is switched off after a LaserOff delay. As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a> and <a href="#">Section "Fly Extension" Commands</a>, page 245.</li> <li><code>wait_for_2_axes</code> requires two list memory positions. The first part is executed as an undelayed short list command prior to the second part, which executes as a normal list command. Any pending delayed short list commands execute first.</li> <li>Depending on the set <b>Mode</b>, it can be waited for an encoder value or (any) <b>McBSP</b> value to be inside or outside the limits. In case of <b>McBSP</b>, a corresponding Processing-on-the-fly correction should be enabled. Make sure that the data is transferred in the correct format.</li> <li>With <code>wait_for_1_axis</code> and <code>wait_for_2_axes</code>, <b>Mode 1...4</b> must not be specified. Instead, <b>Mode 17...20</b> is to be used. <b>Mode 17...18</b> is to be used with an scan system with <b>SCANahead</b> control, if waiting is yet to occur within <b>PreviewTime</b>. Outside of this, <b>Mode 19...20</b> can be used. <b>Mode 17...18</b> and <b>Mode 19...20</b> are identical with <b>intelliSCAN</b> systems.</li> </ul>



<b>Multiple List Command</b>	<b>wait_for_2_axes</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>LaserMode = 0: like before.</li> <li>LaserMode &gt; 0: The laser is switched off after a LaserOff delay.</li> <li>WaitMode is like <a href="#">Mode</a> of <a href="#">wait_for_encoder_mode</a>.</li> <li>WaitMode and WaitMode+16 differentiate the galvanometer scanner movement not <a href="#">set_fly_2d</a> and <a href="#">set_fly_x/set_fly_y</a>.</li> <li>With an unallowed parameter value, <a href="#">wait_for_2_axes</a> is replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC6_PARAM_ERROR</a>).</li> <li>The following command calls are executed in the same way: <ul style="list-style-type: none"> <li>- <code>wait_for_2_axes( 19, EncXmin, EncXmax, 20, EncYmin, EncYmax, 0, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 0 ) and set_fly_2d session, intelliSCAN</code></li> <li>- <code>wait_for_2_axes( 19, EncXmin, EncXmax, 20, EncYmin, EncYmax, 16, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 0 ) and set_fly_x/set_fly_y session, intelliSCAN</code></li> <li>- <code>wait_for_2_axes( 17, EncXmin, EncXmax, 18, EncYmin, EncYmax, 0, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 2 ) and set_fly_2d session, excelliSCAN</code></li> <li>- <code>wait_for_2_axes( 17, EncXmin, EncXmax, 18, EncYmin, EncYmax, 16, 0 ) = wait_for_encoder_in_range_mode(EncXmin, EncXmax, EncYmin, EncYmax, 2 ) and set_fly_x/set_fly_y session, excelliSCAN</code></li> </ul> </li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<a href="#">wait_for_1_axis</a>



<b>Normal List Command</b>	<b>wait_for_encoder</b>
<b>Function</b>	Waits until the selected encoder counter has overstepped or understepped the specified count for the first time.
<b>Call</b>	<code>wait_for_encoder( Value, EncoderNo )</code>
<b>Parameters</b>	<p>Value      Count as a signed 32-bit value. Allowed value range: <math>[-2^{31} \dots + (2^{31}-1)]</math>.</p> <p>EncoderNo    Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values: = 0:      Encoder counter "Encoder0". = 1:      Encoder counter "Encoder1".</p>
	<ul style="list-style-type: none"> <li>• <b>wait_for_encoder</b> is synonymous with <b>wait_for_encoder_mode</b> with parameter Mode = 0 (see comments there).</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">wait_for_encoder_mode</a>



<b>Multiple List Command</b>	<b>wait_for_encoder_in_range</b>
<b>Function</b>	Waits until both encoder counters simultaneously lie within the specified range (including limits).
<b>Call</b>	<code>wait_for_encoder_in_range( EncXmin, EncXmax, EncYmin, EncYmax )</code>
<b>Parameters</b>	EncXmin      Limit values. As signed 32-bit values. Allowed value range: $[-2^{31} \dots + (2^{31}-1)]$ .
	EncXmax      Like EncXmin (analogously).
	EncYmin      Like EncXmin (analogously).
	EncYmax      Like EncXmin (analogously).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <code>wait_for_encoder_in_range</code>, see <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 284</a> and <a href="#">Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237</a>.</li> <li><code>wait_for_encoder_in_range</code> requires two list memory positions. The first part is executed as an undelayed short list command prior to the second part, which executes as a normal list command. Any pending delayed short list commands execute first.</li> <li>If <code>EncXmin &gt; EncXmax</code>, then both values are interchanged.</li> <li>If <code>EncYmin &gt; EncYmax</code>, then both values are interchanged.</li> <li>If no encoder-based Processing-on-the-fly correction is active, then <code>wait_for_encoder_in_range</code> merely creates a waiting period (without galvanometer scanner motion).</li> <li>If <code>EncXmin = EncXmax</code> (or <code>EncYmin = EncYmax</code>), then waiting until a specific encoder value is possible.</li> <li><code>wait_for_encoder_in_range</code> is available even if the <a href="#">Option Processing-on-the-fly</a> is not enabled.</li> <li><code>wait_for_encoder_in_range</code> does not alter the laser control signals. If you want the laser off during the wait, then this command must be preceded by some other command that switches off the signals for "laser active" operation (for example, a <code>list_nop</code>).</li> <li>The active Processing-on-the-fly mode determines whether the galvanometer scanners remain stationary during the wait or move in accordance with encoder changes, see <a href="#">Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237</a>: They move with <code>set_fly_2d</code>, but otherwise remain stationary.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">wait_for_encoder_mode</a> , <a href="#">park_position</a> , <a href="#">park_return</a>



<b>Multiple List Command</b>	<b>wait_for_encoder_in_range_mode</b>
<b>Function</b>	Waits until both encoder counters simultaneously lie within the specified range (including limits).
<b>Call</b>	<code>wait_for_encoder_in_range_mode( EncXmin, EncXmax, EncYmin, EncYmax, Mode )</code>
<b>Parameters</b>	<p>EncXmin      Limit values. As signed 32-bit values. Allowed value range: <math>[-2^{29} \dots + (2^{29}-1)]</math>.</p> <p>EncXmax      Like EncXmin (analogously).</p> <p>EncYmin      Like EncXmin (analogously).</p> <p>EncYmax      Like EncXmin (analogously).</p> <p>Mode          Mode. As a signed 32-bit value. = 0 or 1:    Waits for direct encoder values ("classical behavior"). = 2:           Waits for encoder values that are expected to be present in <b>(set_scanahead_params parameter)</b> PreviewTime (SCANAhead system behavior).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage, see <a href="#">Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications"</a>, page 237 and <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals"</a>, page 284 .</li> <li>See also comments on <a href="#">wait_for_encoder_in_range</a>.</li> <li>Use Mode like in <a href="#">wait_for_encoder_mode</a>: <ul style="list-style-type: none"> <li>– 0, 1 for intelliSCAN or SCANAhead systems outside an marking process</li> <li>– 2 for SCANAhead systems during an marking process</li> </ul> </li> <li><a href="#">wait_for_encoder_in_range</a> is synonymous with <code>wait_for_encoder_in_range_mode( Mode = ,0 )</code>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 612, OUT 612, RBF 617.
References	<a href="#">wait_for_encoder_mode</a>



<b>Multiple List Command</b>	<b>wait_for_encoder_mode</b>						
<b>Function</b>	Waits until the selected encoder counter has overstepped or understepped the specified count for the first time.						
<b>Call</b>	<code>wait_for_encoder_mode( Value, EncoderNo, Mode )</code>						
<b>Parameters</b>	<table> <tr> <td>Value</td> <td>Count as a signed 32-bit value. Allowed value range: <math>[-2^{29} \dots +(2^{29}-1)]</math>.</td> </tr> <tr> <td>EncoderNo</td> <td>Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values: = 0: Encoder counter "Encoder0". = 1: Encoder counter "Encoder1".</td> </tr> <tr> <td>Mode</td> <td>Mode. As a signed 32-bit value. = 0: Waits for understepping/overstepping (position dependent). &gt; 0: Waits for overstepping (position independent). &lt; 0: Waits for understepping (position independent).</td> </tr> </table>	Value	Count as a signed 32-bit value. Allowed value range: $[-2^{29} \dots +(2^{29}-1)]$ .	EncoderNo	Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values: = 0: Encoder counter "Encoder0". = 1: Encoder counter "Encoder1".	Mode	Mode. As a signed 32-bit value. = 0: Waits for understepping/overstepping (position dependent). > 0: Waits for overstepping (position independent). < 0: Waits for understepping (position independent).
Value	Count as a signed 32-bit value. Allowed value range: $[-2^{29} \dots +(2^{29}-1)]$ .						
EncoderNo	Number of the to-be-used encoder counter. As an unsigned 32-bit value. Allowed values: = 0: Encoder counter "Encoder0". = 1: Encoder counter "Encoder1".						
Mode	Mode. As a signed 32-bit value. = 0: Waits for understepping/overstepping (position dependent). > 0: Waits for overstepping (position independent). < 0: Waits for understepping (position independent).						
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <code>wait_for_encoder_mode</code>, see <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 284</a>.</li> <li>If <code>Mode = 0</code>, ensure that the size and sign of the parameter <code>Value</code> is appropriate for the counting direction of the selected encoder (for external triggering, this corresponds to the workpiece's direction of motion). If <code>Value &gt; 0</code>, the command waits for overstepping, otherwise for understepping. If <code>Value</code> is positive and already less than the current encoder count, then <code>wait_for_encoder_mode</code> waits for a complete traversal of the counter (likewise if <code>Value</code> is negative and larger than the current encoder count). At a 1 MHz counter rate, this can take up to approx. 36 minutes!</li> <li>If <code>Mode ≠ 0</code>, then <code>wait_for_encoder_mode</code> waits for overstepping/understepping of the <code>Value</code> parameter independently of the current position and direction of motion.</li> <li>If <code>EncoderNo &gt; 1</code>, then <code>wait_for_encoder_mode</code> is replaced with a <a href="#"><code>list_nop</code></a> (<a href="#"><code>get_last_error</code></a> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>If no encoder-based Processing-on-the-fly correction is active, then <code>wait_for_encoder_mode</code> merely creates a waiting period (without galvanometer scanner motion) that allows implementation of an externally triggered synchronization (as an alternative to external starts, <code>set_wait</code> or <code>if_cond</code>, etc.).</li> <li><code>wait_for_encoder_mode</code> is available even if the <a href="#">Option Processing-on-the-fly</a> is not enabled.</li> <li>For <code>Mode = 0</code>, <code>wait_for_encoder_mode</code> is synonymous with <code>wait_for_encoder</code>.</li> <li><code>wait_for_encoder_mode</code> does not alter the laser control signals. If you want the laser off during the wait, then <code>wait_for_encoder_mode</code> must be preceded by some other command that switches off the "laser active" laser control signals (for example, a <a href="#"><code>list_nop</code></a>).</li> </ul>						



<b>Multiple List Command</b>	<b>wait_for_encoder_mode</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The active Processing-on-the-fly mode determines whether the galvanometer scanners remain stationary during the wait or move in accordance with encoder changes, see <a href="#">Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications", page 237</a>: They move with <a href="#">set_fly_2d</a>, but otherwise remain stationary.</li> <li>Mode = 0 or <math>\pm 1</math> waits for direct encoder values: "classic" behavior.</li> <li>Mode = <math>\pm 2</math> waits for encoder values that are expected to be present in <a href="#">(set_scanahead_params parameter)</a> <a href="#">PreviewTime: SCANAhead system behavior</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">get_encoder</a> , <a href="#">store_encoder</a> , <a href="#">read_encoder</a> , <a href="#">simulate_encoder</a> , <a href="#">wait_for_encoder</a> , <a href="#">wait_for_encoder_in_range</a> , <a href="#">park_position</a> , <a href="#">park_return</a>



<b>Normal List Command</b>	<b>wait_for_mcbsp</b>						
<b>Function</b>	Waits until the input value at the <b>McBSP interface</b> has reached, overstepped or understepped the specified value for the first time.						
<b>Call</b>	<code>wait_for_mcbsp( Axis, Value, Mode )</code>						
<b>Parameters</b>	<table> <tr> <td>Axis</td> <td>Selects which half-word of the input value is used for evaluation (see below). As an unsigned 32-bit value. Allowed values: = 0: lower half-word (x axis, galvanometer scanner 2) = 1: upper half-word (y axis, galvanometer scanner 1)</td> </tr> <tr> <td>Value</td> <td>Threshold value as a signed 32-bit value</td> </tr> <tr> <td>Mode</td> <td>As a signed 32-bit value. = 0: Waits for equality. &gt; 0: Waits for overstepping. &lt; 0: Waits for understepping.</td> </tr> </table>	Axis	Selects which half-word of the input value is used for evaluation (see below). As an unsigned 32-bit value. Allowed values: = 0: lower half-word (x axis, galvanometer scanner 2) = 1: upper half-word (y axis, galvanometer scanner 1)	Value	Threshold value as a signed 32-bit value	Mode	As a signed 32-bit value. = 0: Waits for equality. > 0: Waits for overstepping. < 0: Waits for understepping.
Axis	Selects which half-word of the input value is used for evaluation (see below). As an unsigned 32-bit value. Allowed values: = 0: lower half-word (x axis, galvanometer scanner 2) = 1: upper half-word (y axis, galvanometer scanner 1)						
Value	Threshold value as a signed 32-bit value						
Mode	As a signed 32-bit value. = 0: Waits for equality. > 0: Waits for overstepping. < 0: Waits for understepping.						
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <b>wait_for_mcbsp</b>, see <a href="#">Chapter 9.3.4 "Synchronization and Online Positioning by McBSP Signals", page 286</a>.</li> <li><b>wait_for_mcbsp</b> is comparable to <b>wait_for_encoder_mode</b>, but the <b>McBSP interface</b> is queried.</li> <li>If <b>set_fly_x_pos</b> and <b>set_fly_y_pos</b> are <i>simultaneously</i> activated, then <b>Value</b> consists of two 16-bit half-words, see <a href="#">Section "Correction via McBSP Interface", page 230</a>. Only in this case does <b>Axis</b> control which half-word is used for evaluation. In all other cases, <b>Axis</b> is irrelevant and <b>Value</b> is interpreted as a signed 32-bit value.</li> <li><b>Axis</b> must be either 0 or 1 (even if it is irrelevant). Otherwise, <b>wait_for_mcbsp</b> is replaced by <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b>).</li> <li>If neither <b>set_fly_x_pos</b>, <b>set_fly_y_pos</b> nor <b>set_fly_rot_pos</b> are activated, then <b>wait_for_mcbsp</b> merely creates a waiting period (without galvanometer scanner motion) that allows implementation of an externally triggered synchronization (as an alternative to external starts, <b>set_wait</b> or <b>if_cond</b>, etc.).</li> <li><b>wait_for_mcbsp</b> is available even if the <b>Option Processing-on-the-fly</b> is not enabled.</li> <li><b>wait_for_mcbsp</b> does not alter the laser control signals. If you want the laser off during the wait, then <b>wait_for_mcbsp</b> must be preceded by some other command that switches off the signals for "laser active" operation (for example, a <b>list_nop</b>).</li> </ul>						
RTC4→RTC6	New command.						
RTC5→RTC6	Unchanged functionality.						
Version info	Available as of version DLL 600, OUT 600, RBF 600.						
References	<a href="#">wait_for_encoder_mode</a>						



<b>Normal List Command</b>	<b>wait_for_timestamp_counter</b>
Function	Waits for a "time stamp counter" value.
Call	<code>wait_for_timestamp_counter( TimeStampCounter )</code>
Parameters	<p>TimeStamp    "Time stamp counter" offset to wait for.          Counter      As an unsigned 32-bit value.</p>
Comments	<ul style="list-style-type: none"> <li>• <b>wait_for_timestamp_counter</b> waits until the current "time stamp counter" has reached the value TimeStampStorage (from <b>store_timestamp_counter</b> / <b>store_timestamp_counter_list</b>) + TimeStampCounter.</li> <li>• Delayed short list commands are executed first.</li> <li>• This allows absolute time references with an accuracy of 10 µs from the TimeStampStorage point in time to be established when loading a list.</li> <li>• If the desired time has already passed when <b>wait_for_timestamp_counter</b> is reached, a full 32-bit counter cycle is waited for (duration approx. 12 hours). This can be cancelled by <b>stop_execution</b> or /STOP.</li> <li>• Synchronization with other commands, such as <b>wait_for_encoder</b> or conditional commands, depends on external hardware.</li> <li>• See <a href="#">Chapter 8.12 "Time Measurements", page 267</a>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 617, OUT 617, RBF 623.
References	<b>store_timestamp_counter</b> , <b>store_timestamp_counter_list</b>



<b>Ctrl Command</b>	<b>write_8bit_port</b>
<b>Function</b>	Writes a value to the 8-bit digital output port on the EXTENSION 2 socket connector.
<b>Call</b>	<code>write_8bit_port( Value )</code>
<b>Parameters</b>	Value      8-bit output value (DATA0...DATA7). As an unsigned 32-bit value. Only the least significant 8 bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 9.1.2 "8-Bit Digital Output Port", page 269</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">write_8bit_port_list</a>

<b>Delayed Short List Command</b>	<b>write_8bit_port_list</b>
<b>Function</b>	Like <a href="#">write_8bit_port</a> , but a list command.
<b>Call</b>	<code>write_8bit_port_list( Value )</code>
<b>Parameters</b>	Value      8-bit output value (DATA0...DATA7). As an unsigned 32-bit value. Only the least significant 8 bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">write_8bit_port</a>.</li> <li>• As of version DLL 602, OUT 602: When the 8-Bit digital output (for example, for laser power) is to be executed synchronous to the laser switch times, use <a href="#">set_laser_power( 2, Value )</a>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">write_8bit_port, set_laser_power</a>



<b>Ctrl Command</b>	<b>write_abc_to_file</b>																				
<b>Function</b>	Writes the ABC values directly into a specified correction file on the PC.																				
<b>Call</b>	<code>ErrorNo = write_abc_to_file( Name, A, B, C )</code>																				
<b>Parameters</b>	<table> <tr> <td>Name</td><td>Name of the correction file. As a pointer to a \0-terminated ANSI string.</td></tr> <tr> <td>A</td><td>Coefficient A of the parabolic function <math>z_{out} = A + B/I + C/I^2</math> which is used for calculating the Z output values. As a 64-bit IEEE floating point value. Allowed range: see <a href="#">load_z_table</a>.</td></tr> <tr> <td>B</td><td>Like A (analogously).</td></tr> <tr> <td>C</td><td>Like A (analogously).</td></tr> </table>	Name	Name of the correction file. As a pointer to a \0-terminated ANSI string.	A	Coefficient A of the parabolic function $z_{out} = A + B/I + C/I^2$ which is used for calculating the Z output values. As a 64-bit IEEE floating point value. Allowed range: see <a href="#">load_z_table</a> .	B	Like A (analogously).	C	Like A (analogously).												
Name	Name of the correction file. As a pointer to a \0-terminated ANSI string.																				
A	Coefficient A of the parabolic function $z_{out} = A + B/I + C/I^2$ which is used for calculating the Z output values. As a 64-bit IEEE floating point value. Allowed range: see <a href="#">load_z_table</a> .																				
B	Like A (analogously).																				
C	Like A (analogously).																				
<b>Result</b>	<table> <tr> <td>ErrorNo</td><td>Error code. As an unsigned 32-bit value.</td></tr> <tr> <td>0</td><td>No error.</td></tr> <tr> <td>1</td><td>A exceeded the maximum allowed value.</td></tr> <tr> <td>2</td><td>A undercut the minimum allowed value.</td></tr> <tr> <td>4</td><td>B exceeded the maximum allowed value.</td></tr> <tr> <td>8</td><td>B undercut the minimum allowed value.</td></tr> <tr> <td>16</td><td>C exceeded the maximum allowed value.</td></tr> <tr> <td>32</td><td>C undercut the minimum allowed value.</td></tr> <tr> <td>3</td><td>File-open error (empty string, file not found etc.).</td></tr> <tr> <td>12</td><td>File error (checksum could not be determined, file corrupt).</td></tr> </table>	ErrorNo	Error code. As an unsigned 32-bit value.	0	No error.	1	A exceeded the maximum allowed value.	2	A undercut the minimum allowed value.	4	B exceeded the maximum allowed value.	8	B undercut the minimum allowed value.	16	C exceeded the maximum allowed value.	32	C undercut the minimum allowed value.	3	File-open error (empty string, file not found etc.).	12	File error (checksum could not be determined, file corrupt).
ErrorNo	Error code. As an unsigned 32-bit value.																				
0	No error.																				
1	A exceeded the maximum allowed value.																				
2	A undercut the minimum allowed value.																				
4	B exceeded the maximum allowed value.																				
8	B undercut the minimum allowed value.																				
16	C exceeded the maximum allowed value.																				
32	C undercut the minimum allowed value.																				
3	File-open error (empty string, file not found etc.).																				
12	File error (checksum could not be determined, file corrupt).																				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>write_abc_to_file</code> is available even without explicit access rights to a specific RTC6 board.</li> <li>• <code>write_abc_to_file</code> is not available as a multi-board command.</li> <li>• The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 120</a>) are neither generated nor altered by <code>write_abc_to_file</code>.</li> <li>• If the error code is not 0 or 12, the ABC values are not outputted.</li> </ul>																				
RTC4→RTC6	New command.																				
RTC5→RTC6	Unchanged functionality.																				
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.																				
<b>References</b>	<a href="#">read_abc_from_file</a>																				



<b>Ctrl Command</b>	<b>write_da_1</b>
<b>Function</b>	See <a href="#">write_da_x</a> .
<b>Call</b>	<code>write_da_1( Value )</code>
<b>Parameters</b>	<p>Value      12-bit output value for the ANALOG OUT1 analog output port.      As an unsigned 32-bit value.      Higher bits are ignored.      Value = 0 corresponds to an output value of 0 V.      Value = <math>2^{12}-1</math> corresponds to an output value of 10 V.</p>
RTC4→RTC6	Unchanged functionality.  In RTC4 Compatibility Mode: as <a href="#">write_da_x</a> .
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">write_da_x</a>

<b>Delayed Short List Command</b>	<b>write_da_1_list</b>
<b>Function</b>	See <a href="#">write_da_x_list</a> .
<b>Call</b>	<code>write_da_1_list( Value )</code>
<b>Parameters</b>	<p>Value      12-bit output value for the ANALOG OUT1 analog output port.      As an unsigned 32-bit value.      Higher bits are ignored.      Value = 0 corresponds to an output value of 0 V.      Value = <math>2^{12}-1</math> corresponds to an output value of 10 V.</p>
RTC4→RTC6	Unchanged functionality.  In RTC4 Compatibility Mode: as <a href="#">write_da_x</a> .
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">write_da_x_list</a>



<b>Ctrl Command</b>	<b>write_da_2</b>
<b>Function</b>	See <a href="#">write_da_x</a> .
<b>Call</b>	<code>write_da_2( Value )</code>
<b>Parameters</b>	<p>Value      12-bit output value for the ANALOG OUT2 analog output port.      As an unsigned 32-bit value.      Higher bits are ignored.      Value = 0 corresponds to an output value of 0 V.      Value = <math>2^{12}-1</math> corresponds to an output value of 10 V.</p>
RTC4→RTC6	Unchanged functionality.  In RTC4 Compatibility Mode: as <a href="#">write_da_x</a> .
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">write_da_x</a>

<b>Delayed Short List Command</b>	<b>write_da_2_list</b>
<b>Function</b>	See <a href="#">write_da_x_list</a> .
<b>Call</b>	<code>write_da_2_list( Value )</code>
<b>Parameters</b>	<p>Value      12-bit output value for the ANALOG OUT2 analog output port.      As an unsigned 32-bit value.      Higher bits are ignored.      Value = 0 corresponds to an output value of 0 V.      Value = <math>2^{12}-1</math> corresponds to an output value of 10 V.</p>
RTC4→RTC6	Unchanged functionality.  In RTC4 Compatibility Mode: as <a href="#">write_da_x</a> .
RTC5→RTC6	Unchanged functionality.
<b>Version info</b>	Available as of version DLL 600, OUT 600, RBF 600.
<b>References</b>	<a href="#">write_da_x_list</a>



<b>Ctrl Command</b>	<b>write_da_x</b>
<b>Function</b>	Writes an output value to one of the analog output ports of the RTC6.
<b>Call</b>	<code>write_da_x( x, Value )</code>
<b>Parameters</b>	<p>x      Number of the analog output port. As an unsigned 32-bit value. Allowed values: [1, 2] (1: ANALOG OUT1, 2: ANALOG OUT2).</p> <p>Value    12-bit output value for the selected analog output port. As an unsigned 32-bit value. Higher bits are ignored. Value = 0 corresponds to an output value of 0 V. Value = <math>2^{12}-1</math> corresponds to an output value of 10 V.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 9.1.4 "12-Bit Analog Output Ports", page 269</a>.</li> <li>The output range of the analog output ports is 0 V...10 V.</li> <li>For <math>x &lt; 1</math> or <math>x &gt; 2</math>, <code>write_da_x</code> is ignored (<code>get_last_error</code> return code <code>RTC6_PARAM_ERROR</code>).</li> <li><code>write_da_1 / write_da_2</code> can be used alternatively to <code>write_da_x</code> (without parameter x).</li> </ul>
RTC4→RTC6	Unchanged functionality. In <a href="#">RTC4 Compatibility Mode</a> , the RTC6 multiplies the specified <code>Value</code> by 4.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">write_da_x_list</a>



<b>Delayed Short List Command</b>	<b>write_da_x_list</b>
Function	Like <a href="#">write_da_x</a> , but a list command.
Call	<code>write_da_x_list( x, Value )</code>
Parameters	<p>x      Number of the analog output port.            As an unsigned 32-bit value.            Allowed values: [1, 2] (1: ANALOG OUT1, 2: ANALOG OUT2).</p> <p>Value    12-bit output value for the selected analog output port.            As an unsigned 32-bit value.            Higher bits are ignored.            Value = 0 corresponds to an output value of 0 V.            Value = <math>2^{12}-1</math> corresponds to an output value of 10 V.</p>
	<ul style="list-style-type: none"> <li>For <math>x &lt; 1</math> or <math>x &gt; 2</math>, the command is replaced with a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <code>RTC6_PARAM_ERROR</code>).</li> <li>As of version DLL 602, OUT 602: When the ANALOG OUT output port (for example, for the laser power) is to be executed synchronous to the laser switch times, use <a href="#">set_laser_power</a>( <math>x-1</math>, Value ).</li> </ul>
RTC4→RTC6	Unchanged functionality.  In <a href="#">RTC4 Compatibility Mode</a> : as <a href="#">write_da_x</a> .
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<a href="#">write_da_x</a> , <a href="#">set_laser_power</a>



<b>Ctrl Command</b>	<b>write_hi_pos</b>																
<b>Function</b>	Writes the specified values to the flash memory of the RTC6 board to be used as ASC reference values (= Home-In reference positions, not to confuse with Home-In positions).																
<b>Call</b>	Result = write_hi_pos ( HeadNo, X1, X2, Y1, Y2 )																
<b>Parameters</b>	<p>HeadNo      Number of the scan head connector. As an unsigned 32-bit value.            Allowed values:            = 1: First scan head connector.            = 2: Second scan head connector.</p> <p>X1      X1 reference position. In bits.            As a signed 32-bit value.</p> <p>X2      Like X1 (analogously).</p> <p>Y1      Like X1 (analogously).</p> <p>Y2      Like X1 (analogously).</p>																
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <table> <tr> <td>0</td> <td>Kein Fehler.</td> </tr> <tr> <td>Bit #0</td> <td>=1: Wrong HeadNo.</td> </tr> <tr> <td>Bit #1</td> <td>=1: Wrong sensor position for X1.</td> </tr> <tr> <td>Bit #2</td> <td>=1: Wrong sensor position for X2.</td> </tr> <tr> <td>Bit #3</td> <td>=1: Wrong sensor position for Y1.</td> </tr> <tr> <td>Bit #4</td> <td>=1: Wrong sensor position for Y2.</td> </tr> <tr> <td>Bit #5</td> <td>=1: Invalid ASC version.</td> </tr> <tr> <td>Bit #6</td> <td>=1: Download failed.            The values have possibly not been saved.</td> </tr> </table>	0	Kein Fehler.	Bit #0	=1: Wrong HeadNo.	Bit #1	=1: Wrong sensor position for X1.	Bit #2	=1: Wrong sensor position for X2.	Bit #3	=1: Wrong sensor position for Y1.	Bit #4	=1: Wrong sensor position for Y2.	Bit #5	=1: Invalid ASC version.	Bit #6	=1: Download failed. The values have possibly not been saved.
0	Kein Fehler.																
Bit #0	=1: Wrong HeadNo.																
Bit #1	=1: Wrong sensor position for X1.																
Bit #2	=1: Wrong sensor position for X2.																
Bit #3	=1: Wrong sensor position for Y1.																
Bit #4	=1: Wrong sensor position for Y2.																
Bit #5	=1: Invalid ASC version.																
Bit #6	=1: Download failed. The values have possibly not been saved.																
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>write_hi_pos</b> is used in cases when a scan head is moved from RTC6 board "A" to RTC6 board "B" in order to transfer its Home-In reference positions from RTC6 board "A" to RTC6 board "B".</li> <li>• Use <b>get_hi_pos( HeadNo )</b> to read out the Home-In reference positions of RTC6 board "A" (only after <b>init_RTC6_dll</b>, see the <b>get_hi_pos</b> command description).</li> <li>• <b>write_hi_pos</b> is also executed if the scan head is switched off or even a scan head is not attached.</li> <li>• With an ASC type-1 equipped scan head attached, <b>auto_cal(Command = 4)</b> (or <b>auto_cal(Command = 0)</b> because this command implicitly executes <b>auto_cal(Command = 4)</b>) must have been successfully executed at last on the RTC6 board "B".            A cross-check with <b>get_auto_cal(HeadNo)</b> needs to return 100. Otherwise, <b>write_hi_pos</b> would return error Bit #5 = 1.            If required, connect the scan head to RTC6 board "B" and execute <b>auto_cal(HeadNo, 4)</b> before executing <b>write_hi_pos</b>.</li> <li>• <b>write_hi_pos</b> takes several hundred <math>\mu</math>s. During this time, the 10 <math>\mu</math>s clock of RTC6 boards is interrupted.</li> </ul>																
RTC4→RTC6	New command.																
RTC5→RTC6	Unchanged functionality.																
Version info	Available as of version DLL 600, OUT 600, RBF 600.																
References	<b>get_hi_pos, get_auto_cal, auto_cal</b>																



<b>Ctrl Command</b>	<b>write_image_eth</b>
<b>Function</b>	<b>Standalone Functionality:</b> Reads out data for automatic booting from a binary file on the PC and saves it to the <b>NAND memory</b> .
<b>Prerequisite</b>	Minimum requirement: RTC6 Software Package V1.7.0 and RTC6BIOSETH_26.
<b>Call</b>	Result = write_image_eth( Name )
<b>Parameters</b>	Name      Name of the binary file. As a pointer to a \0-terminated ANSI string.
<b>Result</b>	Result      Error code. Like <b>Result</b> of <b>read_image_eth</b> . As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>write_image_eth</b> is not executed (<b>get_last_error</b> return code <b>RTC6_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> status of the board is set (list is being processed or has been halted by <b>pause_list</b>)</li> <li>– the <b>INTERNAL-BUSY</b> status of the board status is set</li> </ul> </li> <li>• If the <b>Name</b> cannot be opened, a <b>get_last_error</b> return code <b>RTC6_PARAM_ERROR</b> is generated.</li> <li>• During the execution of <b>read_image_eth</b> the 10 µs clock period of the <b>DSP</b> is interrupted for up to 2 minutes.</li> <li>• <b>write_image_eth</b> is only allowed with RTC6 Ethernet Boards. Otherwise, a <b>get_last_error</b> return code <b>RTC6_TYPE_REJECTED</b> is generated.</li> <li>• See <b>Chapter 15.7 "Standalone Functionality", page 859</b>.</li> </ul>
RTC4→RTC6	New command.
RTC5→RTC6	New command.
Version info	Available as of version DLL 618, OUT 618, RBF 623.
References	<b>read_image_eth, store_program</b>



<b>Ctrl Command</b>	<b>write_io_port</b>
<b>Function</b>	Writes a value to the 16-bit digital output port on the EXTENSION 1 socket connector.
<b>Call</b>	<code>write_io_port( Value )</code>
<b>Parameters</b>	Value      16-bit output value (DIGITAL OUT0...DIGITAL OUT15). As an unsigned 32-bit value. Only the least significant 16 bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>Use <code>set_io_cond_list</code> and <code>clear_io_cond_list</code> to set or clear individual bits of the 16-bit digital output port, depending on the state of the <i>input</i> port.</li> <li><code>write_io_port_mask</code> and <code>write_io_port_mask_list</code> also allow changing selectable bits of the 16-bit digital output port.</li> <li>See also Chapter 9.1.1 "16-Bit Digital Output Port", page 268.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>write_io_port_list</code> , <code>write_io_port_mask</code> , <code>write_io_port_mask_list</code> , <code>set_io_cond_list</code> , <code>clear_io_cond_list</code> , <code>get_io_status</code>

<b>Delayed Short List Command</b>	<b>write_io_port_list</b>
<b>Function</b>	Like <code>write_io_port</code> , but a list command.
<b>Call</b>	<code>write_io_port_list( Value )</code>
<b>Parameters</b>	Value      16-bit output value (DIGITAL OUT0...DIGITAL OUT15). As an unsigned 32-bit value. Only the least significant 16 bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <code>write_io_port</code>.</li> <li>As of version DLL 602, OUT 602: When the 16-Bit digital output (for example, for laser power) is to be executed synchronous to the laser switch times, use <code>set_laser_power( 3, Value )</code>.</li> </ul>
RTC4→RTC6	Unchanged functionality.
RTC5→RTC6	Unchanged functionality.
Version info	Available as of version DLL 600, OUT 600, RBF 600.
References	<code>write_io_port</code> , <code>set_laser_power</code>



<b>Ctrl Command</b>	<b>write_io_port_mask</b>	
<b>Function</b>	Writes those bits of <code>Value</code> specified by the <code>Mask</code> parameter to the 16-bit digital output port on the EXTENSION 1 socket connector.	
<b>Call</b>	<code>write_io_port_mask( Value, Mask )</code>	
<b>Parameters</b>	<code>Value</code>	16-bit output value (DIGITAL OUT0...DIGITAL OUT15). As an unsigned 32-bit value. Only the least significant 16 bits are evaluated.
	<code>Mask</code>	16-bit mask (for DIGITAL OUT0...DIGITAL OUT15). As an unsigned 32-bit value. Only the least significant 16 bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>The <code>Mask</code> parameter determines <i>which</i> bits of the 16-bit digital output port are to be altered, the <code>Value</code> parameter determines <i>how</i> they are altered. All bits of the 16-bit digital output port that were not set in <code>Mask</code> remain unaltered, that is, they are outputted again as previously.</li> <li>For <code>Mask</code> = <code>0xFFFF</code>, <code>write_io_port_mask</code> behaves like <code>write_io_port</code>.</li> <li>See also <a href="#">Chapter 9.1.1 "16-Bit Digital Output Port", page 268</a>.</li> </ul>	
RTC4→RTC6	New command.	
RTC5→RTC6	Unchanged functionality.	
Version info	Available as of version DLL 600, OUT 600, RBF 600.	
References	<a href="#">write_io_port</a> , <a href="#">write_io_port_list</a>	

<b>Delayed Short List Command</b>	<b>write_io_port_mask_list</b>	
<b>Function</b>	Like <code>write_io_port_mask</code> , but a list command.	
<b>Call</b>	<code>write_io_port_mask_list( Value, Mask )</code>	
<b>Parameters</b>	<code>Value</code>	16-bit output value (DIGITAL OUT0...DIGITAL OUT15). As an unsigned 32-bit value. Only the least significant 16 bits are evaluated.
	<code>Mask</code>	16-bit mask (for DIGITAL OUT0...DIGITAL OUT15). As an unsigned 32-bit value. Only the least significant 16 bits are evaluated.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">write_io_port_mask</a>.</li> <li><code>write_io_port_mask_list</code> cannot be used for laser power control, see <a href="#">write_io_port_list</a>.</li> </ul>	
RTC4→RTC6	New command.	
RTC5→RTC6	Unchanged functionality.	
Version info	Available as of version DLL 600, OUT 600, RBF 600.	
References	<a href="#">write_io_port_mask</a> , <a href="#">write_io_port_list</a> , <a href="#">set_laser_power</a>	



## 10.3 Unsupported RTC4/RTC5 Commands

Some RTC4/RTC5 commands are not supported by the RTC6. Most of them can be replaced by RTC6 commands, see following tables.

<b>Ctrl Command</b>	<b>dsp_start</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	This command is not needed for normal operation. The <b>DSP</b> starts automatically after the program file is loaded by <b>load_program_file</b> .

<b>Ctrl Command</b>	<b>free_RTC5_dll</b>
Support status	This RTC5 command is not supported by the RTC6.
Replaced by	<b>free_RTC6_dll</b>

<b>Ctrl Command</b>	<b>get_xy_pos</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	Replaced by <b>get_value, get_values</b>

<b>Ctrl Command</b>	<b>get_xyz_pos</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	Replaced by <b>get_value, get_values</b>

<b>Ctrl Command</b>	<b>init_RTC5_dll</b>
Support status	This RTC5 command is not supported by the RTC6.
Replaced by	<b>init_RTC6_dll</b>

<b>Ctrl Command</b>	<b>read_pixel_ad</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	There is no equivalent RTC6 command.

<b>Ctrl Command</b>	<b>rtc4_count_cards</b>
Support status	This RTC4 command is not supported by the RTC6.
Replaced by	<b>rtc6_count_cards</b>

<b>Ctrl Command</b>	<b>rtc5_count_cards</b>
Support status	This RTC5 command is not supported by the RTC6.
Replaced by	<b>rtc6_count_cards</b>



<b>Ctrl Command</b>	<b>select_list</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	<b>select_list(0)</b> can be replaced by <b>set_extstartpos(0)</b> , <b>select_list(1)</b> by <b>set_extstartpos(Mem1)</b> . Thereby, Mem1 is the memory size of "List 1" (and the absolute start address of "List 2"). After initialization (with <b>load_program_file</b> ), Mem1 is 4000, otherwise as set by <b>config_list</b> . Mem1 can be determined (for example, after a board changed "ownership") by <b>set_start_list_2</b> and <b>get_input_pointer</b> .

<b>Ctrl Command</b>	<b>set_list_mode</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	See <a href="#">Chapter 6.5.4 "RTC4-Circular Queue Mode", page 111</a> .

<b>Ctrl Command</b>	<b>set_piso_control</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	<p>This command is not needed for operation of the RTC6.</p> <p>For data transfer in accordance with the SL2-100 protocol, the bi-directional communication between the scan system and the RTC5/RTC6 does not depend on the data cable length.</p> <p>For data transfer in accordance with the XY2-100 protocol, the timing of the communication must be further on adjusted to reflect the length of the data cable; but the adjustment is now realized by a jumper at the XY2-100 converter, see <a href="#">Chapter 4.5.2 "XY2-100 Converter (Accessory)", page 58</a>.</p>

<b>Ctrl Command</b>	<b>set_wobbel_xy</b>
Support status	This RTC4 command is not supported by the RTC6.
Replaced by	<b>set_wobbel</b> , <b>set_wobbel_mode</b>

<b>Ctrl Command</b>	<b>z_out</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	For setting the Z value in a 3-axis scan system, the command <b>set_defocus</b> can be used (only together with an RTC6 with enabled <a href="#">Option "3D"</a> ). After <b>load_z_table(0.0, 1.0, 0.0)</b> , <b>set_defocus(z)</b> has the same effect as <b>z_out(z)</b> (see also <b>set_offset_xyz</b> ).

<b>List Command</b>	<b>z_out_list</b>
Support status	This RTC4 command is not supported by the RTC6.
RTC4→RTC6	After <b>load_z_table(0.0, 1.0, 0.0)</b> , <b>set_defocus_list(z)</b> has the same effect as <b>z_out_list(z)</b> (see also <b>set_offset_xyz_list</b> ).



## 11 Demo Programs

- Currently the RTC6 Software Package does not contain demo programs.

## 12 Troubleshooting

Problem	Remedy
<b>PC does not boot</b>	<p>Switch off the PC and check the following:</p> <ul style="list-style-type: none"> <li>Check if the RTC6 board is correctly seated in the PCIe slot. Refer to the instructions in your PC manual.</li> <li>Check for metal parts that may have fallen into the PC housing during installation.</li> <li>Check for loose cables or connectors.</li> </ul>
<b>RTC6 board does not respond</b>	<ul style="list-style-type: none"> <li>Check the driver installation. See <a href="#">Chapter 5.4 "Installing the RTC6 Software", page 80</a>.</li> <li>Use the included HPGL converter program to check if the board can be properly accessed. If not: Check the cable type and the cable length. If the scan system is controlled by an XY2-100 converter, then check, whether the solder jumpers of the converter are set appropriate for the cable length. See <a href="#">figure 11</a>. If yes: Check the DLL import declarations in your user program. See <a href="#">Chapter 6.2.2 "Importing Commands", page 85</a>.</li> </ul>
<b>User program fails</b>	<ul style="list-style-type: none"> <li>Check the driver installation. See <a href="#">Chapter 5.4 "Installing the RTC6 Software", page 80</a>.</li> <li>Check the RTC6 board initialization in the user program.</li> <li>Check the DLL import declarations in your user program. See <a href="#">Chapter 6.2.2 "Importing Commands", page 85</a>.</li> </ul>
<b>Scan head control fails</b>	<ul style="list-style-type: none"> <li>Check if the scan head is properly connected to the RTC6 board by the data cable. Make sure to follow the specifications for the data cable. See <a href="#">Chapter 4.5.3 "Data Cables (Accessories)", page 60</a>.</li> <li>Check the power supply of the scan head. Refer to your scan head operating manual.</li> <li>Check your user program.</li> </ul>
<b>Laser control fails</b>	<ul style="list-style-type: none"> <li>Check the interface between the RTC6 board and the laser.</li> </ul>
<b>Irregular marking results</b>	<ul style="list-style-type: none"> <li>Check the laser and scanner delays. See <a href="#">Chapter 7.2.3 "Notes on Optimizing the Delays", page 144</a>.</li> </ul>
<b>Laser does not switch off during jump commands</b>	<ul style="list-style-type: none"> <li>Check the laser and scanner delays. See <a href="#">Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control", page 134</a>.</li> </ul>



Additionally, the following commands are helpful for troubleshooting:

- With `get_error` and `get_last_error`, nearly any command can be checked for proper execution, see [Chapter 6.8 "Error Handling", page 120](#).
- With `set_verify`, you can verify that all downloads (commands, tables) were performed error-free, see [Section "Download Verification", page 121](#).
- With `get_value` or `get_values`, you can query specific values returned from the scan-system. With `set_trigger`/`set_trigger4` and `get_waveform`, you can record an entire series of returned values, see [Section "Status Monitoring and Diagnostics", page 172](#) (for iDRIVE scan systems see also [Chapter 8.1 "iDRIVE Functions", page 199](#)).
- With `set_wait` and `get_wait_status`, you can check if program branches (conditional jumps) executed as intended.
- With `get_overrun`, you can check if overruns of the 10 µs clock period occurred, see [Section "Clock Overruns", page 171](#).
- With `get_status` or `get_out_pointer`, you can determine which command number the program is currently executing (for example, for "infinite loops" due to a circular argument in the program flow).
- `get_startstop_info` provides information about the laser signals and possible transmission errors to and from the attached scan system.

If specific outputs from a port have no effect, then check if the user program is performing directly consecutive accesses of that same port. Because so-called short list commands (see [Section "Normal, Short, Variable and Multiple List Commands", page 288](#)) are typically used for this, one command might overwrite the other's output value within the same 10 µs clock period. In this situation, separate both short commands with a (normal) list command, for example, `list_nop` or `list_continue`.

If the execution time (measured by `save_and_restart_timer` and `get_time`) does not correspond with your calculation, check if your user program contains so-called short list commands, which generally do not require their own clock period for execution. Another possibility is that additional scanner delays were automatically inserted to prevent (improper) overlap of LaserOn and LaserOff, see [Section "Automatic Delay Adjustments", page 144](#).

If the problems persist, contact SCANLAB.



## 13 Customer Service

### 13.1 Servicing and Repairs

All RTC6 board servicing and repairs should be performed only at SCANLAB. The warranty expires if the RTC6 board has been altered.

### 13.2 Warranty

SCANLAB guarantees this product to be free of defects in manufacturing and material. The warranty is valid for 12 months after delivery. Repairs covered under the warranty are performed at SCANLAB.

The scope of the warranty is limited to repair or replacement of the SCANLAB product.

SCANLAB is responsible for the return delivery of products repaired under warranty; the customer is responsible for delivery to SCANLAB.

SCANLAB is not held responsible:

- when the product has been damaged through misuse or improper operation
- for RTC6 board repairs not performed by SCANLAB
- for damage resulting from improper packaging of a product returned to SCANLAB
- if the RTC6 board has been altered
- for consequential damages

### 13.3 Contacting SCANLAB

For service, repairs, advice or information, simply contact SCANLAB using one of the contact possibilities listed below:

SCANLAB GmbH  
Siemensstr. 2a  
82178 Puchheim  
Germany

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

[info@scanlab.de](mailto:info@scanlab.de)  
[www.scanlab.de](http://www.scanlab.de)

### 13.4 Product Disposal

The RTC6 board can be returned to SCANLAB for a fee to be properly disposed of.



## 14 Technical Specifications of the RTC6 PCIe Board

### System Requirements

Windows PC with free PCI-Express slot

Operating system      With  
RTC6 Software Package  
≥ V1.3:  
Microsoft Windows 10, 8,  
7 as 32-bit or 64-bit  
version.  
*Windows XP and  
Windows Vista are not  
supported!*

### Dimensions

Length	160 mm
Height	104.5 mm

### Interface to the PC

PCI-Express x-1, version 1.0

### Scan System Control

Number of list memory      Up to 3, configurable  
areas

Total capacity of list      8,388,608 list positions  
memory

Output interval of      10 µs  
microsteps

Maximum range for the      -524,288...+524,287  
image field coordinates (20-bit signed)

Virtual image field      -268,435,456...  
(for example, for      +268,435,455  
Processing-on-the-fly)      (28 bit, signed  
= 29 bit)<sup>(a)</sup>

(a) With RTC6 Software Package ≥ 1.4.0.

### Interface to Scan System

- SCANHEAD connector

Connector	9-pin D-SUB connector (female)
-----------	-----------------------------------

- 2. SCANHEAD socket connector

Connector	10-pin socket connector.
-----------	--------------------------

- xy output values only  
with enabled Option  
"Second Scan Head  
Control"

- z output values only  
with enabled Option  
"3D"

Signal transmission	SL2-100 protocol. Via XY2-100 converter (accessory): XY2-100 protocol
---------------------	--



### Interfaces to the Laser and Peripherals

		Inputs for external start and stop signals	TTL active-LOW, internally connected to +3.3 V by pull-up resistors (4.7 kΩ)
• LASER Connector		• /START	edge sensitive HIGH level > 2.3 V LOW level < 0.6 V
Connector	15-pin D-SUB connector (female)	• /STOP	level sensitive HIGH level > 2.3 V LOW level < 0.6 V
Laser output signals	LASER1, LASER2, LASERON	• Reference	GND <sup>(a)</sup>
• TTL level	5 V, active-HIGH or active-LOW (programmable)	Other signals	
• Max. current load	20 mA	• BUSY OUT	5 V, TTL active-HIGH, max. 10 mA
• Reference	GND <sup>(a)</sup> (GND2 with Option "DC/DC Converter")	• +5 V	max. 100 mA
Analog outputs	ANALOG OUT1, ANALOG OUT2	• Reference	GND <sup>(a)</sup>
• Output voltage range	0 V...10 V		
• Resolution	12 Bit		
• Max. current load	5 mA		
• Reference	GND <sup>(a)</sup>		
Digital input	2 Bits		
• LOW level	< 0.6 V		
• HIGH level	> 2.3 V		
• Max. input voltage range	-0.5 V...+5.5 V		
• Input resistance (pull-up)	> 4.7 kΩ		
• Reference	GND <sup>(a)</sup>		
Digital output	2 bits, buffered		
• LOW level	< 0.55 V		
• HIGH level	> 3.8 V		
• Max. current load	20 mA		
• Reference	GND <sup>(a)</sup>		

(a) See footnote on [page 62](#).

• EXTENSION 1 Socket Connector

Connector	40-pin socket connector, output signal level config- urable by JP1
Digital output	16 bits, buffered
• LOW level	< 0.4 V
• HIGH level	> 2.0 V (3.3 V or 5 V)
• Max. current load	±8 mA
• Reference	GND <sup>(a)</sup>
• LATCH signal	3.3 V or 5 V, TTL active-HIGH, 5 µs pulse, max. 10 mA
Digital input	16 bits
• LOW level	< 0.5 V
• HIGH level	> 2.6 V...24 V
• Max. input voltage range	-0.5 V...+26 V
• Input resistance	> 10 kΩ
• Reference	GND <sup>(a)</sup>
• SYNC signal	3.3 V or 5 V, TTL active-HIGH, square wave signal (5 µs pulse, 10 µs period) max. 10 mA
Other signals	
• BUSY OUT	3.3 V or 5 V, TTL active-HIGH, max. 10 mA
• VCC	3.3 V or 5 V, max. 100 mA
• +5 V	max. 100 mA
• Reference	GND <sup>(a)</sup>

• EXTENSION 2 Socket Connector

Connector	26-pin socket connector, configurable by JP2...JP8
Digital output	8 bits, buffered
• LOW level	< 0.4 V
• HIGH level	> 2.0 V
• Max. current load	±8 mA
• Reference	GND <sup>(a)</sup>
• LATCH signal	5 V, TTL active-HIGH, 5 µs pulse, max. 10 mA
Laser signals	LASERON, LASER1, LASER2 (see LASER connector)
Other signals	
• +5 V	max. 100 mA
• Reference	GND <sup>(a)</sup>
• MARKING ON THE FLY Socket Connector	
Connector	16-pin socket connector
2 Encoder inputs for incremental encoders	ENCODER X(1±,2±) and ENCODER Y(1±,2±), designed for a pair of stan- dardized differential input signals (RS-422) each. HIGH level ≥ 2.0 V LOW level ≤ 0.8 V $f \leq 4 \text{ MHz}$
Analog outputs	ANALOG OUT2 (see LASER connector)
Inputs for external start and stop signals	/START2, /STOP2 (see /START, /STOP of the LASER connector)
Other signals	
• BUSY OUT	5 V, TTL active-HIGH, max. 10 mA
• +5 V	max. 100 mA
• Reference	GND <sup>(a)</sup>



- **RS232 Socket Connector**

Connector	10-pin socket connector
Input	RxD
	• Max. voltage range -25 V...+25 V
Output	TxD
	• Max. voltage range -13 V...+13 V
Reference	GND <sup>(a)</sup>
Baud rate	300...115200

- **McBSP/ANALOG Socket Connector**

Connector	10-pin socket connector
Analog inputs	ANALOG IN0, ANALOG IN1
	• Input voltage range 0 V...10 V
	• Input resistance 4 kΩ
	• ADC resolution 12 bit
	• Reference GND <sup>(a)</sup>
McBSP interface	See <a href="#">Section "McBSP Interface", page 73.</a>
	• Transmitter signal level 3.3 V TTL
	• Receiver signal level 3.3 V or 5 V TTL
	• McBSP mode Single Phase Frame Single Element per Frame 32 bits per Element DataDelay N bit
	• Reference GND <sup>(a)</sup>
SPI interface functionality	No

2 Analog inputs (ANALOG IN0, ANALOG IN1):

- Input voltage range 0 V ... 10 V
- Input impedance > 5 kΩ
- ADC resolution 12 bit
- Reference GND<sup>(a)</sup>

- **STEPPER MOTOR Socket Connector**

Connector	10-pin socket connector
	Signals for controlling two stepper motors:
	• ENABLE and 5 V, TTL DIRECTION outputs
	• CLOCK output 5 V, TTL active high, 5 µs pulse
	• SWITCH input TTL active-LOW, internally connected to +3.3 V by pull-up resistors (10 kΩ)
	• Reference GND <sup>(a)</sup>



## **14.1 Compliance with EC Guidelines for Electromagnetic Compatibility (EMC)**

The RTC6 PCIe Board has been determined to be in compliance with EC directive 2014/30/EU (electromagnetic compatibility).

For that purpose, an RTC6 PCIe Board has been integrated to a PC and was tested together with an excelliSCAN 14 scan head (with SL2-100 interface).

### **Test Specifications**

Evidence of fulfillment of the protection goals of EC directive 2014/30/EU (CE Conformity for EMC) based on

- EN 61000-6-2: 2005 + AC: 2005
- EN 61000-6-4: 2007+ A1: 2011

### **Result**

The devices under test fulfill the specifications.

## **14.2 Compliance with FCC Rules**

The RTC6 PCIe Board has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules.

These limits are designed to provide reasonable protection against harmful interference when the RTC6 PCIe Board is operated in a commercial environment. The RTC6 PCIe Board generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with this instruction manual, may cause harmful interference to radio communications. Operation of the RTC6 PCIe Board in a residential area is likely to cause harmful interference in which case the user is required to correct the interference at his own expense.



## 15 Appendix A: The RTC6 Ethernet Board

### 15.1 Product Overview

#### 15.1.1 RTC6 Ethernet Board vs. RTC6 PCIe Board – Usage and Comparison

RTC6 Ethernet Boards and RTC6 PCIe Boards share the same *core functionality*:

- Real-time control of the laser and scan system.
- Transfer of direct commands (control commands) from the computer to the board. These commands are executed immediately.
- Transfer of list commands from the computer to the list memory of the board. These commands execute only after the list is started.
- List execution occurs in real time.
- Calculation of the set position occurs every 10 µs.
- Field correction is applied to the set positions.
- The interface to the scan system uses the 20-bit SL2-100 protocol.
- For software development, the total command set for RTC6 Ethernet Boards and RTC6 PCIe Boards is contained in [RTC6DLL.dll/RTC6DLLx64.dll](#).

RTC6 Ethernet Boards differ in the following aspects:

- Ethernet interface<sup>(1)</sup> instead of PCIe bus
  - Communication with the computer is by TCP/IP and UDP.
  - RTC6 Ethernet Boards work *without* RTC6 board driver.
  - The board does not need to be plugged into a PC.
  - Real-time clock, see [Chapter 15.2.15 “Real-Time Clock”, page 849](#).

(1) Supports Gigabit Ethernet according to 1000BASE-T.

## 15.1.2 System Requirements



### Caution!

- To avoid interference coupling and emissions, never operate the RTC6 Ethernet Board without shielding (e.g. outside a metal housing).
- The RTC6 Ethernet Board is designed exclusively for industrial use. It is intended for integration in a machine (typically in a laser system) and does **not** fulfill all requirements of a ready-to-use consumer end product. Only operate the RTC6 Ethernet Board after integration in a machine meeting all applicable directives and standards (of your local jurisdiction). It is *not* suitable for use as a toy, in households or inclement environments (e.g. outdoors). The operator must take appropriate precautionary measures to prevent such improper usage.
- Installation and commissioning should only be performed by personnel with sufficient qualifications, e.g. knowledge of electrical equipment safety. Only perform installation and maintenance if power and lasers are switched off.
- Because the RTC6 Ethernet Board is a Class A device capable of generating interference in residential areas, the operator may be required to carry out appropriate measures.

### Hardware

- Shielded housing with:
  - Provisions for mechanical mounting, incl. mounting hardware
  - appropriate thermal coupling (cooling) of the board
  - Power outlet
  - Ethernet cable
  - Power supply as specified
  - Cable with connector for powering the RTC6 Ethernet Board
  - Flat ribbon cables for peripheral equipment (laser, scan head, extensions)
- Windows PC
- Gigabit Ethernet (preferred), 10/100 Mbit/s Ethernet

### Software

- TCP/IP protocol IPv4
- UDP protocol
- Used ports (default)
  - 63749 (UDP)
  - 63750 (UDP, TCP)

For the development of user programs:

- DLL/Import declarations (contained in RTC6 Software Package)

## 15.1.3 Options

Same as RTC6 PCIe Boards, see [Chapter 2.3 "Options", page 30](#).

## 15.1.4 Labeling

Same as RTC6 PCIe Boards, see [Chapter 1.2 "Labeling", page 24](#).

## 15.1.5 Type Identification

Same as RTC6 PCIe Boards, see [Chapter 2.4 "Jumper Settings and Type Designations", page 32](#).



### **15.1.6 Unpacking Instructions and Typical Scope of Delivery**

Same as RTC6 PCIe Boards, see [Chapter 1.3 "Unpacking Instructions and Typical Scope of Delivery", page 24.](#)

### **15.1.7 Delivered Software**

Same as RTC6 PCIe Boards, see [Chapter 1.3.1 "Delivered RTC6 Software Package", page 24.](#)

### **15.1.8 Accessories for the RTC6 PCIe Board**

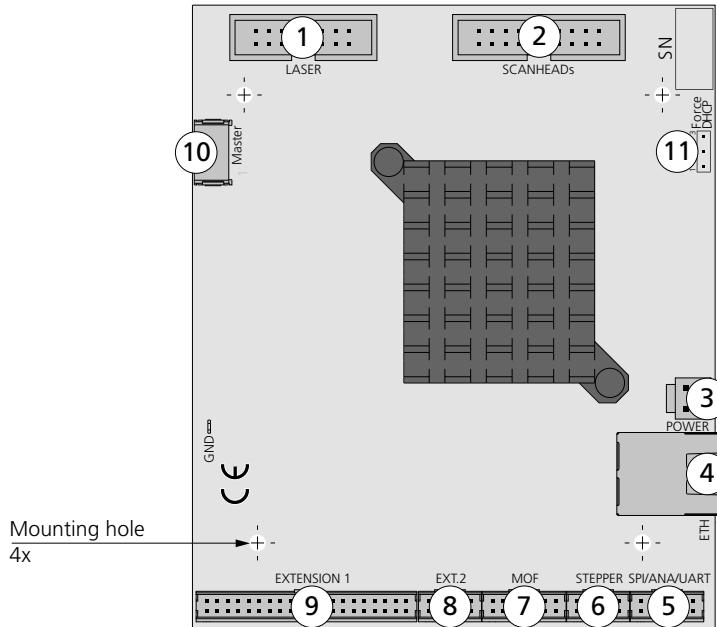
Accessories mentioned in [Chapter 2.5 "Accessories for the RTC6 PCIe Board", page 36](#) for RTC6 PCIe Board is (without suiting adapters) not suitable for RTC6 Ethernet Boards (because of different connector plugs and pitch of the pins).

### **15.1.9 Supplementary Software**

As with RTC6 PCIe Boards, see [Chapter 2.6 "Supplementary Software", page 37.](#)

## 15.2 Layout, Interfaces, Jumper Settings

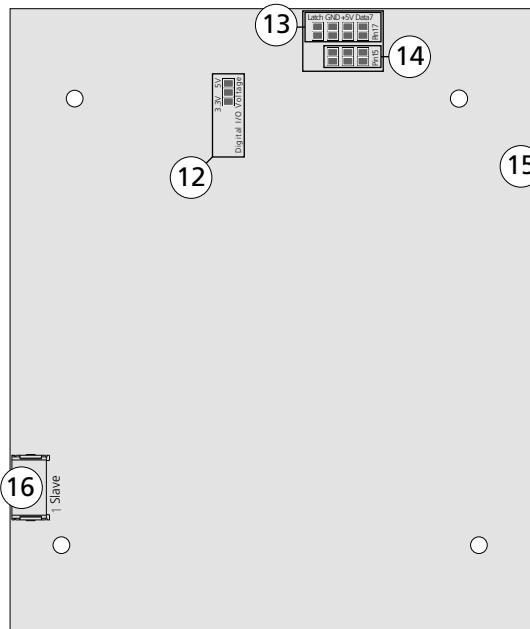
### 15.2.1 Layout – Upper Side



#### Legend

1. LASER ..... 16-pin socket connector. To connect the laser. With digital and analog outputs.  
For details, see "[LASER Socket Connector](#)", page 834.
2. SCANHEADS ..... 20-pin socket connector. To connect the first and second scan head.  
For details, see "[SCANHEADS Socket Connector](#)", page 837.
3. POWER ..... 2-pin socket connector. For the voltage supply.  
For details, see "[POWER Socket Connector](#)", page 839.
4. ETH ..... RJ-48 8P8C connector. For the network connection.  
For details, see [Chapter 15.2.7 "ETH Connector"](#), page 839.
5. SPI/ANA/UART ..... 12-pin socket connector. Hardware interface for several data exchange methods.  
With analog inputs. For details, see "[SPI/ANA/UART Socket Connector](#)", page 840.
6. STEPPER ..... 10-pin socket connector. For controlling up to two stepper motors.  
For details, see "[STEPPER Socket Connector](#)", page 841.
7. MOF ..... 14-pin socket connector. Hardware interface for encoder pulses, for example, for Processing-on-the-fly applications. For details, see "[MARKING ON THE FLY Socket Connector](#)", page 71.
8. EXT. 2 ..... 10-pin socket connector. With a 8-bit digital output port.  
For details, see "[EXTENSION 2 Socket Connector](#)", page 69.
9. EXTENSION 1 ..... 40-pin socket connector. With a 16-Bit Digital Output and  
a 16-Bit Digital Input. For details, see "[EXTENSION 1 Socket Connector](#)", page 67.
10. Master ..... 6-pin socket connector. To connect with another RTC6 board for the purpose of synchronize clocks. For details, see "[Master Socket Connector, Slave Socket Connector](#)", page 845.
11. Force DHCP ..... Jumper field Jumper field to ignore or not to ignore the saved static IP address.  
For details, see "[Jumper Field 'Force DHCP'](#)", page 849.

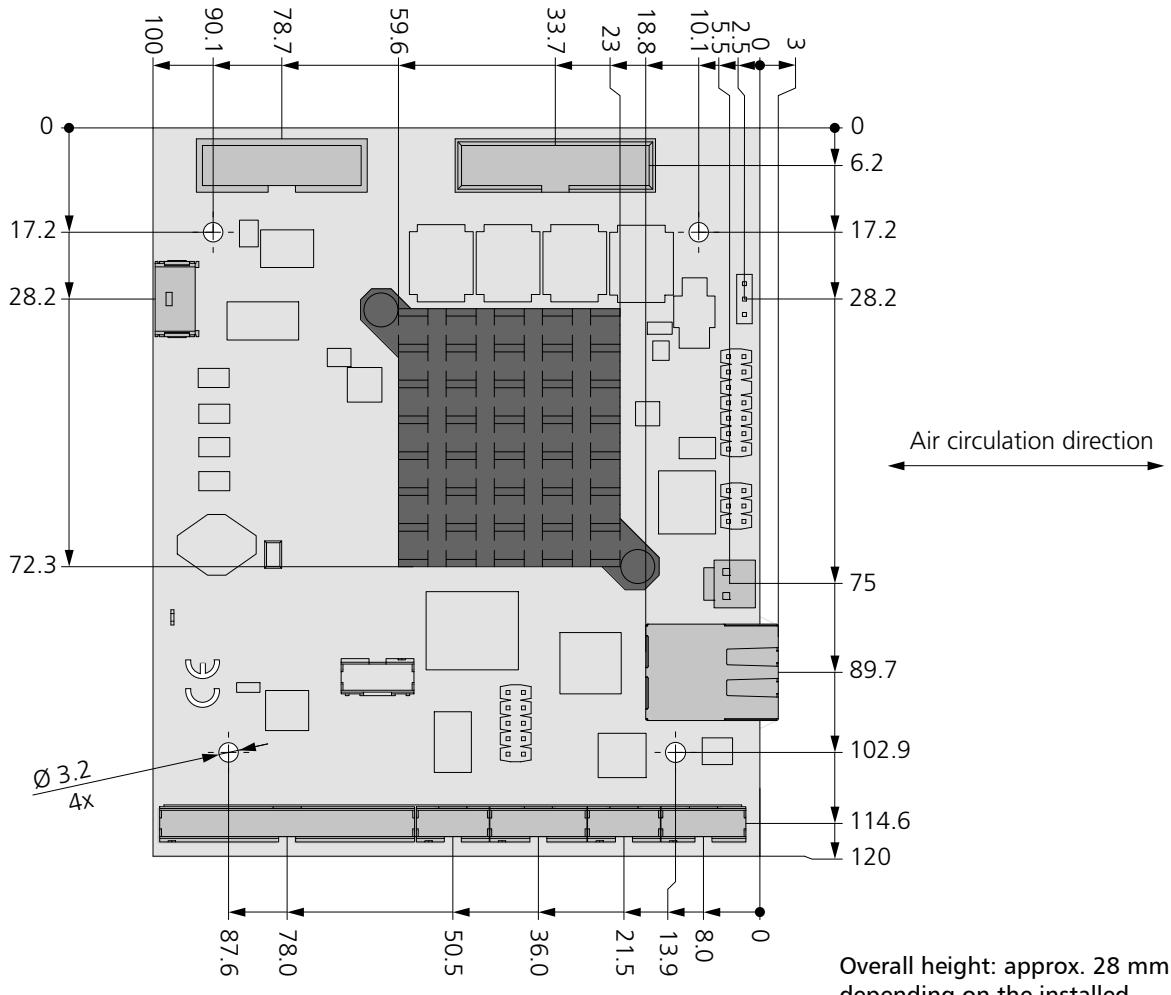
### 15.2.2 Layout – Lower Side



#### Legend

- 12. Solder jumper field A . . . . To configure the level of the output signals at the EXTENSION 1 socket connector.  
For details, see "[Solder Jumper Field A – Output Signal Level at the EXTENSION 1 Socket Connector Configuration](#)", page 33.
- 13. Solder jumper field B . . . . To configure the signal at EXT. 2 socket connector pin (09).  
For details, see "[Solder Jumper Field B – Configuring pin \(09\) of the EXT. 2 Socket Connector](#)", page 847.
- 14. Solder jumper field C . . . . To configure the signal at EXT. 2 socket connector pin (08).  
For details, see "[Solder Jumper Field C – Pin \(15\) of the EXTENSION 2 Socket Connector Configuration](#)", page 35.
- 15. ETH connector . . . . See [figure 67](#).
- 16. Slave. . . . . 6-pin socket connector. To connect with another RTC6 board for the purpose of synchronize clocks. For details, see "[Master Socket Connector, Slave Socket Connector](#)", page 845.

### 15.2.3 Dimensions and Connector Positions



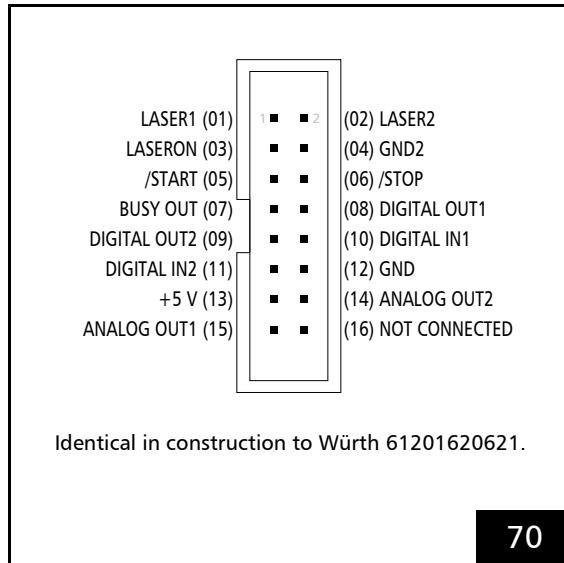
69

RTC6 Ethernet Board V1.2. Dimensions and connector positions. All dimensions in mm.

### 15.2.4 LASER Socket Connector

The LASER socket connector has 16 pins<sup>(1)</sup>. It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 1.

The pin-out is shown in [figure 70](#).



LASER socket connector: pin-out. The pitch of the pins is 2.54 mm. On RTC6 Ethernet Boards without [Option "DC/DC Converter"](#), GND and GND2 are identical. On RTC6 Ethernet Boards with [Option "DC/DC Converter"](#), GND2 and the laser output signal are galvanically decoupled from GND.

#### Notes

- With RTC6 Ethernet Boards, GND is the ground at the POWER connector. See [Chapter 15.2.6 "POWER Socket Connector", page 839](#). With the RTC6 PCIe Board, GND is the PC ground.
- A suitable 0.2 m cable (#116048) is available from SCANLAB, see also [figure 71](#).

### Laser Output Signals

As with RTC6 PCIe Boards, see [Chapter 4.6.1 "LASER Connector", page 62](#). However, note that the pin numbers differ.

Signal	RTC6 Ethernet Board	RTC6 PCIe Board
LASERON	Pin (03)	Pin (02)
LASER1	Pin (01)	Pin (01)
LASER2	Pin (02)	Pin (09)

### External Control Signals

As with RTC6 PCIe Boards, see [Chapter 4.6.1 "LASER Connector", page 62](#).

### BUSY Status

As with RTC6 PCIe Boards, see [Chapter 4.6.1 "LASER Connector", page 62](#). However, note that the pin numbers differ.

Signal	RTC6 Ethernet Board	RTC6 PCIe Board
BUSY OUT	Pin (07)	Pin (04)

### 2-Bit Digital Input and 2-Bit Digital Output

As with RTC6 PCIe Boards, see [Chapter 4.6.1 "LASER Connector", page 62](#). However, note that the pin numbers differ.

Signal	RTC6 Ethernet Board	RTC6 PCIe Board
DIGITAL IN1	Pin (10)	Pin (13)
DIGITAL IN2	Pin (11)	Pin (06)
DIGITAL OUT1	Pin (08)	Pin (12)
DIGITAL OUT2	Pin (09)	Pin (05)

(1) With RTC6 PCIe Boards: 15 pin D-SUB connector, female.

## 12-Bit Analog Output

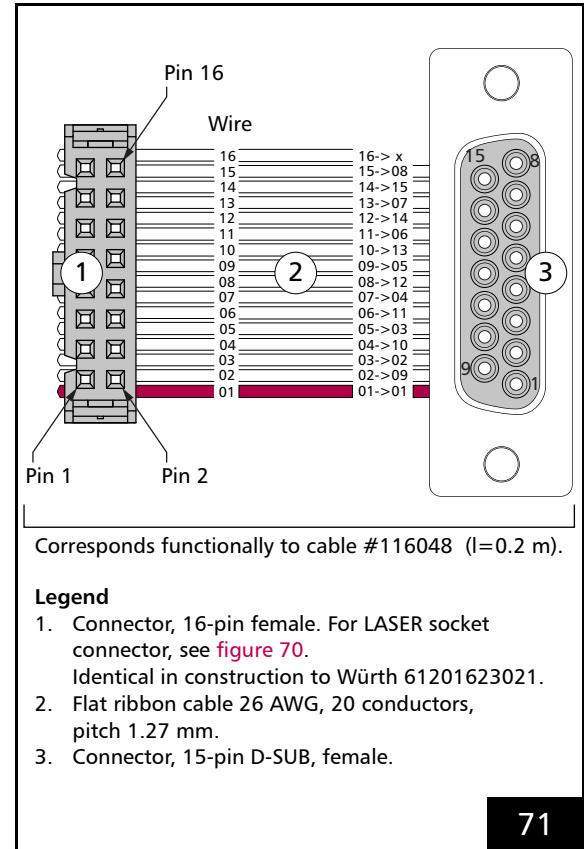
As with RTC6 PCIe Boards, see [Chapter 4.6.1 "LASER Connector", page 62](#). However, note that the pin numbers differ.

Signal	RTC6 Ethernet Board	RTC6 PCIe Board
ANALOG OUT1	Pin (15)	Pin (08)
ANALOG OUT2 <sup>(a)</sup>	Pin (14)	Pin (15)

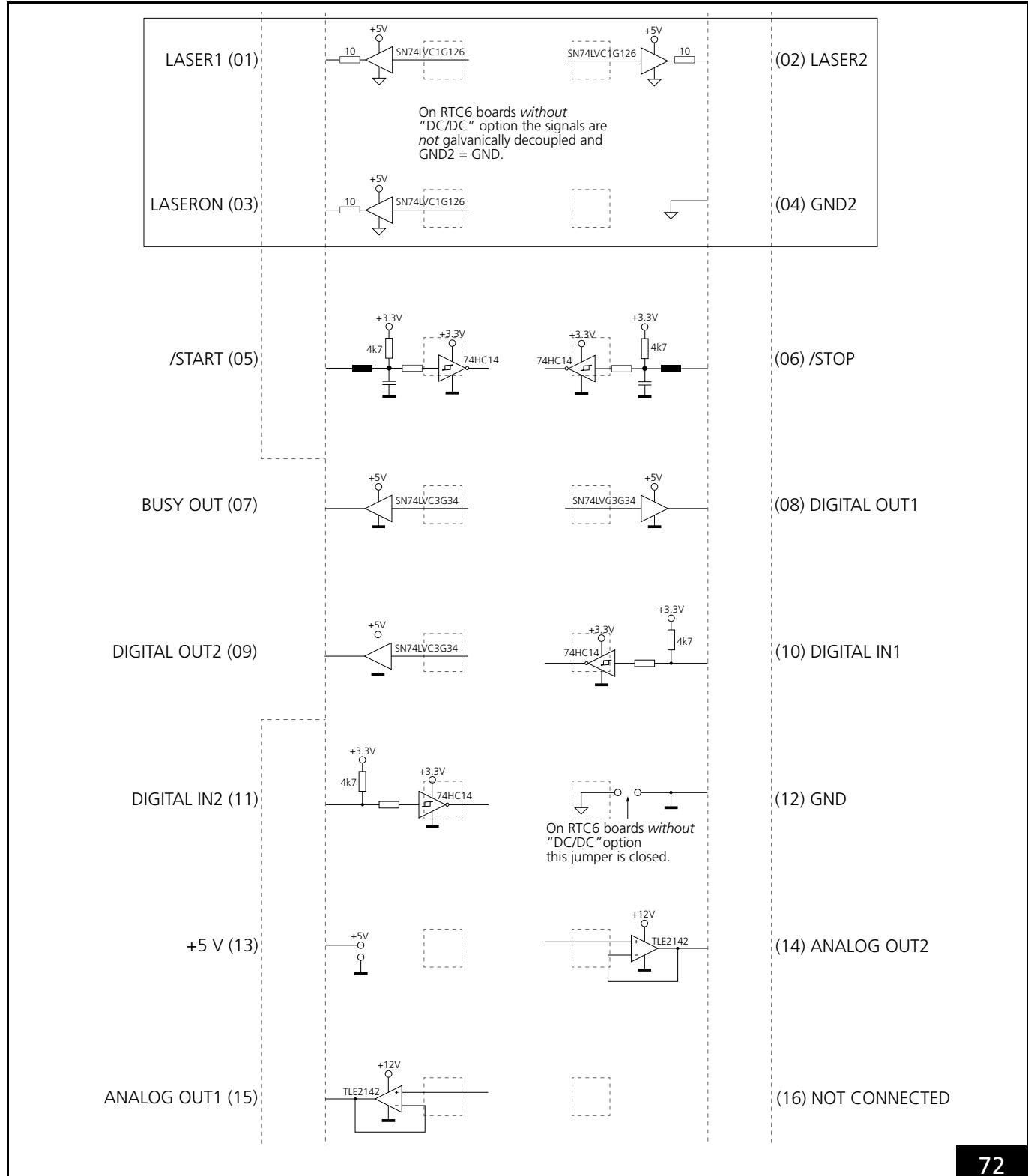
- (a) With the RTC6 PCIe Board the ANALOG OUT2 signal is additionally available at the MARKING ON THE FLY socket connector.

## Input and Output Wiring

The input and output wiring of the 16-pin LASER socket connector is shown in [figure 72](#). It is identical to RTC6 PCIe Boards, except the pin numbers.



Cable (proposal). Actual implementation may differ according to customer needs.  
The depicted items are not in the standard scope of delivery of the RTC6 Ethernet Board!



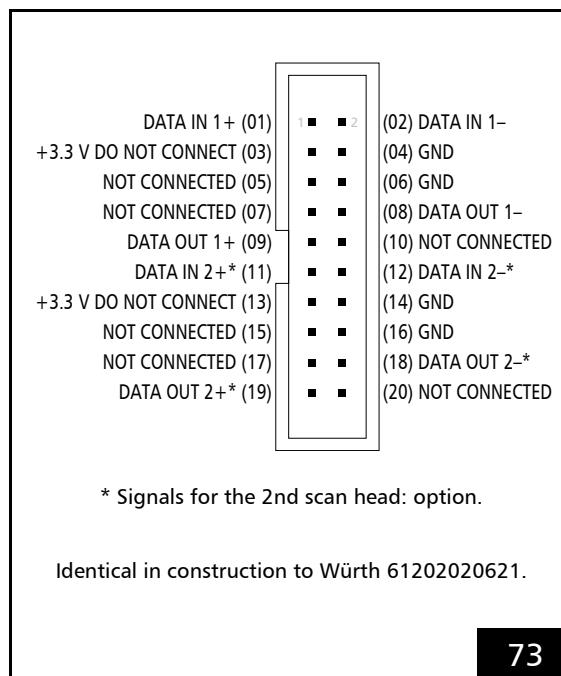
LASER socket connector (detail, indicated by dashed line): input/output wiring plan.

### 15.2.5 SCANHEADs Socket Connector

The SCANHEADs socket connector has 20 pins. It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 2.

The SCANHEADs socket connector provides the same signals as the SCAN HEAD and 2. SCANHEAD socket connector of the RTC6 PCIe Board for the first scan head connector (pin (01)...pin (10)) and the second scan head-connector (pin (11)...pin (20)).

The pin-out is shown in [figure 73](#).



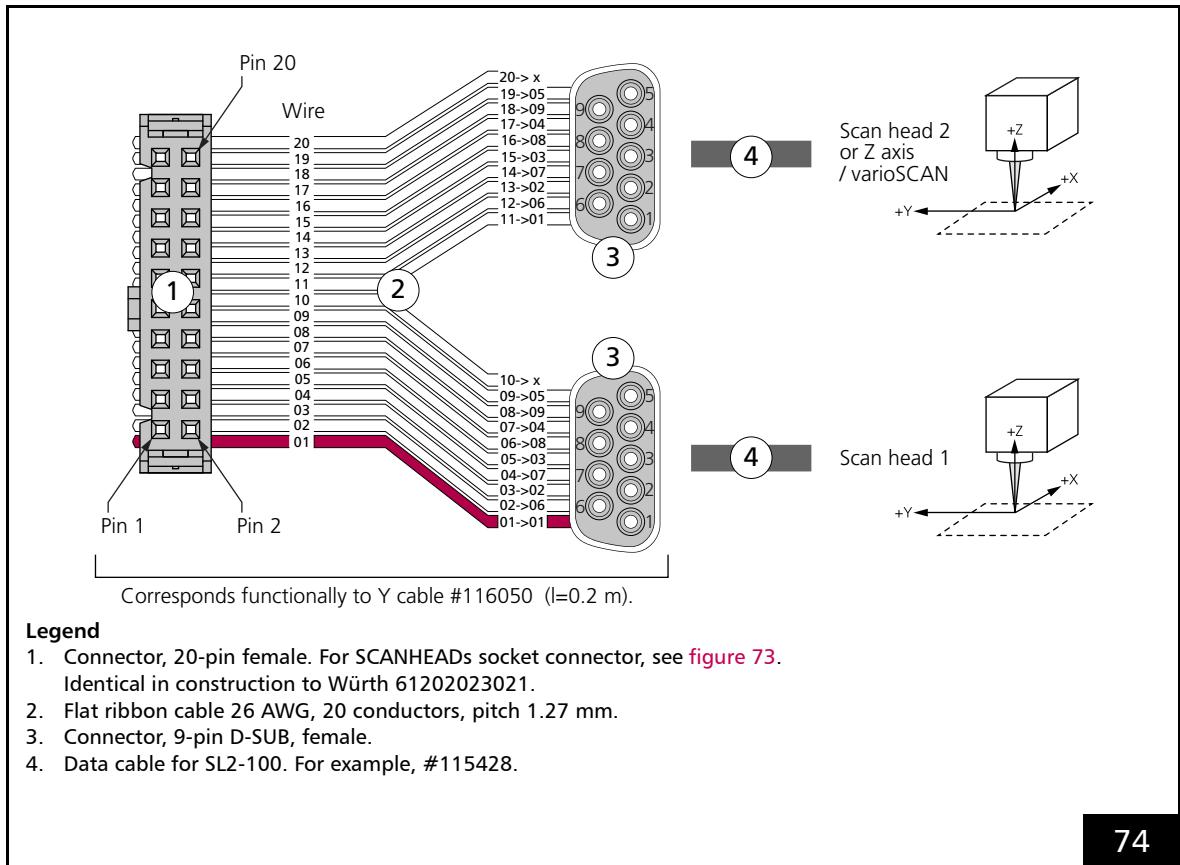
73

SCANHEADs socket connector: pin-out. The pitch of the pins is 2.54 mm.

A cabling example is shown in [figure 74](#) (the shown Y cable is available from SCANLAB with a length of 0.2 m: #116050).

Scan system control (via an XY2-100 converter, if needed) is identical to that of the RTC6 PCIe Board.

For further information, see [Chapter 4.5 "Interfaces to Scan System", page 56](#).



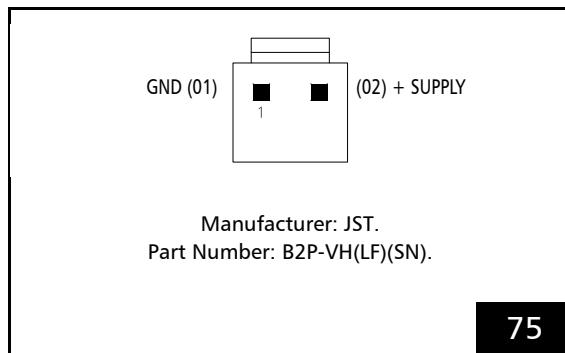
74

Cabling (proposal). Actual implementation may differ according to customer needs.  
The depicted items are not in the standard scope of delivery of the RTC6 Ethernet Board!

### 15.2.6 POWER Socket Connector

The POWER socket connector has 2 pins. It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 3. It serves to provide the voltage supply for the RTC6 Ethernet Board.

The pin-out is shown in [figure 75](#).



POWER socket connector. The pitch of the pins is 3.96 mm.

Pin	Power supply
(01) GND	Ground.
(02) + Supply	See <a href="#">Chapter 15.8 "Technical Specifications for the RTC6 Ethernet Board", page 866.</a>

#### Notes

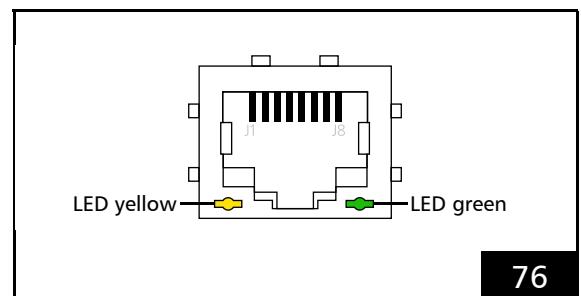
- Required parts for the mating connector are *not included in the scope of delivery*:
- Manufacturer: JST  
 1x Housing, part number VHR-2N  
 2x Contact, part number SVH-41T-P1.1.

### 15.2.7 ETH Connector

The ETH connector is an 8P8C type RJ connector (colloquially: RJ-45). It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 4. It serves to connect<sup>(1)</sup> the RTC6 Ethernet Board to the network.

For status indication the following LEDs are part of the connector, see [figure 76](#):

- LED yellow, function "IP address assigned". The LED is *permanently on* as soon as the RTC6 Ethernet Board has an assigned IP address (static IP, DHCP or as of [BIOS](#) 24 link-local address<sup>(2)</sup>). Note: Without stored static IP address the switched-on LED indicates that the IP address assignment has been carried-out successfully by DHCP. In case if there is a stored static IP address a switched-on LED does not mean that the IP address is valid in the subnet in every case. As soon as the board has been acquired by a PC the LED *flashes* when there is network traffic ("Traffic").
- LED green, function "Link / Traffic". The LED is *permanently on* as soon as a connection is established ("Link"). This is usually the case when the Ethernet cable is plugged-in and there is an active router or a LAN adapter at the remote end. The LED *flashes* when there is network traffic ("Traffic").



ETH connector. RJ-48 8P8C.

(1) Category 5e (Cat 5e) Ethernet cable recommended.

(2) See [page 850](#).

### 15.2.8 SPI/ANA/UART Socket Connector

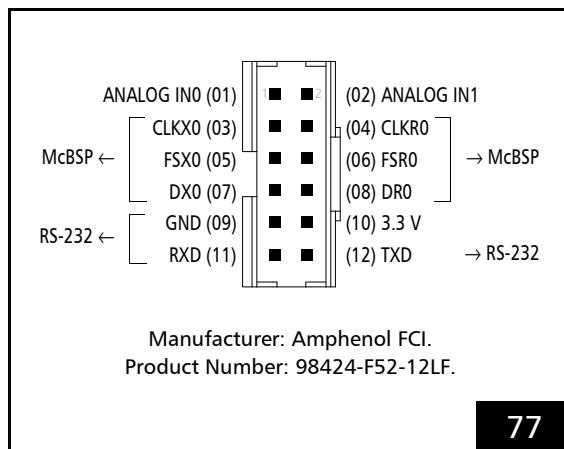
The SPI/ANA/UART socket connector has 12 pins. It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 5.

The SPI/ANA/UART socket connector is a hardware interface intended for analog inputs as well as for the data exchange methods **McBSP** (Multichannel Buffered Serial Port) and RS-232.

The SPI/ANA/UART socket connector corresponds functionally to the:

- McBSP/ANALOG socket connector and RS232 socket connector of RTC6 PCIe Boards

The relevant pin-outs are shown in [figure 77](#).



77

SPI/ANA/UART socket connector: pin-out. The pitch of the pins is 2.00 mm.

#### Notes

- Required parts for the mating connector are *not included in the scope of delivery*:  
Manufacturer: Amphenol FCI  
Series. Minitek®  
1x Housing, part number 90311-012LF  
12x Contact, part number 77138-101LF.

#### Analog Inputs

The SPI/ANA/UART socket connector provides two analog inputs: ANALOG IN0 and ANALOG IN1.

See [Section "Analog Input Ports", page 76](#).

#### Specifications

- Input voltage range: 0 V...10 V.
- Input impedance: > 5 kΩ
- ADC resolution: 12 bit

The input signals are referenced to GND, see ["Notes", page 834](#).

#### McBSP Interface

See [Section "McBSP Interface", page 73](#).

#### RS-232 Interface

With RTC6 Ethernet Boards, the RS232 socket connector pins of the RTC6 PCIe Board are integrated to the SPI/ANA/UART<sup>(1)</sup> socket connector.

Specifications and further information as with the RTC6 PCIe Board, see [Chapter 4.6.5 "RS232 Socket Connector", page 72](#).

The signals are referenced to GND, see ["Notes", page 834](#).

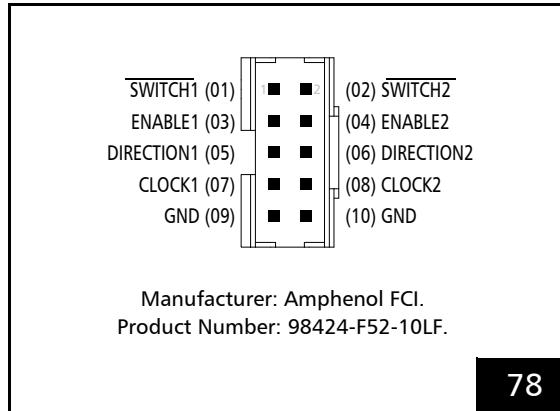
(1) The socket connector is labeled ".../UART" (Universal asynchronous receiver/transmitter). Refers to the electronic circuit which generates the data bits and the necessary data frame to be transferred on the RS-232 interface.

### 15.2.9 STEPPER Socket Connector

The STEPPER socket connector has 10 pins. It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 6.

At the STEPPER socket connector signals for controlling up to two stepper motors can be outputted.

The pin-out is shown in [figure 78](#).



78

STEPPER socket connector: pin-out. The pitch of the pins is 2.00 mm.

All signals are referenced to GND, see "[Notes](#)", [page 834](#).

Specifications and further information as with the RTC6 PCIe Board, [Chapter 4.6.7 "STEPPER MOTOR Socket Connector"](#), [page 77](#).

#### Notes

- Required parts for the mating connector are *not included in the scope of delivery*:

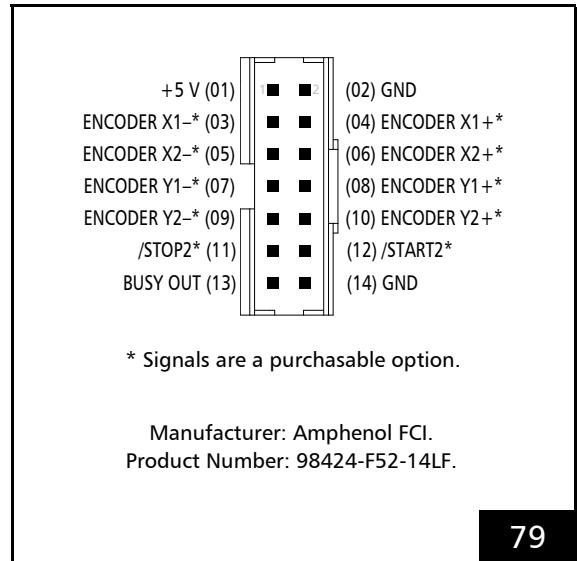
Manufacturer: Amphenol FCI  
Series. Minitek®

1× Housing, part number 90311-010LF  
10× Contact, part number 77138-101LF.

### 15.2.10 MOF Socket Connector

The MOF<sup>(1)</sup> socket connector has 14 pins. It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 7.

The pin-out is shown in [figure 79](#).



79

MOF socket connector: pin-out. The pitch of the pins is 2.00 mm.

#### Notes

- Required parts for the mating connector are *not included in the scope of delivery*:  
Manufacturer: Amphenol FCI  
Series. Minitek®  
1× Housing, part number 90311-014LF  
14× Contact, part number 77138-101LF.
- RTC6 Ethernet Boards *do not* have a pin for the ANALOG OUT2 signal at their MOF socket connector, see [Chapter 4.6.4](#)  
"[MARKING ON THE FLY](#) Socket Connector", [page 71](#).

(1) Marking on the Fly (= Processing-on-the-fly).

### Encoder Inputs

As with RTC6 PCIe Board, see [Section "Encoder Inputs", page 71.](#)

### External Control Signals

As with RTC6 PCIe Board, see [Section "External Control Signals", page 71.](#)

The signals are referenced to GND, see ["Notes", page 834.](#)

### BUSY OUT Status

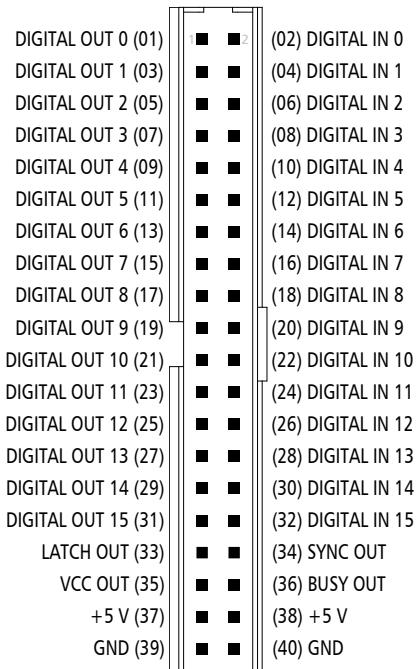
The BUSY OUT signal at pin (13) is identical to the BUSY OUT signal at the LASER socket connector, see [Chapter 15.2.4 "LASER Socket Connector", page 834.](#)

The signal is referenced to GND, see ["Notes", page 834.](#)

## 15.2.11 EXTENSION 1 Socket Connector

The EXTENSION 1 socket connector has 40 pins. It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 9.

The pin-out is shown in [figure 80.](#)



Manufacturer: Amphenol FCI.  
Product Number: 98424-F52-40LF.

80

EXTENSION 1 socket connector: pin-out. The pitch of the pins is 2.00 mm.

### Notes

- Required parts for the mating connector are *not included in the scope of delivery*:  
Manufacturer: Amphenol FCI Series. Minitek®  
1 x Housing, part number 90311-040LF  
40 x Contact, part number 77138-101LF.
- The EXTENSION 1 socket connector on RTC6 Ethernet Boards has a 2.00 mm pitch of the pins, whereas RTC6 PCIe Boards have 2.54 mm.

## Configuring the Output Signal Level

With the solder jumper field A on the lower side of the RTC6 Ethernet Board, the level of all output signals at the EXTENSION 1 socket connector (DIGITAL OUT 0...DIGITAL OUT 15, LATCH\_OUT, SYNC\_OUT, BUSY\_OUT, VCC\_OUT) can be configured for 5 V or 3.3 V, see [Chapter 2.4.1](#)

["Solder Jumper Field A – Output Signal Level at the EXTENSION 1 Socket Connector Configuration", page 33.](#)

The configured signal level is stationary outputted at pin (35): signal VCC\_OUT. VCC\_OUT is referenced to GND, see ["Notes", page 834.](#)

The maximum current load of the signal is 100 mA.

## 16-Bit Digital Output and 16-Bit Digital Input

As with RTC6 PCIe Board, see [Section "16-Bit Digital Input and 16-Bit Digital Output", page 67.](#)

The signals are referenced to GND, see ["Notes", page 834.](#)

## Synchronization of Data Acquisition

As with RTC6 PCIe Board, see [Section "Synchronization of Data Acquisition", page 68.](#)

The signals are referenced to GND, see ["Notes", page 834.](#)

## BUSY Status

The BUSY OUT signal at pin (36) is identical to the BUSY OUT signal at the LASER socket connector, see [Chapter 15.2.4 "LASER Socket Connector", page 834.](#)

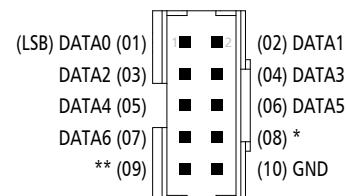
The signal is referenced to GND, see ["Notes", page 834.](#)

## 15.2.12 EXT. 2 Socket Connector

The EXT. 2 socket connector<sup>(1)</sup> has 10 pins. It is located on the upper side of the RTC6 PCIe Board, see [figure 67, number 8.](#)

It provides a buffered 8-bit digital output port ((DATA0...DATA7).

The pin-out is shown in [figure 20.](#)



\* Jumper setting-dependent:  
+5V or DATA7 or GND.

\*\* Jumper setting-dependent:  
+5V or DATA7 or GND or LATCH.

Manufacturer: Amphenol FCI.  
Product Number: 98424-F52-10LF.

81

EXT. 2 socket connector: pin-out. The pitch of the pins is 2.00 mm.

On RTC6 PCIe Boards the designation is EXTENSION 2.

Apart from the total number of pins, the provided signals differ as well as the pin numbers, see following table.

(1) On the RTC4, the functional corresponding socket connector is labeled LASER EXTENSION. On RTC5 and RTC6 PCIe Boards, it is labeled with EXTENSION 2.

Signal	RTC6 Ethernet Boards, EXT. 2 socket connector 10 pins	RTC6 PCIe Boards, EXTENSION 2, 26 pins
DATA0	Pin (01)	Pin (01)
DATA1	Pin (02)	Pin (03)
DATA2	Pin (03)	Pin (05)
DATA3	Pin (04)	Pin (07)
DATA4	Pin (05)	Pin (09)
DATA5	Pin (06)	Pin (11)
DATA6	Pin (07)	Pin (13)
* , see in figure 20	Pin (08)	Pin (15)
** , see in figure 20	Pin (09)	Pin (17)
GND	Pin (10)	Pin (02)
+5 V	–	Pin (06), (18), (25)
LASER1	–	Pin (22)
LASER2	–	Pin (19)
GND2	–	Pin (23)

#### Notes

- Required parts for the mating connector are *not included in the scope of delivery*:  
Manufacturer: Amphenol FCI  
Series. Minitek®  
1× Housing, part number 90311-010LF  
10× Contact, part number 77138-101LF.
- The pin (08) and pin (09) are configurable by solder jumpers.
- RTC6 Ethernet Boards *do not have* pins for LASER1 and LASER2 at their (10-pin) EXT. 2 socket connectors.

#### Configuration by Solder Jumpers

The pin (08) of the EXT. 2 socket connector is configured by the solder jumper field C whereas pin (09) is configured by solder jumper field B. Both jumper fields are on the lower side of the RTC6 Ethernet Board, see [figure 68](#). For further information, see [Section "Solder Jumper Field B – Configuring pin \(09\) of the EXT. 2 Socket Connector", page 847](#) and [Section "Solder Jumper Field C – Configuring pin \(08\) of the EXT. 2 Socket Connector", page 848](#).

#### Notes

- If the DATA7 bit (DATA7) is assigned to pin (08), then the full 8-bit output value is available at the output port (pins (1) to pin (08) of the EXT. 2 socket connector).
- If pin (08) is set to +5 V (HIGH level), an offset of 128 results for the output value. That is, the output value range is from 128...255.
- If pin (08) is set to GND (LOW level) the output value range is restricted to 0 ... 127.
- The DATA7 bit can be used for other purposes by assigning it to pin (09).

#### 8-Bit Digital Output Port

As with RTC6 PCIe Board, see [Section "8-Bit Digital Output Port", page 70](#).

### 15.2.13 Master Socket Connector, Slave Socket Connector

The Master socket connector has 6 pins, see [figure 82](#). It is located on the upper side of the RTC6 Ethernet Board, see [figure 67](#), number 10.

The Slave socket connector has 6 pins, see [figure 82](#). It is located on the lower side of the RTC6 Ethernet Board, see [figure 67](#), number 16.

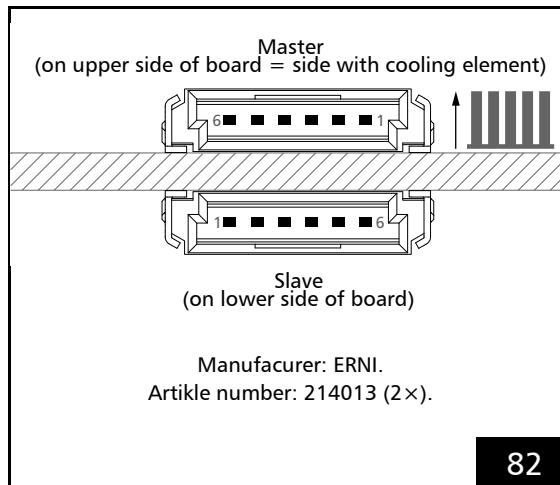
The both socket connectors serve to synchronize the clock cycles of several RTC6 boards.

Connection cables are available from SCANLAB, see [figure 6, page 55](#). The necessary information for assembling your own cables is shown in [figure 5, page 55](#).

For further information, see

- [Chapter 6.6.3 "Master/Slave Operation", page 114](#)
- [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 275.](#)

A specific hardware revision of the RTC6 Ethernet Board is required for proper functioning of the master/slave clock synchronization (see label with part number and modification index). If you have an older RTC6 Ethernet Board or are unsure, contact SCANLAB.



82

Master socket connector and Slave socket connector.  
The pitch of the pins is 1.27 mm.

### Notice!

- Master/Slave-connected RTC6 Ethernet Boards are destroyed by different potentials. This may even be the case, if the boards are connected to the same power supply but the cabling is unfavourable.  
To avoid different potentials:
  - (1) Make sure that the power supply is switched off.
  - (2) First connect one RTC6 Ethernet Board to the power supply, and then connect the other one through a branch of this cable (keep the length as short as possible).
  - (3) Only then switch the power supply back on again.
- Always switch off the power supply of all Master/Slave-connected RTC6 Ethernet Boards before disconnecting the cabling from them.

### 15.2.14 Jumper Settings

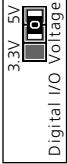
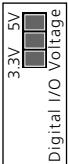
SCANLAB ships RTC6 Ethernet Boards in various jumper configurations. The jumpers can be reconfigured at a later time. See [Chapter 2.4 "Jumper Settings and Type Designations"](#), page 32.

#### Notice!

- Only configure allowed jumper settings.  
Otherwise, the board gets damaged!

#### Solder Jumper Field A – Configuring the Output Signal Level at the EXTENSION 1 Socket Connector

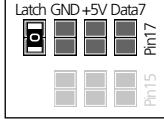
- Position on the board: lower side, see [figure 68](#), number 12.
- Purpose: to configure the level (5 V or 3.3 V) of all output signals at the EXTENSION 1 socket connector, see the following table.
- See also "[Configuring the Output Signal Level](#)", page 67.

Allowed jumper setting	At the EXTENSION 1 socket connector
	closed* open  Output signal level 5 V.
	open closed*  Output signal level 3.3 V.
	open open  No signal output.

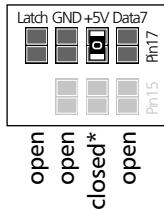
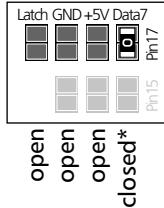
\* Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

### Solder Jumper Field B – Configuring pin (09) of the EXT. 2 Socket Connector

- Position on the board: lower side, see [figure 68](#), number [13](#).
- Purpose: to configure the signal at pin (09) of the EXT. 2 socket connector, see the following table.
- See also "[Configuration by Solder Jumpers](#)", [page 69](#).
- Configurations of solder jumper field B and solder jumper field C are independent from each other.
- On the RTC6 Ethernet Board the printed label of solder jumper field B is 'Pin 17'. This has been chosen deliberately in order to keep consistency with already existing RTC boards. Even though pin (09) is actually configured.

Allowed jumper setting	Output at the EXT. 2 socket connector pin (09)
 open open open open	No signal.
 closed* open open open	LATCH signal.
 open closed* open open	GROUND (low level).

\*Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

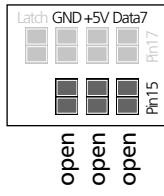
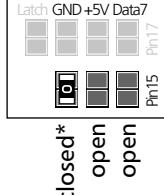
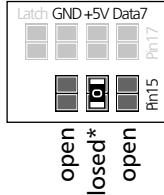
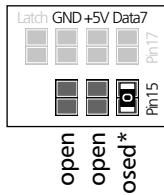
Allowed jumper setting (cont'd)	Output at the EXT. 2 socket connector pin (09) (cont'd)
 open open closed* open	+5 V (high level).
 open open open closed*	DATA7 <sup>(a)</sup> .

\*Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

(a) Synonym: Data Bit #7. **MSB** of the 8-bit output value.

### Solder Jumper Field C – Configuring pin (08) of the EXT. 2 Socket Connector

- Position on the board: lower side, see [figure 68](#), number **14**.
- Purpose: to configure the signal at pin (08) of the EXT. 2 socket connector, see the following table.
- See also "[Configuration by Solder Jumpers](#)", [page 69](#).
- Configurations of solder jumper field C and solder jumper field B are independent from each other.
- On the RTC6 Ethernet Board the printed label of solder jumper field C is 'Pin 15'. This has been chosen deliberately in order to keep consistency with already existing RTC boards. Even though pin (08) is actually configured.

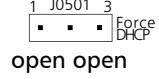
Allowed jumper setting	Output at the EXT. 2 socket connector pin (08)
	No signal.
	GROUND (low level).
	+5 V (high level).
	DATA7 <sup>(a)</sup> .

\* Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

(a) Synonym: Data Bit #7. **MSB** of the 8-bit output value.

### Jumper Field 'Force DHCP'

- Position on the board: lower side, see [figure 67](#), number [11](#).
- Purpose: see the following table.

Allowed jumper setting	Effect
 open closed	<p>When the RTC6 Ethernet Board is switched on, an IP address is to be obtained by DHCP (= "Force DHCP position")<sup>(a)</sup>. For all configurable network parameters (static IP, pertaining net mask, gateway, UDP and TCP ports), the default settings are used (instead the ones which are saved on the board).</p>
 closed open*	<p>If user-defined network settings are saved, these are used when switching on the RTC6 Ethernet Board.</p>
 open open	<p>Same as "open closed".</p>

\*Position as delivered by the factory.

(a) See also [page 850](#).

### 15.2.15 Real-Time Clock

The RTC6 Ethernet Board features a real-time clock. If the board is without power, the clock continues to run for about > 1 week. With a [Hardware reset](#), this time is automatically adopted (no [time\\_update](#) required).

By [time\\_update](#), the real-time clock is automatically synchronized with the PC time. The accuracy can be fine-tuned by [time\\_control\\_eth](#).

## 15.3 Installation and Operation

### 15.3.1 Hardware Installation

#### Notice!

- Store the board in an electrostatically neutral environment using the supplied anti-static bag.
- Observe ESD precautions when installing the board.
- Do not touch the electrical contacts of the board.
- Protect the board from moisture, dust, corrosive vapors and mechanical loads.

Proceed as follows to install the RTC6 Ethernet Board in a shielded housing:

- (1) Remove the RTC6 Ethernet Board from its anti-static bag. Do not touch the electrical contacts of the board.
- (2) Mount the RTC6 Ethernet Board at the intended location in your shielded housing.
- (3) Connect the RTC6 Ethernet Board to:
  - your power supply using a power cable
  - your Ethernet port using an Ethernet cable
- (4) Connect the RTC6 Ethernet Board using appropriate cables to:
  - the scan head
  - the laser
- (5) If you want to use the signals at the RTC6 Ethernet Board socket connectors, then attach appropriate cables.

### 15.3.2 Software Installation

For installing and starting-up an RTC6 Ethernet Board and its software proceed analogously as with RTC6 PCIe Boards, [Chapter 5 "Installation and Start-Up", page 78](#). However, step 3, [page 78](#) does not apply:

RTC6 Ethernet Boards *do not require the RTC6 board driver*.

### 15.3.3 Connecting to a Network

Connect the RTC6 Ethernet Board with the desired network segment and switch on the supply voltage. The green LED at the ETH connector (see [Chapter 15.2.7 "ETH Connector", page 839](#)) should now light up and possibly blink.

If no static IP address is stored on the RTC6 Ethernet Board, a DHCP server is necessary to obtain a dynamic IP address. In this case the yellow LED should light up after a few seconds. This indicates a successful receipt of an IP address.

If in "Force DHCP position"<sup>(1)</sup> (as of [BIOS 24](#)) no IP address has been assigned within 60 seconds after Power-On, then the link-local address 169.254.1.0/16 is automatically used.

If a static IP address is stored on the RTC6 Ethernet Board, the yellow LED immediately lights up after switching on the RTC6 Ethernet Board.

Important: it is possible that the IP configuration stored on the RTC6 Ethernet Board is invalid for the connected network segment. In this case the yellow LED lights up as well. However, the RTC6 Ethernet Board is unresponsive in the network.

(1) See [page 849](#).



## 15.4 Notes on Migrating Existing and Programming New RTC6 User Programs

RTC6 Ethernet Boards must be registered manually to the RTC6 board management, see [Chapter 15.5.3 "About the RTC6 Board Management", page 856](#) (`init_rtc6_dll` only searches RTC6 PCIe Boards).

After that, RTC6 Ethernet Boards can be used generally in the same way as RTC6 PCIe Boards.

In principle, communication via Ethernet is more unsteady compared to a PCIe bus connection within a PC. Therefore, further attention should be paid to potentially occurring errors and to its handling than with RTC6 PCIe Boards (see `get_last_error`).

### 15.4.1 Finding RTC6 Ethernet Boards in the Network and Querying their Properties

**Case 1: IP address of the RTC6 Ethernet Board is known**

```
card_no = eth_assign_card_ip(ip, 0); // 0: assign to first free number
Result = select_rtc(card_no);           // If Result equals CardNo: Success
...
...
```

**Case 2: IP address of the RTC6 Ethernet Board is not known**

```
result = eth_search_cards(eth_convert_string_to_ip("192.168.250.1"),
                         eth_convert_string_to_ip("255.255.255.0"));
if (result != 0)
{
    card_no = eth_assign_card(1, 0); // take first found card, assign to first free number
    result = select_rtc(card_no);
    if (result == card_no)
    {
        ip = eth_get_ip(); // now IP address is known
        ... // do anything
    }
}
```



### **15.4.2 Example Code (C++): Initialization Covering**

#### **RTC6 Ethernet Boards**

```

// Abstract
//      A console application to demonstrate how to initialize RTC6 boards
//
// Comment
// This application demonstrates how to properly initialize RTC6 boards.
// It will enumerate all available PCIe boards as well as search for Ethernet boards in a given subnet.
//
// Platform
//      Win32 - 32-bit Windows
//      x64   - 64-bit Windows
//
// Necessary Files
//      RTC6impl.h
//      RTC6DLL.dll for Win32 or RTC6DLLx64.dll for x64
//      RTC6DLL.lib for Win32 or RTC6DLLx64.lib for x64
//      RTC6DAT.dat
//      RTC6OUT.out for PCIe boards and/or RTC6ETH.out for Ethernet boards
//      RTC6RBF.rbf
//      Cor_1to1.ct5 or any other correction file
//
// Compiler
//      - tested with Microsoft C++ Compiler 19.00.24215.1.

#include "RTC6impl.hpp"

#include <iostream>
#include <array>
#include <string>
#include <conio.h>

using namespace std;

// RTC error codes
const UINT ERROR_NO_ERROR = 0U;
const UINT ERROR_NO_CARD = 1U;
const UINT ERROR_VERSION_MISMATCH = 256U;

// Use current working directory as path
const char* PROGRAM_FILE_PATH = nullptr;
// Use Cor_1to1.ct5 in current working directory as correction file
const char* CORRECTION_FILE_PATH = "./Cor_1to1.ct5";

// Subnet to use for Ethernet board search
const char* ETH_SEARCH_IP = "192.168.250.253";
const char* ETH_SEARCH_NETMASK = "255.255.255.0";

bool LoadProgramAndCorrectionFile(UINT card)
{
    // DAT, ETH/OUT and RBF need to be in the current working directory
    const auto loadProgram = n_load_program_file(card, PROGRAM_FILE_PATH);
    if (loadProgram != ERROR_NO_ERROR)
    {
        cout << "n load_program_file for card " << card << " failed with error code: " << loadProgram << endl;
        return false;
    }

    // Acquire board for further access
    const auto acquire = acquire_rtc(card);
    if (acquire != card)
    {
        cout << "acquire_rtc for card " << card << " failed with error code: " << acquire << endl;
        return false;
    }
}

```



```
// Load 2D correction file as table 1
const auto loadCorrection = n_load_correction_file(card, CORRECTION_FILE_PATH, 1, 2);
if (loadCorrection != ERROR_NO_ERROR)
{
    cout << "n_load_correction_file for card " << card << " failed with error code: " << loadCorrection << endl;
    return false;
}

// select_cor_table( 1, 0 ); is done internally. Call select_cor_table, if you want a different setting.

const auto serialNumber = n_get_serial_number(card);
cout << "Initialized card " << card << " (SN " << serialNumber << ")" << endl;

return true;
}

bool InitializeRTC()
{
    // Initialize DLL
    const auto initDLL = init_rtc6_dll();
    if (initDLL != ERROR_NO_ERROR)
    {
        if (initDLL & ERROR_VERSION_MISMATCH)
        {
            // Version mismatch can happen if a board has been previously used with a different software version.
            // This error can be fixed by loading the current program file on to the board.
        }
        else if (initDLL & ERROR_NO_CARD)
        {
            // The DLL will return ERROR_NO_CARD if no PCIe board has been found.
            // We can still use Ethernet boards if available.
        }
        else
        {
            cout << "init_rtc6_dll failed with error code: " << initDLL << endl;
            return false;
        }
    }

    // Initialize PCIe boards
    const auto foundCards = rtc6_count_cards();
    for (auto card = 1; card <= foundCards; card++)
    {
        if (!LoadProgramAndCorrectionFile(card))
        {
            return false;
        }
    }
}
```



```
// Search for and initialize Ethernet boards
const auto ethSearchIP = eth_convert_string_to_ip(ETH_SEARCH_IP);
const auto ethSearchNetmask = eth_convert_string_to_ip(ETH_SEARCH_NETMASK);
const auto foundEthCards = eth_search_cards(ethSearchIP, ethSearchNetmask);
if (foundEthCards > 0)
{
    for (auto searchCard = 1; searchCard <= foundEthCards; searchCard++)
    {
        array<UINT, 16> cardInfo;
        eth_get_card_info_search(searchCard, (UINT)cardInfo.data());
        const auto serialNumber = cardInfo[1];
        // Param 0 automatically assigns the board to the next free index
        const auto card = eth_assign_card(searchCard, 0);
        if (card <= 0)
        {
            cout << "eth_assign_card for SN " << serialNumber << " failed with error code: " << card;
            return false;
        }
        if (!LoadProgramAndCorrectionFile(card))
        {
            return false;
        }
    }
}
// Atleast one PCIe or Ethernet board available
return (foundCards > 0) || (foundEthCards > 0);
}

int main(int argc, char* argv[])
{
    if (!InitializeRTC())
    {
        return 1;
    }
    // Use boards for marking etc...
    _getch();
    return 0;
}
```

## 15.5 RTC6 Ethernet Board **Commands and Functions**

### 15.5.1 Notes on Working with IP Addresses

With the RTC6 commands, all IP addresses (always IPv4, IPv6 is not supported) are specified as decimal values in Big Endian format ("Big Endian byte order").

For example, the IP address "192.168.250.1" must be specified as "33204416", see following table.

Format	IP address	Hex value	Decimal value
Little Endian	192.168.250.1 (a)	0xC0A8FA01	3232299521
Big Endian	1.250.168.192	0x01FAA8C0	33204416

(a) Usual dotted decimal notation.

**eth\_convert\_ip\_to\_string** converts the IP address in Big Endian byte order to usual dotted decimal notation.

**eth\_convert\_string\_to\_ip** converts the IP address in usual dotted decimal notation to Big Endian byte order.

### 15.5.2 About Searching

#### RTC6 Ethernet Boards

To search for all RTC6 Ethernet Boards available in the network, **eth\_search\_cards** can be used. If the search is to be limited to a certain address range only, then **eth\_search\_cards\_range** is to be used.

The data of all RTC6 Ethernet Boards which have been answered within a configurable timeout (see **eth\_set\_search\_cards\_timeout**) are registered to the search result list (see below).

By a card search, RTC6 Ethernet Boards with unknown IP address can be identified in the network (for example, because they have received it dynamically by a DHCP server).

The found RTC6 Ethernet Boards are registered in a temporary list which is the search result list.

Index	Record in the search result list
1	Information <sup>(a)</sup> on RTC6 Ethernet Board 1
2	Information <sup>(a)</sup> on RTC6 Ethernet Board 2
n	Information <sup>(a)</sup> on RTC6 Ethernet Board n

(a) IP address, serial number, connection status, etc., see **eth\_get\_card\_info\_search**.

For a specified search result list index **eth\_get\_card\_info\_search** returns the available information on the RTC6 Ethernet Board, whereas **eth\_get\_ip\_search** returns only the IP address and **eth\_get\_serial\_search** only the serial number.

On the one hand the number of found RTC6 Ethernet Boards is already returned by **eth\_search\_cards**. On the other hand **eth\_found\_cards** also returns it at any time as well (without the need to perform the search in the network again).

Several subsequent calls of **eth\_search\_cards** can deliver different results, depending how many RTC6 Ethernet Boards are available in the network and in which chronological order the answers come in.

### 15.5.3 About the RTC6 Board Management

RTC6 boards are addressed by a unique number under which they must be entered in the RTC6 board management.

The RTC6 board management is an **RTC6 DLL<sup>(1)</sup>**-internal list consisting of 255 possible RTC6 board records, see following table.

Index	RTC6 board record
1	RTC6 PCIe Board 1
2	RTC6 PCIe Board 2
3	RTC6 PCIe Board 3
4	"No card"
5	Information on RTC6 Ethernet Board <sup>(a)</sup>
6	Information on RTC6 Ethernet Board <sup>(a)</sup>
7	"No card"
...	...
42	Information on RTC6 Ethernet Board <sup>(a)</sup>
43	"No card"
...	...
255	"No card"

(a) IP address, serial number, connection status, etc., see [eth\\_get\\_card\\_info\\_search](#).

The list starts with RTC6 PCIe Boards, if any are present in the PC.

RTC6 PCIe Boards are automatically searched by **init\_rtc6\_dll** and consecutively numbered. The numbering cannot be changed. **rtc6\_count\_cards** only returns the number of RTC6 PCIe Boards found.

RTC6 Ethernet Board, on the other hand, must be entered manually into the RTC6 board management:

- If the IP address is known, **eth\_assign\_card\_ip** can be used to enter a card at any index between **rtc6\_count\_cards** + 1...255.
- If the IP address is unknown, by **eth\_assign\_card** an RTC6 Ethernet Board from the search result list, see [Chapter 15.5.2 "About Searching RTC6 Ethernet Boards", page 855](#), can be entered at an arbitrary index. However, neither an RTC6 PCIe Board nor an RTC6 Ethernet Board must be registered at this index

**eth\_max\_card** returns the highest index where an RTC6 Ethernet Board in the RTC6 board management is registered.

In contrast, **eth\_count\_cards** returns the total number of entered RTC6 Ethernet Boards.

By **get\_card\_type** the registered board type can be queried:

- 0 = "No card"
- 1 = RTC6 PCIe Board
- 2 = RTC6 Ethernet Board

(1) [RTC6DLL.dll](#), [RTC6DLLx64.dll](#).



#### 15.5.4 Checking the Connection to the RTC6 Ethernet Board

In general, communication via Ethernet is more unreliable than PCIe bus connections within PCs (simple example: Ethernet cable is not plugged in).

By `eth_check_connection` it can be simply checked, whether the RTC6 Ethernet Board responds and therefore, the Ethernet connection still exists.

#### 15.5.5 Command Set for the RTC6 Ethernet Board

- See [Chapter "Control Commands for RTC6 Ethernet Boards", page 295.](#)

## 15.6 Safe Startup and Shutdown Sequences

To assure safety during startup, switch on the components of the laser system in the following order:

- (1) Switch on the network PC.
- (2) Switch on the power supply for the RTC6 Ethernet Board.
- (3) Start the control software.
- (4) Switch on any required peripheral devices.
- (5) Switch on the power supply for the scan system.
- (6) Switch on the laser.

To assure safety during shutdown, switch off the components of the laser system in exactly the reverse order:

- (1) Switch off the laser.
- (2) Switch off the power supply for the scan system.
- (3) Switch off the peripheral devices.
- (4) Terminate the control software.
- (5) Switch off the power supply for the RTC6 Ethernet Board.
- (6) Shut down the network PC.



### Caution!

- When the PC switches on or off, the RTC6 Ethernet Board board output levels might briefly fluctuate, resulting in unintended changes to laser control signals. The above-mentioned startup and shutdown sequences must therefore be strictly followed. Otherwise, the laser might briefly, unexpectedly and dangerously switch on.
- Always start up the PC and control software prior to turning on the scan system. And switch the scan system back off prior to shutting down the control software and PC. Otherwise, unintended scan system motions might occur. The laser must always be switched on last and switched off first. Otherwise, there is the risk that the laser beam might be deflected in an arbitrary direction.

## 15.7 Standalone Functionality

- In contrast to PC operation, the aim of the **Standalone Operation Mode** is that the RTC6 Ethernet Board operates independently without a connected PC. Among other things, for this purpose, the list commands to be processed, all correction tables to be used and the control commands to be executed during automatic booting<sup>(1)</sup> must be stored in the **NAND memory**.

### Notice!

- The following requirements must be met for **Standalone Operation Mode**:
  - (1) At least **BIOS** version 0x26 (**RTC6BIOSETH\_26**) is installed<sup>(a)</sup> on the RTC6 Ethernet Board. See [Chapter 15.7.1 "Upgrading to RTC6BIOSETH\\_26", page 860](#).
  - (2) The user program uses **RTC6 Software Package**  $\geq$  V1.7.0, that is, a combination of
    - $\geq$  DLL 618
    - $\geq$  ETH 618
    - $\geq$  RBF 623
    - $\geq$  DAT 603

(a) See [RTC6conf.exe](#) and [RTC6BIOSETH\\_26.out](#).

- Then the RTC6 Ethernet Board can be configured to boot automatically<sup>(1)</sup> (depending on the data stored in **NAND memory** by **store\_program( Mode )** after a **Hardware reset**.

After that, it is in one of the following states:

- “Normal PC Operation State”
- “Standalone Basic State”
- “Standalone Full State”

- The **“Normal PC Operation State”** is achieved by:

- **BIOS**  $<$  **RTC6BIOSETH\_26**
- $\geq$  **RTC6BIOSETH\_26** and **set\_eth\_boot\_control( 0 )**
- $\geq$  **RTC6BIOSETH\_26** and **store\_program( 1 )**

In these cases, **load\_program\_file** must be called for further operation.

- The **“Standalone Basic State”** is achieved by:
  - **BIOS**  $\geq$  **RTC6BIOSETH\_26**, **store\_program( 0 )** and **set\_eth\_boot\_control( 1 )**
- A **load\_program\_file** call is not required. See also [Chapter 15.7.4 “Boot Image”, page 862](#). The procedure for **“Standalone Basic State”** is described in [Chapter 15.7.2 “Preparing the “Standalone Basic State””, page 860](#).
- The **“Standalone Full State”** is achieved by:
  - **BIOS**  $\geq$  **RTC6BIOSETH\_26**, **store\_program( 2 )** and **set\_eth\_boot\_control( 1 )**
- Compared to the **“Standalone Basic State”** it is also no longer necessary to load correction files. The same applies to loading of the list commands which have been stored upon **store\_program( 2 )**. Furthermore, control commands for configuration<sup>(2)</sup> can be executed automatically as required. See also [Chapter 15.7.4 “Boot Image”, page 862](#). The procedure for **“Standalone Full State”** is described in [Chapter 15.7.3 “Preparing the “Standalone Full State””, page 861](#). In **“Standalone Full State”**, the RTC6 Ethernet Board can process its list independently and without a connected PC after an /START.
- The **NAND memory** contents can be written to the PC as a so-called **“Boot Image”** by **read\_image\_eth** as a binary file and copied to any number of RTC6 Ethernet Boards by **write\_image\_eth**, see [Chapter 15.7.4 “Boot Image”, page 862](#).
- Details on automatic booting can be found in [Chapter 15.7.6 “Automatic Booting - Process in Detail”, page 865](#).

(1) In this chapter, “automatic booting” means that in addition to the actual booting, additional data is read out from the **NAND memory**.

(2) See [Chapter 15.7.5 “Control Commands Allowed for Automatic Booting”, page 863](#).

### 15.7.1 Upgrading to RTC6BIOSETH\_26

Proceed as follows to upgrade the **BIOS** of an RTC6 Ethernet Board with  $\leq$  RTC6BIOSETH\_26 only (no subsequent **Standalone Operation Mode**):

- (1) Carry out a **Hardware reset**.
- (2) Call **load\_program\_file** by specifying the relevant data from RTC6 Software Package  $\geq$  V1.7.0.
- (3) Call **store\_program( Mode = 1 )**.  
The **NAND memory** content is erased.
- (4) Call **set\_eth\_boot\_control( 0 )**.  
Thus, the board does *not* boot automatically<sup>(1)</sup> after a **Hardware reset** (no matter what is stored in **NAND memory**).
- (5) Run **RTC6conf.exe** and install **RTC6BIOSETH\_26.out** via **Upgrade BIOS** button.  
Thus, RTC6BIOSETH\_26 is installed.
- (6) Carry out a **Hardware reset**.
  - The RTC6 Ethernet Board *does not* boot automatically<sup>(1)</sup>
  - Is now ready for PC operation

### 15.7.2 Preparing the "Standalone Basic State"

Proceed as follows to put the board to **"Standalone Basic State"**:

- (1) Prerequisite: Chapter 15.7.1 "Upgrading to RTC6BIOSETH\_26", page 860 has been carried out.
- (2) Call **load\_program\_file** by specifying the relevant data from RTC6 Software Package  $\geq$  V1.7.0.
- (3) Call **store\_program( Mode = 0 )**.  
Data for the **"Standalone Basic State"**, see page 758, is saved to the **NAND memory**.
- (4) Call **set\_eth\_boot\_control( Ctrl = 1 )**.  
Thus, the board boots automatically after a **Hardware reset**<sup>(1)(2)</sup>.
- (5) Carry out a **Hardware reset**.
  - The RTC6 Ethernet Board boots automatically<sup>(1)</sup>
  - Is now ready for PC operation
  - Subsequently, you do *not* need to call **load\_program\_file**

#### Notes

- The actual **RTC6 files** no longer need to be provided in the user program. An accidentally wrong **RTC6 DLL** version is automatically detected, because a wrong **RTC6 DLL** leads to a **get\_last\_error** return code **RTC6\_VERSION\_MISMATCH** (whereas a **load\_program\_file** does not, as long as **RTC6 files** are compatible with the **RTC6 DLL**).

(1) See Footnote, page 859.

(2) This is also the default setting with "new" (= **set\_eth\_boot\_control( Ctrl = 0 )** has never been executed) RTC6 Ethernet Boards.

### 15.7.3 Preparing the "Standalone Full State"

Proceed as follows to put the board to "Standalone Full State":

- (1) Prerequisite: Chapter 15.7.2 "Preparing the "Standalone Basic State""", page 860 has been carried out.
- (2) Verify that the RTC6 Ethernet Board works properly in PC operation.
- (3) Work out a suitable sequence of control commands for automatic booting<sup>(1)</sup>.
  - The allowed control commands are listed in Chapter 15.7.5 "Control Commands Allowed for Automatic Booting", page 863, separated according to **Boot Phase 1** and **Boot Phase 2**.
  - Call **eth\_boot\_dcmsg** before each of these commands.
  - Pay attention to the correct order, for example,
    - **config\_list** should be called before loading list commands
    - if **set\_scanahead\_params** is used, a scan system of the excelliSCAN series must already be connected and switched on
    - if an "Automatic Laser Control" is used, it may only be activated after **set\_scanahead\_params**
- (4) Test the result of step 3:  
execute it in PC operation in exactly the same way as it is to be executed later in **Standalone Operation Mode**.
- (5) Call **load\_program\_file** in order to initialize the RTC6 Ethernet Board.
- (6) Load all required control commands, list commands and correction files.

- (7) Emergency provision in case an error occurs during saving: call **set\_eth\_boot\_control( 0 )**. This prevents automatic booting.
- (8) Call **store\_program( Mode = 2 )**. The process can take up to 2 minutes.
- (9) If an error occurred: call **store\_program( Mode = 2 )** again.
- (10) Call **set\_eth\_boot\_control( Ctrl = 1 )**. This activates automatic booting.
- (11) Carry out a **Hardware reset**.
  - The RTC6 Ethernet Board is now in "Standalone Full State"
  - See also Chapter 15.7.6 "Automatic Booting - Process in Detail", page 865

#### Notes

- In "Standalone Full State" the RTC6 Ethernet Board is already configured with the control commands for automatic booting<sup>(2)</sup> and provided with all correction tables. If list commands have been stored, they are ready to be executed automatically after an /START.

(1) See Footnote, page 859.

(2) See Chapter 15.7.5 "Control Commands Allowed for Automatic Booting", page 863.

### 15.7.4 Boot Image

- Prerequisites: see [page 859](#).

The data in the **NAND memory** can be written to the PC as “boot image” and copied from there to any RTC6 Ethernet Boards (for example, of several production machines):

- **`read_image_eth`**,  
see [Creating a Boot Image on the PC](#)
- **`write_image_eth`**,  
see [Copying Boot Image to Board\(s\)](#)

The properties of boot images are shown in Table 7 (RTC6 Software Package V1.7.0, approximate values):

Table 7: Properties of Boot Images

Boot Image	File size	Reading duration <sup>(a)</sup>	Writing duration <sup>(b)</sup>
for “Standalone Basic State”	2 MB	a few seconds	a few seconds
for “Standalone Full State”	160 MB ... 264 MB	up to 2 minutes	up to 2 minutes

(a) With `read_image_eth`.

(b) With `write_image_eth`.

#### Creating a Boot Image on the PC

- Prerequisites: see [page 859](#).

`read_image_eth( Name )` read out the contents of the **NAND memory** and writes it in binary to the “`Name`” file on the PC. The user program must have write permission for this file. For reading duration, see Table 7.

See also Section “Procedure after an Transmission Abortion”, page 862.

#### Copying Boot Image to Board(s)

- Prerequisites: see [page 859](#).

`write_image_eth( Name )` reads the file “`Name`” and writes its content to the **NAND memory**. The user program must have read permission for this file. For writing duration, see Table 7.

- (1) Call `set_eth_boot_control( 0 )`.  
Thus, the board does not boot automatically after a **Hardware reset** (recommended as automatic booting might be faulty).
  - (2) Call `write_image_eth( Name )` and do not interrupt the process! Otherwise, observe Section “Procedure after an Transmission Abortion”, page 862.
  - (3) If the process went without errors, call `set_eth_boot_control( 1 )`. Otherwise, repeat step 2.
  - (4) Carry out a **Hardware reset**.  
Provided the same options are enabled: This RTC6 Ethernet Board boots exactly the same way as the RTC6 Ethernet Board from which the boot image has been taken.
- Procedure after an Transmission Abortion**
- If the transmission is aborted, for example, due to an Ethernet connection interruption, the RTC6 Ethernet Board probably remains in an **INTERNAL-BUSY** state. The aborted process is not continued, even if the connection is re-established:
- (1) Call either `stop_execution` or execute an /STOP in order to release the board from this **INTERNAL-BUSY** state.
  - (2) Call `read_image_eth( Name )` (once again).  
Thus, the board does not boot automatically after a **Hardware reset**.
  - (3) Carry out a **Hardware reset**.
  - (4) Call `load_program_file`.
  - (5) If you
    - create a boot image on the PC:  
call `read_image_eth( Name )` again.
    - copy a boot image to the board:  
call `write_image_eth( Name )`.



### 15.7.5 Control Commands Allowed for Automatic Booting

- Only certain control commands are allowed in each boot phase, see table 8 and [Chapter 15.7.6 "Automatic Booting - Process in Detail", page 865](#):
  - [Boot Phase 1](#)
  - [Boot Phase 2](#)
- They meet one or more of the following criteria:
  - Requires no communication with the PC
  - Does not send a response to the PC
  - There is no corresponding list command
- Exceptions are, for example,
  - [set\\_free\\_variable](#), see [Comments](#) there
  - [set\\_auto\\_laser\\_control](#) or [set\\_scanahead\\_params](#)
    - They check the connected scan systems at the time of the call
    - They save exclusively the parameters to be finally set only in case of success
  - [set\\_jump\\_mode](#) only sets the tuning numbers [VA1...JB2](#) during automatic booting and does not check whether they match the scan system.
  - The table containing the jump delay values [[JumpTable<No>](#)], see [page 205](#) and [load\\_jump\\_table\\_offset](#), must have been previously saved via [create\\_dat\\_file](#).
- Upon (later) automatic booting, the connected scan systems are *not* checked anymore!
- [Boot Phase 2](#)-control commands are control commands which:
  - Necessarily need to be executed after a [Boot Phase 1](#)-control command
  - Require functioning peripherals, for example, an [iDRIVE](#) scan system<sup>(1)</sup> with "Automatic Laser Control". Important: users must make sure themselves that the periphery is actually working at runtime

(1) See glossary entry on [page 879](#).



Table 8: Control commands allowed for automatic booting (alphabetically)

Boot Phase 1	Boot Phase 2
bounce_supp	control_command
config_laser_signals	init_fly_2d
config_list	select_cor_table
home_position	set_auto_laser_control
home_position_xyz	set_dsp_mode
mcbsp_init	set_hi
set_control_mode	start_loop
set_eth_boot_timeout	
set_extstartpos	
set_free_variable	
set_jump_mode	
set_laser_control	
set_laser_mode	
set_matrix( HeadNo = 4 )	
set_max_counts	
set_mcbsp_freq	
set_mcbsp_out_ptr	
set_offset_xyz( HeadNo = 4 )	
set_pulse_picking_length	
set_rot_center	
set_scanahead_params	
set_scanahead_speed_control	
set_timelag_compensation	
simulate_encoder	
uart_config	

### 15.7.6 Automatic Booting - Process in Detail

- (1) After a **Hardware reset**, the RTC6 Ethernet Board tries to read data for automatic booting from the **NAND memory**. During this time the yellow LED flashes dimmed with approx. 1 Hz until the end of the initialization.
- Case 1: There are *no* data present (after `store_program( Mode= 1 )`) or `set_eth_boot_control( 0 )` has been carried out.  
The initialization is completed without these data. Afterwards, the RTC6 Ethernet Board is in "Normal PC Operation State", page 859.
  - Case 2: There are only data for "Standalone Basic State", see [page 758](#) (after `store_program( Mode= 0 )`). The RTC6 Ethernet Board is initialized (as by `load_program_file`). Afterwards, it is immediately ready for PC operation = "Standalone Basic State".
  - Case 3: There are also data for "Standalone Full State", see [page 758](#) (after `store_program( Mode= 2 )`). Then step 2 is executed.
- (2) The RTC6 Ethernet Board
- switches on the **BUSY pin**
  - reads the "remaining" data for automatic booting to "Standalone Full State", see [page 758](#) from the **NAND memory**.
  - The subsequent boot steps 3...6 are two-phase.
- (3) In **Boot Phase 1** (duration: some seconds):
- The correction file(s) are read out
  - **Boot Phase 1**-control commands are executed (in the order in which they have been saved). If such commands are stored more than once, they are executed correspondingly often, with newer data overwriting older data.

- (4) After **Boot Phase 1**:
- Initialization ceases
  - The **BUSY pin** is switched off
  - Do not switch the laser on yet!
  - Switch on scan systems and other peripheral devices
  - Make sure peripheral devices are ready for operation. Only then, trigger **Boot Phase 2** by an /START
  - If your peripheral devices are always ready for operation after a certain time, you can specify a waiting time (`TimeOut`) with the **Boot Phase 1**-control command `set_eth_boot_timeout`. After its expiry, initialization continues automatically (= without an explicit /START). With an /START within the waiting time, **Boot Phase 2** starts immediately. Outside the waiting time, an /START is ignored as long as **Boot Phase 2** is still running (see **BUSY pin**!). The /START is automatically enabled (similar to `set_control_mode( Bit #0 = 1 )`). Further /STARTs after an /STOP must be enabled by users themselves by `set_control_mode( Bit #3 = 1 )`.
- (5) In **Boot Phase 2**:
- The **BUSY pin** is switched on
  - The **RTC6 List Memory** is read out (duration: approx. 1 minute)
  - **Boot Phase 2**-control commands are executed.
  - Initialization ceases
  - The **BUSY pin** is switched off
- (6) After **Boot Phase 2**:
- If an error has occurred (further operation of the board may not be possible), the LED continues to flash at about 4 Hz
  - When the initialization is successfully completed, the dimmed LED flashing stops. The LED returns to its normal state, see [Chapter 15.2.7 "ETH Connector", page 839](#)
  - At this point, it would be a suitable time to also switch on the laser (observe the notices on laser safety in [Chapter 3.2 "Laser Safety", page 52](#)) as well as other peripherals.
  - The RTC6 Ethernet Board waits endlessly for an /START, which starts the processing of the list
  - The RTC6 Ethernet Board is ready for PC operation as well



## 15.8 Technical Specifications for the RTC6 Ethernet Board

### System Requirements

Windows PC with Ethernet interface  
(Gigabit Ethernet (preferred), 10/100 Mbit/s Ethernet)

Operating system Microsoft Windows 10, 8, 7 as 32 bit or 64 bit version.

### Dimensions (without plugged-in plug connectors)

Length	120 mm
Width	103 mm
Height	Approx. 30 mm

### Other Connections and Specifications

Voltage supply range +11 V...+50 V

Maximum power consumption, no peripherals attached

In order to minimize power loss, a low input voltage should be selected.

### Operating temperature

- with natural convection 10 °C...50 °C
- with forced convection 10 °C...60 °C

The user must ensure sufficient cooling of the RTC6 Ethernet Board.

Standalone functionality Yes, see [Chapter 15.7 "Standalone Functionality", page 859](#).

### Interface to the Network

Ethernet	Gigabit Ethernet (preferred), 10/100 Mbit/s Ethernet
IP address obtaining	<ul style="list-style-type: none"> <li>• By a DHCP server</li> <li>• Link-local address (as of <a href="#">BIOS 24</a>)<sup>(a)</sup></li> <li>• Static IP address (storable on the board)</li> </ul>
Required local ports <sup>(b)</sup>	<ul style="list-style-type: none"> <li>• 63749 (UDP)</li> <li>• 63750 (UDP, TCP)</li> </ul>

(a) See [page 850](#).

(b) Configurable by [eth\\_set\\_port\\_numbers](#).

### Scan System Control

Number of list buffer Up to 3, configurable areas

Total capacity of the list 8,388,608 list positions buffer

Output interval of 10 µs microsteps

Maximum value range -524,288...+524,287 for the image field (20 bit, signed) coordinates

Virtual image field, e.g. -268,435,456... for +268,435,455 Processing-on-the-fly (28 bit, signed = 29 bit)<sup>(a)</sup>

(a) With RTC6 Software Package ≥ 1.4.0.



## Interfaces to Scan Systems

- SCANHEADs socket connector

Connector	20 pin socket connector with 2.54 mm pitch of the pins <sup>(a)</sup> . The signals for the first scan head are outputted at pin (01)...pin (10). The signals for the second scan head are outputted at pin (11)...pin (20): <ul style="list-style-type: none"> <li>• xy output values only with enabled Option "Second Scan Head Control"</li> <li>• z output values only with enabled Option "3D"</li> </ul>
Signal transmission	SL2-100 protocol. Via XY2-100 converter (accessory): XY2-100 protocol

(a) Not as with RTC6 PCIe Board.

## Interfaces to the Laser and Peripherals

- LASER socket connector

Connector	16 pin socket connector with 2.54 mm pitch of the pins <sup>(a)</sup>
Laser output signals	LASER1, LASER2, LASERON
• TTL level	5 V, active-HIGH or active-LOW programmable
• Max. current load	20 mA
• Reference	GND - boards without Option "DC/DC Converter". GND2 - boards with Option "DC/DC Converter"
Analog output ports	ANALOG OUT1, ANALOG OUT2 <sup>(b)</sup>
• Output voltage range	0 V...10 V
• Resolution	12 Bit
• Max. current load	5 mA
• Reference	GND
Digital input ports	2 Bits
• LOW level	< 0.6 V
• HIGH level	> 2.3 V
• Max. input voltage range	-0.5 V...+5.5 V
• Input resistance (pull-up)	> 4.7 kΩ
• Reference	GND
Digital output port	2 Bits, buffered
• LOW level	< 0.55 V
• HIGH level	> 3.8 V
• Max. current load	20 mA
• Reference	GND

(a) Not as with RTC6 PCIe Board.

(b) With RTC5 boards and RTC6 PCIe Boards, the ANALOG OUT2 signal is also available at the MARKING ON THE FLY socket connector.



- **LASER socket connector (cont'd)**

Inputs for external start and stop signals      TTL active-LOW, internally connected to +3.3 V by pull-up resistors (4.7 kΩ)

- /START      Edge sensitive
- /STOP      Level sensitive

- Reference      GND

**Other signals**

- BUSY OUT      5 V, TTL active-HIGH, Max. 10 mA, Reference GND

- +5 V      Max. 100 mA

- Reference      GND

- **EXTENSION 1 socket connector**

Connector

40 pin socket connector with 2.00 mm pitch of the pins<sup>(a)</sup>.

Output signal level is configurable by solder jumper field A

Digital output port      16 Bits, buffered

- LOW level      < 0.4 V

- HIGH level      > 2.0 V (3.3 V or 5 V)
- Max. current load      ±8 mA

- Reference      GND

- LATCH-Signal      3.3 V or 5 V, TTL active-HIGH, 5 µs pulse, max. 10 mA

Digital input port      16 Bit, protected

- LOW level      < 0.5 V

- HIGH level      > 2.6 V...24 V

- Max. input voltage range      -0.5 V...+26 V

- Input resistance      > 10 kΩ

- Reference      GND

- SYNC-Signal      3.3 V or 5 V, TTL active-HIGH, square wave signal (5 µs pulse, 10 µs period), max. 10 mA

**Other signals**

- BUSY OUT

3.3 V or 5 V, TTL active-HIGH, max. 10 mA

- VCC      3.3 V or 5 V, max. 100 mA

- +5 V      Max. 100 mA

- Reference      GND

(a) With RTC6 PCIe Boards: 2.54 mm pitch of the pins.

- **EXT. 2 socket connector**

Connector	10 pin socket connector with 2.00 mm pitch of the pins. pin (09) <sup>(a)</sup> is configurable by solder jumper field B. pin (08) <sup>(b)</sup> is configurable by solder jumper field C
Digital output port	8 Bits, buffered <ul style="list-style-type: none"> <li>• LOW level &lt; 0.4 V</li> <li>• HIGH level &gt; 2.0 V</li> <li>• Max. current load <math>\pm 8</math> mA</li> <li>• Reference GND</li> <li>• LATCH-Signal 5 V, TTL active-HIGH, 5 <math>\mu</math>s pulse, max. 10 mA</li> </ul>
Laser output signals	None <sup>(c)(d)</sup>
Other signals	<ul style="list-style-type: none"> <li>• +5 V Max. 100 mA</li> <li>• Reference GND</li> </ul>

(a) pin (09) of RTC6 Ethernet Boards = pin (17) of RTC6 PCIe Boards.

(b) pin (08) of RTC6 Ethernet Boards = pin (15) of RTC6 PCIe Boards.

(c) Not as with RTC6 PCIe Board.

(d) With RTC6 PCIe Boards: LASER1 and LASER2.

- **MOF socket connector<sup>(a)</sup>**

Connector	14 pin socket connector with 2.00 mm pitch of the pins <sup>(a)</sup>
2 encoder input ports for incremental encoders	ENCODER X( $1\pm, 2\pm$ ) and ENCODER Y( $1\pm, 2\pm$ ), designed for a pair of standardized differential input signals (RS-422) each. HIGH level $\geq 2.0$ V LOW level $\leq 0.8$ V $f \leq 4$ MHz
Analog output port	None <sup>(b)(c)</sup>
Input ports for external start and stop signals	/START2, /STOP2 (see /START, /STOP of LASER socket connector, page 868)
Other signals	<ul style="list-style-type: none"> <li>• BUSY OUT Identical with BUSY OUT on LASER socket connector</li> <li>• +5 V Max. 100 mA</li> <li>• Reference GND</li> </ul>
	(a) With RTC6 PCIe Boards: 16 pin socket connector with 2.54 mm pitch of the pins. Furthermore, the printed label on the board is MARKING ON THE FLY.
	(b) Not as with RTC6 PCIe Board.
	(c) The RTC6 Ethernet Board has no pin for the ANALOG OUT2 signal on its MOF socket connector.



- **SPI/ANA/UART socket connector**

With RTC6 Ethernet Boards, the RS232 socket connector pins and McBSP/ANALOG socket connector pins of the RTC6 PCIe Board are integrated into the SPI/ANA/UART socket connector.

Connector	10 pin socket connector with 2.00 mm pitch of the pins.
Analog input ports	ANALOG IN0, ANALOG IN1
• Input voltage range	0 V...10 V
• Input impedance	> 5 kΩ <sup>(a)</sup>
• ADC resolution	12 Bit
• Reference	GND

(a) Not as with RTC6 PCIe Board.

- **SPI/ANA/UART socket connector (cont'd)**

Pins for RS-232

Input port	RxD
• Voltage range	Max. -25 V...+25 V
Output port	TxD
• Voltage range	Max. -13 V...+13 V
Reference	GND
Baud rate	300...115,200

Pins for McBSP, see also [Section "McBSP Interface", page 73](#)

• Transmitter signal level	3.3 V TTL
• Receiver signal level	3.3 V or 5 V TTL
• McBSP mode	Single Phase Frame Single Element per Frame 32 Bits per Element DataDelay N Bit
• Reference	GND

SPI interface functionality	No.
-----------------------------	-----

- **STEPPER socket connector**

Connector 10 pin socket connector with 2.00 mm pitch of the pins.

Signals for controlling up to two stepper motors:

• Output ports	5 V, TTL
ENABLE1,	
ENABLE2,	
DIRECTION1,	
DIRECTION2	
• Output ports	5 V, TTL active-HIGH, 5 µs pulse
CLOCK1,	
CLOCK2	
• Input ports	TTL active-LOW, internally connected to +3.3 V by pull-up resistors (10 kΩ)
SWITCH1,	
SWITCH2	
• Reference	GND



## **15.9 Compliance with EC Guidelines for Electromagnetic Compatibility (EMC)**

The RTC6 Ethernet Board has been determined to be in compliance with EC directive 2014/30/EU (electromagnetic compatibility).

For that purpose, an RTC6 Ethernet Board has been integrated to a PC and was tested together with an powerSCAN II scan head (with SL2-100 interface).

### **Test Specifications**

Evidence of fulfillment of the protection goals of EC directive 2014/30/EU (CE Conformity for EMC) based on

- EN 61000-6-2: 2005 + AC: 2005
- EN 61000-6-4: 2007+ A1: 2011

### **Result**

The devices under test fulfill the specifications.

## **15.10 Compliance with FCC Rules**

The RTC6 Ethernet Board has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules.

These limits are designed to provide reasonable protection against harmful interference when the RTC6 Ethernet Board is operated in a commercial environment. The RTC6 Ethernet Board generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with this instruction manual, may cause harmful interference to radio communications. Operation of the RTC6 Ethernet Board in a residential area is likely to cause harmful interference in which case the user is required to correct the interference at his own expense.

## 16 Appendix B: The UFP Ext Board

The UFP Ext Board<sup>(1)</sup> has been developed for RTC6 PCIe Boards<sup>(2)</sup>.

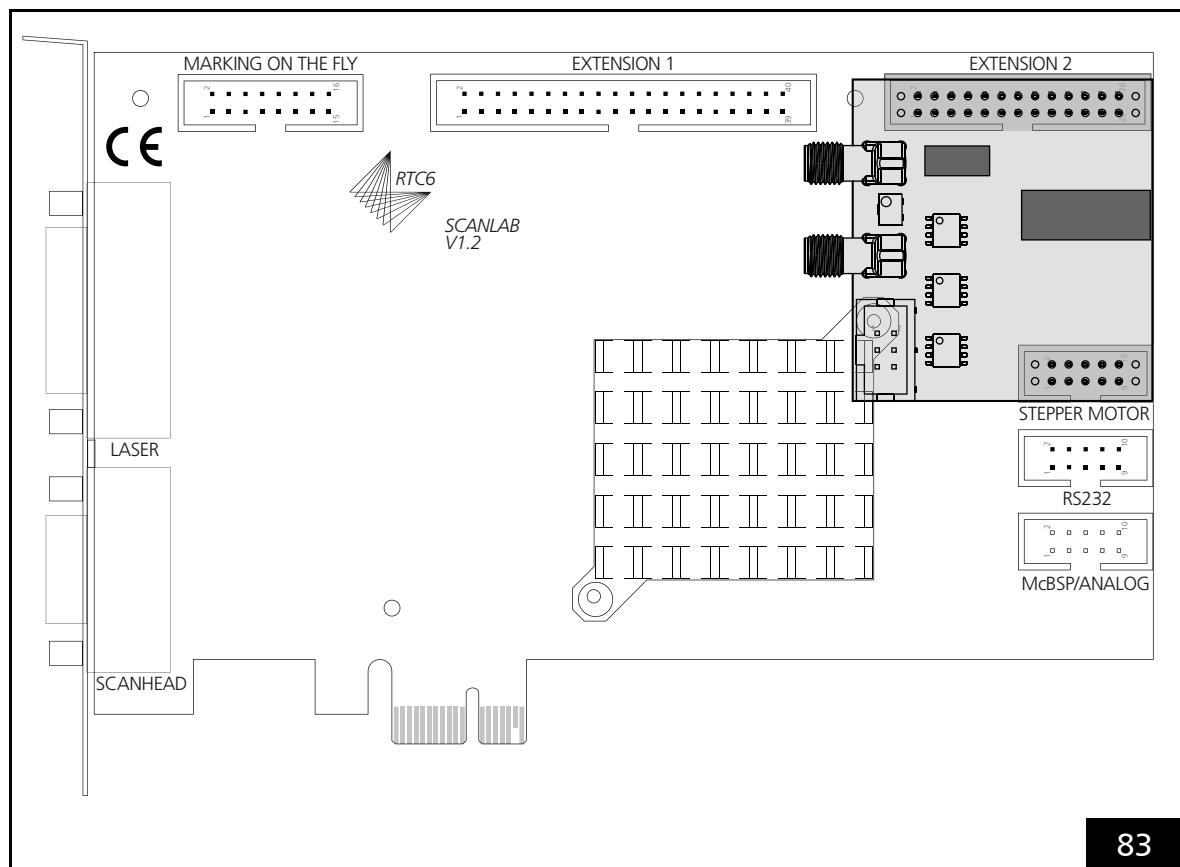
The assembly is shown in [figure 83](#), dimensions and details in [figure 84](#).

The UFP Ext Board converts 8-bit digital signals into analog voltage values using a fast digital-to-analog converter.

It is recommended, if:

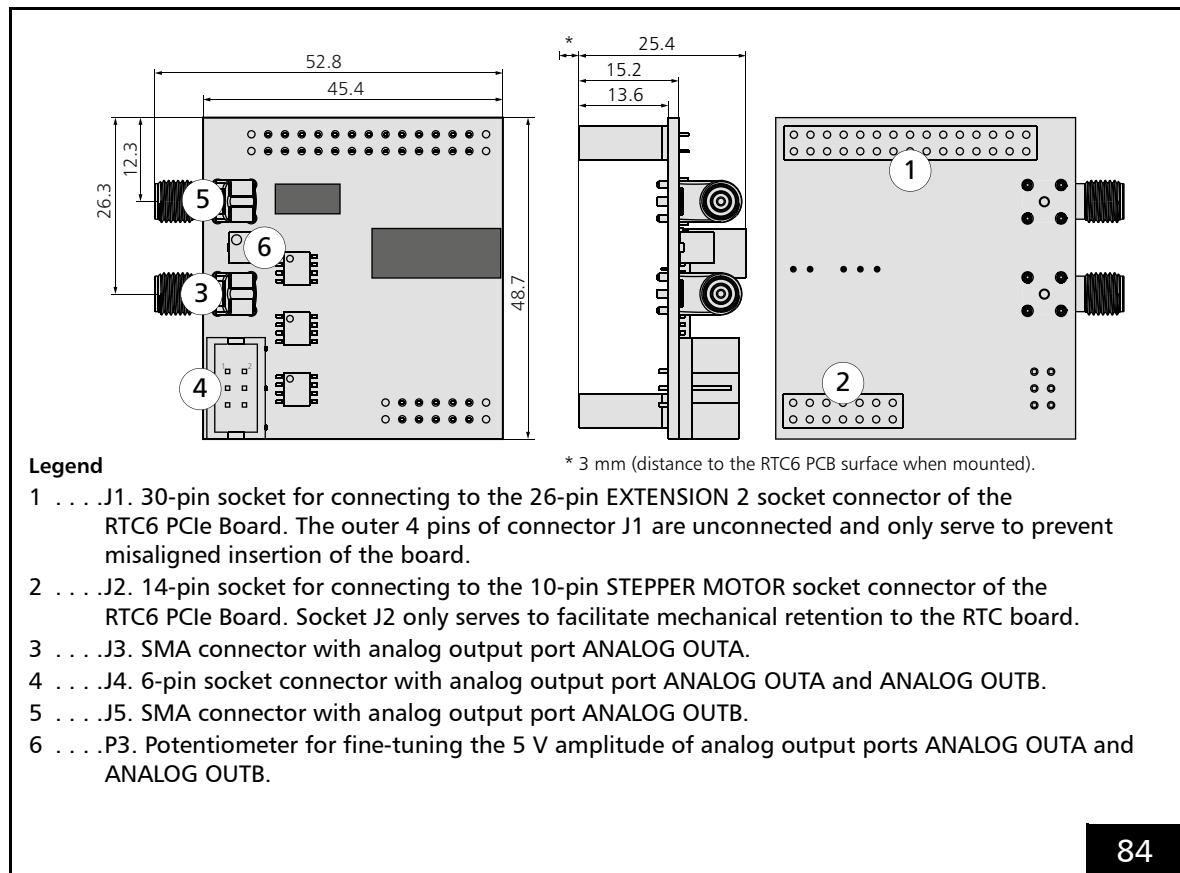
- (1) For example, #137980, #140965.
- (2) Cannot be used with RTC6 Ethernet Boards for mechanical reasons.

- (3) Option "UFP" is mandatory for pixel output frequencies 800 kHz...3.2 MHz.
- (4) The UFP Ext Board also supports pixel output frequencies < 100 kHz, of course.



UFP Ext Board: Assembly with the RTC6 PCIe Board.

Note: The SSHC slot bracket (#115132) cannot be used.



UFP Extension Board: dimensions and details. All dimensions in mm.

84

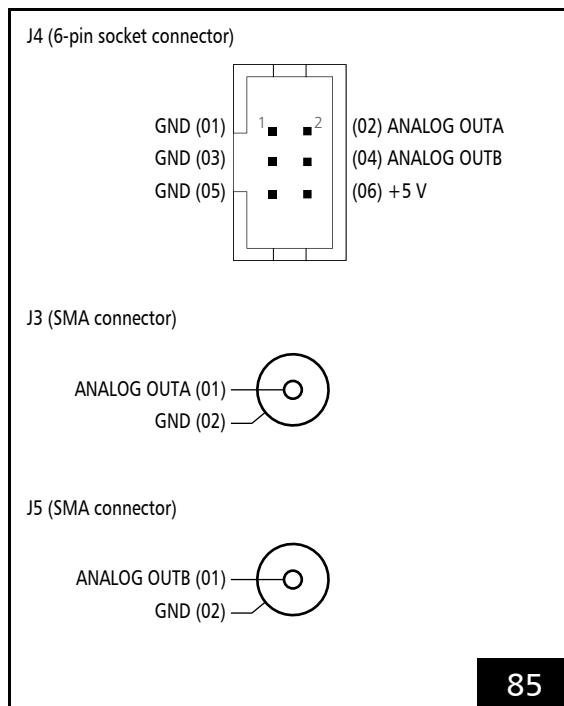
The UFP Extension Board can be used to control the pixel-to-pixel variation in laser power via analog voltage values with 8 bit resolution.

Output voltage range: 0 V...5 V.

For this purpose, the outputs must be sent to Port = 3 (8-bit digital output port at the EXTENSION 2 socket connector), see [set\\_pixel\\_line](#).

For all pixel output modes Modus = 0, Mode = 16, Mode = 32, Mode = 64 including their extensions, the laser pulse duration must be specified before the beginning of the pixel line by [set\\_laser\\_pulses](#), [set\\_laser\\_pulses\\_ctrl](#) or [set\\_laser\\_timing](#) with at least 1/64  $\mu$ s duration (even if the pulse duration is not used for laser control). Otherwise, no pulses are outputted at LASER1.

The UFPM Extension Board adapts the LATCH signal duration to the pixel frequency (approx. HalfPeriod...5 µs). It synchronously converts the 8-bit digital values to analog values in the range 0 V...5 V and makes them available at its analog output ports ANALOG OUTA and ANALOG OUTB. The connector pinouts on the UFPM Extension Board are shown in **figure 85**. Potentiometer P3 lets you fine-tune the exact voltage amplitude, see **figure 84**.



85

UFPM Extension Board: connector pinouts.

## Software Requirements

- The UFPM Extension Board is supported by default by the RTC6 Software Package package.

## Hardware Requirements

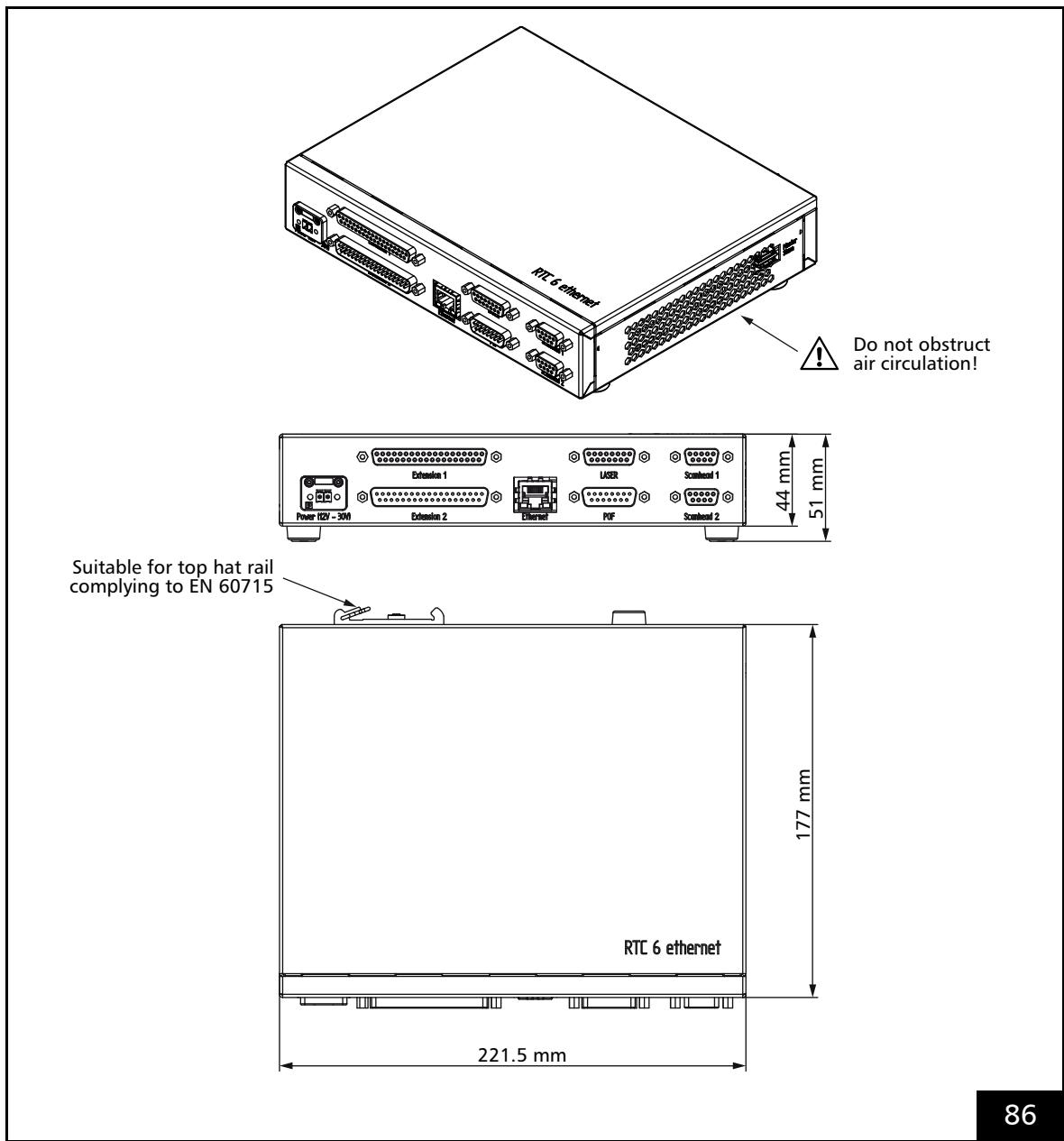
- The pixel values are outputted as 8-bit digital values at the 8-bit digital output of the RTC6 PCIe Board, EXTENSION 2 socket connector. To enable outputting of (complete) pixel values as well as the latch signal at the EXTENSION 2 socket connector, the following solder jumper on the RTC6 PCIe Board must be closed<sup>(1)</sup>:
  - DATA7 in solder jumper field C (= the DATA7 signal is outputted at pin 15 of the EXTENSION 2 socket connector)
  - LATCH in solder jumper field B (= the LATCH signal is outputted at pin 17 of the EXTENSION 2 socket connector)

## Technical Specifications

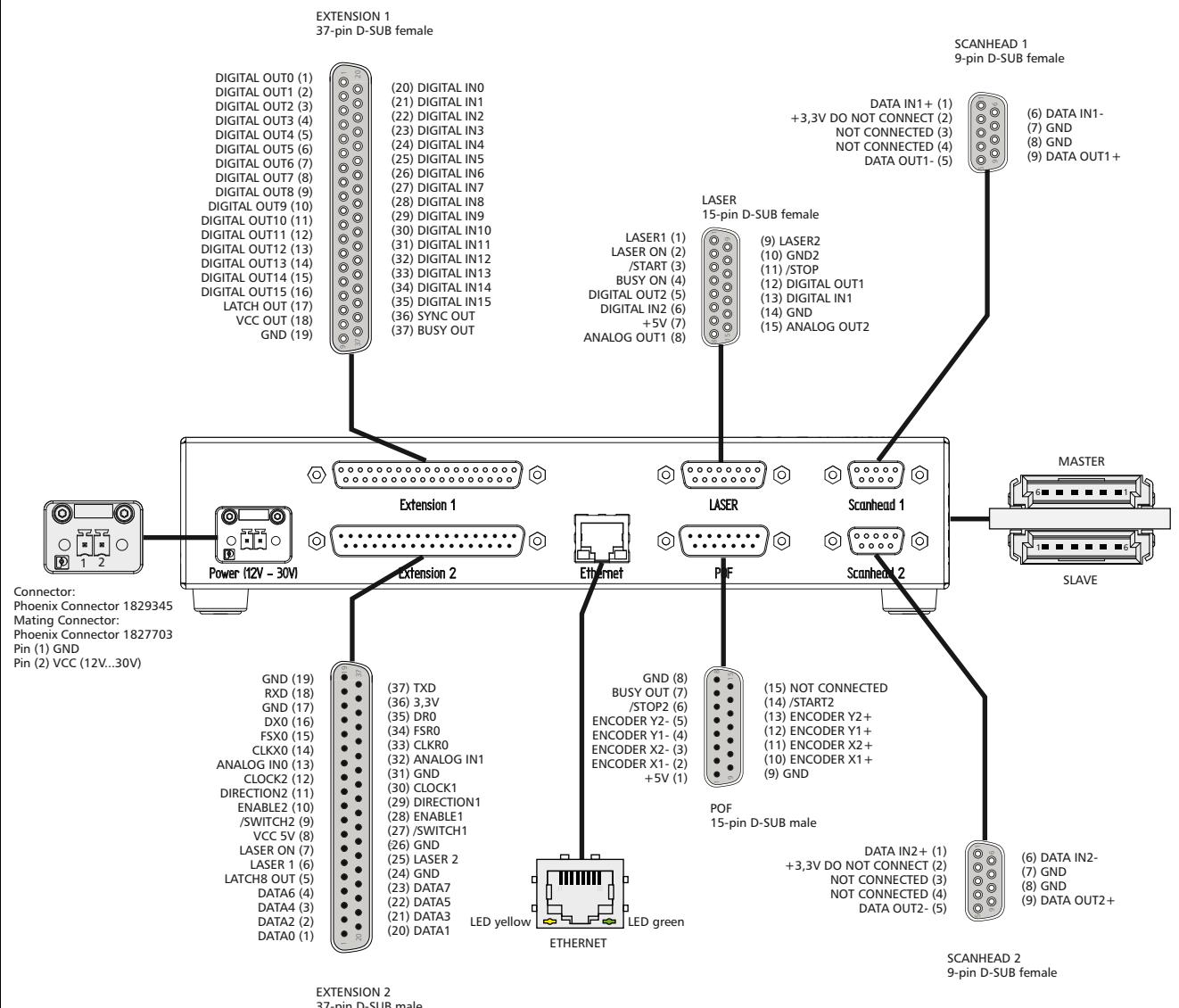
- Dimensions
  - Length 52.8 mm
  - Width 48.7 mm
- Analog Output Ports ANALOG OUTA, ANALOG OUTB
  - Connectors 1 6-pin socket connector, 2 SMA connectors (coaxial connectors)
  - Output voltage range 0 V...5 V
  - Resolution 8 bits
  - Max. current load 5 mA
  - Reference GND

(1) Corresponds to RTC6 PCIe Boards TYPE n24.

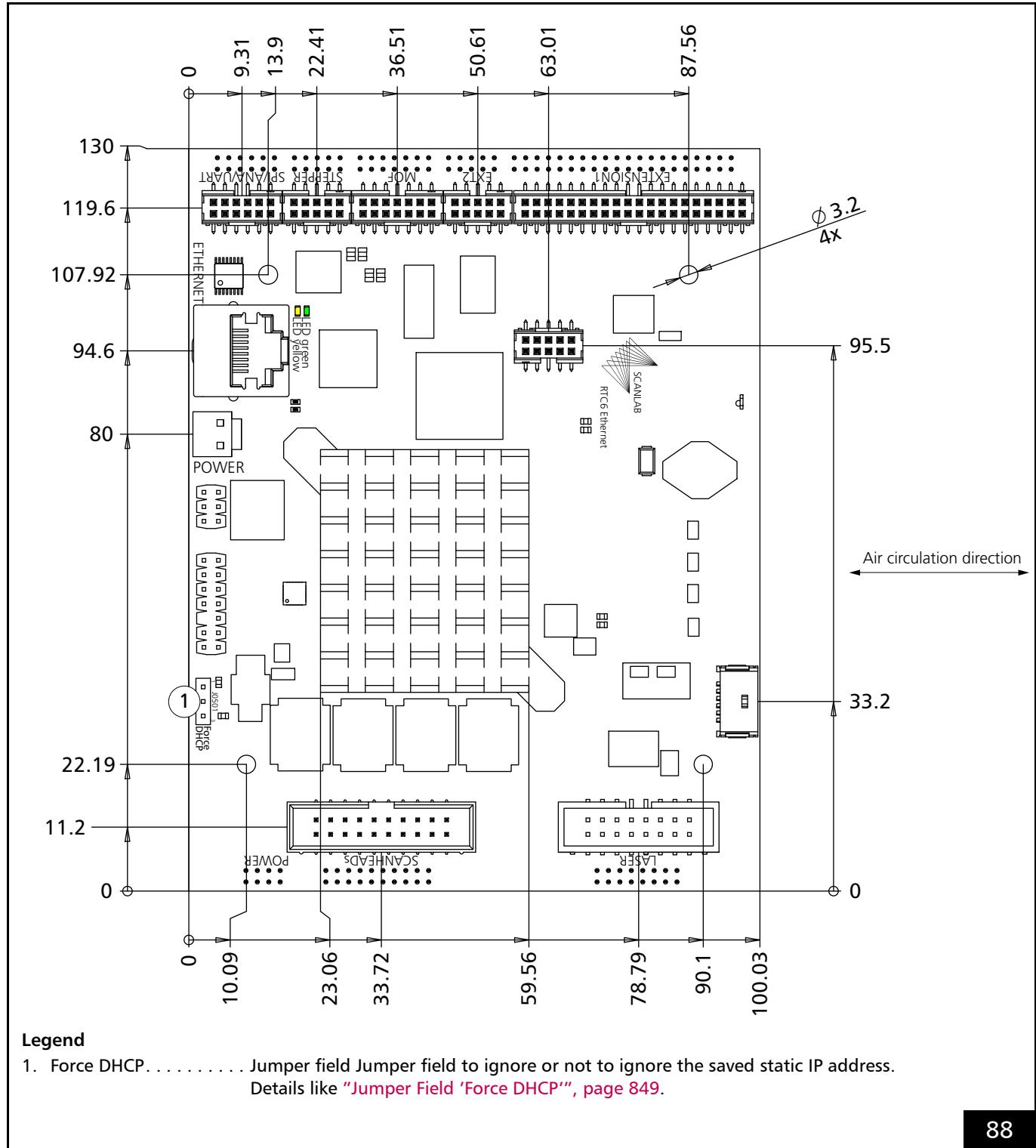
## 17 Appendix C: The RTC6 Ethernet Plug-in + Box



RTC6 Ethernet Plug-in + Box. Dimensions. For example, #141600, #143089, #143312, #143505.



RTC6 Ethernet Plug-in + Box. Pin-outs.



RTC6 Ethernet Plug-in + Box: Installed RTC6 Ethernet Board. Upper side. Dimensions and connector positions. All dimensions in mm.

## 17.1 Mounting

### Notice!

- The RTC6 board will be irreversibly destroyed if overheated. Make sure that the perforated sides of the RTC6 Ethernet Plug-in + Box are never closed or covered.
- Use the top-hat rail bracket on the backside, see [figure 86](#), to hang the RTC6 Ethernet Plug-in + Box in a top hat rail. alternatively, place the device free.

## 17.2 Cabling



### Caution!

- Even when both RTC6 Ethernet Plug-in + Boxes are connected to the same power supply, different potentials can result of unfavorable cabling and can destroy the boards. To avoid them:
  - Connect one RTC6 Ethernet Plug-in + Box to the power supply. Then connect the other RTC6 Ethernet Plug-in + Box by a cable junction to the same power supply.
  - Keep the connection between the RTC6 Ethernet Plug-in + Boxes as short as possible.
  - First do the cabling of the components correctly, then switch on the power supply.
  - Switch off the power supply before disconnect the cabling.

- Power is supplied via a Phoenix connector, see [figure 87](#). Use the delivered Phoenix mating connector (manufacturer's article number: 1827703).
- Connect the PC to the Ethernet connector via an Ethernet cable.
- Connect your desired devices to the respective connector. Observe the pin-outs of the connectors, see [figure 87](#).
- If you use several RTC6 Ethernet Plug-in + Boxes together, take care of a correct master/slave connection. See also [Chapter 6.6.3 "Master/Slave Operation", page 114](#).

## 17.3 Installation and Operation

- When installing the RTC6 Ethernet Board, a static IP address must be obtained. For that, you have to set the jumper field **Force DHCP** on the RTC Ethernet board.
  - Make sure that the RTC6 Ethernet Plug-in + Box is disconnected from the power supply.
  - Unscrew the two Torx screws on the back of the RTC6 Ethernet Plug-in + Box and remove the housing cover.
  - Set the jumper field '**Force DHCP**', see [figure 88](#), to "Force DHCP position" (see also [page 849](#)).
 
  - Put the housing cover back on and screw the two Torx screws from step 2.
  - Connect the RTC6 Ethernet Board to the power supply back on again and a network, see also [Chapter 15.3.3 "Connecting to a Network", page 850](#).
  - More information about installation and use of the RTC6 Ethernet Board (inside the RTC6 Ethernet Plug-in + Box) can be found in the main part of the RTC6 Manual.



## 18 Glossary and Abbreviations

[*]mark[*] command	All commands with "mark" as part of their names. See <a href="#">Section "Mark Commands", page 126</a> .
[*]para[*] command	All commands with "para" as part of their names. See <a href="#">Section "[*]Para[*] Commands", page 128</a> .
3D image field	Synonym: working volume (process volume).
BCD	Binary Coded Decimal.
BIOS	Basic Input/Output System. Is a part of the RTC6 firmware and permanently stored in the flash memory of the RTC6 board.
<b>BUSY</b> pin	Pin with BUSY OUT signal. <ul style="list-style-type: none"><li>• RTC6 PCIe Board<ul style="list-style-type: none"><li>– "LASER Connector", page 62</li><li>– "EXTENSION 1 Socket Connector", page 67</li></ul></li><li>• RTC6 Ethernet Board<ul style="list-style-type: none"><li>– "LASER Socket Connector", page 834</li><li>– "EXTENSION 1 Socket Connector", page 842</li></ul></li></ul>
DSP	Digital signal processor on the RTC6 board.
Dynamic focusing unit	This includes, for example, the following SCANLAB products: <i>varioSCAN</i> , <i>varioSCAN<sub>de</sub></i> , <i>varioSCAN FC</i> and <i>varioSCAN FLEX</i> , <i>excelliSHIFT</i> .
Flash memory	Non-volatile memory on the RTC6 board that replaces the EEPROM of the RTC5 board.
FPGA	Field programmable gate array on the RTC6 board.
Hardware reset	New start after powering the RTC6 board. Synonym: "power up", "power cycle".
Hard jump	Direct output to a specified position. Decomposition into microsteps within a single 10 µs clock cycle.
iDRIVE scan systems	In this manual, the term subsumes, for example, the following SCANLAB products: <i>intelliSCAN</i> , <i>intelliSCAN<sub>de</sub></i> , <i>intelliSCAN<sub>se</sub></i> , <i>intelliDRILL</i> , <i>intellicube</i> , <i>intelliWELD</i> , <i>varioSCAN<sub>de</sub></i> , <i>powerSCAN II 50i</i> , <i>excelliSCAN</i> .
Image field	Synonym: <a href="#">Working field</a> .
intelliSCAN	In this manual, the term subsumes, for example, the following SCANLAB products: <i>intelliSCAN</i> , <i>intelliSCAN<sub>de</sub></i> , <i>intelliSCAN<sub>se</sub></i> , <i>intelliDRILL</i> , <i>intellicube</i> , <i>intelliWELD</i> , <i>powerSCAN II 50i</i> .



Jump command	Serves to move the scan system axes to a new position while the laser is off. <a href="#">Jump command</a>
LSB	Least Significant Bit.
Mark command	Serves to perform marking motions while the laser is switched on. Examples: mark, arc and ellipse. See " <a href="#">Mark Commands</a> ", page 296, and " <a href="#">3D Mark Commands<sup>(1)</sup></a> ", page 296.
McBSP	Multi channel Buffered Serial Port.
MSB	Most Significant Bit.
NAND memory	Non-volatile memory on the RTC6 Ethernet Board. Synonym: NAND flash.
NULL	Means on the one hand the number 0, on the other hand a pointer with the value 0. The spelling for this is different in the different programming languages.
PCB	Printed Circuit Board.
Pixel mode	Brief for "Pixel output mode". See <a href="#">Chapter 8.7 "Pixel Output Mode – Marking Pixel Images (Bitmaps)</a> ", page 249.
Polyline	A direct sequence of [*]mark[*] commands or arc commands. The marking is continuous.
Processing-on-the-fly session	A section of a list that uses external inputs (encoder pulses, McBSP transmissions) to correct moving work-piece positions.
Reset of the RTC6 board	Synonym: <a href="#">Software reset</a> .
SCANAhead system	SCANLAB scan system with SCANAhead servo control, for example, scan heads of the excelliSCAN series. For further information see manual "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards".
Software reset	Restart after <a href="#">load_program_file</a> . This does not reset everything, for example, loaded correction tables are retained. Synonym: <a href="#">Reset of the RTC6 board</a> .
Standalone Operation Mode	See <a href="#">Chapter 15.7 "Standalone Functionality"</a> , page 859.
Trajjectory	In this manual: curve with 10 µs parameterization.
Tracking Error	Time difference between the planned and actual reaching of a certain mirror position.
Working field	Synonym: <a href="#">Image field</a> .

## 19 Change Index

The following are changes in this manual due to the technical evolution of the product as well as significant editorial changes.

### Changes from document revision 0.0 to document revision 1.0.0

Name of chapter / command table	Notes / Changes
Global	Initial Document-Revision. Document Revision 1.0.0 applies to RTC6 Software Package V1.5.0.

### Changes from document revision 1.0.0 to document revision 1.0.1

Name of chapter / command table	Notes / Changes
Global	Document Revision 1.0.1 applies to RTC6 Software Package V1.5.2.
<a href="#">eth_get_com_timeouts</a> , page 355	Software change. <b>RTC6 DLL</b> -only settings are now possible even without access to an RTC6 Ethernet Board.
<a href="#">eth_set_com_timeouts</a> , page 366	Same as <a href="#">eth_get_com_timeouts</a> .
<a href="#">load_disk</a> , page 475	Software change. Has now a version control.
<a href="#">range_checking</a> , page 549	Software change. <code>Mode = 2</code> added: a <b>simulate_ext_stop</b> is forwarded to all slave boards.
<a href="#">save_disk</a> , page 572	Same as <a href="#">load_disk</a> .
<a href="#">set_wobbel_vector</a> , page 735	Editorial enhancement. To activate "Freely definable wobbel shape" by <a href="#">set_wobbel_control</a> and <a href="#">set_wobbel_mode</a> ( <code>Mode = 3</code> ), see <a href="#">page 736</a> .
<a href="#">sync_slaves</a> , page 765	Software change. Has no function anymore.

### Changes from document revision 1.0.1 to document revision 1.0.2

Name of chapter / command table	Notes / Changes
Global	Document Revision 1.0.2 applies to RTC6 Software Package V1.6.0.
Section "Folder iSCANCfg", page 25	iSCANCfg6.exe is replaced in the RTC6 Software Package by the generic iSCANCfg.exe.
Chapter 7.4.1 "Enabling, Activating and Switching Laser Control Signals"	Editorial change. Figure 47, page 174.

Name of chapter / command table (cont'd)	Notes / Changes (cont'd)
<a href="#">control_command</a> , page 332	Editorial change. Description of "PowerOK" and "TempOK" has been changed, see <a href="#">page 332</a> . Furthermore, the description of 05H28H and 05H29H has been removed from the document (these data signal types are not intended for users).
<a href="#">get_error</a> , page 384	Editorial change. Error Bit #15 no longer "reserved", see <a href="#">page 386</a> .
<a href="#">Chapter 15.2.13 "Master Socket Connector, Slave Socket Connector"</a>	Editorial change. Safety notice on Master/Slave-connected RTC6 Ethernet Boards, see <a href="#">page 845</a> .

#### Changes from document revision 1.0.2 to document revision 1.0.3

Name of chapter / command table	Notes / Changes
Global	Document Revision 1.0.3 applies to RTC6 Software Package V1.6.1.
Global	Editorial change. The previously used term "Online Positioning" (due to the introduction of " <a href="#">Global Online Positioning</a> ") now reads - where applicable - "Local Online Positioning", for example, in <a href="#">Chapter 8.3.1</a> " <a href="#">"Local Online Positioning"</a> ", page 214.
<a href="#">Chapter 8.3.2 ""Global Online Positioning""</a> , page 217	Editorial enhancement. Description of " <a href="#">Global Online Positioning</a> ".
<a href="#">Chapter 8.4 "Wobbel Mode"</a>	Editorial enhancement. Section " <a href="#">Example Code</a> ", page 219.
<a href="#">Chapter 8.6.12 "Fly Extension" Commands</a> , page 245	Editorial enhancement. Description of " <a href="#">Fly Extension</a> " Commands.
<a href="#">activate_fly_1_axis</a> , page 304	Software change. <sup>(a)</sup> New command.
<a href="#">activate_fly_2_axes</a> , page 305	Software change. <sup>(a)</sup> New command.
<a href="#">control_command</a> , page 332	Editorial change. The description of 05H2BH and 05H2CH has been removed from the document (these data signal types are not intended for users).
<a href="#">fly_return_1_axis</a> , page 374	Software change. <sup>(a)</sup> New command.
<a href="#">fly_return_2_axes</a> , page 375	Software change. <sup>(a)</sup> New command.
<a href="#">fly_return_3_axes</a> , page 376	Software change. <sup>(a)</sup> New command.
<a href="#">get_startstop_info</a> , page 410	Software change. <sup>(a)</sup> New Bit #14.



Name of chapter / command table (cont'd)	Notes / Changes (cont'd)
<a href="#">park_position_1_axis, page 539</a>	Software change. <sup>(a)</sup> New command.
<a href="#">park_position_2_axes, page 540</a>	Software change. <sup>(a)</sup> New command.
<a href="#">park_return_1_axis, page 543</a>	Software change. <sup>(a)</sup> New command.
<a href="#">park_return_2_axes, page 544</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_fly_1_axis, page 611</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_fly_2_axes, page 612</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_fly_3_axes, page 616</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_matrix, page 656</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_matrix_list, page 656</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_rot, page 657</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_rot_list, page 657</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_x, page 658</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_x_list, page 658</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_y, page 659</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_mcbsp_global_y_list, page 659</a>	Software change. <sup>(a)</sup> New command.
<a href="#">store_timestamp_counter, page 759</a>	Software change. <sup>(a)</sup> New command.
<a href="#">store_timestamp_counter_list, page 759</a>	Software change. <sup>(a)</sup> New command.
<a href="#">wait_for_1_axis, page 796</a>	Software change. <sup>(a)</sup> New command.
<a href="#">wait_for_2_axes, page 798</a>	Software change. <sup>(a)</sup> New command.
<a href="#">wait_for_timestamp_counter, page 806</a>	Software change. <sup>(a)</sup> New command.

(a) See also [RTC6\\_Software\\_RevisionHistory\\_<Date>\\_<Rev>.pdf](#).



#### Changes from document revision 1.0.3 to document revision 1.0.4

Name of chapter / command table	Notes / Changes
Global	Document Revision 1.0.4 applies to RTC6 Software Package V1.7.0.
Section "Folder RTC6 Tools", page 26	Software change. <sup>(a)</sup> New file. <a href="#">RTC6BIOSETH_26.out</a> is a prerequisite for using the <b>Standalone Functionality</b> .
<a href="#">eth_boot_dcmd</a> , page 347	Software change. <sup>(a)</sup> New command.
<a href="#">set_eth_boot_control</a> , page 606	Software change. <sup>(a)</sup> New command.
<a href="#">set_eth_boot_timeout</a> , page 606	Software change. <sup>(a)</sup> New command.
<a href="#">set_free_variable</a> , page 628	Software change. <sup>(a)</sup> Changed command.
<a href="#">read_image_eth</a> , page 554	Software change. <sup>(a)</sup> New command.
<a href="#">store_program</a> , page 758	Software change. <sup>(a)</sup> New command.
<a href="#">write_image_eth</a> , page 814	Software change. <sup>(a)</sup> New command.
Chapter 15.2.15 "Real-Time Clock", page 849	Editorial enhancement. Description of the RTC6 Ethernet Board real-time clock.
Chapter 15.7 "Standalone Functionality", page 859	Software change. <sup>(a)</sup> New functionality for RTC6 Ethernet Boards: <b>Standalone Functionality</b> .
Chapter 17 "Appendix C: The RTC6 Ethernet Plug-in + Box", page 875	Editorial enhancement. Description of the RTC6 Ethernet Plug-in + Box.

(a) See also [RTC6\\_Software\\_RevisionHistory\\_<Date>\\_<Rev>.pdf](#).