

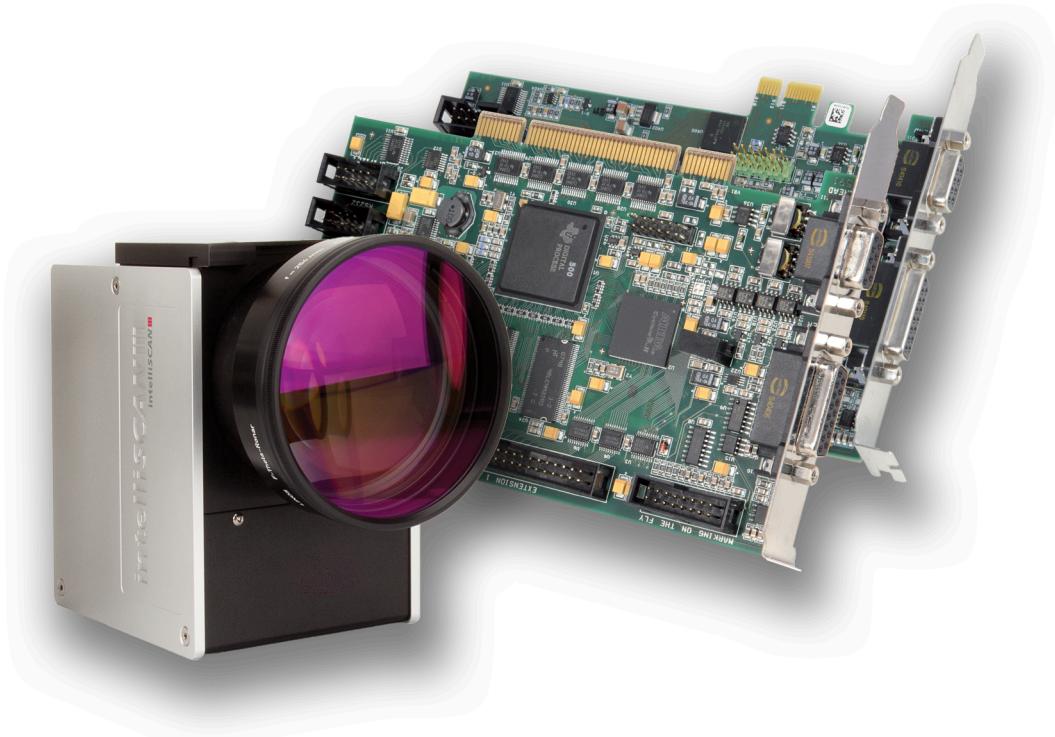


# Installation and Operation

**RTC5 PCI Board**

**RTC5 PCIe Board**

Real Time Control of Scan Systems and Lasers



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

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

No part of this manual 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.

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



## Contents

<b>1</b>	<b>About this Manual</b>	<b>21</b>
1.1	Manufacturer	21
1.2	Related Documents	22
1.3	Glossary and Abbreviations	23
<b>2</b>	<b>Product Overview</b>	<b>27</b>
2.1	Labeling	27
2.2	Unpacking Instructions and Typical Scope of Delivery	27
2.3	Delivered RTC5 Software Package	27
	Folder Correction Files	27
	Folder CorrectionFileConverter	27
	Folder DemoFiles	28
	Folder HPGL	28
	Folder iSCANConfig	28
	Folder RTC5 Driver	28
	Folder RTC5 Files	28
	Folder RTC5 Tools	29
2.4	Intended Use	30
2.5	System Requirements	32
2.5.1	Hardware	32
2.5.2	Software	32
2.6	Options	34
2.7	Jumper Settings and Type Identification	35
2.7.1	Jumper JP1 – Configuring the Output Signal Level at the EXTENSION 1 Socket Connector	36
2.7.2	Jumper JP2...JP8 – Configuring Pin15 and Pin17 at the EXTENSION 2 Socket Connector	36
2.8	Accessories for the RTC5 PCI Board	37
2.8.1	XY2-100 Converter	37
2.8.2	Laser Adapter	37
2.8.3	Data Cables	37
2.8.4	Slot Cover with 9-pin D-SUB Connector for 2. SCANHEAD Socket Connector	38
2.8.5	Slot Cover with 15-pin D-SUB Connector for MARKING ON THE FLY Socket Connector	38
2.8.6	Slot Cover with 15-pin D-SUB Connector and 9-pin D-SUB Connector	39
2.8.7	ADC Add-On Board	39
2.8.8	Extension Board "RTC5/6 varioSCAN FLEX Extension"	39
2.9	Supplementary Software	40
2.10	Notes for RTC4 Users	40
2.10.1	Hardware Changes	40
	Controlling Scan Systems	40
	Controlling the Laser	41
	EXTENSION 1 Socket Connector	41
	EXTENSION 2 Socket Connector	41
	MARKING ON THE FLY Socket Connector	41
	Other Hardware Interfaces	41
2.10.2	Porting RTC4 Source Code to the RTC5 PCI Board	42
	Changed Initialization	42
	Command Changes	42
	Increased Parameter Resolution	43
	Changed Timing Behavior	44
2.10.3	New and Changed Functionality	45
	Interface to the PC	45



Controlling Scan Systems .....	45
Controlling the Laser .....	46
Interfaces for Peripheral Equipment .....	46
General Programming .....	47
Laser Marking .....	48
Special Functions .....	48
<b>3 Safety During Installation and Operation .....</b>	<b>50</b>
3.1 Steps for Safe Operation .....	50
3.2 Laser Safety .....	50
<b>4 RTC5 PCI Board – Layout and Interfaces .....</b>	<b>51</b>
4.1 Layout – Upper Side .....	51
4.2 Layout – Lower Side .....	52
4.3 Interface to PC .....	52
4.4 Master Socket Connector, Slave Socket Connector .....	53
4.5 Interfaces to Scan System .....	54
4.5.1 Scan Head Connectors and Transfer Protocol .....	54
SCANHEAD Connector .....	54
2. SCANHEAD Socket Connector .....	55
4.5.2 XY2-100 Converter (Accessory) .....	56
4.5.3 Data Cables (Accessories) .....	58
4.6 Interfaces for the Laser and Peripheral Equipment .....	60
4.6.1 LASER Connector .....	60
Laser Control Signals .....	60
External Control Signals .....	61
BUSY List Execution Status .....	61
2-Bit Digital Input Port .....	61
2-Bit Digital Output Port .....	61
12-Bit Analog Output Port 1 and 2 .....	61
Input/Output Wiring .....	61
Laser Adapter (Accessory) .....	63
4.6.2 EXTENSION 1 Socket Connector .....	65
Configuring the Output Signal Level .....	65
16-Bit Digital Input Port and 16-Bit Digital Output Port .....	65
Synchronization of Data Transmission .....	66
BUSY List Execution Status .....	66
4.6.3 EXTENSION 2 Socket Connector .....	67
Configuration by Solder Jumpers .....	67
Laser Control Signals .....	68
8-Bit Digital Output Port .....	68
4.6.4 MARKING ON THE FLY Socket Connector .....	69
Encoder Input Ports .....	69
External Control Signals .....	69
Analog Output Port .....	69
BUSY list execution status Status .....	69
MARKING ON THE FLY Slot Cover (Accessory) .....	70
4.6.5 RS232 Socket Connector .....	70
4.6.6 SPI / I2C Socket Connector .....	71
McBSP Interface .....	71
I <sup>2</sup> C Interface .....	73
SPI Interface .....	73
4.6.7 STEPPER MOTOR Socket Connector .....	75



4.6.8	Analog Inputs (Accessory) .....	75
<b>5</b>	<b>Installation and Start-Up .....</b>	<b>77</b>
5.1	Checking Jumper Settings .....	77
5.2	Installing the RTC5 PCI Board .....	77
5.3	Installing the RTC5 Board Driver .....	78
5.3.1	RTC5 Software Package Upgrades .....	79
5.3.2	Exchange of RTC Boards and Update of RTC Board Driver .....	79
5.4	Installing the RTC5 Software .....	80
5.5	Safe Start-up and Shutdown Sequences .....	80
5.6	Functionality Test .....	81
5.7	User Programs and Demo Software .....	81
5.8	Exchanging RTC5 Boards .....	82
<b>6</b>	<b>Developing RTC5 User Programs .....</b>	<b>83</b>
6.1	RTC5 Software Concept Basics .....	83
6.1.1	Controlling Scan Systems and Lasers – An Introductory Example .....	83
6.1.2	Control Commands and List Commands .....	83
6.2	Initialization and Program Start-Up .....	84
6.2.1	DLL Calling Convention .....	84
6.2.2	Importing Commands .....	84
Pascal .....	84	
C .....	85	
C++ .....	85	
C# .....	85	
6.2.3	Initializing the RTC5 DLL and Board Management .....	86
6.2.4	Start of RTC5 PCI Board Operation .....	87
Initializing the Board .....	87	
Configuring the Board .....	87	
Initializing the Scan System Control .....	88	
Initializing the Laser Control .....	88	
Loading and Executing Lists .....	88	
6.2.5	Example Code (C) .....	89
6.3	RTC5 List Memory .....	91
6.3.1	Lists and the Protected List Memory Area .....	91
"List 1" and "List 2" .....	91	
"List 3" – Protected List Memory Area .....	91	
6.3.2	Configuring the RTC5 List Memory .....	92
6.4	List Handling .....	94
6.4.1	Loading Lists .....	94
"Unconditional" Loading .....	94	
Loading with Protection .....	95	
Terminating Lists .....	95	
6.4.2	List Status .....	96
6.4.3	List Execution Status .....	97
6.4.4	Starting and Stopping Lists .....	98
6.4.5	Interrupting Lists for Synchronization of Processing .....	99
6.4.6	Automatic List Changing .....	99
One-Time List Change .....	99	
Alternating List Changes .....	100	
6.5	Structured Programming .....	101
6.5.1	Subroutines .....	101
Non-Indexed Subroutines .....	101	



Indexed Subroutines .....	102
General Information on Calling Subroutines .....	103
Index Management and Defragmentation .....	104
Subsequent Protection and Conversion of Non-Indexed Subroutines .....	105
Unprotecting Subroutines .....	106
6.5.2 Character Sets and Text Strings .....	107
Defining Indexed Character Sets .....	107
Calling Indexed Characters .....	108
Defining Indexed Text Strings for Times, Dates and Serial Numbers .....	108
Calling Indexed Text Strings .....	109
Managing Indexed Characters and Text Strings .....	109
6.5.3 Jumps .....	109
6.5.4 RTC4 Circular Queue Mode .....	110
6.5.5 Loops .....	111
6.6 Using Several RTC5 PCI Boards in One PC .....	112
6.6.1 Multi-Board Programming .....	112
Examples (Pascal) .....	112
6.6.2 Single-Board Programming .....	113
6.6.3 Master/Slave Operation .....	113
Initialization .....	114
Matching of Short-Command Counts .....	114
Synchronous Starts and Synchronous Stops .....	115
Example Code (Delphi) .....	116
6.7 Usage of RTC5 PCI Boards by Several User Programs .....	117
6.7.1 Board Acquisition by a User Program .....	117
6.8 Error Handling .....	119
6.8.1 Download Verification .....	120
6.8.2 Checking for Overruns .....	120
6.8.3 Example Code (C) .....	121
6.9 Miscellaneous .....	123
6.9.1 Free Variables .....	123
<b>7 Basic Functions for Scan Head and Laser Control .....</b>	<b>124</b>
7.1 Marking Dots, Lines and Arcs .....	124
7.1.1 Marking with Vector Commands and "Arc" Commands .....	124
Jump Commands .....	125
Mark Commands .....	125
Arc Commands .....	125
Polylines .....	125
Ellipse Commands .....	126
[*]Para[*] Commands .....	127
7.1.2 Microstepping .....	128
7.1.3 Marking Single Dots .....	129
7.1.4 Example Code (C) .....	130
7.2 Delay Settings – Coordinating Scan Head Control and Laser Control .....	133
7.2.1 Laser Delays .....	133
LaserOn Delay .....	134
LaserOff Delay .....	134
7.2.2 Scanner Delays .....	135
Jump Delay .....	135
Variable Jump Delays .....	136
Mark Delay .....	137
Polygon Delay .....	138



Variable Polygon Delays .....	139
User-defined "Variable Polygon Delays" .....	141
7.2.3 Notes on Optimizing the Delays .....	143
Recommended Optimization Sequence .....	143
Automatic Delay Adjustments .....	143
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
Carrying Out Adjustment .....	159
Checking the z Axis Calibration .....	159
Testing 3-Axis Scan Systems with F-Theta Objective .....	161
7.3.5 Image Field Correction and Correction Tables .....	162
Field Distortion .....	162
Image Field Correction Algorithm .....	163
Activating Image Field Correction .....	163
2D Correction Files 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 .....	171
Precalculation and Diagnosis .....	171
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 Mode 1, 2, 3, 5 .....	179
Q-Switch Signal .....	179
FirstPulseKiller Signal .....	179
Differences Between the YAG Modes .....	180
Lamp Current (Laser Power) .....	180



7.4.5	Laser Mode 4 .....	182
7.4.6	Laser Mode 6 .....	183
7.4.7	Softstart Mode .....	184
7.4.8	Pulse Picking Laser Mode .....	186
7.4.9	"Automatic Laser Control" .....	187
	Position-Dependent Laser Control .....	189
	Speed-Dependent Laser Control .....	191
	Encoder-Speed-Dependent Laser Control .....	192
	Loading and Determining the Nonlinearity Curve .....	192
	Vector-Defined Laser Control .....	194
7.4.10	Output Synchronization .....	195
7.5	Marking Dates, Times and Serial Numbers .....	196
7.5.1	Marking the Date and Time .....	196
7.5.2	Marking Serial Numbers .....	196
8	Advanced Functions for Scan Head Control and Laser Control .....	198
8.1	iDRIVE Functions .....	198
8.1.1	Transfer Protocol .....	198
8.1.2	Configuring the Data Signal Transmission Behavior of the Scan System .....	199
	Setting Data Types .....	199
	Reading Out Data .....	199
8.1.3	Monitoring the Positioning .....	200
8.1.4	Configuring Tuning (Dynamics Settings) .....	202
8.1.5	Jump Mode .....	202
	Functional Principle .....	202
	Requirements and Activation .....	203
	Jump-Length-Dependent Jump Delays .....	204
	Experimental Determination of Jump Delay Values .....	204
	Notes on Loading Determined Jump Delay Values .....	205
	Automatic Determination of the Jump Delay Table .....	206
8.1.6	Configuring the PosAck Limit Value .....	207
8.1.7	Configuring the Effective Calibration .....	207
8.1.8	Configuring the Start Behavior .....	208
8.1.9	Fault Diagnosis and Functional Test .....	208
8.2	Coordinate Transformations .....	209
8.3	Online Positioning .....	213
8.3.1	"Local Online Positioning" .....	213
	Configuring "Local Online Positioning" .....	214
8.3.2	"Global Online Positioning" .....	216
	Configuring "Global Online Positioning" .....	216
8.4	Wobbel Mode .....	217
	Example Code (C++) .....	218
8.4.1	Wobbel Shapes – Important Notes on Choosing Appropriate Parameter Values .....	219
	"Classic" Wobbel Shapes .....	219
	"Freely Definable Wobbel Shapes" .....	220
8.5	Controlling 2D Scan Systems and 3D Scan Systems .....	221
8.5.1	2D Scan Systems .....	221
8.5.2	3D Scan Systems .....	222
	Intended Use .....	222
	Connection and Initialization .....	222
	3D Commands .....	223
	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 Motions .....	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	Compensating Rotary Motions .....	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 Corrections .....	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 .....	242
8.6.11	Processing-on-the-fly Correction for the z Axis .....	242
8.7	Pixel Output Mode .....	244
8.7.1	Principle of Operation .....	244
8.7.2	RTC5 Commands .....	244
8.7.3	Laser Control .....	246
8.7.4	Synchronization .....	247
8.8	micro_vector[*] Commands .....	249
8.9	Timed Commands .....	250
8.10	Automatic Self-Calibration .....	251
8.10.1	Using for Drift Compensation .....	251
8.10.2	How it Works .....	251
8.10.3	Determining Reference Values .....	253
8.10.4	Calibration During the Application .....	253
	Automatic Self-Calibration .....	253
	Customer-Specific Calibration .....	254
	Supplemental Information about Calibration .....	254
8.11	Camming .....	255
8.12	Time Measurements .....	257
8.12.1	RTC5 Timer .....	257
8.12.2	Timestamps .....	257
9	Programming Peripheral Interfaces .....	258
9.1	Signal Output .....	258
9.1.1	16-Bit Digital Output Port .....	258
9.1.2	8-Bit Digital Output Port .....	259
9.1.3	2 Bit Digital Output Port .....	259
9.1.4	12-Bit Analog Output Port 1 and 2 .....	259
9.1.5	Controlling Stepper Motors .....	260
	Output Signals .....	260
	Reference Motions and Position Initialization .....	261
	Set-Position Motions .....	261



Querying Signals and Status Values .....	262
Terminating Infinite Motions .....	262
WaitTime Parameter .....	262
9.1.6 RS-232 interface .....	263
9.1.7 McBSP Interface .....	263
9.2 Signal Input .....	264
9.2.1 16-Bit Digital Input Port .....	264
9.2.2 2-Bit Digital Input Port .....	264
9.2.3 RS-232 Interface .....	264
9.2.4 McBSP Interface .....	264
9.3 Control by External Signals .....	265
9.3.1 Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization .....	265
External Stop .....	265
External Start .....	266
External Start with Track Delay .....	268
Regular (Periodic) External Starts .....	269
9.3.2 Conditional Command Execution .....	271
Example Code (Pascal) .....	272
9.3.3 Synchronization by Encoder Signals .....	274
Intended Use .....	274
Input Ports for External Encoder Signals .....	275
Encoder Simulation .....	275
9.3.4 Synchronization and Online Positioning by McBSP/SPI Signals .....	276
9.4 Periodical I/O Signals .....	277
<b>10 RTC5 Commands .....</b>	<b>278</b>
10.1 Overview .....	278
10.1.1 Designations .....	278
Multi-Board Commands .....	278
Normal, Short, Variable and Multiple List Commands .....	278
Undelayed and Delayed Short List Commands .....	279
Multiple List Commands .....	280
10.1.2 Compatibility .....	280
10.1.3 Version Information .....	281
10.1.4 Optional Functions .....	281
10.1.5 Control Commands .....	282
10.1.6 List Commands .....	285
10.1.7 Data Types .....	289
Pointer to Locations in the PC Memory .....	289
10.2 RTC5 Command Set .....	290
acquire_rtc .....	291
activate_fly_2d .....	292
activate_fly_2d_encoder .....	293
activate_fly_xy .....	294
activate_fly_xy_encoder .....	294
apply_mcbsp .....	295
apply_mcbsp_list .....	296
arc_abs .....	297
arc_abs_3d .....	298
arc_rel .....	299
arc_rel_3d .....	300
auto_cal .....	301



auto_change .....	304
auto_change_pos .....	305
bounce_supp .....	306
camming .....	307
clear_fly_overflow .....	309
clear_fly_overflow_ctrl .....	309
clear_io_cond_list .....	310
config_laser_signals .....	311
config_laser_signals_list .....	312
config_list .....	313
control_command .....	315
copy_dst_src .....	317
disable_laser .....	318
enable_laser .....	319
execute_at_pointer .....	320
execute_list .....	320
execute_list_1 .....	321
execute_list_2 .....	321
execute_list_pos .....	321
fly_return .....	323
fly_return_z .....	324
free_rtc5_dll .....	325
get_auto_cal .....	326
get_char_pointer .....	327
get_config_list .....	328
get_counts .....	328
get_dll_version .....	329
get_encoder .....	329
get_error .....	330
get_fly_2d_offset .....	334
get_free_variable .....	334
get_galvo_controls .....	335
get_head_para .....	337
get_head_status .....	338
get_hex_version .....	340
get_hi_data .....	340
get_hi_pos .....	341
get_input_pointer .....	342
get_io_status .....	342
get_jump_table .....	343
get_lap_time .....	343
get_laser_pin_in .....	344
get_last_error .....	344
get_list_pointer .....	345
get_list_serial .....	346
get_list_space .....	346
get_marking_info .....	347
get_master_slave .....	350
get_mcbsp .....	351
get_mcbsp_list .....	351
get_out_pointer .....	351
get_overrun .....	352



get_RTC_mode .....	352
get_RTC_version .....	353
get_serial .....	354
get_serial_number .....	354
get_standby .....	355
get_startstop_info .....	356
get_status .....	358
get_stepper_status .....	360
get_sub_pointer .....	361
get_sync_status .....	362
get_table_para .....	363
get_text_table_pointer .....	364
get_time .....	364
get_transform .....	365
get_value .....	368
get_values .....	370
get_wait_status .....	371
get_waveform .....	372
get_z_distance .....	373
goto_xy .....	374
goto_xyz .....	375
home_position .....	376
home_position_xyz .....	377
if_cond .....	378
if_fly_x_overflow .....	378
if_fly_y_overflow .....	379
if_fly_z_overflow .....	379
if_not_activated .....	380
if_not_cond .....	381
if_not_fly_x_overflow .....	382
if_not_fly_y_overflow .....	382
if_not_fly_z_overflow .....	383
if_not_pin_cond .....	383
if_pin_cond .....	384
init_fly_2d .....	385
init_RTC5_dll .....	386
jump_abs .....	388
jump_abs_3d .....	389
jump_abs_drill .....	389
jump_abs_drill_2 .....	389
jump_rel .....	390
jump_rel_3d .....	391
jump_rel_drill .....	391
jump_rel_drill_2 .....	391
laser_on_list .....	392
laser_on_pulses_list .....	393
laser_signal_off .....	395
laser_signal_off_list .....	395
laser_signal_on .....	396
laser_signal_on_list .....	396
list_call .....	397
list_call_abs .....	399

list_call_abs_cond .....	400
list_call_abs_repeat .....	400
list_call_cond .....	401
list_call_repeat .....	402
list_continue .....	403
list_jump_cond .....	404
list_jump_pos .....	405
list_jump_pos_cond .....	406
list_jump_rel .....	407
list_jump_rel_cond .....	408
list_next .....	409
list_nop .....	409
list_repeat .....	410
list_return .....	411
list_until .....	412
load_auto_laser_control .....	413
load_char .....	415
load_correction_file .....	416
load_disk .....	420
load_fly_2d_table .....	422
load_jump_table .....	423
load_jump_table_offset .....	424
load_list .....	426
load_position_control .....	428
load_program_file .....	430
load_stretch_table .....	433
load_sub .....	435
load_text_table .....	436
load_varpolydelay .....	437
load_z_table .....	439
load_zoom_correction_file .....	440
long_delay .....	441
mark_abs .....	442
mark_abs_3d .....	443
mark_char .....	444
mark_char_abs .....	445
mark_date .....	446
mark_date_abs .....	448
mark_ellipse_abs .....	449
mark_ellipse_rel .....	450
mark_rel .....	451
mark_rel_3d .....	452
mark_serial .....	453
mark_serial_abs .....	455
mark_text .....	456
mark_text_abs .....	457
mark_time .....	458
mark_time_abs .....	460
mcbsp_init .....	461
mcbsp_init_spi .....	462
measurement_status .....	463
micro_vector_abs .....	464



micro_vector_abs_3d .....	465
micro_vector_rel .....	466
micro_vector_rel_3d .....	467
move_to .....	467
number_of_correction_tables .....	468
para_jump_abs .....	469
para_jump_abs_3d .....	470
para_jump_rel .....	471
para_jump_rel_3d .....	472
para_laser_on_pulses_list .....	473
para_mark_abs .....	475
para_mark_abs_3d .....	476
para_mark_rel .....	477
para_mark_rel_3d .....	478
park_position .....	479
park_return .....	481
pause_list .....	483
periodic_toggle .....	484
periodic_toggle_list .....	485
quit_loop .....	486
range_checking .....	487
read_abc_from_file .....	489
read_analog_in .....	490
read_encoder .....	491
read_io_port .....	492
read_io_port_buffer .....	493
read_io_port_list .....	494
read_mcbsp .....	495
read_multi_mcbsp .....	496
read_status .....	497
read_user_data .....	499
release_rtc .....	500
release_wait .....	501
reset_error .....	502
restart_list .....	503
rs232_config .....	503
rs232_read_data .....	504
rs232_write_data .....	505
rs232_write_text .....	505
rs232_write_text_list .....	506
rtc5_count_cards .....	507
save_and_restart_timer .....	508
save_disk .....	509
select_char_set .....	511
select_cor_table .....	512
select_cor_table_list .....	515
select_rtc .....	516
select_serial_set .....	518
select_serial_set_list .....	518
send_user_data .....	519
set_angle .....	520
set_angle_list .....	521



set_auto_laser_control .....	522
set_auto_laser_params .....	525
set_auto_laser_params_list .....	525
set_char_pointer .....	526
set_char_table .....	527
set_control_mode .....	528
set_control_mode_list .....	530
set_default_pixel .....	530
set_default_pixel_list .....	530
set_defocus .....	531
set_defocus_list .....	532
set_defocus_offset .....	532
set_defocus_offset_list .....	533
set_delay_mode .....	534
set_delay_mode_list .....	535
set_dsp_mode .....	536
set_ellipse .....	537
set_encoder_speed .....	538
set_encoder_speed_ctrl .....	539
set_end_of_list .....	540
set_extstartpos .....	541
set_extstartpos_list .....	541
set_ext_start_delay .....	542
set_ext_start_delay_list .....	543
set_firstpulse_killer .....	544
set_firstpulse_killer_list .....	544
set_fly_2d .....	545
set_fly_limits .....	546
set_fly_limits_z .....	547
set_fly_rot .....	548
set_fly_rot_pos .....	549
set_fly_tracking_error .....	550
set_fly_x .....	551
set_fly_x_pos .....	552
set_fly_y .....	553
set_fly_y_pos .....	554
set_fly_z .....	555
set_free_variable .....	556
set_free_variable_list .....	556
set_hi .....	557
set_input_pointer .....	558
set_io_cond_list .....	559
set_jump_mode .....	560
set_jump_mode_list .....	563
set_jump_speed .....	564
set_jump_speed_ctrl .....	564
set_jump_table .....	565
set_laser_control .....	566
set_laser_delays .....	569
set_laser_mode .....	570
set_laser_off_default .....	570
set_laser_pin_out .....	571



set_laser_pin_out_list .....	571
set_laser_pulses .....	572
set_laser_pulses_ctrl .....	573
set_laser_timing .....	574
set_list_jump .....	575
set_mark_speed .....	576
set_mark_speed_ctrl .....	576
set_matrix .....	577
set_matrix_list .....	579
set_max_counts .....	580
set_mcbsp_freq .....	580
set_mcbsp_global_matrix .....	581
set_mcbsp_global_matrix_list .....	581
set_mcbsp_global_rot .....	582
set_mcbsp_global_rot_list .....	582
set_mcbsp_global_x .....	583
set_mcbsp_global_x_list .....	583
set_mcbsp_global_y .....	584
set_mcbsp_global_y_list .....	584
set_mcbsp_in .....	585
set_mcbsp_in_list .....	586
set_mcbsp_matrix .....	587
set_mcbsp_matrix_list .....	588
set_mcbsp_out .....	588
set_mcbsp_out_ptr .....	589
set_mcbsp_out_ptr_list .....	590
set_mcbsp_rot .....	591
set_mcbsp_rot_list .....	591
set_mcbsp_x .....	592
set_mcbsp_x_list .....	592
set_mcbsp_y .....	593
set_mcbsp_y_list .....	593
set_multi_mcbsp_in .....	594
set_multi_mcbsp_in_list .....	596
set_n_pixel .....	597
set_offset .....	598
set_offset_list .....	598
set_offset_xyz .....	599
set_offset_xyz_list .....	600
set_pause_list_cond .....	601
set_pause_list_not_cond .....	602
set_pixel .....	603
set_pixel_line .....	604
set_pixel_line_3d .....	606
set_port_default .....	607
set_port_default_list .....	608
set_pulse_picking .....	609
set_pulse_picking_length .....	609
set_pulse_picking_list .....	610
set_qswitch_delay .....	610
set_qswitch_delay_list .....	610
set_rot_center .....	611



set_rot_center_list .....	611
set_rtc4_mode .....	612
set_rtc5_mode .....	613
set_scale .....	614
set_scale_list .....	614
set_scanner_delays .....	615
set_serial .....	615
set_serial_step .....	616
set_serial_step_list .....	616
set_sky_writing .....	617
set_sky_writing_limit .....	618
set_sky_writing_limit_list .....	618
set_sky_writing_list .....	618
set_sky_writing_mode .....	619
set_sky_writing_mode_list .....	620
set_sky_writing_para .....	621
set_sky_writing_para_list .....	623
set_softstart_level .....	624
set_softstart_level_list .....	625
set_softstart_mode .....	626
set_softstart_mode_list .....	627
set_standby .....	628
set_standby_list .....	629
set_start_list .....	630
set_start_list_1 .....	630
set_start_list_2 .....	630
set_start_list_pos .....	631
set_sub_pointer .....	632
set_text_table_pointer .....	633
set_trigger .....	634
set_trigger4 .....	642
set_vector_control .....	643
set_verify .....	645
set_wait .....	646
set_wobble .....	647
set_wobble_control .....	649
set_wobble_direction .....	650
set_wobble_mode .....	651
set_wobble_offset .....	653
set_wobble_vector .....	654
set_zoom .....	656
set_zoom_list .....	656
simulate_encoder .....	657
simulate_ext_start .....	658
simulate_ext_start_ctrl .....	659
simulate_ext_stop .....	660
start_loop .....	661
stepper_abs .....	662
stepper_abs_list .....	663
stepper_abs_no .....	663
stepper_abs_no_list .....	664
stepper_control .....	665



stepper_control_list .....	665
stepper_disable_switch .....	666
stepper_enable .....	667
stepper_enable_list .....	667
stepper_init .....	668
stepper_rel .....	670
stepper_rel_list .....	670
stepper_rel_no .....	671
stepper_rel_no_list .....	671
stepper_wait .....	672
stop_execution .....	673
stop_list .....	673
stop_trigger .....	674
store_encoder .....	674
sub_call .....	675
sub_call_abs .....	676
sub_call_abs_cond .....	676
sub_call_abs_repeat .....	677
sub_call_cond .....	677
sub_call_repeat .....	678
switch_iport .....	679
sync_slaves .....	680
time_fix .....	682
time_fix_f .....	682
time_fix_f_off .....	683
time_update .....	684
timed_arc_abs .....	685
timed_arc_rel .....	686
timed_jump_abs .....	687
timed_jump_abs_3d .....	688
timed_jump_rel .....	689
timed_jump_rel_3d .....	690
timed_mark_abs .....	691
timed_mark_abs_3d .....	692
timed_mark_rel .....	693
timed_mark_rel_3d .....	694
timed_para_jump_abs .....	695
timed_para_jump_abs_3d .....	696
timed_para_jump_rel .....	697
timed_para_jump_rel_3d .....	698
timed_para_mark_abs .....	699
timed_para_mark_abs_3d .....	700
timed_para_mark_rel .....	701
timed_para_mark_rel_3d .....	702
transform .....	703
upload_transform .....	706
verify_checksum .....	708
wait_for_encoder .....	709
wait_for_encoder_in_range .....	710
wait_for_encoder_mode .....	711
wait_for_mcbsp .....	713
wwrite_8bit_port .....	714



write_8bit_port_list .....	714
write_abc_to_file .....	715
write_da_1 .....	716
write_da_1_list .....	716
write_da_2 .....	717
write_da_2_list .....	717
write_da_x .....	718
write_da_x_list .....	719
write_hi_pos .....	720
write_io_port .....	721
write_io_port_list .....	721
write_io_port_mask .....	722
write_io_port_mask_list .....	722
10.3 Unsupported RTC2/RTC3/RTC4 Commands .....	723
<b>11 Demo Programs .....</b>	<b>728</b>
<b>12 Troubleshooting .....</b>	<b>729</b>
<b>13 Customer Service .....</b>	<b>731</b>
13.1 Servicing and Repairs .....	731
13.2 Warranty .....	731
13.3 Contacting SCANLAB .....	731
13.4 Product Disposal .....	731
<b>14 Legal .....</b>	<b>732</b>
14.1 EU Declaration of Conformity – RTC5 PCI Board .....	732
14.2 Compliance with FCC Rules .....	733
<b>15 Technical Specifications – RTC5 PCI Board .....</b>	<b>734</b>
<b>16 Appendix A: RTC5 PCIe Board .....</b>	<b>738</b>
16.1 Product Overview .....	738
16.1.1 Intended Use – Comparison to the RTC5 PCI Board .....	738
16.1.2 System Requirements .....	738
Hardware .....	738
Software .....	738
16.1.3 Optional Functionality .....	738
16.1.4 Labeling .....	739
16.1.5 Type Identification .....	739
16.1.6 Unpacking Instructions and Typical Scope of Delivery .....	739
Delivered Software .....	739
16.1.7 Accessories .....	739
16.1.8 Supplementary Software .....	739
16.2 RTC5 PCIe Board – Layout and Interfaces .....	740
16.2.1 Layout – Upper Side .....	740
16.2.2 Layout – Lower Side .....	741
16.2.3 Interface to PC .....	741
16.2.4 SPI/I2C Socket Connector .....	742
McBSP Interface .....	742
Analog Input Ports .....	742
16.3 Installation and Start-Up .....	743
16.4 Legal .....	744
16.4.1 EU Declaration of Conformity – RTC5 PCIe Board .....	744
16.4.2 Compliance with FCC Rules .....	745
16.5 Technical Specifications – RTC5 PCIe Board .....	746



<b>17 Appendix B: iDRIVE Scan Systems</b>	
– Control Commands and Signals Transmitted to RTC Control Boards .....	<b>747</b>
<b>18 Appendix C: Change Index .....</b>	<b>790</b>



## 1 About this Manual

This manual describes the available SCANLAB RTC5 Boards and their usage for synchronous control of scan systems, lasers and peripheral equipment.

The main chapters of this manual use the RTC5 PCI Board to exemplify. For a product overview see [Chapter 2 "Product Overview", page 27](#).

Other RTC5 boards variants are described in the Appendix:

- RTC5 PCIe Board, [page 738](#)

The manual is a part of the product. Read these instructions carefully before you proceed with installing and operating the RTC5 Board.

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", page 21](#).

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

This manual refers to:

- [RTC5\\_Software\\_2024-09-27](#)<sup>(1)(2)(3)</sup>

DLL file for 32 bit user programs <sup>(a)</sup>	<a href="#">RTC5DLL.dll</a>	Version 551 (DLL 551) <sup>(b)</sup>
DLL file for 64 bit user programs <sup>(a)</sup>	<a href="#">RTC5DLLx64.dll</a>	
Program file for the <a href="#">DSP</a>	<a href="#">RTC5OUT.out</a>	Version 553 (OUT 553 <sup>(b)</sup> )
Firmware file for the <a href="#">FPGA</a>	<a href="#">RTC5RBF.rbf</a>	Version 534 (RBF 534 <sup>(b)</sup> )
Auxiliary file	<a href="#">RTC5DAT.dat</a>	Version 500 (DAT 500 <sup>(b)</sup> )

(a) Software for laser-scan processes, which controls RTC5 boards based on this [RTC5 DLL](#) file is consistently denoted as "user program" in this manual.

(b) Abbreviated version in this manual.

### 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) Software changes prior to [RTC5\\_Software\\_Release\\_2015\\_07\\_15](#) are no longer described in this manual.
- (2) The version numbers of the supplied [RTC5 DLL](#) and [RTC5 files](#) are indicated in the names of the corresponding zip files, see [Section "Folder RTC5 Files", page 28](#).
- (3) To identify the version numbers of your files after installation, use [get\\_dll\\_version](#), [get\\_hex\\_version](#) and [get\\_RTC\\_version](#).



## 1.2 Related Documents

- "Calibrating a 3-Axis Laser Scan System" Manual
- "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual



### 1.3 Glossary and Abbreviations

["*]mark["*] Command	All commands with "mark" as part of their names. See <a href="#">Section "Mark Commands", page 125</a> .
["*]para["*] Command	All commands with "para" as part of their names. See <a href="#">Section "[*]Para[*] Commands", page 127</a> .
3D image field	Synonym: working volume (process volume). See <a href="#">Chapter 7.3.4 "3D Image Field", page 159</a> .
"Arc" command	Umbrella term for <a href="#">Arc Commands</a> and <a href="#">Ellipse Commands</a> .
BCD	Binary Coded Decimal.
DSP	Digital signal processor on the RTC5 board.
Dynamic focusing unit	This includes, for example, the following SCANLAB products: varioSCAN, varioSCAN <sub>de</sub> , varioSCAN FC and varioSCAN FLEX, excelliSHIFT.
EEPROM	Non-volatile memory of the RTC5 board.
FPGA	Field programmable gate array on the RTC5 board.
Hardware reset	New start after powering the RTC5 board. Synonym: "power up", "power cycle".
Hard jump	Direct output to a specified position. Decomposition into <a href="#">Microsteps</a> into a single 10 µs clock cycle.
High-Bandwidth Return Channel Multiplexing	Functionality not yet released.
iDRIVE scan systems	In this manual, the term subsumes, for example, the following SCANLAB products: intelliSCAN, intelliSCAN <sub>de</sub> , intelliSCAN <sub>se</sub> , intelliDRILL, intellicube, intelliWELD, varioSCAN <sub>de</sub> , powerSCAN II 50i, excelliSCAN.
Image field	Synonym: <a href="#">Working field</a> .
intelliSCAN	In this manual, the term subsumes, for example, the following SCANLAB products: intelliSCAN, intelliSCAN <sub>de</sub> , intelliSCAN <sub>se</sub> , intelliDRILL, intellicube, intelliWELD, powerSCAN II 50i.
Jump command	Serves to move the scan system axes to a new position while the laser is off. See also <a href="#">Section "Jump Commands", page 125</a> . See <a href="#">2D Jump Commands, page 285</a> and <a href="#">3D Jump Commands, page 285</a> .
Laser Control Signals	LASER1, LASER2, LASERON. See, for example, <a href="#">Chapter 7.4.1 "Enabling, Activating and Switching Laser Control Signals", page 173</a> .



LSB	Least Significant Bit.
Mark command	Serves to perform marking motions while the laser is switched <i>on</i> . Examples: mark, arc and ellipse. See <a href="#">2D Mark Commands, page 285</a> and <a href="#">3D Mark Commands<sup>(1)</sup>, page 285</a> .
McBSP	Multi channel Buffered Serial Port.
Microstep	See <a href="#">Chapter 7.1.2 "Microstepping", page 128</a> .
Microvector	The term refers to <a href="#">micro_vector[*] commands</a> , see <a href="#">Chapter 8.8 "micro_vector[*] Commands", page 249</a> .
<b>micro_vector[*] command</b>	All commands with "micro_vector" as part of their names. See <a href="#">Chapter 8.8 "micro_vector[*] Commands", page 249</a> .
MSB	Most Significant Bit.
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 <a href="#">"Pixel Output Mode"</a> . See <a href="#">Chapter 8.7 "Pixel Output Mode", page 244</a> .
Polyline	A direct sequence of <a href="#">[*]mark[*] Commands</a> or <a href="#">"Arc" commands</a> . The marking is continuous. Synonym: polygonal chain, polygonal line, polygonal traversal.
PosAck	"Position Acknowledge", see <a href="#">PosAck signal</a> .
<a href="#">PosAck signal</a>	<ul style="list-style-type: none"> <li>As of scan system firmware <math>\geq</math> 2001.</li> <li><a href="#">PosAck signal</a>-capable SCANLAB scan systems, for example, <a href="#">intelliSCAN</a></li> <li>Refers to the meaning of: <ul style="list-style-type: none"> <li>Bit #12, Bit #4, Bit #11, Bit #3 of the XY2-100 status word transferred with 16-bit protocol</li> <li>Bit #16, Bit #8, Bit #15, Bit #7 of the XY2-100 status word transferred with 20-bit protocol</li> </ul> </li> <li>These bits are set, if the position errors of x axis and y axis are in the allowed range: <ul style="list-style-type: none"> <li>See also <a href="#">"excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual, Chapter 2.1.3 "TrAck Signal", page 15</a>.</li> <li>For more information, refer to the corresponding scan head manuals.</li> </ul> </li> <li>Compare to <a href="#">TrAck signal</a>.</li> </ul>
Preprocessing Servo Control	Servo control (of newer generation) in <a href="#">Preprocessing Systems</a> = with <a href="#">Tracking error</a> and with preprocessing.
Preprocessing System	SCANLAB scan system with <a href="#">Preprocessing Servo Control</a> . Compare to <a href="#">Tracking error System</a> , <a href="#">SCANAhead System</a> .

Processing-on-the-fly session	A section of a list that uses external inputs (encoder pulses, <a href="#">McBSP</a> transmissions) to correct moving workpiece positions.
Reset of the RTC5 board	Synonym: <a href="#">Software reset</a> .
RTC5 files	See <a href="#">RTC5 files, page 28</a> .
SCANAhead Servo Control	Equipment feature of certain SCANLAB scan systems (= " <a href="#">SCANAhead Systems</a> "), essentially based on an ISB1 as a servo board for the 2 galvanometer scanner with corresponding firmware and parameterization. Refer to " <a href="#">excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards</a> " Manual.
SCANAhead System	SCANLAB scan system with <a href="#">SCANAhead Servo Control</a> , for example, excelliSCAN series. Compare to <a href="#">Tracking error System</a> , <a href="#">Preprocessing System</a> .
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 RTC5 board</a> .
SPI	Serial Peripheral Interface.
TrAck	"Trajectory Acknowledge", see <a href="#">TrAck signal</a> .
TrAck signal	<ul style="list-style-type: none"> <li>As of scan system firmware <math>\geq 5102 + \geq 5112</math>.</li> <li><a href="#">TrAck signal</a>-capable SCANLAB scan systems, for example, <a href="#">SCANAhead Systems</a></li> <li>Refers to the meaning of: <ul style="list-style-type: none"> <li>Bit #12, Bit #4, Bit #11, Bit #3 of the XY2-100 status word transferred with 16-bit protocol</li> <li>Bit #16, Bit #8, Bit #15, Bit #7 of the XY2-100 status word transferred with 20-bit protocol</li> </ul> </li> <li>These bits are set, if the <a href="#">Trajectory errors</a> of x axis and y axis are in the allowed range: <ul style="list-style-type: none"> <li>See also "<a href="#">excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards</a>" Manual, Chapter 2.1.3 "TrAck Signal", page 15.</li> <li>For more information, refer to the corresponding scan head manuals.</li> </ul> </li> <li>Compare to <a href="#">PosAck signal</a>.</li> </ul>
Tracking error	Time difference between the planned and actual reaching of a certain mirror position at constant speed.
Tracking error Servo Control	The "conventional" servo control of <a href="#">Tracking error Systems</a> .
Tracking error System	SCANLAB scan system with conventional control (1st generation) = with <a href="#">Tracking error</a> and without preprocessing. For example, intelliSCAN III. Compare to <a href="#">SCANAhead System</a> , <a href="#">Preprocessing System</a> .
Trajectory	In this manual: curve with $10 \mu\text{s}$ parameterization.



Trajectory error	Deviation of the actual <b>Trajectory</b> from the set <b>Trajectory</b> .
Vector command	Umbrella term for <b>Jump Commands</b> and <b>Mark Commands</b> .
Working field	Synonym: <b>Image field</b> .

## 2 Product Overview

### 2.1 Labeling

The serial number of the RTC5 PCI Board is printed on a label attached to the board (format: "RTC SN...").

The ID number and configuration of the board are described in the packaging list, see [Chapter 2.6 "Options", page 34](#) and [Chapter 2.7 "Jumper Settings and Type Identification", page 35](#).

### 2.2 Unpacking Instructions and Typical Scope of Delivery

- (1) Carefully remove the RTC5 Board from the package.
- (2) Keep the packaging, including the antistatic bag the RTC5 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 RTC5 PCI Board and a data CD (with the RTC5 software package, see below, and this manual). Possibly additional components are also contained, see [Chapter 2.8 "Accessories for the RTC5 PCI Board", page 37](#).

### 2.3 Delivered RTC5 Software Package

The delivered RTC5 software package contains the RTC5 board driver for the 32-bit and 64-bit versions of the operating systems Microsoft Windows 10, 8, 7.

The data CD contains all files as unzipped versions. The complete RTC5 software package is also delivered zipped for easy identification and management of different software versions:

- RTC5\_Software\_<Date>.zip

The content of the RTC5 software package is as follows:

- Readme.txt  
Description in English
- Liesmich.txt  
Description in German

#### Folder Correction Files

- Cor\_1to1.ct5  
1to1 correction file<sup>(1)(2)</sup>

#### Folder CorrectionFileConverter

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

(1) Additional correction files (D2\_XXX.CT5, D3\_YYY.CT5) and \*\_ReadMe.txt file(s) are not part of the RTC5 software package.

(2) See also [Section "1to1 Correction Tables", page 166](#).

### Folder DemoFiles

- Source codes in C of Demo1...Demo7  
(Demo1.cpp...Demo7.cpp)
- Demo1 compiled to executable files for 32-bit as well as 64-bit
- Source code in C# of Demo3 (Class1.cs)
- Project generating file for CMake  
(CMakeLists.txt) and the corresponding include file (RTC\_Variables.cmake)<sup>(1)</sup>

### Folder HPGL

- Hpgl.exe is a Win32-based HPGL-to-RTC5 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 **RTC5 files** and the **RTC5DLL.dll** in the same folder.

### Folder iSCANConfig

- iSCANcfg.exe is a Win32-based diagnosis and configuration program for iDRIVE scan systems<sup>(2)</sup>. RTC4, RTC5 and RTC6 (PCI/PCIe as well as Ethernet variants) are supported.
- Needs **RTC5 files** and the **RTC5DLL.dll** in the same folder 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 RTC5 Driver

(RTC5 board driver for Windows)

- RTC5DRV.sys, RTC5DRVx64.sys, RTC5DRVx86.sys  
**RTC5 driver files**
- RTC5DRV.inf  
**Installation file (setup information)**
- RTC5DRV.cat, rtc5drvx64.cat, rtc5drvx86.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 RTC5 Files

- **RTC5 DLL**
  - RTC5DLL.dll  
**Win32-based RTC5 dynamic link library**
  - RTC5DLLx64.dll  
**Win64-based RTC5 dynamic link library**
- **RTC5 files**
  - RTC5OUT.out  
**Program file for the DSP**
  - RTC5RBF.rbf  
**Firmware file for the FPGA**
  - RTC5DAT.dat  
**Binary auxiliary file**
- **Utility Files for C, C++ and C#**
  - RTC5expl.c  
C functions for **RTC5 DLL** handling for explicit linking
  - RTC5expl.h  
C function prototypes of the RTC5 for explicit linking of the **RTC5 DLL**
  - RTC5DLL.lib  
Visual C++ import libraries for implicit linking of the **RTC5 DLL** for Win32-based user programs
  - RTC5DLLx64.lib  
Visual C++ import libraries for implicit linking of the **RTC5 DLL** for Win64-based user programs
- RTC5\_Software\_RevisionHistory\_<Date>.pdf  
**Description of RTC5 software package changes in English**
- RTC5\_Software\_Aenderungshistorie\_<Date>.pdf  
**Description of RTC5 software package changes in German**

(1) These are CMake (cross-platform make) files to easily generate executable demo programs. CMake (<https://cmake.org>) is a free and open-source cross-platform programming tool for the development and creating software. Using Cmake, make files and projects for many integrated development environments and compilers can be generated by script files (CMakeLists.txt).

(2) See Glossary entry on [page 23](#).



- RTC5impl.h
  - C function prototypes of the RTC5 for implicit linking of the **RTC5 DLL**
- RTC5impl.hpp
  - C++ function prototypes of the RTC5 for implicit linking of the **RTC5 DLL**
- RTC5Wrap.cs
  - Import declarations of the wrapper class for implicit linking in C#
- Utility File for Delphi
  - RTC5Import.pas
    - Import declarations for Delphi

Differing versions of **RTC5 files** and **RTC5 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):

- RTC5DAT\_<current DAT version number>.zip
- RTC5DLL\_<current DLL version number>.zip
  - (includes DLLs and related utility files)
- RTC5OUT\_<current OUT version number>.zip
- RTC5RBF\_<current RBF version number>.zip

### Folder RTC5 Tools

- SleepMode.cmd
  - Script to deactivate *all* Windows sleep and hibernate modes.
- ReadMe\_SleepMode.pdf
  - Description of the use of **SleepMode.cmd**

## 2.4 Intended Use

The SCANLAB RTC5 PCI Board and its associated RTC5 software package is intended for synchronous real-time control of scan systems, lasers and peripheral equipment by a Windows PC with a PCI bus interface.

RTC5 boards are available in different hardware variants, [Chapter 1 "About this Manual", page 21](#).

The delivered [RTC5 DLL](#) provides an extensive command set for control. This allows a quick and flexible software development for laser-scan processes.

The RTC5 PCI 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 RTC5 PCI Board and start their execution at a later time. Command execution by the RTC5 PCI Board 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 RTC5 PCI 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 RTC5 PCI 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 "RTC5 PCI Board – Layout and Interfaces", page 51](#)) and associated software commands.

As many RTC5 PCI Boards can be operated in one PC at the same time as the PC provides PCI slots.

Moreover, the [RTC5 DLL](#) allows multi-threading as well as multi-processing. Therefore, several user programs can even 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 [RTC5 DLL](#) serializes these accesses automatically.

RTC5 PCI Boards are available in various configurations, see [Chapter 2.6 "Options", page 34](#) and [Chapter 2.7 "Jumper Settings and Type Identification", page 35](#).

The RTC5 PCI Board interfaces are described on [Chapter 4 "RTC5 PCI Board – Layout and Interfaces", page 51](#), installation and start-up on [Chapter 5 "Installation and Start-Up", page 77](#), and programming on [Chapter 6 "Developing RTC5 User Programs", page 83](#). Individual command descriptions are listed beginning with [Chapter 10 "RTC5 Commands", page 278](#).

The technical specifications of the RTC5 PCI Board are summarized in [Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734](#).



## Caution!

- Do not operate the RTC5 PCI Board outside of the PC.
- The RTC5 PCI 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 RTC5 PCI 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 unfavorable 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 RTC5 PCI 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.

## 2.5 System Requirements

### 2.5.1 Hardware

The RTC5 PCI Board requires a Windows PC with a PCI bus interface and at least one free PCI slot.

RTC5 PCI Boards intended for synchronized master/slave operation should (recommended) be installed in adjacent PCI slots.

### 2.5.2 Software

To operate the RTC5 PCI Board, RTC5 board driver and [RTC5 DLL](#) files for Microsoft Windows operating systems must be used. These are included in the scope of delivery (RTC5 software package). For the supported Windows versions, see [Chapter 2.3 "Delivered RTC5 Software Package", page 27](#).

The RTC5 board driver supports the plug and play capability of the RTC5 PCI Board as well as the simultaneous operation of any number of RTC5 PCI Boards. The [RTC5 DLL](#) files contain the command set to control laser scan systems.

#### Notice!

- The RTC5 PCI Board does not support power-saving modes, that switch off power to the PCI bus. Accordingly, you must disable standby or sleep modes of the operating system. See also [Section "Notes", page 33](#).

#### 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 RTC5 board driver.
    - (2)Run `ScanlabClassChecker.cmd` as Administrator (part of the delivered RTC5 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

- RTC5 software package version 2013\_02\_21 and later contain (newer) WDF<sup>(1)</sup> drivers (version 6.1.7600.16385).  
Earlier RTC5 software packages still contain a WDM<sup>(2)</sup> driver (outdated today; version 1.0.4.0 or version 2.0.6.0).
- DLL  $\leq$  533 are *not* compatible with the WDF drivers.
- DLL  $\geq$  535 are even compatible with the WDM drivers.
- In comparison to the outdated WDM driver, the WDF driver offers the following new functionality: If `init_RTC5_DLL` is called, then the WDF driver prevents automatic activation of standby or sleep modes (continuously until the next system restart). This enables RTC5 PCI Boards to continue processing lists autonomously even when the initiating program has already terminated.  
However, standby or sleep modes cannot be prevented if triggered manually or by discharged batteries. Afterward, loaded lists and other settings of the RTC5 PCI Board are lost. After a wake-up, the RTC5 PCI Board might no longer be correctly addressable. Before calling `init_RTC5_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 RTC5 software package deactivates *all* sleep and hibernation modes, see `ReadMe_SleepMode.pdf`.

- RTC5 software packages versions *newer* than 2022-03-09 (that is, with  $\geq$  DLL 551)
  - RTC5 board driver and **RTC5 DLL** files are designed for 32-bit as well as 64-bit versions of Microsoft Windows 10, 8, 7. **RTC5 DLL** files from these packages *do not* support Microsoft Vista, XP SP2, XP SP3 and earlier any more.
- RTC5 software packages version 2013\_02\_21 through 2022-03-09 (with  $\geq$  DLL 535...DLL 550)
  - RTC5 board driver and **RTC5 DLL** files were designed for 32-bit as well as 64-bit versions of Microsoft Windows 10, 8, 7, Vista, XP SP2, XP SP3. **RTC5 DLL** files from these packages *did not* support Microsoft Windows 2000 and XP  $\leq$  SP1 any more.
- RTC5 software packages version 2011\_09\_29 through 2012\_09\_05 (with DLL 528...DLL 535)
  - RTC5 board driver and **RTC5 DLL** files were designed for Microsoft 32-bit as well as 64-bit versions of Windows 7, Vista, XP SP2, XP SP3. **RTC5 DLL** files from these packages *did not* support Microsoft Windows 2000 and XP  $\leq$  SP1 any more.
- RTC5 software packages of version 2011\_07\_27 and earlier (with  $\leq$  DLL 527)
  - RTC5 board driver and **RTC5 DLL** files were designed for 32-bit versions of Microsoft Windows 7, Vista, XP and 2000.

(1) Windows Driver Framework. Current technology.

(2) Windows Driver Model. Legacy technology.

## 2.6 Options

RTC5 PCI Boards can be equipped with different options (features). 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 34](#).

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

The following options are available for RTC5 PCI 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 RTC5 PCI 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 axis and y axis (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 RTC5 PCI 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 54](#) and [Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 221](#).

- **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, see [Section "Laser Control Signals", page 60](#), and [Section "Laser Control Signals", page 68](#).

### Notes

- Each RTC5 PCI 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"](#)
  - "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 RTC5 PCI Board: you can activate them by special upgrade software available from SCANLAB. When requesting the upgrade, you will need to supply the serial number of your board.
- For retrofitting your RTC5 PCI Board with the extra DC/DC converter (optoelectronic coupler) you need to send it to SCANLAB.
- For optical data transfer between the RTC5 PCI Board and the scan system, solely a suitable data cable is required, see [Chapter 4.5.3 "Data Cables \(Accessories\)", page 58](#). Neither the RTC5 PCI Board side nor the scan system side requires a special optical data interface for that.

## 2.7 Jumper Settings and Type Identification

SCANLAB ships RTC5 PCI Boards in various jumper configurations. Jumpers are connections which are either open or closed. The *factory* solder jumper configuration of an RTC5 PCI Board can be identified by its article number.

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

- “RTC5 PCI TYPE 000”
  - No signals (that is, all solder jumpers are open)
- “RTC5 PCI 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.

Trained 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 2.7.1 “Jumper JP1 – Configuring the Output Signal Level at the EXTENSION 1 Socket Connector”, page 36](#).
- (2) For the solder jumper setting, see [Chapter 2.7.2 “Jumper JP2...JP8 – Configuring Pin15 and Pin17 at the EXTENSION 2 Socket Connector”, page 36](#).
- (3) For the solder jumper setting, see [Chapter 2.7.2 “Jumper JP2...JP8 – Configuring Pin15 and Pin17 at the EXTENSION 2 Socket Connector”, page 36](#).

## 2.7.1 Jumper JP1 – Configuring the Output Signal Level at the EXTENSION 1 Socket Connector

By Jumper **JP1** on the lower side of the RTC5, see **Figure 6**, the level of all output signals at the EXTENSION 1 socket connector can be configured for 5 V or 3.3 V.

See also Section “Configuring the Output Signal Level”, page 65.

Jumper JP1	Signal Level
Position 1-2*	 5 V
Position 2-3*	 3.3 V
open	 no output signals

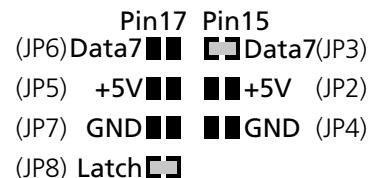
\*Caution: make sure that *only one* position is closed in this solder jumper field. Other combinations are not allowed and cause damage to the board!

## 2.7.2 Jumper JP2...JP8 – Configuring Pin15 and Pin17 at the EXTENSION 2 Socket Connector

By the jumpers **JP2**...**JP8** on the lower side of the RTC5, see **Figure 6**, pins (15) and (17) of the EXTENSION 2 socket connector can be configured. See also **Section "Configuration by Solder Jumpers"**, **page 67**).

The most significant bit (DATA7) of the 8-bit output value can be assigned to pin (15) or to pin (17). Alternatively, each of the two pins can be set permanently to +5 V (HIGH) or to GND (LOW level). Alternatively, pin (17) can be configured for the LATCH signal by closing the correspondingly labeled jumper.

Figure 1 shows an example jumper configuration assigning DATA7 to pin (15) and the LATCH signal to pin (17).



Example configuration for jumpers JP2...JP8: Pin (15): DATA7,  
Pin (17): LATCH signal

## Notice!

- Make sure that not more than **one** of the three jumpers for pin (15) is closed. Also make sure that not more than **one** of the 4 jumpers for pin (17) is closed.  
Other combinations are not allowed and cause damage to the board!



## 2.8 Accessories for the RTC5 PCI Board

Only hardware extensions from SCANLAB should be used in combination with the RTC5 PCI Board. In addition to the RTC5 PCI Board and its software package, the following accessories can be obtained:

- XY2-100 Converter, page 37
- Laser Adapter, page 37
- Data Cables, page 37
- Slot Cover with 9-pin D-SUB Connector for 2. SCANHEAD Socket Connector, page 38
- Slot Cover with 15-pin D-SUB Connector for MARKING ON THE FLY Socket Connector, page 38
- Slot Cover with 15-pin D-SUB Connector and 9-pin D-SUB Connector, page 39
- ADC Add-On Board, page 39
- Extension Board "RTC5/6 varioSCAN FLEX Extension", page 39

### 2.8.1 XY2-100 Converter

The XY2-100 Converter (Accessory) allows the RTC5 PCI Board to control scan systems which are equipped with a conventional XY2-100 interface.

### 2.8.2 Laser Adapter

The SCANLAB laser adapter is plugged into the 15-pin LASER connector of the RTC5 PCI 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. See also [Section "Laser Adapter \(Accessory\)", page 63](#).

### 2.8.3 Data Cables

To connect the RTC5 PCI 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 58](#).

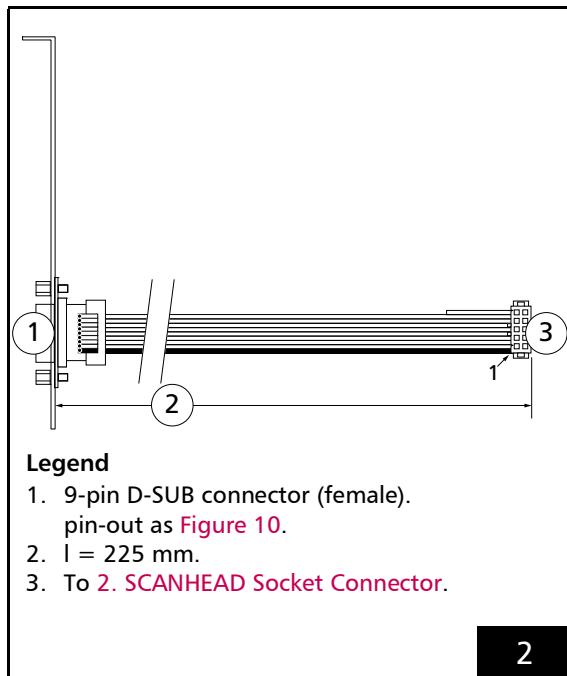
#### 2.8.4 Slot Cover with 9-pin D-SUB Connector for 2. SCANHEAD Socket Connector

For connecting a second scan head or a z axis to the 2. SCANHEAD Socket Connector a slot cover is available, see Figure 2:

- #0115132

Its 9-pin D-SUB connector has the same pin-out as the SCANHEAD Connector, see Figure 10.

See also Section "Second Scan Head Slot Cover (Accessory)", page 55.



Second Scan Head Slot Cover (Accessory) #0115132.

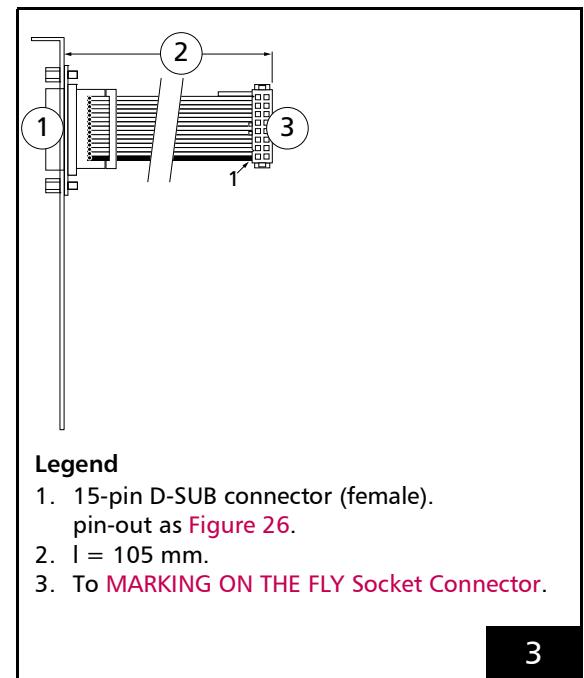
#### 2.8.5 Slot Cover with 15-pin D-SUB Connector for MARKING ON THE FLY Socket Connector

For using the inputs and signals of the MARKING ON THE FLY Socket Connector, a slot cover is available, see Figure 3:

- 0109272

The pin-out of its 15-pin D-SUB connector (female) is shown in Figure 26.

See also Section "MARKING ON THE FLY Slot Cover (Accessory)", page 70.

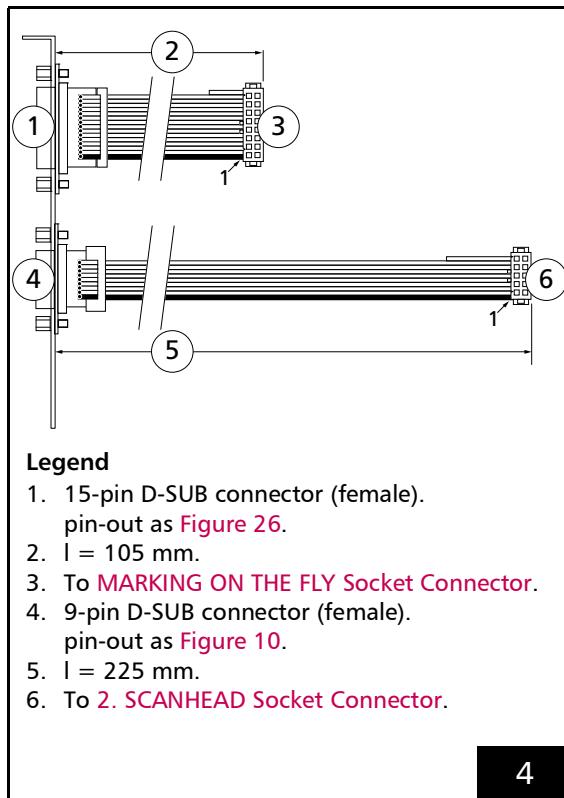


MARKING ON THE FLY Slot Cover (Accessory) 0109272.

## 2.8.6 Slot Cover with 15-pin D-SUB Connector and 9-pin D-SUB Connector

The connectors of #0115132 and 0109272 together on a single bracket (for the same purpose as these) features bracket, see Figure 4:

- #0130209



4

Slot cover #0130209.

## 2.8.7 ADC Add-On Board

Only for the RTC5 PCI Board<sup>(1)</sup>, the SCANLAB **ADC Add-On Board** (#0121126) is available, see Figure 33.

With this add-on board installed, the RTC5 PCI Board provides three 10 V analog inputs, see Chapter 4.6.8 "Analog Inputs (Accessory)", page 75.

The RTC5 PCIe Board provides two 10 V analog inputs even without add-on board, see Chapter 16.2.4 "SPI/I2C Socket Connector", page 742.

## 2.8.8 Extension Board "RTC5/6 varioSCAN FLEX Extension"

For RTC5/6 Boards, SCANLAB offers the "RTC5/6 varioSCAN 40 FLEX Extension" extension board (#0128683)<sup>(2)</sup>.

It has been specially developed to control analog and digital varioSCAN 40 FLEX devices by the SCANLAB laserDESK software. A position change of the varioSCAN 40 FLEX focusing optics is caused by the step motor which changes the working distance of the 3D scan system in the end.

For further information, refer to the pertaining manual "Installation and Operation RTC5/6 varioSCAN FLEX Extension for RTC5 and RTC6 Control Boards".

### Notes

- "RTC5/6 varioSCAN 40 FLEX Extension" extension board (#0128683) does not support **move\_to**<sup>(3)</sup>.

(1) Not for use with the RTC5 PCIe Board.

(2) In contrast to extension board "RTC4 STEP MOTOR EXTENSION" (#0112097) – which can also be used with RTC5 Boards – it has the advantage that the EXTENSION 1 socket connector remains unoccupied.

(3) **move\_to** has been introduced for extension board "RTC4 STEP MOTOR EXTENSION" (#0112097).



## 2.9 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 "Image 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 RTC5 functions are supported.<sup>(1)</sup>

For further information on laserDESK, refer to the SCANLAB homepage.

## 2.10 Notes for RTC4 Users

This chapter provides an overview of key changes introduced by the RTC5 PCI Board in comparison to the RTC4 PCI Board.

For example, [Chapter 2.10.2 "Porting RTC4 Source Code to the RTC5 PCI Board", page 42](#) discusses a possible approach to porting RTC4 programs to run on the RTC5.

The individual command descriptions in particular note changes.

### 2.10.1 Hardware Changes

When migrating from the RTC4 to the RTC5, you need to consider the following hardware changes for correct cabling of the system components.

#### Controlling Scan Systems

- To control a scan system with the XY2-100 or XY2-100 Enhanced interface, you also need the [XY2-100 Converter \(Accessory\)](#).
- The RTC5 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 58](#).
- To use the second scan head connector, you need a different adapter cable (including slot cover available from SCANLAB), see [Section "Second Scan Head Slot Cover \(Accessory\)", page 55](#).
- For controlling a 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 RTC5 Boards, see [Chapter 8.5.2 "3D Scan Systems", page 222](#).

(1) See also [Notice!](#), page 60.

## Controlling the Laser

- The female D-SUB laser connector at the RTC5 slot cover has 15 pins (the RTC4 laser connector, on the other hand, has 9 pins). The pin-outs are not jumper-configurable.
  - The RTC5 does not require jumpers X6 and X7 of the RTC4 because all signals are available at the RTC5 laser connector.
  - The voltage range of the analog outputs is always 0...10 V (0 V...2.50 V is no longer supported; the RTC4's jumper X3 does not exist on the RTC5).
- If you want to connect a laser to the RTC5 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 laser adapter from SCANLAB (see [Section "Laser Adapter \(Accessory\)", page 63](#)), 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 software-configured (see [set\\_laser\\_control](#)).
  - Jumper X10 of the RTC4 does not exist on the RTC5.

## EXTENSION 1 Socket Connector

- The RTC5 EXTENSION 1 socket connector is – except for the additionally provided signals at pins 33-35 – identical to the EXTENSION 1 socket connector of the RTC4 (see [Figure 22](#)).
- With the RTC5, 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 65](#)).

## EXTENSION 2 Socket Connector

- The RTC4 EXTENSION 2 socket connector does not exist on the RTC5. Accordingly, no I/O extension board can be attached to the RTC5.
- The RTC5 EXTENSION 2 socket connector is – except for the optional LATCH signal at pin 17 – identical to the LASER EXTENSION socket connector of the RTC4 (see [Figure 24](#)).
  - The RTC5 provides jumpers **JP2...JP8** for configuring pin-outs (these jumpers are equivalent to jumpers X8 and X9 of the RTC4) (see also [page 67](#)).

## MARKING ON THE FLY Socket Connector

- The RTC5 MARKING ON THE FLY socket connector is identical to the RTC4 MARKING ON THE FLY socket connector, see [Figure 25](#). Observe the safety notice on encoder counter direction, [page 49](#).

## Other Hardware Interfaces

- PCI bus requirements are identical to those of the RTC4.
- PCI-Express version only available for the RTC5.
- The following interfaces only exist on the RTC5:
  - MASTER and SLAVE (see [Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 53](#))
  - RS 232 (see [Chapter 4.6.5 "RS232 Socket Connector", page 70](#))
  - SPI / I<sup>2</sup>C / McBSP (see [Chapter 4.6.6 "SPI / I<sup>2</sup>C Socket Connector", page 71](#))
  - STEPPER MOTOR (see [Chapter 4.6.7 "STEPPER MOTOR Socket Connector", page 75](#))

## 2.10.2 Porting RTC4 Source Code to the RTC5 PCI Board

User programs written for the RTC4 can only run on the RTC5 after suitable code revision. This applies even when the actual program flow shall remain unchanged.

### Changed Initialization

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

- At the beginning of the user program, a `init_rt5_dll` command must be inserted for initializing the **RTC5 DLL** and **RTC5 board management** (see [Chapter 6.2.3 "Initializing the RTC5 DLL and Board Management", page 86](#)).
- 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, the `set_laser_control` command 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 RTC2/RTC3/RTC4 Commands", page 723](#).

Changed or enhanced RTC4 commands need to be handled differently in the program (for example, by modifying supplied parameter values or evaluating returned values differently). Changes to supported commands are listed in the individual command descriptions (in [Chapter 10.2 "RTC5 Command Set", page 290](#)) under the heading "RTC4→RTC5". RTC4 commands that need to be replaced or checked are:

- |                                |               |
|--------------------------------|---------------|
| • <code>aut_change</code>      | not supported |
| • <code>auto_cal</code>        | changed       |
| • <code>auto_change_pos</code> | changed       |

• <code>control_command</code>	changed
• <code>dsp_start</code>	not supported
• <code>get_head_status</code>	changed
• <code>get_hi_data</code>	changed
• <code>get_list_space</code>	changed
• <code>get_marking_info</code>	changed
• <code>get_RTC_version</code>	changed
• <code>get_startstop_info</code>	changed
• <code>get_status</code>	changed
• <code>get_value</code>	changed
• <code>get_waveform</code>	changed
• <code>get_xy_pos</code>	not supported
• <code>get_xyz_pos</code>	not supported
• <code>goto_xy</code>	changed
• <code>goto_xyz</code>	changed
• <code>list_jump_cond</code>	changed
• <code>list_nop</code>	changed
• <code>load_cor</code>	not supported
• <code>load_correction_file</code>	changed
• <code>load_pro</code>	not supported
• <code>load_program_file</code>	changed
• <code>read_pixel_ad</code>	not supported
• <code>read_status</code>	changed
• <code>rtc3_count_cards</code>	not supported
• <code>rtc4_count_cards</code>	not supported
• <code>select_cor_table</code>	changed
• <code>select_list</code>	not supported
• <code>select_rt5</code>	changed
• <code>set_control_mode</code>	enhanced
• <code>set_control_mode_list</code>	enhanced
• <code>set_laser_mode</code>	enhanced
• <code>set_laser_timing</code>	changed
• <code>set_list_mode</code>	not supported
• <code>set_matrix</code>	changed
• <code>set_matrix_list</code>	changed
• <code>set_offset</code>	changed
• <code>set_offset_list</code>	changed
• <code>set_piso_control</code>	not supported
• <code>set_pixel</code>	changed
• <code>set_pixel_line</code>	changed
• <code>set_softstart_mode</code>	changed
• <code>set_trigger</code>	enhanced
• <code>set_wobbel</code>	changed
• <code>set_wobbel_xy</code>	not supported



## Increased Parameter Resolution

When switching from the RTC4 to the RTC5 – 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 coordinate values and y coordinate values are specified with 20-bit resolution for the RTC5 (whereas with 16-bit resolution for the RTC4), see also [Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156](#).
- An extended, virtual 24-bit value range is available with the RTC5 for Processing-on-the-fly applications, 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 RTC5 (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 RTC5 (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  $0.5 \mu\text{s}$  resolution for the RTC5 (whereas with  $1 \mu\text{s}$  resolution for the RTC4).

Here, though, it generally suffices to set the RTC5 DLL to [RTC4 Compatibility Mode](#) by `set_rtc4_mode`. Then the RTC5 DLL automatically converts parameter values so that many RTC4 command sequences can run unchanged on the RTC5.

**RTC4 Compatibility Mode** affects the following RTC4 commands (descriptions of the respective commands include relevant information under the heading "RTC4→RTC5"):

- `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 execute in **RTC5 Standard Mode** or **RTC4 Compatibility Mode**.

### Changed Timing Behavior

The following RTC4 list commands are processed by the RTC5 as “short list commands”. This can result in a changed timing behavior during execution (see **Section “Normal, Short, Variable and Multiple List Commands”, page 278**).

- `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` (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 143**).

## 2.10.3 New and Changed Functionality

### Interface to the PC

- The RTC5 board driver supports simultaneous control of any number of RTC5 PCI Boards in a single PC, see [Chapter 6.6 "Using Several RTC5 PCI Boards in One PC"](#), page 112.
- Connectors and commands for master/slave synchronization of several RTC5 PCI Boards, see [Chapter 4.4 "Master Socket Connector, Slave Socket Connector"](#), page 53.

### Controlling Scan Systems

- New interface to the scan system, see [Chapter 4.5.1 "Scan Head Connectors and Transfer Protocol"](#), page 54:
  - 9-pin female D-SUB connector at the RTC5 PCI 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 *iDRI*VE scan systems<sup>(1)</sup>
- An [XY2-100 Converter \(Accessory\)](#) is available for data transfer according to the XY2-100 protocol.
  - 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 \(Accessory\)](#)

The RTC5 PCI Board provides power for the [XY2-100 Converter \(Accessory\)](#).

- For optical data transfer between the RTC5 PCI Board and scan systems, *no* special variant of the RTC5 PCI 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 58.
- 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 2 RTC5 PCI 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
    - Queryable information in the correction file header
    - For the RTC5 PCI Board, RTC4 correction files (.ctb) need to be newly calculated or converted by [CorrectionFileConverter.exe](#) which is part of the RTC5 software package.
  - Up to 4 correction files can be loaded to the RTC5 PCI 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 209):
  - 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 213

(1) See Glossary entry on [page 23](#).

- Coordinate transformations in the virtual **Image field** (incl. "Global Online Positioning"), see **Chapter 7.3.3 "Virtual Image Field"**, page 158
  - 24-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 200
- Automatic self-calibration, see **Chapter 8.10 "Automatic Self-Calibration"**, page 251:
  - Optimization of previous functions
  - ASC hardware check
- **Jump Mode**, see **Chapter 8.1.5 "Jump Mode"**, page 202
- Output synchronization, see **Chapter 7.4.10 "Output Synchronization"**, page 195

### Controlling the Laser

- The signal levels of the laser control signals are no longer determined by a jumper configuration. Instead, they are software-configured, see **set\_laser\_control**
- 15-pin female D-SUB LASER connector with all laser signals at the RTC5 PCI Board slot cover, see **Chapter 4.6.1 "LASER Connector"**, page 60, 9-pin female D-SUB connector only by the SCANLAB laser adapter, see **Section "Laser Adapter (Accessory)"**, page 63
- 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 Mode 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 Mode 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

- **Pulse Picking Laser Mode**, see **Chapter 7.4.8 "Pulse Picking Laser Mode"**, page 186
- Laser pulse period, pulse length or analog output are also programmable within a **Polyline** between two vectors – where the laser remains on<sup>(2)</sup>, see "short list commands" in **Section "Normal, Short, Variable and Multiple List Commands"**, page 278
- Commands for position-dependent, speed-dependent, vector-defined and encoder-speed-dependent laser control, see **Chapter 7.4.9 ""Automatic Laser Control""**, page 187

### Interfaces for Peripheral Equipment

- 16-bit digital output, see **Section "16-Bit Digital Input Port and 16-Bit Digital Output Port"**, page 65, and **Chapter 9.1.1 "16-Bit Digital Output Port"**, page 258:
  - 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 port, see **Section "8-Bit Digital Output Port"**, page 68 and **Chapter 9.1.2 "8-Bit Digital Output Port"**, page 259:
  - 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 output ports, see **Section "12-Bit Analog Output Port 1 and 2"**, page 61, and **Chapter 9.1.4 "12-Bit Analog Output Port 1 and 2"**, page 259:
  - 12 bit resolution
  - 0...10 V (0 V...2.50 V no longer available)
  - Adjustable "stop output value"
- 16-bit digital input port, see **Section "16-Bit Digital Input Port and 16-Bit Digital Output Port"**, page 65, and **Chapter 9.2.1 "16-Bit Digital Input Port"**, page 264:
  - SYNC signal for synchronization of data transmission

(1) See Glossary entry on [page 23](#).

(2) Is switched off with the RTC4.

- Programmable debouncing of external start signals, see [bounce\\_supp](#) and [Section "External Start", page 266](#)
- Regular (periodic) [External Starts](#), see [Section "Regular \(Periodic\) External Starts", page 269](#)
- 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 61](#) and [Section "2-Bit Digital Output Port", page 61](#)
  - RS-232 interface by an 10-pin socket connector, see [Chapter 4.6.5 "RS232 Socket Connector", page 70](#)
  - Stepper motor signals for 2 motors by an on-board 10-pin socket connector, see [Chapter 4.6.7 "STEPPER MOTOR Socket Connector", page 75](#)
  - McBSP, I<sup>2</sup>C and SPI by a 10-pin socket connector, see [Chapter 4.6.6 "SPI / I2C Socket Connector", page 71](#)
- For the RTC5 PCI Board there is no 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 RTC5 EXTENSION 2 socket connector corresponds to the RTC4 "LASER EXTENSION" socket connector).

## General Programming

- Utility files for C, C++, C# and Delphi, see [Chapter 6.2.2 "Importing Commands", page 84](#), but no longer utility files for Basic
- Commands for changing access rights to RTC5 PCI Boards, see [Chapter 6.7 "Usage of RTC5 PCI Boards by Several User Programs", page 117](#)
- Improved and extended list handling, see [Chapter 6.4 "List Handling", page 94](#)
  - List memory with 1,048,576 storage positions
  - List memory free configurable (in 2 unprotected list memory areas and 1 protected list memory area)
  - Defining protected subroutines
  - Loading lists with protection
  - Loops in lists and subroutines
  - RTC4-Circular queue mode is not available (but can be coded alternatively)
  - List Status
  - 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 cycle, see [Section "Normal, Short, Variable and Multiple List Commands", page 278](#))
- Functions for error handling and download verification, see [Chapter 6.8 "Error Handling", page 119](#)



## Laser Marking

- Vectors and arcs
  - Commands for marking ellipses, see [Section "Ellipse Commands", page 126](#)
  - Commands for marking helices, see [Chapter "3D Commands", page 223](#)
  - Timed arc commands, timed 3D vector commands, see [Chapter 8.9 "Timed Commands", page 250](#)
  - Para-Mark commands and Para-Jump commands for "vector-controlled laser control", see [Section "Vector-Defined Laser Control", page 194](#)
- Characters and texts
  - Defining character sets and text strings, see [Section "Defining Indexed Character Sets", page 107](#)
  - Commands for marking individual characters and for marking texts (with selectable character set), see [Section "Calling Indexed Characters", page 108](#)
  - 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 196](#)
- `micro_vector[*]` commands (direct position output without Microstepping), see [Chapter 8.8 "micro\\_vector\[\\*\] Commands", page 249](#).

## Special Functions

- Synchronization of scan system and laser control
  - Sky Writing, see [Chapter 7.2.4 "Sky Writing", page 149](#)
- Pixel Output Mode (marking of bitmaps), see [Chapter 8.7 "Pixel Output Mode", page 244](#):
  - Pixel output frequencies up to 300 kHz, irrespective of the 10  $\mu$ s clock cycle
  - 15 ns resolution
  - 0...100% laser pulse length
  - Pixel-Mode 0 not supported
  - Reading of analog voltages is not supported
  - Pixel marking on sloped surfaces
  - 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 271](#)
- 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 "Wobble Mode", page 217](#)
- Camming functionality, see [Chapter 8.11 "Camming", page 255](#)
- Enhanced signal recording, see `set_trigger` and `set_trigger4`

- Processing-on-the-fly, see [Chapter 8.6 "Processing-on-the-fly", page 227](#)
  - 2 encoder input ports (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 274](#); alternatively: Processing-on-the-fly correction with McBSP signals, see [Chapter 9.3.4 "Synchronization and Online Positioning by McBSP/SPI Signals", page 276](#)
  - 24-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 266](#)
  - 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 table)
  - 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 242](#)

## Notice!

- The encoder counter direction of RTC4 boards (includes RTC4 Ethernet Board and RTC SCANalone Board) is opposite to that of RTC5 boards and RTC6 boards. Therefore, when changing RTC-control board you need to either adapt the cabling or your user program accordingly.

### 3 Safety During Installation and Operation

Read these operating instructions completely before you proceed with installing and operating the RTC5 PCI Board.

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

The following conventions apply to safety notices 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 notices 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 RTC5 PCI 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 RTC5 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.



#### 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.

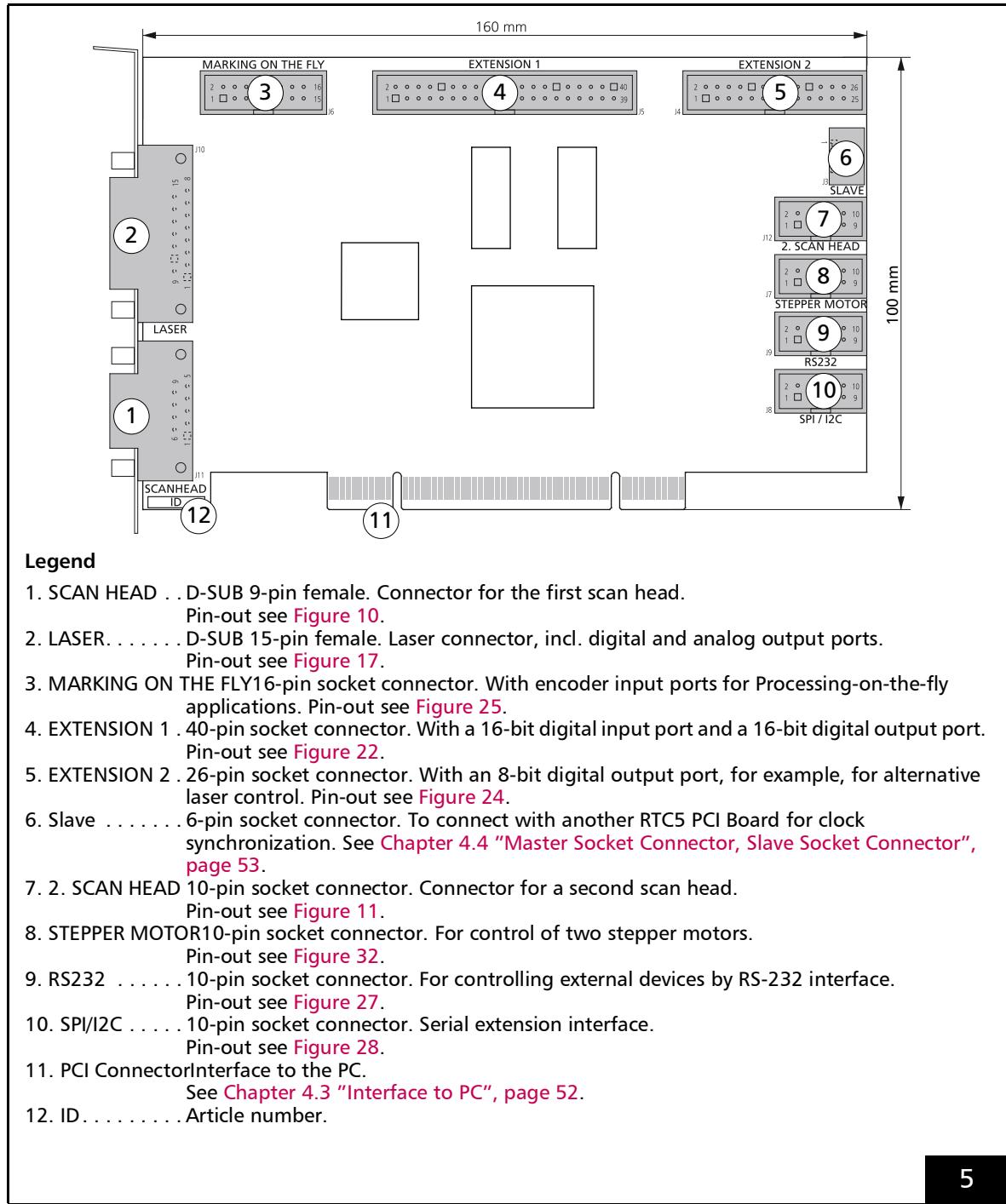
#### 3.1 Steps for Safe Operation

##### 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.

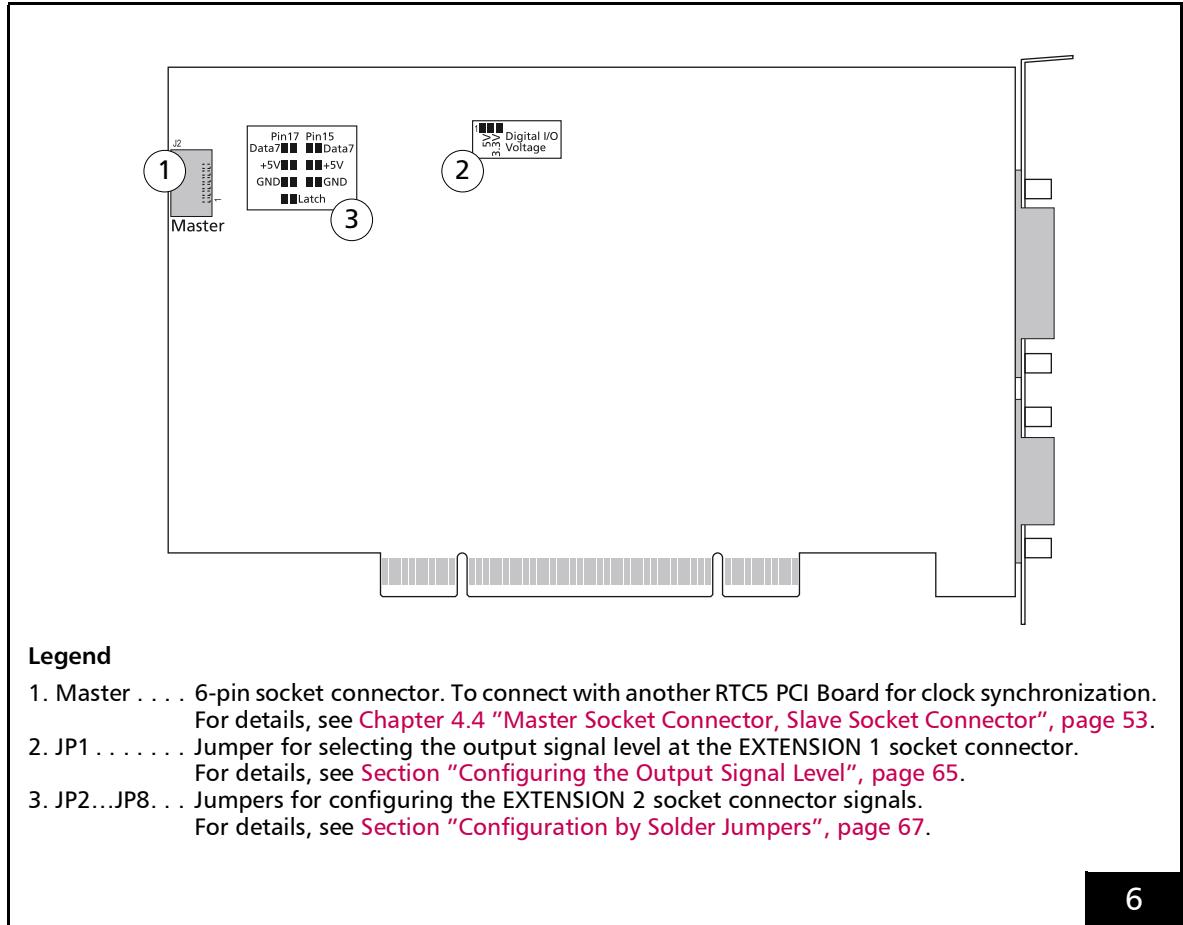
## 4 RTC5 PCI Board – Layout and Interfaces

### 4.1 Layout – Upper Side



RTC5 PCI Board: upper side.

## 4.2 Layout – Lower Side



6

RTC5 PCI Board: lower side.

## 4.3 Interface to PC

The RTC5 PCI Board PCI connector is the interface to the PC.

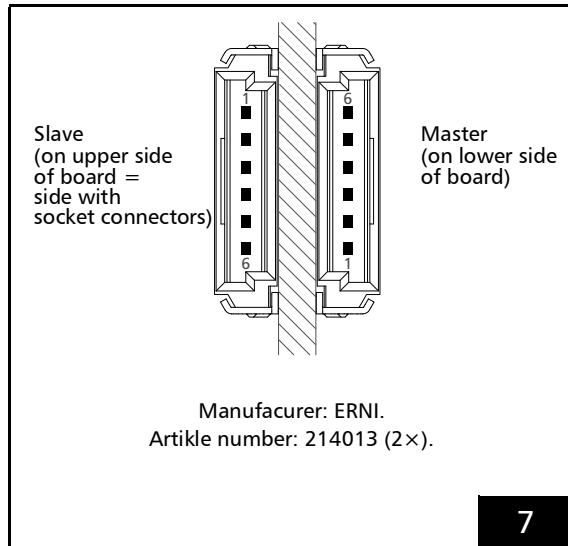
The RTC5 PCI Board can be installed into any Windows PC with a PCI bus interface and at least one free PCI slot.

### Notice!

- The RTC5 PCI Board does not support power-saving modes that switch off power to the PCI bus. Accordingly, you must disable standby or sleep modes of the operating system. See also [Section "Notes", page 33](#).

#### 4.4 Master Socket Connector, Slave Socket Connector

The Slave socket connector as well as the Master socket connector have 6 pins, see [Figure 7](#). The Slave socket connector is located on the upper side of the RTC5 PCI Board, see [Figure 5](#). The Master socket connector is located on the lower side, see [Figure 6](#).



7

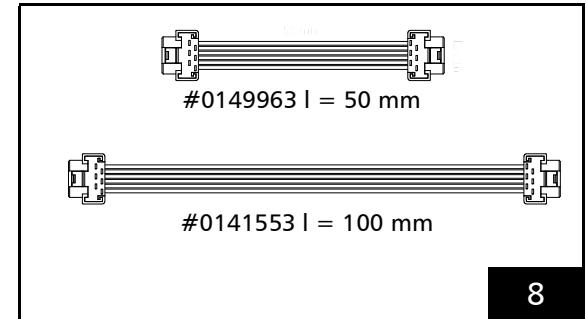
Slave socket connector and Master socket connector.  
The pitch of the pins is 1.27 mm.

Purpose of the both socket connectors is to make a clock cycle synchronization of several RTC5 PCI 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 8](#)). The necessary information for assembling your own cables is shown in [Figure 9](#).

Interconnected RTC5 PCI Boards should (recommended) be plugged into adjacent PCI slots.

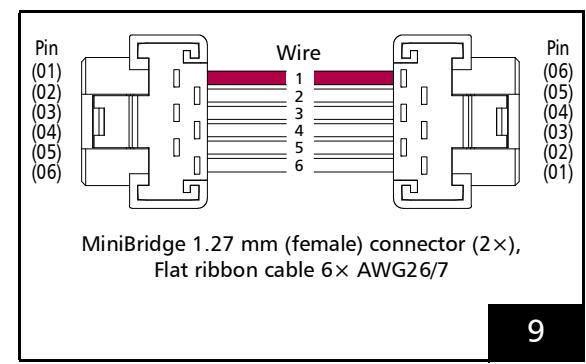
Important: In order to use the master/slave functionality, see prerequisites in [Chapter 6.6.3 "Master/Slave Operation", page 113](#).

See also [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 265](#).



8

SCANLAB Master/Slave connecting cable.



9

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

## 4.5 Interfaces to Scan System

### 4.5.1 Scan Head Connectors and Transfer Protocol

The first scan head connector SCAN HEAD and the optionally activated second scan head connector "2. SCAN HEAD" are available for digitally controlling scan systems (see [Figure 5](#)). 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 RTC5 and the scan system is in accordance with the SL2-100 protocol. The [XY2-100 Converter \(Accessory\)](#) is available for converting the signals (for data transmission according to the XY2-100 protocol).

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 RTC5 PCI 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 ([Option "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 Correction Files and 3D Correction Files", page 163](#).

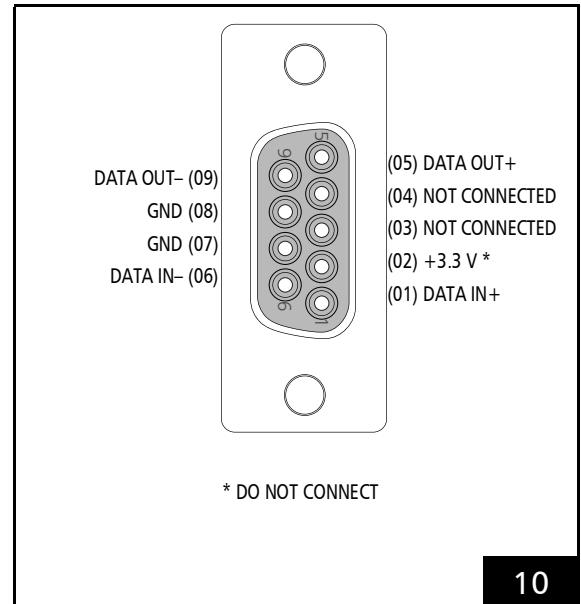
If several RTC5 PCI Boards with enabled [Option "3D"](#) are installed in a PC, then that many 3-axis systems can be simultaneously controlled.

For master/slave functionality, multiple RTC5 PCI Boards can be run – synchronously clocked – in one PC if [load\\_program\\_file](#) has been executed on all boards (see [Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 53](#) and [Section "Initializing the Board", page 87](#)).

#### SCANHEAD Connector

(connector for the first scan head)

The pin-out of the first scan head connector SCANHEAD (D-SUB 9-pin female) is shown in [Figure 10](#).



10

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

The differential DATA OUT output port transmits control values to the scan system.

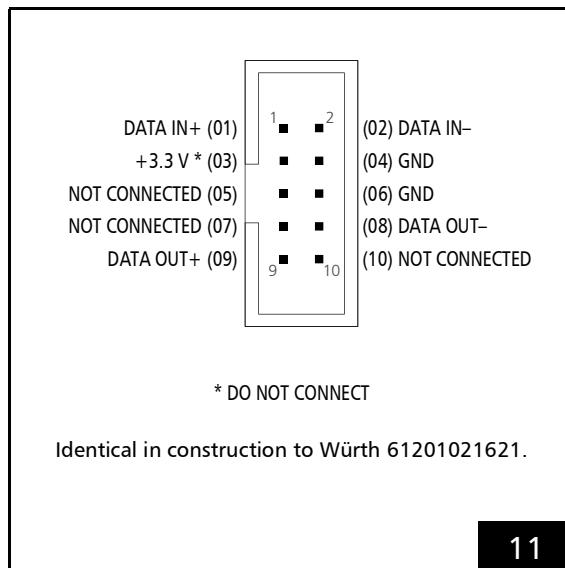
The differential DATA IN input port receives the status signals returned by the scan system.

Pin 2 supplies 3.3 V power for the [XY2-100 Converter \(Accessory\)](#) (or a Polymer Optical Fiber converter for optical data transmission). This voltage should not be used for other purposes.

## 2. SCANHEAD Socket Connector

(connector for the second scan head)

The pin-out of the second scan head connector  
 2. SCANHEAD (10-pin socket connector) is shown in  
**Figure 11.**



11

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"](#).

### Second Scan Head Slot Cover (Accessory)

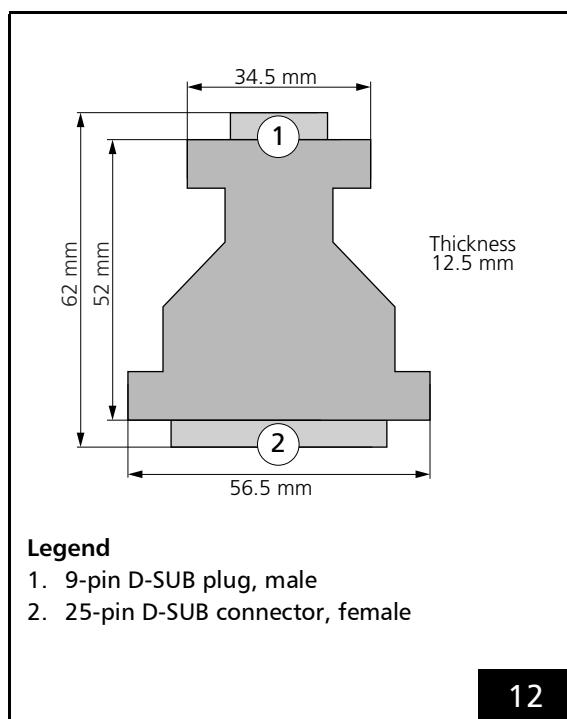
SCANLAB recommends using an additional slot cover for connecting a second scan head or a z axis to the 2. SCANHEAD Socket Connector, see [Chapter 2.8.4 "Slot Cover with 9-pin D-SUB Connector for 2. SCANHEAD Socket Connector", page 38](#).

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

The SCANLAB **XY2-100 Converter (Accessory)** (#0125377) converts the RTC5's SL2-100 control signals (20 bit) into XY2-100-compliant signals (16 bit), and converts scan system XY2-100 status signals into SL2-100-compliant signals (see page 172).

The **XY2-100 Converter (Accessory)** introduces a  $10\ \mu\text{s}$  runtime latency to scan-system control. This runtime latency can be compensated by increasing the **LaserOn Delay** and **LaserOff Delay** by  $10\ \mu\text{s}$  each (see [set\\_laser\\_delays](#)).

The dimensions are shown in [Figure 12](#).

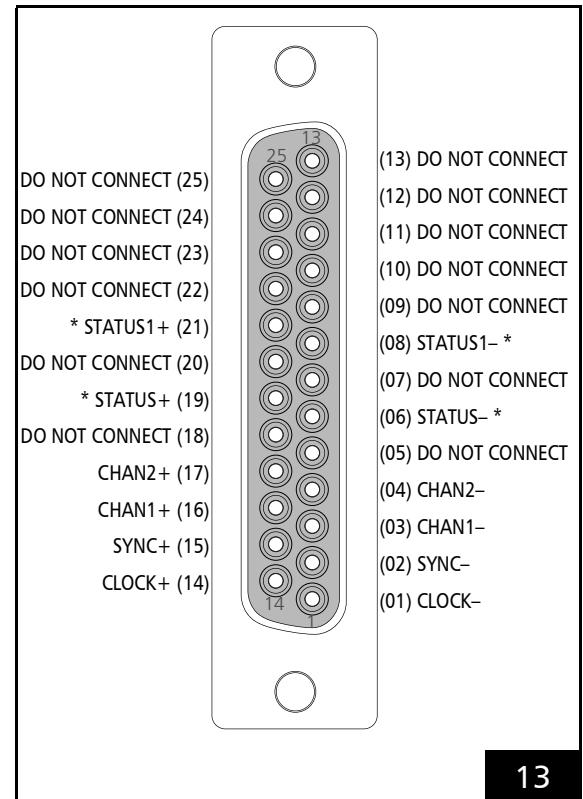


12

[XY2-100 Converter \(Accessory\)](#): dimensions.

The **XY2-100 Converter (Accessory)**'s 9-pin D-SUB connector should be directly plugged into the RTC5's first scan head connector or (by a short-as-possible 1:1 cable) connected to the corresponding pins of the RTC5's second scan head connector. For the pin-out, see [Figure 10](#) and [Figure 11](#).

The 25-pin D-SUB connector of the **XY2-100 Converter (Accessory)** is compatible with scan heads that provide an XY2-100 standard interface. The pin-out is shown in [Figure 13](#).



13

[XY2-100 Converter \(Accessory\)](#): pin-out of the 25-pin D-SUB connector (female).

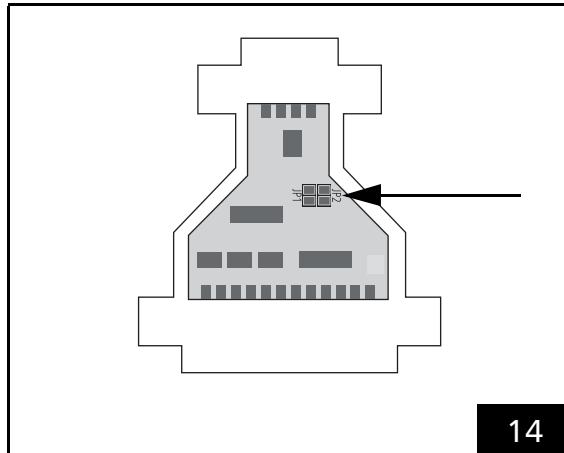
\* For **iDRIVE** scan systems (intelliSCAN, intelliSCAN<sub>de</sub>, intelliDRILL, intellicube, intelliWELD, varioSCAN<sub>de</sub>), the STATUS $\pm$  channel is the status channel of axis 2 (x axis; this channel is then also called STATUS2 $\pm$ ) and the STATUS1 $\pm$  channel is the status channel 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 (Accessory)** and the scan system, then the synchronization of bidirectional communication between the scan system and the RTC5 should be configured.

This can be accomplished by two solder jumpers in the **XY2-100 Converter (Accessory)**.

To do so, carefully open the housing of the **XY2-100 Converter (Accessory)** by its 4 clip latches<sup>(1)</sup>. The solder jumpers **JP1** and **JP2** are on the PCB, see arrow in Figure 14.



**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).

#### Notes

- Observe the note on [get\\_head\\_status](#), page 199.

(1) For example, using a slotted screwdriver.

### 4.5.3 Data Cables (Accessories)

For transmission of the data signals between the RTC5 (or the **XY2-100 Converter (Accessory)**) 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 also optical fiber cables<sup>(1)</sup> (only upon request).

For SCANLAB scan systems equipped with a conventional 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 RTC5.

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

- For SL2-100-compliant data transmission, see **Figure 15**, 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 RTC5's DATA OUT signal flows to the scan system's DATA IN input). 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 **Figure 16**, 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 an **XY2-100 Converter (Accessory)** 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 (Accessory)** need to be reconfigured.
- For XY2-100-compliant data transmission, the controller end of the data cable must be fitted with a ferrite ring (for example, Würth WE 74271132).

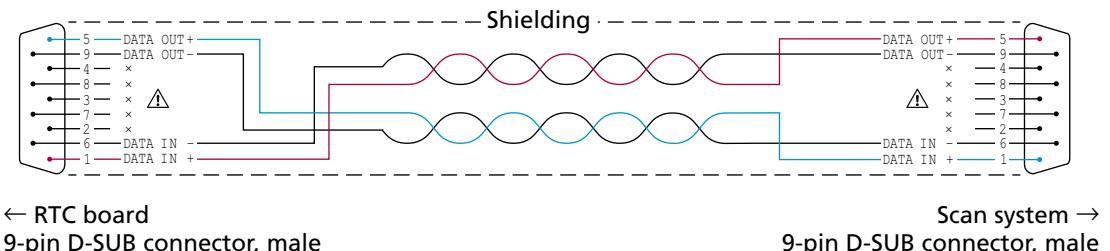
Independently from the data transmission protocol, the following applies in addition:

- 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 should *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 a single connector which provides both the power supply and the data signals. For these scan heads, SCANLAB recommends using a Y cable (voltage and data are to be transferred in separate cables; on the condition see above).

(1) The optical fiber cables, too, are attached by 9-pin D-SUB connectors. Optical conversion (Polymer Optical Fiber conversion) for optical data transmission takes place inside the D-SUB connectors. The operating voltage for Polymer Optical Fiber conversion is supplied at the RTC5 scan head connectors and the digital interface of the scan system.

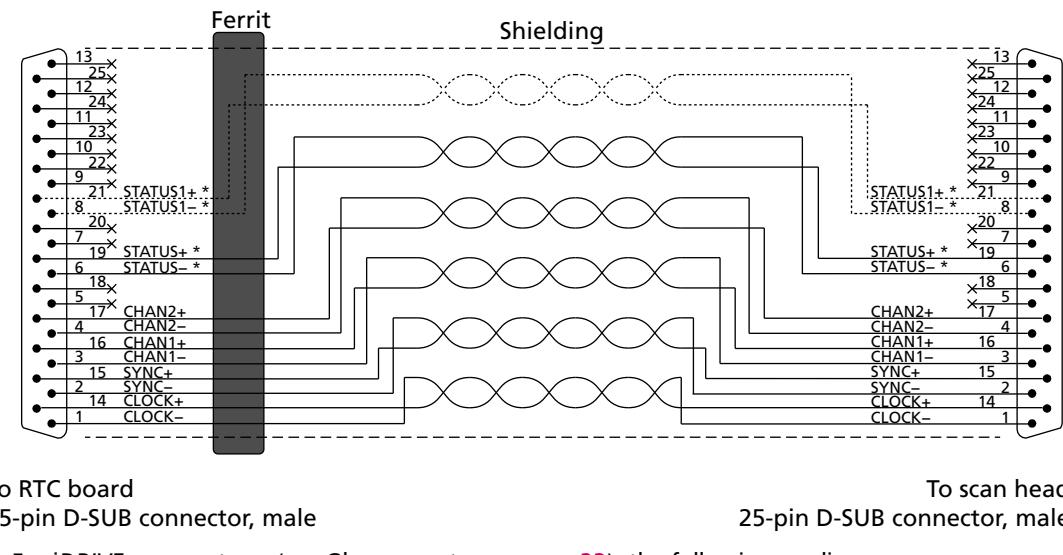
Cable for SL2-100-compliant data transmission



15

Cable for data transmission compliant to SL2-100 protocol: requirements and pin assignments:  
requirements and pin assignments.

Cable for XY2-100-compliant data transmission



- \* For iDRIVE scan systems (see Glossary entry on [page 23](#)), 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.

16

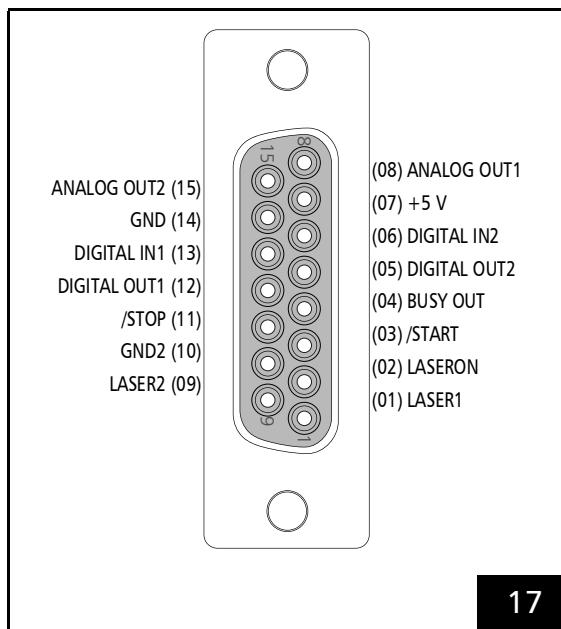
Cable for data transmission compliant to XY2-100 protocol:  
requirements and pin assignments.

## 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 RTC5 PCI Board, see [Figure 5](#).

The pin-out is shown in [Figure 17](#).



17

LASER connector (15-pin D-SUB connector, female): pin-out. On RTC5 PCI Boards without [Option "DC/DC Converter"](#), GND2 are GND identical.

#### Notice!

- If you want to use the RTC5 PCI Board in conjunction with laserDESK, observe the following:
  - laserDESK uses additional pins on the [EXTENSION 1 Socket Connector](#), for example, to control the laser.
  - Refer to the extended documentation for the LASER connector which can be found in the laserDESK online help (alternatively available from SCANLAB or in laserDESK-zip, which can be downloaded from the SCANLAB website).

#### Laser Control Signals

The laser control signals LASER1 and LASER2 at pin (1) and pin (9) depend on the selected laser control mode, see the corresponding 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, 2, 3, 5	Q-Switch signal	FirstPulseKiller signal
Laser Mode 4	Standby signal	FirstPulseKiller signal
Laser Mode 6	Standby signal	–

All laser control signals (LASERON, LASER1 and LASER2) are permanently generated by the RTC5 Board. They are digital TTL level signals and are referenced to GND2.

On RTC5 boards with the [Option "DC/DC Converter"](#) the GND2 and the laser output signal are galvanically decoupled from GND<sup>(1)</sup>. On RTC5 boards without [Option "DC/DC Converter"](#), GND2 are GND identical, see [Chapter 2.6 "Options", page 34](#).

For the maximum current load see [Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734](#).

All laser signals can be set to either active-LOW or active-HIGH logic by [set\\_laser\\_control](#). 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.

[config\\_laser\\_signals](#) and [config\\_laser\\_signals\\_list](#) can be used to configure pins (1), (2) and (9) of the laser connector, see [Chapter 7.4.2 "Configuring the LASER Connector", page 176](#).

(1) With RTC5 boards, GND is the PC ground.



## 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 18](#). 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 265](#).

## BUSY List Execution Status

The **BUSY list execution status** is available as the BUSY OUT signal at pin (04). When the **BUSY list execution status** is set, the BUSY OUT signal is HIGH. See also [Chapter 6.4.3 "List Execution Status", page 97](#). The signal is referenced to GND<sup>(1)</sup>.

## 2-Bit Digital Input Port

The RTC5 PCI Board provides a 2-bit digital input port (DIGITAL IN1 and DIGITAL IN2).

For technical data see [Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734](#).

For programming the input port see [Chapter 9.2.2 "2-Bit Digital Input Port", page 264](#).

## 2-Bit Digital Output Port

The RTC5 PCI Board provides a buffered 2-bit digital output port (DIGITAL OUT1 and DIGITAL OUT2).

For technical data see [Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734](#).

For programming the output port see [Chapter 9.1.3 "2 Bit Digital Output Port", page 259](#).

## 12-Bit Analog Output Port 1 and 2

The RTC5 PCI Board provides two 12-bit analog output ports:

- ANALOG OUT1
- ANALOG OUT2<sup>(2)</sup>

For technical data see [Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734](#).

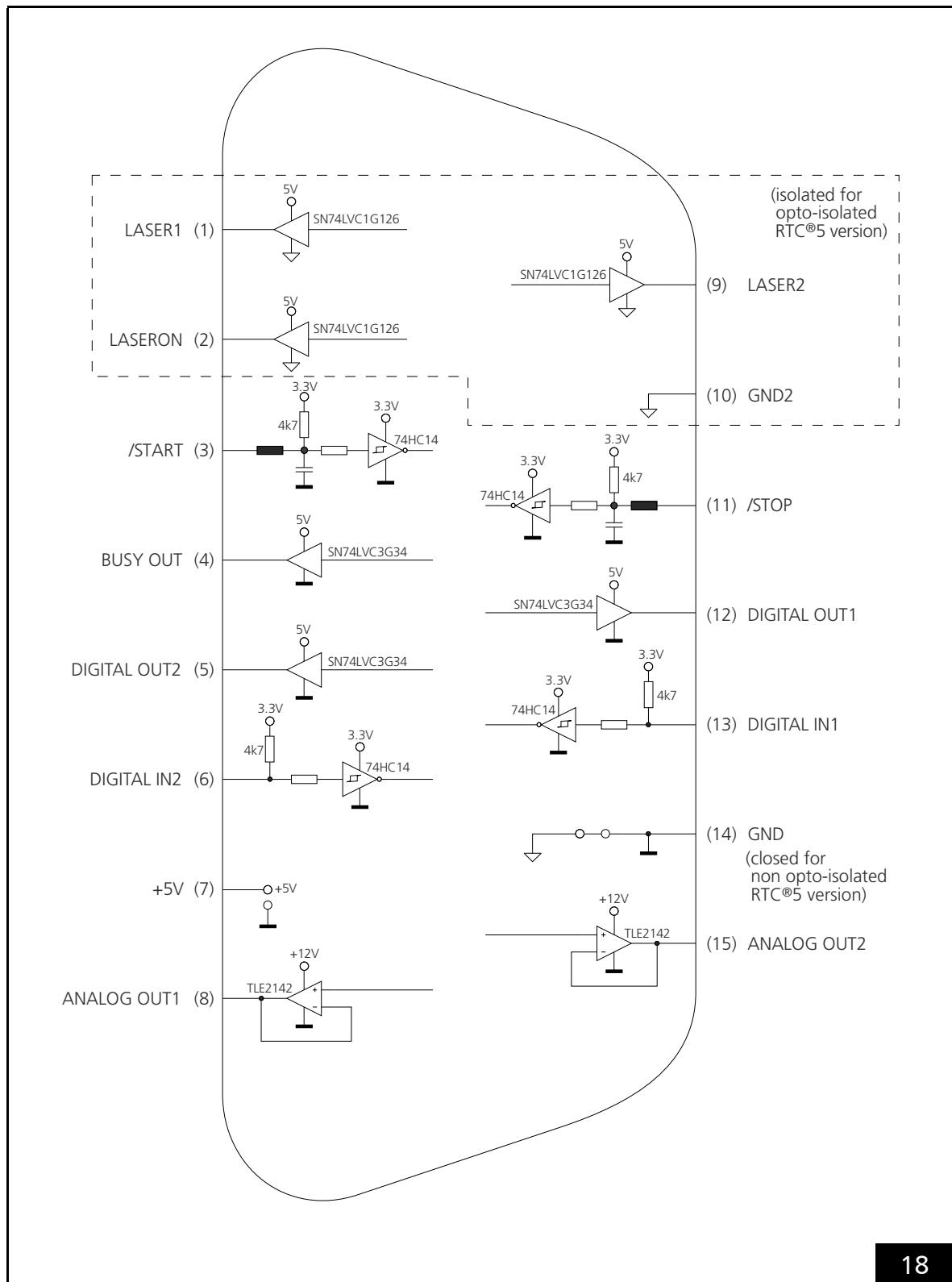
For programming the outputs see [Chapter 9.1.4 "12-Bit Analog Output Port 1 and 2", page 259](#).

## Input/Output Wiring

The input/output wiring of the LASER connector is shown in [Figure 18](#).

(1) See footnote on [page 60](#).

(2) The signal of ANALOG OUT2 is also available by the MARKING ON THE FLY socket connector, see [Section "Analog Output Port", page 69](#).



LASER connector, see also [Figure 17: input/output wiring](#).  
 See also [Section "Option "DC/DC Converter""](#), page 34.

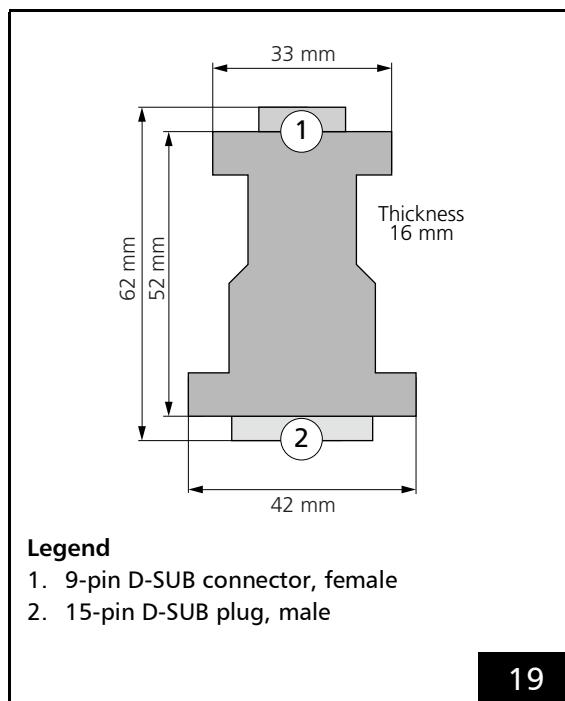
### Laser Adapter (Accessory)

The SCANLAB laser adapter (accessory; #0114774) is plugged into the 15-pin LASER connector of the RTC5 PCI Board. Then its 9-pin D-SUB connector (female) provides the same signals and pin-out as the RTC4 9-pin LASER connector.

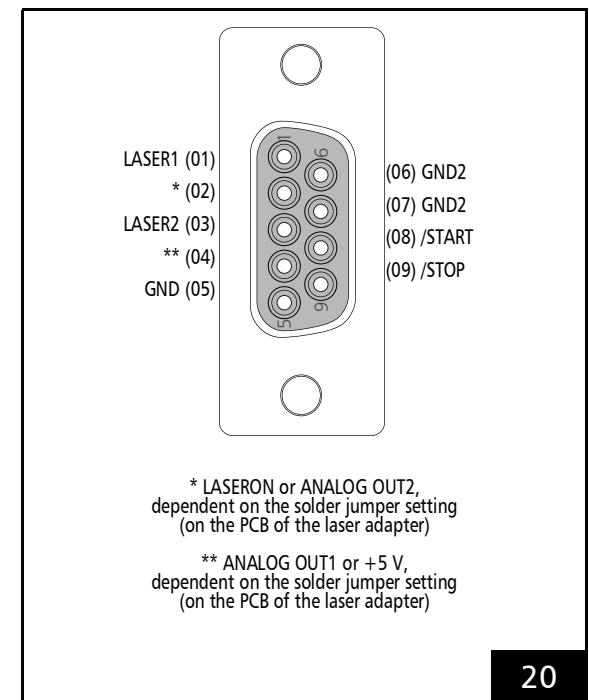
#### Notes

- The laser adapter can *not* be directly plugged into the RTC5 PCI Board, if an **XY2-100 Converter (Accessory)** 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 19**.



The pin-out of the 9-pin D-SUB connector is shown in **Figure 20**.



**20**

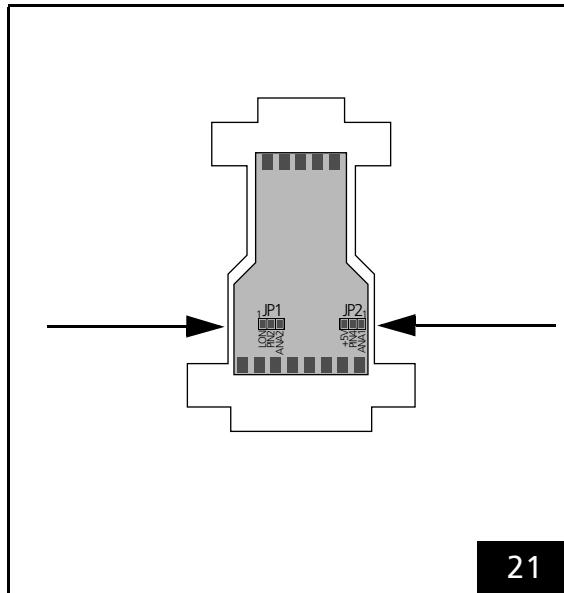
Laser adapter (accessory): pin-out of the 9-pin D-SUB connector, female.

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's housing by its 4 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.

**19**

Laser adapter (accessory): dimensions.

The positions of the solder jumpers on the PCB is shown in [Figure 21](#).



21

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

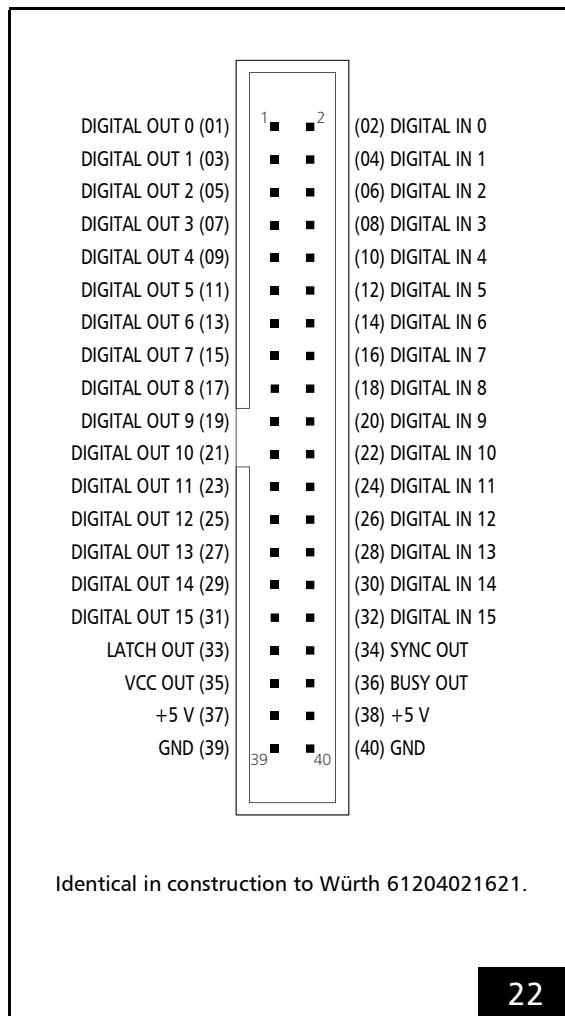
The following table shows the possible jumper settings. Other jumper settings are not allowed.

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

#### 4.6.2 EXTENSION 1 Socket Connector

The EXTENSION 1 socket connector has 40 pins. It is located on the upper side of the RTC5 PCI Board, see [Figure 5](#).

The pin-out is shown in [Figure 22](#).



EXTENSION 1 socket connector: pin-out. The pitch of the pins is 2.54 mm.

#### Configuring the Output Signal Level

By Jumper **JP1** on the lower side of the RTC5 PCI Board, see [Figure 6](#), the level of all output signals at the EXTENSION 1 socket connector (DIGITAL OUT0-15, LATCH\_OUT, SYNC\_OUT, BUSY\_OUT, VCC\_OUT) can be configured for 5 V or 3.3 V, see [Chapter 2.7.1 "Jumper JP1 – Configuring the Output Signal Level at the EXTENSION 1 Socket Connector"](#), page 36.

For user monitoring purposes, the selected signal level is also continuously outputted at pin (35): signal VCC\_OUT. VCC\_OUT is referenced to PC ground (GND). The maximum current load is 100 mA.

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

The 40-pin EXTENSION 1 socket connector provides a 16-bit digital TTL input and a buffered 16-bit digital TTL output, see [Figure 22](#). This requires the output signals level to be configured by the jumper setting, see [Section "Configuring the Output Signal Level"](#), page 65.

For programming see:

- [Chapter 9.1.1 "16-Bit Digital Output Port"](#), page 258
- [Chapter 9.2.1 "16-Bit Digital Input Port"](#), page 264
- [Chapter 9.3.2 "Conditional Command Execution"](#), page 271

The input and output signals are referenced to PC ground (GND). The maximum current load of the output signals is 8 mA.

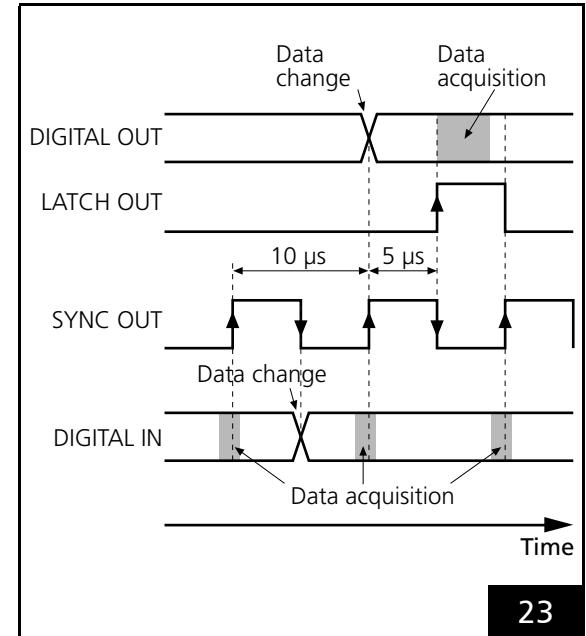
### Synchronization of Data Transmission

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

The LATCH signal (a 5  $\mu$ s 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 RTC5 automatically generates the LATCH signal when the output value at the 16-bit digital output port is outputted. The output value should be read-out with the rising edge of the LATCH signal, which is generated 5  $\mu$ s after the value has been outputted at the 16-bit digital output port (see Figure 23).

To synchronize data transmission at the 16-bit digital input port, pin (34) continuously provides a SYNC signal (a square wave with 5  $\mu$ s pulse length and 10  $\mu$ s period). Value changes at the 16-bit digital input port should be made with a falling edge of the SYNC signal. The RTC5 **DSP** always accepts the currently provided value with the rising edge of the SYNC signal (see Figure 23).

The synchronization signals are referenced to PC ground (GND). The maximum current load is 10 mA.



23

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

### BUSY List Execution Status

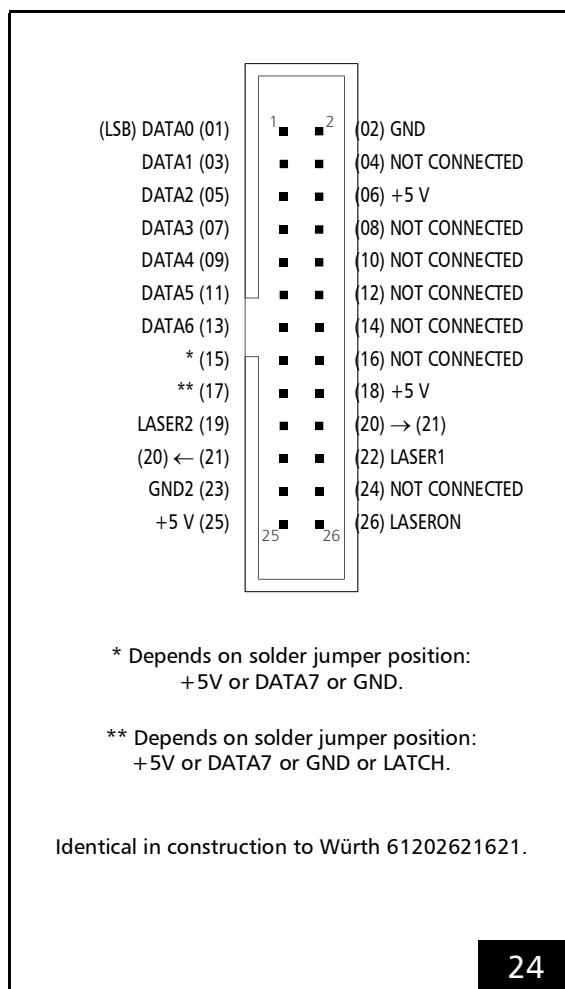
The BUSY OUT signal at pin (36) is identical to the BUSY OUT signal at the LASER connector (see Section "BUSY List Execution Status", page 61). The signal is referenced to GND.

### 4.6.3 EXTENSION 2 Socket Connector

The EXTENSION 2 socket connector<sup>(1)</sup> has 26 pins. It is located on the upper side of the RTC5 PCI Board, see [Figure 5](#).

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

The pin-out is shown in Figure 24.



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.

## Configuration by Solder Jumpers

Pins (15) and (17) of the EXTENSION 2 socket connector have to be configured by the jumpers JP2...JP8 on the lower side of the RTC5 (see [Figure 6](#)). The most significant bit (DATA7) of the 8-bit output value can be assigned to pin (15) or to pin (17). Alternatively, each of the two pins can be set permanently to +5 V (HIGH) or to GND (LOW level). Alternatively, pin (17) can be configured for the LATCH signal (see below) by closing the correspondingly labeled jumper (see [Chapter 2.7.2 "Jumper JP2...JP8 – Configuring Pin15 and Pin17 at the EXTENSION 2 Socket Connector", page 36](#)).

## Notes

- If the DATA7 bit is assigned to pin (15), the full 8-bit output value is available at the output port (odd numbered pins (1) to (15) of the EXTENSION 2 socket connector).
  - Setting pin (15) permanently to HIGH results in an offset of 128 for the output value and restricts the output value range to (128...255).
  - Setting pin (15) to LOW restricts the output value range to 0...127.
  - The DATA7 bit can be used for other purposes by assigning it to pin (17).



## Laser Control Signals

Like the laser signals of the LASER connector, the output signals LASER1 and LASER2 depend on the selected laser control mode (see [Section "Laser Control Signals", page 68](#)) and are referenced to GND2. On RTC5 Boards with the [Option "DC/DC Converter"](#) the GND2 and the laser output signal are galvanically decoupled from the PC ground (GND). Otherwise, GND2 are GND identical (see [Chapter 2.6 "Options", page 34](#)).

## 8-Bit Digital Output Port

The buffered 8-bit digital output port (TTL level) is intended for YAG lasers with a digital lamp current control. However, 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 259](#).

The most significant bit ([MSB](#)) (DATA7) of the output value can be used for other purposes. To do this, the [MSB](#) can be assigned to an extra pin on the EXTENSION 2 socket connector (see above).

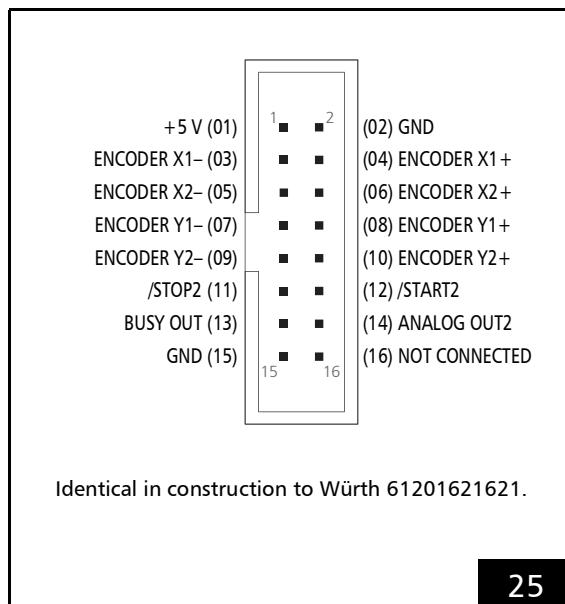
By an appropriate jumper setting (see above), Pin (17) can output a LATCH signal. The RTC5 automatically generates the LATCH signal (a 5  $\mu$ s pulse, active-HIGH) when the value at the 8-bit digital output port has been outputted. If several bits are simultaneously transferred as a data word (that is, if the bits are not transferred independently from each other) by the 8-bit digital output port, then the LATCH signal should be used for synchronization of data transmission (for example, as a trigger signal for the laser to assume a different lamp current). 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  $\mu$ s after the value at the 8-bit digital output port has been outputted (see also [Figure 23](#)). The LATCH signal is referenced to PC ground (GND). The maximum current load is 10 mA.

#### 4.6.4 MARKING ON THE FLY Socket Connector

The MARKING ON THE FLY socket connector provides, among other things, encoder input ports for incorporating encoder signals.

Together with the optional Processing-on-the-fly functionality, this enables laser material processing of moving workpieces (for example, on a moving conveyor belt or rotating disk, see [Chapter 8.6 "Processing-on-the-fly", page 227](#)).

The pin-out is shown in [Figure 25](#).



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

#### Encoder Input Ports

The RTC5 PCI Board provides 2 encoder input ports:

- ENCODER X
- ENCODER Y

ENCODER X and ENCODER Y are each designed for a pair of standardized differential input signals (RS-422; HIGH level  $\geq 2.0$  V, LOW level  $\leq 0.8$  V). See also [Section "Input Ports for External Encoder Signals", page 275](#).

#### External Control Signals

The external control signals /START2 and /STOP2 (TTL active-LOW) are referenced to PC ground (GND). 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 265](#).

#### 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 61](#).

The signal is referenced to GND.

#### BUSY list execution status Status

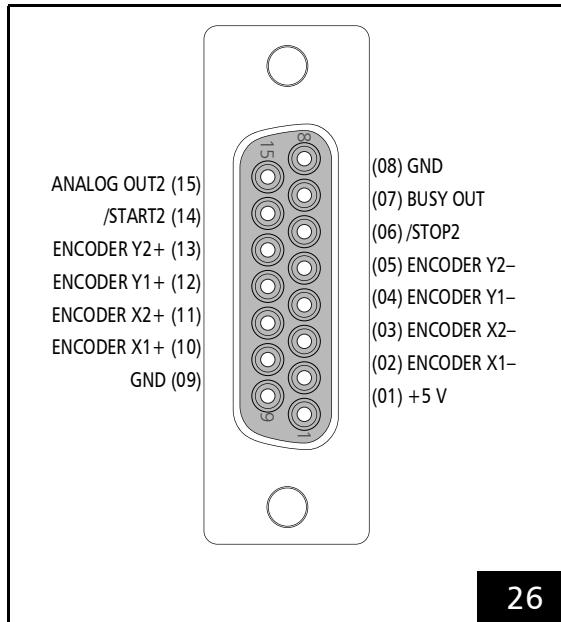
The BUSY OUT signal at pin (13) is identical to the BUSY OUT signal at the LASER connector, see [Section "BUSY List Execution Status", page 61](#).

The signal is referenced to GND.

### MARKING ON THE FLY Slot Cover (Accessory)

SCANLAB recommends using an additional slot cover for using the inputs and signals of the **MARKING ON THE FLY Socket Connector**, see **Chapter 2.8.5 "Slot Cover with 15-pin D-SUB Connector for MARKING ON THE FLY Socket Connector", page 38.**

The pin-out is shown in **Figure 26.**



26

**MARKING ON THE FLY Slot Cover (Accessory), 0109272:** pin-out of the 15-pin female D-SUB connector.

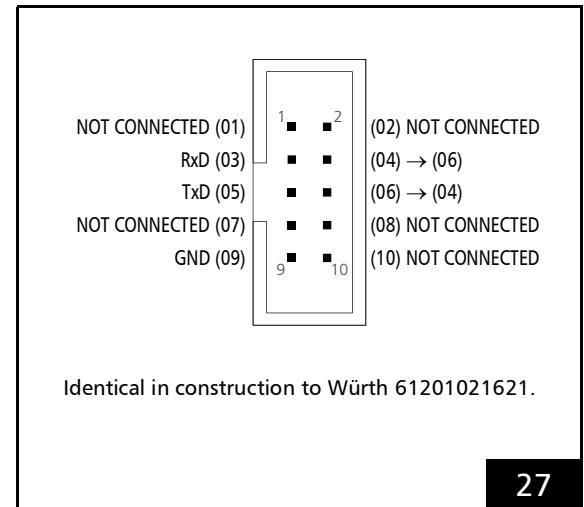
Alternatively, the **Slot Cover with 15-pin D-SUB Connector and 9-pin D-SUB Connector, page 39** is also available.

### 4.6.5 RS232 Socket Connector

The RS232 socket connector has 10 pins and is located on the upper side of the RTC5 PCI Board, see **Figure 5.**

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

The pin-out is shown in **Figure 27.**



27

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.

**rs232\_config** can be used to set the baud rate (default setting: 9600 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 263.**

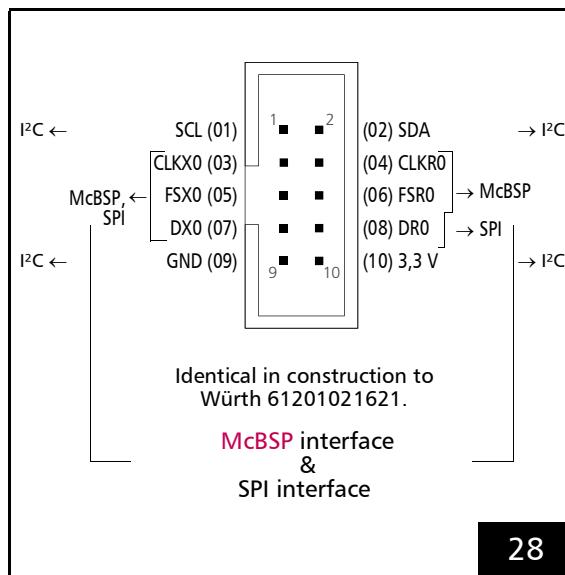
For reading-in data by the RS-232 interface, see **Chapter 9.2.3 "RS-232 Interface", page 264.**

#### 4.6.6 SPI / I<sup>2</sup>C Socket Connector

The SPI/I<sup>2</sup>C socket connector is a hardware interface for bidirectional data exchange:

- **McBSP**
- **SPI**
- **I<sup>2</sup>C (Inter-Integrated Circuit)**

The pin-out is shown in [Figure 28](#).



SPI/I<sup>2</sup>C socket connector: pin-out. The pitch of the pins is 2.54 mm.

#### McBSP Interface

The **McBSP interface**, see [Figure 28](#), is initialized by **load\_program\_file** with:

- `XDelay = RDelay = 1`
- 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  $\mu$ s 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 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 2.6 "Options", page 34](#).
- 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/SPI Signals", page 276](#), and ["Global Online Positioning"](#).
- 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 [Section "Correction via McBSP Interface with Additional McBSP Input", page 231](#). Prerequisite: Option **Processing-on-the-fly**, see [Chapter 2.6 "Options", page 34](#).
- Permanent data output in 10  $\mu$ s cycles. With **set\_mcbsp\_out** and **set\_mcbsp\_out\_ptr** it can be selected which data signals are outputted.

For data output using the **McBSP interface**, see [Chapter 9.1.7 "McBSP Interface", page 263](#).

For data input using the **McBSP interface**, see [Chapter 9.2.4 "McBSP Interface", page 264](#).

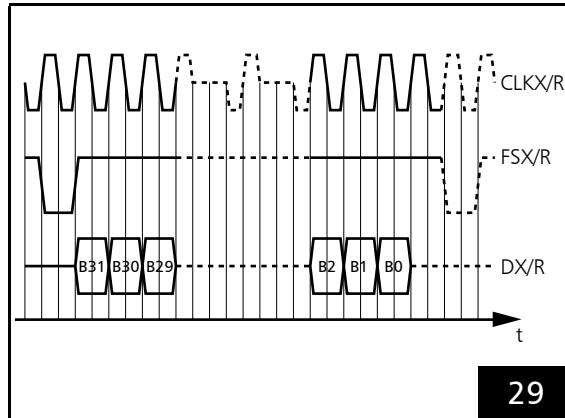
**mcbsp\_init** allows setting a data delay for the **McBSP** data transmission independently for the transmitter and receiver. Possible values range from 0 to 2 (default: 1).

All signals are referenced to GND.

### RTC5 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
  - Data delay *XDelay* bits
- The timing diagram of the McBSP signals is shown in [Figure 29](#). 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  $\mu$ s 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).

### RTC5 as Receiver

The following specifications apply to CLKR0, FSR0, DR0:

- Signal level 3.3 V or 5 V TTL
- McBSP mode:
  - Single phase frame
  - Single element per frame
  - 32 bits per element
  - Data delay *RDelay* bits
- The timing diagram of the McBSP signals is shown in [Figure 29](#). 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](#). Therefore, the data provided 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 a reset by `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 214
    - 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 216
  - 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.

## I<sup>2</sup>C Interface

The SPI/I<sup>2</sup>C socket connector is used to transmit the digital data to the **DSP**, when the **ADC Add-On Board** is operated, see also [Chapter 4.6.8 “Analog Inputs \(Accessory\)”, page 75](#).

At present, the I<sup>2</sup>C clock cycle signals (SCL at pin (1) and I<sup>2</sup>C data signals (SDA at pin (2) are not discretionary for other purposes.

## SPI Interface

The SPI/I<sup>2</sup>C socket connector can be set-up for a serial synchronous data transmission, if required. Then **SPI** functionality is used instead of the McBSP functionality (default). For this purpose, `mcbsp_init_spi` can be called at any time. A data transmission which is in progress is canceled by that. The previous state (McBSP functionality) can be reestablished with `mcbsp_init`.

After initialization only one data word can be transmitted every 10  $\mu$ s, for example, see `set_mcbsp_in`.

The default transmission frequency is 8 MHz (4  $\mu$ s per data word). Alternatively, a value between 4 MHz and 16 MHz can be set. For this purpose, `set_mcbsp_freq` is called (see also `mcbsp_init` or `mcbsp_init_spi`).

After `mcbsp_init_spi` has been called the RTC5 acts as **SPI** master device in terms of the **SPI** standard. The RTC5 is a master device for a single **SPI** slave only.

Necessary for operation are:

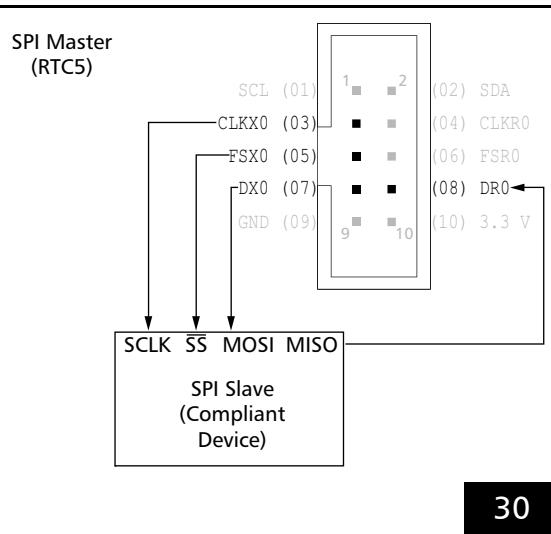
- 1 control line for the clock signal (CLKX0 at pin (3))
- 1 control line for the active-LOW slave select signal (FSX0 at pin (5))
- 1 data line – from a master point of view – outgoing (DX0 at pin (7))
- 1 data line – from a master point of view – incoming (DR0 at pin (8))

Pin (4) CLK0 and pin (6) FS0 are not connected.

The specification 3.3 V TTL or 5 V TTL applies for all signal levels.

The signal sequence is the same as with the McBSP mode, though `XDelay` and `RDelay` are *always* = 1.

The electrical connection of the RTC5 and an **SPI** slave device is shown in [Figure 30](#).



30

RTC5 as **SPI** master with **SPI** slave – signals and electrical connection.

SCLK = Serial Clock, MOSI = Master Output - Slave Input, MISO = Master Input - Slave Output, SS = Slave Select. The relevant abbreviations and designations are inconsistent in technical references. That is, they are arbitrarily chosen in this document and a variety of synonyms does exist.

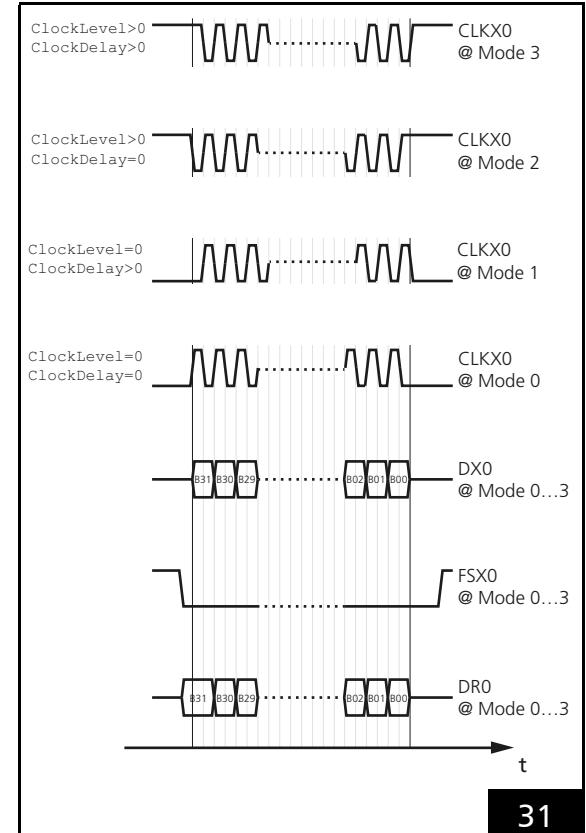
With **SPI** (in contrast to McBSP data transmission), data input and data output take place synchronously together with a clock signal (the clock signal is generated by the **SPI** master). The **SPI** timing diagram is shown in [Figure 31](#).

The clock signal properties 'polarity' (`ClockLevel`) and 'phase' (`ClockDelay`) can be set by `mcbsp_init_spi`. The resulting data transmission formats are designated Mode 0...Mode 3 according to the **SPI** standard, see the following table.

<b>SPI mode</b>	<b>Clock signal polarity</b>	<b>Clock signal phase</b>
0	clock idle LOW – <code>ClockLevel = 0</code>	is simultaneous with data bits – <code>ClockPhase = 0</code>
1	clock idle LOW – <code>ClockLevel = 0</code>	is delayed a half clock signal – <code>ClockPhase &gt; 0</code>
2	clock idle HIGH – <code>ClockLevel &gt; 0</code>	is simultaneous with data bits – <code>ClockPhase = 0</code>
3	clock idle HIGH – <code>ClockLevel &gt; 0</code>	is delayed a half clock signal – <code>ClockPhase &gt; 0</code>

All signals are referenced to PC ground GND.

In contrast to the McBSP configuration, immediately after initialization all signals at the corresponding pins are static and the RTC5 does not generate a clock signal permanently.



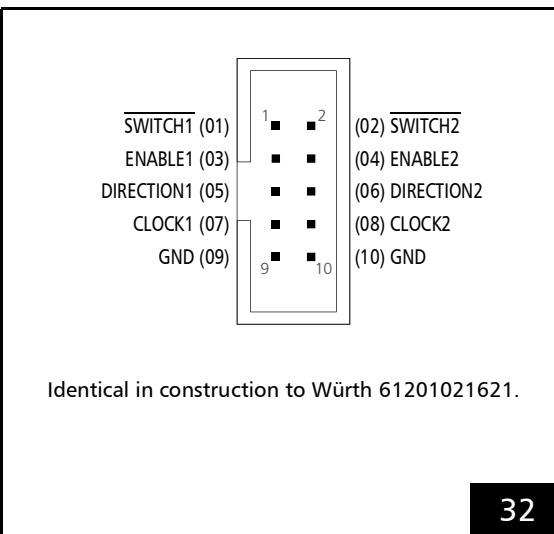
31

Timing-Diagram of the **SPI** signals. The clock signal differs in regards to polarity and phase for mode 0, 1, 2 and 3. Grid spacing is a *half* clock signal.

#### 4.6.7 STEPPER MOTOR Socket Connector

The STEPPER MOTOR socket connector can supply signals for controlling two stepper motors<sup>(1)</sup>.

The pin-out is shown in [Figure 32](#).



32

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

All signals are referenced to PC ground GND.

The output signals (ENABLE, DIRECTION and CLOCK) are TTL-level signals (5 V). The RTC5 generates a periodic clock signal of active-HIGH 5  $\mu$ s pulses (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 SWITCH inputs (active-LOW) are connected internally to +3.3 V by 10 k $\Omega$  pull-up resistors to facilitate integration of a final switch signal (3.3 V or 5 V TTL signal or input referenced to ground).

For programming the stepper motor signals, see [Chapter 9.1.5 "Controlling Stepper Motors"](#), [page 260](#).

#### 4.6.8 Analog Inputs (Accessory)

When the RTC5 PCI Board is equipped with the [ADC Add-On Board](#), then three 10 V analog inputs are available

- ANALOG IN0
- ANALOG IN1
- ANALOG IN2, see [Section "Notes", page 76](#)

After the first [read\\_analog\\_in](#) call, voltages that are applied at the 2 (RTC5 PCIe Board) or 3 ([ADC Add-On Board](#)) analog input ports are

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

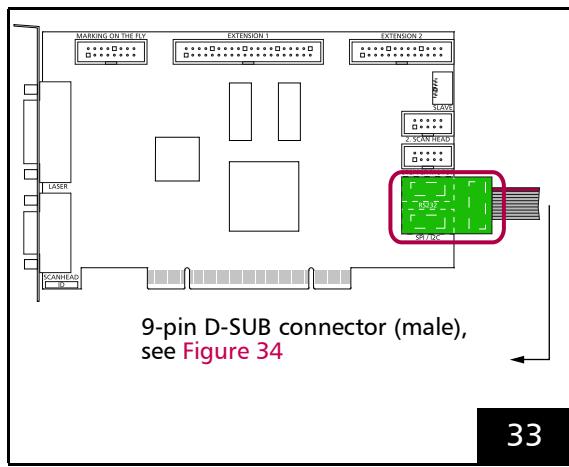
The current input values can be read by the control command [read\\_analog\\_in](#) at any time.

As of version DLL 543, OUT 543, RBF 524: The analog input values ([ANALOG IN0](#) and [ANALOG IN1](#), see [read\\_analog\\_in](#)) can be recorded with signal 54 (see [set\\_trigger/set\\_trigger4](#)). However, old bits are not recorded. The format of the data is ([ANALOG IN1 << 16](#)) + [ANALOG IN0](#)).

#### Installation and Pin-out

The [ADC Add-On Board](#) must be plugged into the SPI/I<sup>2</sup>C and RS232 socket connectors of the RTC5 PCI Board. When installing, ensure that the add-on board is correctly oriented: the soldered-on flat ribbon cable must point outward, see [Figure 33](#).

(1) Stepper motor signals are available only for RTC5 Boards with DSP version numbers 2 and higher (see [get\\_rtc\\_version Bit #16...Bit #23](#)). For older RTC5 Boards, stepper motor commands do not execute.

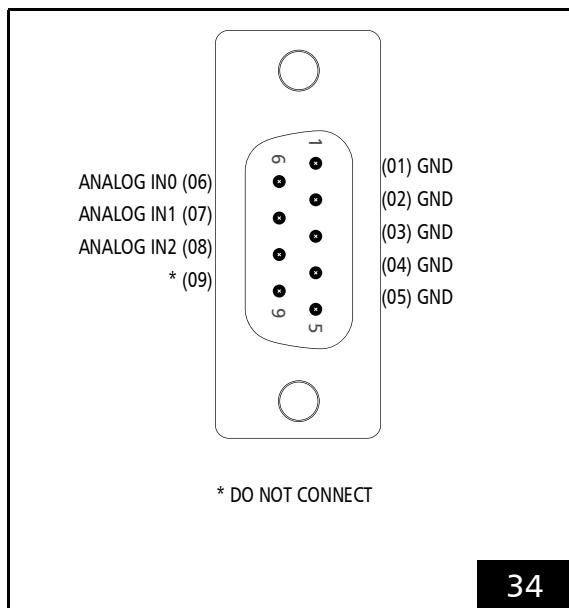


ADC Add-On Board (#0121126) on RTC5 PCI Board.

Internally, the **ADC Add-On Board** is only electrically connected to pins 01, 02, 09 and 10 of the SPI/I<sup>2</sup>C socket connector.

Attachment to the RS232 socket connector is only for mechanical stability.

The analog inputs are then available by a 9-pin D-SUB connector (male) at the soldered-on flat ribbon cable. Its pin-out is shown in [Figure 34](#).



ADC Add-On Board (#0121126): pin-out of the 9-pin D-SUB connector, male.

## Specifications

- Input voltage range: 0 V...10 V
- Input impedance: 4 kΩ
- ADC resolution: 12 bit

The input signals are referenced to ground GND.

## Notes

- The **ADC Add-On Board** even provides a *third* 10 V analog input with identical specifications, see [Figure 34](#):
  - ANALOG IN2
 However, the input values of ANALOG IN2 can neither be queried by `read_analog_in` nor recorded by `set_trigger` or `set_trigger4`.

## 5 Installation and Start-Up

Installation of the RTC5 PCI Board consists of the following steps:

- (1) Checking the jumpers, see [Chapter 5.1 "Checking Jumper Settings", page 77](#).
- (2) Hardware installation: Installing the RTC5 Board, see [Chapter 5.2 "Installing the RTC5 PCI Board", page 77](#).
- (3) Installing the RTC5 board driver, see [Chapter 5.3 "Installing the RTC5 Board Driver", page 78](#).
- (4) Installing the RTC5 software, see [Chapter 5.4 "Installing the RTC5 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 80](#).
- (6) The included HPGL converter program can be used for conducting a post-installation functionality test, see [Chapter 5.6 "Functionality Test", page 81](#).

### 5.1 Checking Jumper Settings

SCANLAB ships RTC5 PCI Boards in various jumper configurations.

- (1) Make sure that the to-be-installed RTC5 PCI Board has the required jumper configuration. If you need to change the factory settings, proceed as described in [Chapter 2.7 "Jumper Settings and Type Identification", page 35](#).
- (2) Proceed with [Chapter 5.2 "Installing the RTC5 PCI Board", page 77](#).

### 5.2 Installing the RTC5 PCI 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 RTC5 in a PC:

- (1) Remove all diskettes and CDs 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 PCI slot and remove the slot cover. A height from the slot connector's top of at least 105 mm is required.
- (6) Remove the RTC5 from the antistatic bag. Do not touch the contacts of the board.

- (7) If you want to use the signals on the RTC5's socket connectors, then attach the appropriate cables. SCANLAB recommends installing additional slot covers with suitable connectors, so that the signals are accessible even when the PC's housing is closed. All RTC5 connectors and signals are described in **Chapter 4 "RTC5 PCI Board – Layout and Interfaces"**, page 51.
- (8) Install the RTC5 into the PCI slot. Observe the instructions in the manual of your PC<sup>(1)</sup>.
- (9) Close the housing of the PC.
- (10) Place the PC at its operational location and connect all peripheral equipment.
- (11) Connect the 9-pin D-SUB connector on the RTC5 – using the **XY2-100 Converter (Accessory)**, if necessary – to the scan system by a data cable (see **Chapter 4.5 "Interfaces to Scan System"**, page 54).
- (12) Connect the 15-pin D-SUB connector on the RTC5 to the laser by a suitable interface (see **Chapter 4.6 "Interfaces for the Laser and Peripheral Equipment"**, page 60).
- (13) If necessary, connect one or more of the RTC5's 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 multiple master/slave-synchronized RTC5 Boards are to be used in a PC, then the RTC5 Boards must first be connected pairwise with each other by the MASTER and SLAVE connectors and then installed in adjacent PCI slots. (see also **Chapter 4.4 "Master Socket Connector, Slave Socket Connector"**, page 53).

## 5.3 Installing the RTC5 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 RTC5 board driver.
    - (2) Run `ScanlabClassChecker.cmd` as Administrator (part of the delivered RTC5 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 RTC5 board driver for Windows 10, 8, 7, proceed as follows:

- After the RTC5 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 folder that includes the RTC5 board driver. During installation, the operating system automatically selects the appropriate driver file (32- or 64-bit).

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 folder that includes the RTC5 board driver.
- (2) Run `ScanlabClassChecker.cmd` as Administrator (part of the delivered RTC5 software package; see there also the background information in [ReadMe\\_ScanlabClassChecker.pdf](#)).

### Notice!

- The RTC5 PCI Board does not support power-saving modes that switch off power to the PCI 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 RTC5 software package (under `RTC5 Tools`) to do this.
- See also [Section "Notes", page 33](#).

### 5.3.1 RTC5 Software Package Upgrades

If you currently use an RTC5 software package version 2012\_09\_05 or earlier (still contains WDM driver) and you want to upgrade to an RTC5 software package version 2013\_02\_21 or later (contains already WDF driver), then proceed as follows:

- (1) First install only the RTC5 software files, see [Chapter 5.4 "Installing the RTC5 Software", page 80](#).
- (2) Test the proper functioning of your user program.
- (3) Only afterwards install the new driver (this is a WDF driver). This is because quickly changing back to older RTC5 software files is then only possible if you also change the driver (see also note on [page 33](#)).
- (4) Right-click on the delivered script `ScanlabClassChecker.cmd` and choose 'Run as Administrator' from the appearing list. For further information, refer to the file [ReadMe\\_ScanlabClassChecker.pdf](#).

### 5.3.2 Exchange of RTC Boards and Update of RTC Board Driver

After the exchange of any RTC board of type RTC3, RTC4, RTC SCANalone, RTC5 for an RTC5 (also: RTC5 PCIe Board) or, to update an (outdated) WDM driver to a (newer) WDF driver proceed as follows:

- (1) Install the latest driver for the new board (this is a WDF driver; downloadable from the SCANLAB homepage).
- (2) Right-click on the delivered script `ScanlabClassChecker.cmd` and choose 'Run as Administrator' from the appearing list. For further information, refer to the file [ReadMe\\_ScanlabClassChecker.pdf](#).

## 5.4 Installing the RTC5 Software

Additionally to installing the drivers, the RTC5 software must be copied and unzipped onto the hard drive of the PC. The following files are required:

- **RTC5 DLL**
  - **RTC5DLL.dll**  
Win32-based RTC5 dynamic link library  
or
  - **RTC5DLLx64.dll**  
Win64-based RTC5 dynamic link library
- **RTC5 files**
  - **RTC5OUT.out**,  
Program file for the **DSP**
  - **RTC5RBF.rbf**  
Firmware file for the **FPGA**
  - **RTC5DAT.dat**  
Binary auxiliary file

The files are included in the RTC5 software package. For easy identifying and archiving of different software versions, some of the files are also delivered zipped (the `zip` file names `RTC5<..>_<Version>.zip` include the version numbers).

To install the RTC5 software, follow these steps:

- Copy or unzip the files `RTC5DLL.dll` (or `RTC5DLLx64.dll`), `RTC5OUT.out`, `RTC5RBF.rbf` and `RTC5DAT.dat` (of desired version) to the hard drive of your PC.

When replacing individual software files, note that differing program 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 RTC5 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 scan system's power supply.
- (3) Turn off the peripheral equipment.
- (4) Close the control software and turn off the PC containing the RTC5 Board.



### Caution!

- While the PC is turned on and off, short-term level variations at the output ports of the RTC5 PCI 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 motions 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 28](#). This program lets you load graphics files in Hewlett Packard HPGL format (vector graphic plotter files `*.PLT`) for transfer to the RTC5.

- (1) Copy the HPGL converter and the supplied demo file into the same folder as the [RTC5 DLL](#), the [RTC5 DSP](#) program file(s) and correction file(s).
- (2) Start the HPGL converter.
- (3) Choose **Options > Correction**, and then select a correction file.
- (4) Choose **File > Load HPGL-File**, and then select a `*.plt`.
- (5) To start output, choose **Mark > Start Marking**.

## 5.7 User Programs and Demo Software

The DLLs for RTC5 user programs ([RTC5DLL.dll](#), [RTC5DLLx64.dll](#)) support the RTC5 under 32-bit and 64-bit Microsoft operating systems Windows 10, 8, 7. The DLLs provide all required functions for operating the RTC5.

Programming of user programs is described in detail in [Chapter 6 "Developing RTC5 User Programs", page 83](#). [Chapter 6.2 "Initialization and Program Start-Up", page 84](#) shows how to import the functions of the [RTC5 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 RTC5 board driver supports function calls from Win64-based applications as well as from Win32-based applications.

Therefore, existing Win32-based applications for the RTC5 are able to execute even on 64-bit systems, if the Win32-based [RTC5DLL.dll](#) from the RTC5 software package is used. For Win64-based applications, the Win64-based [RTC5DLLx64.dll](#) is included in the RTC5 software package. In case a user program utilizes implicit linking to the [RTC5DLLx64.dll](#), it must be linked with the Win64-based import library [RTC5DLLx64.lib](#).

To help programmers get started, some example codes are listed on [page 89](#), [page 121](#) and [page 130](#). In addition the RTC5 software package includes a variety of demo programs (see [page 728](#)).

### 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.



## 5.8 Exchanging RTC5 Boards

If an already installed RTC5 board of an older type is replaced by an RTC5 board of a newer type, then it is recommended to carry-out a software update.

This is particularly important when RTC5 Boards with **DSP** version 0 get replaced by RTC5 Boards with **DSP** version 2 (**get\_RTC\_version** Bit #16...Bit #23). The suitable RTC5 software package can be found on the delivered CD. You can also download the latest files From the SCANLAB website you can also download the latest RTC5 software package.

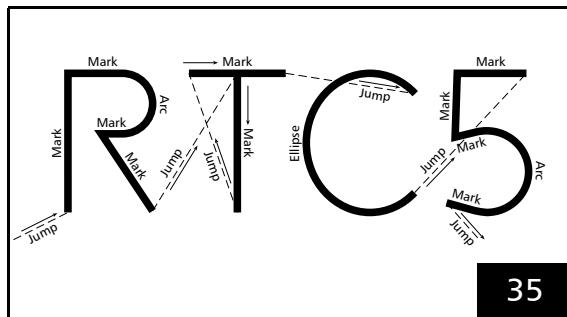
See also [Chapter 5.3.2 "Exchange of RTC Boards and Update of RTC Board Driver", page 79](#).

## 6 Developing RTC5 User Programs

### 6.1 RTC5 Software Concept Basics

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

The SCANLAB RTC5 PCI Board and its related software are designed for controlling scan systems and lasers. To illustrate the principle of operation, Figure 35 shows a simple laser marking sample<sup>(1)</sup>.



A laser marking sample.

The lettering is made up of straight line segments (vectors) and arc segments. The **RTC5 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 software commands are available for controlling the laser during the marking process. The RTC5 processes the commands it receives and precisely transmits the required marking signals to the scan head (using a predefined 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.

The current status of a scan head can also be queried via the RTC5 PCI Board using appropriate commands.

For laser control, the RTC5 provides various analog and digital signal outputs freely available for tailoring laser control to customer-specific requirements.

#### 6.1.2 Control Commands and List Commands

The RTC5 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 RTC5, 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 94](#)). List commands sent to the RTC5 afterwards are not executed immediately, but stored in a list memory area. The RTC5 begins reading the commands from the list memory area and processing them in real time only after the list is started.

Some list commands, which are called short list commands, do not require the full 10 µs clock cycle for command execution - instead, they execute along with the next list command, one directly after the other within a single 10 µs clock cycle, during which control commands cannot execute. The total list processing time is thereby reduced.

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.

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

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

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 RTC5 commands are described in detail in [Chapter 10 "RTC5 Commands", page 278](#). An overview is provided in [Chapter 10.1 "Overview", page 278](#).

## 6.2 Initialization and Program Start-Up

To use the RTC5's commands and functions in a user program:

- The RTC5 must be fully installed – this includes the RTC5 hardware, driver and software (see [Chapter 5 "Installation and Start-Up", page 77](#))
- The desired **RTC5 DLL** (Win32- or Win64-based) must be linked to the user program (see [Chapter 6.2.1 "DLL Calling Convention", page 84](#))
- The DLL functions must be imported to the user program (see [Chapter 6.2.2 "Importing Commands", page 84](#))

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

- that initialize the **RTC5 DLL** for the calling user program and assigns access rights for the installed RTC5 Boards (see [Chapter 6.2.3 "Initializing the RTC5 DLL and Board Management", page 86](#))
- that initializes the installed RTC5 Boards themselves (see [Chapter 6.2.4 "Start of RTC5 PCI Board Operation", page 87](#)).

Only afterward can the user program load its command lists into the RTC5's list memory and start processing.

These steps are described individually in the following sections and summarized by a simple example program in the last section of this subchapter (see [page 89](#)).

### 6.2.1 DLL Calling Convention

Link the user program to the [RTC5DLL.dll](#)/[RTC5DLLx64.dll](#)<sup>(1)</sup>.

The **RTC5 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 **RTC5 DLL** into a C, C++, C# or Pascal user program, the RTC5 software package contains corresponding utility files.

#### Notice!

- Some RTC5 commands can only be *completely* executed with appropriate options, see also [Chapter 2.6 "Options", page 34](#).
- **get\_RTC\_version** provides information about the options installed on your RTC5 PCI Board.

#### Pascal

Use the file [RTC5Import.pas](#) as a unit and call the RTC5 commands you need, like

`goto_xy(1000, 2500);`

for performing a jump to location 1000, 2500.

(1) See [Chapter 5.4 "Installing the RTC5 Software", page 80](#).

## 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 `RTC5impl.h` and link with the (C) import library `RTC5DLL.lib` (for Win32-based applications or with `RTC5DLLx64.lib` for Win64-based applications) for building the executable.

Call the RTC5 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 `RTC5expl.h`. Before calling any RTC5 function, initialize the **RTC5 DLL** by calling the function `RTC5open` (which is defined in the file `RTC5expl.c`). When you are finished using the RTC5, close the **RTC5 DLL** by calling the function `RTC5close` (also defined in the file `RTC5expl.c`).

For building the executable, link with the file `RTC5EXPL.OBJ`, which you can generate from the source code `RTC5expl.c`.

Call the RTC5 commands you need, like

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

### Pros and Cons of Implicit and Explicit Linking

	Implicit Linking	Explicit Linking
Necessary Files	<code>RTC5impl.h</code> or <code>RTC5impl.hpp</code> , <code>RTC5DLL.lib</code> or <code>RTC5DLLx64.lib</code>	<code>RTC5expl.h</code> , <code>RTC5expl.c</code>
Advantage	Easiest linking method	Eliminates the need to link the user program with an import library
Negative Aspect	Need to link the user program with a compiler-specific import library	Need to initialize ( <code>RTC5open</code> ) and close ( <code>RTC5close</code> ) the DLL

## 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 `RTC5impl.hpp` instead of `RTC5impl.h`. Otherwise, the command import is realized as in C (see above).

## C#

For implicit linking of the **RTC5 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 RTC5 DLL and Board Management

As many RTC5 PCI Boards can be operated in one PC at the same time as the PC provides PCI slots.

The **RTC5 DLL** allows multi-processing<sup>(1)</sup> as well as multi-threading.

However, no board can be simultaneously used by multiple applications. Access rights (even if temporary) to existing boards are assigned on an exclusive basis by the DLL. Multiple threads of one user program can use the same board, but can not send commands to it at the same time (the **RTC5 DLL** automatically serializes the command calls).

The **RTC5 DLL**-internal RTC5 board management coordinates usage of different boards by different applications. RTC5 board management is initiated by **init\_rt5\_dll**.

**init\_rt5\_dll** is a prerequisite for RTC5 access and must be called at the beginning of every user program – even if only one RTC5 Board is to be used by only one user program.

#### **init\_rt5\_dll:**

- searches for all existing RTC5 Boards
- establishes corresponding 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 **RTC5 DLL**-internal numbers for all found RTC5 Boards (important for multi-board commands)
- sets one board as the active board, which is the target for non-multi-board commands

Further commands enable subsequent changes to access rights and changing the active board. Usage of multiple boards is described in [Chapter 6.6 "Using Several RTC5 PCI Boards in One PC", page 112](#); usage by multiple applications is described in [Chapter 6.7 "Usage of RTC5 PCI Boards by Several User Programs", page 117](#).

After **RTC5 DLL** initialization by **init\_rt5\_dll**, users can additionally select one of two operation modes (see **set\_rt5\_mode** and **set\_rt4\_mode**). The default is **RTC5 Standard Mode**. An

**RTC4 Compatibility Mode** is also provided so that applications written for the **RTC4** can be processed by the **RTC5** (to a large extent) without modification. However, a prerequisite here is that the program can only contain **RTC4** commands that also exist with unchanged functionality as **RTC5** commands. In the command reference of this manual ([Chapter 10.2 "RTC5 Command Set", page 290](#)), when applicable, notes such changes in the "**RTC4**→**RTC5**" section (see also [Section "Increased Parameter Resolution", page 43](#)).

(1) Several user programs can run simultaneously.

### 6.2.4 Start of RTC5 PCI Board Operation

At the beginning of every RTC5 user program – after initialization of the **RTC5 DLL**, see [Chapter 6.2.3 "Initializing the RTC5 DLL and Board Management", page 86](#) – it is recommended to carry out the following sequence of steps in order to start RTC5 PCI Board operation.

- (1) [Initializing the Board, page 87](#)
- (2) [Configuring the Board, page 87](#)
- (3) [Initializing the Scan System Control, page 88](#)
- (4) [Initializing the Laser Control, page 88](#)
- (5) [Loading and Executing Lists, page 88](#)

#### Initializing the Board

- Call **load\_program\_file**. For the individual actions, see [Function, page 430](#). 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)(3)</sup>. In order to be able to combine **RTC5 files** (**RTC5DLL.dll**/**RTC5DLLx64.dll**, **RTC5OUT.out**, **RTC5RBF.rbf**, **RTC5DAT.dat**), they must have certain file versions, see also [Chapter 5.4 "Installing the RTC5 Software", page 80](#). **load\_program\_file** performs a version compatibility check. If a version error exists (error code **7** and **get\_last\_error** return code **RTC5\_VERSION\_MISMATCH**), the board is released by **release\_RTC**. Thus, it is not available for further RTC5 commands other than those not requiring granted access rights.

RTC5 PCI Boards can only be operated if all **RTC5 files** are available with a compatible combination of versions, see [Chapter 5.4 "Installing the RTC5 Software", page 80](#). If the version compatibility check detects an error and the board's access rights are not assigned to another user program, the multi-board command **n\_load\_program\_file** can be used to load a correct program version, followed by **select\_RTC** or **acquire\_RTC** to access the board. If multiple RTC5 Boards are connected as master and slave, then **load\_program\_file** must have been called on all boards and a clock phase synchronization should be performed (see [Chapter 6.6.3 "Master/Slave Operation", page 113](#)).

#### Configuring the Board

- If necessary, configure the RTC5 list memory by **config\_list**. By default, the list memory is preconfigured such that the memory areas "List 1" and "List 2" can each store 4,000 list commands after **load\_program\_file**. The protected list memory area "List 3" comprises the remaining 1,040,576 of the  $2^{20}$  storage positions.

- (1) Center of the **Image** field.
- (2) There are no laser control signals at the corresponding pins (LASERON, LASER1, LASER2). They are in high-impedance state.
- (3) On the state of the various output ports, see [page 431](#).

## Initializing the Scan System Control

- (1) Use `load_correction_file` to download the necessary correction file(s) to the RTC5 (you can load a correction table before or after `load_program_file`<sup>(1)</sup> but you should load it before `select_cor_table`; you should at least load a 1to1 correction table).  
See [Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 221](#), 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`. 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 (for instance "Variable `Polygon Delay`" or constant `Polygon Delay`; command `set_delay_mode`).
- (4) If necessary, 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.

## 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 (laser timing or [Laser Delays](#)) are set by further control commands or list commands, see example in [Chapter 7.1.4 "Example Code \(C\)", page 130](#) 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.

(1) `load_program_file` deletes correction tables number 3 and 4.



### 6.2.5 Example Code (C)

The following C source code for a console user program (environment: Win32) illustrates the programming fundamentals of **RTC5 DLL** and **RTC5 PCI Board** initialization (for complete demo programs, see [Chapter 11 "Demo Programs", page 728](#)).

Necessary sources: **RTC5impl.h**, **RTC5DLL.lib** (for implicit linking) or **RTC5expl.h**, **RTC5expl.c** (for explicit linking) to link the **RTC5 DLL** to the program (see [Chapter 6.2.1 "DLL Calling Convention", page 84](#)). If the operating system does not find the **RTC5DLL.dll** on user program startup, it produces a corresponding error message and terminates the program.

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

// RTC5 header file for implicitly linking to the RTC5DLL.dll (for building the executable, also link
// with the (Visual C++) import library RTC5DLL.lib):
#include "RTC5impl.h"
// Alternatively: RTC5 header file for explicitly linking to the RTC5DLL.dll (for building the executable,
// link with the file RTC5EXPL.OBJ, which you can generate from the source code RTC5expl.c):
//#include "RTC5expl.h"

void __cdecl main( void*, void* )
{
    // only for explicitly linking:
    // if ( RTC5open() ) // error detected, RTC5open returns 0 for no error
    // {
    //     printf( "Error: RTC5DLL.dll not found\n" );
    //     terminateDLL();
    //     return;
    // }

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

    UINT ErrorCode;

    // Initializing the RTC5 DLL (the following command must be called as the first RTC5 command)
    ErrorCode = init_rtc5_dll();

    // Following init_rtc5_dll 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 \(C\)", page 121.
}
```



```

// Initializing the RTC5 PCI Board:
// - Selecting the RTC5 Board number 1 as the active Board for this user program
// - If desired: Selecting the RTC4 Compatibility Mode as operation mode
// (default: RTC5 Standard Mode).
// - Stopping any list running on RTC5 PCI Board number 1
// (if this board has been used previously by another user program,
// a list might still be executed. This would prevent load_program_file and load_correction_file
// from being executed).
// - Calling load_program_file for resetting 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 (C)", page 121).
// - Clearing all previous error codes (stop_execution might have created an RTC5_TIMEOUT or
// RTC5_BUSY error).
// - Configuring the RTC5 list memory, default: 4000 storage positions for list 1,
// 4000 for list 2).
(void) select_rtc( 1 );
set_rtc4_mode();
stop_execution();
ErrorCode = load_program_file( 0 ); // Path = 0: path of current working directory
if ( ErrorCode )
{
    printf( "Program file loading error: %d\n", ErrorCode );
    free_rtc5_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 in Chapter 7.1.4 "Example Code (C)", page 130.

// End of main program
terminateDLL();
return;
}

void terminateDLL()
{
    printf( "- Press any key to shut down \n" );
    while( !kbhit() );
    (void) getch();
    printf( "\n" );
    // Close the RTC5 DLL
    free_rtc5_dll();
    // only for explicitly linking:
    // RTC5close();
}

```

## 6.3 RTC5 List Memory

The RTC5 list memory serves as intermediate storage for list commands.

Before list commands can be transferred to the RTC5 PCI 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 94](#).

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 RTC5 list memory offers  $1M = 1,048,576 = 2^{20}$  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 RTC4, RTC3 or RTC2 Boards, for example, for continuous loading and processing of command lists.

However, the RTC5 PCI 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 RTC5 List Memory", page 92](#).

For list handling, see [Chapter 6.4 "List Handling", page 94](#).

#### "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 RTC5 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 101](#).

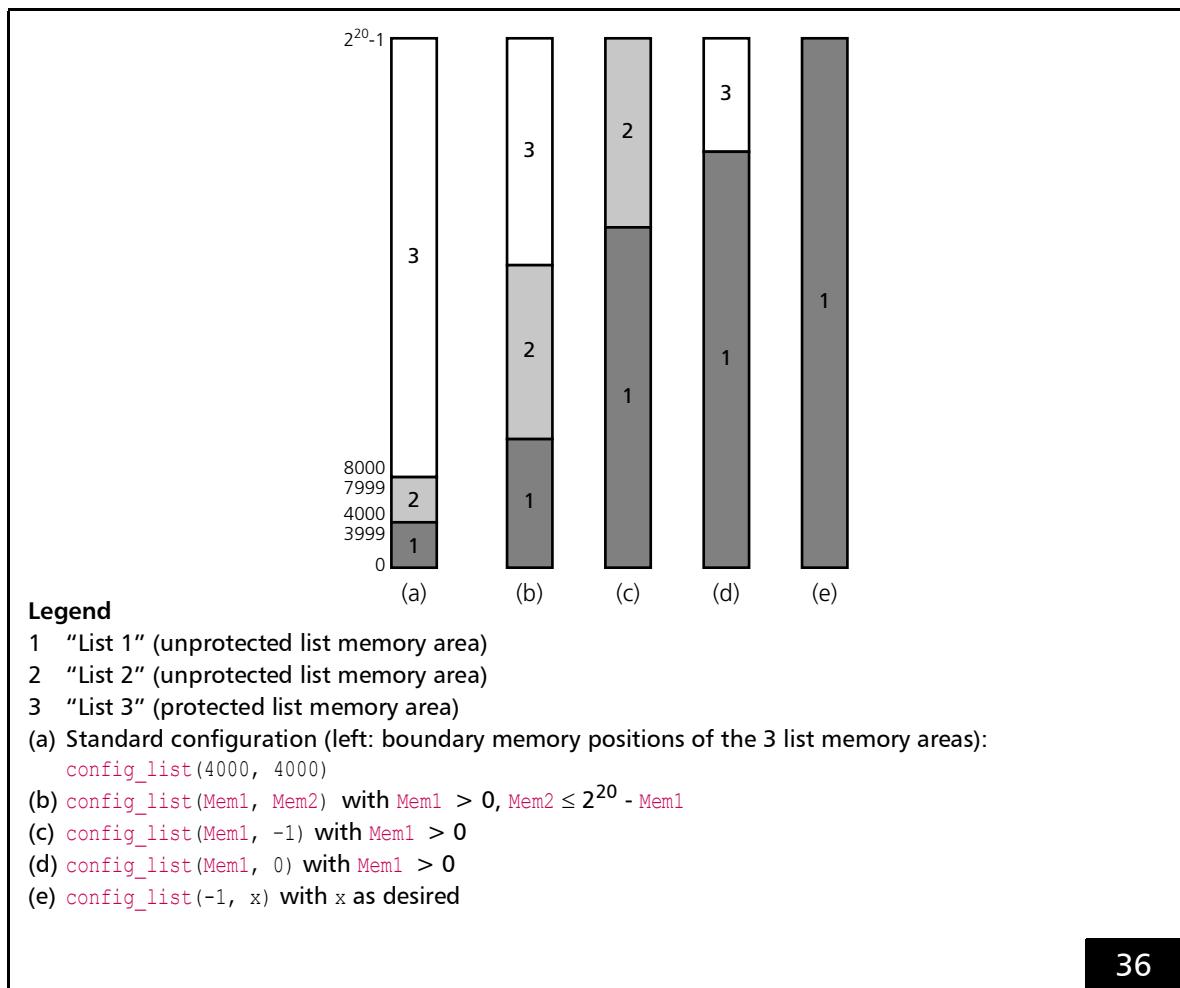
### 6.3.2 Configuring the RTC5 List Memory

For compatibility, `load_program_file` configures the RTC5 list memory area such that “List 1” and “List 2” can each accept 4,000 list commands by default.

The protected “List 3” then owns the remaining 1040576 of the  $2^{20}$  storage positions (see Configuration (a) in [Figure 36](#)).

By `config_list`, the list memory areas can be reconfigured.

For example, if a user program only needs one list, then the RTC5’s entire memory can be treated as a single list with a total capacity of  $2^{20}$  positions (see Configuration (e) in [Figure 36](#)). In this case, the single list (“List 1”) can be loaded with up to  $2^{20}$  commands.



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  $2^{20}$ .

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 (d) or (e) in [Figure 36](#).

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 109](#).
- 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 104](#).



## 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.

### 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 called, 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:

- to the beginning of the selected list by
  - `set_start_list`
  - `set_start_list_1`
  - `set_start_list_2`
- to the specified address of the selected list by
  - `set_start_list_pos`
  - `set_input_pointer`

If needed, the current positions of the input pointer 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 RTC5 PCI 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 100](#).

## Terminating Lists

A command list 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 "Automatic List Changing", page 99](#).

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 list status values (compare to [Chapter 6.4.3 "List Execution Status", page 97](#)).

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

- **LOAD** list status

The **LOAD** list status **LOAD1** or **LOAD2** indicates that the input pointer is currently in this list. In any case, the **LOAD** list status of the other list is *not* set.

- **READY** list status

The **READY** list status **READY1** or **READY2** is set when a **set\_end\_of\_list** is written into the list during the loading procedure. The **READY** list status is reset when the **LOAD** list status of the list is newly set.

- **BUSY** list status

The **BUSY** list status **BUSY1** or **BUSY2** 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** list status of the other list are then *not* set. The **BUSY** list 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** list status still remains set.

- **USED** list status

The **USED** list status **USED1** or **USED2** is set when a **set\_end\_of\_list** is reached during processing. The **USED** list status is reset when the **LOAD** list status of the list is set.

### 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** list status is set for both lists (as by initialization). See also **load\_list( ListNo = 3 )**.
- If you want to explicitly set the **USED** list 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 **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 **USED** list status 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).

### 6.4.3 List Execution Status

In addition to the list status values, see [Chapter 6.4.2 "List Status", page 96](#), list execution status values are provided.

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

- **BUSY** list execution status

The **BUSY** list execution status is set when:

- the RTC5 PCI Board currently processes one of the two lists
- a list has been paused by the control commands `pause_list` or `stop_list`

The **BUSY** list execution status is not set when a list has been paused by the list command `set_wait` (is set again by a subsequent `release_wait`).

- **PAUSED** list execution status

The **PAUSED** list execution status is set when processing of the list has been paused 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 99](#).

- **INTERNAL-BUSY** list execution status

The **INTERNAL-BUSY** list execution status is set when the RTC5 PCI Board is busy with executing a control command, which needs more than  $10 \mu\text{s}$  for executing a scan motion (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`).

### Notes

- The **INTERNAL-BUSY** list execution status and the **BUSY** list execution status cannot be set at the same time.
- With the RTC5 PCI Board, the **BUSY** list execution status is also available as **BUSY OUT** signal at:
  - LASER connector, see [Figure 17](#)
  - EXTENSION 1 socket connector, see [Figure 22](#)
  - MARKING ON THE FLY socket connector, see [Figure 25](#)
- Some control commands are ignored (not executed), when the **BUSY** list execution status and/or **INTERNAL-BUSY** list execution status are set (for example, `auto_cal`, `goto_xy`, `load_correction_file`) or – with set **INTERNAL-BUSY** list execution status – are only executed with delay after the **INTERNAL-BUSY** list execution 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 by:

- The control command `execute_list`
- An external start signal, see [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 265](#)

`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 265](#), then `set_extstartpos_list` allows definition of a start address for the [External Start](#).

The RTC5 PCI Board starts the execution immediately. Even during 10  $\mu$ s-clocked execution of the list commands, you can still send control commands to the RTC5 PCI 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 signal is transferred to the RTC5 PCI Board, the currently executed list is aborted and the [Signals for "Laser Active" Operation](#) 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 RTC5 PCI 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 "Automatic List Changing", page 99](#). 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 RTC5 PCI Board provides corresponding control inputs, see [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 265](#).
- `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 RTC5 PCI Board reaches a break point during list execution, see [Chapter 6.5 "Structured Programming", page 101](#), 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, list processing can be resumed (at the list command that follows) by the control command `release_wait`.

By `set_wait` the **PAUSED** list execution status (queryable by `get_status`) is set and the **BUSY** list execution 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`, Signals for "Laser Active" Operation 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 motions (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** list execution status (queryable by `get_status`) is set and is reset by `restart_list`. The **BUSY** list execution status is left unchanged by both commands.

## 6.4.6 Automatic List Changing

If the RTC5 list memory is configured for two list memory areas "List 1" and "List 2", see [Chapter 6.3.2 "Configuring the RTC5 List Memory", page 92](#), 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 RTC5 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 94](#).
- 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 94](#), and so-called short list commands can be used, see [Chapter 6.1.2 "Control Commands and List Commands", page 83](#)). Otherwise, "old" commands might be unintentionally executed.
- The RTC5 PCI Board does not support the RTC4 circular queue mode, see [Chapter 6.5.4 "RTC4 Circular Queue Mode", page 110](#). However, this operating mode can also be effectively replaced using an alternating list change and **load\_list** described above.

## 6.5 Structured Programming

The RTC5 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 91](#).
- A list boundary should not run through a subroutine.
- A subroutines 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 94](#)). 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 105](#). 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.  
*Valid as of version OUT 540:* 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 RTC5 PCI Board automatically.

For more information on index management, see [Section "Index Management and Defragmentation", page 104](#).

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 104](#).

## Notes on Not-Indexed Calls

- Index management or defragmentation, see [Section "Index Management and Defragmentation", page 104](#), 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 109](#). Therefore, absolute jumps cannot be used in indexed subroutines.
- When the subroutine is processed, also ignored are:
  - Relative jumps that exceed the boundaries of an indexed subroutine
  - **Jump commands** which initiate a jump to themselves

## 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 commands** 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 Commands and "Arc" Commands", page 124](#), 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 commands** 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 ("normal" calls and "**AbsCalls**") dependent on external control signals, additional commands are available for conditional branching during program execution (see [Chapter 9.3.2 "Conditional Command Execution", page 271](#)).

### 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 104](#).

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 109](#).

With converting to indexed subroutines (method 2), also all other programming rules for indexed subroutines must be observed, see [Section "Rules for Programming", page 103](#).

*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 RTC5 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 RTC5 PCI 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 RTC5 PCI 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 RTC5 PCI 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 102](#) and [Section "Subsequent Protection and Conversion of Non-Indexed Subroutines", page 105](#).

#### 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

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 196](#).

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. Make sure that no unwanted memory overflow of the respective memory area occurs.

### Defining Indexed Text Strings for Times, Dates and Serial Numbers

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 RTC5 PCI 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 102](#) and [Section "Subsequent Protection and Conversion of Non-Indexed Subroutines", page 105](#).

### 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 `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 196](#).

## 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 104](#), 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 RTC5 PCI 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 RTC5 PCI 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 RTC5 PCI 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 RTC5 PCI 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 271](#).
- **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 RTC5 PCI Board, the RTC4 circular queue mode does not exist.

Nevertheless, users can actually replace this operational mode with the RTC5 PCI 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** list execution status), without needing to explicitly specifying the number of the list, see also [Section "Alternating List Changes", page 100](#) and [Section "Loading with Protection", page 95](#).

## 6.5.5 Loops

Although list jumps, see [Chapter 6.5.3 "Jumps", page 109](#), and conditional jumps, see [Chapter 9.3.2 "Conditional Command Execution", page 271](#), 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 109](#).

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 RTC5 PCI Boards in One PC

As many RTC5 PCI Boards can be operated in one PC at the same time as the PC provides PCI slots.

The RTC5 PCI Boards work completely independently of each other. The command lists of all boards can be loaded and executed at any time.

There are two different methods for writing user programs using several RTC5 PCI Boards:

- “Multi-Board Programming”, page 112
- “Single-Board Programming”, page 113

### 6.6.1 Multi-Board Programming

In this programming method, the multi-board versions (command names with prefix “n\_”) of the RTC5 commands are used.

Compared to single-board commands (command names without prefix “n\_”), the board number<sup>(1)</sup> (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 RTC5 PCI Boards are numbered in the order found during initialization (starting with 1).

The multi-board command `n_get_serial_number` can be used to determine which RTC5 PCI Boards have been assigned numbers. See also example (3) below.

`rtc5_count_cards` returns the number of RTC5 PCI Boards in the RTC5 board management.

#### Notes

- Multi -board commands are sent to the active board (default board), if the specified number is `> rtc5_count_cards` or 0 (real boards begin at 1).
- If no real card is entered in the RTC5 board management under the specified number, the Multi -board command is rejected.

All multi-board commands are listed in [Chapter 10.1 “Overview”, page 278](#). 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 “RTC5 Command Set”, page 290](#)).

#### Examples (Pascal)

(1) Write a `Jump command` to the point (500, 500) into the current list of RTC5 PCI Board #1:

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

(2) Process list with number `list_no` (1 or 2) on the RTC5 PCI Board with the number specified by the variable `RTC5_no`:

```
n_execute_list(RTC5_no, list_no)
```

(3) Return the serial number of RTC5 board #1:

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

(1) As an unsigned 32-bit value.

### 6.6.2 Single-Board Programming

During single board programming, one of the inserted RTC5 PCI 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 RTC5 PCI Board.

#### Notice!

- **select\_rtc** defines the active RTC5 PCI 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 RTC5 PCI 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 PCI slots. Connect the Master connector of one board to the Slave connector of another board. Suitable connection cables (see [Figure 8](#)) are available from SCANLAB.

An RTC5 PCI Board automatically gets the master board of a master/slave chain, if a further RTC5 PCI Board is connected to its Master connector but no further RTC5 PCI Board is connected to its Slave connector. All other RTC5 PCI Boards are slave boards. See also [Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 53](#).

**get\_master\_slave** can be used to query separately for each RTC5 PCI 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 RTC5 PCI Board is the master and which one is slave, see [Section "Example Code \(Delphi\)", page 116](#).

## Initialization

On all RTC5 PCI 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 RTC5 PCI 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 RTC5 PCI 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`.

After synchronization, the clock phase of each slave board is (reproducibly) delayed by approx. 0.16  $\mu$ s 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).

## 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 160 ns clock phase shift.
- The correction file and lists must be loaded separately onto all RTC5 PCI Boards.

## Matching of Short-Command Counts

The maximum allowed number of directly consecutive short list commands within a 10  $\mu$ s clock cycle depends on the `DSP` version. To ensure that short list commands execute synchronously even when using multiple RTC5 Boards with differing `DSP` versions in a master/slave chain, the `sync_slaves` command (if sent to the master board) reduces the short list count on all faster boards in the master/slave chain to equal that of the slowest board (that is, the board with the lowest `DSP` version number).

## Notes

- The CPU clock frequency is not altered, only the count of short list commands.
- You can also use the `set_dsp_mode` command to make appropriate individual adjustments for each RTC5 Board.
- But note that some commands (for example, `mark_ellipse_abs`) are only available on boards with higher-numbered `DSP` versions. Adjusting the short-command count does not change this fact. To ensure that the master/slave chain remains synchronized, only use commands that are available even for the board with the lowest `DSP` version number.



## Synchronous Starts and Synchronous Stops

Within a master/slave chain, **External Starts** (if enabled with `set_control_mode`) and **External Stops** are passed on:

- from one board to all downstream slave boards

Therefore, it can be triggered:

- A synchronous start of all boards (with presettable track delays) by an external start signal, a `simulate_ext_start` or a `simulate_ext_start_ctrl` at the master board
- A synchronous stop of all boards by an external stop signal or a `simulate_ext_stop` at the master board

In contrast, *not* passed on are:

- Starts by `execute_list`
- Starts by `execute_at_pointer`
- Stops by `stop_execution`

Therefore, these must be separately executed even at master/slave-synchronized boards.

## Notes

- See also [Chapter 9.3.1 "Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization", page 265](#).



### Example Code (Delphi)

The following example Delphi source code shows how to check which RTC5 PCI 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 (C)", page 89.

```
if init_RTC5_dll() <> 0 then halt; // Initialize the RTC5 DLL
if RTC5_count_cards() <> 2 then halt; // Are 2 RTC5 in the PC?

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

// Detect master board
// Is board 1 the master and board 2 a single slave?
if (n_get_master_slave(1) = 2) and (n_get_master_slave(2) = 1) then
begin
  Master := 1;
  Slave := 2;
end else
// Is board 2 the master and board 1 a single slave?
if (n_get_master_slave(1) = 1) and (n_get_master_slave(2) = 2) then
begin
  Master := 2;
  Slave := 1;
end
else
halt; // Something wrong with master-slave configuration

n_select_cor_table(Master, 1, 0);
n_select_cor_table(Slave, 1, 0);

n_set_control_mode(Master, 1 + 8); // Master slave, activate Start Stop control
n_set_control_mode(Slave, 1 + 8); // For external control, you must use the master
// board's control inputs
n_sync_slaves(Master); // Synchronize master and slave boards
// check synchronization status at any time
// provide an External Start or a simulated External Start before checking
n_simulate_ext_start_ctrl(Master);
Result = n_get_sync_status(Master); // must be 640
Result = n_get_sync_status(Slave); // must be < 11
```

## 6.7 Usage of RTC5 PCI Boards by Several User Programs

Usage of RTC5 PCI Boards by several user programs is coordinated by the (RTC5 DLL-internal) RTC5 board management. By `init_rtc5_dll`, it is initialized.

By `acquire_rtc`, the `init_rtc5_dll` command automatically assigns access rights to the found boards, as long as access rights are not already assigned to another user program (any number of RTC5 Boards or applications can be used simultaneously, but no board can be simultaneously used by multiple applications). Access rights (even if temporary) to existing boards are assigned on an exclusive basis by the DLL. Multiple threads of one user program can use the same board, but can not send commands to it at the same time (the RTC5 DLL automatically serializes the command calls).

Without access rights, a board can only be accessed by a user program through purely RTC5 DLL-internal functions that do not require access rights – for example, `get_error`, `get_last_error` or `select_rtc`.

If a user program has gained access rights for a board, then this board can be accessed by another user program only after the original user program explicitly releases its access rights by `release_rtc` or `free_rtc5_dll`. Acquisition of a released board can then be achieved by `acquire_rtc` (or `init_rtc5_dll` or `select_rtc`).

When a board is acquired by `acquire_rtc` (or `init_rtc5_dll` or `select_rtc`), a version compatibility check is performed on the RTC5 DLL and the program files `RTC5OUT.out` and `RTC5RBF.rbf`. If no program files have been loaded, then a version check cannot be explicitly performed but the check is still regarded as successful and does thus not hinder the board's acquisition. If program files have been loaded and the version check detects an error, then access is denied (`get_last_error` return code

`RTC5_ACCESS_DENIED` | `RTC5_VERSION_MISMATCH`).

### 6.7.1 Board Acquisition by a User Program

`init_rtc5_dll`, `acquire_rtc`, `free_rtc5_dll`, `release_rtc` and `select_rtc` affect the access rights of the installed RTC5 PCI Boards. They do not initialize RTC5 PCI Boards.

A user program acquiring a board by `acquire_rtc` (or `init_rtc5_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 RTC5 Board can be further used without changes, because in this situation all RTC5 DLL configuration data remain unaltered. However, the above does not apply if the acquired board has been released by `free_rtc5_dll` and reacquired with `init_rtc5_dll`.

When an RTC5 Board is acquired by another user program, some of the previous user program's key data managed only in the RTC5 DLL is *not* (automatically) inherited. 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 reset by `load_program_file`, then all settings are newly defined anyway and such missing information would not be relevant.

On the other hand, if the acquiring user program intends to further use the RTC5 Board's inherited state, then it must explicitly query the missing **RTC5 DLL** information, receive it from the previous user program and explicitly re-establish it so that the board's **RTC5 DLL** remains consistent. In this regard, observe the following:

- For a correct behavior of the input pointer at the list borders, the memory configuration currently set in the **RTC5 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 **RTC5 DLL** correspondingly.
- The management table of protected functions (indexed subroutines, character sets and text strings) is saved on the board and therefore inherited through board acquisition. 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 page 104). An alternative restoration method is not possible. The intermixed loading of protected functions by differing applications should therefore be avoided.
- If, during loading a protected function, **release\_rtc** is called before a subsequent **list\_return** command is transferred, then the function is not stored on the RTC5 Board.
- The input pointer is generally not inherited (the input pointer location currently saved in the **RTC5 DLL** for the acquiring user program is used, maybe corrected by **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 (with reference to **LOAD list status** and **READY list status**) 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 DLL. Though settings used by the previous user program can not generally be queried, new settings can of course be established 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

So that RTC5 errors can be caught at program runtime by suitable programming, the RTC5 performs general error handling. In addition, some commands allow for specific error handling.

General errors occur, for example, if a user program has no access rights for the board ([RTC5\\_ACCESS\\_DENIED](#)), or if the board fails to respond to a control command ([RTC5\\_TIMEOUT](#)), or if PCI communication problems occur during loading ([RTC5\\_SEND\\_ERROR](#)).

Examples of specific errors are: calling a command with an unallowed (uncorrectable) parameter ([RTC5\\_PARAM\\_ERROR](#), for example, see [get\\_value](#) or [write\\_da\\_x](#)), or rejected loading of a list command ([RTC5\\_REJECTED](#), for example, due to an illegal input pointer), or transmitting a control command at an improper time ([RTC5\\_BUSY](#), for example, [goto\\_xy](#), when a list is still being processed).

In such cases, the control commands are not executed; list commands are typically replaced with [list\\_nop](#) commands (for example, for [RTC5\\_PARAM\\_ERROR](#) or [RTC5\\_IGNORED](#), see [set\\_end\\_of\\_list](#) as an example).

The bits assigned to these errors are set or accumulated in the [RTC5 DLL](#) with each command in the board-specific error variables:

- `LastError`
  - Error code
  - Is automatically reset at the beginning of every command
  - Therefore, is a listing of occurred errors from the most recently executed command
  - Can be queried by [get\\_last\\_error](#)<sup>(1)</sup>
- `AccError`
  - Cumulative error code
  - Is reset when initializing the [RTC5 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 and 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 (neither is any other board data exchanged).

Error handling only takes place during initialization and when loading onto the RTC5, but not during execution of a list program.

An example program of how to incorporate board-specific error variables is provided in the description of [get\\_error](#).

Some control commands (for example, [init\\_RTC5\\_dll](#), [load\\_correction\\_file](#) or [load\\_program\\_file](#)) additionally return a special error code 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 RTC5. Commands that *do not* establish communication with the RTC5 (for example, [rtc5\\_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 RTC5 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 `RTC5_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 RTC5 PCI 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:

- `RTC5DAT.dat` is tested by a checksum for file correctness and PCI-transfer correctness.
- `RTC5RBF.rbf` is only checked by a bitwise transfer handshake. No other checking is possible.
- Each loaded section of `RTC5OUT.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 `RTC5_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  $\mu$ s clock cycle have occurred. See also Section "Clock Overruns", page 171.



### 6.8.3 Example Code (C)

The following example C source code shows how to catch an error during initialization. It ensures the program terminates with an error message, if:

- An error occurs during initialization with `init_rtc5_dll` (for example, if no RTC5 PCI Board has been detected)
- The desired RTC5 PCI Board (here: the board with serial number 12345) is not detected
- Access is denied to the desired RTC5 PCI 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 \(C\)", page 89](#).

```
UINT ErrorCode;

ErrorCode = init_rtc5_dll();
if ( ErrorCode )
{
    // Reading the number of RTC5 Boards detected during initialization with init_rtc5_dll
    const UINT RTC5CountCards = rtc5_count_cards();
    if ( RTC5CountCards )
    {
        // Detailed error analysis for all detected boards
        UINT AccError( 0 );
        for ( UINT i = 1; i <= RTC5CountCards; i++ )
        {
            // Errors which occurred during execution of init_rtc5_dll
            const UINT Error = n_get_last_error( i );
            if ( Error != 0 )
            {
                AccError |= Error;
                const UINT SerialNumber = n_get_serial_number ( i );
                printf( "RTC5 Board number %d (serial number %d): Error %d detected\n",
                        i, SerialNumber, Error );
                n_reset_error( i, Error );
            }
        }
        if ( AccError )
        {
            free_rtc5_dll();
            return;
        }
    }
}
```



```
else
{
    printf( "Initializing the DLL: Error %d detected\n", ErrorCode );
    free_rtc5_dll();
    return;
}
}
else
{
    // Reading the internal board number for the desired RTC5 Board
    const UINT SerialNumberOfDesiredBoard ( 12345 );
    const UINT RTC5CountCards = rtc5_count_cards();
    UINT InternalNumberOfDesiredBoard ( 0 );
    for ( UINT i = 1; i <= RTC5CountCards; i++ )
    {
        if ( n_get_serial_number( i ) == SerialNumberOfDesiredBoard )
        {
            InternalNumberOfDesiredBoard = i;
        }
    }
    if ( InternalNumberOfDesiredBoard == 0 )
    {
        printf( "RTC5 Board with serial number %d not detected.\n", SerialNumberOfDesiredBoard );
        free_rtc5_dll();
        return;
    }
    // Selecting the desired RTC5 Board as the active RTC5 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 )    // RTC5_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 RTC5 Board with serial number %d\n", SerialNumberOfDesiredBoard );
            free_rtc5_dll();
            return;
        }
        if ( ErrorCode )
        {
            printf( "No access to RTC5 Board with serial number %d\n", SerialNumberOfDesiredBoard );
            free_rtc5_dll();
            return;
        }
        else
        {
            // if n_load_program_file has been 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 263](#)
- 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 unsigned 32-bit 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 and Laser Control

### 7.1 Marking Dots, Lines and Arcs

#### 7.1.1 Marking with Vector Commands and "Arc" Commands

As explained in [Chapter 6.1 "RTC5 Software Concept Basics", page 83](#), positioning of the scan system axes (and thus of the laser beam) under RTC5 PCI Board control is achieved by calling:

- [Jump Commands](#)
- [Mark Commands](#)
- [Arc Commands](#)
- [Ellipse Commands](#)

Each of these commands describes one vector or arc.<sup>(1)</sup> By using [micro\\_vector\[\\*\] Commands](#), arbitrarily shaped trajectories<sup>(2)</sup> can be implemented.

Even numeric and alphabetic characters ultimately consist of the constituent lines, dots and arcs that define them, see [Chapter 7.5 "Marking Dates, Times and Serial Numbers", page 196](#).

[Vector commands](#) ([Jump Commands](#), [Mark Commands](#)) 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" commands](#) ([Arc Commands](#) 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 output position after a RTC5 PCI Board [Hardware reset](#) 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 RTC5 PCI Board into [Microsteps](#), see [Chapter 7.1.2 "Microstepping", page 128](#)<sup>(4)</sup>.

[Jump commands](#) serve to move the scan system axes to a new position while the laser is switched *off*.

In contrast, [Mark commands](#) initiate a marking motion while the laser is switched *on* (see also the following description).<sup>(5)</sup>

To mark a point, the [Signals for "Laser Active" Operation](#) must be switched on for the desired time period after a [Jump command](#) or [\[\\*\]mark\[\\*\] Command](#), see [Chapter 7.1.3 "Marking Single Dots", page 129](#).

For line and arc marking, the RTC5 PCI Board automatically switches the [Signals for "Laser Active" Operation](#) on at the beginning of a [Mark command](#) and later switches it back off (for example, at the beginning of a subsequent [Jump command](#)).

The synchronization of scan head control and laser control can be adjusted by the user to the respective application by setting delays, see [Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control", page 133](#).

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 \(C\)", page 130](#).

(1) Here, wider line widths can be specified by [set\\_wobble\\_mode](#).

(2) See [Glossary entry on page 25](#).

(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 [iDRIVE](#) scan systems (see [Glossary entry on page 23](#)) which are equipped with an appropriate tuning can execute jumps also in [Jump Mode](#), see [Chapter 8.1.5 "Jump Mode", page 202](#).

(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 **Signals for "Laser Active" Operation** 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 133](#) and [Chapter 7.2.2 "Scanner Delays", page 135](#). 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 RTC5 PCIe Board automatically turns on the **Signals for "Laser Active" Operation** at the beginning of a `[*]mark[*]` Command, see also [Section "Polylines", page 125](#).

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>. The parameters to be specified are 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 **Signals for "Laser Active" Operation** are automatically switched on at the beginning of the execution of an arc command, see also [Section "Polylines", page 125](#).

## 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 140](#).

(1) For using abs and rel commands, see [Section "AbsCalls", page 103](#). Additionally are available: timed vector commands, see [Chapter 8.9 "Timed Commands", page 250](#), para vector commands, see [Section "Vector-Defined Laser Control", page 194](#) and 3D vector commands, see [Chapter "3D Commands", page 223](#).

(2) For using abs and rel commands, see [Section "AbsCalls", page 103](#). Additionally, timed arc commands are available, see [Chapter 8.9 "Timed Commands", page 250](#).

## Ellipse Commands

The RTC5 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` as well as `mark_ellipse_abs` or `mark_ellipse_rel`<sup>(1)</sup>.

By `set_ellipse` the arc shape is specified, see Figure 37:

- Lengths  $a$  and  $b$  of the ellipse half-axes
- The beginning phase angle  $\text{Phi0}$  (and thereby the arc starting point 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)

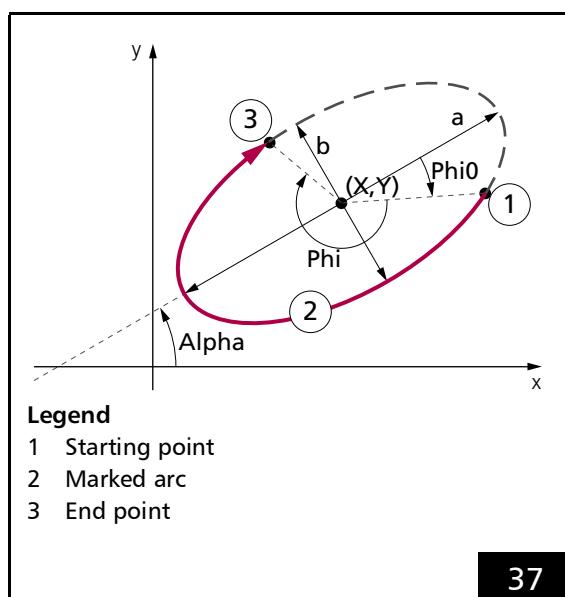
By `mark_ellipse_abs` or `mark_ellipse_rel`, position and orientation of the to-be-executed arc is specified, see Figure 37:

- The coordinates  $(X, Y)$  of the ellipse midpoint

The angle  $\text{Alpha}$  between the ellipse 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).  $\text{Phi0}$ ,  $\text{Phi}$  and  $\text{Alpha}$  are always relative to axis  $a$ .
- $\text{Phi0}$  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 Commands** and **Arc Commands**, the laser focus moves with the specified mark speed along the specified arc when the **Ellipse command** is executed. The **Signals for "Laser Active" Operation** are automatically switched on at the beginning of an **"Arc" command**, see also **Section "Polylines"**, page 125.
- `set_ellipse` is a short list command, see also **Section "Normal, Short, Variable and Multiple List Commands"**, page 278. Therefore, it can be called between a **Mark command** and an **Ellipse command** without thereby interrupting the **Polyline** (the laser remains on).
- **Ellipse Commands** always begin marking at the starting point determined by the above-mentioned parameters (in contrast to **Mark Commands** and **Arc Commands** which automatically begin marking at the current output position). If the starting point and current position do not match, then a **Hard jump** to the starting point is executed at the beginning of marking (without a **Jump Delay** becoming effective).



Marking ellipse-shaped arcs.

(1) For using abs and rel commands, see **Section "AbsCalls"**, page 103.

- 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 direction/y direction) has been 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  $10 \mu\text{s}$  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  $10 \mu\text{s}$  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%). However, the outputted point always lies exactly on the ellipse. 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 themselves must ensure that the parameter values used are consistent with the required precision.

#### [\*]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 194](#).

[\*]**para\_mark**[\*] commands generally do not take **Sky Writing** into account.

### 7.1.2 Microstepping

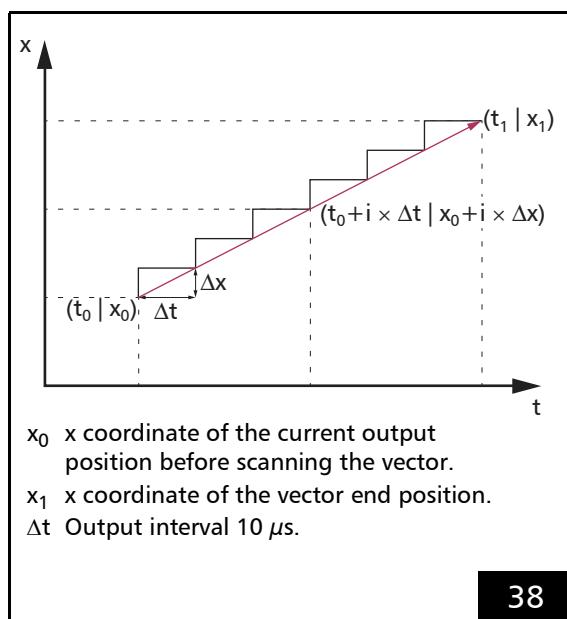
The RTC5 PCI Board splits up each

- **Jump command**,
- **[\*]mark[\*] Command** and
- **"Arc" command**

into so-called

- **Microsteps**  
(not: **Microvectors**, see **Chapter 8.8 "micro\_vector[\*] Commands"**, page 249).

The split-up of the x component of a vector into **Microsteps** is shown in **Figure 38**.



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 following applies:

$$\text{Length } \Delta s \text{ of a Microstep} = v \times \Delta t \text{ }^{(1)}$$

( $v$  = current jump speed or mark speed)

### Notes

- Custom curves can be implemented by using **micro\_vector[\*] commands**, see **Chapter 8.8 "micro\_vector[\*] Commands"**, page 249.
- Only **iDRIVE** scan systems<sup>(2)</sup> which are equipped with an appropriate tuning can execute jumps also in **Jump Mode**, see **Chapter 8.1.5 "Jump Mode"**, page 202.

(1) Alternatively see **Chapter 8.9 "Timed Commands"**, page 250.

(2) See Glossary entry on page 23.



### 7.1.3 Marking Single Dots

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 dot can also be marked by a timed [Mark command](#) of length zero, see [Chapter 8.9 "Timed Commands"](#), page 250.



### 7.1.4 Example Code (C)

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 \(C\)", page 89](#).

```
// 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_rtc5_dll();
    return;
}
select_cor_table( 1, 0 ); // assigning table #1 to the first scan head connector (default)

// Laser control initialization, see Chapter 7.4 "Laser Control", page 173
// Setting the CO2 Mode
set_laser_mode( 0 );

// Setting and enabling the Signals for "Laser Active" Operation
set_laser_control( 0x18 ); // All laser signals LOW active (Bit #3 and #4)
                           // This command 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  $\mu$ s.
                  // The RTC5 multiplies the specified values by 8 to convert to integer-multiple of 1/64  $\mu$ s.
                  // Half of the standby output period = 100  $\mu$ s.
                  // Pulse length of the standby pulses = 1  $\mu$ s.

// Timing, delay and speed preset
// Setting the Scanner Delays (see Chapter 7.2.2 "Scanner Delays", page 135):
set_scanner_delays( 25, 10, 5 );
                     // Jump Delay = 250  $\mu$ s (specified in [10  $\mu$ s])
                     // Mark Delay = 100  $\mu$ s (specified in [10  $\mu$ s])
                     // Polygon Delay = 50  $\mu$ s (specified in [10  $\mu$ s])
```



```
// Setting the jump speed and mark speed:  
set_jump_speed( 1000.0 );  
set_mark_speed( 250.0 );  
    // In RTC4 Compatibility Mode the RTC5 multiplies the speed values by 16.  
    // Jump speed = 1000.0 bits/ms.  
    // Mark 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  $\mu$ s.  
    // The RTC5 multiplies the specified values by 8 to convert in integer-multiple of 1/64  $\mu$ s.  
    // Laser HalfPeriod = 100  $\mu$ s.  
    // Laser pulse length = 50  $\mu$ s.  
  
// Setting the Laser Delays, Chapter 7.2.1 "Laser Delays", page 133.  
set_laser_delays( 100, 100 );  
    // In RTC4 Compatibility Mode the Laser Delays are specified in units of 1  $\mu$ s.  
    // The RTC5 multiplies the specified values by 2 to convert in integer-multiple of 0.5  $\mu$ s.  
    // LaserOn Delay = 100  $\mu$ s.  
    // LaserOff Delay = 100  $\mu$ s.  
  
// Defining the end of the list and the end of command transfer to the RTC5 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 RTC5 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.  
laser_on_list( 5 ); // Turning on the laser control signals for 50  $\mu$ s.
```



```
// 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 4 [*]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:  
// 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 RTC5 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 Delays](#) 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 143](#)

The [LaserOn Delay](#) and the [LaserOff Delay](#) are set by the undelayed short list command [set\\_laser\\_delays](#). Their unit is  $0.5 \mu\text{s}$  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 143](#).

(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 187.

## 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 41](#). 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 a scanner delay. This scanner delay is automatically extended, if a preceding **LaserOff Delay** has not yet been expired, see [Section "Automatic Delay Adjustments", page 143](#).

## 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 motion leads to a time difference between the respective set position and the actual position of the mirrors, see [Figure 41](#).

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 the end of each marking by the first "normal" list command (no "short" list command), see also [Section "Notes", page 137](#). This can be used to compensate for the **Tracking error** of the scan head.

The following applies with 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 143](#).

### 7.2.2 Scanner Delays

There are three different types of **Scanner Delays**:

- **Jump Delays** (optionally: variable)
- **Mark Delays**
- **Polygon Delays** (optionally: variable)

After each **Jump command** and **Mark command**, the RTC5 PCI 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.

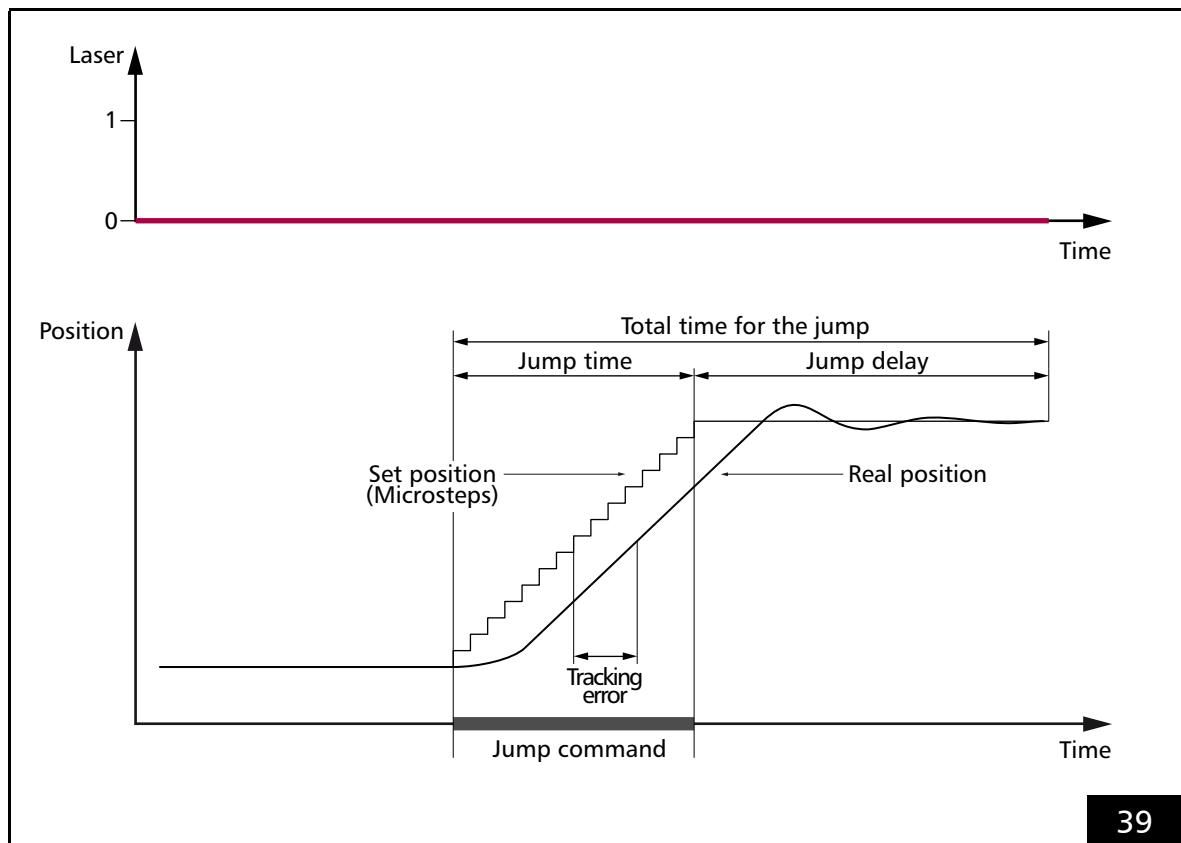
#### Jump Delay

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 39.

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 Delays

With short jumps, the scan head often does not reach the full jump speed. Then *shorter* **Jump Delays** are sufficient for settling of the mirrors:

- “Variable Jump Delays”

For this, there is:

- “Variable Jump Delays” mode

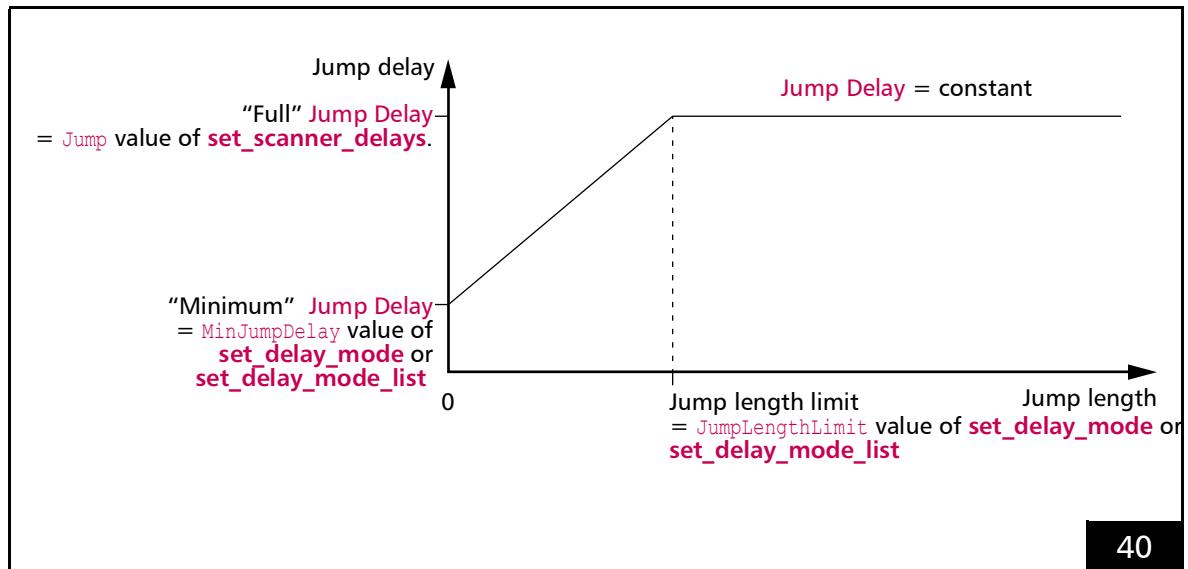
“Variable Jump Delays” mode is switched on by `JumpLengthLimit > 0` (at `set_delay_mode` and `set_delay_mode_list`).

Then the RTC5 PCI 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 **Jump length limit**, see Figure 40.

With jump lengths larger than `JumpLengthLimit` a “full” **Jump Delay** is always inserted.

## Notes

- With the “Variable Jump Delays” mode, total marking time is reduced, especially when there are many short jumps.
- The “Variable Jump Delays” mode is switched off by `JumpLengthLimit = 0` (at `set_delay_mode` and `set_delay_mode_list`).
- After jump vectors of length 0, the duration of “Variable Jump Delays” is 0.
- The “minimum” **Jump Delay** should not be larger than the “normal” **Jump Delay**. Otherwise, “Variable Jump Delays” can become very large.



“Variable Jump Delays” mode: **Jump Delay** value depending on the jump length.

## Mark Delay

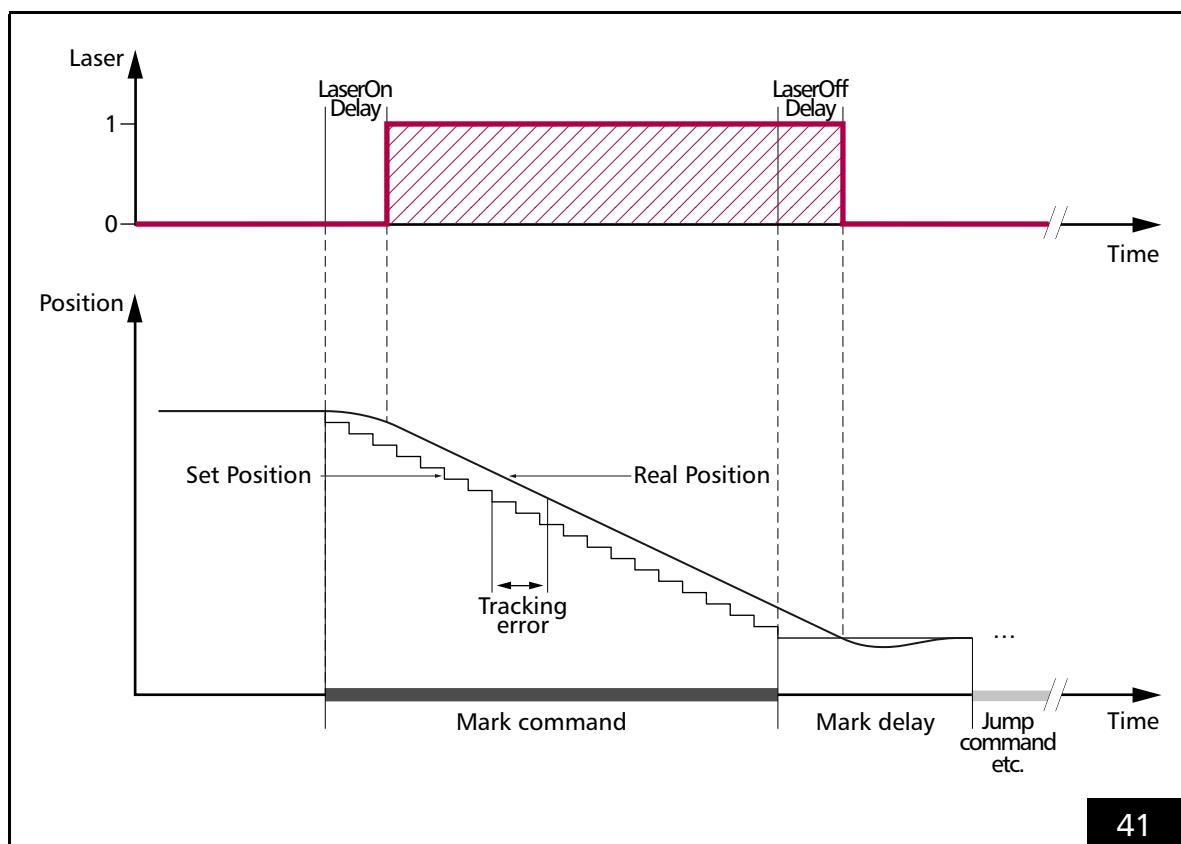
The **Mark Delay** is specified by `set_scanner_delays` with the **Mark** parameter.

A typical course for a single **Mark command** with the and the belonging **Laser Delays** is shown in Figure 41.

The duration of the **Mark Delay** should suit the dynamic properties of the scan head (**Tracking error**, tuning) and the mark speed set.

## 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 Delays"**, page 139.
- Short control commands do not lead to insertion of a **Mark Delay** or **Polygon Delay** and not to a laser switch off.
- Note that a **Mark command** of length 0 corresponds to a **list\_continue**, if it is not timed, see **Chapter 8.9 "Timed Commands"**, page 250.



Scan head control and laser control timing during a **Mark command** or "Arc" command with a **Mark Delay**. The laser is on in the hatched period.

## Polygon Delay

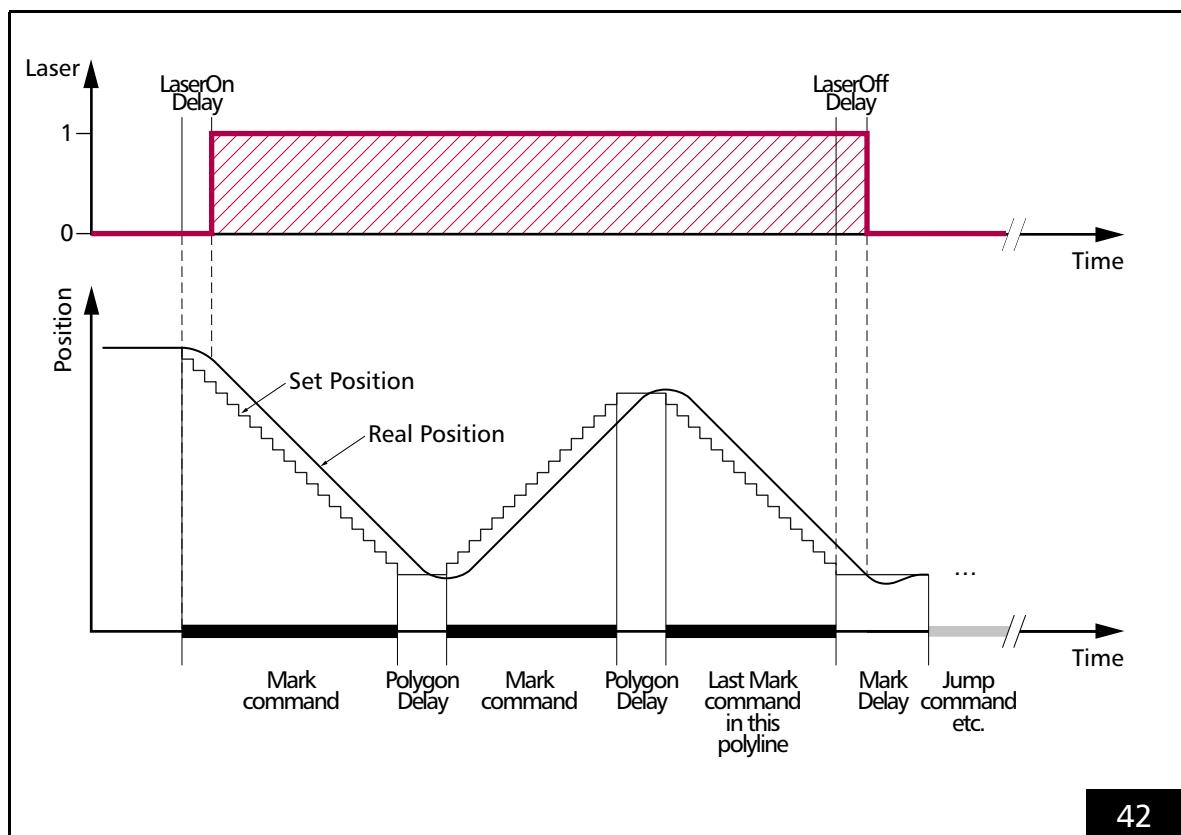
The **Polygon Delay** is specified by `set_scanner_delays` with the **Polygon** parameter.

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 42](#).

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 Delays", page 139](#).



Scan head control and laser control timing during a **Polyline** with a constant **Polygon Delay**.

### Variable Polygon Delays

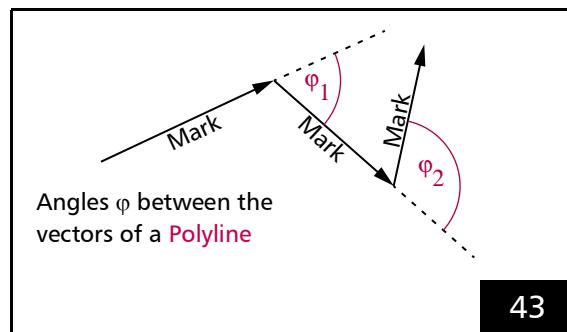
By setting the `VarPoly` value  $> 0$  (at `set_delay_mode` or `set_delay_mode_list`) it is enabled:

- “Variable Polygon Delays” mode

Then, the RTC5 PCI Board calculates the “Variable Polygon Delay”  $v\_delay(\phi)$  for every `Polyline` corner according to:

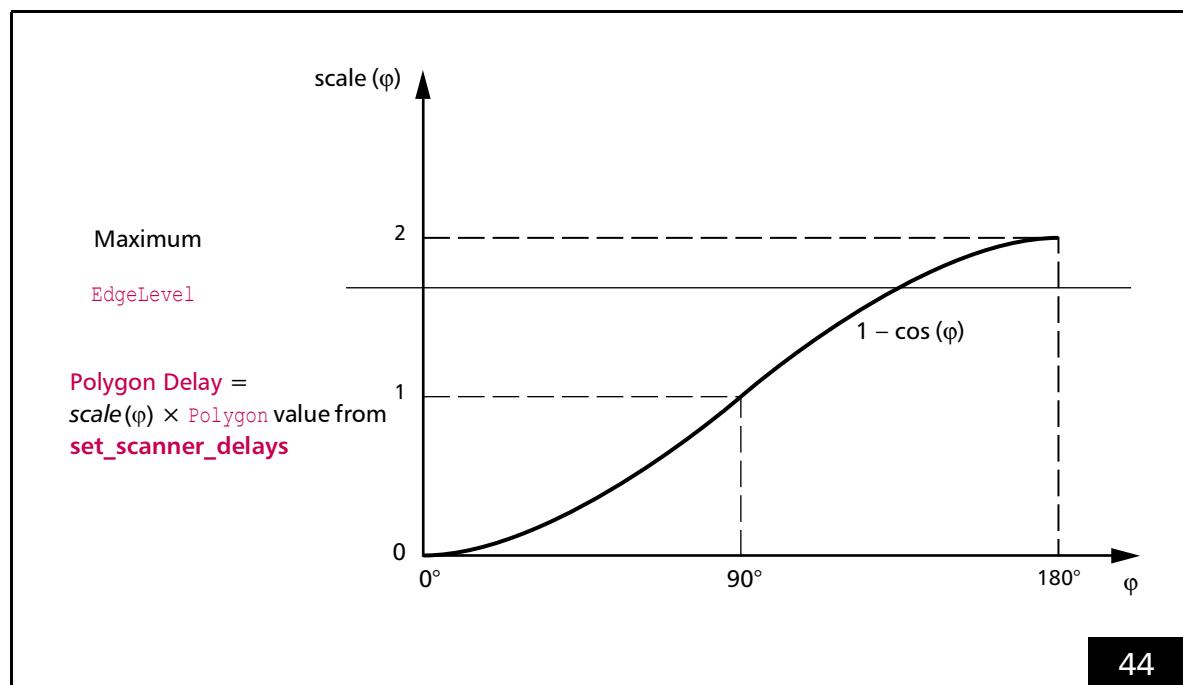
$$v\_delay(\phi) = scale(\phi) \times polygon\_delay$$

For the definition of the angle  $\phi$  using the example of mark vectors, see [Figure 43](#).



[Variable Polygon Delays](#). Definition of the angle  $\phi$ .

For circular arcs and ellipses, analogously the tangents at the connection point are relevant. `scale(\phi)` is a scaling function for the `Polygon` value from `set_scanner_delays` (constraint  $0 \leq scale(\phi) \leq 2$ ), see [Figure 44](#). The default scaling function after `load_program_file` is  $scale(\phi) = 1 - \cos(\phi)$ . It can be replaced by a user-defined scaling function, see [Section “User-defined “Variable Polygon Delays””, page 141](#).



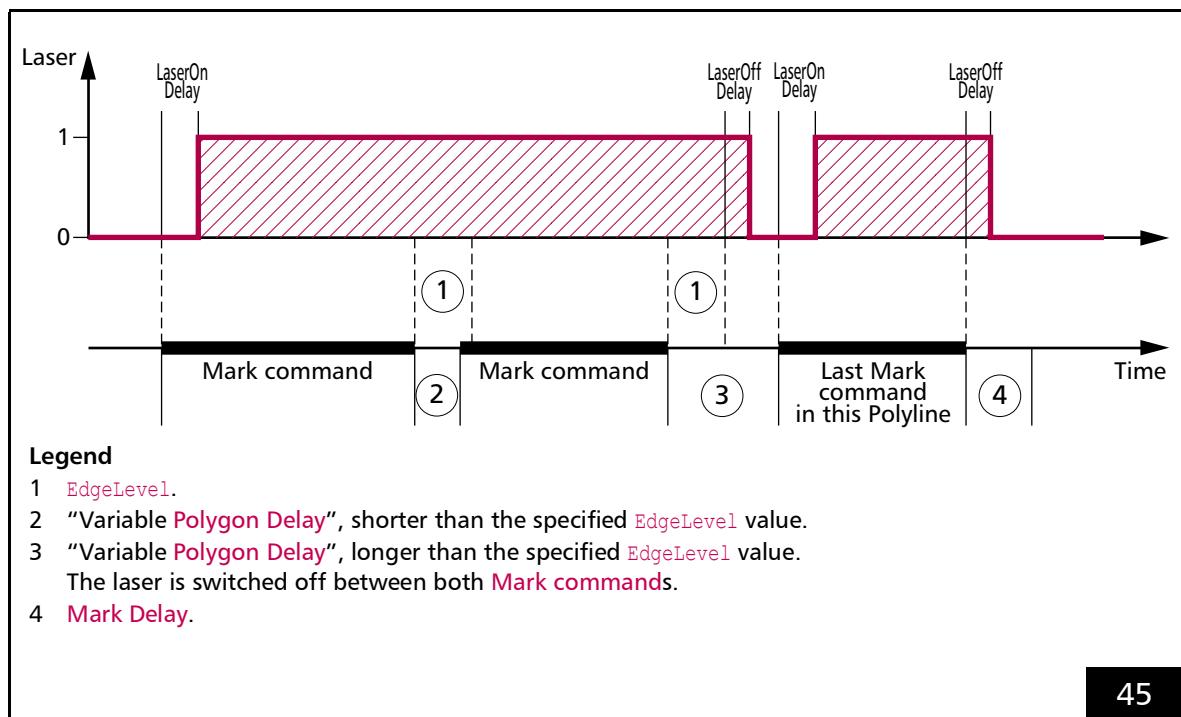
[Variable Polygon Delays](#). Variation of the `Polygon Delay` (default scaling function after `load_program_file`).

#### EdgeLevel

The “Variable Polygon Delay” becomes quite long with angles  $\varphi$  close to  $180^\circ$ , see Figure 44. 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 RTC5 PCI 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 45.

The next **Mark command** starts as usual with the current “Variable Polygon Delay”.



45

Laser control timing during a **Polyline** with “Variable **Polygon Delay**”.  
An `EdgeLevel` value has been defined with `set_delay_mode`.

### User-defined "Variable Polygon Delays"

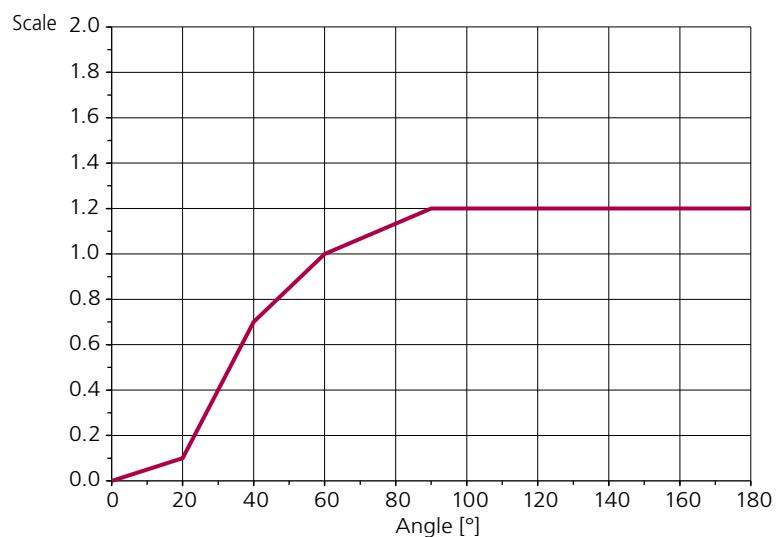
- For the  $scale(\varphi)$  scaling function, **load\_varpolydelay** loads a table from an ASCII text file. See also [Section "Variable Polygon Delays", page 139](#).
- The ASCII text file can contain one or more tables<sup>(1)</sup>.
- Each table can contain up to 50 data points ( $\varphi \mid scale(\varphi)$ ).
- The  $scale(\varphi)$  function is linearly interpolated from the data points.
- As an example, [Figure 46](#) shows a table with 4 data points and the corresponding scaling function.

(1) Even of another type, see [table 1, page 141](#).

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 141</a>
[PositionCtrlTable<No>]	Scaling function , see <a href="#">page 189</a>
[AutoLaserCtrlTable<No>]	Nonlinearity curve, see <a href="#">page 192</a>
[JumpTable<No>]	Jump Delay values, see <a href="#">page 205</a>
[StretchTable<No>]	2D stretch correction table, see <a href="#">page 225</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
```



46

Example: Table for [User-defined "Variable Polygon Delays"](#) with 4 data points (left) and corresponding scaling function  $scale(\varphi)$  (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  $0.0 \leq scale(\varphi) \leq 2.0$ .
- 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  
 $scale(0^\circ) = 0$  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  $scale(180^\circ)$  is  
added.

After initialization by `load_program_file`, the  
RTC5 PCI Board uses the internal (default) table for  
the "Variable Polygon Delay" (1-cos( $\varphi$ ), see  
Figure 44). Alternatively, this can also be achieved  
with `Name = NULL` in `load_varpolydelay`.

### 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
- the `ControlPreview` time of the used **Preprocessing System**

Non-optimized delays may lead to marking results of insufficient quality and excessive scan times, see also **Section "Potential Errors when Optimizing the Delays", page 146**.

#### Notes

- The **Laser Delays** are specified in units of  $0.5 \mu\text{s}$ .
- In contrast, the **Scanner Delays** (**Jump Delay**, **Mark Delay** and **Polygon Delay**) are specified in units of  $10 \mu\text{s}$ .

#### Recommended Optimization Sequence

##### (1) Laser Delays.

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).

##### (2) Scanner Delays.

#### Automatic Delay Adjustments

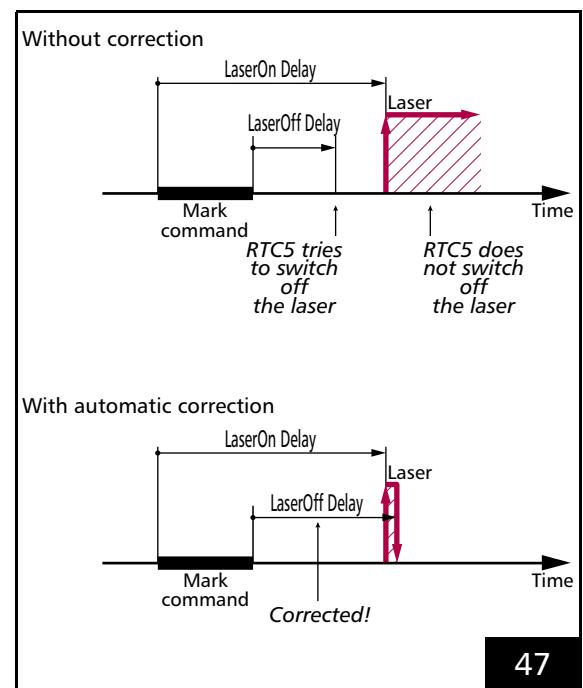
In principle, all delays can be set for any value within the corresponding allowed range. In some situations the laser control could be disturbed, for example, when Laser-On and Laser-Off overlap. Therefore, for each command that switches the laser, the RTC5 PCI Board checks the set **Laser Delays**, 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 RTC5 PCI Board corrects the faulty laser delay so that:

- the Laser-On occurs  $0.5 \mu\text{s}$  after the currently effective **LaserOff Delay**
- or the Laser-Off occurs  $0.5 \mu\text{s}$  after the still running **LaserOn Delay**

In both cases, the chronological next mark is cut off somewhat, see also **Figure 47**.



Automatic adjustment of **LaserOff Delay**.

47

#### Laser-On and Laser-Off 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 RTC5 PCI Board in this case inserts a scanner delay "artificially". This increases the processing time and changes the dynamics of the galvanometer scanner motion.

### 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)

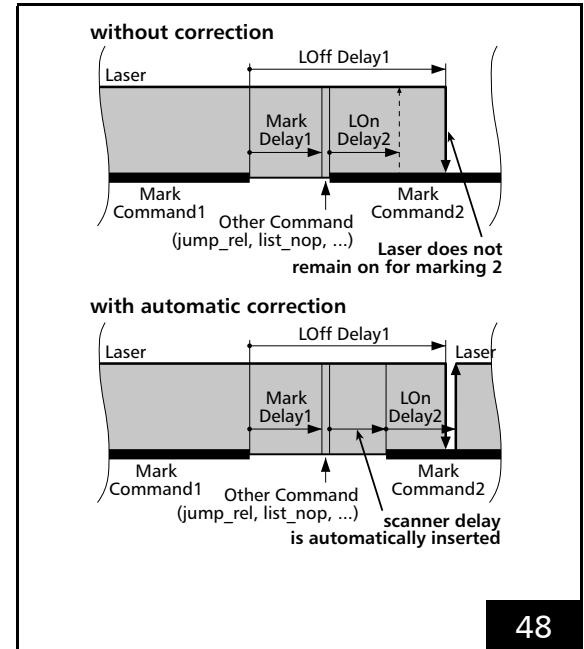
To avoid this overlap, the RTC5 PCI Board in this case extends the short **LaserOff Delay** so long that it only expires after the previous one (and of course only after the intermediate **LaserOn Delay**).

### Several Simultaneously Expiring Laser Delays

This can happen if a laser delay is only effective beyond the length of the next command. The RTC5 PCI Board can only process 1 **LaserOn Delay** and 1 **LaserOff Delay** at the same time. The RTC5 PCI Board ensures this by adjusting the **Scanner Delays**.

### Adjustment of the Scanner Delays

Under some circumstances with the RTC4, a short delay could result in laser control errors if a non-mark command (for example, `jump_rel( 0, 0 )` or `list_nop`) comes between two **Mark Commands**. Here, the laser would be switched off during the second mark command, but unexpectedly not switched on again if the **LaserOff Delay** is longer than the sum of the **Mark Delay** (and any **Jump Delay**) and the **LaserOn Delay** (see Figure 48).



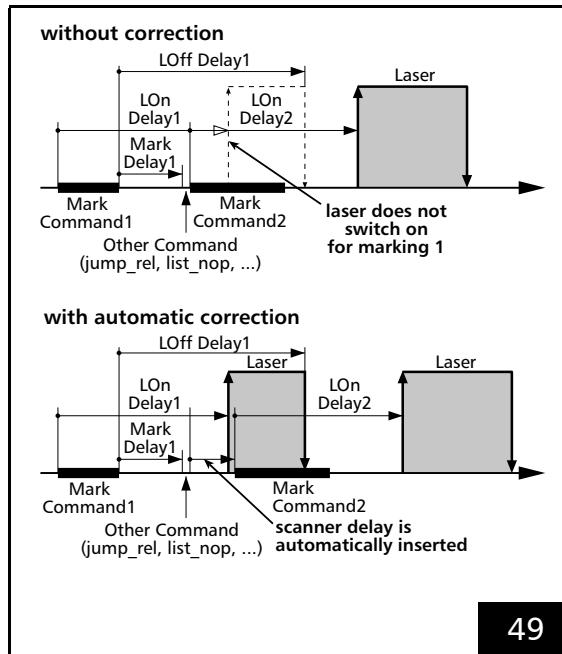
48

Automatic adjustment of the **Scanner Delays** for small **Mark Delays**.

One way of avoiding this problem is to generally select a **Mark Delay** that is larger than the difference between the **LaserOff Delay** and the **LaserOn Delay** (**Mark Delay**>**LaserOff Delay**-**LaserOn Delay**, as with the predecessor boards). But here, a minimum **Mark Delay** size is required that otherwise would not be necessary where **Mark Commands** directly follow another; this thus unnecessarily increases the total marking time. On the other hand, the RTC5 now checks each mark command's **Laser Delays** and automatically extends the scanner delay so that a preceding **LaserOff Delay** first finishes before another Laser-On follows. Thus, marking time is only increased when necessary. The **Scanner Delays** can now be optimized independently of the **Laser Delays**.

The same applies for the **edgelevel** of the "Variable **Polygon Delay**" as well as for a too-short **Polygon Delay** if a polygon chain is continued by a list change but no **auto\_change** or **start\_loop** is used and the second list is started immediately after the first list finishes processing. For predecessor boards in the latter case, the laser might remain switched-off for the rest of the **Polyline**.

For very large **LaserOn Delays** extending across a mark command as well as a subsequent non-mark command and up to the next mark command, appropriate **Scanner Delays** are inserted to prevent overlap of a not-yet-expired **LaserOn Delay** with the start of the second marking command. Otherwise, the currently executing **LaserOn Delay** would have gotten overwritten and the laser would not have even switched on for the first marking command, but instead only for the second marking command and only after expiration of the **LaserOff Delay** (see **Figure 49**).

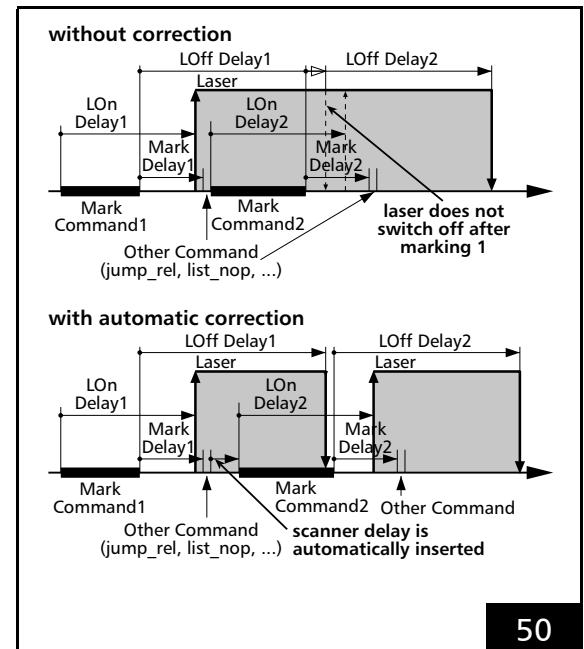


49

Automatic adjustment of the **Scanner Delays** for large **LaserOn Delays**.

Similarly, overlaps would occur for very large **LaserOff Delays** set for the beginning of a non-mark command and extending across the next mark command up to yet another non-mark command. Here, the laser would not switch off between the two mark commands and instead only switch off after the second non-mark command and only after expiration of the **LaserOn Delay** (see **Figure 50**). To prevent this situation, a check is performed at the start of a mark command to determine if the laser would have already switched off prior to the end of the command. If not, then the mark command gets postponed by an appropriate scanner delay.

On the other hand, laser switch-off during a **Polyline** with sharp corners (edgelevel parameter) only occurs if the laser has been already on during this **Polyline** (that is, when all previously initiated **LaserOn Delays** have already expired and no **LaserOff Delays** remains active).



50

Automatic adjustment of the **Scanner Delays** for large **LaserOff Delays**

## Notes

- Thanks to these previously described automatic adjustments, the RTC5 always fulfills the scanner delay conditions  
(**Mark Delay**>**LaserOff Delay**-**LaserOn Delay**,  
**EdgeLevel**>**LaserOff Delay**-**LaserOn Delay**,  
**Polygon Delay**+**LaserOn Delay**>**LaserOff Delay**) at run-time that had to be explicitly observed with the RTC4.
- Automatic adjustment of the **Scanner Delays** might result in a total marking time longer than that expected from just adding-up the predefined vector lengths and delays.

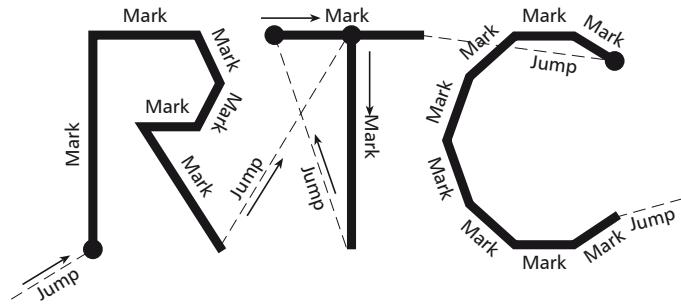
## Potential Errors when Optimizing the Delays

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.

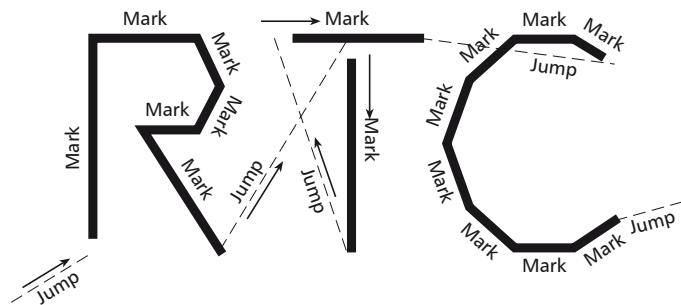
Burn-in effects at the start points of the respective vectors result.



### LaserOn Delay too long

The laser is switched on too late at the beginning of a mark vector.

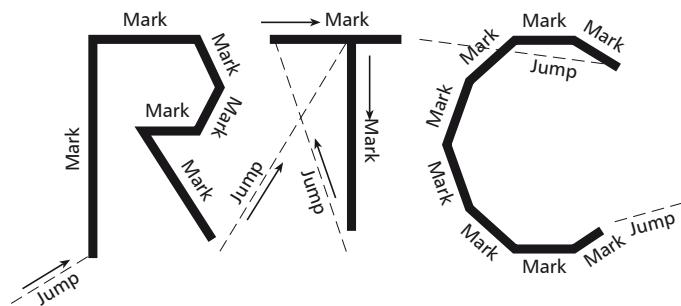
The first part of the vector is not marked.



### LaserOff Delay too short

The laser is switched off after a mark command before the mirrors have reached the vector end position.

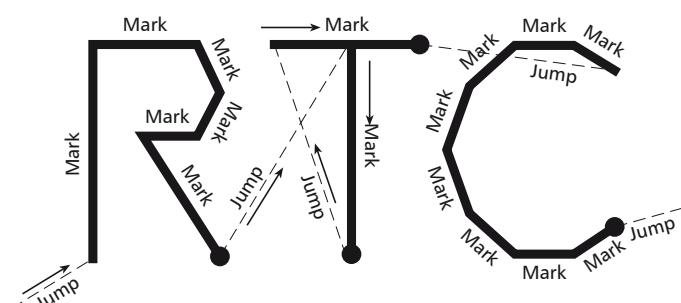
The respective vectors are not marked completely.



### LaserOff Delay too long

The laser is switched off too late after a Mark command. The laser is still on, although the mirrors are already slowed down considerably or have already stopped.

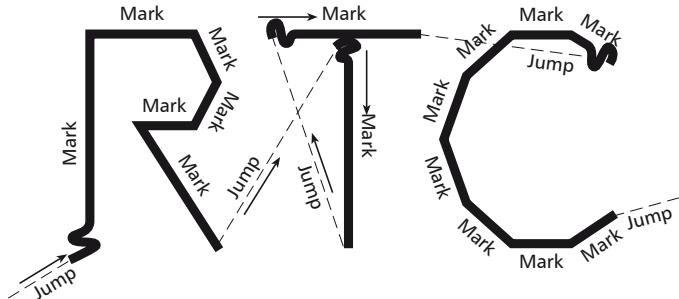
The results are burn-in effects at the end points of the respective vectors.



## Jump Delay too short

The mark vector that follows a **Jump command** has already started although the scanners have not yet settled.

A running-in oscillation or overshoot is visible.



## Jump Delay too long

There are no visible effects if the **Jump Delay** is too long.

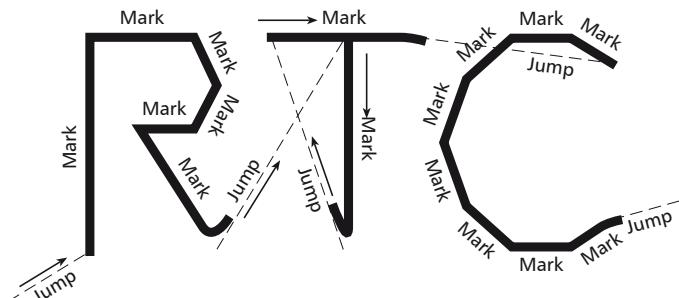
However, the scanning time is extended.



## Mark Delay too short

The traversing of a vector is started before the mirrors have reached the end position of the preceding mark vector.

The end of the mark vector is turned towards the direction of the jump vector.



## Mark Delay too long

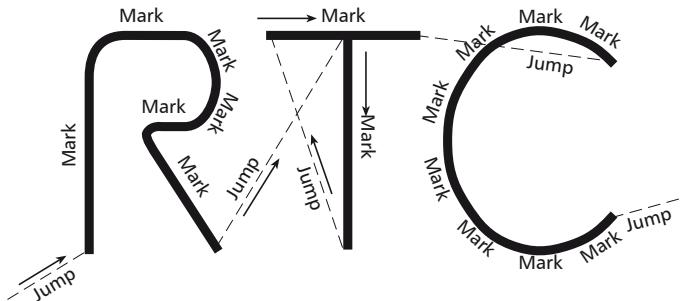
There are no visible effects if the **Mark Delay** is too long, but the scanning time is increased.



### 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.

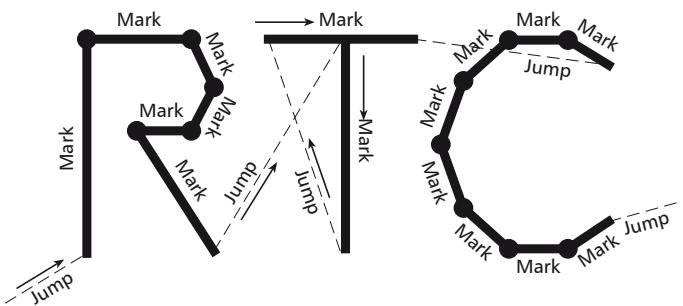
The corners of the **Polyline** are rounded off.



### 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  
**"Variable Polygon Delays"** mode  
, a maximum length  
("EdgeLevel") can be defined,  
see **Section "EdgeLevel"**,  
page 140 for details.

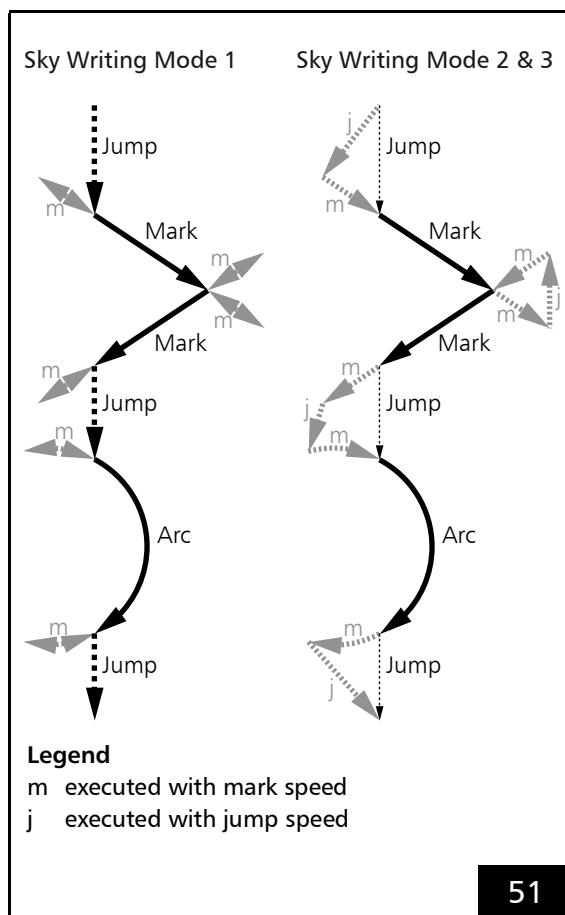
### 7.2.4 Sky Writing

For applications with elevated accuracy requirements the so-called **Sky Writing** can be switched on. 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 RTC5 PCI Board automatically performs non-marking **Sky Writing** scan motions before and after the to-be-executed vectors and arcs, see [Figure 51](#).

The following are always executed *without Sky Writing*:

- **micro\_vector[\*] Commands**
- **[\*]para[\*] Commands**



By the **Sky Writing** command parameters you can specify:

- the duration of these run-in motions
- the duration of these run-out motions
- the synchronization between scan motion and laser control

**Sky Writing** is available in the following modes:

- [Sky Writing Mode 1, page 149](#)
- [Sky Writing Mode 2, page 150](#)
- [Sky Writing Mode 3, page 151](#)

#### Sky Writing Mode 1

In **Sky Writing Mode 1**, the RTC5 PCI 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” motion performed by the galvanometer scanners at mark speed, see [Figure 51](#): the galvanometer 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 gets a short deceleration and retrace motion of the galvanometer scanners (at mark speed) appended in the run-out phase.

**Sky Writing Mode 1** can be switched on/off by:

- [set\\_sky\\_writing](#)
- [set\\_sky\\_writing\\_list](#)
- [set\\_sky\\_writing\\_para](#)
- [set\\_sky\\_writing\\_para\\_list](#)

## Sky Writing Mode 2

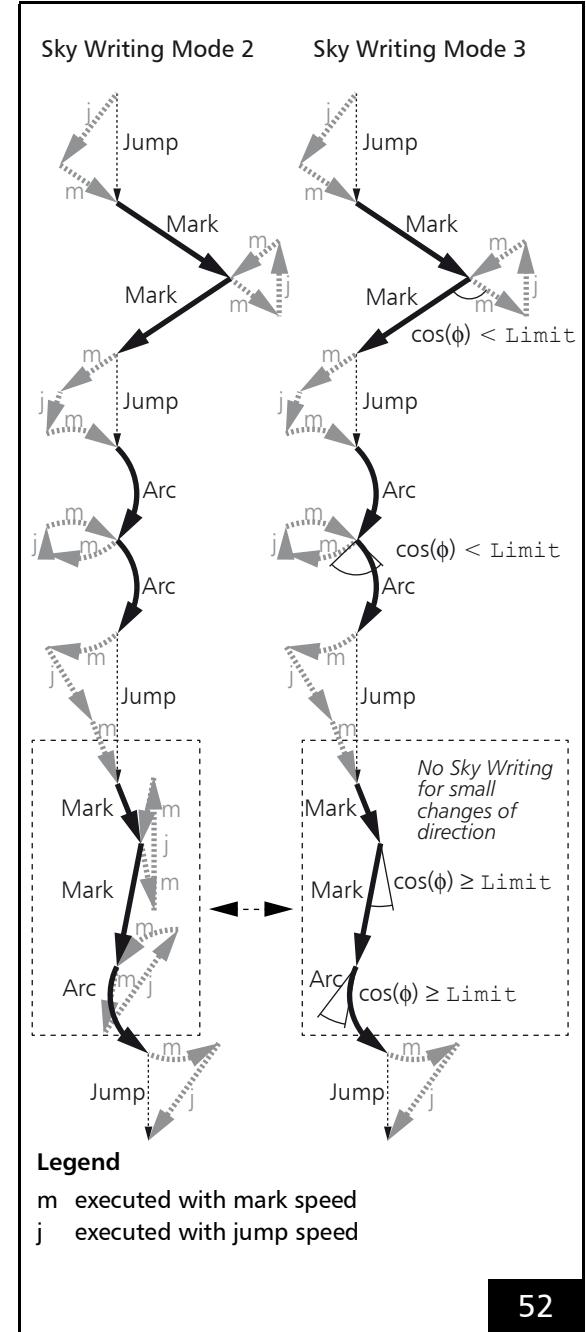
In **Sky Writing Mode 2**, the RTC5 PCI Board calculates *time-shortened* **Sky Writing** motions.

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 Mode 2** marking sequence, however, neither forerun motions (in the run-in phase) nor retrace motions (in the run-out phase) of the galvanometer scanners occur. Instead, the RTC5 PCI 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 51](#).

`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[*]` Commands, arc commands and **Jump commands** (may also contain timed or 3D commands). If such a sequence gets interrupted by some other “non-**Sky Writing Mode 2-capable**” list command (for example, a `[*]para[*]` Command, ellipse command or any short list command), then the RTC5 PCI Board suspends **Sky Writing Mode 2** and complete the preceding command in **Sky Writing Mode 1** (with a retrace motion of the galvanometer scanners). 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 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 ( $\cos(\varphi) < \text{Limit}$ ) between successive **Mark commands** of a **Polyline** a **Sky Writing** motions is executed as in **Sky Writing Mode 2**.

In contrast, smaller angular changes ( $\cos(\varphi) \geq \text{Limit}$ ) result in a (variable) polygonal delay, see **Section "Variable Polygon Delays", page 139**. See also **Figure 52**.

#### 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 Mode 2**-capable" list commands, a **Sky Writing Mode 1** motion 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 53**. The timing diagram for **Sky Writing Mode 2** and **Sky Writing Mode 3** is similar, but here the galvanometer scanners 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  $1 \mu\text{s}$ ) and `LaserOnShift` (whole number in units of  $0,5 \mu\text{s}$ ) define the delay of the **Signals for "Laser Active" Operation** switch-on and switch-off point in times relative to the set starting position and set ending position:
  - Delay of switch-on point in time relative to the set starting position /  $\mu\text{s}$   
= `Timelag` +  $0,5 \mu\text{s} \times \text{LaserOnShift}$
  - Delay of switch-off point in time relative to the set ending position /  $\mu\text{s}$   
= `Timelag`



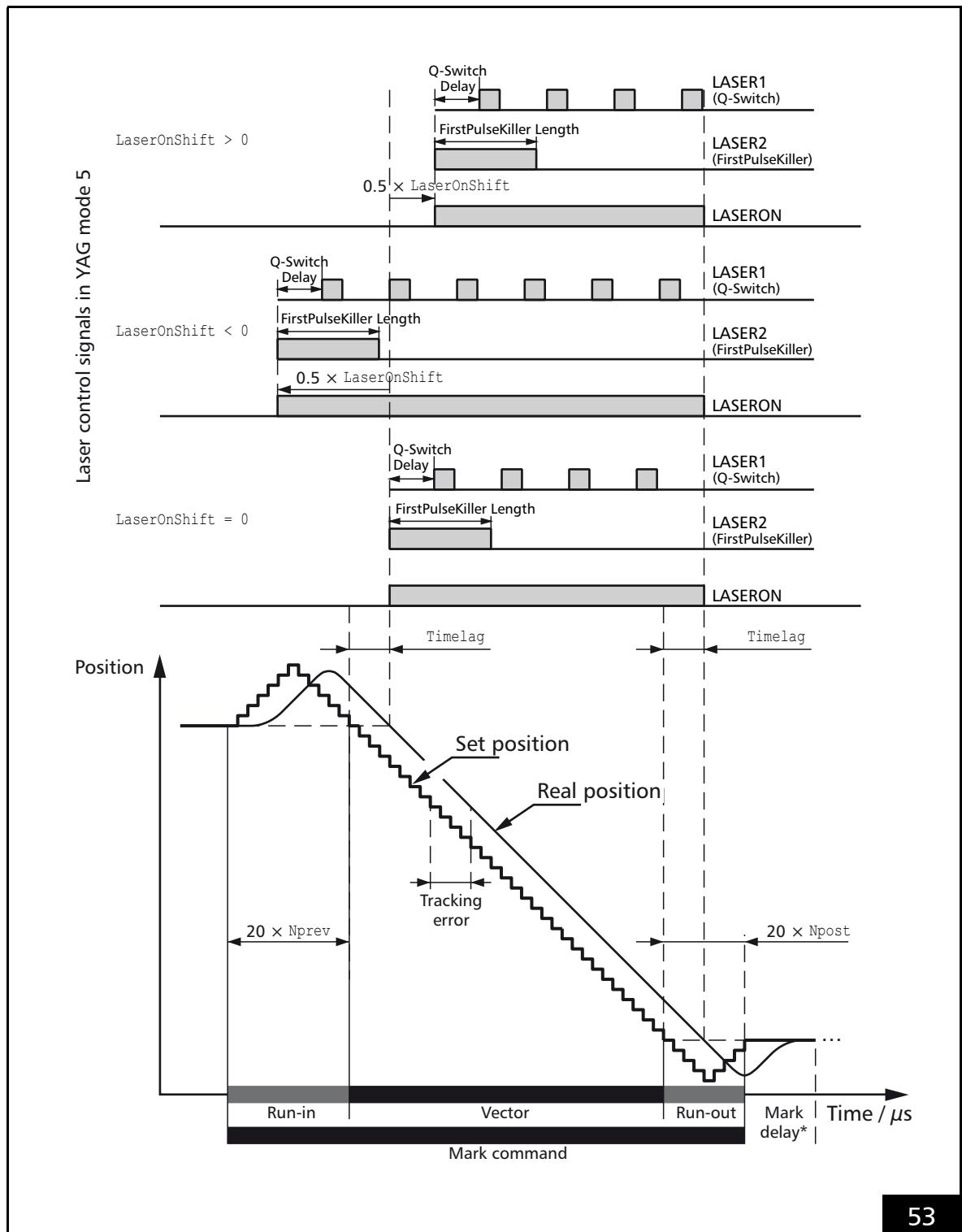
For `LaserOnShift = 0`, the **Signals for "Laser Active" Operation** are switched on (off) with a delay of `Timelag` relative to the set starting position (set ending position).

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 point in time 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 delay or a `FirstPulseKiller` signal in one of the YAG modes.



Scan-head control 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**).

## Notes

- By `set_sky_writing` and `set_sky_writing_mode_list`, you can switch back and forth between **Sky Writing Mode 1** and **Sky Writing Mode 2** or **Sky Writing Mode 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 speed and mark speed 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  $\mu$ s) should exceed the smallest occurring marking time by no more than  $(20 \times Npost - Timelag)$  – and in **Sky Writing Mode 2** and **Sky Writing Mode 3** by no more than  $(10 \times Npost - Timelag)$ . Otherwise, the laser is not switched on for this marking.

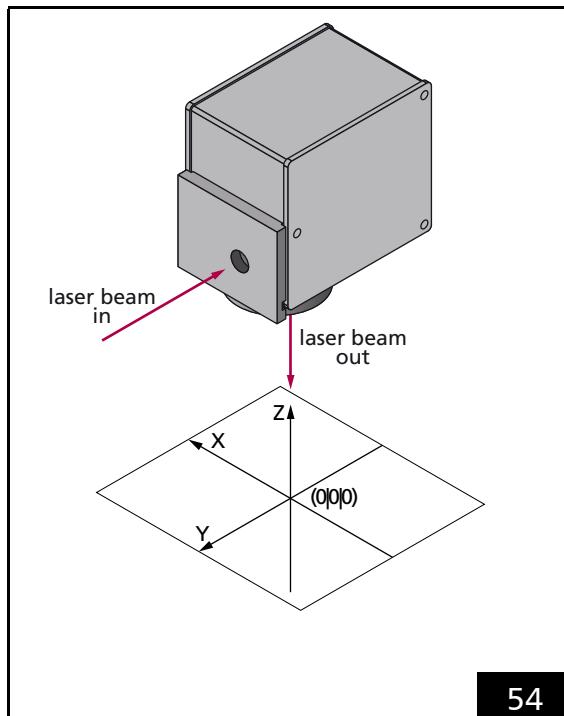
- Pulse lengths and frequencies (in YAG modes additionally the Q-Switch delay and the FirstPulseKiller signal parameters) previously defined for the **Signals for "Laser Active" Operation** are kept unchanged in the **Sky Writing** mode. On the other hand, previously defined **LaserOn Delays**, **LaserOff Delays**, **Mark Delays**, **Polygon Delays** 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 53](#)).
- 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  $(2 \times \text{Nprev} + 2 \times \text{Npost})$ . In **Sky Writing Mode 2** and **Sky Writing Mode 3** execution is extended by a time period (in  $10 \mu\text{s}$ ) of at least  $(\text{Nprev} + \text{Npost})$ .
- 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.  $(2 \times \text{Nprev})$  or even to 0. In **Sky Writing Mode 2** and **Sky Writing Mode 3**, the **Jump Delay** is automatically (internally, dynamically) reduced by up to  $\text{Nprev}$ .
- Time-based **[\*]mark[\*] Commands** and **"Arc" commands** can also be performed in **Sky Writing** mode (see [Chapter 8.9 "Timed Commands", page 250](#)). Here, however, a shorter marking period is coupled with a higher mark speed and therefore with longer distances and acceleration phase in the run-in and run-out phases. Therefore, **Nprev** and **Npost** can be separately adjusted with respect to the specified mark speed.
- **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 194](#)
  - During execution of **micro\_vector[\*] commands**, see [Chapter 8.8 "micro\\_vector\[\\*\] Commands", page 249](#)
- With **Sky Writing Mode 2** or **Sky Writing Mode 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 202](#)
  - Coordinate transformations with **at\_once = 2**, see list item "With **at\_once = 2 ...**", [page 211](#).

## 7.3 Scan Head Control

### 7.3.1 Reference System

The reference system for the **Image field** which is used by the RTC5 PCI Board is shown in [Figure 54](#).

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**.



54

Reference system for the RTC5 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 and y coordinates of a vector must be specified as signed 20-bit values (that is, as numbers between -524,288 and +524,287) whereas z coordinates for a 3D scan system must be specified as signed 16-bit values (that is, as numbers between -32,768 and +32,767).

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:

$$a = \frac{2^{20}}{K}$$

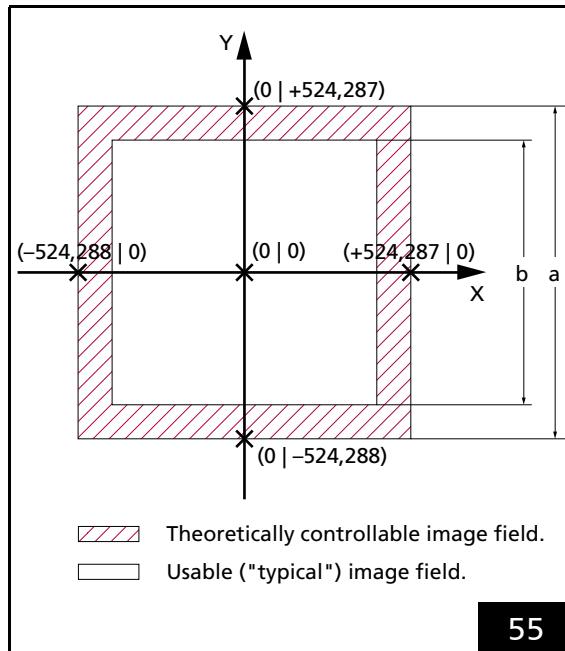
#### Notes

- The calibration factor is the same for the X and y direction.
- With 3D scan systems, the following applies:  
 $K_z = K_{xy} / 16$   
("z calibration factor")

(1) The expression "bits" is here synonymous with "digital control value" (see footnote <sup>(3)</sup> on [page 124](#)).

## 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**.



## Compatibility Modes

### RTC5 Standard Mode

(Default after `load_program_file` and `set_RTC5_mode`)

The **Image field** coordinates for x axis and y axis and all associated parameters are to be specified as signed 20-bit values.

The **Image field** coordinates for the z axis are to be specified as signed 16-bit values. The RTC5 PCI 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 axis, y axis and z axis and all related parameters are to be specified as signed 16-bit values. The RTC5 PCI Board automatically multiplies them by  $16^{(1)}$ .

As xy calibration factor and z calibration factor, a value 16 times smaller must be used. See also [Chapter 2.10.2 “Porting RTC4 Source Code to the RTC5 PCI Board”, page 42](#).

55

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 55](#)).
- 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**:

$$M = K \times b/2$$

(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 24-bit is available.

Therefore, **Vector commands** and **“Arc” commands** can be loaded for objects up to 16 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...#6** 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 209](#)). Otherwise, certain edge areas of the real **Image field** may remain inaccessible during such coordinate transformations.

### Coordinate Transformations in the Virtual Image Field

For **set\_fly\_2d** Processing-on-the-fly applications, you can also define coordinate transformations in the virtual **Image field**. As of version DLL 541, OUT 541 this is also possible with **set\_fly\_x** and/or **set\_fly\_y**.

“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\_angle( HeadNo = 4 )**
- **set\_matrix( HeadNo = 4 )**
- **set\_offset( HeadNo = 4 )**
- **set\_offset\_xyz( 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 are deactivated 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  $\pm 23$  bits. Matrix coefficients must not exceed an absolute value of 1.5. The offset is applied after the matrix operation.

For “virtual coordinate transformations” *cannot* be used:

- **set\_scale**
- all list commands for coordinate transformations

### 7.3.4 3D Image Field

#### Carrying Out 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 "RTC5/6 varioSCAN 40 FLEX Extension" extension board (#0128683)
- 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 RTC5 PCI Board by `load_z_table`.

A, B and C are coefficients of the parabolic function

$$z_{\text{out}} = A + BI + CI^2$$

They determine the relationship between the z output value  $z_{\text{out}}$  and the focus length value  $I$ . For each point  $(x|y|z)$  in the **3D image field**, the focus length value  $I$  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`.

#### Notes

- ABC values from the `*_ReadMe.txt` file of correction file are to be interpreted as 16-bit values.

(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

- Refer also to the "Calibrating a 3-Axis Laser Scan System" Manual.

- (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 RTC5 PCI 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  $(0|0|0)$ .

(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.

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](#).

- (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.
- (4) The focus length value  $l$  can be positive or negative, see notes on [get\\_z\\_distance](#).
- (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.



## Testing 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 56.
- (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 galvanometer 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.

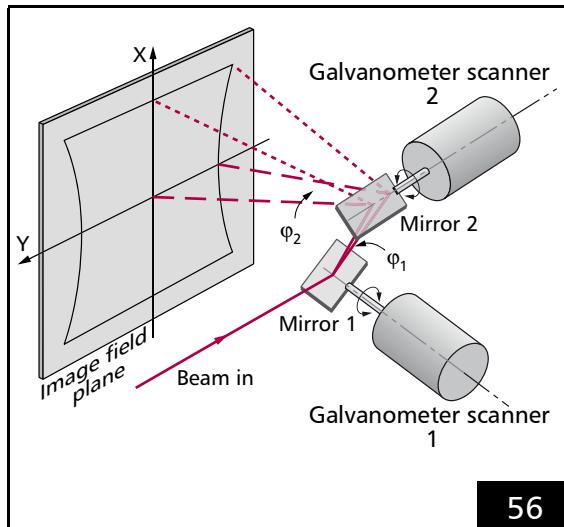
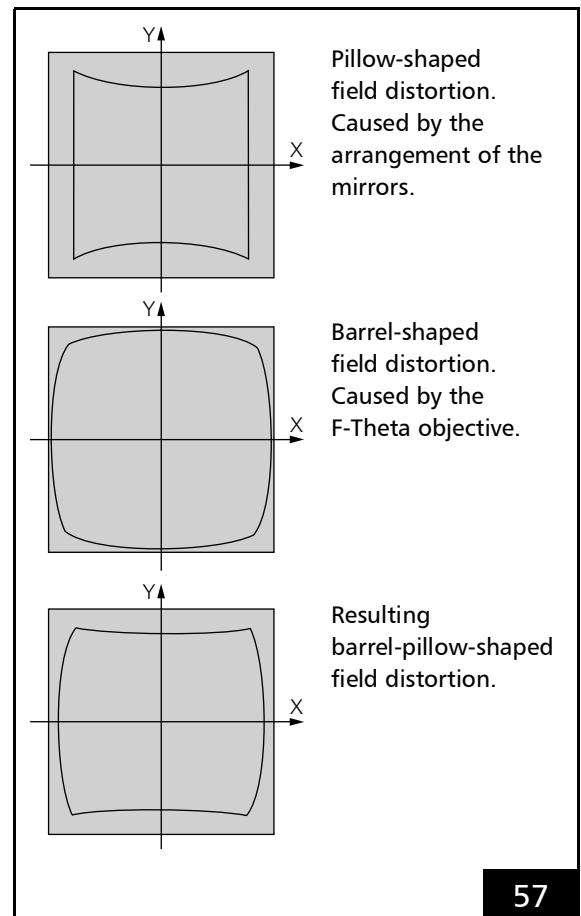


Image field distortion when deflecting a beam in a 2-mirror deflection system.

By focusing the deflected laser beam with an F-Theta objective, effect 2 and effect 3 can be avoided. However, this causes a barrel-shaped distortion of the **Image field**, see Figure 57.



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.

## Image Field Correction Algorithm

For these field distortions, the RTC5 PCI 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 RTC5 PCI 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 128](#).

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). Individual system characteristics and errors during adjustment are not taken into account.

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 data derived from actual test measurements performed by the customer. 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 RTC5 PCI 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 Correction Files and 3D Correction Files

SCANLAB supplies 2D correction files 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 4<sup>(1)</sup> correction tables can be loaded from their corresponding correction file into the RTC5 PCI Board memory, see also [Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 221](#).

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` after the assignment by `select_cor_table` (see below), see [Chapter 7.3.4 "3D Image Field", page 159](#).

### Notes

- ct5-correction files 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 )`).

(1) Up to 8 on request. With additional limitations on data recording and available list memory.

## Notes

- If no correction table has been loaded into the RTC5 PCI 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 correction tables #1 and #2. The correction table values are random immediately after switching on the RTC5 PCI 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 [RTC5 files](#) have been loaded by `load_program_file`.



## 1to1 Correction Tables

### 1to1 correction tables

- 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 1to1 correction table or 3D 1to1 correction table is being generated internally.

Because some user programs require a file name and do not accept **NULL** as the name, the 1to1 correction file **Cor\_1to1.ct5** is supplied within the RTC5 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 27** (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 200**
- **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.txt](#) 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 <a href="#">Image field</a> 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 <code>D2_145.ct5</code> or <code>D3_145.ct5</code> ).

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 <code>ctb</code> file</li> <li>• <math>P_{Orig} = 30000</math>: reconstructed from <code>txt</code> 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</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 <code>ctb</code> 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 <b>Image field</b> 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 RTC5 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](#),  $P_{\text{Orig}} = 20,000$ ).

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

Calculation steps for generating the RTC output values to the scan system from the current x, y, z coordinate values	
(1)	Decomposing vectors and arcs to Microsteps <sup>(a)</sup>
(2)	If applicable: Applying a wobble motion <sup>(b)</sup> – for example, from <code>set_wobble_mode</code> call
(3)	If applicable: Applying a global coordinate transformation in virtual Image field <sup>(c)</sup> – for example, from calls of <code>set_matrix</code> , <code>set_angle</code> , <code>set_offset</code> with HeadNo = 4
(4)	If applicable: Applying the set Processing-on-the-fly correction <sup>(d)</sup> – for example, from calls of <code>set_fly_x</code> or <code>set_fly_x_pos</code>
(5)	If applicable: Applying coordinate transformations to align the 1 (2) scan system(s) relative to the Image field <sup>(e)</sup> (a) Transforming x coordinate values and y coordinate values as per matrix × angle × scale ("2D transformation") • for example, from calls of <code>set_matrix</code> , <code>set_angle</code> , <code>set_scale</code> and corresponding list commands (b) Applying an offset to x coordinate values and y coordinate values as well as z coordinate values <sup>(f)</sup> • for example, from calls of <code>set_offset</code> or <code>set_offset_xyz</code> and corresponding list commands
(6)	Clipping x coordinate values and y coordinate values to the boundary values of the real 20-bit Image field as soon as they exceed the real Image field range (virtual Image field) <sup>(g)</sup>
(7)	If applicable: Correcting x coordinate values and y coordinate values according to the assigned correction table ("2D Image field correction") <sup>(h)</sup> – From <code>select_cor_table</code> call or <code>select_cor_table_list</code> call
(8)	If applicable: Applying the focus shift to z coordinate values – for example, from <code>set_defocus</code> call or <code>set_defocus_list</code> call
(9)	If applicable: Correcting z coordinate values according to the assigned correction table ("3D image field correction") <sup>(h)</sup> – From <code>select_cor_table</code> call or <code>select_cor_table_list</code> call
(10)	Clipping z coordinate values to the boundary values of the real 20-bit Image field as soon as they exceed the real Image field range (virtual Image field) <sup>(i)</sup>
(11)	If applicable (only some legacy scan systems): Applying an explicitly <sup>(j)</sup> or automatically <sup>(k)</sup> set gain correction and offset correction for the x galvanometer scanner and y galvanometer scanner – From <code>set_hi</code> call (explicit) – From <code>auto_cal</code> call (automatic)

(a) See Chapter 7.1.2 "Microstepping", page 128.

(b) See Chapter 8.4 "Wobble Mode", page 217.

(c) See Section "Coordinate Transformations in the Virtual Image Field", page 158.

(d) See Chapter 8.6 "Processing-on-the-fly", page 227.

(e) See Chapter 8.2 "Coordinate Transformations", page 209 and Chapter 8.5.2 "3D Scan Systems", page 222.

(f) A complete 3D calculation is always performed. If the Option "3D" is not enabled, z values are just not outputted.

(g) See Chapter 7.3.3 "Virtual Image Field", page 158.

(h) See Chapter 7.3.5 "Image Field Correction and Correction Tables", page 162.

(i) See Chapter 7.3.3 "Virtual Image Field", page 158.

(j) See Section "Customer-Specific Calibration", page 254.

(k) See Chapter 8.10 "Automatic Self-Calibration", page 251.

## 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 RTC5 PCI Board, see also:

- **Section "RTC5 Standard Mode", page 157**
- **Section "RTC4 Compatibility Mode", page 157**
- **Chapter 4.5.2 "XY2-100 Converter (Accessory)", page 56**

## 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 cycle might not always suffice for calculating all data required by the computation-intensive **Jump Commands**, **Mark Commands** 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 cycle, 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 #5a.
  - any z coordinate offset defined by `set_offset_xyz` or `set_offset_xyz_list`, see #5b.
  - 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 RTC5 PCI 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.

The current status of a measurement session started with `set_trigger/set_trigger4` can be queried by `measurement_status`.

The values returned by the scan system are always 20-bit values.

### Status Information Returned from the Scan System

The scan system transmits the following signals to the RTC5 PCI Board every 10 µs via the SL2-100 protocol:

- A 20-bit status word. It can mean the following data types:
  - The actual 20-bit status word (the upper 16 bits are identical to the 16-bit status word of the XY2-100 protocol ([XY2-100 status word](#)), for a description see `get_head_status`).
  - Only for iDRIVE scan systems<sup>(1)</sup>, see [Chapter 8.1 "iDRIVE Functions", page 198](#): 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 iDRIVE 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 187](#)) 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 [XY2-100 Converter \(Accessory\)](#).
- For iDRIVE scan systems<sup>(1)</sup> with XY2-100 protocol, the actual [XY2-100 status word](#) must be set as the to-be-returned data type (default type after switching on).

(1) See Glossary entry on [page 23](#).

## 7.4 Laser Control

At its laser signal output ports (LASERON, LASER1 and LASER2), the RTC5 PCI 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 60](#)
- the EXTENSION 2 socket connector, see [Chapter 4.6.3 "EXTENSION 2 Socket Connector", page 67](#)

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.

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 60](#). Their current setting can be queried by `get_startstop_info` (Bit #13).

For the maximum current load values, see [Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734](#).

### 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`

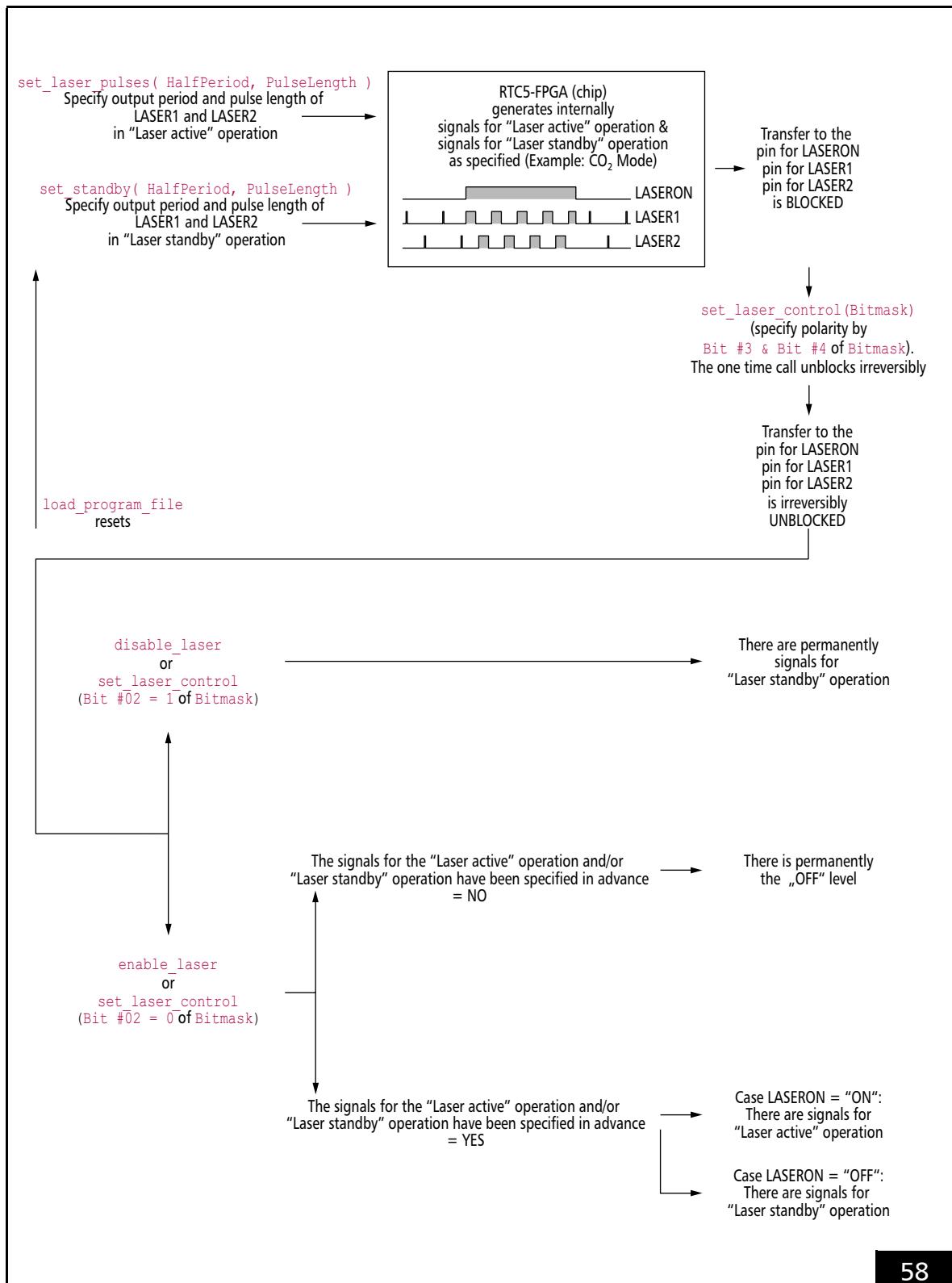
These can be suppressed again by `disable_laser` or `pause_list`. In both cases, all laser control signals are set to their respective "Off" levels.

By `get_startstop_info` (Bit #13, Bit #10 as well as Bit #9) it can be read out:

- The polarities of LASERON and LASER1/LASER2
- The "global unblock" by `set_laser_control`
- With  $\geq$  DLL 546,  $\geq$  OUT 546: The enabling of the **Signals 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 186](#).

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](#).



RTC5 boards hardware and laser control-related RTC5 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

- activated: `HalfPeriod ≠ 0` and `PulseLength ≠ 0`
- deactivated: `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 Commands and “Arc” Commands”, page 124](#)

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

- activated: `HalfPeriod ≠ 0` and `PulseLength ≠ 0`
- deactivated: `HalfPeriod = 0` and/or `PulseLength = 0`

The Signals for “Laser Standby” Operation are continuously outputted at the output ports after their activation (without further commands), if no Signals for “Laser Active” Operation are switched on.

If the Signals for “Laser Active” Operation are deactivated (for example, by `PulseLength = 0`), there is no changeover. Then the Signals for “Laser Standby” Operation are continuously outputted even during the execution of **Mark commands**.

If the Signals for “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 “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 “Laser Standby” Operation, pulse completion, see [Section “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)

### 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; that is, 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 17.

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").

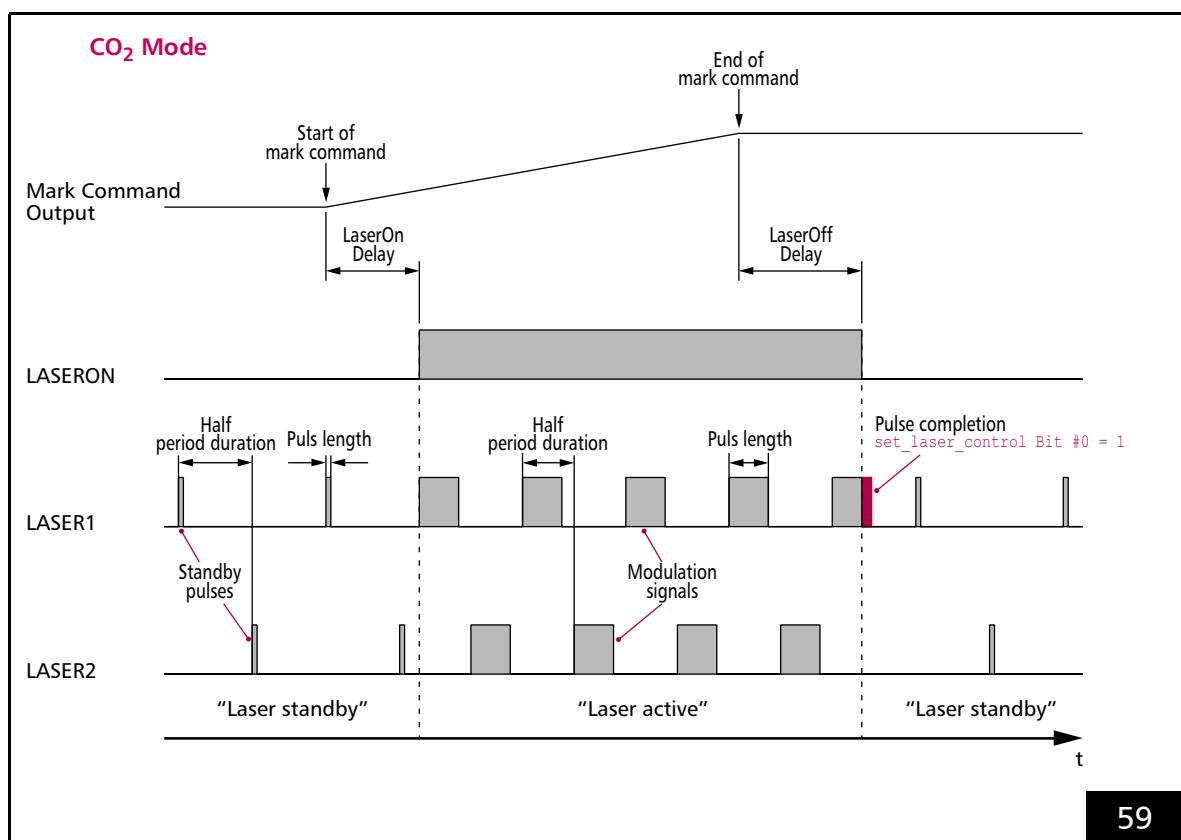
The timing diagram, see Figure 59, shows the corresponding signals using the example of an isolated mark command.

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 signals and LASER2 signals can be varied. Both signals share the same pulse lengths and periods.
- For the LASER1 signals and LASER2 signals, *half* of the output period must be specified.
- [set\\_laser\\_pulses](#) and [set\\_laser\\_timing](#) are delayed short control 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  $\mu$ s in [RTC5 Standard Mode](#),  
see also [Section "RTC5 Standard Mode"](#),  
[page 157](#)
  - 1/8  $\mu$ s in [RTC4 Compatibility Mode](#),  
see also [Section "RTC4 Compatibility Mode"](#),  
[page 157](#)
- [Section "Pulse Completion"](#), page 175 affects also LASER2 signals.

#### 7.4.4 YAG Mode 1, 2, 3, 5

`set_laser_mode( [1, 2, 3 or 5] )` sets one of the YAG modes.

The timing diagram, see [Figure 60](#), 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 [Section “Q-Switch Signal”, page 179](#).
- A adjustable *FirstPulseKiller signal* is outputted at the LASER2 output port, see [Section “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 [Section “Pulse Completion”, page 175 \(Signals for “Laser Active” Operation\)](#).
- See also [Section “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 60](#).

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  $\mu$ s
- 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](#).

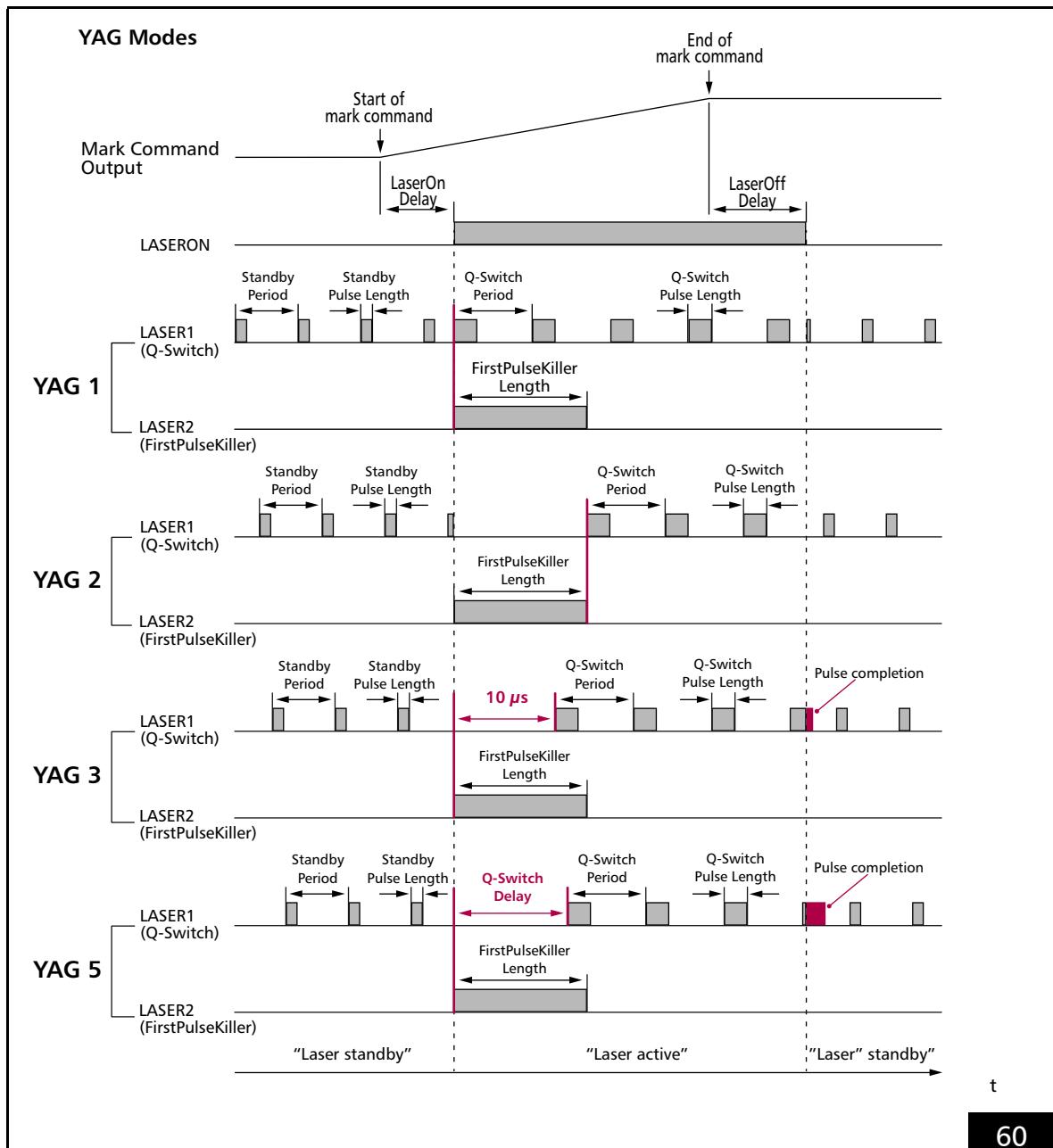
## Lamp Current (Laser Power)

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 17](#). [ANALOG OUT2](#) is also available at the MARKING ON THE FLY socket connector, see [Figure 25](#).

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 24](#)), see also [Section "8-Bit Digital Output Port", page 68](#). 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 61](#), 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:

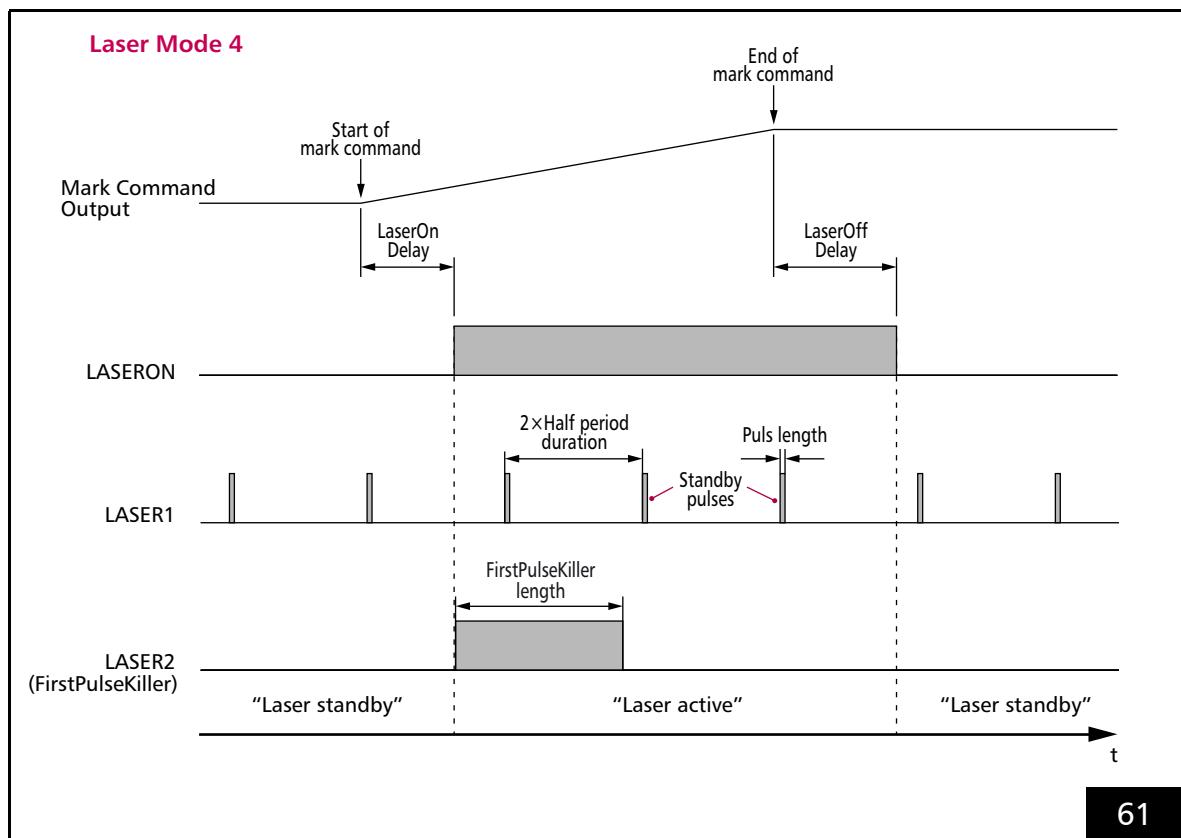
- The LASERON signal is switched on
- A programmable FirstPulseKiller signal is outputted at the LASER2 output

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.

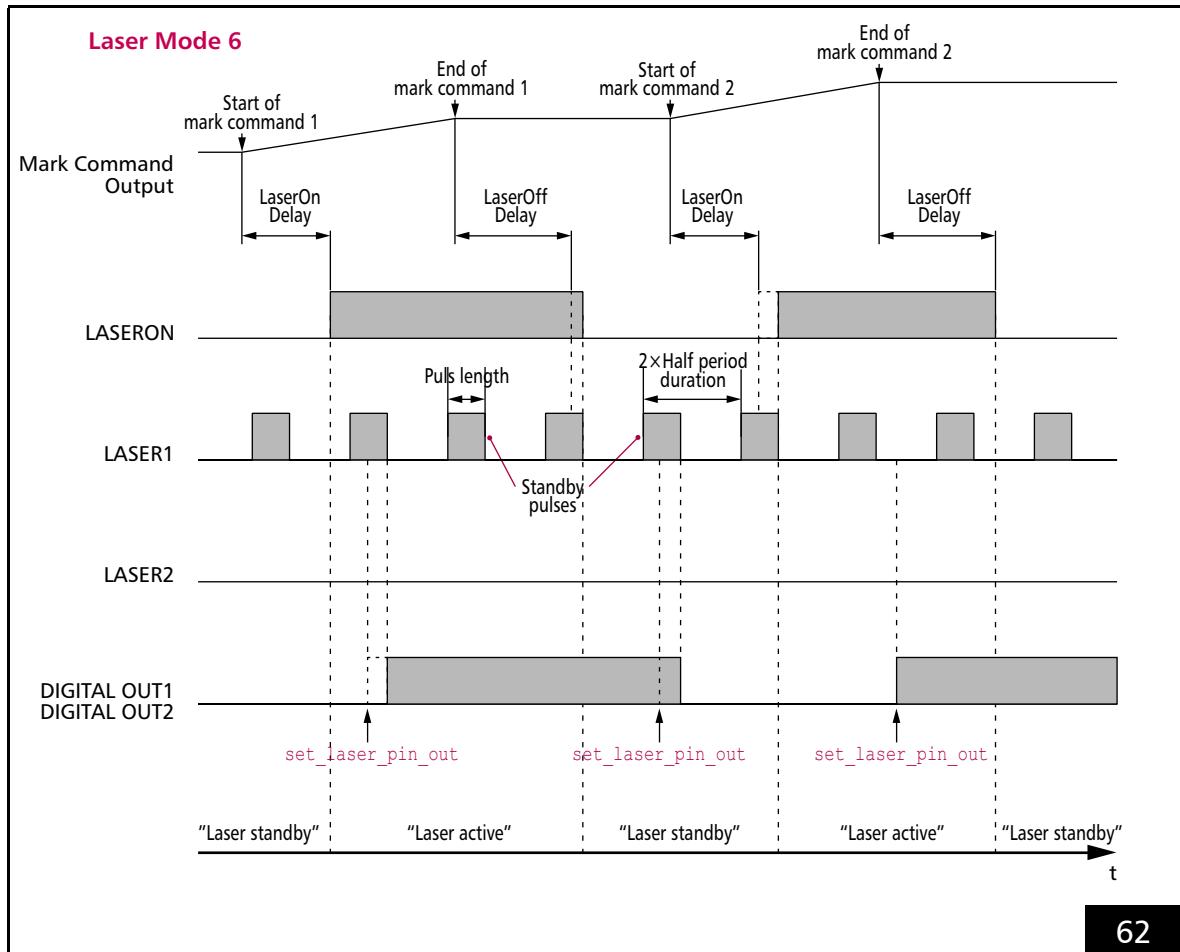
The timing diagram, see [Figure 62](#), shows the corresponding signals using the example of an isolated mark command.

**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 62](#).



Timing diagram of the signals in **Laser Mode 6** (with active-HIGH laser control signals). Example: isolated **Mark Commands**.

### 7.4.7 Softstart Mode

For some applications it is important to control the laser intensity at the beginning of a marking process. SCANLAB provides a convenient solution in the form of the **Softstart Mode**, which can be used for all laser modes (for "laser active" operation).

In the **Softstart Mode**, various control values (`Level(0)`...`Level(Number - 1)`, `Number`  $\leq$  1024) for as many as the first 1024 pulses (at the beginning of a marking process) can be defined. Either pulse length values for the laser signal at the LASER1 output (see [Figure 17](#)) or analog voltage levels for variable laser power can be defined. Analog voltage softstart values can be outputted either at the [ANALOG OUT1](#) or [ANALOG OUT2](#) output port, see [Figure 17](#).

Initialization of the **Softstart Mode** is performed by `set_softstart_mode` or `set_softstart_mode_list`. Here, the type of softstart value is defined. If analog voltage values are to be outputted, then the analog output port can also be defined. Afterwards the individual softstart control values can be defined by `set_softstart_level` or `set_softstart_level_list` commands.

After the laser control signals are switched on, the defined softstart values (max. 1024) are outputted at the selected output port simultaneously with the laser pulses of the LASER1 signal – always with the leading edge of the laser pulse. The `set_softstart_mode`/`set_softstart_mode_list` calls allows specification of a time period (`Delay`) for which the laser must have been switched off before softstart is activated at the next switch-on.

#### Notes

- With a large enough value for the `Delay` time, one can avoid the **Softstart Mode** being also activated after very short **Jump commands**.
- As soon as the LASERON signal (the laser) remains switched off longer than `Delay`, the value `Level(0)` is assigned to the output port, provided the **Softstart Mode** has not been meanwhile deactivated.
- When the LASERON signal is switched on, then the values `Level(0)`...`(Number - 1)` are automatically assigned to the output port simultaneously with the first laser pulses.
- Analog softstart values are outputted simultaneously with the laser pulses of the LASER1 signal. It might be beneficial under some circumstances to acquire these analog softstart values triggered by a signal shifted back 180° related to the LASER1 signal. For this purpose, the **CO<sub>2</sub> Mode** provides a LASER2 signal at the LASER2 output. And in the YAG version's, the signal at the LASER1 output can be phase-shifted back 180° by `set_laser_control` (Bit #1 = 1). For the **CO<sub>2</sub> Mode**, this is equivalent to exchanging LASER1 with LASER2.
- The laser pulse frequency should not exceed a value of approx. 308 kHz (this corresponds to a `set_laser_pulses-HalfPeriod` of 104), because the changing values cannot be transmitted faster. However, the laser pulse frequency cannot be automatically checked for this case.
- If Softstart values are to be outputted as analog voltage levels, then also note that digital-to-analog conversion of laser frequencies above around 100 kHz (that is, for a `set_laser_pulses-HalfPeriod` < approx. 320) cannot always be fully completed. For such laser frequencies, users must carefully verify that sufficient capability is available.
- If **Softstart Mode** is inactive, then values for the analog outputs can be set by `write_da_x` or `write_da_x_list`.



- The **Softstart Mode** is also available in **Laser Mode 4** and **Laser Mode 6**. In this modes, the LASER1 output only outputs standby pulses and therefore only analog voltage levels can be variably defined (`set_softstart_mode Mode = 1, 2, 11 or 12`). Here, too, the defined softstart values are outputted simultaneously with the pulses of an internally-generated (but not outputted) LASER1 signal. The standby pulses cannot be phase shifted in **Laser Mode 4** and **Laser Mode 6** and are not synchronized with the internal LASER1 pulses. If the period durations of the standby and LASER1 pulses are set (by `set_laser_pulses` and `set_standby`) to be equal, then the analog voltage softstart values are outputted in parallel with the standby pulses, though they might have a (random) constant phase shift with respect to the standby pulses.
- The **Softstart Mode** cannot be used simultaneously with the "freely definable wobble shapes" wobble mode, see `set_wobble_vector`.

#### 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 63](#) shows the corresponding signals.

For "laser active" operation:

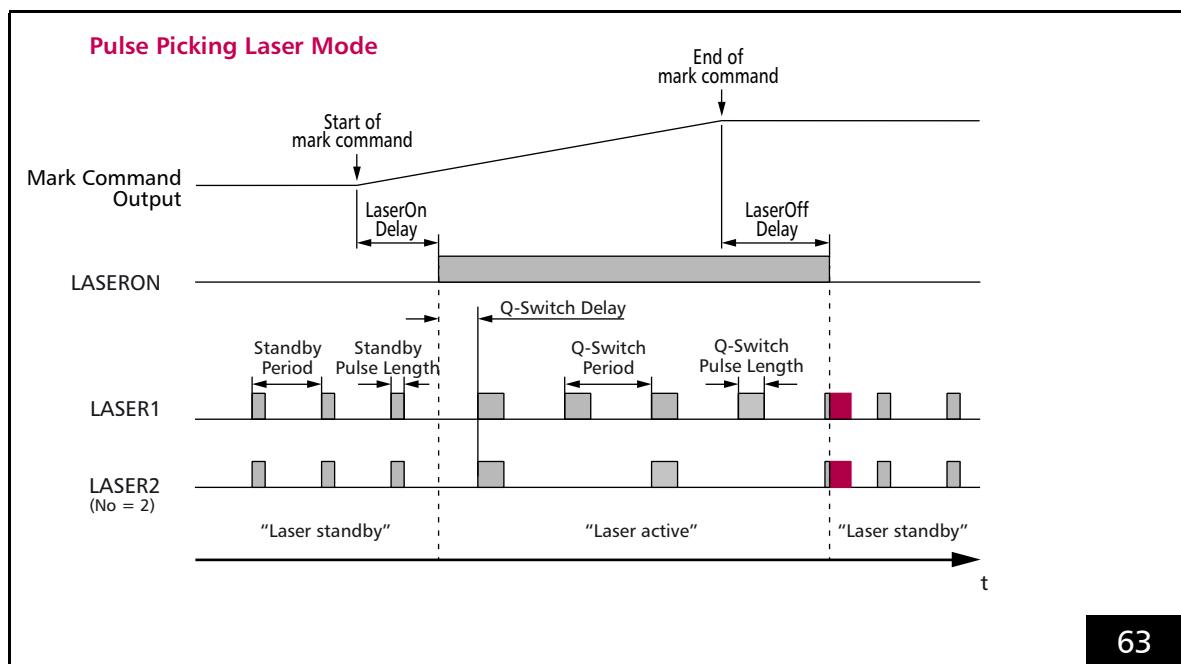
- The LASERON signal is switched on
- A modulation signal is provided at the LASER1 output with pulse length and period duration that can be defined by `set_laser_pulses`, `set_laser_pulses_ctrl` or `set_laser_timing`
- At the LASER2 output every  $No^{th}$  pulse of the signals is outputted at the LASER1 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 180](#).
- 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 **Signals for "Laser Active" Operation** – 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 189, 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 191, allows compensation of laser energy input variations during execution of **Mark commands** that resulting from an uneven galvanometer scanner motion (acceleration phase and deceleration phase, time-dependent **Mark commands**).
- *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 scanners and workpiece move simultaneously.

- *Vector-defined* laser control, see **Section "Vector-Defined Laser Control"**, page 194, allows linear readjustment of a signal parameter along parameterized mark vectors or jump vectors (see **Section "[\*]Para[\*] Commands"**, page 127).

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 61
- Output value at the 16-bit digital output port of the **EXTENSION 1** socket connector, see also **Section "16-Bit Digital Input Port and 16-Bit Digital Output Port"**, page 65
- Output value at the 8-bit digital output port of the **EXTENSION 2** socket connector, see also **Section "8-Bit Digital Output Port"**, page 68
- 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 192.
- 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:
  - Signals for “Laser Active” Operation are switched off when “Automatic Laser Control” is active
  - “Automatic Laser Control” itself is deactivated If 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 it
 The **Mode** parameter cannot be changed subsequently.

(1) < DLL 546: 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 189](#).

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.

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 (#5a and #5b)
- excluding `Image field` correction (#7)

### 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 141](#).



- For  $\langle\text{Value}\rangle$ , the following ranges apply:  
 $0.0 \leq \text{Position} \leq 150.0$  and  
 $0.0 \leq \text{Scale}(\text{Position}) \leq 4.0$ .
- Each instruction must be in a separate line.
- Spaces and tabs in a line (for example, between '=' and  $\langle\text{Value}\rangle$ ) are ignored.
- Empty lines are ignored.
- Data points with invalid values are ignored.
- The data point of a particular index  $\langle n \rangle$  is ignored if the corresponding  $\text{Position}\langle n \rangle$  and/or  $\text{Scale}\langle n \rangle$  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, **load\_position\_control** has no effect (return value 1 or 13).
- If there is no entry for  $\text{Position} = 0.0$ , then an entry with  $\text{Scale} = \text{Min}(\text{Scale}\langle i \rangle)$  is inserted (the smallest allowed value defined in the table is used for lower positions values). Likewise for  $\text{Position} = 150.0$  with  $\text{Max}(\text{Scale}\langle i \rangle)$ .
- If the selected text file only contains a single valid data point with  $\text{Scale}\langle n \rangle = S$ , then (for the entire position range) the scaling function  $\text{Scale}(\text{Position}) = S$  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** = **NULL** in **load\_position\_control**.

## 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  $\mu$ s is used as input. It may deviate from the target speed due to the 10  $\mu$ s 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.

## 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 RTC5 PCI 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 23](#).

### 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 141.

- 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 194.

In the following example, a nonlinearity factor of 1.2 is set for a 1.5x multiple of the target value:

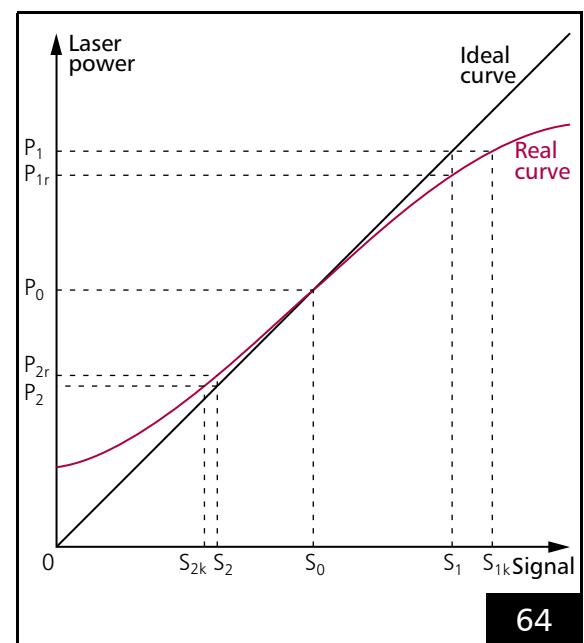
Percent<n> = 150

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 = NULL** in **load\_auto\_laser\_control**.

The example diagram in **Figure 64** illustrates how the nonlinearity curve can be determined.



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:

$$\text{Percent}_1 = S_1/S_0 \times 100 = P_1/P_0 \times 100$$

$$\text{Scale}_1 = S_{1k}/S_1$$

$$\text{Percent}_2 = S_2/S_0 \times 100 = P_2/P_0 \times 100$$

$$\text{Scale}_2 = S_{2k}/S_2$$



## 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[\*] Command**s 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[\*] Command**s.
- If the same output port is selected via **set\_auto\_laser\_control** for **Ctrl**, the control parameter from the **[\*]para[\*] Command**s 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[\*] Command**s, 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 Output Synchronization

The RTC5 provides the so-called (galvanometer scanner) "output synchronization" functionality for synchronizing the scan system's scanning motions (during all marking commands) to the laser pulses of a free-running laser.

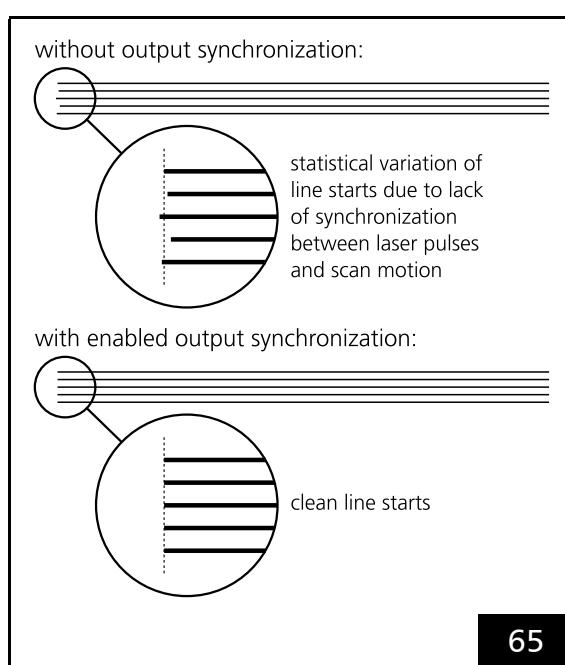
The laser pulse signal (the synchronization master clock) must be supplied at the LASER connector's DIGITAL IN1 digital input (see [Section "2-Bit Digital Input Port", page 61](#)).

Output synchronization is enabled or disabled by Bit #6 of [set\\_laser\\_control](#).

When enabled, output synchronization affects all [Mark command](#) marking commands (mark, arc and ellipse). At each marking command's start, the RTC5 determines the time shift between that marking command's starting time and the first laser pulse after LASERON<sup>(1)</sup> and appropriately corrects the scan system output values. The RTC5 thus ensures that the position of the galvanometer scanners upon the first laser pulse output after LASERON does not depend on random phase shifts of the laser pulse. This way, jittery line images can be avoided, see [Figure 65](#).

#### Notes

- The supplied laser pulse signal must have a frequency between 50 kHz and 6.4 MHz and its pulse length must exceed 0.080  $\mu$ s.
- The output period of the laser control signals must be set according to the laser pulse frequency in the user program (before enabling output synchronization) by [set\\_laser\\_pulses](#), [set\\_laser\\_pulses\\_ctrl](#) or [set\\_laser\\_timing](#). This also applies, if [Laser Mode 4](#) standby signals are used as external input.
- For frequencies between 50 kHz and 100 kHz, the [LaserOn Delay](#) must be set to 50  $\mu$ s, otherwise the [LaserOn Delay](#) must exceed 40  $\mu$ s.
- If no laser pulse appears at the DIGITAL IN1 digital input within 20  $\mu$ s of a marking command's start, then no output synchronization is performed on that command's output.
- Output synchronization can also be used in conjunction with [Sky Writing](#).
- Output synchronization cannot be used together with [Pixel mode](#).



65

Example of line image marked with a free-running laser.

(1) the first laser pulse output following the [LaserOn Delay](#)

## 7.5 Marking Dates, Times and Serial Numbers

The **RTC5 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 Times, Dates and Serial Numbers", page 108](#).

### 7.5.1 Marking the Date and Time

By **time\_update** the PC time/date is transferred to the RTC5 PCI 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  $\mu$ s 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 RTC5 PCI 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 has been or has not been 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 RTC5 PCI 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 transmit to the controller
- To choose from different dynamic settings (tunings)
- To set the **PosAck** limit 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>.

During the **control\_command** execution, the RTC5 PCI Board does not transfer position data to the scan system. Due to this, the motion of the galvanometer scanners is briefly interrupted.

#### 8.1.1 Transfer Protocol

Data transfer between RTC5 PCI 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 axis and y axis 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 **XY2-100 Converter (Accessory)** 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 23](#).

(2) See also [Section "Folder iSCANConfig", page 28](#).

### 8.1.2 Configuring the Data Signal Transmission Behavior of the Scan System

#### Setting Data Types

The digital servo architecture of iDRIVE scan systems<sup>(1)</sup> allows a wide variety of data signals to be returned from the scan system to the RTC-control board.

Each axis has its own status channel on which data is transmitted to the RTC-control board every 10 µs:

- STATUS channel
  - Designed for the x axis  
(Galvanometer scanner 2)
- STATUS1 channel
  - 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 transmit to the RTC5 PCI 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 RTC5 PCI 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 788](#).

The value ranges of these data and the possible status states are described in [Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747](#).

#### 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 `XY2-100 Converter (Accessory)` is used for control, then it must explicitly be set that the scan system has to transfer the `XY2-100 status word` to the RTC5 PCI Board. Otherwise, `get_head_status` returns unusable values.

(1) See Glossary entry on [page 23](#).

### 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 RTC5 PCI 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 RTC5 PCI Board itself. For this, all correction and transformation settings currently assigned to the scan system can be transferred from the RTC5 PCI 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 (#5a and #5b on <a href="#">page 170</a> )	yes
Offset to the z coordinate (#5b on <a href="#">page 170</a> )	yes
Clipping to the limits of the controllable <b>Image field</b> (#6 on <a href="#">page 170</a> )	no
2D <b>Image field</b> correction (#7 on <a href="#">page 170</a> )	yes
3D <b>image field</b> correction (#9 on <a href="#">page 170</a> )	yes
Gain and offset correction of automatic self-calibration (#11 on <a href="#">page 170</a> )	yes
Clipping to the maximum possible control values	no

- Furthermore, backward transformation is not possible if a noninvertible 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 209](#)). The non-invertibility is already reported by **upload\_transform**.

(1) Due to communication runtimes, the currently returned actual positions are several clock pulses later than the currently outputted control signals.

- 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.
- By **get\_values**, 4 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.
- By **set\_trigger**, you can simultaneously record two desired signals by two measurement channels (with **set\_trigger4** 4 signals by 4 measurement channels).
- By **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 RTC5 PCI 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 RTC5 PCI 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 Configuring Tuning (Dynamics Settings)

SCANLAB can optimize the dynamics setting of scan systems (tuning) to accommodate differing requirements of diverse applications regarding the laser positioning dynamics, for example,

- Vector tuning  
to execute vectors or circular arcs at a constant processing speed
- Jump tuning  
to execute jumps of minimized duration

*iDRIVE* scan systems<sup>(1)</sup> can be optionally equipped with several tunings. For different applications, the suitable tuning can be switched to – separately for each axis – by **control\_command**(*Data* = 11nn<sub>H</sub>).

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 202](#).

The default set start behavior is that the scan system starts with tuning number 0 upon power-up or after a reset.

### 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 203](#).

#### 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 has been set before the jump, then the RTC5 PCI 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 RTC5 PCI 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**).

(1) See Glossary entry on [page 23](#).

- 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** has been 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  $\mu$ s clock cycle. 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 has been most recently set at the end of **Jump Mode**, unless deactivation has been 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, intellicube, intelliWELD or intelliDRILL scan system.
- As of scan system firmware  $\geq$  2078.
- 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 in the scan system 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** has not been 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 RTC5 PCI 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 RTC5 PCI Board in a user-specific **Jump Delay** table. With **Jump Mode** enabled, the RTC5 PCI 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 **load\_jump\_table\_offset**.

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.

## Experimental Determination of **Jump Delay** Values

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  $\pm 0.01$ .

- 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* = *Min(Delay<i>)* is inserted (the smallest valid value encountered is filled downward). The same applies to *Length* = 524288.0 with *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.

(1) Even of another type, see table 1, page 141.

## 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 point in time at which the specified position tolerance ( **PosAck** parameter) has been 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]. **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, 4 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 only whole multiples of 16 are returned with an **XY2-100 Converter (Accessory)**). 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 (Accessory)**, 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. `load_jump_table` and `load_jump_table_offset(Offset = 0)` are identical.
- 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 **PosAck** Limit Value

`control_command(Data = 15nnH)` can be used to set the **PosAck** limit value nn. The default start behavior is for the scan system to set the limit value to 0.28% of the full position range after every power-up.

If other limit 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 RTC5 PCI Board by a specific factor (1, 1/2, 1/4 or 1/8). The position signals (optionally) returned by the scan systems to the RTC5 PCI Board remain unaffected, as do the pre-configured calibrations of SCANLAB 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 scaling factor of 1 upon power-up.

By `control_command(Data = 053FH)`, the currently set scaling factor can be queried.

### 8.1.8 Configuring the Start Behavior

The default configuration of iDRI<sup>VE</sup> scan systems<sup>(1)</sup> is set as follows:

- Tuning number 0, see also [Section "Configuring Tuning \(Dynamics Settings\)", page 202](#)
- **PosAck** limit value is B8<sub>H</sub> (corresponds to 0.28% of the full position range of 2<sup>16</sup> bit), see also [Section "Configuring the PosAck Limit Value", page 207](#)
- Scaling factor = 1, see also [Chapter 8.1.7 "Configuring the Effective Calibration", page 207](#)

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 transmission behavior of the scan system can only be temporarily changed, see also [Chapter 8.1.2 "Configuring the Data Signal Transmission Behavior of the Scan System", page 199](#). After a power-up, the scan system transmits the XY2-100 status word.

### 8.1.9 Fault Diagnosis and Functional Test

If a problem occurs, the versatile status return functions of the iDRI<sup>VE</sup> scan system<sup>(2)</sup> can be used for scan system diagnosis, too.

For example, an event code can be queried that indicates which event has been responsible for the change to an error state.

To verify that data transfer capability between the RTC5 PCI 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 Transmission Behavior of the Scan System", page 199](#).

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 transmission behavior as before the test.

(1) See Glossary entry on [page 23](#).

(2) See Glossary entry on [page 23](#).

## 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 commands** 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 x 2) total matrix M is thereby automatically calculated by the RTC5 PCI 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 counterclockwise 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 RTC5 PCI 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 called `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 RTC5 PCI 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 list execution 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 sequence 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 );
```

#### Notes on data recording:

- The galvanometer scanner motion of **jump\_abs** or **jump\_rel** at the beginning of the list can be recorded with **set\_trigger** (signal type **7, 8, 9**), if no coordinate transformation with `at_once = 2` has been applied before.
- In case of motions after a coordinate transformation (for example, by **set\_offset\_list** with `at_once = 2`) the "sample values" (**set\_trigger** signal type **7, 8, 9**) are immediately set to the target coordinates and remain unchanged for the duration of the motion. The motion recorded here does not appear like **Microstepping**<sup>(1)</sup> has been performed, but rather like a **Hard jump**. The real galvanometer scanner motion can be recorded with **set\_trigger**, signal type **25** and **26** (transformed control values).
- In case of motions caused by coordinate transformation in a list (for example, by **set\_offset\_list** with `at_once = 1`), the "sample values" (**set\_trigger** signal type **7, 8, 9**) remain unchanged for the duration of the motion. The real galvanometer scanner motion can be recorded with **set\_trigger**, signal type **25** and **26** (transformed control values).
- Settings via control commands by `at_once = 0` are only saved as long as no list is running. When the execution is started or the list is already 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 RTC5 PCI Board. They take effect when a correction table is assigned.

(1) See **Chapter 7.1.2 "Microstepping", page 128.**



- 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 RTC5 PCI 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.
- RTC5 PCI 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, #5a and #5b** 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 209](#) 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 conveyor belt and then execute the associated transformations.

However, it is not easy to achieve well-controlled timing (referenced to the RTC5's 10  $\mu$ s clock) while positioning a workpiece and aligning the scan system by the (control) commands described in [Chapter 8.2 "Coordinate Transformations", page 209](#). 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 (ever present) **"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 #5a and #5b 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 213](#).
- 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 216](#).

**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 #5a and #5b 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 activated and configured as desired with the commands **set\_mcbsp\_x**, **set\_mcbsp\_y** and/or **set\_mcbsp\_rot** or **set\_mcbsp\_matrix** (or by the equivalent list commands) (see below). 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 209](#) 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 271](#)).

The **McBSP interface** is described on [Chapter 4.6.6 "SPI / I2C Socket Connector", page 71](#).

### 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 RTC5 and which internal memory location is used to read the values:

- Depending on which of the above commands is called, the RTC5 interprets the inputted values as offsets in the x direction and/or y direction 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 4 options becomes activated, all values subsequently inputted at the **McBSP interface** are internally stored in memory location 1 or possibly memory location 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 or 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 signed 16-bit values, 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 RTC5 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 and 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 RTC5 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 signed 32-bit 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/SPI 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 possibly 2 (see above) and are then available for “**Local Online Positioning**”, but no longer for a Processing-on-the-fly application.
  - 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 copied possibly instead of or additionally to the internal memory location 0 and/or 3 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 DLL 545, OUT 545) 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 213](#), 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 (see [Chapter 8.3.1 ““Local Online Positioning””, page 213](#)) 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.

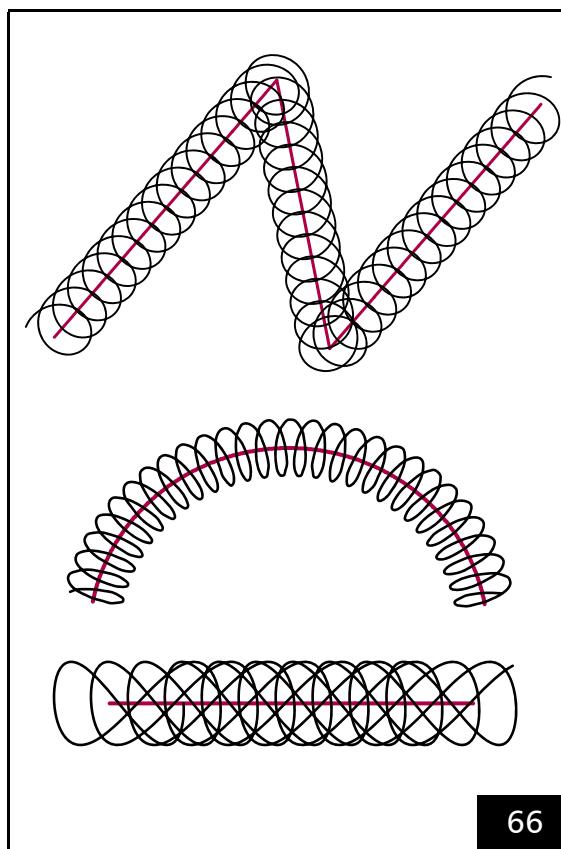
#### Configuring “Global Online Positioning”

The “Global Online Positioning” is configured the same way as the “Local Online Positioning”, see [Chapter 8.3.1 ““Local Online Positioning””, page 213](#).

## 8.4 Wobble Mode

The wobble mode allows varying the line width for laser marking.

For this purpose, an ellipse-shaped motion is added to the regular, linear motion of the output position. This results in a spiral motion of the laser focus in the **Image field**, see [Figure 66](#). Alternatively, a combination with a figure-of-8 (horizontal or vertical to the direction of motion) can be activated.



66

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 motion. 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.

As of version DLL 543, OUT 543, RBF 524 the present wobble amplitude (from [set\\_wobble](#) or [set\\_wobble\\_mode](#)) can be recorded with trigger signal 53 (see [set\\_trigger](#) or [set\\_trigger4](#)). The format of the data is ((transversal << 16) + longitudinal).

During Processing-on-the-fly correction by the McBSP/SPI or encoder interfaces where the scan head only compensates differences between actual external motions and intended total motion (see [Chapter 8.6.2 "Compensation of Linear Motions", page 228](#), a slight jitter in the direction of galvanometer scanner motion might occur, particularly during exact external path motions. Here, the wobble motion 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.

For optimum marking results, the wobble frequency should be appropriate for the specified mark speed. In some cases it can be useful to adjust the mark speed when the wobble mode is used.

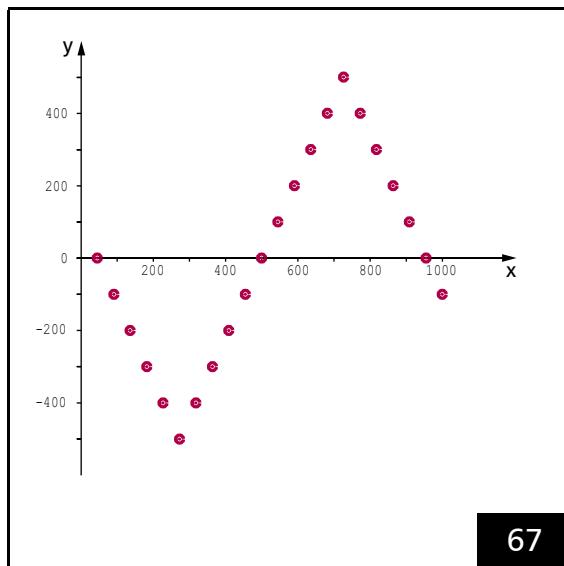
After [set\\_wobble](#) or [set\\_wobble\\_mode](#), the wobble start point is always set for the same value relative to the vector/arc startpoint and direction. The Wobble phase is then continued both within an uninterrupted [Polyline](#) (including arcs) and after interruptions (for example, by a [Jump command](#)) until [set\\_wobble](#) or [set\\_wobble\\_mode](#) are called again.

The wobble mode cannot be combined with:

- [Sky Writing](#)
- [Pixel Output Mode](#)
- Jumps
- [laser\\_on\\_list](#)

For further details, see [set\\_wobble](#) and [set\\_wobble\\_mode](#).

For many welding applications, the default set of "classic" wobble shapes (circle, ellipse, sine, figure-8) does not produce optimal (smooth) results in the area of the weld seam. With ellipses, for example, high-speed motion occurs parallel to translation motions and low-speed motion occurs in the opposing direction, resulting in uneven energy deposition. `set_wobble_vector` lets you define a wobble shape customized for your user program, consisting of up to 512 piecewise linear sections, while also specifying variation of laser power along that shape (see also `set_wobble_control`). However, `set_wobble_vector` cannot be used together with the Softstart function nor with automatic or vector-based laser control.



67

See Section "Example Code (C++)", page 218.

### Example Code (C++)

The following example C++ source code shows a zigzag pattern, see Figure 67.

The code must be included in a user program, see Chapter 6.2.5 "Example Code (C)", page 89.

```
// 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. The galvanometer scanners 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 galvanometer scanners 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  $T = \text{Tracking error}^{(1)}$

(2) Wobble amplitude<sup>(2)</sup>:  $A = T \times V$

where  $V = \text{typical positioning speed}^{(1)}$

#### 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` 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 you can also read-out the present temperatures of galvanometer scanners and/or digital control boards (see `control_command`).

#### Example of Use

##### System Specification

- `Tracking error`  $T = 0.33 \text{ ms}$ .
- Calibration  $\pm 0.349 \text{ rad}_{\text{optical}}$   
( $= \pm 10^\circ_{\text{mechanical}}$ ) at  $\pm 503,316 \text{ bit}$ .
  - 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>(3)</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  $V/K = 28.8 \text{ m/s}$ .

(1) See technical specifications in the scan head manual.

(2) At the maximum frequency estimated with (1).

(3) See `*_ReadMe.txt` file of correction file or `get_table_para`.

### 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 (!)

- shall 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 wobbel frequency estimated by the rule of thumb (1), [page 219](#).
- The sum of the path (**Trajectory**) velocity and the velocity on the freely defined wobbel 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 wobbel figures, see safety notice on [page 219](#) 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 RTC5 PCI Board.

The second one requires the [Option "Second Scan Head Control", page 34](#).

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", page 55](#).

**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...4).
```

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

```
select_cor_table(HeadA, HeadB)  
with (HeadA and HeadB = 1...4).
```

(3) By [set\\_offset](#), [set\\_scale](#), [set\\_angle](#) or the corresponding list commands, specify offset, scaling 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 RTC5 PCI 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
- [load\\_program\\_file](#) deletes correction tables number 3 and 4.

## 8.5.2 3D Scan Systems

An RTC5 PCI Board can also be used to control a 3-axis scan system<sup>(1)</sup>. This requires the [Option "3D", page 34](#).

### 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.

### Connection and Initialization

In order to control a 3-axis scan system by an RTC5 PCI Board:

- The [Option "3D", page 34](#) of the RTC5 PCI 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 34](#), 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 54](#) and [Section "2D Correction Files 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 84](#).

### Notice!

- The standard [RTC5 DLL \(RTC5DLL.dll\)](#) supports all RTC5 commands for controlling 3-axis scan systems. However, the full functionality of the RTC5 commands – the actual *output* of z coordinates – is only available with enabled [Option "3D", page 34](#).

Several 3-axis scan systems can be simultaneously controlled (by multi-board commands) from a single PC. This requires a corresponding number of RTC5 PCI Boards with enabled [Option "3D"](#)s to be installed in that PC.

(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 23](#).

### 3D Commands

These are, for example, the following RTC5 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 128](#)
- The jump speed and mark speed are specified by `set_jump_speed` and `set_mark_speed`, see [Chapter 7.1.1 "Marking with Vector Commands and "Arc" Commands", page 124](#). 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 133](#)

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 Standard Mode**, the 3 coordinate values are always scaled up to 20-bit values internally, so that the 3 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 3 spatial directions are treated equally with regard to the jump speed 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 commands** 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 194](#). See also `set_defocus` or `set_defocus_list`.
- The size of the usable **Image 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 34** 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".



- By **set\_offset\_xyz** or **set\_offset\_xyz\_list**, an offset can be defined for the z coordinate. In addition, **set\_defocus** or **set\_defocus\_list** can be used for defining an offset to the calculated focal length, which then appropriately affect the z output value (in the direction opposite to the offset).
- During a marking process, the scan system focal length is continuously readjusted. During execution of a **Jump command** (or **goto\_xyz**) this readjustment can be switched off. With **DirectMove3D = 1** in **set\_delay\_mode** or **set\_delay\_mode\_list**, the z output value is directly (linearly) taken to its final value during the jump.

## Enhanced 3D Correction

- For more information on the actual procedure, refer to the "Calibrating a 3-Axis Laser Scan System" Manual, Chapter 5.4 "Step 4: Correcting Stretch".

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 RTC5 PCI 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 RTC5 PCI 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.

(1) Even of another type, see table 1, page 141.



### 8.5.3 Using Several Correction Tables

The RTC5 PCI Board memory can store 2 different correction tables at the same time. These remain after a `load_program_file` call. 2 more correction tables can be loaded into the data recording memory (then only half of the entries are available there, see `set_trigger`). These correction tables are deleted by `load_program_file`.

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 )` ( $n = 1 \dots \text{number\_of\_correction\_tables}$ ).

`number_of_correction_tables` serves

- to protect other commands (for example, `load_correction_file` and `select_cor_table`) from unwanted table numbers and
- to configure the memory organization, if necessary.

There is no need to change existing user programs except when user input is to be rejected in the future (by means of explicit RTC5 error messages).

## 8.6 Processing-on-the-fly

- "Processing-on-the-fly" means the combination of workpiece motion and scan system motion.
- 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 RTC5 PCI 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 RTC5 PCI 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 motion as encoder pulses, RTC5-internal encoder counters are triggered. The counter values correspond to the current position. They get a scaling factor (from certain RTC5 commands) assigned and are then used as *Processing-on-the-fly* correction.

See also [Chapter 9.3.3 "Synchronization by Encoder Signals"](#), page 274.

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/SPI Signals"](#), page 276.

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 215](#).

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 242](#).

### 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 Motions", page 228](#).
- `set_fly_rot`

To compensate rotary motions of the workpiece (for example, by rotating plate), see [Chapter 8.6.3 "Compensating Rotary Motions", 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 Motions", page 228](#).
- **set\_fly\_rot\_pos**  
To compensate rotary motions of the scan system (for example, rotary table), see [Chapter 8.6.3 "Compensating Rotary Motions", 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 Motions", page 228](#) and [Chapter 8.6.3 "Compensating Rotary Motions", 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 242](#)
- 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 Motions

Processing-on-the-fly correction for linear motions (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 Corrections", page 236](#).

### Correction via Encoder Counters

To be able to pass position values via the board-internal encoder counters<sup>(2)</sup>, 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>(3)</sup>

(1) Different Processing-on-the-fly corrections cannot be combined arbitrarily, see the [Section "Overview", page 227](#).

(2) See [Notice!](#), page 49.

(3) 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 motion.
- (3) Stop the motion.
- (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 has been activated by `simulate_encoder`, then the scaling factors are calculated as follows:

$$\text{Scalex} = \frac{K \times v_x}{(1,000,000 \text{ counts} / \text{s})}$$

$$\text{Scaley} = \frac{K \times v_y}{(1,000,000 \text{ counts} / \text{s})}$$

- (8) Adjust the signs of the scaling factors according to the motion 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 RTC5 PCI 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 "SPI / I2C Socket Connector", page 71**, there are available:

- For 1D corrections – signed 32-bit values
- For 2D corrections – signed 16-bit values 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 motion.
- (3) Stop the motion.
- (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 "SPI / I2C Socket Connector"**, page 71.

## 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 direction and y direction, but also in the z direction and with laser power variation. Furthermore, up to 4 additional signals can be transmitted for usage as you wish.

Each 10  $\mu$ s, 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 Compensating Rotary Motions

Before activating Processing-on-the-fly correction for rotary xy motion, 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 Corrections"**, 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 encoder counters<sup>(1)</sup>, encoder input port **ENCODER X**<sup>(2)</sup> must be connected, see also **Section "Input Ports for External Encoder Signals"**, page 275.

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>(3)</sup>

The output value is calculated from the current output position via a rotation matrix around the specified rotation center with rotation angle /  $360^\circ$  = current "Encoder0" counter reading / Resolution).

This correction is performed every  $10 \mu\text{s}$ .<sup>(4)</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>(5)</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 has been 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 motion.

(1) See **Notice!**, page 49.

(2) **ENCODER X** = Encoder0.

(3) 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.

(4) 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).

(5) Alternatively, the counter start and end values can be stored in a cache on the RTC5 PCI 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 motion 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 motions 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

- You cannot combine `set_fly_2d` correction 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 `set_fly_2d`-Processing-on-the-fly session with changed values, if they have been loaded before with the corresponding control commands. As of version DLL 541, OUT 541 this is also possible with `set_fly_x` and/or `set_fly_y`.

#### 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, you can use the `load_fly_2d_table` command to load a 2D compensation table onto the RTC5 (see below).

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 RTC5 PCI Board).

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 compensation table. This 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 table from an ASCII-readable text file gets loaded onto the RTC5 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 caption `[Fly2DTable<No>]`, whereby `<No>` corresponds to the number supplied with `load_fly_2d_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 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 141.

### 8.6.5 Deactivating Processing-on-the-fly Corrections

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 has been 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 215](#). 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 has been 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 (24-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 16 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 motion 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 RTC5 command or an external start signal, see [Chapter 6.4.4 "Starting and Stopping Lists", page 98](#).

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 start relative to the triggering start signal or RTC5 command, see [Section "External Start", page 266](#).

(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 266](#)). 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 execute several lists 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 scanners 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 Corrections", 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 24-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 ).

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 RTC5 PCI 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 RTC5 PCI 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

- The error bits **Bit #0...Bit #3** can be used to determine *which* edge of the **Image field** has been exceeded. Each boundary exceedance results in setting of the corresponding error bit.
- The error bits **Bit #0...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 #5a and #5b...#11, 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.
- The error bit **Bit #9** indicates if the 24-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 RTC5 PCI 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 ([Bit #4...Bit #7](#)). These can be read out by [get\\_marking\\_info](#).

Moreover, the following conditional commands allow execution of any list command to be made dependent upon whether boundary exceedance in a customer-defined monitoring area occurred or not:

- [if\\_fly\\_x\\_overflow](#)
- [if\\_fly\\_y\\_overflow](#)
- [if\\_not\\_fly\\_x\\_overflow](#)
- [if\\_not\\_fly\\_y\\_overflow](#)

If the condition specified in the command parameter to the error bits [Bit #4...Bit #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 bits [Bit #0...Bit #3](#)) only occurs if the maximum [Image field](#) ( $-524,288\dots+524,287$  bit) is exceeded (the customer-defined monitoring area is typically smaller).
- The error bits [Bit #4...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](#) or [clear\\_fly\\_overflow\\_ctrl](#) (see command description).
- The error bits [Bit #4...Bit #7](#) do *not* take into account transformations of type [#5a](#) and [#5b...#11](#), 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

Tracking errors of a scan system's galvanometer scanners can result in a certain amount of positioning inaccuracy, particularly for Processing-on-the-fly applications with variable encoder speeds. Accordingly, you can activate Tracking error compensation by `set_fly_tracking_error` for applications requiring higher precision.

### 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 axis and y axis 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 222.

### 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 has been 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 271](#). You can use `clear_fly_overflow` and `clear_fly_overflow_ctrl` 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`.

## 8.7 Pixel Output Mode

The **Vector commands** described in [Chapter 7.1 "Marking Dots, Lines and Arcs"](#), page 124 are intended for scanning vector based images. Furthermore, the RTC5 PCI 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 68](#). 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.

For controlling the laser, pulse lengths (digital) and voltage levels (analog) are outputted.

- 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**
  - Parameter **Channel** = ANALOG output port

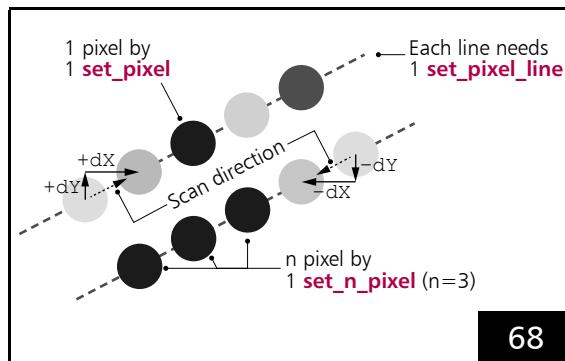
#### Notes

- 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** *should not* be used in conjunction with "Automatic Laser Control" (see [Chapter 7.4.9 "Automatic Laser Control"](#), page 187) if "Automatic Laser Control" readjusts the 12-bit output values at the **ANALOG OUT1** or **ANALOG OUT2** output port or pulse lengths (**PulseLength**) or output periods (**HalfPeriod**) of the laser signals **LASER1** and **LASER2**.
- The **Pixel Output Mode** *cannot* be combined with **Sky Writing**, see [Chapter 7.2.4 "Sky Writing"](#), page 149.
- The **Pixel Output Mode** *cannot* be combined with **Wobbel**, see [Chapter 8.4 "Wobbel Mode"](#), page 217.

### 8.7.2 RTC5 Commands

Before writing an image line, 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.



68

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**.

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.

The length of a pulse and a 12-bit-valued analog voltage value must be specified, see [Chapter 8.7.3 "Laser Control", page 246](#). Pixel pulses are always outputted at the LASER1 output port, even for the purpose of synchronizing the laser (gating).

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 RTC5 PCI 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 `n_set_pixel` (multi-board version of `set_pixel`).

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 278](#).

## Notes

- The number of pixels in an image line is limited only by the capacity of the RTC5 PCI Board list memory, see [Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734](#). 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 `set_pixel_line` or `set_pixel_line_3d`.
- The pixel distance (`dx, dy`) in the x direction and y direction (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  $1/64 \mu\text{s}$ . It is independent of the position output period of  $10 \mu\text{s}$ , which is used to move the galvanometer scanners of the scan system (split-up into `Microsteps`). 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  $10 \mu\text{s}$  raster with variable speed and/or curvilinear paths with the help of `micro_vector[*]` commands, see [Chapter 8.8 "micro\\_vector\[\\*\] Commands", page 249](#).

### 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”<sup>(1)</sup>

The “Classic Mode” corresponds to the **Pixel Output Mode** of the RTC4. Maximum pixel output frequency: 308 kHz.

- Variation Of Laser Pulse Length: **set\_pixel** defines – for each pixel – the length of the pixel pulse (in units of 1/64  $\mu$ s), which is outputted at the LASER1 port. For synchronization, see [Chapter 8.7.4 “Synchronization”, page 247](#).
- 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** or **ANALOG OUT2** output port (see above). For synchronization, see [Chapter 8.7.4 “Synchronization”, page 247](#). **set\_n\_pixel** defines the analog voltage level for directly successive pixels of an image line.

#### Notes

- 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: a periodic signal with **HalfPeriod** from **set\_pixel\_line** and a **PulseLength** from **set\_pixel**/**set\_n\_pixel** 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** and **ANALOG OUT2**.
- The LASER2 signal follows the specifications of the respective laser mode.
- No Softstart is performed, see [Chapter 7.4.7 “Softstart Mode”, page 184](#).
- **Pixel Output Mode** and **Sky Writing** cannot be combined.
- The values for **HalfPeriod** and **PulseLength** set prior to **set\_pixel\_line** are not restored again. **set\_laser\_pulses** must be explicitly called again.

(1) RTC6 supports additional modes.

#### 8.7.4 Synchronization

The pixel output timing diagram for one image line with 3 pixels is shown in [Figure 69](#).

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.<sup>(1)</sup> 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 69](#). 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 output 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 RTC5 board waits – especially at pixel output frequencies  $< 100 \text{ kHz}$  – until the default pixel is outputted.

The galvanometer scanners continue to run during this time and stop afterwards.<sup>(1)</sup>

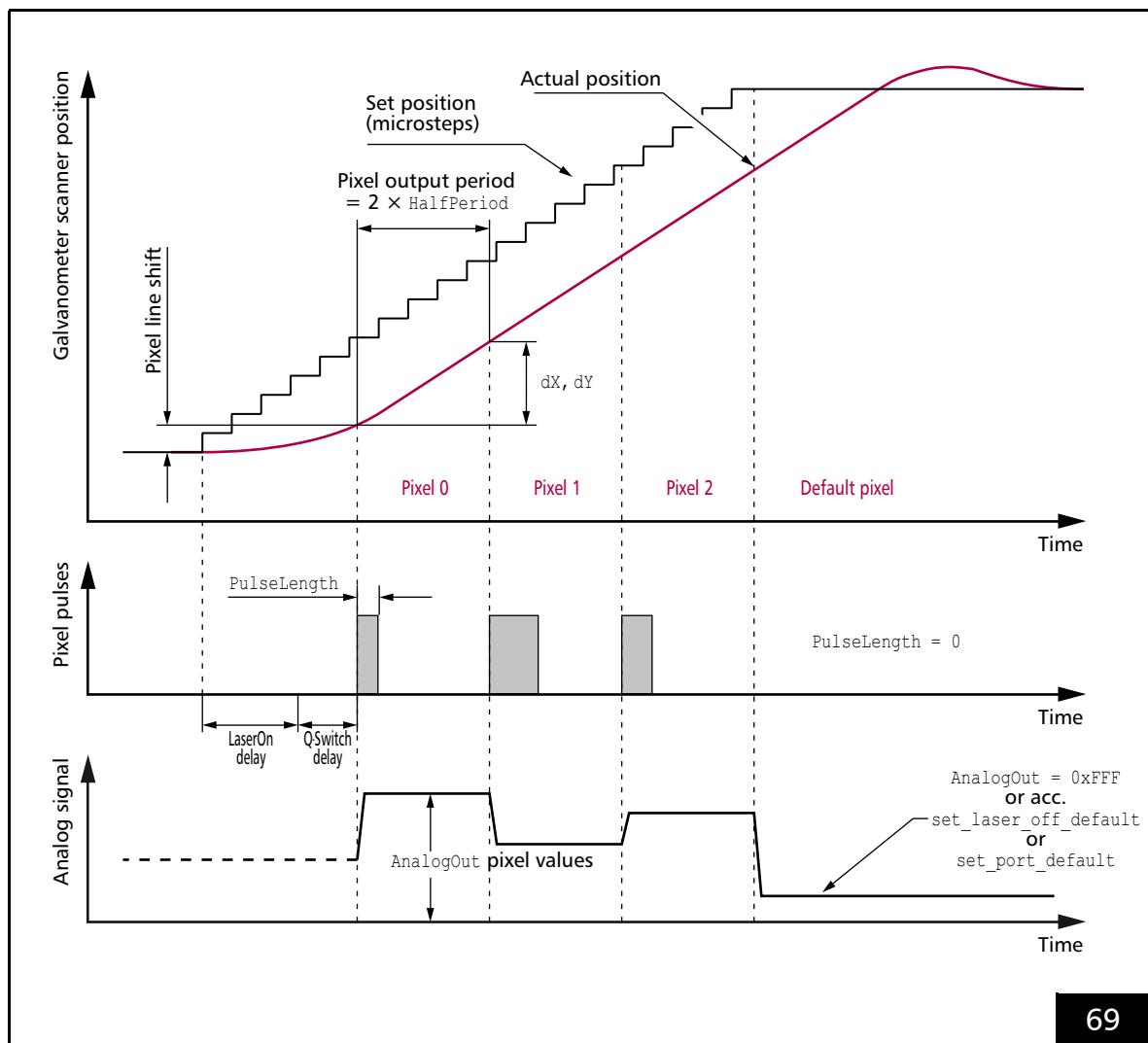
No scanner delay is automatically inserted.

The `Tracking error` and the hidden acceleration phase mean a pixel line shift", see [Figure 69](#). Normally, this needs to be compensated for by an adjusted run-in.

(1) The `set_pixel_line` parameter `Channel` + 256 can be used to suppress the stopping in the run-in phase and run-out phase. Then the galvanometer scanners move on (with the speed defined by `HalfPeriod` and pixel distance) in equidistant `Microsteps`. In this way, jerk-free run-in and run-out motions can be programmed.

## 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 (mark speed) and the properties of the laser system (power modulation).
- Output synchronization cannot be used together with **Pixel mode**. You should first deactivate output synchronization.



Timing of the galvanometer scanner positions and the laser control signals in the **Pixel mode** Mode = 0 and a YAG Mode, see [Chapter 7.4.4 "YAG Mode 1, 2, 3, 5"](#), page 179.

The pixel output period in this example is approx. 4.5 Microsteps.

## 8.8 `micro_vector[*]` Commands

`micro_vector[*]` 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 `micro_vector[*]` 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 `micro_vector[*]` commands can be used to mark trajectories (see Glossary entry on [page 25](#)) of arbitrary shape and at variable speed, as long as the dynamic limits of the scan system are not exceeded.

### Notes

- `micro_vector[*]` commands wait for a preceding scanner delay. They never trigger new `Scanner Delays`.
- Users themselves are responsible for appropriately parameterizing `micro_vector[*]` 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 143](#)
- A Wobbel motion specified by `set_wobbel` or `set_wobbel_mode`, see [Chapter 8.4 "Wobbel Mode", page 217](#) is not taken into account (but also not deactivated).
- The `Sky Writing` mode 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 Commands

“Normal” **Vector 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  $\mu$ s (the output period of the **Microsteps**) and in the range from 10  $\mu$ s to 167,772,160  $\mu$ s ( $\approx$  2.8 min).

A vector or arc is split-up into the specified ( $T$  Parameter) number of **Microsteps**.

For  $T \geq 5$ , the following applies:

$$\text{Length } \Delta s \text{ of a Microstep} = L / t \\ \text{with } t = \text{integer}((T + 5) / 10)^{(2)}$$

Thus, jump speed and mark speed speed automatically depend on the length of the vectors or 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 135**.

(1) Either jump speed or mark speed.

(2) For  $T < 5$ , the command is carried out as non-timed command, see **Chapter 7.1.2 “Microstepping”, page 128**.

## 8.10 Automatic Self-Calibration

- This functionality does not apply to current SCANLAB scan systems
- Previously, some SCANLAB scan systems have been equipped with an additional internal sensor system for automatic self-calibration (ASC sensor system, home-In sensors) to meet higher requirements for long-term repeatability

### 8.10.1 Using 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 may have been equipped with an additional internal sensor system for automatic self-calibration (ASC sensor system, Home-In sensors).

The ASC sensor system (see above) 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 253](#)
- 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 253](#)

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 #11 in [Chapter 7.3.6 "Output Values to the Scan System", page 170](#).

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 254](#).

## 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 #11](#).  
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 set as to-be-returned by 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 191](#).  
The "Automatic Laser Control" must be deactivated before performing automatic self-calibration.
  - `auto_cal` also aborts (result value 1, 10 or 11), if the scaling factor has been set by `control_command( Data = 12xxH )` to a value < 1. This is because the sensor positions then are not reachable, see also `control_command( Data = 053FH )`. The scaling factor must have been set to 1 (default setting) by prior to automatic self-calibration
    - `control_command( Data = 1283H )` and
    - `control_command( Data = 1200H )`.

(1) See Glossary entry on [page 23](#).

### 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 254](#). The reference values are stored in the EEPROM. 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 RTC5 PCI 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.

### 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 253](#)) 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 or 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 70](#).

The galvanometer scanner motion here is a lever motion 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 = Round((EncoderCurrent - EncoderStart) × Scale)`. `FirstPos` is the first **mark\_abs** command’s list position and `Scale` is a freely selectable scaling factor. `EncoderStart` gets automatically determined when **camming** is called. There is no automatic encoder reset. The first outputted point is always `Index = 0`.

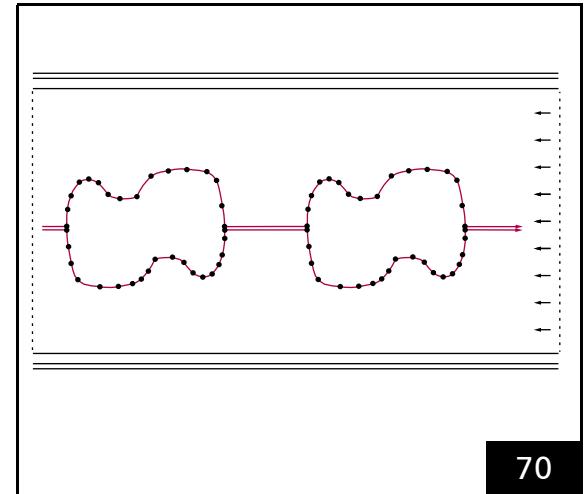
The individual points are executed every 10  $\mu$ s as a “Hard jump” without **Scanner Delays**.

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  $\mu$ s clock cycle 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.



70

**Camming** process 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 RTC5 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 cycle.

#### 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 94](#)), then the curve shape can be tested using a normal execution 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  $\mu$ s) can be used as well, but not `timed_para_mark_abs_3d`.

## 8.12 Time Measurements

### 8.12.1 RTC5 Timer

RTC5 boards are equipped with an integrated timer, which is referred to as the “RTC5 Timer” in the following. The **RTC5 Timer** only counts clock cycles of control 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 **RTC5 Timer** value and resets it to 0. The elapsed time can then be read by **get\_time**, which returns the **RTC5 Timer** value saved during the most recent call of **save\_and\_restart\_timer**.

The present **RTC5 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 **RTC5 Timer** to zero. In this way the interim execution time of lengthy marking processes can be monitored.

### 8.12.2 Timestamps

#### 32-bit “Timestamp Counter”

As of DLL 543, OUT 543 a

**32-bit “Timestamp Counter”** is additionally available. It starts counting at 0 with **load\_program\_file** and counts uninterruptible all  $10 \mu\text{s}$  clock periods.

Using the **32-bit “Timestamp Counter”**, an absolute reference to the individual time periods can be established, even if the **RTC5 Timer** has been reset by **save\_and\_restart\_timer**. The **32-bit “Timestamp Counter”** can be recorded by the trigger signal 52 (see **set\_trigger** or **set\_trigger4**) and is read-out by **get\_value(52)**.

#### Notes

- **get\_time** and **get\_lap\_time** only take list execution times into account (because the **RTC5 Timer** only counts list command clock cycles and pauses during interruptions by **set\_wait** or **pause\_list**).
- To compare RTC5-internal **save\_and\_restart\_timer** time measurements to external time measurements by the BUSY pin, you should insert a **list\_nop** between **save\_and\_restart\_timer** and **set\_end\_of\_list**. This ensures that any 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 RTC5 PCI Board provides a variety of peripheral interfaces, see [Chapter 4.6 "Interfaces for the Laser and Peripheral Equipment", page 60](#).

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 Port and 16-Bit Digital Output Port", page 65](#)).

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 state until an initial value is assigned to it. In addition, `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:

- With position-dependent and speed-dependent laser control (see `set_port_default`)
- Upon terminating **Pixel mode**

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 187](#). You can log this by `set_trigger/set_trigger4` (signal 24).

When the output value is outputted, 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 to DATA7) (see [Section "8-Bit Digital Output Port", page 68](#)).

Its output values can be set by `write_8bit_port` or `write_8bit_port_list`. The output is in high-impedance state until an initial value is assigned to it. In addition, the commands `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:

- With position-dependent and speed-dependent laser control (see `set_port_default`)
- Upon terminating `Pixel mode`

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 187. This can be recorded by `set_trigger/set_trigger4` (signal 24).

When the output value is outputted, 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 68](#)).

### 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 61](#).

By `set_laser_pin_out` or `set_laser_pin_out_list`, values are assigned to this output port.

The value is 0 (pins are LOW) until an initial value is assigned to it.

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 Port 1 and 2

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 61.

The `ANALOG OUT2` output port is also available by the MARKING ON THE FLY socket connector, see [Chapter 4.6.4 "MARKING ON THE FLY Socket Connector"](#), page 69.

The output values of the output ports can be separately set by `write_da_x` or `write_da_x_list`. The values are 1 (corresponds to approx. 0 V) until initial values are assigned to it.

In addition, the commands `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-dependent 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"](#), page 184
- In `Pixel Output Mode`, see [Chapter 8.7 "Pixel Output Mode"](#), page 244

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 187. 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<sup>(1)</sup>:

- 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 RTC5 generates periodic CLOCK signal pulses (during reference motions by `stepper_init` and set-position motions 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  $\mu$ s cycles).
- You can explicitly set the DIRECTION signal (and thereby the direction of stepper motor motion) during reference motions by `stepper_init`. In contrast, the DIRECTION signal is internally controlled during set-position motions 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 cycles 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 75](#).
- For querying signals, see [Section "Querying Signals and Status Values", page 262](#).
- 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 the stepper motor 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.

(1) Stepper motor signals are available only for RTC5 Boards with DSP version numbers 2 and higher (see `get_rtc_version Bit #16...Bit #23`). For older RTC5 Boards, stepper motor commands do not execute.

## Reference Motions and Position Initialization

With **stepper\_init**, you can initiate reference motions 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 motion with a short *CLOCK* pulse period `Period` first and then a further (shorter but slower) reference motion with a longer *CLOCK* pulse period.

Once the reference motion 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 motions. The reached reference position is offset from the limit switch position by  $\pm \text{Tol}$  (direction dependent).

During reference motion, the status is "Init" (Bit #5 = 1), see [Section "Querying Signals and Status Values", page 262](#). 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 motion, see [Section "Set-Position Motions", page 261](#), 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 4 times.

If `Period = 0` and/or `Dir < 0`, then the reference motion 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 motion, you could also use **stepper\_init** as an emergency stop for the stepper motor.

Parallel execution of reference motions for both stepper motors is also possible, but cannot be simultaneously started through a single command.

## Set-Position Motions

Set-position motions can be initiated by **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 motions 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 motion is completed.

The list commands for set-position motions are short list commands. Therefore, a stepper motor motion can also execute synchronously with a galvanometer scanner motion.

If the limit switch activates during a set-position motion, then the motion immediately aborts and cannot be resumed as a normal set-position motion. You either have to request a reference motion or mechanically or via the software, see below, deactivate the limit switch. If a stepper motor motion aborts once (for example, also by `Period = 0`), then the existing set position gets overwritten by the current position value. Therefore the stepper motor motion 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 motion 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 motion 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 motions and the Busy status during set-position motions. 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 `RTC5_BUSY` and list commands wait until the Init status gets reset. In some circumstances, the list itself or the motion process must be aborted.

## Terminating Infinite Motions

Depending on chosen parameters, very long or even infinite stepper motor motions 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 motion 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 motion 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 motion 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 motion 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 motion 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 motion 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 70](#).

For configuring the RS-232 interface, see [Chapter 4.6.5 "RS232 Socket Connector", page 70](#).

`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 "SPI / I2C Socket Connector", page 71](#), a 32-bit data word every 10  $\mu$ s 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  $\mu$ s. 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  $\mu$ s clock cycle, 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 "SPI / I2C Socket Connector", page 71](#).
- In the default setting, the [McBSP interface](#) always outputs Bit #4...Bit #19 of the Cartesian control values for the x axis and y axis (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 – by 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 Port and 16-Bit Digital Output Port", page 65](#)) for receiving 16-bit digital values. For synchronization of data transmission, the EXTENSION 1 socket connector also provides 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 271](#)).

### 9.2.2 2-Bit Digital Input Port

For querying 2-bit digital values, the LASER connector provides a 2-bit digital input port (DIGITAL IN1 and DIGITAL IN2, see [Section "2-Bit Digital Input Port", page 61](#)).

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 271](#)).

### 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 70](#).

For configuring the RS-232 interface, see [Chapter 4.6.5 "RS232 Socket Connector", page 70](#).

Data can be read in by `rs232_read_data`.

### 9.2.4 McBSP Interface

At the [McBSP interface](#), see [Chapter 4.6.6 "SPI / I2C Socket Connector", page 71](#), the 32-bit data word most recently fully transmitted to the specified memory location can be queried with `read_mcbsp`. The interpretation as one 32 bit data word or two 16 bit data words is the responsibility of the user.

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/SPI Signals", page 276](#)) or can be used for an [Online Positioning](#), see [Chapter 8.3.1 ""Local Online Positioning""](#), page 213 and [Chapter 8.3.2 ""Global Online Positioning""](#), page 216.

#### Notes

- For signals and operating conditions, see [Chapter 4.6.6 "SPI / I2C Socket Connector", page 71](#).

## 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 in command lists at appropriate locations.
- 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 96](#) and the list execution status by [get\\_status](#), see [Chapter 6.4.3 "List Execution Status", page 97](#).
- In addition, the **BUSY** list execution status **List Execution Status** is provided by the **BUSY OUT** signal:
  - at the **LASER** connector, [see Section "BUSY List Execution Status", page 61](#)
  - at the **EXTENSION 1** socket connector, [see Section "BUSY List Execution Status", page 66](#)
  - at the **MARKING ON THE FLY** socket connector, [see Section "BUSY list execution status Status", page 69](#)

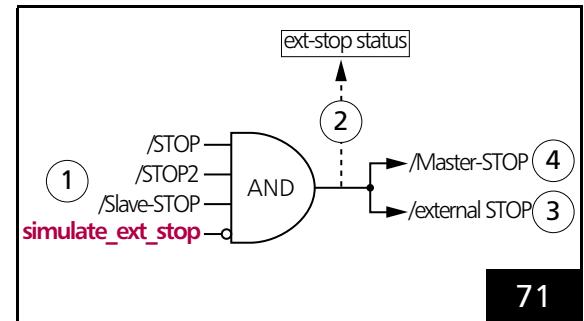
Moreover, the RTC5 PCI Board provides commands and interfaces (described in the following sections) that allow external control signals (for example, from a light-barrier or from an encoder) to directly control and synchronize execution of command lists or individual commands (including the output of peripheral signals).

Moreover, the RTC5 PCI 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 71](#).



71

**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 61](#), the /STOP2 input at the MARKING ON THE FLY socket connector, see [Section "External Control Signals", page 69](#). 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 RTC5 PCI Board is connected to other boards by the Master or Slave connector, see [Figure 7](#), then external stops from Master to Slave. See also [Chapter 6.6.3 "Master/Slave Operation", page 113](#).

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 **simulate\_ext\_start** or **simulate\_ext\_start\_ctrl**, an **External Start** can be initiated (see (1), (5) and (7) in [Figure 72](#)).

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** list execution status (as during list execution) nor the

**INTERNAL-BUSY** list execution status (as for example, with **goto\_xy**) nor the **PAUSED** list execution 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).

As of version DLL 543, OUT 543, 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 61](#), the /START2 input at the MARKING ON THE FLY socket connector, see [Section "External Control Signals", page 69](#). 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 RTC5 PCI Board is connected to other boards by the Master or Slave connector, see [Figure 7](#), then external starts are passed on from Master to Slave. See also [Chapter 6.6.3 "Master/Slave Operation", page 113](#).

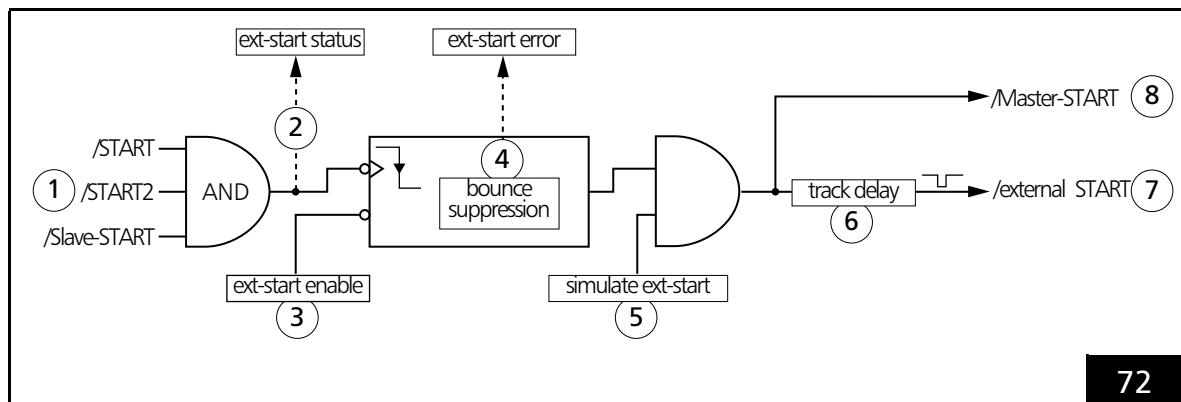
Internal starts triggered by **execute\_list** or **execute\_at\_pointer** are *not* passed on. In contrast, External Starts triggered by **simulate\_ext\_start** or **simulate\_ext\_start\_ctrl** are passed on, see also [Chapter 6.6.3 "Master/Slave Operation", page 113](#).

**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, for instance, 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, `set_ext_start_delay`, `set_ext_start_delay_list` or `simulate_ext_start` allow configuring a track delay (see (6) in [Figure 72](#)) 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 274](#).

**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

- The `/START`, `/START2` and `/Slave-START` inputs are *edge* sensitive (HIGH to LOW level *transition*).
- The `/STOP`, `/STOP2` and `/Slave-STOP` inputs are *level* sensitive.
- A `stop_execution` call 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` list execution 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 94](#).
- 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 has been 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 point in time 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 point in time of the **External Start** – exactly synchronized to the 10  $\mu$ s clock of the RTC5 PCI 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 point in time 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 73](#). 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
```

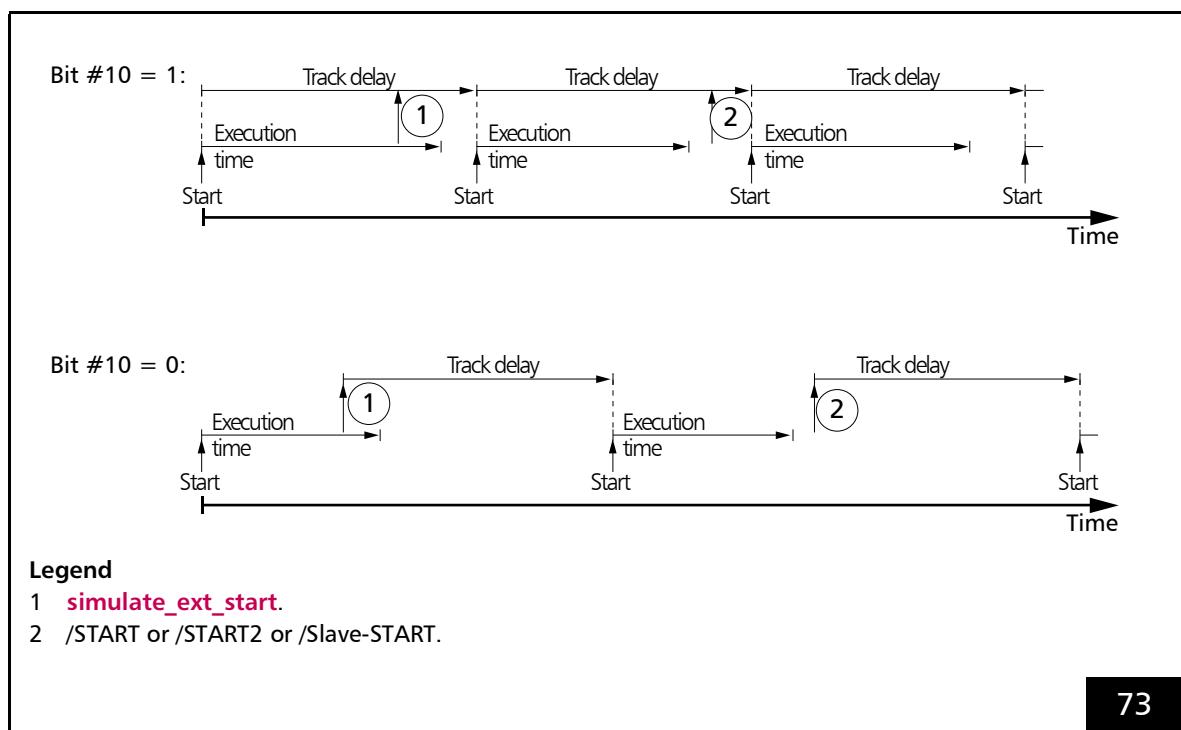
If the first start is to be triggered externally (for example, by /START or by `simulate_ext_start_ctrl`) rather than by an `execute_list_pos` command, but all subsequent starts triggered by `simulate_ext_start`, then `set_control_mode_list` in the above example must be called before `simulate_ext_start`.

After setting `set_control_mode( Bit #1 = 1 )`, the external start queue 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` calls.

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 Port and 16-Bit Digital Output Port", page 65](#), 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 Port and 16-Bit Digital Output Port", page 65](#), 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.

As of version DLL 543, OUT 543, 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 cycle. A conditional `pause_list` (as of DLL 544, OUT 544) 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 met.



### 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 (that is, Bit #0 of the digital input turns HIGH)
list_jump_rel_cond(0, 1, 0);
// clear Bit #0 of the 16-bit output
clear_io_cond_list(0, 0, 1);
// loop until the signal is confirmed
list_jump_rel_cond(1, 0, 0);
...
set_end_of_list;
execute_list(1);
```

#### (2) If the lower 4 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 RTC5 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 RTC5 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 RTC5 PCI Board 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 motion) to the RTC5 PCI Board which can result in incrementing or decrementing of its two internal encoder counters<sup>(2)</sup>. The states of the RTC5 encoder counters thereby correspond directly to the position of the workpiece<sup>(3)</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 by the list command **store\_encoder** and then retrieved from there by the control command **read\_encoder**.

In addition, the RTC5 PCI Board 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 commands** 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 motions, **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 start relative to the triggering input signal or corresponding command, see [Section "External Start", page 266](#).
- 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 RTC5 PCI Board (see [Chapter 9.3.4 "Synchronization and Online Positioning by McBSP/SPI Signals", page 276](#)).

(2) See [Notice!](#), page 49.

(3) 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.

## Input Ports for External Encoder Signals

For receiving encoder signals, the MARKING ON THE FLY socket connector provides 2 encoder input ports, see [Section "Encoder Input Ports", page 69](#).

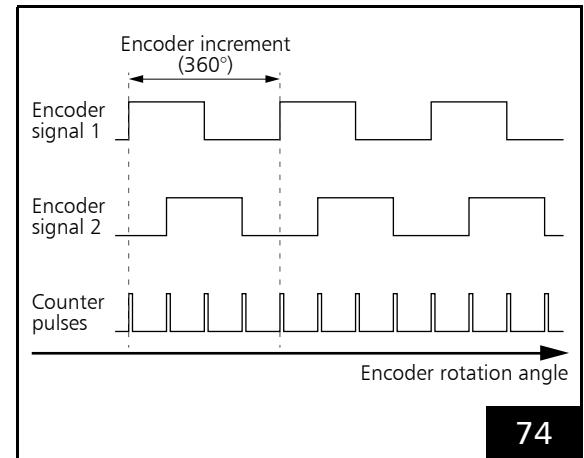
- **ENCODER X**
- **ENCODER Y**

For linear motions 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 direction and y direction) can be connected to these inputs.

For rotational motions only *one* incremental encoder is necessary. For Processing-on-the-fly applications, it must be connected to the encoder input port **ENCODER X**.

**ENCODER X** and **ENCODER Y** are each 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 74](#). The second encoder signal is usually phase-shifted by  $90^\circ$  relative to the first signal. The corresponding RTC5-internal encoder counter<sup>(1)</sup> is triggered at each edge of both signals, that is, one encoder increment results in 4 counter pulses (counts). The relative  $90^\circ$  phase-shift of the two signals allows the RTC5 PCI Board to detect the motion direction of the workpiece as well. Depending on the direction of motion, the counter value is increased or decreased.



74

Timing diagram of a typical encoder signal pair and of the corresponding RTC5 counter pulses.

The signals at encoder input port:

- **ENCODER X** trigger encoder counter "Encoder0"
- **ENCODER Y** trigger encoder counter "Encoder1"

The encoder signals 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 RTC5 PCI Board provides a 1 MHz clock signal (counter pulses), which replaces the signals of an external incremental encoder.

(1) See [Notice!](#), page 49.

### 9.3.4 Synchronization and Online Positioning by McBSP/SPI Signals

Instead of incremental encoders, the motion 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 RTC5 PCI 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 commands](#) 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 213 and [Chapter 8.3.2 ""Global Online Positioning""](#), page 216), 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 61](#)
- ANALOG OUT2 output port, see [Section "12-Bit Analog Output Port 1 and 2", page 61](#)
- 8-bit digital output port, see [Section "8-Bit Digital Output Port", page 68](#)
- 16-bit digital output port, see [Section "16-Bit Digital Input Port and 16-Bit Digital Output Port", page 65](#)
- 2-bit digital output port, see [Section "2-Bit Digital Output Port", page 61](#)

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**.
- As of version DLL 543, OUT 543, **periodic\_toggle** and **periodic\_toggle\_list** toggle endless with Count =  $2^{32}-1$ .

## 10 RTC5 Commands

### 10.1 Overview

The following pages describe the complete RTC5 command set (control commands and list commands). The commands are listed according to their intended use. The page numbers refer to [Chapter 10.2 "RTC5 Command Set", page 290](#), where the commands (alphabetically) are explained in detail.

#### 10.1.1 Designations

##### Multi-Board Commands

All commands marked (n\_) in the following list also exist in a version for multiple RTC5 Boards installed in one computer, see also [Chapter 6.6 "Using Several RTC5 PCI Boards in One PC", page 112](#).

An associated multi-board command is also provided for almost all commands. Exceptions are explicitly noted in the description list (in [Chapter 10.2 "RTC5 Command Set", page 290](#)).

##### Normal, Short, Variable and Multiple List Commands

The list commands of the RTC5 command set vary somewhat in their length of command execution. To differentiate, list commands in the list description (in [Chapter 10.2 "RTC5 Command Set", page 290](#)) are designated as "normal", "short", "variable" and "multi".

- Normal list commands require a full 10 µs clock cycle 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 cycle, during which control commands cannot execute. In contrast, a short list command that immediately follows a normal list command executes in the subsequent 10 µs clock cycle.

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 vectors (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 cycle), 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 cycle. Insertion of [list\\_nop](#) (but not insertion of [list\\_continue](#)) results in the interruption of the [Polyline](#) ([Signals for "Laser Active" Operation](#) are switched off). Currently, up to 12 short list commands per 10 µs clock cycle are possible. However, the maximum number can be lower, depending on the workload of the RTC5 Board and the [DSP](#) version<sup>(1)</sup>. 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 cycle 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 cycle 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 cycle.

(1) On older RTC5 Boards where [DSP](#) version < 2 ([get\\_RTC\\_version](#) Bit #16...Bit #23), only up to 8 short list commands per 10 µs clock cycle are possible.

The maximum number of currently up to 12 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 [page 280](#).

#### 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  $10 \mu\text{s}$  (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 and Delayed Short List Commands

Most short list commands (for example, **list\_jump\_pos**, **list\_call**, **sub\_call** or **list\_return**) (with all software versions) execute before the next list command and prior to a possible scanner delay (**Jump Delay**, **Mark Delay**, **Polygon Delay**). These commands are called "undelayed short list commands" in the corresponding command descriptions (in [Chapter 10.2 "RTC5 Command Set", page 290](#)).

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**.

**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  $\mu$ s clock cycle.

### Multiple List Commands

A multiple list command has multiple components that accordingly occupy multiple list memory area 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 RTC5 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 RTC5 users who previously used the RTC4, the command descriptions (in [Chapter 10.2 “RTC5 Command Set”, page 290](#)) note in each command’s “RTC4→RTC5” section:

- to what extent a command differs from that of the RTC4,
- whether the command is new to the command set,
- whether and how parameter values are converted in [RTC4 Compatibility Mode](#).

Some RTC3/RTC4 commands and a few RTC2 commands emulated by the RTC3/RTC4 are not supported by the RTC5. These commands are listed in [Chapter 10.3 “Unsupported RTC2/RTC3/RTC4 Commands”, page 723](#).

For general information about migrating from the RTC4 to the RTC5 and about the [RTC4 Compatibility Mode](#), see [Chapter 2.10 “Notes for RTC4 Users”, page 40](#).



### 10.1.3 Version Information

Descriptions for a number of commands include version information, listed under “[Version info](#)”:

- For newly added commands: the software or [DSP](#) version in which they become available
- For some older commands: information on implemented changes.

New commands as well as all changes to existing commands are described in

[RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).

### 10.1.4 Optional Functions

If an option is not present on the RTC5 PCI Board, see [Chapter 2.6 “Options”, page 34](#), then the associated commands have only partial or non-existent effect:

- Without [Option Processing-on-the-fly](#), commands to activate a Processing-on-the-fly correction have no effect
- Without [Option “3D”](#), 3D vector commands are executed, however, no z axis signals are outputted
- Without [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 command descriptions contain corresponding notes.

## 10.1.5 Control Commands

### DLL Initialization

free_RTC5_dll .....	325
get_RTC_mode .....	352
init_RTC5_dll .....	386
set_RTC4_mode .....	612
set_RTC5_mode .....	613

### Using Several RTC5 PCI Boards in One PC

acquire_RTC .....	291
release_RTC .....	500
rtc5_count_cards .....	507
select_RTC .....	516

### List Memory Commands

(n_) config_list .....	313
(n_) get_config_list .....	328
(n_) get_list_space .....	346
(n_) load_disk .....	420
(n_) save_disk .....	509

### Board Initialization and Image Field Correction

(n_) get_head_para .....	337
(n_) get_sync_status .....	362
(n_) get_table_para .....	363
(n_) load_correction_file .....	416
(n_) load_program_file .....	430
(n_) load_stretch_table <sup>(1)</sup> .....	433
(n_) load_z_table <sup>(1)</sup> .....	439
(n_) number_of_correction_tables .....	468
read_abc_from_file .....	489
(n_) select_cor_table .....	512
(n_) set_dsp_mode .....	536
write_abc_to_file .....	715

### Laser Mode and Parameters

(n_) config_laser_signals .....	311
(n_) get_standby .....	355
(n_) load_auto_laser_control .....	413
(n_) load_position_control .....	428
(n_) set_auto_laser_control .....	522
(n_) set_auto_laser_params .....	525
(n_) set_encoder_speed_ctrl .....	539
(n_) set_firstpulse_killer .....	544
(n_) set_laser_control .....	566

(n_) set_laser_mode .....	570
(n_) set_laser_pulses_ctrl .....	573
(n_) set_pulse_picking .....	609
(n_) set_pulse_picking_length .....	609
(n_) set_qswitch_delay .....	610
(n_) set_softstart_level .....	624
(n_) set_softstart_mode .....	626
(n_) set_standby .....	628

### Setting the Scanner Parameters

(n_) load_varpolydelay .....	437
(n_) set_delay_mode .....	534
(n_) set_jump_speed_ctrl .....	564
(n_) set_mark_speed_ctrl .....	576
(n_) set_sky_wrting .....	617
(n_) set_sky_wrting_limit .....	618
(n_) set_sky_wrting_mode .....	619
(n_) set_sky_wrting_para .....	621

### Coordinate Transformations

(n_) set_angle .....	520
(n_) set_defocus <sup>(1)</sup> .....	531
(n_) set_defocus_offset <sup>(1)</sup> .....	532
(n_) set_matrix .....	577
(n_) set_offset .....	598
(n_) set_offset_xyz <sup>(2)</sup> .....	599
(n_) set_scale .....	614

### Online Positioning

(n_) apply_mcbsp .....	295
(n_) set_mcbsp_global_matrix .....	581
(n_) set_mcbsp_global_rot .....	582
(n_) set_mcbsp_global_x .....	583
(n_) set_mcbsp_global_y .....	584
(n_) set_mcbsp_matrix .....	587
(n_) set_mcbsp_rot .....	591
(n_) set_mcbsp_x .....	592
(n_) set_mcbsp_y .....	593

(1) Only with Option "3D".

(2) Limited functionality if no Option "3D".

<b>Status Monitoring and Diagnostics</b>		
(n_) <code>get_head_status</code>	338	
(n_) <code>get_overrun</code>	352	
(n_) <code>get_value</code>	368	
(n_) <code>get_values</code>	370	
(n_) <code>get_waveform</code>	372	
(n_) <code>measurement_status</code>	463	
(n_) <code>stop_trigger</code>	674	
<b>iDRIVE Commands</b>		
(n_) <code>control_command</code>	315	
(n_) <code>get_transform</code>	365	
(n_) <code>read_user_data</code>	499	
(n_) <code>send_user_data</code>	519	
<code>transform</code>	703	
(n_) <code>upload_transform</code>	706	
<b>Pixel Output Mode</b>		
(n_) <code>set_default_pixel</code>	530	
<b>I/O Commands</b>		
(n_) <code>get_free_variable</code>	334	
(n_) <code>get_io_status</code>	342	
(n_) <code>get_mcbsp</code>	351	
(n_) <code>mcbsp_init</code>	461	
(n_) <code>mcbsp_init_spi</code>	462	
(n_) <code>periodic_toggle</code>	484	
(n_) <code>read_analog_in</code> <sup>(1)</sup>	490	
(n_) <code>read_io_port</code>	492	
(n_) <code>read_io_port_buffer</code>	493	
(n_) <code>read_mcbsp</code>	495	
(n_) <code>read_multi_mcbsp</code>	496	
(n_) <code>rs232_config</code>	503	
(n_) <code>rs232_read_data</code>	504	
(n_) <code>rs232_write_data</code>	505	
(n_) <code>rs232_write_text</code>	505	
(n_) <code>set_free_variable</code>	556	
(n_) <code>set_laser_off_default</code>	570	
(n_) <code>set_mcbsp_freq</code>	580	
(n_) <code>set_mcbsp_out_ptr</code>	589	
(n_) <code>set_port_default</code>	607	
(n_) <code>write_8bit_port</code>	714	
(n_) <code>write_da_1</code>	716	
(n_) <code>write_da_2</code>	717	
(n_) <code>write_da_x</code>	718	
(n_) <code>write_io_port</code>	721	
(n_) <code>write_io_port_mask</code>	722	
<b>Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization</b>		
(n_) <code>get_counts</code>	328	
(n_) <code>get_master_slave</code>	350	
(n_) <code>get_startstop_info</code>	356	
(n_) <code>set_control_mode</code>	528	
(n_) <code>set_extstartpos</code>	541	
(n_) <code>set_max_counts</code>	580	
(n_) <code>simulate_ext_start_ctrl</code>	659	
(n_) <code>sync_slaves</code>	680	
<b>List Handling and Status</b>		
(n_) <code>auto_change</code>	304	
(n_) <code>auto_change_pos</code>	305	
(n_) <code>get_lap_time</code>	343	
(n_) <code>get_status</code>	358	
(n_) <code>get_wait_status</code>	371	
(n_) <code>pause_list</code>	483	
(n_) <code>quit_loop</code>	486	
(n_) <code>read_status</code>	497	
(n_) <code>release_wait</code>	501	
(n_) <code>restart_list</code>	503	
(n_) <code>set_pause_list_cond</code>	601	
(n_) <code>set_pause_list_not_cond</code>	602	
(n_) <code>start_loop</code>	661	
(n_) <code>stop_execution</code>	673	
(n_) <code>stop_list</code>	673	
<b>Input Pointer Commands</b>		
(n_) <code>get_input_pointer</code>	342	
(n_) <code>get_list_pointer</code>	345	
(n_) <code>load_list</code>	426	
(n_) <code>set_input_pointer</code>	558	
(n_) <code>set_start_list</code>	630	
(n_) <code>set_start_list_1</code>	630	
(n_) <code>set_start_list_2</code>	630	
(n_) <code>set_start_list_pos</code>	631	
<b>Output Pointer Commands</b>		
(n_) <code>execute_at_pointer</code>	320	
(n_) <code>execute_list</code>	320	
(n_) <code>execute_list_1</code>	321	
(n_) <code>execute_list_2</code>	321	
(n_) <code>execute_list_pos</code>	321	
(n_) <code>get_out_pointer</code>	351	

(1) Only with RTC5 PCIe Board or with RTC5 PCI Board and installed ADC Add-On Board.

<b>Subroutine Commands</b>		<b>Processing-on-the-fly Commands</b>	
(n_) <code>copy_dst_src</code> .....	317	(n_) <code>clear_fly_overflow_ctrl</code> .....	309
(n_) <code>get_char_pointer</code> .....	327	(n_) <code>get_encoder</code> .....	329
(n_) <code>get_sub_pointer</code> .....	361	(n_) <code>get_fly_2d_offset</code> .....	334
(n_) <code>get_text_table_pointer</code> .....	364	(n_) <code>get_marking_info</code> .....	347
(n_) <code>load_char</code> .....	415	(n_) <code>init_fly_2d</code> .....	385
(n_) <code>load_sub</code> .....	435	(n_) <code>load_fly_2d_table</code> .....	422
(n_) <code>load_text_table</code> .....	436	(n_) <code>read_encoder</code> .....	491
(n_) <code>set_char_pointer</code> .....	526	(n_) <code>set_ext_start_delay</code> .....	542
(n_) <code>set_char_table</code> .....	527	(n_) <code>set_fly_tracking_error</code> .....	550
(n_) <code>set_sub_pointer</code> .....	632	(n_) <code>set_mcbsp_in</code> .....	585
(n_) <code>set_text_table_pointer</code> .....	633	(n_) <code>set_multi_mcbsp_in</code> .....	594
<b>Direct Laser and Scan Head Control</b>		<b>Controlling Stepper Motors</b>	
(n_) <code>disable_laser</code> .....	318	(n_) <code>get stepper_status</code> .....	360
(n_) <code>enable_laser</code> .....	319	(n_) <code>stepper_abs</code> .....	662
(n_) <code>get_laser_pin_in</code> .....	344	(n_) <code>stepper_abs_no</code> .....	663
(n_) <code>get_z_distance</code> <sup>(1)</sup> .....	373	(n_) <code>stepper_control</code> .....	665
(n_) <code>goto_xy</code> .....	374	(n_) <code>stepper_enable</code> .....	667
(n_) <code>goto_xyz</code> <sup>(2)</sup> .....	375	(n_) <code>stepper_disable_switch</code> .....	666
(n_) <code>laser_signal_off</code> .....	395	(n_) <code>stepper_init</code> .....	668
(n_) <code>laser_signal_on</code> .....	396	(n_) <code>stepper_rel</code> .....	670
(n_) <code>set_laser_pin_out</code> .....	571	(n_) <code>stepper_rel_no</code> .....	671
<b>Version Commands</b>		<b>Jump Mode</b>	
<code>get_dll_version</code> .....	329	(n_) <code>get_jump_table</code> .....	343
(n_) <code>get_hex_version</code> .....	340	(n_) <code>load_jump_table</code> .....	423
(n_) <code>get_rtc_version</code> .....	353	(n_) <code>load_jump_table_offset</code> .....	424
(n_) <code>get_serial_number</code> .....	354	(n_) <code>set_jump_mode</code> .....	560
<b>Error Commands</b>		(n_) <code>set_jump_table</code> .....	565
(n_) <code>get_error</code> .....	330	<b>Other Control Commands</b>	
(n_) <code>get_last_error</code> .....	344	(n_) <code>auto_cal</code> .....	301
(n_) <code>reset_error</code> .....	502	(n_) <code>bounce_supp</code> .....	306
(n_) <code>set_verify</code> .....	645	(n_) <code>get_auto_cal</code> .....	326
<code>verify_checksum</code> .....	708	(n_) <code>get_galvo_controls</code> .....	335
<b>Date, Time, Serial Numbers</b>		(n_) <code>get_hi_data</code> .....	340
(n_) <code>get_list_serial</code> .....	346	(n_) <code>get_hi_pos</code> .....	341
(n_) <code>get_serial</code> .....	354	(n_) <code>get_time</code> .....	364
(n_) <code>select_serial_set</code> .....	518	(n_) <code>home_position</code> .....	376
(n_) <code>set_serial</code> .....	615	(n_) <code>home_position_xyz</code> <sup>(2)</sup> .....	377
(n_) <code>set_serial_step</code> .....	616	(n_) <code>load_zoom_correction_file</code> .....	440
(n_) <code>time_update</code> .....	684	(n_) <code>move_to</code> .....	467

(1) Only with **Option "3D"**.

(2) Limited functionality if no **Option "3D"**.

### 10.1.6 List Commands

Meanings:

nor	normal list command
var	variable list command
us	undelayed short list command
ds	delayed short list command
mul	multiple list command

### Board Initialization and Image Field Correction

(n\_) `select_cor_table_list var` ..... 515

### 2D Jump Commands

(n_) <code>jump_abs nor</code> .....	388
(n_) <code>jump_rel nor</code> .....	390
(n_) <code>para_jump_abs nor</code> .....	469
(n_) <code>para_jump_rel nor</code> .....	471
(n_) <code>timed_jump_abs nor</code> .....	687
(n_) <code>timed_jump_rel nor</code> .....	689
(n_) <code>timed_para_jump_abs nor</code> .....	695
(n_) <code>timed_para_jump_rel nor</code> .....	697

### 3D Jump Commands<sup>(1)</sup>

(n_) <code>jump_abs_3d nor</code> .....	389
(n_) <code>jump_rel_3d nor</code> .....	391
(n_) <code>para_jump_abs_3d nor</code> .....	470
(n_) <code>para_jump_rel_3d nor</code> .....	472
(n_) <code>timed_jump_abs_3d nor</code> .....	688
(n_) <code>timed_jump_rel_3d nor</code> .....	690
(n_) <code>timed_para_jump_abs_3d mul</code> .....	696
(n_) <code>timed_para_jump_rel_3d mul</code> .....	698

### micro\_vector[\*] Commands

(n_) <code>micro_vector_abs nor</code> .....	464
(n_) <code>micro_vector_abs_3d nor</code> .....	465
(n_) <code>micro_vector_rel nor</code> .....	466
(n_) <code>micro_vector_rel_3d nor</code> .....	467

### 2D Mark Commands

(n_) <code>arc_abs nor</code> .....	297
(n_) <code>arc_rel nor</code> .....	299
(n_) <code>mark_abs nor</code> .....	442
(n_) <code>mark_ellipse_abs nor</code> .....	449
(n_) <code>mark_ellipse_rel nor</code> .....	450
(n_) <code>mark_rel nor</code> .....	451
(n_) <code>para_mark_abs nor</code> .....	475
(n_) <code>para_mark_rel nor</code> .....	477
(n_) <code>set_ellipse us</code> .....	537
(n_) <code>timed_arc_abs nor</code> .....	685
(n_) <code>timed_arc_rel nor</code> .....	686
(n_) <code>timed_mark_abs nor</code> .....	691
(n_) <code>timed_mark_rel nor</code> .....	693
(n_) <code>timed_para_mark_abs nor</code> .....	699
(n_) <code>timed_para_mark_rel nor</code> .....	701

### 3D Mark Commands<sup>(1)</sup>

(n_) <code>arc_abs_3d nor</code> .....	298
(n_) <code>arc_rel_3d nor</code> .....	300
(n_) <code>mark_abs_3d nor</code> .....	443
(n_) <code>mark_rel_3d nor</code> .....	452
(n_) <code>para_mark_abs_3d nor</code> .....	476
(n_) <code>para_mark_rel_3d nor</code> .....	478
(n_) <code>timed_mark_abs_3d nor</code> .....	692
(n_) <code>timed_mark_rel_3d nor</code> .....	694
(n_) <code>timed_para_mark_abs_3d mul</code> .....	700
(n_) <code>timed_para_mark_rel_3d mul</code> .....	702

### Text Commands

(n_) <code>mark_char us</code> .....	444
(n_) <code>mark_char_abs us</code> .....	445
(n_) <code>mark_text var</code> .....	456
(n_) <code>mark_text_abs var</code> .....	457
(n_) <code>select_char_set us</code> .....	511

### Date, Time, Serial Numbers

(n_) <code>mark_date nor</code> .....	446
(n_) <code>mark_date_abs nor</code> .....	448
(n_) <code>mark_serial nor</code> .....	453
(n_) <code>mark_serial_abs nor</code> .....	455
(n_) <code>mark_time nor</code> .....	458
(n_) <code>mark_time_abs nor</code> .....	460
(n_) <code>select_serial_set_list us</code> .....	518
(n_) <code>set_serial_step_list nor</code> .....	616
(n_) <code>time_fix nor</code> .....	682
(n_) <code>time_fix_f nor</code> .....	682
(n_) <code>time_fix_f_off nor</code> .....	683

(1) Limited functionality if no Option "3D".

<b>Status Monitoring and Diagnostics</b>		<b>Setting the Scanner Parameters</b>	
(n_) <code>set_trigger</code> <i>ds</i> .....	634	(n_) <code>set_delay_mode_list</code> <i>mul</i> .....	535
(n_) <code>set_trigger4</code> <i>ds</i> .....	642	(n_) <code>set_jump_speed</code> <i>us</i> .....	564
<b>Starting and Stopping Lists by External Control Signals and Master/Slave Synchronization</b>		(n_) <code>set_mark_speed</code> <i>ds</i> .....	576
(n_) <code>set_control_mode_list</code> <i>nor</i> .....	530	(n_) <code>set_scanner_delays</code> <i>ds</i> .....	615
(n_) <code>set_extstartpos_list</code> <i>us</i> .....	541	(n_) <code>set_sky_writing_limit_list</code> <i>us</i> .....	618
<b>List Handling and Structured Programming</b>		(n_) <code>set_sky_writing_list</code> <i>nor</i> .....	618
(n_) <code>list_continue</code> <i>nor</i> .....	403	(n_) <code>set_sky_writing_mode_list</code> <i>nor</i> .....	620
(n_) <code>list_jump_pos</code> <i>us</i> .....	405	(n_) <code>set_sky_writing_para_list</code> <i>nor</i> .....	623
(n_) <code>list_jump_rel</code> <i>us</i> .....	407		
(n_) <code>list_next</code> <i>us</i> .....	409		
(n_) <code>list_nop</code> <i>nor</i> .....	409		
(n_) <code>list_repeat</code> <i>us</i> .....	410		
(n_) <code>list_until</code> <i>us</i> .....	412		
(n_) <code>long_delay</code> <i>nor</i> .....	441		
(n_) <code>set_end_of_list</code> <i>nor</i> .....	540		
(n_) <code>set_list_jump</code> <i>us</i> .....	575		
(n_) <code>set_wait</code> <i>nor</i> .....	646		
<b>Subroutine Commands</b>			
(n_) <code>list_call</code> <i>us</i> .....	397	<b>Coordinate Transformations</b>	
(n_) <code>list_call_abs</code> <i>us</i> .....	399	(n_) <code>set_angle_list</code> <i>var</i> .....	521
(n_) <code>list_call_abs_repeat</code> <i>us</i> .....	400	(n_) <code>set_defocus_list</code> <i>var</i> (1) .....	532
(n_) <code>list_call_repeat</code> <i>us</i> .....	402	(n_) <code>set_defocus_offset_list</code> <i>var</i> (1) .....	533
(n_) <code>list_return</code> <i>us</i> .....	411	(n_) <code>set_matrix_list</code> <i>var</i> .....	579
(n_) <code>sub_call</code> <i>us</i> .....	675	(n_) <code>set_offset_list</code> <i>var</i> .....	598
(n_) <code>sub_call_abs</code> <i>us</i> .....	676	(n_) <code>set_offset_xyz_list</code> <i>var</i> (2) .....	600
(n_) <code>sub_call_abs_repeat</code> <i>us</i> .....	677	(n_) <code>set_scale_list</code> <i>var</i> .....	614
(n_) <code>sub_call_repeat</code> <i>us</i> .....	678		
<b>Setting the Laser Parameters</b>		<b>Online Positioning</b>	
(n_) <code>config_laser_signals_list</code> <i>ds</i> .....	312	(n_) <code>apply_mcbsp_list</code> <i>nor</i> .....	296
(n_) <code>set_auto_laser_params_list</code> <i>ds</i> .....	525	(n_) <code>set_mcbsp_global_matrix_list</code> <i>us</i> .....	581
(n_) <code>set_encoder_speed</code> <i>ds</i> .....	538	(n_) <code>set_mcbsp_global_rot_list</code> <i>us</i> .....	582
(n_) <code>set_firstpulse_killer_list</code> <i>us</i> .....	544	(n_) <code>set_mcbsp_global_x_list</code> <i>us</i> .....	583
(n_) <code>set_laser_delays</code> <i>us</i> .....	569	(n_) <code>set_mcbsp_global_y_list</code> <i>us</i> .....	584
(n_) <code>set_laser_pin_out_list</code> <i>us</i> .....	571	(n_) <code>set_mcbsp_matrix_list</code> <i>us</i> .....	588
(n_) <code>set_laser_pulses</code> <i>ds</i> .....	572	(n_) <code>set_mcbsp_rot_list</code> <i>us</i> .....	591
(n_) <code>set_laser_timing</code> <i>ds</i> .....	574	(n_) <code>set_mcbsp_x_list</code> <i>us</i> .....	592
(n_) <code>set_pulse_picking_list</code> <i>us</i> .....	610	(n_) <code>set_mcbsp_y_list</code> <i>us</i> .....	593
(n_) <code>set_qswitch_delay_list</code> <i>us</i> .....	610		
(n_) <code>set_softstart_level_list</code> <i>nor</i> .....	625	<b>Direct Laser and Scan Head Control</b>	
(n_) <code>set_softstart_mode_list</code> <i>var</i> .....	627	(n_) <code>laser_on_list</code> <i>var</i> .....	392
(n_) <code>set_standby_list</code> <i>ds</i> .....	629	(n_) <code>laser_on_pulses_list</code> <i>var</i> .....	393
(n_) <code>set_vector_control</code> <i>us</i> .....	643	(n_) <code>laser_signal_off_list</code> <i>nor</i> .....	395
		(n_) <code>laser_signal_on_list</code> <i>nor</i> .....	396
		(n_) <code>para_laser_on_pulses_list</code> <i>var</i> .....	473
		<b>Pixel Output Mode</b>	
		(n_) <code>set_default_pixel_list</code> <i>us</i> .....	530
		(n_) <code>set_n_pixel</code> <i>var</i> .....	597
		(n_) <code>set_pixel</code> <i>var</i> .....	603
		(n_) <code>set_pixel_line</code> <i>nor</i> .....	604
		(n_) <code>set_pixel_line_3d</code> <i>mul</i> (2) .....	606

(1) Only with Option "3D".

(2) Limited functionality if no Option "3D".

**I/O Commands**

(n_) <code>clear_io_cond_list us</code> .....	310
(n_) <code>periodic_toggle_list us</code> .....	485
(n_) <code>read_io_port_list us</code> .....	494
(n_) <code>rs232_write_text_list var</code> .....	506
(n_) <code>set_free_variable_list us</code> .....	556
(n_) <code>set_io_cond_list us</code> .....	559
(n_) <code>set_mcbsp_out us</code> .....	588
(n_) <code>set_mcbsp_out_ptr_list mul</code> .....	590
(n_) <code>set_port_default_list us</code> .....	608
(n_) <code>write_8bit_port_list ds</code> .....	714
(n_) <code>write_da_1_list ds</code> .....	716
(n_) <code>write_da_2_list ds</code> .....	717
(n_) <code>write_da_x_list ds</code> .....	719
(n_) <code>write_io_port_list ds</code> .....	721
(n_) <code>write_io_port_mask_list ds</code> .....	722

**Conditional Commands**

(n_) <code>if_cond us</code> .....	378
(n_) <code>if_not_cond us</code> .....	381
(n_) <code>if_not_pin_cond us</code> .....	383
(n_) <code>if_pin_cond us</code> .....	384
(n_) <code>list_call_abs_cond us</code> .....	400
(n_) <code>list_call_cond us</code> .....	401
(n_) <code>list_jump_cond us</code> .....	404
(n_) <code>list_jump_pos_cond us</code> .....	406
(n_) <code>list_jump_rel_cond us</code> .....	408
(n_) <code>sub_call_abs_cond us</code> .....	676
(n_) <code>sub_call_cond us</code> .....	677
(n_) <code>switch_ioport us</code> .....	679

**Processing-on-the-fly Commands**

(n_) <code>activate_fly_2d var (1)</code> .....	292
(n_) <code>activate_fly_2d_encoder mul (1)</code> .....	293
(n_) <code>activate_fly_xy var (1)</code> .....	294
(n_) <code>activate_fly_xy_encoder mul (1)</code> .....	294
(n_) <code>clear_fly_overflow us</code> .....	309
(n_) <code>fly_return nor</code> .....	323
(n_) <code>fly_return_z nor</code> .....	324
(n_) <code>if_fly_x_overflow us</code> .....	378
(n_) <code>if_fly_y_overflow us</code> .....	379
(n_) <code>if_fly_z_overflow us</code> .....	379
(n_) <code>if_not_activated us</code> .....	380
(n_) <code>if_not_fly_x_overflow us</code> .....	382
(n_) <code>if_not_fly_y_overflow us</code> .....	382
(n_) <code>if_not_fly_z_overflow us</code> .....	383
(n_) <code>park_position var (1)</code> .....	479
(n_) <code>park_return var (1)</code> .....	481
(n_) <code>set_ext_start_delay_list nor</code> .....	543
(n_) <code>set_fly_2d nor (1)</code> .....	545
(n_) <code>set_fly_limits us</code> .....	546
(n_) <code>set_fly_limits_z us</code> .....	547
(n_) <code>set_fly_rot nor (1)</code> .....	548
(n_) <code>set_fly_rot_pos nor (1)</code> .....	549
(n_) <code>set_fly_x nor (1)</code> .....	551
(n_) <code>set_fly_x_pos nor (1)</code> .....	552
(n_) <code>set_fly_y nor (1)</code> .....	553
(n_) <code>set_fly_y_pos nor (1)</code> .....	554
(n_) <code>set_fly_z nor (1)</code> .....	555
(n_) <code>set_mcbsp_in_list us (2)</code> .....	586
(n_) <code>set_multi_mcbsp_in_list nor (2)</code> .....	596
(n_) <code>set_rot_center_list ds</code> .....	611
(n_) <code>simulate_ext_start nor</code> .....	658
(n_) <code>store_encoder us</code> .....	674
(n_) <code>wait_for_encoder nor</code> .....	709
(n_) <code>wait_for_encoder_in_range mul</code> .....	710
(n_) <code>wait_for_encoder_mode mul</code> .....	711
(n_) <code>wait_for_mcbsp nor</code> .....	713

(1) Only with **Option Processing-on-the-fly**.

(2) Limited functionality if no **Option Processing-on-the-fly**.



### Controlling Stepper Motors

(n_) <b>stepper_abs_list</b> <sup>us</sup> .....	663
(n_) <b>stepper_abs_no_list</b> <sup>us</sup> .....	664
(n_) <b>stepper_control_list</b> <sup>us</sup> .....	665
(n_) <b>stepper_enable_list</b> <sup>us</sup> .....	667
(n_) <b>stepper_rel_list</b> <sup>us</sup> .....	670
(n_) <b>stepper_rel_no_list</b> <sup>us</sup> .....	671
(n_) <b>stepper_wait</b> <sup>nor</sup> .....	672

### Jump Mode

(n_) <b>set_jump_mode_list</b> <sup>nor</sup> .....	563
---	-----

### Camming

(n_) <b>camming</b> <sup>nor</sup> .....	307
--	-----

### Wobbel Mode

(n_) <b>set_wobbel</b> <sup>ds</sup> .....	647
(n_) <b>set_wobbel_control</b> <sup>us</sup> .....	649
(n_) <b>set_wobbel_direction</b> <sup>us</sup> .....	650
(n_) <b>set_wobbel_mode</b> <sup>ds</sup> .....	651
(n_) <b>set_wobbel_offset</b> <sup>ds</sup> .....	653
(n_) <b>set_wobbel_vector</b> <sup>us</sup> .....	654

### Other List Commands

(n_) <b>jump_abs_drill</b> <sup>nor</sup> .....	389
(n_) <b>jump_abs_drill_2</b> <sup>nor</sup> .....	389
(n_) <b>jump_rel_drill</b> <sup>nor</sup> .....	391
(n_) <b>jump_rel_drill_2</b> <sup>nor</sup> .....	391
(n_) <b>range_checking</b> <sup>us</sup> .....	487
(n_) <b>save_and_restart_timer</b> <sup>ds</sup> .....	508
(n_) <b>set_zoom_list</b> <sup>us</sup> .....	656



### 10.1.7 Data Types

The following table defines the formats and ranges of the different data types used by the RTC5 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 \0-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 memory as parameters. In C# and Pascal, appropriate pointer data types are therefore 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-based applications, ULONG_PTR = unsigned 64-bit value for Win64-based applications`):

```
#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 RTC5 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 naming.

<b>Category of the command</b>	<code>example_command_name_one</code>				
Function	Short description describing the purpose of the command.				
Call	Shows the correct spellings and the sequence of the parameters. Note, there is no semicolon at the end of the line. A '&' (address operator) is only used in this table row and indicates a pointer. Examples:  <code>example_command_name_one( parameter_A, &amp;parameter_B, parameter_C )</code> <code>example_variable = example_command_name_one( parameter_A )</code>				
Parameters(*)	<table border="0"> <tr> <td>A</td> <td>Short text. Data type. (*) in some C/C++ code descriptions labeled with <code>_out</code>.</td> </tr> <tr> <td>C</td> <td>Short text. Data type.</td> </tr> </table>	A	Short text. Data type. (*) in some C/C++ code descriptions labeled with <code>_out</code> .	C	Short text. Data type.
A	Short text. Data type. (*) in some C/C++ code descriptions labeled with <code>_out</code> .				
C	Short text. Data type.				
Returned parameter values(**)	<table border="0"> <tr> <td>B</td> <td>Short text. Data type. (**) in some C/C++ code descriptions labeled with <code>_out</code>.</td> </tr> </table>	B	Short text. Data type. (**) in some C/C++ code descriptions labeled with <code>_out</code> .		
B	Short text. Data type. (**) in some C/C++ code descriptions labeled with <code>_out</code> .				
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." <i>List Commands</i> generally have no return values.				
Comments	<ul style="list-style-type: none"> <li>• Additional information on this command.</li> <li>• References to other chapters and publications.</li> </ul>				
RTC4→RTC5	States the differences to the command (of the same name) of the RTC4 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: <code>command_name_two</code> , <code>command_name_three</code>				

<b>Ctrl Command</b>	<b>acquire_rtc</b>
<b>Function</b>	Acquires the specified RTC5 board for a user program.
<b>Call</b>	NoOfAcquiredCard = acquire_rtc( CardNo )
<b>Parameters</b>	CardNo <b>RTC5 DLL</b> -internal number (RTC5 board management index) of the desired board. As an unsigned 32-bit value.
<b>Result</b>	<p>The return value is:</p> <ul style="list-style-type: none"> <li>• CardNo, if the acquisition has been successful</li> <li>• 0, if the board is currently acquired by another user program or the version check detects an error</li> </ul> <p>As an unsigned 32-bit value.</p>
<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 board by multiple user programs.</li> <li>• <b>acquire_rtc</b> has no effect if CardNo exceeds the number of RTC5 Boards found during initialization (see <b>rtc5_count_cards</b>) or if CardNo = 0 (real boards begin with 1) (return value 0, <b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>• Access rights to existing boards are assigned exclusively (always to only one user program at a time). <b>acquire_rtc</b> therefore has no effect if the requested board is already reserved by another user program (return value 0, <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>). Before an already-reserved board can be acquired by <b>acquire_rtc</b>, the user program with current access rights must explicitly release its rights by <b>release_rtc</b> or <b>free_rtc5_dll</b>. On the other hand, if the board has been already freed prior to initialization of a user program, then initialization by <b>init_rtc5_dll</b> result in RTC5 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 the CardNo.</li> <li>• Assorted versions of the <b>RTC5 DLL</b> and the program files <b>RTC5OUT.out</b>, <b>RTC5RBF.rbf</b> and <b>RTC5DAT.dat</b> cannot be arbitrarily combined with another. <b>acquire_rtc</b> performs a version compatibility check. If no program files are loaded, then a version check cannot be explicitly executed (<b>get_last_error</b> return code <b>RTC5_TIMEOUT</b>), but the check still regarded as successful and acquisition is not hindered. If program files have been loaded and the version check determines an error, then access is denied (return value 0, <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>   <b>RTC5_VERSION_MISMATCH</b>).</li> <li>• A board successfully requested by <b>acquire_rtc</b> does not automatically become the active board. Activation is only achieved by <b>select_rtc</b> or <b>init_rtc5_dll</b>.</li> <li>• Running boards are neither halted nor initialized by <b>acquire_rtc</b>.</li> <li>• <b>acquire_rtc</b> is available even without explicit access rights to a particular RTC5 Board.</li> <li>• <b>acquire_rtc</b> is not available as a multi-board command.</li> <li>• See also <b>Chapter 6.7.1 "Board Acquisition by a User Program"</b>, page 117.</li> </ul>

Ctrl Command	acquire_rtc
RTC4→RTC5	New command.
Version info	–
References	<a href="#">init_rtc5_dll</a> , <a href="#">select_rtc</a> , <a href="#">free_rtc5_dll</a> , <a href="#">release_rtc</a> , <a href="#">rtc5_count_cards</a>

Variable List Command	activate_fly_2d
Function	Activates a <a href="#">set_fly_2d</a> Processing-on-the-fly application without encoder resets.
Restriction	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <a href="#">activate_fly_2d</a> terminates the Processing-on-the-fly process (even though it could not have been activated).
Call	<code>activate_fly_2d( ScaleX, ScaleY )</code>
Parameters	<p>ScaleX      Scaling factor as for <a href="#">activate_fly_2d_encoder</a>.</p> <p>ScaleY      Like ScaleX.</p>
Comments	<ul style="list-style-type: none"> <li>If no Processing-on-the-fly correction is active, then <a href="#">activate_fly_2d</a> activates <a href="#">set_fly_2d</a> Processing-on-the-fly correction (see <a href="#">Chapter 8.6.4 "Compensating 2D Motions", page 234</a>). Unlike <a href="#">set_fly_2d</a>, <a href="#">activate_fly_2d</a> does not thereby reset the encoders, but instead recalculates the last Processing-on-the-fly-uncorrected 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 24-bit virtual <a href="#">Image field</a>, then Processing-on-the-fly correction is not activated and an error bit is set that is queryable by <a href="#">get_marking_info</a> (<a href="#">Bit #9</a>).</li> <li>If Processing-on-the-fly correction is <i>active</i>, then <a href="#">activate_fly_2d</a> is a short list command without further effect and merely sets an error bit queryable by <a href="#">get_marking_info</a> (<a href="#">Bit #9</a>). Therefore, <a href="#">activate_fly_2d</a> 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 <a href="#">if_notActivated</a> in order to possibly jump to an appropriate error handling routine.</li> <li>Successful activation by <a href="#">activate_fly_2d</a> does <i>not</i> reset any already-set error bit. It remains set for <a href="#">get_marking_info</a> until <a href="#">get_marking_info</a> is called.</li> <li>If unallowed parameter values are supplied (for example, for ScaleX = 0), then <a href="#">activate_fly_2d</a> is (already during loading) replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC5_PARAM_ERROR</a>).</li> <li><a href="#">activate_fly_2d</a> does not affect the laser signals (<a href="#">Signals for "Laser Active" Operation</a> remain on/off if they were on/off).</li> </ul>
RTC4→RTC5	<p>New command.</p> <p><a href="#">RTC4 Compatibility Mode</a>: see <a href="#">set_fly_2d</a>.</p>
Version info	–
References	<a href="#">set_fly_2d</a> , <a href="#">activate_fly_2d_encoder</a> , <a href="#">activate_fly_xy</a>

Multiple List Command	activate_fly_2d_encoder
Function	Activates a <b>set_fly_2d</b> Processing-on-the-fly application with encoder reset and encoder offsets.
Restriction	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).
Call	activate_fly_2d_encoder( ScaleX, ScaleY, EncX, EncY )
Parameters	<p>ScaleX      Scaling factor as for <b>set_fly_2d</b>.</p> <p>ScaleY      Like ScaleX.</p> <p>EncX        Encoder offset. As a signed 32-bit value.</p> <p>EncY        Like EncX.</p>
Comments	<ul style="list-style-type: none"> <li>• <b>activate_fly_2d_encoder</b> occupies two list memory area positions and also needs two 10 <math>\mu</math>s clock cycles to execute (the first part of <b>activate_fly_2d_encoder</b> 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 <b>Image field</b> coordinates, but the parameter values <b>EncX</b> and <b>EncY</b>. 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 motion (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 <b>EncX</b> and <b>EncY</b>, 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 <b>EncX</b> or <b>EncY</b> is not allowed (that is, the Processing-on-the-fly-corrected virtual <b>Image field</b> coordinates are outside the virtual <b>Image field</b> limits), then <b>activate_fly_2d_encoder</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>• If the value of <b>ScaleX</b> or <b>ScaleY</b> 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 <b>RTC5_PARAM_ERROR</b>).</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→RTC5	New command. <b>RTC4 Compatibility Mode:</b> see <a href="#">set_fly_2d</a> .
Version info	Available as of DLL 544, OUT 544.
References	<a href="#">set_fly_2d</a> , <a href="#">activate_fly_2d</a> , <a href="#">activate_fly_xy_encoder</a>

<b>Variable List Command</b>	<b>activate_fly_xy</b>
Function	Activates a <a href="#">set_fly_x</a> / <a href="#">set_fly_y</a> Processing-on-the-fly application without encoder resets.
Restriction	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <b>activate_fly_xy</b> terminates the Processing-on-the-fly process (even though it could not have been activated).
Call	<code>activate_fly_xy( ScaleX, ScaleY )</code>
Parameters	<code>ScaleX</code> Scaling factor as for <a href="#">set_fly_x</a> . <code>ScaleY</code> Scaling factor as for <a href="#">set_fly_y</a> .
Comments	<ul style="list-style-type: none"> <li>Like <a href="#">activate_fly_2d</a> with the difference that a <a href="#">set_fly_x</a>/<a href="#">set_fly_y</a> Processing-on-the-fly session is activated.</li> </ul>
RTC4→RTC5	New command. <b>RTC4 Compatibility Mode:</b> see <a href="#">set_fly_x</a> / <a href="#">set_fly_y</a> .
Version info	–
References	<a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">activate_fly_xy_encoder</a>

<b>Multiple List Command</b>	<b>activate_fly_xy_encoder</b>
Function	Activates a <a href="#">set_fly_x</a> / <a href="#">set_fly_y</a> Processing-on-the-fly application with encoder reset and encoder offsets.
Restriction	If the <a href="#">Option Processing-on-the-fly</a> 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).
Call	<code>activate_fly_xy_encoder( ScaleX, ScaleY, EncX, EncY )</code>
Parameters	<code>ScaleX</code> Scaling factor as for <a href="#">set_fly_x</a> . <code>ScaleY</code> Scaling factor as for <a href="#">set_fly_y</a> . <code>EncX</code> Encoder offset. As a signed 32-bit value. <code>EncY</code> Encoder offset. As a signed 32-bit value.
Comments	<ul style="list-style-type: none"> <li>Like <a href="#">activate_fly_2d_encoder</a> with the difference that a <a href="#">set_fly_x</a>/<a href="#">set_fly_y</a> Processing-on-the-fly session is activated.</li> </ul>
RTC4→RTC5	New command. <b>RTC4 Compatibility Mode:</b> see <a href="#">set_fly_x</a> / <a href="#">set_fly_y</a> .
Version info	Available as of DLL 544, OUT 544.
References	<a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">activate_fly_xy</a> , <a href="#">activate_fly_2d_encoder</a>

<b>Ctrl Command</b>	<b>apply_mcbsp</b>
<b>Function</b>	Queries 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><b>HeadNo</b>      Number of the scan head connector.            As an unsigned 32-bit value.            = 1: The definition only affects the first scan head connector.            = 2: The definition only affects the second scan head connector.            = 0, 3: The definition affects <i>both</i> scan head connectors.            Only the two least significant bits are evaluated.</p> <p><b>at_once</b>      Determines when the defined transformation becomes effective.            As 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 currently <b>BUSY list execution status</b> or <b>INTERNAL-BUSY list execution status</b> is set) 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 <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, <b>apply_mcbsp</b> only defines (as with <b>set_offset</b>) an x offset 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 213). As with the commands described in <b>Chapter 8.2 “Coordinate Transformations”</b>, page 209, the parameter <b>at_once</b> 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> ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>. That is, data provided is not transmitted, see <b>Section “RTC5 as Transmitter”</b>, page 72.</li> </ul>

<b>Ctrl Command</b>	<b>apply_mcbsp</b>
RTC4→RTC5	New command.
Version info	–
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	<code>apply_mcbsp_list( HeadNo, at_once )</code>				
Parameters	<table> <tr> <td>HeadNo</td> <td>Like <a href="#">apply_mcbsp</a>.</td> </tr> <tr> <td>at_once</td> <td> <p>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 collected 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 collected until then) and processed prior to the next list command.</li> <li>= 2: The transformation settings are only collected and intermediately stored (as with <code>at_once = 0</code>). However, The transformation is immediately calculated (including all transformation settings that were collected and intermediately stored until then) and applied to the current position when the next <a href="#">jump_abs</a> or <a href="#">jump_rel</a> (only if no list is currently being executed: also <a href="#">goto_xy</a> or <a href="#">goto_xyz</a>) is executed.</li> <li>&gt; 2: As <code>at_once = 2</code>.</li> </ul> </td> </tr> </table>	HeadNo	Like <a href="#">apply_mcbsp</a> .	at_once	<p>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 collected 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 collected until then) and processed prior to the next list command.</li> <li>= 2: The transformation settings are only collected and intermediately stored (as with <code>at_once = 0</code>). However, The transformation is immediately calculated (including all transformation settings that were collected and intermediately stored until then) and applied to the current position when the next <a href="#">jump_abs</a> or <a href="#">jump_rel</a> (only if no list is currently being executed: also <a href="#">goto_xy</a> or <a href="#">goto_xyz</a>) is executed.</li> <li>&gt; 2: As <code>at_once = 2</code>.</li> </ul>
HeadNo	Like <a href="#">apply_mcbsp</a> .				
at_once	<p>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 collected 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 collected until then) and processed prior to the next list command.</li> <li>= 2: The transformation settings are only collected and intermediately stored (as with <code>at_once = 0</code>). However, The transformation is immediately calculated (including all transformation settings that were collected and intermediately stored until then) and applied to the current position when the next <a href="#">jump_abs</a> or <a href="#">jump_rel</a> (only if no list is currently being executed: also <a href="#">goto_xy</a> or <a href="#">goto_xyz</a>) is executed.</li> <li>&gt; 2: As <code>at_once = 2</code>.</li> </ul>				
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">apply_mcbsp</a>.</li> </ul>				
RTC4→RTC5	New command.				
Version info	–				
References	<a href="#">apply_mcbsp</a>				



Normal List Command	arc_abs						
Function	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 <b>Image field</b> .						
Parameters	<table> <tr> <td>X</td><td>Absolute x coordinate of the arc center. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td>Y</td><td>Like X (analogously).</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 x coordinate of the arc center. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.	Y	Like X (analogously).	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 x coordinate of the arc center. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.						
Y	Like X (analogously).						
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.						
Comments	<ul style="list-style-type: none"> <li>If the mark speed has not been previously explicitly set by <b>set_mark_speed</b> or <b>set_mark_speed_ctrl</b>, then the marking is executed at a predefined mark speed of 1000 <i>bits/ms</i>.</li> <li>The <b>Signals for "Laser Active" Operation</b> are automatically turned on at the beginning of the marking (or remain on after a directly preceding <b>[*]mark[*]</b> Command or <b>"Arc"</b> command). The defined <b>Scanner Delays</b> and <b>Laser Delays</b> are thereby taken into account (see <b>Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control"</b>, page 133). Note that other delays are executed in <b>Sky Writing</b> mode . Exception: zero-length <b>"Arc"</b> commands (see notes on <b>page 137</b>).</li> </ul>						
RTC4→RTC5	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for X and Y by 16. The allowed value ranges decrease accordingly.						
Version info	–						
References	<b>set_mark_speed</b> , <b>set_scanner_delays</b> , <b>arc_rel</b> , <b>timed_arc_abs</b> , <b>mark_abs</b> , <b>arc_abs_3d</b> , <b>mark_ellipse_abs</b>						



Normal List Command	arc_abs_3d								
Function	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.								
Restriction	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <a href="#">select_cor_table</a> ), then <b>arc_abs_3d</b> has the same effect as <a href="#">arc_abs</a> .								
Call	<code>arc_abs_3d( X, Y, Z, Angle )</code>								
Parameters	<table> <tr> <td>X</td><td>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: [-8,388,608...+8,388,607]. 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>Absolute z end coordinate. In bits. As a signed 32-bit value. Allowed value range: [-32,768...+32,767]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td>Angle</td><td>Arc angle. In degrees. As a 64-bit IEEE floating point value. Positive angle values correspond to clockwise angles. Allowed value range: [-3600.0°...+3600.0°] (<math>\pm 10</math> full circles). Out-of-range values are clipped to the boundary values.</td></tr> </table>	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: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.	Y	Like X (analogously).	Z	Absolute z end coordinate. In bits. As a signed 32-bit value. Allowed value range: [-32,768...+32,767]. Out-of-range values are clipped to the boundary values.	Angle	Arc angle. In degrees. As a 64-bit IEEE floating point value. Positive angle values correspond to clockwise angles. Allowed value range: [-3600.0°...+3600.0°] ( $\pm 10$ full circles). Out-of-range values are clipped to the boundary values.
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: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.								
Y	Like X (analogously).								
Z	Absolute z end coordinate. In bits. As a signed 32-bit value. Allowed value range: [-32,768...+32,767]. Out-of-range values are clipped to the boundary values.								
Angle	Arc angle. In degrees. As a 64-bit IEEE floating point value. Positive angle values correspond to clockwise angles. Allowed value range: [-3600.0°...+3600.0°] ( $\pm 10$ full circles). Out-of-range values are clipped to the boundary values.								
Comments	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>arc_abs_3d</b> functions similarly to <a href="#">arc_abs</a> (see the comments there).</li> <li>The z motion is not taken into account during calculation of the number of <a href="#">Microsteps</a>.</li> </ul>								
RTC4→RTC5	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the values specified for X and Y by 16. The allowed value range decreases accordingly. The value range for Z is identical in <a href="#">RTC5 Standard Mode</a> and <a href="#">RTC4 Compatibility Mode</a> .								
Version info	–								
References	<a href="#">arc_abs</a> , <a href="#">arc_rel_3d</a>								



Normal List Command	arc_rel
Function	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 <b>Image field</b> .
Call	arc_rel( dx, dy, Angle )
Parameters	<p>dx      Relative x coordinate of the arc center. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.</p> <p>dy      Like dx (analogously).</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>
Comments	<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, <b>arc_rel</b> is identical to <b>arc_abs</b> (see the comments there).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for dx and dy by 16. The allowed value range decreases accordingly.
Version info	–
References	<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>



Normal List Command	<a href="#">arc_rel_3d</a>
Function	Moves the laser focus at mark speed from the current position spirally 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.
Restriction	If the <a href="#">Option "3D"</a> is not enabled or no 3D correction table has been assigned (see <a href="#">select_cor_table</a> ), then <a href="#">arc_rel_3d</a> has the same effect as <a href="#">arc_rel</a> .
Call	<code>arc_rel_3d( dx, dy, dz, Angle )</code>
Parameters	<p><code>dx</code> Position of the helical axis (parallel to the z axis) as relative x coordinate. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.</p> <p><code>dy</code> Like <code>dx</code> (analogously).</p> <p><code>dz</code> Relative z end coordinate. In bits. As a signed 32-bit value. Allowed value range: [-32,768...+32,767]. 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>
Comments	<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 supplied as relative coordinates with respect to the current position. Otherwise, <a href="#">arc_rel_3d</a> is identical to <a href="#">arc_abs_3d</a> (see the comments there).</li> </ul>
RTC4→RTC5	New command. <a href="#">RTC4 Compatibility Mode</a> : see <a href="#">arc_abs_3d</a>
Version info	–
References	<a href="#">arc_abs_3d</a> , <a href="#">arc_rel</a>

<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>	<code>ErrorCode = auto_cal( HeadNo, Command )</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.                     Requires <b>Option "Second Scan Head Control"</b>.</p> <p>Command      Control parameter.                     As an unsigned 32-bit value.                     Allowed value range: [0...4].                     = 0:    The RTC5 detects the current Home-In positions, stores them in the DLL and in the <b>EEPROM</b> as Home-In reference values and initializes the gain values and offset values (Gain = 1.0, Offset = 0).                     = 1:    The RTC5 detects the current Home-In positions, calculates and sets the new gain values and offset values and thereby activates drift compensation.                     = 2:    The RTC5 deactivates drift compensation by initializing the gain and offset values (Gain = 1.0, Offset = 0).                     = 3:    The RTC5 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 RTC5 checks the ASC hardware (that is, checks 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>EEPROM</b>.</p>
<b>Result</b>	<p>Error code or type of sensor system.                     As an unsigned 32-bit value.</p> <p>3              <b>auto_cal</b> cannot be executed because the <b>BUSY list execution status</b> or <b>INTERNAL-BUSY list execution status</b> is currently set.</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 (<b>Galvanometer scanner 2</b>)                     10: for y axis (<b>Galvanometer scanner 1</b>)                     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 x axis or y axis 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   EEPROM write error (for this error, the <code>get_last_error</code> return code <code>RTC5_EEPROM_ERROR</code> 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 (<i>Galvanometer scanner 2</i>): a sensor system of type1 is included and is functioning. y axis (<i>Galvanometer scanner 1</i>): sensor system is defective.</p> <p>29   x axis (<i>Galvanometer scanner 2</i>): a sensor system of type2 is included and is functioning. y axis (<i>Galvanometer scanner 1</i>): sensor system is defective.</p> <p>91   x axis (<i>Galvanometer scanner 2</i>): sensor system is defective. y axis (<i>Galvanometer scanner 1</i>): a sensor system of type1 is included and is functioning.</p> <p>92   x axis (<i>Galvanometer scanner 2</i>): sensor system is defective. y axis (<i>Galvanometer scanner 1</i>): 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 <code>auto_cal</code> usage, see <a href="#">Chapter 8.10 "Automatic Self-Calibration", page 251</a>.</li> <li>At the end of this command's execution, the RTC5 always moves the galvanometer scanners back to the position held prior to the call, possibly with corrections attributable to changed gain values and offset values.</li> <li>During determination of the current Home-In positions with <code>auto_cal</code> (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 <code>auto_cal</code> (Command = 0). Otherwise, the RTC5 uses the stored reference values of a previously operated (possibly different) scan system when later performing an automatic self-calibration.</li> <li>After initialization of the RTC5, or after a reset, drift compensation is turned off (gain = 1.0, offset = 0). However, previously determined reference values are still available.</li> </ul>

Ctrl Command	auto_cal
Comments (cont'd)	<ul style="list-style-type: none"> <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>RTC5_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>RTC5_BUSY</code>), if: <ul style="list-style-type: none"> <li>the <b>BUSY list execution status</b> is set</li> <li>the <b>INTERNAL-BUSY list execution status</b> 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 list execution status</b> set)</li> </ul> </li> <li>For <code>RTC5_PARAM_ERROR</code>, the <b>BUSY list execution status</b> is not checked; therefore return codes <code>RTC5_BUSY</code> and <code>RTC5_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>EEPROM</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>EEPROM</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>EEPROM</b> by explicitly calling <code>auto_cal</code> (Command = 0) or <code>auto_cal</code> (Command = 4). The error is <i>not</i> cleared from the <b>EEPROM</b> by calling <code>auto_cal</code> (Command = 1).</li> </ul>
RTC4→RTC5	The functions for automatic self-calibration (Command = 0...2) are (largely) unchanged. New: Command = 3 and Command = 4.
Version info	–
References	<code>set_hi</code> , <code>get_hi_pos</code> , <code>get_auto_cal</code> , <code>write_hi_pos</code>



<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→RTC5	Unchanged functionality.
Version info	–
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>	<code>auto_change_pos( Pos )</code>
<b>Parameters</b>	<p>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.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>auto_change_pos</code> triggers a subsequent <i>one-time-only</i> list change or a list new start. For further list changes or list new starts, <code>auto_change_pos</code> must be called again.</li> <li>• <code>auto_change_pos</code> can be called at any desired point in time.</li> <li>• If the automatic list change is activated during processing of a list, then upon reaching <code>set_end_of_list</code> execution continues without delay at the supplied start position of the other list. If there is only <i>one</i> list (<code>Mem2 = 0</code>, see <a href="#">config_list</a>), then upon reaching <code>set_end_of_list</code> 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 <a href="#">Chapter 6.4.6 "Automatic List Changing", page 99</a>).</li> <li>• So that <code>auto_change_pos</code> can function at all, any already active list must absolutely be finalized by <code>set_end_of_list</code>; 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 <code>set_end_of_list</code>, 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 has been already processed and the other list has been started (at the supplied start position). If no list has been 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: <code>Pos ≥ Mem1</code> or <code>Pos ≥ Mem2</code>), then the start position is set to the beginning of the list (<code>Pos = 0</code>).</li> <li>• If, during processing of a list, the <code>auto_change_pos( Pos &gt; 0 )</code> and <code>start_loop</code> commands are called, then upon the next <code>set_end_of_list</code> the command <code>auto_change_pos( Pos &gt; 0 )</code> is executed; and at the next one the <code>start_loop</code> command is executed.</li> <li>• The current <a href="#">List Status</a> values can be queried by <code>read_status</code>.</li> <li>• The current <a href="#">List Execution Status</a> values can be queried by <code>get_status</code>.</li> <li>• <code>auto_change_pos</code> triggers a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 94</a>.</li> </ul>

<b>Ctrl Command</b>	<b>auto_change_pos</b>
RTC4→RTC5	<p>Essentially unchanged, however:</p> <p>The list memory address (<code>Pos</code>) is supplied to the RTC5 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 <a href="#">config_list (Mem2 = 0)</a>, then the RTC5 command behaves like the RTC4 command.</p>
Version info	–
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	<p>Length      Debouncing time. In ms.            As an unsigned 32-bit value.            Allowed value range: [0...1023]. The 22 bits with higher significance are ignored.</p>
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 266</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 <a href="#">get_marking_info (Bit #8)</a>. Increase the debouncing time until this additional signal is no longer detected.</li> <li>• The debouncing time default value is 0 ms.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_startstop_info</a>

Normal List Command	camming
Function	Enables <b>Camming</b> functionality.
Call	camming ( FirstPos, NPos, EncoderNo, Ctrl, Scale, Code )
Parameters	<p>FirstPos      Starting address of the <b>Camming</b> command list (absolute address in list memory).  As an unsigned 32-bit value. Allowed value range: [0...(2<sup>20</sup>-1)].</p> <p>NPos      Length of the <b>Camming</b> command list (without terminating <b>set_end_of_list</b> or <b>list_return</b>).  As an unsigned 32-bit value.  Allowed value range: [1...(2<sup>20</sup>-FirstPos)].</p> <p>EncoderNo      Number of the encoder counter, whose pulses is evaluated for controlling the <b>Camming</b> process.  As an unsigned 32-bit value.  Allowed values:  = 0:      Encoder counter "Encoder0".  = 1:      Encoder counter "Encoder1".  Bits with higher significance are ignored.</p> <p>Ctrl      <b>Camming</b> control mode.  As an unsigned 32-bit value.  Allowed values:  = 0:      RTC5 controls laser similarly to a <b>[*]mark[*] Command</b>.  <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 <b>Camming</b> command list is executed until the end point (0 or NPos-1) and then waits for a new external /START.  = 3:      User controls laser.  <b>camming</b> does <i>not</i> terminate automatically.  The <b>Camming</b> command list is continuously processed in circulation (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 value range: 2<sup>-60</sup> &lt;  Scale  &lt; 2<sup>60</sup>.</p> <p>Code      As an unsigned 32-bit value. Is not used.</p>
Comments	<ul style="list-style-type: none"> <li>• See also <b>Chapter 8.11 "Camming", page 255</b>.</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>RTC5_PARAM_ERROR</b>).</li> </ul>



Normal List Command	camming
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 point in time, the corresponding command index of the <b>Camming</b> 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 <b>Camming</b> 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 mark command) switched on at the beginning of the <b>Camming</b> 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 then 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 <b>Camming</b> 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 <b>External Starts</b> 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 <b>Camming</b> process can only be terminated by <b>stop_execution</b> or an <b>External Stop</b>. However, in these two modes (as an exception) <b>External Starts</b> are allowed if the list (or <b>camming</b>) is still active. If <b>Ctrl</b> = 2, then the index is executed until the end point (0 or <b>NPos</b>-1) and then waits for a new external /START. If <b>Ctrl</b> = 3, then the index is set to modulus <b>NPos</b> (whereby the encoder speed is supposed not to be so high that a complete rotation is skipped over). 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→RTC5	New command.
Version info	–
References	<b>set_encoder_speed</b> , <b>set_auto_laser_control</b>



<b>Undelayed Short List Command</b>	<b>clear_fly_overflow</b>
Function	Resets the specified error bits (from <a href="#">get_marking_info</a> ) for customer-defined monitoring of Processing-on-the-fly applications.
Call	<code>clear_fly_overflow( Mode )</code>
Parameters	<p>Mode      The error bits to be reset.      As an unsigned 32-bit value.</p> <p>Bit #0    = 1: error bit <a href="#">Bit #4</a> (underflow X).      Bit #1    = 1: error bit <a href="#">Bit #5</a> (overflow X).      Bit #2    = 1: error bit <a href="#">Bit #6</a> (underflow Y).      Bit #3    = 1: error bit <a href="#">Bit #7</a> (overflow Y).      Bit #4    = 1: error bit <a href="#">Bit #24</a> (underflow Z).      Bit #5    = 1: error bit <a href="#">Bit #25</a> (overflow Z).      Bit #0...Bit #5 can be combined as desired.      Higher-order bits are ignored.</p>
Comments	<ul style="list-style-type: none"> <li>For usage of <a href="#">clear_fly_overflow</a>, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>All 6 error bits are reset for:       <ul style="list-style-type: none"> <li>Mode = 0</li> <li>Mode = 63</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_marking_info</a> , <a href="#">clear_fly_overflow_ctrl</a>

<b>Ctrl Command</b>	<b>clear_fly_overflow_ctrl</b>
Function	Like <a href="#">clear_fly_overflow</a> , but a control command.
Call	<code>clear_fly_overflow_ctrl( Mode )</code>
Parameters	Mode      Like <a href="#">clear_fly_overflow</a> .
Comments	<ul style="list-style-type: none"> <li>See <a href="#">clear_fly_overflow</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 548, OUT 548.
References	<a href="#">clear_fly_overflow</a>



<b>Undelayed Short List Command</b>	<b>clear_io_cond_list</b>
Function	<p>Clears the bits of the 16-bit digital output port on the EXTENSION 1 socket connector that are set in the parameter <code>MaskClear</code>, if the current <code>IOvalue</code> at the 16-bit digital <i>input</i> port on the EXTENSION 1 socket connector 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).</p>
Call	<code>clear_io_cond_list( Mask1, Mask0, MaskClear )</code>
Parameters	<p><code>Mask1</code> 16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</p> <p><code>Mask0</code> See <code>Mask1</code>.</p> <p><code>MaskClear</code> See <code>Mask1</code>.</p>
Comments	<ul style="list-style-type: none"> <li>• <code>clear_io_cond_list</code> clears only those bits of the digital output port that are set in the parameter <code>MaskClear</code> and leaves the other bits unchanged.</li> <li>• See also <a href="#">Section "16-Bit Digital Input Port and 16-Bit Digital Output Port", page 65</a> and <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 271</a>.</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 Bit #1 to Bit #3 (DIGITAL IN1...3) of the input port are not set: <code>clear_io_cond_list(\$0001, \$000E, \$0010)</code></li> <li>• Always clear Bit #15 of the output port (and leave the other bits unchanged): <code>clear_io_cond_list(0, 0, \$8000)</code></li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
References	<a href="#">set_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>config_laser_signals</b>																				
<b>Function</b>	Configures the laser signal types to be outputted on pin 1 (LASER1), pin 2 (LASERON) and pin 9 (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.                     The following bits configure:</p> <table> <tr><td>Bit #0...1:</td><td>Pin 2 (LASERON channel)</td></tr> <tr><td>Bit #2...3:</td><td>Pin 1 (LASER1 channel)</td></tr> <tr><td>Bit #4...5:</td><td>Pin 9 (LASER2 channel)</td></tr> <tr><td>Bit #6:</td><td>Ignored</td></tr> <tr><td>...</td><td></td></tr> <tr><td>Bit #31:</td><td>Ignored</td></tr> </table> <p>Signal type:</p> <table> <tr><td>= 0 = 00<sub>b</sub>:</td><td>LASERON signal</td></tr> <tr><td>= 1 = 01<sub>b</sub>:</td><td>LASER1 signal</td></tr> <tr><td>= 2 = 10<sub>b</sub>:</td><td>LASER2 signal</td></tr> <tr><td>= 3 = 11<sub>b</sub>:</td><td>FirstPulseKiller signal</td></tr> </table> <p>The default setting (after <a href="#">load_program_file</a>) is:  <code>Config = 100100<sub>b</sub> = 0x24 = 36</code></p>	Bit #0...1:	Pin 2 (LASERON channel)	Bit #2...3:	Pin 1 (LASER1 channel)	Bit #4...5:	Pin 9 (LASER2 channel)	Bit #6:	Ignored	...		Bit #31:	Ignored	= 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
Bit #0...1:	Pin 2 (LASERON channel)																				
Bit #2...3:	Pin 1 (LASER1 channel)																				
Bit #4...5:	Pin 9 (LASER2 channel)																				
Bit #6:	Ignored																				
...																					
Bit #31:	Ignored																				
= 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																				
<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> <li>See also <a href="#">set_laser_control</a>, Bit #3 and Bit #4.</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 only generates LASER1 signals if the LASERON signal is on).</li> </ul>																				
<b>RTC4→RTC5</b>	New command.																				
<b>Version info</b>	–																				
<b>References</b>	<a href="#">config_laser_signals_list</a> , <a href="#">set_laser_control</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      Like <a href="#">config_laser_signals</a> .
Comments	<ul style="list-style-type: none"><li>• See <a href="#">config_laser_signals</a>.</li></ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">config_laser_signals</a>

<b>Ctrl Command</b>	<b>config_list</b>
<b>Function</b>	Configures the list memory, that is, assigns specific memory locations to the list memory areas.
<b>Call</b>	<code>config_list( Mem1, Mem2 )</code>
<b>Parameters</b>	Mem1      Storage positions for list memory area "List 1". As an unsigned 32-bit value.
	Mem2      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 RTC5 list memory contains 1,048,576 (= <math>2^{20}</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 through parameters <b>Mem1</b> and <b>Mem2</b>. <b>config_list</b> automatically assigns to the protected area ("List 3") all remaining memory not assigned to "List 1" or "List 2".</li> <li>The following rules apply to the parameters <b>Mem1</b> and <b>Mem2</b> (invalid values are automatically corrected in the specified order): <ul style="list-style-type: none"> <li><b>Mem1 &gt; 0</b> ("List 1" cannot be empty) <b>Mem1 = 0</b> is corrected to <b>Mem1 = 1</b>.</li> <li><b>Mem1 ≤ 2<sup>20</sup></b> ("List 1" can contain a maximum of <math>2^{20}</math> storage positions) <b>Mem1 &gt; 2<sup>20</sup></b> is corrected to <b>Mem1 = 2<sup>20</sup></b></li> <li><b>Mem1 = "-1"</b> is interpreted as <b>Mem1 = (2<sup>32</sup>-1)</b> and corrected to <b>Mem1 = 2<sup>20</sup></b>. Example: With <code>config_list( -1, x )</code> where <b>x</b> is any value (also <b>x = -1</b>), "List 1" is automatically assigned the entire list memory (<b>Mem1 = 2<sup>20</sup>, Mem2 = 0</b>, no memory for "List 3").</li> <li><b>Mem2 ≤ 2<sup>20</sup> - Mem1</b> ("List 2" can maximally receive the "rest" of list memory) <b>Mem2 = 0</b> is allowed. <b>Mem2 &gt; 2<sup>20</sup> - Mem1</b> is corrected to <b>Mem2 = 2<sup>20</sup> - Mem1</b>.</li> <li><b>Mem2 = "-1"</b> is interpreted as <b>Mem2 = (2<sup>32</sup>-1)</b> and corrected to <b>Mem2 = 2<sup>20</sup> - Mem1</b>. Example: With <code>config_list( Mem1, -1 )</code>, "List 2" is automatically assigned with the "rest" of list memory (<b>Mem2 = 2<sup>20</sup> - Mem1</b>, no memory for "List 3").</li> <li>Storage positions for "List 3": <math>2^{20} - \text{Mem1} - \text{Mem2}</math></li> </ul> </li> <li>By default, the RTC5's list memory area is preconfigured so that "List 1" and "List 2" can each accept 4000 list commands (<b>Mem1 = Mem2 = 4000</b>). The protected "List 3" then owns the remaining 1040576 of the <math>2^{20}</math> storage positions.</li> <li><b>config_list</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY list execution status</b> is set</li> <li>a list has been paused by <b>set_wait</b> (<b>PAUSED list execution status</b> set)</li> </ul> </li> </ul>



Ctrl Command	config_list
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 has been 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> </ul> <p>Also observe the notes in <a href="#">Chapter 6.3.2 "Configuring the RTC5 List Memory", page 92</a>.</p> <ul style="list-style-type: none"> <li>If you do not know the current configuration data for list memory (Mem1 and Mem2), you can find out after <code>load_list( ListNo, 0 )</code> or <code>set_start_list_pos( ListNo, 0 )</code> by using <b>get_list_space</b> (if the board changed "ownership" previously call <b>get_config_list</b>).</li> </ul>
RTC4→RTC5	New command.
References	<a href="#"><b>get_config_list</b></a>



<b>Ctrl Command</b>	<b>control_command</b>	
<b>Function</b>	Sends a control command. Whether and how the addressed scan system reacts depends on its properties (among other things, the scan system firmware). See also <a href="#">Comments, page 316</a> .	
<b>Call</b>	<code>control_command( Head, Axis, Data )</code>	
<b>Parameters</b>	<b>Head</b>	Scan head connector number of the RTC control board. As an unsigned 32-bit value. Allowed values: = 1: The scan system at 1st scan head connector. "Scan head A". = 2: The scan system at 2nd scan head connector. "Scan head B".
	<b>Axis</b>	Number of the axis. As an unsigned 32-bit value. Allowed values: = 1: x axis ( <a href="#">STATUS channel, Galvanometer scanner 2</a> ). = 2: y axis ( <a href="#">STATUS1 channel, Galvanometer scanner 1</a> ).
	<b>Data</b>	Command code with optional parameter. See <a href="#">Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747</a> . As an unsigned 32-bit value. The upper part (Bit #16...Bit #31) is <i>not</i> evaluated. The lower part (Bit #0...Bit #15) is evaluated as follows: <ul style="list-style-type: none"> <li>• <b>Code<sub>HIGH</sub></b> The more significant data byte (Bit #8...Bit 15). Represents a command code. Presented as hexadecimal number in <a href="#">Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747</a>.</li> <li>• <b>Code<sub>LOW</sub></b> The less significant data byte (Bit #0...Bit #7). Represents an optional parameter. Presented as hexadecimal number in <a href="#">Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747</a>.</li> </ul> Example: By <b>Data</b> = <b>0501<sub>H</sub></b> , that is, <b>Code<sub>HIGH</sub></b> = <b>05</b> ( <a href="#">SetMode</a> ) and <b>Code<sub>LOW</sub></b> = <b>01</b> the actual position is set as the data type to be transmitted.



Ctrl Command	control_command
Comments	<ul style="list-style-type: none"> <li>• <b>control_command</b> can only be used in conjunction with iDRIVE scan systems (see Glossary entry on <a href="#">page 23</a>).</li> <li>• <b>control_command</b> is not evaluated by conventional scan systems (without iDRIVE technology).</li> <li>• <b>control_command</b> is not executed with invalid values of Head and/or Axis (<a href="#">get_last_error</a> return code <a href="#">RTC5_PARAM_ERROR</a>).</li> <li>• Command code <a href="#">Data</a> is transmitted to the scan system instead of the usual position data. Therefore, the corresponding galvanometer scanner <a href="#">Microstep</a> is omitted, if <b>control_command</b> is called during execution of a list.</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 <a href="#">set_auto_laser_control( Mode = 2 )</a>.</li> </ul>
RTC4→RTC5	<p>Unchanged functionality.</p> <p>Note that all returned data selected by <b>control_command</b> are always in 20-bit range, even in <a href="#">RTC4 Compatibility Mode</a> (see also comments on <a href="#">get_value</a>).</p>
Version info	–
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>	<code>copy_dst_src( Dst, Src, Mode )</code>
<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> <p>Bit #0 = 0: <b>Dst</b> is the index of an indexed character or the index of an indexed text string.</p> <p>Bit #0 = 1: <b>Dst</b> is the index of an indexed subroutine.</p> <p>Bit #1 = 0: <b>Src</b> is the index of an indexed character or the index of an indexed text string.</p> <p>Bit #1 = 1: <b>Src</b> is the index of an indexed subroutine.</p> <p>Bit #2...Bit #31: Not evaluated.</p>
<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 (<a href="#">get_last_error</a> return code <b>RTC5_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 list memory area's memory contents. 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 (after the <b>copy_dst_src</b> command) with <a href="#">save_disk/load_disk</a>. 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 buffer overflow of the protected list memory area "List 3".</li> <li>See also <a href="#">Section "Managing Indexed Characters and Text Strings", page 109</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">save_disk</a> , <a href="#">load_disk</a>



Ctrl Command	<b>disable_laser</b>
Function	Disables the <a href="#">Signals for "Laser Active" Operation</a> .
Call	<code>disable_laser()</code>
Comments	<ul style="list-style-type: none"> <li>• <b>disable_laser</b> disables the <a href="#">Signals for "Laser Active" Operation</a> at the output ports LASER1, LASER2 and LASERON (the signals are set to their respective "Off" level). Pin (2) 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 the pins (1) and (9) 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 <a href="#">Signals for "Laser Active" Operation</a> 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 RTC5 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 <b>disable_laser</b> results in an <a href="#">RTC5_TIMEOUT</a> error (for example, if no program file has been 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>.</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 enabled" (yes or no) can be queried.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	Last change: DLL 546, OUT 546: <a href="#">get_startstop_info</a> ( <a href="#">Bit #14</a> ).
References	<a href="#">enable_laser</a> , <a href="#">set_laser_control</a> , <a href="#">get_startstop_info</a>



Ctrl Command	<code>enable_laser</code>
Function	Enables the Signals for "Laser Active" Operation.
Call	<code>enable_laser()</code>
Comments	<ul style="list-style-type: none"> <li>After a <a href="#">Hardware reset</a> (and after <a href="#">load_program_file</a>), <a href="#">set_laser_control</a> 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, <a href="#">enable_laser</a> has no effect.</li> <li>After initialization of the RTC5 with <a href="#">load_program_file</a>, the laser control signals are <i>deactivated</i>. For first-time activation, <a href="#">set_laser_control</a> must be called (see above). After a subsequent disabling by <a href="#">disable_laser</a> (or <a href="#">set_laser_control</a>), the <a href="#">enable_laser</a> (or <a href="#">set_laser_control</a>) command can be used for reenabling.</li> <li>Even if the laser control signals have been enabled with <a href="#">enable_laser</a> or <a href="#">set_laser_control</a>, they are not outputted without further commands (see <a href="#">Chapter 7.4 "Laser Control", page 173</a>).</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 enabled" can be queried.</li> </ul>
RTC4→RTC5	Basically unchanged functionality. In some circumstances, <a href="#">set_laser_control</a> must be called before <a href="#">enable_laser</a> (see above).
Version info	–
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>execute_at_pointer</b>
Function	Starts list execution ("List 1" or "List 2") at the specified address in the RTC5 list memory area.
Call	<code>execute_at_pointer( Pos )</code>
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>• <code>execute_at_pointer</code> essentially functions like <code>execute_list_pos</code> (see the 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</code> <math>\geq</math> <code>Mem1 + Mem2</code> (see <code>config_list</code>), <code>Pos</code> is set to 0.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
References	<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→RTC5	Unchanged functionality.
Version info	–
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>
<b>Function</b>	See <a href="#">execute_list</a> .
<b>Call</b>	<code>execute_list_1</code>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<a href="#">execute_list</a>

<b>Ctrl Command</b>	<b>execute_list_2</b>
<b>Function</b>	See <a href="#">execute_list</a> .
<b>Call</b>	<code>execute_list_2</code>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<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>	<table> <tr> <td>ListNo</td> <td>Number of the list to be executed. As an unsigned 32-bit value. Allowed values: [uneven: “List 1”, even: “List 2”].</td> </tr> <tr> <td>Pos</td> <td>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...(<math>2^{20}</math>–1)].</td> </tr> </table>	ListNo	Number of the list to be executed. As an unsigned 32-bit value. Allowed values: [uneven: “List 1”, even: “List 2”].	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...( $2^{20}$ –1)].
ListNo	Number of the list to be executed. As an unsigned 32-bit value. Allowed values: [uneven: “List 1”, even: “List 2”].				
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...( $2^{20}$ –1)].				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>execute_list_pos</b> is not executed (<a href="#">get_error</a> return value = <code>RTC5_BUSY</code>), if:             <ul style="list-style-type: none"> <li>– the <b>BUSY list execution status</b> is set</li> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED list execution status</b> set)</li> </ul> </li> <li>• If the <b>INTERNAL-BUSY list execution status</b> is set, <b>execute_list_pos</b> is only executed with a delay (after <b>INTERNAL-BUSY list execution status</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 94</a>.</li> <li>• Programs in the protected 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 <code>ListNo</code> values cause “List 1” to be executed; otherwise “List 2” is executed. This allows automatically generated continuous list changing by an incremented count.</li> </ul>				



Ctrl Command	execute_list_pos
Comments (cont'd)	<ul style="list-style-type: none"> <li>If "List 2" has not been assigned memory (<code>Mem2</code> = 0, see <a href="#">config_list</a>) then "List 1" is opened.</li> <li>If <code>Pos</code> 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 <code>Pos</code> is set to 0.</li> <li>The <b>BUSY list status</b> of the selected list is set and the <b>BUSY list status</b> of the other corresponding list is reset (see <a href="#">read_status</a>). The <b>BUSY list execution status-List Execution Status</b> (see <a href="#">get_status</a>) is set.</li> <li>Execution stops when a <a href="#">set_end_of_list</a> is encountered. If the end of a list area is reached without encountering a <a href="#">set_end_of_list</a>, 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 <a href="#">set_end_of_list</a> has been encountered and an <a href="#">auto_change_pos</a> or <a href="#">start_loop</a> has been previously called. For both lists to be treated as a single list, you must set the configuration appropriately: for example, <code>config_list( Mem1+Mem2, 0 )</code>.</li> <li>If a home jump (defined with <a href="#">home_position</a> or <a href="#">home_position_xyz</a>) has been executed by <a href="#">set_end_of_list</a>, then <a href="#">execute_list_pos</a> leads to a corresponding home return (the <b>INTERNAL-BUSY list execution status</b> is set while the home return is executed).</li> <li><a href="#">execute_list_pos</a> triggers a flush of the buffered list input (see <a href="#">Chapter 6.4.1 "Loading Lists", page 94</a>), even if the start has been unsuccessful.</li> <li><a href="#">execute_list_pos</a> also covers the specialized variants <a href="#">execute_list_1</a>, <a href="#">execute_list_2</a>, <a href="#">execute_list</a> and <a href="#">execute_at_pointer</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">execute_list</a> , <a href="#">execute_at_pointer</a> , <a href="#">set_start_list_pos</a>



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 <a href="#">Option Processing-on-the-fly</a> is not enabled, <b>fly_return</b> only executes the jump to the specified new output position.	
Call	<code>fly_return( X, Y )</code>	
Parameters	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 <a href="#">jump_abs</a> (see comments there).</li> <li>If Processing-on-the-fly-correction has been activated within a subroutine called by an "AbsCall" and subsequently gets deactivated by <b>fly_return</b>, then the coordinate values specified with <b>fly_return</b> receive an offset (based on the current coordinates at the time of the call, see also <a href="#">Section ""AbsCalls""</a>, page 103).</li> <li>See also <a href="#">Chapter 8.6 "Processing-on-the-fly"</a>, page 227.</li> </ul>	
RTC4→RTC5	Unchanged functionality. In addition: increased value range, and "AbsCall", see above. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the values specified for X and Y by 16. The allowed value range decreases accordingly.	
Version info	–	
References	<a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">set_fly_rot</a> , <a href="#">set_fly_x_pos</a> , <a href="#">set_fly_y_pos</a> , <a href="#">set_fly_rot_pos</a>	



Normal List Command	<b>fly_return_z</b>						
Function	Deactivates the previously set Processing-on-the-fly correction. Subsequently executes a jump to the defined position.						
Restriction	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.						
Call	<b>fly_return_z( X, Y, Z )</b>						
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> <tr> <td>Z</td><td>Like X (analogously). Allowed value range: [-32,768...+32,767].</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).	Z	Like X (analogously). Allowed value range: [-32,768...+32,767].
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).						
Z	Like X (analogously). Allowed value range: [-32,768...+32,767].						
Comments	<ul style="list-style-type: none"> <li>Like <b>fly_return</b>, however, with additional Z output position.</li> </ul>						
RTC4→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified value for X and Y by 16. The allowed value range decreases accordingly.</p> <p>The value range for Z is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b>.</p>						
Version info	–						
References	<b>set_fly_z</b>						



Ctrl Command	<b>free_RTC5_dll</b>
Function	Frees up all resources allocated by the <b>RTC5 DLL</b> for a user program.
Call	<code>free_RTC5_dll</code>
Comments	<ul style="list-style-type: none"> <li>• <b>RTC5 DLL</b>-allocated resources particularly include memory area in the <b>RTC5 DLL</b> allocated for the RTC5 board management (which is created by <b>init_RTC5_dll</b>). <b>free_RTC5_dll</b> deletes the RTC5 board management of the <b>RTC5 DLL</b>. Afterward, the user program has no access to boards (<b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>).</li> <li>• The calling of <b>free_RTC5_dll</b> is not absolutely necessary, because <b>RTC5 DLL</b>-assigned resources are automatically freed up when the user program terminates and the <b>RTC5 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>RTC5 DLL</b> is unloaded. The calling of <b>free_RTC5_dll</b> eliminates this annoyance.</li> <li>• <b>free_RTC5_dll</b> is not available as a multi-board command.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>init_RTC5_dll</b> , <b>release_RTC</b>



<b>Ctrl Command</b>	<b>get_auto_cal</b>
<b>Function</b>	Returns the attached scan system's type of ASC hardware previously detected by <b>auto_cal</b> .
<b>Call</b>	ASCtype = get_auto_cal( HeadNo )
<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>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 has been previously detected by <b>auto_cal</b>, then <b>get_auto_cal</b> returns the same value as <b>auto_cal(Command = 0)</b>, see comments there.</li> <li>If the ASC hardware type has <i>not</i> been previously detected by <b>auto_cal</b>, then <b>get_auto_cal</b> returns the value 255 (initialized value).</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<b>auto_cal</b>



<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 has been loaded into the protected list memory area "List 3" or an unprotected subroutine has been only subsequently referenced.</li> <li>• If <code>Index &gt; 1023</code> or if no character has been 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→RTC5	New command.
Version info	–
References	<a href="#">get_sub_pointer</a> , <a href="#">get_text_table_pointer</a>

<b>Ctrl Command</b>	<b>get_config_list</b>
<b>Function</b>	Passes the parameters of the current list memory configuration ( <code>Mem1</code> , <code>Mem2</code> ) to the RTC5 board management of the <b>RTC5 DLL</b> and initializes it as if the <b>config_list</b> command has been called.
<b>Call</b>	<code>get_config_list()</code>
<b>Result</b>	–
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>get_config_list</b> is useful when a board changes “ownership” and the new RTC5 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</code> = 4000 and <code>Mem2</code> = 4000; the board by <b>load_program_file</b> and board management when starting the corresponding user program). See also <b>Chapter 6.7.1 “Board Acquisition by a User Program”</b>, page 117.</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> list execution status.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>config_list</b>

<b>Ctrl Command</b>	<b>get_counts</b>
<b>Function</b>	Reads the current number of successful <b>External Starts</b> .
<b>Call</b>	<code>Counts = get_counts()</code>
<b>Result</b>	Number of successful <b>External Starts</b> . 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. The counter can be reset to zero by <b>set_control_mode</b>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
References	<b>set_max_counts</b> , <b>set_control_mode</b> , <b>get_startstop_info</b>

<b>Ctrl Command</b>	<b>get_dll_version</b>
<b>Function</b>	Returns the version number of the RTC5 DLL.
<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>RTC5 DLL</b> version numbers are in the range 500-599.</li> <li><b>get_dll_version</b> is available even without explicit access rights to a specific RTC5 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 119</a>) are neither generated nor altered by <b>get_dll_version</b>.</li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<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>	<table> <tr> <td>Encoder0</td> <td>Current count of encoder counter "Encoder0". As a pointer to a signed 32-bit value.</td> </tr> <tr> <td>Encoder1</td> <td>Current count of encoder counter "Encoder1". As a pointer to a signed 32-bit value.</td> </tr> </table>	Encoder0	Current count of encoder counter "Encoder0". As a pointer to a signed 32-bit value.	Encoder1	Current count of encoder counter "Encoder1". As a pointer to a signed 32-bit value.
Encoder0	Current count of encoder counter "Encoder0". As a pointer to a signed 32-bit value.				
Encoder1	Current count of encoder counter "Encoder1". As a pointer to a signed 32-bit value.				
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>get_encoder</b> usage, 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 274</a>.</li> <li>If the workpiece motion is registered with an incremental encoder: <ul style="list-style-type: none"> <li>Encoder counter "Encoder0" is triggered by the signals at encoder input port <b>ENCODER X</b></li> <li>Encoder counter "Encoder1" is triggered by the signals at encoder input port <b>ENCODER Y</b></li> </ul> </li> <li>If an encoder simulation has been started by <b>simulate_encoder( 3 )</b>, both encoder counters are triggered by an internal periodic 1 MHz clock signal.</li> </ul>				
<b>RTC4→RTC5</b>	Unchanged functionality. In addition: increased value range.				
<b>Version info</b>	–				
<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>RTC5_NO_ERROR</td> <td>= 0</td> </tr> <tr> <td>Bit #0 (LSB)</td> <td>= 1: No board found (this error can only occur by <a href="#">init_RTC5_dll</a>).</td> <td>RTC5_NO_CARD</td> <td>= 1</td> </tr> <tr> <td>Bit #1</td> <td>= 1: Access denied (this error can occur by <a href="#">init_RTC5_dll</a>, <a href="#">select_RTC</a>, <a href="#">acquire_RTC</a> or any multi-board command).</td> <td>RTC5_ACCESS_DENIED</td> <td>= 2</td> </tr> <tr> <td>Bit #2</td> <td>= 1: Command not forwarded (this error implies an internal, PCI or driver error, for example, caused by a hardware defect or an incorrect connection).</td> <td>RTC5_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 RTC5; this error can especially occur in connection with control commands that expect a response, for example, <a href="#">get_head_para</a>).</td> <td>RTC5_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>RTC5_PARAM_ERROR</td> <td>= 16</td> </tr> </tbody> </table>				Bit	Error type	Error constant		–	No error.	RTC5_NO_ERROR	= 0	Bit #0 (LSB)	= 1: No board found (this error can only occur by <a href="#">init_RTC5_dll</a> ).	RTC5_NO_CARD	= 1	Bit #1	= 1: Access denied (this error can occur by <a href="#">init_RTC5_dll</a> , <a href="#">select_RTC</a> , <a href="#">acquire_RTC</a> or any multi-board command).	RTC5_ACCESS_DENIED	= 2	Bit #2	= 1: Command not forwarded (this error implies an internal, PCI or driver error, for example, caused by a hardware defect or an incorrect connection).	RTC5_SEND_ERROR	= 4	Bit #3	= 1: No response from board (it is likely that no program has been loaded onto the RTC5; this error can especially occur in connection with control commands that expect a response, for example, <a href="#">get_head_para</a> ).	RTC5_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).	RTC5_PARAM_ERROR	= 16
Bit	Error type	Error constant																														
–	No error.	RTC5_NO_ERROR	= 0																													
Bit #0 (LSB)	= 1: No board found (this error can only occur by <a href="#">init_RTC5_dll</a> ).	RTC5_NO_CARD	= 1																													
Bit #1	= 1: Access denied (this error can occur by <a href="#">init_RTC5_dll</a> , <a href="#">select_RTC</a> , <a href="#">acquire_RTC</a> or any multi-board command).	RTC5_ACCESS_DENIED	= 2																													
Bit #2	= 1: Command not forwarded (this error implies an internal, PCI or driver error, for example, caused by a hardware defect or an incorrect connection).	RTC5_SEND_ERROR	= 4																													
Bit #3	= 1: No response from board (it is likely that no program has been loaded onto the RTC5; this error can especially occur in connection with control commands that expect a response, for example, <a href="#">get_head_para</a> ).	RTC5_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).	RTC5_PARAM_ERROR	= 16																													



Ctrl Command	get_error			
	Bit #5 = 1: List processing is (not) active (for example, for <b>execute_list</b> , if a list is currently being processed for example, for <b>stop_execution</b> , if no list is currently being processed for example, for <b>restart_list</b> , if <b>pause_list</b> has not been previously called).	RTC5_BUSY	= 32	
	Bit #6 = 1: List command rejected, illegal input pointer (for example, for any list command directly after <b>load_char + list_return</b> : the list command is then not loaded).	RTC5_REJECTED	= 64	
	Bit #7 = 1: List command has been converted to a <b>list_nop</b> (for example, <b>set_end_of_list</b> in a protected subroutine).	RTC5_IGNORED	= 128	
	Bit #8 = 1: Version error: <b>RTC5 DLL</b> version, <b>RTC5RBF.rbf</b> version and <b>RTC5OUT.out</b> version are not compatible with each other. See also <b>load_program_file</b> .	RTC5_VERSION_MISMATCH	= 256	
	Bit #9 = 1: Verify error: The download verification (see <a href="#">page 120</a> ) has detected an incorrect download.	RTC5_VERIFY_ERROR	= 512	

Ctrl Command	get_error			
	Bit #10 = 1: DSP version error: DSP version too old (this error only occurs with older RTC5 Boards, see <a href="#">get_RTC_version</a> Bit #16...Bit #23 – and only through a few commands such as <a href="#">mark_ellipse_abs</a> ; the corresponding command description's "Version info" section contains related comments; Commands that generate the error are neither executed nor replaced by <a href="#">list_nop</a> ).		RTC5_TYPE_REJECTED	= 1024
	Bit #11 = 1: A RTC5 DLL-internal Windows memory request failed.		RTC5_OUT_OF_MEMORY	= 2048
	Bit #12 = 1: EEPROM read or write error (can occur during initialization or <a href="#">auto_cal</a> ).		RTC5_EEPROM_ERROR	= 4096
	Bit #13 Reserved.		-	
	Bit #14 Reserved.		-	
	Bit #15 Reserved.		-	
	Bit #16 = 1: Error reading PCI configuration register (can only occur during <a href="#">init_RTC5_dll</a> ).		RTC5_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 119</a>.</li> <li><a href="#">get_error</a> and <a href="#">n_get_error</a> are even available without explicit access rights to a specific RTC5 Board.</li> <li>The board-specific error variables <a href="#">LastError</a> and <a href="#">AccError</a> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <a href="#">get_error</a>.</li> </ul>			



Ctrl Command	get_error
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 = <a href="#">init_RTC5_dll()</a>;// Searches for all installed RTC5 Boards if (Error &amp; RTC5_ACCESS_DENIED) { // at least one board is inaccessible     UINT Count = <a href="#">RTC5_count_cards()</a>; // number of boards found     for (UINT Num = 1; Num &lt;= Count; Num++) {         NoAccess[Num] = <a href="#">n_get_last_error</a>(Num) &amp; RTC5_ACCESS_DENIED;         <a href="#">n_reset_error</a>(Num, RTC5_ACCESS_DENIED);     } }</pre>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_last_error</a> , <a href="#">reset_error</a> , <a href="#">set_verify</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>	<code>OffsetX</code> x reference value. As a pointer to a signed 32-bit value. <code>OffsetY</code> y reference value. As a pointer to a signed 32-bit value.
<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>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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>	<code>No</code> Number of the free variable to be queried. As an unsigned 32-bit value. Allowed value range: [0...7]. Only the 3 least significant bits are evaluated.
<b>Result</b>	The value currently stored in the free variable <code>No</code> . 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 123</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	Last change DLL 539, OUT 539: value range of <code>No</code> increased to [0...7].
<b>References</b>	<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>RTC5_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 unsigned 64-bit value) to an array of five unsigned 32-bit values, where the desired 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 unsigned 64-bit value) to an array of 4 unsigned 32-bit values, where the corresponding control values are to be stored: XA, YA, XB, YB. Range of values in each case is [-524,288...+524,287].</p>
<b>Returned parameter values</b>	XA, YA, XB, YB denote the control values for the x axis/y axis of scan head A/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. Users are solely responsible to apply these by <code>at_once &gt; 0</code> before <b>get_galvo_controls</b> is called. 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>RTC5_BUSY</b> has been generated.</li> </ul>



Ctrl Command	get_galvo_controls
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for x and y by 16. The allowed value range decreases accordingly. Even in <b>RTC4 Compatibility Mode</b> , all returned values are in the RTC5 20-bit range.
Version info	Available as of DLL 539, OUT 539.
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="#">RTC5_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 <a href="#">Option "Second Scan Head Control"</a> 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→RTC5	New command.
Version info	–
References	<a href="#">get_table_para</a>



<b>Ctrl Command</b>	<b>get_head_status</b>																																													
<b>Function</b>	Returns the <b>XY2-100 status word</b> 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><b>XY2-100 status word.</b> As an unsigned 32-bit value.</p> <table> <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. See comment, <a href="#">page 339</a>.</td> </tr> <tr> <td></td> <td>Bit #7</td> <td>Power Status, 1 = OK. See comment, <a href="#">page 339</a>.</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> </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. See comment, <a href="#">page 339</a> .		Bit #7	Power Status, 1 = OK. See comment, <a href="#">page 339</a> .	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. See comment, <a href="#">page 339</a> .																																												
	Bit #7	Power Status, 1 = OK. See comment, <a href="#">page 339</a> .																																												
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 available even 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>.</li> <li>The status bits are returned by <b>get_head_status</b> in Bit #3...Bit #7 and/or Bit #11...Bit #15. Independently of the scan system's current state, Bit #0 and Bit #8 are returned by <b>get_head_status</b> as 1, while Bit #1 and Bit #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 Bit #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>With an <b>XY2-100 Converter (Accessory)</b> the value 0 is returned, if no scan system is connected or not switched on.</li> <li>The PowerOK signal is an electronically generated signal. It means: The scan system servo control is ready for input data ("PowerOK", actually corresponds to a "ServoOK"). Therefore, it cannot be used to check whether a connected scan head is switched on at all. <b>get_startstop_info</b> (Bit #17 and/or Bit #25) can be used for distinguishing.</li> </ul>																																													

Ctrl Command	get_head_status
Comments (cont'd)	<ul style="list-style-type: none"> <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/VE scan systems (see Glossary entry on <a href="#">page 23</a>) <ul style="list-style-type: none"> <li>Without the SL2-100 interface, <b>get_head_status</b> only returns meaningful return values, if the <b>XY2-100 status word</b> has been 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, Bit #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 <b>get_value</b>, <b>get_values</b>, <b>set_trigger</b> and <b>set_trigger4</b>.</li> <li>See also <a href="#">Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 221</a> for information about using two scan heads.</li> </ul>
RTC4→RTC5	<p>Basically unchanged functionality. However:</p> <ul style="list-style-type: none"> <li>The RTC5 allows the simultaneous reading of both scan head connectors' status words, and with iDRI/VE scan systems with SL2-100 interface even independently of the signal to be returned (set by <b>control_command</b>).</li> <li>With iDRI/VE scan systems, Bit #0 and Bit #1 do not return information about the operational readiness of the scan system (see <b>get_value</b>).</li> </ul>
Version info	–
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>RTC5OUT.out</b> , which is currently loaded on the RTC5.
<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 500...599.  <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 2500...2599 (version number + 2000)</li> <li>if the <b>Option "3D"</b> is enabled values in the range 3500...3599 (version number + 3000)</li> </ul> </li> <li>The file name extension for RTC5 program files is no longer <code>*.hex</code> (as with the RTC4), but instead <code>*.out</code> (see also <b>load_program_file</b>).</li> <li>The software version number can also be returned after an <b>RTC5_VERSION_MISMATCH</b> or <b>RTC5_ACCESS_DENIED</b> error. The return value is 0 if no program has yet been loaded.</li> <li>The board-specific error variables <b>LastError</b> and <b>AccError</b>, see <b>Section "Error Handling", page 119</b>, are neither generated nor altered by <b>get_hex_version</b>.</li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<b>get_dll_version</b> , <b>get_RTC_version</b>

<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>x1 coordinate of the currently stored (most recently measured) Home-In position. In bits. As a pointer to a signed 32-bit value.</td> </tr> <tr> <td>X2</td> <td>Like X1 (analogously).</td> </tr> <tr> <td>Y1</td> <td>Like X1 (analogously).</td> </tr> <tr> <td>Y2</td> <td>Like X1 (analogously).</td> </tr> </table>	X1	x1 coordinate of the currently stored (most recently measured) Home-In position. In bits. As a pointer to a signed 32-bit value.	X2	Like X1 (analogously).	Y1	Like X1 (analogously).	Y2	Like X1 (analogously).
X1	x1 coordinate of the currently stored (most recently measured) Home-In position. In bits. As a pointer to a signed 32-bit value.								
X2	Like X1 (analogously).								
Y1	Like X1 (analogously).								
Y2	Like X1 (analogously).								
<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>								
<b>RTC4→RTC5</b>	Unchanged functionality. In addition: increased value range. The returned values are in the RTC5 20-bit range.								
<b>Version info</b>	–								
<b>References</b>	<b>get_hi_pos</b> , <b>write_hi_pos</b>								

<b>Ctrl Command</b>	<b>get_hi_pos</b>	
<b>Function</b>	Returns the Home-In positions, last determined (by <a href="#">auto_cal</a> ) 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>	X1	x1 coordinate of the currently stored (most recently measured) Home-In positions. In bits. As a pointer to a signed 32-bit value.
	X2	Like X1 (analogously).
	Y1	Like X1 (analogously).
	Y2	Like X1 (analogously).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>get_hi_pos</b> usage, see <a href="#">Section "Customer-Specific Calibration", page 254</a>.</li> <li>It is up to users to ensure 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 no scan system equipped with automatic self-calibration (Home-In sensors) is attached to the specified scan head connector or if an error has occurred during determination of the Home-In values for such a system.</li> <li>Directly after initialization (<a href="#">init_rtc5_dll</a>), particularly prior to a first call of <a href="#">auto_cal( Command = 0, 1 or 3 )</a>, the returned values are the Home-In reference values stored in the <a href="#">EEPROM</a>. If such reference values have not been successfully determined at least once by <a href="#">auto_cal( Command = 0 )</a>, <b>get_hi_pos</b> returns 0.</li> <li><b>get_hi_pos</b> is available even without explicit access rights to a specific RTC5 Board.</li> <li>If parameter values are invalid, then all returned coordinates are 0 (<a href="#">get_last_error</a> return code <a href="#">RTC5_PARAM_ERROR</a>).</li> </ul>	
RTC4→RTC5	New command.	
Version info	–	
References	<a href="#">get_hi_data</a> , <a href="#">auto_cal</a> , <a href="#">set_hi</a> , <a href="#">write_hi_pos</a>	

<b>Ctrl Command</b>	<b>get_input_pointer</b>
<b>Function</b>	Returns the present <i>absolute</i> position of the input pointer.
<b>Call</b>	InputPointer = <code>get_input_pointer()</code>
<b>Result</b>	position of the input pointer [0...(2 <sup>20</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 RTC5 list memory area (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><code>get_input_pointer</code> 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 <code>get_input_pointer</code> 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>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <code>get_input_pointer</code>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
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><code>get_io_status</code> is intended 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 272</a>).</li> <li>See also <a href="#">Section "16-Bit Digital Input Port and 16-Bit Digital Output Port", page 65</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
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>	Retrieves the board's currently stored <b>Jump Delay</b> table and copies the 1024 corresponding unsigned 16-bit values to the supplied PC address.
<b>Call</b>	<code>ErrorCode = get_jump_table( Addr )</code>
<b>Parameters</b>	Addr      PC address for the 2048 byte memory area.
<b>Result</b>	Error code as unsigned 32-bit value. 0      No error. 11      RTC5 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→RTC5	New command.
Version info	–
Reference	<b>set_jump_table, load_jump_table_offset</b>

<b>Ctrl Command</b>	<b>get_lap_time</b>
<b>Function</b>	Returns the <i>current</i> <b>RTC5 Timer</b> value (without resetting it to zero).
<b>Call</b>	<code>TimerValue = get_lap_time()</code>
<b>Result</b>	RTC5 Timer value since the last call of <b>save_and_restart_timer</b> . In seconds. 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→RTC5	New command.
Version info	Available as of DLL 541, OUT 541.
Version info	–
References	<b>get_time, save_and_restart_timer</b>



<b>Ctrl Command</b>	<b>get_laser_pin_in</b>
<b>Function</b>	Returns the current status of the two digital inputs at the laser connector.
<b>Call</b>	<code>LaserPinIn = get_laser_pin_in()</code>
<b>Result</b>	<p>As an unsigned 32-bit value.</p> <p>Bit #0      DIGITAL IN1. (<b>LSB</b>)</p> <p>Bit #1      DIGITAL IN2.</p> <p>Bit #2      Reserved.</p> <p>...      ...</p> <p>Bit 31      Reserved.</p>
<b>Comments</b>	• –
RTC4→RTC5	New command.
<b>Version info</b>	–
<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>	<p>Error code.</p> <p>As an unsigned 32-bit value.</p> <p>If multiple errors occurred simultaneously, then multiple bits are set.</p> <p>The meanings of bit numbers, error types and error constants is identical to those for <a href="#">get_error</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For error handling see <a href="#">Chapter 6.8 "Error Handling", page 119</a>.</li> <li><code>get_last_error</code> and <code>n_get_last_error</code> are even available without explicit access rights to a specific RTC5 Board.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <code>get_last_error</code>.</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<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>	Provides the input pointer's current <i>relative</i> position and 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 input pointer's absolute list memory area address (offset relative to the start of "List 1") can be queried by <a href="#">get_input_pointer</a> (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 <a href="#">get_list_space</a>.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <a href="#">get_list_pointer</a>.</li> <li>If the input pointer is invalid when <a href="#">get_list_pointer</a> is called (for example, after a <a href="#">list_return</a>), the return values are <code>ListNo = 0</code> and <code>Pos = 0xFFFFFFF</code>.</li> </ul>				
RTC4→RTC5	New command.				
Version info	–				
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 <a href="#">select_serial_set_list</a> (or of serial-number-set 0 after <a href="#">load_program_file</a> ) and the current serial number of this serial-number-set.
<b>Call</b>	<code>LastMarkedSerialNo = get_list_serial( &amp;Set )</code>
<b>Result</b>	Serial number. As a 64-bit IEEE floating point value.
<b>Returned parameter value</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 <a href="#">mark_serial</a> or <a href="#">mark_serial_abs</a>.</li> <li>For <b>get_list_serial</b> usage, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 196</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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>	<code>ListSpace = get_list_space()</code>
<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>
<b>RTC4→RTC5</b>	<b>get_list_space</b> has been made available on the RTC4 to support the <a href="#">RTC4 Circular Queue Mode</a> and returns the distance between the input pointer and output pointer. The RTC5 does not support the <a href="#">RTC4 Circular Queue Mode</a> . The input pointer position can be queried by <a href="#">get_input_pointer</a> or <a href="#">get_list_pointer</a> and the output pointer position can be queried by <a href="#">get_status</a> or <a href="#">get_out_pointer</a> .
<b>Version info</b>	–
<b>References</b>	<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. The function also returns the error bits of laser-signal auto-suppression.
Call	<code>MarkingInfo = get_marking_info()</code>
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 <b>Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications"</b>, page 237, <b>Chapter 8.6.8 "Encoder Resets"</b>, page 239 and <b>Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections"</b>, page 240.</p> <p>Bit #10 PosAck error bit of scan system A, x axis.</p> <p>Bit #11 TempOK error bit of scan system A, x axis.</p> <p>Bit #12 PowerOK error bit of scan system A, x axis.</p> <p>Bit #13 PosAck error bit of scan system A, y axis.</p> <p>Bit #14 TempOK error bit of scan system A, y axis.</p> <p>Bit #15 PowerOK error bit of scan system A, y axis.</p> <p>Bit #16 = 1: The distance between the edges in signal 1 of encoder input port <b>ENCODER X</b> is too short.</p> <p>Bit #17 = 1: The distance between the edges in signal 1 of encoder input port <b>ENCODER Y</b> is too short.</p> <p>Bit #18 = 1: The distance between the edges in signal 2 of encoder input port <b>ENCODER X</b> is too short.</p> <p>Bit #19 = 1: The distance between the edges in signal 2 of encoder input port <b>ENCODER Y</b> is too short.</p>

Ctrl Command	get_marking_info
Result (cont'd)	<p>Bit #20 = 1: Improper signal sequence at encoder input port <b>ENCODER X</b>.</p> <p>Bit #21 = 1: Improper signal sequence at encoder input port <b>ENCODER Y</b>.</p> <p>Bit #22 = 1: Processing-on-the-fly underflow in z direction (<math>Z &lt; -32,768</math>).</p> <p>Bit #23 = 1: Processing-on-the-fly overflow in z direction (<math>Z &gt; +32,767</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> <p>Bit #26 PosAck error bit of scan system B, x axis.</p> <p>Bit #27 TempOK error bit of scan system B, x axis.</p> <p>Bit #28 PowerOK error bit of scan system B, x axis.</p> <p>Bit #29 PosAck error bit of scan system B, y axis.</p> <p>Bit #30 TempOK error bit of scan system B, y axis.</p> <p>Bit #31 PowerOK error bit of scan system B, y axis.</p>
Comments	<ul style="list-style-type: none"> <li>For usage of <b>get_marking_info</b> and of the error bits <b>Bit #0...Bit #7</b>, see <a href="#">Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections", page 240</a>.</li> <li>The limits for the customer-defined monitoring range are determined for: <ul style="list-style-type: none"> <li><math>X_{min}, X_{max}, Y_{min}, Y_{max}</math> (<b>Bit #4...Bit #7</b>) by <b>set_fly_limits</b></li> <li><math>Z_{min}, Z_{max}</math> (<b>Bit #24...Bit #25</b>) by <b>set_fly_limits_z</b></li> </ul> </li> <li>The error bits <b>Bit #4...Bit #7</b> and <b>Bit #24...Bit #25</b>: <ul style="list-style-type: none"> <li>Are <i>not</i> reset by <b>get_marking_info</b></li> <li>Get implicitly reset by the conditional commands</li> <li>Can also be explicitly reset by <b>clear_fly_overflow</b> and <b>clear_fly_overflow_ctrl</b></li> </ul> </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>The error bits <b>Bit #10...Bit #15</b> and <b>Bit #26...Bit #31</b> 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 <b>set_laser_control</b>. All error bits are reset by <b>get_marking_info</b>.</li> </ul>



Ctrl Command	get_marking_info
Comments (cont'd)	<ul style="list-style-type: none"><li>• All error bits are reset during initialization (by <a href="#">load_program_file</a>).</li><li>• All error bits (except Bit #4...Bit #7 and Bit #24...Bit #25):<ul style="list-style-type: none"><li>– Are reset when reading out by <a href="#">get_marking_info</a></li><li>– However, can be reset at any time as long as the error condition still persists</li></ul></li></ul>
RTC4→RTC5	Unchanged functionality for info bits that are also used on the RTC4.
Version info	–
References	<a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> , <a href="#">set_fly_z</a> , <a href="#">set_fly_rot</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_fly_limits</a> , <a href="#">set_fly_limits_z</a> , <a href="#">clear_fly_overflow</a> , <a href="#">clear_fly_overflow_ctrl</a> , <a href="#">if_notActivated</a>



<b>Ctrl Command</b>	<b>get_master_slave</b>
<b>Function</b>	Returns the master/slave status of the addressed RTC5 board.
<b>Call</b>	MasterSlaveStatus = get_master_slave()
<b>Result</b>	<p>Master/slave status. As an unsigned 32-bit value. Information whether a further RTC5 board is connected to the Master or Slave connector of the addressed RTC5 board:</p> <p>Bit #0 = 1: A RTC5 board is connected to the Slave connector. (LSB)</p> <p>Bit #1 = 1: A RTC5 board is connected to the Master connector.</p> <p>Bit #2 = 0.</p> <p>... ...</p> <p>Bit #31 = 0.</p> <p>Information, whether the addressed board is operated as a master, slave or single board:</p> <p>= 0 Single board.</p> <p>= 1 Slave without any further downstream slave board.</p> <p>= 2 Master.</p> <p>= 3 Slave together with a downstream slave board.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 113</a>.</li> <li><b>get_master_slave</b> only returns the status, if the addressed RTC5 board has been previously initialized by <a href="#">load_program_file</a>. Otherwise, 0 is returned.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 600, OUT 600, RBF 600.
References	<a href="#">sync_slaves</a>

<b>Ctrl Command</b>	<b>get_mcbsp</b>
Function	Returns the most recent input value that has been fully transferred by the <b>McBSP interface</b> to the memory location for Processing-on-the-fly applications.
Call	<code>mcbsp_value = get_mcbsp()</code>
Result	Input value. As a signed 32-bit value.
Comments	• <b>get_mcbsp</b> is equivalent to <b>read_mcbsp(0)</b> (see notes there).
RTC4→RTC5	New command.
Version info	–
References	<b>read_mcbsp</b>

<b>Undelayed Short List Command</b>	<b>get_mcbsp_list</b>
Function	No function.
Call	<code>get_mcbsp_list()</code>
Comments	• <b>get_mcbsp_list</b> has no effect on the user program.
RTC4→RTC5	New command.
Version info	–
References	<b>get_mcbsp</b>

<b>Ctrl Command</b>	<b>get_out_pointer</b>
Function	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.
Call	<code>get_out_pointer( &amp;ListNo, &amp;Pos )</code>
Returned parameter values	<p>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.</p> <p>Pos      Current (or most recent) position of the output pointer (relative memory address). As a pointer to an unsigned 32-bit value.</p>
Comments	• <b>get_out_pointer</b> calls <b>get_status</b> (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 <b>get_out_pointer</b> simplify comparing the output pointer position to the current input pointer position (particularly with respect to list consistency) during an alternative list change.
RTC4→RTC5	New command.
Version info	–
References	<b>get_status, get_input_pointer, get_list_pointer</b>



<b>Ctrl Command</b>	<b>get_overrun</b>
<b>Function</b>	Returns the number of overruns of the 10 $\mu$ s clock cycle 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 $\mu$ s clock cycle since the last call of <b>get_overrun</b> . 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>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	–

<b>Ctrl Command</b>	<b>get_RTC_mode</b>
<b>Function</b>	Returns the currently set operation mode of the RTC5 DLL.
<b>Call</b>	<code>DLLMode = get_RTC_mode()</code>
<b>Result</b>	DLL operation mode as a 32-bit value. = 4: <a href="#">RTC4 Compatibility Mode</a> . = 5: <a href="#">RTC5 Standard Mode</a> (default setting).
<b>Comments</b>	<ul style="list-style-type: none"> <li>• The <a href="#">RTC5 DLL</a> operation mode can be set by <a href="#">set_RTC4_mode</a> or <a href="#">set_RTC5_mode</a>. The default setting is <a href="#">RTC5 Standard Mode</a>.</li> <li>• <a href="#">get_RTC_mode</a> is available even without explicit access rights to a particular RTC5 Board.</li> <li>• <a href="#">get_RTC_mode</a> is not available as a multi-board command.</li> <li>• The board-specific error variables <a href="#">LastError</a> and <a href="#">AccError</a> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <a href="#">get_RTC_mode</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">set_RTC4_mode</a> , <a href="#">set_RTC5_mode</a>



<b>Ctrl Command</b>	<b>get_RTC_version</b>
<b>Function</b>	Returns the version number of the <b>FPGA</b> firmware ( <b>RTC5RBF.rbf</b> ) and information about enabled options of the RTC5 Board.
<b>Call</b>	<code>RTCVersion = get_RTC_version()</code>
<b>Result</b>	<p>Version number of the <b>FPGA</b> firmware and additional information. As an unsigned 32-bit value.</p> <p>Bit #0 (LSB)      Version number of the <b>FPGA</b> firmware (<b>RTC5RBF.rbf</b>). ... Bit #7 Bit #8      = 1: <b>Option Processing-on-the-fly</b> is enabled.                   See also <b>Chapter 8.6 "Processing-on-the-fly"</b>, page 227. Bit #9      = 1: <b>Option "Second Scan Head Control"</b> is enabled.                   See also <b>Section "2. SCANHEAD Socket Connector"</b>, page 55. Bit #10     = 1: <b>Option "3D"</b> is enabled.                   See also <b>Chapter 8.5.2 "3D Scan Systems"</b>, page 222. Bit #11     Reserved. Bit #12     Reserved. Bit #13     Reserved. Bit #14     Reserved. Bit #15     Reserved. Bit #16     <b>DSP</b> version number. ... Bit #23 Bit #24     Subversion number of the <b>FPGA</b> firmware (<b>RTC5RBF.rbf</b>). ... Bit #31</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The <b>FPGA</b> firmware version numbers are in the range 500...599.</li> <li><code>get_RTC_version( Bit #0...Bit #7 )</code> returns values in the range 0...99 (version number – 500).</li> <li>The current <b>DSP</b> version number is 2 (0 and 1 are older versions).</li> <li>The current <b>FPGA</b> firmware subversion is 0.</li> <li>The <b>FPGA</b> firmware version can even be returned after an <b>RTC5_VERSION_MISMATCH</b> or <b>RTC5_ACCESS_DENIED</b> error. The return value is 0 if no program has yet been loaded.</li> <li>The board-specific error variables <b>LastError</b> and <b>AccError</b>, see <b>Section "Error Handling"</b>, page 119, are neither generated nor altered by <code>get_RTC_version</code>.</li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<a href="#">get_hex_version</a> , <a href="#">get_dll_version</a>

<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 <code>get_serial</code> usage, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 196</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 RTC5 Board.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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 RTC5 Board.
<b>Call</b>	<code>RTCSerialNumber = get_serial_number()</code>
<b>Result</b>	RTC5 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 <code>init_rtc5_dll</code> and cached in the DLL, from where you can query them by <code>get_serial_number</code>.</li> <li><code>get_serial_number</code> is helpful when using several RTC5 Boards in one computer (see <a href="#">Chapter 6.6 "Using Several RTC5 PCI Boards in One PC", page 112</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 RTC5 DLL-internal numbers assigned to them during initialization. The RTC5 DLL-internal numbers are newly assigned during each initialization of a user program (see <code>init_rtc5_dll</code>) 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 <code>rtc5_count_cards</code>.</li> <li>The <code>get_serial_number</code> and <code>n_get_serial_number</code> commands are even available without explicit access rights to a specific RTC5 Board.</li> <li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <code>get_serial_number</code>.</li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<a href="#">rtc5_count_cards</a>



<b>Ctrl Command</b>	<b>get_standby</b>
Call	get_standby( &HalfPeriod, &PulseLength )
Returned parameter values	HalfPeriod <i>Half of the currently set standby output period of the standby pulses.</i> As a pointer to an unsigned 32-bit value. 1 bit equals 1/64 $\mu$ s.
	PulseLength <i>Currently set pulse length of the standby pulses.</i> As a pointer to an unsigned 32-bit value. 1 bit equals 1/64 $\mu$ s.
Comments	<ul style="list-style-type: none"> <li>For <b>get_standby</b> usage, see <a href="#">Section "Signals for "Laser Standby" Operation", page 175.</a></li> </ul>
RTC4→RTC5	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 divides the values for <code>HalfPeriod</code> and <code>PulseLength</code> by 8.
Version info	–
References	<a href="#">set_standby</a>

Ctrl Command	get_startstop_info
Function	Provides information about internal and <b>External Starts</b> and <b>External Stops</b> since the last call of <b>get_startstop_info</b> . Also provided are the current <b>External Start</b> and <b>External Stop</b> levels, the status and signal level of the laser control signals, and possible transmission errors to and from the attached scan system.
Call	StartStopInfo = get_startstop_info()
Result	<p>Info signal. As an unsigned 32-bit value.</p> <p>Bit #0 = 1: An internal start has been executed (by <b>execute_list</b> or similar) since the last call of <b>get_startstop_info</b>.</p> <p>Bit #1 = 1: An <b>External Start</b> has been executed (by <b>/START</b>, <b>/START2</b>, <b>/Slave-START</b>, <b>simulate_ext_start</b> or <b>simulate_ext_start_ctrl</b>) since the last call of <b>get_startstop_info</b>.</p> <p>Bit #2 = 1: An internal stop has been executed (by <b>stop_execution</b>) since the last call of <b>get_startstop_info</b>.</p> <p>Bit #3 = 1: An <b>External Stop</b> has been executed (by <b>/STOP</b>, <b>/STOP2</b>, <b>/Slave-STOP</b> or <b>simulate_ext_stop</b>) since the last call of <b>get_startstop_info</b>.</p> <p>Bit #4 Ext-stop status (= logical AND operation of the signals <b>/STOP</b>, <b>/STOP2</b>, <b>/Slave-STOP</b> and <b>simulate_ext_stop</b>, see <b>Figure 71</b>): = 1: <i>No</i> 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 active-LOW (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 <b>External Start</b> has failed (more <b>External Starts</b> 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 <b>/START</b>, <b>/START2</b> and <b>/Slave-START</b>, see <b>Figure 71</b>): = 1: <i>No</i> 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 active-LOW (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 The 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>The info bits <b>Bit #0...Bit #3</b></li> <li>The error bits <b>Bit #16...Bit #31</b></li> </ul> </li> <li>See also <a href="#">Section "External Stop", page 265</a> and <a href="#">Section "External Start", page 266</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality for info bits that are also used on the RTC4.
Version info	Last change DLL 546: <b>Bit #14</b> .
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 <b>List Execution Status</b> values and the current (or most recent) position of the output pointer.																									
<b>Call</b>	get_status( &Status, &Pos )																									
Returned parameter values	<p>Status</p> <p>Status value. As a pointer to an unsigned 32-bit value..</p> <table> <tr> <td>Bit #0</td> <td>= 1: <b>BUSY list execution status</b> set. (<b>LSB</b>)</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 list execution status</b> 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 list execution status</b> set.</td> </tr> <tr> <td>Bit #16</td> <td>0.</td> </tr> <tr> <td>...</td> <td></td> </tr> <tr> <td>Bit #31</td> <td></td> </tr> </table>	Bit #0	= 1: <b>BUSY list execution status</b> set. ( <b>LSB</b> )	Bit #1	Reserved.	...		Bit #6		Bit #7	= 1: <b>INTERNAL-BUSY list execution status</b> set.	Bit #8	Reserved.	...		Bit #14		Bit #15	= 1: <b>PAUSED list execution status</b> set.	Bit #16	0.	...		Bit #31		
Bit #0	= 1: <b>BUSY list execution status</b> set. ( <b>LSB</b> )																									
Bit #1	Reserved.																									
...																										
Bit #6																										
Bit #7	= 1: <b>INTERNAL-BUSY list execution status</b> set.																									
Bit #8	Reserved.																									
...																										
Bit #14																										
Bit #15	= 1: <b>PAUSED list execution status</b> set.																									
Bit #16	0.																									
...																										
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 (cont'd)	<ul style="list-style-type: none"> <li>For a description of when the <b>BUSY list execution status</b>, <b>INTERNAL-BUSY list execution status</b> or <b>PAUSED list execution status</b> values are set or not set, see <b>Chapter 6.4.3 "List Execution Status"</b>, page 97.</li> <li>(<b>BUSY list execution status</b> and <b>PAUSED list execution status</b> set) requires <b>restart_list</b> for continuation, (<b>BUSY list execution status</b> not set and <b>PAUSED list execution status</b> set) requires <b>release_wait</b> and (both <b>BUSY list execution status</b> and <b>PAUSED list execution status</b> not set) requires <b>execute_list_pos</b>. "Continuation" is not allowed with (<b>BUSY list execution status</b> set and <b>PAUSED list execution status</b> not set) and a currently running list. An improper continuation generates the <b>get_last_error</b> return code <b>RTC5_BUSY</b>. With (<b>INTERNAL-BUSY list execution status</b> set), <b>release_wait</b> and <b>execute_list_pos</b> are only executed with a delay (after <b>INTERNAL-BUSY list execution status</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 position <b>Pos</b> is queried, then the position is returned of the list command in the list area ("List 1" or "List 2") in which the output pointer most recently resided (typically from where the subroutine has been 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 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 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 <b>load_list</b>).</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→RTC5	Basically unchanged functionality. However: The Parameter Status returns the <b>BUSY</b> list execution status, <b>INTERNAL-BUSY</b> list execution status and <b>PAUSED</b> list execution status with an RTC5, whereas only the <b>BUSY</b> list execution status with an RTC4.
Version info	–
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 outputs: the current statuses of the stepper motor signals, the currently defined CLOCK pulse period, the status "Busy" and status "Init", and the current values of the internal position variables.																														
<b>Restriction</b>	For older RTC5 Boards with DSP version numbers < 2 ( <a href="#">get_rtc_version</a> Bit #16...Bit #23), <a href="#">get stepper status</a> is neither executed nor replaced by <a href="#">list_nop</a> ( <a href="#">get_last_error</a> return code <a href="#">RTC5_TYPE_REJECTED</a> ).																														
<b>Call</b>	<code>get stepper status( &amp;Status1, &amp;Pos1, &amp;Status2, &amp;Pos2 )</code>																														
<b>Returned parameter values</b>	<table> <tr> <td>Status1</td> <td>Current status of stepper motor output 1. As a pointer to an unsigned 32-bit value.</td> </tr> <tr> <td>    Bit #0</td> <td>ENABLE signal. (LSB)</td> </tr> <tr> <td>    Bit #1</td> <td>DIRECTION signal.</td> </tr> <tr> <td>    Bit #2</td> <td>CLOCK signal.</td> </tr> <tr> <td>    Bit #3</td> <td>SWITCH signal = limit switch signal.</td> </tr> <tr> <td>    Bit #4</td> <td>Status "Busy".</td> </tr> <tr> <td>    Bit #5</td> <td>Status "Init".</td> </tr> <tr> <td>    Bit #6</td> <td>Reserved.</td> </tr> <tr> <td>    Bit #7</td> <td>Reserved.</td> </tr> <tr> <td>    Bit #8</td> <td>CLOCK pulse period (24-bit value).</td> </tr> <tr> <td>    ...</td> <td></td> </tr> <tr> <td>    Bit #31</td> <td></td> </tr> </table> <table> <tr> <td>Pos1</td> <td>Current value of the internal position variable for stepper motor output port 1. As a pointer to a signed 32-bit value.</td> </tr> <tr> <td>Status2</td> <td>Current status of stepper motor output port 1. Otherwise, like Status1.</td> </tr> <tr> <td>Pos2</td> <td>Like Pos1.</td> </tr> </table>	Status1	Current status of stepper motor output 1. As a pointer to an unsigned 32-bit value.	Bit #0	ENABLE signal. (LSB)	Bit #1	DIRECTION signal.	Bit #2	CLOCK signal.	Bit #3	SWITCH signal = limit switch signal.	Bit #4	Status "Busy".	Bit #5	Status "Init".	Bit #6	Reserved.	Bit #7	Reserved.	Bit #8	CLOCK pulse period (24-bit value).	...		Bit #31		Pos1	Current value of the internal position variable for stepper motor output port 1. As a pointer to a signed 32-bit value.	Status2	Current status of stepper motor output port 1. Otherwise, like Status1.	Pos2	Like Pos1.
Status1	Current status of stepper motor output 1. As a pointer to an unsigned 32-bit value.																														
Bit #0	ENABLE signal. (LSB)																														
Bit #1	DIRECTION signal.																														
Bit #2	CLOCK signal.																														
Bit #3	SWITCH signal = limit switch signal.																														
Bit #4	Status "Busy".																														
Bit #5	Status "Init".																														
Bit #6	Reserved.																														
Bit #7	Reserved.																														
Bit #8	CLOCK pulse period (24-bit value).																														
...																															
Bit #31																															
Pos1	Current value of the internal position variable for stepper motor output port 1. As a pointer to a signed 32-bit value.																														
Status2	Current status of stepper motor output port 1. Otherwise, like Status1.																														
Pos2	Like Pos1.																														
<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 260</a>.</li> <li>If the SWITCH signal (limit switch signal) is set to (Pin is LOW), no more CLOCK pulses are generated.</li> <li>Status "Busy" indicates that a previously initiated (by <a href="#">stepper_abs</a>, <a href="#">stepper_rel</a>, etc.) set-position motion has not yet completed.</li> <li>Status "Init" indicates that a previously initiated (by <a href="#">stepper_init</a>) reference motion has not yet completed.</li> </ul>																														
RTC4→RTC5	New command.																														
Version info	–																														
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 = <b>get_sub_pointer( Index )</b>
<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>• <b>get_sub_pointer</b> 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 depends on whether the subroutine has been loaded into the protected list memory area "List 3" or an unprotected subroutine has been only subsequently referenced.</li> <li>• If <b>Index &gt; 1023</b> or if no subroutine has been referenced with the specified index, then <b>get_sub_pointer</b> returns the value "-1" (for example, <math>2^{32}-1</math>).</li> <li>• <b>get_sub_pointer</b> 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→RTC5	New command.
	—
<b>References</b>	<a href="#">get_char_pointer</a> , <a href="#">get_text_table_pointer</a>



<b>Ctrl Command</b>	<b>get_sync_status</b>
<b>Function</b>	Returns the master/slave synchronization status of the addressed RTC5 Board.
<b>Call</b>	<code>MasterSlaveSyncStatus = get_sync_status()</code>
<b>Result</b>	<p>Master/slave synchronization status [0...640].  As an unsigned 32-bit value.</p> <p>&lt; 11: The board is synchronized to the master board (or to the preceding board in the master/slave chain).</p> <p>= 640: The addressed board is operated as master.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <code>get_sync_status</code> usage, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 113</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>First synchronize the boards of the master/slave chain by the <code>sync_slaves</code> command before using <code>get_sync_status</code>.</li> <li>To ensure that <code>get_sync_status</code> actually returns the current master/slave synchronization status, you should first have already triggered an <a href="#">External Start</a> for the master board by an external start signal or by <code>simulate_ext_start_ctrl</code> (see <a href="#">Section "External Start", page 266</a>). This start then initiates measurement of the time differences between each board's /SLAVE start pulse and the corresponding 10 <math>\mu</math>s clock. This time difference gets stored on each board as the master/slave synchronization status (in units of 15 ns) and remains stored there until the a new <a href="#">External Start</a> 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 113</a>): master/slave synchronization status &lt; 11. The master/slave synchronization status is typically 7.</li> <li>In a not-explicitly-synchronized state (prior to <code>sync_slaves</code>), the master/slave synchronization status might coincidentally be &lt; 11. If so, then the boards behave as if synchronized.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">sync_slaves</a> , <a href="#">get_master_slave</a>



<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>	<p>TableNo      Number of the currently loaded correction table.            As an unsigned 32-bit value.            Allowed values: [1...4]. See also <a href="#">number_of_correction_tables</a>.</p> <p>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>.</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 <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>RTC5_PARAM_ERROR</b>).</li> <li>If no correction table with the specified number has been loaded, then the parameter values are undefined.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<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>• <b>get_text_table_pointer</b> 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 unprotected list memory area depends on whether the text string has been loaded into the protected list memory area "List 3" or an unprotected subroutine has been only subsequently referenced.</li> <li>• If <b>Index</b> &gt; 41 or if no text string has been 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>• <b>get_text_table_pointer</b> 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→RTC5	New command.
Version info	–
References	<a href="#">get_char_pointer</a> , <a href="#">get_sub_pointer</a>

<b>Ctrl Command</b>	<b>get_time</b>
<b>Function</b>	Returns the <b>RTC5 Timer</b> value (without resetting it to zero). It has been stored during the most recent call of <b>save_and_restart_timer</b> .
<b>Call</b>	TimerValue = get_time()
<b>Result</b>	<b>RTC5 Timer</b> 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→RTC5	Unchanged functionality.
Version info	–
References	<a href="#">get_lap_time</a> , <a href="#">save_and_restart_timer</a>

<b>Ctrl Command</b>	<b>get_transform</b>																											
<b>Function</b>	Transfers to the PC the position values that were recorded by <a href="#">set_trigger</a> and stored on the RTC5. 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>20</sup>] of to-be-backward-transformed position values. The measured values with index 0 to (Number-1) are backward transformed. As an unsigned 32-bit value.</p> <p>Ptr1      Pointers (in C and C++ data type ULONG_PTR, an unsigned 32-bit value or unsigned 64-bit value) to the two areas of PC main memory to which the backward transformed values should be transferred (see also <code>Code</code>).</p> <p>Ptr2      Pointers (in C and C++ data type ULONG_PTR, an unsigned 32-bit value or unsigned 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>.</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 <a href="#">transform</a> (<code>Ptr1</code> corresponds to <code>Sig1</code> and <code>Ptr2</code> to <code>Sig2</code>).</p> <p>For Bit #0 = 0, the measurement value pairs recorded by <a href="#">set_trigger</a> are backward transformed as XY coordinates:</p> <table> <tr> <td>Bit #1</td> <td>= 0:</td> <td>Values recorded by measurement channel 1 (<code>Signal1</code>) are transferred to the PC using <code>Ptr1</code> and then backward transformed as X coordinates. Values recorded by measurement channel 2 (<code>Signal2</code>) are transferred to the PC using <code>Ptr2</code> and then backward transformed as Y coordinates.</td> </tr> <tr> <td></td> <td>= 1:</td> <td>Values recorded by measurement channel 1 (<code>Signal1</code>) are transferred to the PC using <code>Ptr1</code> and then backward transformed as Y coordinates. Values recorded by measurement channel 2 (<code>Signal2</code>) are transferred to the PC using <code>Ptr2</code> and then backward transformed as X coordinates.</td> </tr> <tr> <td>Bit #2</td> <td>= 0:</td> <td>Gain/offset correction of automatic self-calibration is backward transformed.</td> </tr> <tr> <td>Bit #3</td> <td>= 0:</td> <td>The <a href="#">Image field</a> 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></td> <td>Reserved.</td> </tr> <tr> <td></td> <td>...</td> <td></td> </tr> <tr> <td></td> <td>Bit #31</td> <td></td> </tr> </table>	Bit #1	= 0:	Values recorded by measurement channel 1 ( <code>Signal1</code> ) are transferred to the PC using <code>Ptr1</code> and then backward transformed as X coordinates. Values recorded by measurement channel 2 ( <code>Signal2</code> ) are transferred to the PC using <code>Ptr2</code> and then backward transformed as Y coordinates.		= 1:	Values recorded by measurement channel 1 ( <code>Signal1</code> ) are transferred to the PC using <code>Ptr1</code> and then backward transformed as Y coordinates. Values recorded by measurement channel 2 ( <code>Signal2</code> ) are transferred to the PC using <code>Ptr2</code> and then backward transformed as X coordinates.	Bit #2	= 0:	Gain/offset correction of automatic self-calibration is backward transformed.	Bit #3	= 0:	The <a href="#">Image field</a> 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:	Values recorded by measurement channel 1 ( <code>Signal1</code> ) are transferred to the PC using <code>Ptr1</code> and then backward transformed as X coordinates. Values recorded by measurement channel 2 ( <code>Signal2</code> ) are transferred to the PC using <code>Ptr2</code> and then backward transformed as Y coordinates.																										
	= 1:	Values recorded by measurement channel 1 ( <code>Signal1</code> ) are transferred to the PC using <code>Ptr1</code> and then backward transformed as Y coordinates. Values recorded by measurement channel 2 ( <code>Signal2</code> ) are transferred to the PC using <code>Ptr2</code> and then backward transformed as X coordinates.																										
Bit #2	= 0:	Gain/offset correction of automatic self-calibration is backward transformed.																										
Bit #3	= 0:	The <a href="#">Image field</a> 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	get_transform
Parameters (cont'd)	<p>Code (cont'd) 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> <p>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>.</p> <p>= 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>.</p> <p>Bit #2 = 0: The offset to the focal length defined by <b>set_defocus</b> or <b>set_defocus_list</b> is backward transformed.</p> <p>Bit #3 = 0: The ABC correction is backward transformed.</p> <p>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.</p> <p>Bit #5 Reserved.</p> <p>...</p> <p>Bit #31 As an unsigned 32-bit value.</p>
Comments	<ul style="list-style-type: none"> <li>For backward transformation backward transformation of position values see Chapter 8.1.3 "Monitoring the Positioning", page 200.</li> <li><b>get_transform</b>(Number, <code>Ptr1</code>, <code>Ptr2</code>, <code>Ptr</code>, <code>Code</code>) transfers to the PC the data pairs recorded by <b>set_trigger</b> by executing <b>get_waveform</b>(1, Number, <code>Ptr1</code>) and <b>get_waveform</b>(2, Number, <code>Ptr2</code>) and overwrites the data pairwise by using <b>transform</b>(<code>Sig1</code>, <code>Sig2</code>, <code>Ptr</code>, <code>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 has been 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</code> <math>\times</math> 4 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 (Forts.)	<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 transformation is performed (<code>get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>). The same applies for <code>Number = 0</code> or <code>Number &gt; 2<sup>20</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>RTC5_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 has been assigned at the point in time 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>RTC5_SEND_ERROR</code>.</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>Even in the <b>RTC4 Compatibility Mode</b>, all coordinate values transferred to the PC are in the RTC5 20-bit range. The backward transformed coordinate values must, if necessary, be divided by 16 by users themselves.</p>
Version info	–
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>RTC5_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 <b>iDRIVE</b> scan system (see Glossary entry on <a href="#">page 23</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.</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 RTC5 (<code>Status&lt;AX...BY&gt;</code>), then be mindful of the returned data type's value range when evaluating it (see <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 <b>XY2-100 status word</b> or the serial number) and returned to the RTC5 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 sometimes (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 RTC5 as signed 20-bit values (for example, actual positions or actual speeds) can be safely evaluated as a complete signed 32-bit value returned by <b>get_value</b> (see also code example below).</li> <li>– Although z values must be supplied as a 16 bit values in 3D command parameters, SCANLAB Z axes and 3-axis scan systems (and correspondingly <b>get_value</b>) return z values always as signed 20-bit values.</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→RTC5	<p>Basically unchanged functionality. However:</p> <p>Even in <b>RTC4 Compatibility Mode</b>, all returned values are in the RTC5 20-bit range, but are transferred to the PC as 32-bit values (see above).</p>
Version info	–
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 4 specified signals.
<b>Call</b>	<code>get_values( SignalPtr, ResultPtr )</code>
<b>Parameters</b>	<p>SignalPtr    Pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit value or unsigned 64-bit value) to an array of 4 unsigned 32-bit values, where the desired signal types are specified.</p> <p>ResultPtr    Pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit value or unsigned 64-bit value) to an array of 4 signed 32-bit values, where the current values of the up to 4 specified signals are to be stored.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Up to 4 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 desired 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 has been specified by <code>SignalPtr</code>. A <a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code> is only generated if all 4 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>RTC5_PARAM_ERROR</code> is generated.</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>Even in <a href="#">RTC4 Compatibility Mode</a>, the 4 returned values are in the RTC5 20-bit range, but are transferred to the PC as 32-bit values (see comments for <a href="#">get_value</a>).</p>
<b>Version info</b>	–
<b>References</b>	<a href="#">get_value</a> , <a href="#">set_trigger</a> , <a href="#">transform</a>



<b>Ctrl Command</b>	<b>get_wait_status</b>
<b>Function</b>	Returns the wait state of the RTC5.
<b>Call</b>	<code>WaitStatus = get_wait_status()</code>
<b>Result</b>	Wait state. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"><li>If processing has stopped at a wait marker, <code>get_wait_status</code> returns the number of this marker. See <a href="#">set_wait</a>.</li><li>If no wait marker has been reached, <code>get_wait_status</code> returns zero.</li><li>Processing of the list can be resumed by calling <a href="#">release_wait</a>.</li></ul>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<a href="#">set_wait</a> , <a href="#">release_wait</a>



<b>Ctrl Command</b>	<b>get_waveform</b>
<b>Function</b>	Transfers to the PC the data that has been measured and stored onto the RTC5 by <b>set_trigger</b> or <b>set_trigger4</b> .
<b>Call</b>	<code>get_waveform( Channel, Number, Ptr )</code>
<b>Parameters</b>	<p>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.</p> <p>Number      Number [1...<math>2^{20}</math> for <b>set_trigger</b>, 1...<math>2^{19}</math> for <b>set_trigger4</b>] of measured values to transfer. Values of measurement positions 0 to (Number-1) are transferred. As an unsigned 32-bit value.</p> <p>Ptr      Pointer (data type <b>ULONG_PTR</b> in C and C++, an unsigned 32-bit or unsigned 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>In the following cases, no data is transferred (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>): <ul style="list-style-type: none"> <li>For Number = 0.</li> <li>For Number &gt; <math>2^{20}</math>.</li> <li>For Number &gt; <math>2^{19}</math> when Channel = 3 or 4 has been selected or if a file has been loaded as correction table No=3 or No=4 with <b>load_correction_file</b>.</li> <li>For Ptr = <b>NULL</b>.</li> </ul> </li> <li>PCI transmission errors generate the <b>get_last_error</b> return code <b>RTC5_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 has been 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>
<b>RTC4→RTC5</b>	Basically unchanged functionality. However: Even in <b>RTC4 Compatibility Mode</b> , all values are in the RTC5 20-bit range, but are transferred to the PC as 32-bit values (see comments for <b>get_value</b> ).
<b>Version info</b>	–
<b>References</b>	<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 <i>l</i> 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>	<code>ZDistance = get_z_distance( X, Y, Z )</code>						
<b>Parameters</b>	<table> <tr> <td>X</td> <td>Absolute coordinates of the point (x y z) in the <b>3D image field</b>. 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). Allowed value range: [-32,768...+32,767].</td> </tr> </table>	X	Absolute coordinates of the point (x y z) in the <b>3D image field</b> . 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). Allowed value range: [-32,768...+32,767].
X	Absolute coordinates of the point (x y z) in the <b>3D image field</b> . 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). Allowed value range: [-32,768...+32,767].						
<b>Result</b>	Focus length value. [-32,768...+32,767]. 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>.</li> <li>• The focus length value <i>l</i>: <ul style="list-style-type: none"> <li>– Has no dimension</li> <li>– Corresponds to the focus length difference between the specified point (x y z) and the point (0 0 0)</li> <li>– Can be positive or negative</li> </ul> </li> <li>• With the RTC5, <b>ZDistance</b> is always in 16-bit range [-32,768...+32,767]: <ul style="list-style-type: none"> <li>– In <b>RTC4 Compatibility Mode</b></li> <li>– In <b>RTC5 Standard Mode</b></li> </ul> </li> <li>• <b>get_z_distance</b> first performs a (virtual) jump to the point (x y z) 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>RTC5_BUSY</b>).</li> <li>• <b>get_z_distance</b> is not executed and returns 0 (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status 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> list execution status set)</li> </ul> </li> <li>• For <b>3D image field</b> calibration, see <b>Chapter "3D Commands", page 223</b>.</li> </ul>						
<b>RTC4→RTC5</b>	Unchanged functionality. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified value for X and Y by 16. The allowed value range decreases accordingly. The value range for Z is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .						
<b>Version info</b>	–						
<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 <b>Image field</b> .
<b>Call</b>	<code>goto_xy( X, Y )</code>
<b>Parameters</b>	<p><b>X</b>      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.</p> <p><b>Y</b>      Like <b>X</b> (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the jump speed has not been previously explicitly set by <b>set_jump_speed</b> or <b>set_jump_speed_ctrl</b>, then the jump is executed at a predefined jump speed of 10000 <i>bits/ms</i>.</li> <li><b>goto_xy</b> (unlike the list commands <b>jump_abs</b> and <b>jump_rel</b>) has no effect on the laser control signals and also does not set a <b>Jump Delay</b>.</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 211</a>.</li> <li><b>goto_xy</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if:           <ul style="list-style-type: none"> <li>the <b>BUSY list execution status</b> is set</li> <li>the <b>INTERNAL-BUSY list execution status</b> is set</li> <li>an external stop signal (/STOP, /STOP2 or /Slave STOP) is present            It can also be generated by automatic monitoring, see <b>set_laser_control</b> and <b>range_checking</b>.</li> </ul> </li> <li><b>goto_xy</b> is even executed, if:           <ul style="list-style-type: none"> <li>a list has been paused by <b>set_wait</b> (<b>PAUSED list execution status</b> set)</li> </ul> </li> <li>The <b>INTERNAL-BUSY list execution status</b> is set while <b>goto_xy</b> is executed.</li> <li><b>goto_xy</b> only returns to the user program when the motion has been completed.</li> </ul>
<b>RTC4→RTC5</b>	<ul style="list-style-type: none"> <li>Extended range of values.</li> <li><b>goto_xy</b> returns only after motion has been executed (see comments above).</li> <li>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16.            The allowed value range decreases accordingly.</li> </ul>
<b>Version info</b>	–
<b>References</b>	<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 a <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 <b>Microsteps</b> 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). Allowed value range: [-32,768...+32,767].</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). Allowed value range: [-32,768...+32,767].
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). Allowed value range: [-32,768...+32,767].						
<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 the 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>See also <b>Chapter 7.3.6 "Output Values to the Scan System", page 170</b>.</li> </ul>						
RTC4→RTC5	<ul style="list-style-type: none"> <li>Extended range of values.</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>. The value range for <b>Z</b> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b>.</li> </ul>						
<b>Version info</b>	–						
<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 axis and y axis) and defines the home position.
<b>Call</b>	<code>home_position( XHome, YHome )</code>
<b>Parameters</b>	<p>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.</p> <p>YHome      Like XHome (analogously).</p>
<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. Accordingly, a home return to the last valid position is then executed again at start. The home jump and home return are executed at jump speed.</li> <li>• <b>home_position</b> is intended for a laser system that does not allow fast switching of the laser. After calling the command, 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 list execution status</b> is set.</li> <li>• A <b>beam dump</b> should be placed in the home position.</li> <li>• The home jump mode is deactivated by <code>home_position(0, 0)</code>, even if it has been activated by <b>home_position_xyz</b>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for XHome and YHome by 16. The allowed value range decreases accordingly.
<b>Version info</b>	–
<b>References</b>	<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 axis, y axis 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 <b>Microsteps</b> 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). Allowed value range: [-32,768...+32,767].</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). Allowed value range: [-32,768...+32,767].
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). Allowed value range: [-32,768...+32,767].						
<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>– <b>home_position( 0, 0 )</b></li> <li>– <b>home_position_xyz( 0, 0, 0 )</b></li> </ul> </li> <li>The <b>DirectMove3D</b> parameter of the <b>set_delay_mode</b> command determines the type of z axis motion (linear or with stepwise correction).</li> </ul>						
<b>RTC4→RTC5</b>	New command. <b>RTC4 Compatibility Mode:</b> see <b>home_position</b> . The value range for ZHome is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .						
<b>Version info</b>	–						
<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 <b>IOvalue</b> 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 <b>Mask1</b> are 1 and the bits specified in <b>Mask0</b> are 0). Otherwise, this list command is skipped.
<b>Call</b>	<b>if_cond( Mask1, Mask0 )</b>
<b>Parameters</b>	<p><b>Mask1</b> 16-bit mask.  As an unsigned 32-bit value.  Only the lower 16 bits are evaluated.</p> <p><b>Mask0</b> See <b>Mask1</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 271</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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 <b>Mode</b> for the x axis has been fulfilled. Otherwise, this list command is skipped.
<b>Call</b>	<b>if_fly_x_overflow( Mode )</b>
<b>Parameters</b>	<p><b>Mode</b> To-be-evaluated condition.  As a signed 32-bit value.</p> <p>= 0: Some kind of boundary exceedance occurred  (error bit <b>Bit #4</b> = 1 or error bit <b>Bit #5</b> = 1).</p> <p>&gt; 0: An overflow occurred (error bit <b>Bit #5</b> = 1).</p> <p>&lt; 0: An underflow occurred (error bit <b>Bit #4</b> = 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 <b>Mode</b> (error bit <b>Bit #4</b> and/or error bit <b>Bit #5</b>) are reset.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">clear_fly_overflow_ctrl</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> <b>if_fly_y_overflow</b> immediately executes the directly subsequent list command, if the condition in accordance with <code>Mode</code> for the y axis has been fulfilled. Otherwise, this list command is skipped.
<b>Call</b>	<code>if_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 <code>Bit #6</code> = 1 or error bit <code>Bit #7</code> = 1). > 0: An overflow occurred (error bit <code>Bit #7</code> = 1). < 0: An underflow occurred (error bit <code>Bit #6</code> = 1).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <code>if_fly_y_overflow</code>, 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 <code>Bit #6</code> and/or error bit <code>Bit #7</code>) are reset.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">clear_fly_overflow_ctrl</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> <b>if_fly_z_overflow</b> immediately executes the directly subsequent list command, if the condition in accordance with <code>Mode</code> for the z axis has been fulfilled. Otherwise, this list command is skipped.
<b>Call</b>	<code>if_fly_z_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 <code>Bit #24</code> = 1 or error bit <code>Bit #25</code> = 1). > 0: An overflow occurred (error bit <code>Bit #25</code> = 1). < 0: An underflow occurred (error bit <code>Bit #24</code> = 1).
<b>Comments</b>	<ul style="list-style-type: none"> <li>For usage of <code>if_fly_z_overflow</code>, 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 <code>Mode</code> (error bit <code>Bit #24</code> and/or error bit <code>Bit #25</code>) are reset.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits_z</a> , <a href="#">clear_fly_overflow</a> , <a href="#">clear_fly_overflow_ctrl</a> , <a href="#">if_not_fly_z_overflow</a>



Undelayed Short List Command	<b>if_not_activated</b>
Function	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 is executed immediately, otherwise it is skipped.
Call	<code>if_not_activated()</code>
Comments	<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> has been called.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>activate_fly_2d</b> , <b>activate_fly_2d_encoder</b> , <b>activate_fly_xy</b> , <b>activate_fly_xy_encoder</b> , <b>get_marking_info</b>



<b>Undelayed Short List Command</b>	<b>if_not_cond</b>
<b>Function</b>	<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 the following condition</i>:</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, this list command is skipped.</p>
<b>Call</b>	<code>if_not_cond( Mask1, Mask0 )</code>
<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 271</a>.</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<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 <b>Mode</b> for the x axis is <i>not</i> fulfilled. Otherwise, this list command is skipped.
<b>Call</b>	<b>if_not_fly_x_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 <b>Bit #4</b> = 1 or error bit <b>Bit #5</b> = 1). > 0: An overflow occurred (error bit <b>Bit #5</b> = 1). < 0: An underflow occurred (error bit <b>Bit #4</b> = 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 <b>Mode</b> (error bit <b>Bit #4</b> and/or error bit <b>Bit #5</b>) are reset.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">clear_fly_overflow_ctrl</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 <b>Mode</b> for the y axis is <i>not</i> fulfilled. Otherwise, this list command is skipped.
<b>Call</b>	<b>if_not_fly_y_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 <b>Bit #6</b> = 1 or error bit <b>Bit #7</b> = 1). > 0: An overflow occurred (error bit <b>Bit #7</b> = 1). < 0: An underflow occurred (error bit <b>Bit #6</b> = 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 <b>Mode</b> (error bit <b>Bit #6</b> and/or error bit <b>Bit #7</b>) are reset.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits</a> , <a href="#">clear_fly_overflow</a> , <a href="#">clear_fly_overflow_ctrl</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 <code>Mode</code> for the z axis has not been fulfilled. Otherwise, this list command is skipped.
<b>Call</b>	<code>if_not_fly_z_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 <b>Bit #24</b> = 1 or error bit <b>Bit #25</b> = 1). > 0: An overflow occurred (error bit <b>Bit #25</b> = 1). < 0: An underflow occurred (error bit <b>Bit #24</b> = 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", page 240</a>.</li> <li>The error bits queried in accordance with <code>Mode</code> (error bit <b>Bit #24</b> and/or error bit <b>Bit #25</b>) are reset.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">get_marking_info</a> , <a href="#">set_fly_limits_z</a> , <a href="#">clear_fly_overflow</a> , <a href="#">clear_fly_overflow_ctrl</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 <code>IOvalue</code> at the LASER connector's 2-bit digital input port <i>does not meet</i> the following condition: $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})$ (= if the bits specified in <code>Mask1</code> are <i>not 1</i> or the bits specified in <code>Mask0</code> are <i>not 0</i> ). Otherwise, this list command is skipped.
<b>Call</b>	<code>if_not_pin_cond( Mask1, Mask0 )</code>
<b>Parameters</b>	<code>Mask1</code> 2-bit mask. As an unsigned 32-bit value. Only the lower 2 bits are evaluated. <code>Mask0</code> See <code>Mask1</code> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 271</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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></p> <p><b>if_pin_cond</b> immediately executes the directly following list command, if the current IOvalue 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 Mask1 are 1 and the bits specified in Mask0 are 0). Otherwise, this list command is skipped.</p>
Call	<code>if_pin_cond( Mask1, Mask0 )</code>
Parameters	<p>Mask1      2-bit mask.            As an unsigned 32-bit value.            Only the lower 2 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 271</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 )</code>
<b>Parameters</b>	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.
	OffsetY      Like OffsetX (analogously).
<b>Comments</b>	<ul style="list-style-type: none"> <li>The final encoder value for the 2D correction (that is, current encoder value + reference value) must not exceed the allowed range. Otherwise, clipping occurs (see also <b>load_fly_2d_table</b>).</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<b>set_fly_2d, load_fly_2d_table, get_fly_2d_offset</b>

<b>Ctrl Command</b>	<b>init_rtc5_dll</b>
<b>Function</b>	Initializes control of the installed RTC5 Boards for a user program.
<b>Call</b>	<code>InitErrorNo = init_rtc5_dll()</code>
<b>Result</b>	<p>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>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <i>init_rtc5_dll</i> must be called before starting each user program. Otherwise, no RTC5s are accessible by the user program. <i>init_rtc5_dll</i> detects all available RTC5 Boards and establishes corresponding board management. If no RTC5 Boards are found, the error code <a href="#">RTC5_NO_CARD</a> is returned.</li> <li>• If an error occurs when reading the PCI configuration register (<a href="#">get_last_error</a> return code <a href="#">RTC5_CONFIG_ERROR</a>), then an <a href="#">RTC5_ACCESS_DENIED</a> error is also generated. The board then remains inaccessible until the error gets cleared by a PC restart.</li> <li>• The <i>init_rtc5_dll</i> command automatically assigns the user program access rights to the found boards (as by an <a href="#">acquire_rtc</a> command) 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 <a href="#">RTC5_ACCESS_DENIED</a> error code by <i>init_rtc5_dll</i>. The boards are then only accessible by these applications through RTC5 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_rtc5_dll</a> before the board can be used by another user program by <i>init_rtc5_dll</i> or <a href="#">acquire_rtc</a>. An <a href="#">RTC5_ACCESS_DENIED</a> error code is returned if access has been denied by at least one of the found boards. Which board(s) denied access can be determined by <a href="#">n_get_error</a>( CardNo ) (CardNo from 1 to the number of found boards) or directly after <i>init_rtc5_dll</i> (before the next command) by <a href="#">n_get_last_error</a>( CardNo ).</li> <li>• If a board is acquired by <i>init_rtc5_dll</i> (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 <a href="#">RTC5_ACCESS_DENIED</a> <a href="#">RTC5_VERSION_MISMATCH</a>).</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_RTC5_dll
Comments (cont'd)	<ul style="list-style-type: none"> <li>If a user program calls the <code>init_RTC5_dll</code> command multiple times, then the RTC5 board management created by the prior call is deleted for this user program and the originally-assigned access rights canceled. RTC5 board management is then newly created and access rights newly assigned.</li> <li>Each initialization of a user program by <code>init_RTC5_dll</code> causes the <b>RTC5 DLL</b>-internal numbers for all found RTC5 Boards to be newly reassigned. The relationship between the <b>RTC5 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>RTC5 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>RTC5 DLL</b>-internal number (see <code>rtc5_count_cards</code>) is the active board, in which case only <b>RTC5 DLL</b>-internal commands that require no access rights can be used.</li> <li>Also observe <a href="#">Chapter 6.7.1 "Board Acquisition by a User Program", page 117</a>.</li> <li><code>init_RTC5_dll</code> is available even without explicit access rights to any particular RTC5 Board.</li> <li><code>init_RTC5_dll</code> is not available as a multi-board command.</li> <li>After initialization of the <b>RTC5 DLL</b> with <code>init_RTC5_dll</code>, the <b>RTC5 Standard Mode</b> is set by default. After that, a different <b>RTC5 DLL</b> operational mode can be set, see <code>set_RTC4_mode</code> and <code>set_RTC5_mode</code>.</li> <li><code>init_RTC5_dll</code> does not trigger an initialization of RTC5 boards. Only <code>load_program_file</code> performs an initialization of RTC5 boards.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	–

Normal List Command	<b>jump_abs</b>
Function	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 <b>Image field</b> .
Call	<code>jump_abs( X, Y )</code>
Parameters	<p>X      Absolute x coordinate of the jump vector end point. In bits.  As a signed 32-bit value.  Allowed value range: [-8,388,608...+8,388,607].  Out-of-range values are clipped to the boundary values.  The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p>
Comments	<ul style="list-style-type: none"> <li>If the jump speed has not been previously explicitly set by <b>set_jump_speed</b> or <b>set_jump_speed_ctrl</b>, then the jump is executed at a predefined jump speed of 10000 <i>bits/ms</i>.</li> <li>The <b>Signals for "Laser Active" Operation</b> are switched off before the jump and remain off during the jump.</li> <li>After a <b>Jump command</b>, a (variable) <b>Jump Delay</b> is inserted. Exception: a zero-length jump vector's subsequent (variable) <b>Jump Delay</b> is not executed. However, the command itself still requires a 10 <math>\mu</math>s clock cycle 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 211</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the values specified for X and Y by 16. The allowed value range decreases accordingly.
Version info	–
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 a <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 <b>Microsteps</b> 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><b>X</b>      Absolute x coordinate of the jump vector end point. In bits.  As a signed 32-bit value.  Allowed value range: [-8,388,608...+8,388,607].  Out-of-range values are clipped to the boundary values.  The complete value range is only usable as a virtual <b>Image field</b> 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: [-32,768...+32,767].</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 the <b>set_delay_mode</b> command determines the type of z axis motion (linear or with stepwise correction).</li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16. The allowed value range decreases accordingly. The value range for <b>Z</b> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .
<b>Version info</b>	–
<b>References</b>	<b>jump_abs, jump_rel_3d, timed_jump_abs_3d</b>

<b>Normal List Command</b>	<b>jump_abs_drill</b>
<b>Comments</b>	<ul style="list-style-type: none"> <li>This command is described, for example, in the manual for the <b>intelliDRILL de II 20</b> Module.</li> </ul>

<b>Normal List Command</b>	<b>jump_abs_drill_2</b>
<b>Comments</b>	<ul style="list-style-type: none"> <li>This command is described, for example, in the manual for the <b>intelliDRILL de II 20</b> Module.</li> </ul>

Normal List Command	<b>jump_rel</b>
Function	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 <b>Image field</b> .
Call	<code>jump_rel( dx, dy )</code>
Parameters	<p><code>dx</code>      Relative x coordinate of the jump vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b>, for example, for (enabled) Processing-on-the-fly applications.</p> <p><code>dy</code>      Like <code>dx</code> (analogously).</p>
Comments	<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>jump_rel</b> is identical to <b>jump_abs</b> (see the comments there).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <code>dx</code> and <code>dy</code> by 16. The allowed value range decreases accordingly.
Version info	–
References	<a href="#">jump_abs</a> , <a href="#">jump_rel_3d</a> , <a href="#">timed_jump_rel</a>

Normal List Command	<b>jump_rel_3d</b>
Function	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 a <b>3D image field</b> .
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
Call	<b>jump_rel_3d( dx, dy, dz )</b>
Parameters	<p><b>dx</b>      Relative x coordinate of the jump vector end point. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual <b>Image field</b> 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: [-32,768...+32,767].</p>
Comments	<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>jump_rel_3d</b> is identical to <b>jump_abs_3d</b> (see the comments there).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>dx</b> and <b>dy</b> coordinates by 16. The allowed value range decreases accordingly. The value range for <b>dz</b> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .
Version info	–
References	<b>jump_abs_3d, jump_abs, jump_rel, timed_jump_rel_3d</b>

Normal List Command	<b>jump_rel_drill</b>
Comments	<ul style="list-style-type: none"> <li>This command is described, for example, in the manual for the <b>intelliDRILL de II 20</b> Module.</li> </ul>

Normal List Command	<b>jump_rel_drill_2</b>
Comments	<ul style="list-style-type: none"> <li>This command is described, for example, in the manual for the <b>intelliDRILL de II 20</b> Module.</li> </ul>

<b>Variable List Command</b>	<b>laser_on_list</b>
<b>Function</b>	Turns on the <b>Signals for "Laser Active" Operation</b> 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.            1 bit equals 10 µs.            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 <b>Comments</b> at <a href="#">laser_on_pulses_list</a>).</li> <li>The <b>Signals for "Laser Active" Operation</b> must first be selected with <b>set_laser_mode</b>, defined by further commands, and enabled by <b>set_laser_control</b> or <b>enable_laser</b> before they can be switched on with <b>laser_on_list</b> (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 <b>LaserOn Delay</b> is applied at the beginning of the programmed time interval: The laser control signals turn on after a <b>LaserOn Delay</b>.           <ul style="list-style-type: none"> <li>As with other <b>Mark Commands</b> (for example, <b>mark_abs</b>), 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, <b>jump_abs</b> or <b>list_nop</b>). The currently set <b>LaserOff Delay</b> is applied.</li> </ul> </li> <li><b>laser_on_list</b> is useful for marking single dots (see <a href="#">Chapter 7.1.3 "Marking Single Dots", page 129</a>).</li> <li>Wobbel Mode (see <a href="#">Chapter 8.4 "Wobbel Mode", page 217</a>) is retained but ignored.</li> <li>For Period = 0 the laser control signals do not turn on; then <b>laser_on_list</b> is a short list command.</li> <li><b>laser_on_list</b> does not trigger a scanner delay. To wait until the laser (after a <b>LaserOff Delay</b>) is actually off before a jump, then explicitly a <b>long_delay</b> should be inserted.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range.
Version info	–
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 signal for the specified number of external signal pulses, but for no longer than the specified time interval. For <code>Pulses &gt; 65,535</code> the function of <code>laser_on_pulses_list</code> is identical to <code>laser_on_list</code> .				
<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. 1 bit equals 10 <math>\mu</math>s. 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. 1 bit equals 10 $\mu$ s. 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. 1 bit equals 10 $\mu$ s. 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>• <code>laser_on_pulses_list</code> is useful for marking single dots in <code>Laser Mode 6</code>, 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 61</a>). By <code>set_laser_control</code>(Bit #5), you can specify whether signal pulses should be counted at rising or falling edges.</li> <li>• If <code>Period = 0</code>, then <code>laser_on_pulses_list</code> has no effect and <code>laser_on_pulses_list</code> becomes a short list command.</li> <li>• If <math>0 &lt; \text{Period} \leq (2^{31}-1)</math>, then <code>laser_on_pulses_list</code> duration is always <code>Period</code> clocks (that is, <code>Period</code> <math>\times</math> 10 <math>\mu</math>s), 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>, <code>laser_on_pulses_list</code> maximum duration is <math>(\text{Period} - 2^{31})</math> clocks (that is, <math>(\text{Period} - 2^{31}) \times 10 \mu</math>s). Here, however, <code>laser_on_pulses_list</code> terminates as soon as the specified number of external signal pulses has been detected.</li> <li>• If <code>Pulses &gt; 65,535</code>, then <code>laser_on_pulses_list</code>'s function is identical to <code>laser_on_list</code> (external signal pulses are not taken into account, see comments there). Otherwise (for <math>0 \leq \text{Pulses} \leq 65,535</math>), <code>laser_on_pulses_list</code> (in contrast to <code>laser_on_list</code>) do not toggle the laser control signals between "laser active" and "laser standby" operation, but instead only switches the LASERON signal, whereby <code>Laser Delays</code> are not taken into account. Likewise during this command, unexpired <code>Laser Delays</code> 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> </ul>				



Variable List Command	<code>laser_on_pulses_list</code>
Comments (cont'd)	<ul style="list-style-type: none"> <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 <code>set_laser_control</code>(Bit #5)) of the first external pulse (unless it is already on due to an unexpired <code>LaserOff Delay</code>) and remains on for the specified number of pulses, but no longer than until the end of the number of <math>10 \mu\text{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 <code>laser_on_pulses_list</code> 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> <li>If <code>Pulses = 0</code>, then <code>laser_on_pulses_list</code> has no effect (the LASERON signal is not switched on). Then <code>laser_on_pulses_list</code> becomes a short list command.</li> <li>The LASERON signal must first be defined and enabled by <code>set_laser_control</code> or <code>enable_laser</code> before it can be switched on with <code>laser_on_pulses_list</code> (see <a href="#">Chapter 7.4 "Laser Control", page 173</a>).</li> <li>While <code>laser_on_pulses_list</code> is executed, the set position of the scanners is not changed. The next list command is executed when the programmed time interval <code>Period</code> has passed.</li> <li>The Wobbel mode (see <a href="#">Chapter 8.4 "Wobbel Mode", page 217</a>) is retained but ignored.</li> <li>Any softstarts that were defined are not executed.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">laser_on_list</a> , <a href="#">para_laser_on_pulses_list</a>



<b>Ctrl Command</b>	<b>laser_signal_off</b>
<b>Function</b>	Turns off the <b>Signals for "Laser Active" Operation</b> immediately.
<b>Call</b>	<code>laser_signal_off()</code>
<b>Comments</b>	<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>RTC5_BUSY</b>), if:               <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</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> list execution status set)</li> <li>– the <b>INTERNAL-BUSY</b> list execution status is set</li> </ul> </li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<b>laser_signal_on</b>

<b>Normal List Command</b>	<b>laser_signal_off_list</b>
<b>Function</b>	Like <b>laser_signal_off</b> , but a list command.
<b>Call</b>	<code>laser_signal_off_list()</code>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	–



<b>Ctrl Command</b>	<b>laser_signal_on</b>
<b>Function</b>	Turns on the <b>Signals for "Laser Active" Operation</b> 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 <b>Signals for "Laser Active" Operation</b> must first be selected with <b>set_laser_mode</b>, defined by further commands, and enabled by <b>set_laser_control</b> or <b>enable_laser</b> before they can be switched on with <b>laser_signal_on</b> (see <a href="#">Chapter 7.4 "Laser Control", page 173</a>).</li> <li>• <b>laser_signal_on</b> is intended for turning on the laser directly, for example, for alignment purposes.</li> <li>• The <b>Signals for "Laser Active" Operation</b> must be turned off with <b>laser_signal_off</b>.</li> <li>• <b>laser_signal_on</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status is set</li> </ul> </li> <li>• <b>laser_signal_on</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> list execution status set)</li> </ul> </li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<a href="#">laser_signal_off</a>

<b>Normal List Command</b>	<b>laser_signal_on_list</b>
<b>Function</b>	Like <b>laser_signal_on</b> , but a list command and never ignored.
<b>Call</b>	<code>laser_signal_on_list()</code>
<b>RTC4→RTC5</b>	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<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	<code>list_call( Pos )</code>
Parameters	<code>Pos</code> Absolute jump address $[0\dots(2^{20}-1)]$ . 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 101</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 <code>Pos &gt; (2<sup>20</sup>-1)</code> or if <code>Pos</code> is also the current address (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>If a called subroutine executes a further <b>list_call</b> command 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). As of version OUT 540 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, if necessary, altered by <b>list_call</b> if the called address is in the list area ("List 1" or "List 2"). If this address is instead in protected memory ("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 <b>List Execution Status</b>, and <b>get_out_pointer</b> returns the position of the calling <b>list_call</b> command as the output pointer position.</li> <li>Absolute <b>Vector commands</b> and "<b>Arc</b>" <b>commands</b> execute absolutely after <b>list_call</b> is called. If the subroutine needs to execute at various locations within the <b>Image field</b>, then either the subroutine can only contain relative <b>[*]mark[*]</b> <b>Commands</b>, "<b>Arc</b>" <b>commands</b> or <b>Jump commands</b> 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 has been 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→RTC5	Essentially unchanged functionality.
Version info	–
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 list memory area address (in any desired area of list memory). In the called subroutine, any absolute <b>Vector commands</b> and <b>"Arc" commands</b> receive an offset (based on the current coordinates at the time of the call).
Call	<code>list_call_abs( Pos )</code>
Parameters	Pos      Absolute jump address [0...(2 <sup>20</sup> -1)]. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"> <li>The <b>list_call_abs</b> command is basically identical to <b>list_call</b> (see the comments there). However, <b>list_call_abs</b> results in a different execution of absolute <b>Vector commands</b> and <b>"Arc" commands</b> 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 <b>Vector commands</b> and <b>"Arc" commands</b> 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 <b>Vector commands</b> and <b>"Arc" commands</b> even if they are intended to be repeated in different parts of the <b>Image</b> 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→RTC5	New command.
Version info	–
References	<b>list_call</b> , <b>list_call_abs_cond</b> , <b>list_call_abs_repeat</b>

<b>Undelayed Short List Command</b>	<b>list_call_abs_cond</b>
<b>Function</b>	<i>Conditional subroutine call (AbsCall):</i> <b>list_call_abs_cond</b> executes <a href="#">list_call_abs</a> ( Pos ), 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 directly following list command is immediately executed.
<b>Call</b>	<code>list_call_abs_cond( Mask1, Mask0, Pos )</code>
<b>Parameters</b>	<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...(<math>2^{20}-1</math>)].  As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">list_call_abs</a>.</li> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 271</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">list_call_abs</a>

<b>Undelayed Short List Command</b>	<b>list_call_abs_repeat</b>
<b>Function</b>	Causes an unconditional jump to a subroutine that starts at the specified absolute list memory area address (in any desired area of list memory) and executes its body several times.
<b>Call</b>	<code>list_call_abs_repeat( Pos, Number )</code>
<b>Parameters</b>	<p>Pos      Absolute jump address [0...(<math>2^{20}-1</math>)].  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>
<b>Comments</b>	<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→RTC5	New command.
Version info	Available as of DLL 540, OUT 540.
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>(<i>Pos</i>), 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	<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>20</sup>-1)].                As an unsigned 32-bit value.</p>
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 271</a>.</li> </ul>
RTC4→RTC5	Essentially unchanged functionality (see <a href="#">list_call</a> ).
Version info	–
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 list memory area 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>20</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>20</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>20</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 <a href="#">list_call_repeat( Pos, 1 )</a>.</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>• With <a href="#">list_call_repeat</a>, for example, trajectories (see Glossary entry on <a href="#">page 25</a>) from <a href="#">micro_vector[*] commands</a> 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 <a href="#">micro_vector[*] command</a> 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 <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→RTC5	New command.				
Version info	Available as of DLL 540, OUT 540.				
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>				



Normal List Command	<b>list_continue</b>
Function	Inserts a null operation (no operation) into the list memory area.
Call	<code>list_continue()</code>
Comments	<ul style="list-style-type: none"> <li>Execution of <b>list_continue</b> (as does <b>list_nop</b>) requires 10 <math>\mu</math>s.</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 <math>\mu</math>s clock cycle (the short list command's "effective" execution time is then 10 <math>\mu</math>s). Unlike <b>list_nop</b>, however, <b>list_continue</b> neither modifies laser signals nor pauses for <b>Scanner Delays</b>. In contrast to <b>list_nop</b>, therefore, <b>list_continue</b> allows postponing short list commands to the next 10 <math>\mu</math>s clock cycle without interrupting a <b>Polyline</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 <b>list_continue</b>, for example, multiple directly consecutive <b>write_da_x_list</b> commands (short list commands) would occur in a single 10 <math>\mu</math>s clock cycle, whereby some values to the analog output port would never get outputted.</li> <li>The RTC5 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→RTC5	New command.
Version info	–
References	<b>list_nop</b>



<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 <a href="#">list_jump_pos( Pos )</a>, if the current <code>IOvalue</code> 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 <code>Mask1</code> are 1 and the bits specified in <code>Mask0</code> are 0). Otherwise, the directly following list command is immediately executed.</p>
Call	<code>list_jump_cond( Mask1, Mask0, Pos )</code>
Parameters	<p><code>Mask1</code> 16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</p> <p><code>Mask0</code> See <code>Mask1</code>.</p> <p><code>Pos</code> Absolute jump address <math>[0 \dots (2^{20}-1)]</math>. As an unsigned 32-bit value.</p>
Comments	<ul style="list-style-type: none"> <li>• <b>list_jump_cond</b> is synonymous with <a href="#">list_jump_pos_cond</a> (see comments there).</li> </ul>
RTC4→RTC5	<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 <b>Jump commands</b> are ignored during processing, see also <a href="#">list_jump_pos</a>).</li> </ul>
Version info	–
References	<a href="#">list_jump_pos</a> , <a href="#">list_jump_pos_cond</a>

<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	<code>Pos</code> Absolute jump address $[0\dots(2^{20}-1)]$ . 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 <b>Jump commands</b> are transmitted unaltered to the RTC5, but are ignored during processing. Instead, the next command is executed. Hence, the user program does probably no longer perform as expected. Therefore, <b>Jump commands</b> must be carefully programmed (see also <a href="#">Chapter 6.5.3 "Jumps", page 109</a>).</li> <li><b>Jump commands</b> initiating a jump to themselves (<code>Pos</code> = list position of the <b>Jump command</b>) are 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 <b>External Stop</b>).</li> <li>Decisive are the runtime conditions. When reconfiguring list memory or converting a subroutine, an originally legal jump address might become illegal 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 96</a>).</li> <li>The <b>BUSY list status</b> of the two lists is alternatingly set by <b>list_jump_pos</b>, the <b>USED list status</b> of the two lists remains unchanged (see <a href="#">read_status</a>).</li> <li>The <b>list_jump_pos</b> command is synonymous with <b>set_list_jump</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_list_jump</a> , <a href="#">list_jump_rel</a> , <a href="#">list_jump_pos_cond</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	<p>Mask1      16-bit mask. As an unsigned 32-bit value. Only the lower 16 bits are evaluated.</p> <p>Mask0      See <i>Mask1</i>.</p> <p>Pos      Absolute jump address [0...(<math>2^{20}-1</math>)]. As an unsigned 32-bit value.</p>
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 <a href="#">Jump command</a>), 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 271</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> <ul style="list-style-type: none"> <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="#">Chapter "Example Code (Pascal)", page 272</a>.</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
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	<code>list_jump_rel( Pos )</code>
Parameters	Pos      Jump distance $[(-2^{20}+1)...(2^{20}-1)]$ . As a signed 32-bit value.
Comments	<ul style="list-style-type: none"> <li>The <b>list_jump_rel</b> command 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 list input 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 area "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 <b>list_jump_rel</b> 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 <b>Jump commands</b> are transmitted unaltered to the RTC5, but are ignored during processing. Instead, the next command is executed. Hence, the user program does probably no longer perform as expected. Therefore, <b>Jump commands</b> must be carefully programmed (see also <a href="#">Chapter 6.5.3 "Jumps", page 109</a>).</li> <li><b>Jump commands</b> initiating a jump to themselves (<code>Pos = 0</code>) 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 <b>External Stop</b>).</li> <li>Decisive are the runtime conditions. When reconfiguring list memory or converting a subroutine, an originally legal jump address might become illegal due to new list boundaries or a relocated subroutine storage position.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">list_jump_pos</a> , <a href="#">list_jump_rel_cond</a>

<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	<p>Mask1      16-bit mask.                  As an unsigned 32-bit value.                  Only the lower 16 bits are evaluated.</p> <p>Mask0      See <i>Mask1</i>.</p> <p>Pos        Jump distance <math>[(-2^{20}+1) \dots (2^{20}-1)]</math>.                  As a signed 32-bit value.</p>
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_pos_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 271</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 272</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 <math>\mu</math>s clock cycle.</li> <li>• <b>list_next</b> is a proper place holder for another command in the list, because it prevents an extra 10 <math>\mu</math>s clock cycle, which is unavoidable with other place holders such as <b>list_nop</b> or <b>list_continue</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 541, OUT 541.
References	<b>list_nop</b> , <b>list_continue</b>

<b>Normal List Command</b>	<b>list_nop</b>
Function	Inserts a null operation (no operation) into the list memory area.
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 <b>Signals for "Laser Active" Operation</b> after a <b>LaserOff Delay</b> and waits for a possible scanner delay. Even if no delays need to be waited for, execution of <b>list_nop</b> requires 10 <math>\mu</math>s.</li> <li>• <b>list_nop</b> serves as a placeholder for rejected list commands (<b>get_last_error</b> return code <b>RTC5_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 <math>\mu</math>s clock cycle (the short list command's "effective" execution time is then 10 <math>\mu</math>s).</li> </ul>
RTC4→RTC5	The RTC4 command is a pure placeholder and is more like <b>list_continue</b> . The RTC5 command switches the <b>Signals for "Laser Active" Operation</b> off and waits for a scanner delay.
Version info	–
References	<b>long_delay</b> , <b>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 <code>list_repeat</code> and the next <code>list_until</code> call is possibly repeated multiple times.</li> <li>Each executed <code>list_repeat...list_until</code> loop requires a full 10 <math>\mu</math>s clock cycle. Accordingly, the first list command after a repetition is executed with a 10 <math>\mu</math>s delay.</li> <li>Nesting up to 8 <code>list_repeat...list_until</code> loops deep is allowed. Each further <code>list_repeat</code> call beyond that limit is ignored as long as no <code>list_until</code> call has terminated a loop.</li> <li>If a list terminates (by <code>set_end_of_list</code> or <code>stop_execution</code> or /STOP), then all not-yet-fully-processed <code>list_repeat...list_until</code> loops are deleted. However, this does not occur if an automatic list change (by <code>auto_change</code>, <code>auto_change_pos</code> or <code>start_loop</code>) is active.</li> <li>Complete <code>list_repeat...list_until</code> 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 <code>list_jump_pos</code>).</li> <li>List jumps into a <code>list_repeat...list_until</code> loop 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→RTC5	New command.
Version info	–
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 immediately following list command (possibly after a <b>list_nop</b> ) immediately and without delay.
Call	<code>list_return()</code>
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", <b>list_return</b> 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 <b>list_return</b>, a flush of the buffered list input is triggered, see <a href="#">Chapter 6.4.1 "Loading Lists", page 94</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→RTC5	No changes to the previous functionality (termination of a subroutine). New: impact during loading into the protected list memory area "List 3".
Version info	–
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>

<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	<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>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 has been 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→RTC5	New command.
Version info	–
References	<a href="#">list_repeat</a>



Ctrl Command	load_auto_laser_control																				
Function	Loads a table with data points from an ASCII text file. Determines – by linear interpolation – the nonlinearity curve (see <a href="#">Section "Loading and Determining the Nonlinearity Curve", page 192</a> ) for "Automatic Laser Control" (except Vector-Defined Laser Control), see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a> , page 187.																				
Call	NoOfDataPoints = load_auto_laser_control( Name, No )																				
Parameters	<table> <tr> <td>Name</td><td>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.</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 <a href="#">[AutoLaserCtrlTable&lt;No&gt;]</a> at the beginning of the desired table. As an unsigned 32-bit value.</td></tr> </table>	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.	No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of <a href="#">[AutoLaserCtrlTable&lt;No&gt;]</a> at the beginning of the desired table. As an unsigned 32-bit value.																
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.																				
No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of <a href="#">[AutoLaserCtrlTable&lt;No&gt;]</a> at the beginning of the desired table. As an unsigned 32-bit value.																				
Result	<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> <tr> <td>Value</td><td>Description</td></tr> <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 192</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><b>DSP</b> memory error.</td></tr> <tr> <td>5</td><td>BUSY error, board has been BUSY or <b>INTERNAL-BUSY</b> list execution status, no download (<a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a>).</td></tr> <tr> <td>8</td><td>Board is locked by another user program (<a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a>).</td></tr> <tr> <td>11</td><td>PCI error (<a href="#">get_last_error</a> return code <a href="#">RTC5_SEND_ERROR</a>), verify error (<a href="#">get_last_error</a> return code <a href="#">RTC5_VERIFY_ERROR</a>).</td></tr> <tr> <td>13</td><td>The specified table number could not been found in the file.</td></tr> </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 192</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	<b>DSP</b> memory error.	5	BUSY error, board has been BUSY or <b>INTERNAL-BUSY</b> list execution status, no download ( <a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a> ).	8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a> ).	11	PCI error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_SEND_ERROR</a> ), verify error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_VERIFY_ERROR</a> ).	13	The specified table number could 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 also <a href="#">Section "Loading and Determining the Nonlinearity Curve", page 192</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	<b>DSP</b> memory error.																				
5	BUSY error, board has been BUSY or <b>INTERNAL-BUSY</b> list execution status, no download ( <a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a> ).																				
8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a> ).																				
11	PCI error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_SEND_ERROR</a> ), verify error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_VERIFY_ERROR</a> ).																				
13	The specified table number could not been found in the file.																				
Comments	<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 192</a>. When loading the table, the RTC5 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 (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status 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 <b>set_wait</b> (<b>PAUSED</b> list execution status set)</li> </ul> </li> <li>• During the runtime of <b>load_auto_laser_control</b>, <b>External Starts</b> 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 <b>4</b> in case of error.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_auto_laser_control</b>

<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>	<p>Char      Index of the indexed character.                As an unsigned 32-bit value.                Allowed value range: [0...1023].</p>
<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} \times 256 + \text{ASCII number of the character}</math>                (character sets are numbered 0 to 3).                If <code>Char &gt; 1023</code> then <code>load_char</code> is ignored (<code>get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>).</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 <code>load_sub</code>) and text string definitions (see <code>load_text_table</code>).</li> <li>Indexed character definitions must be terminated with a <code>list_return</code> call. This is a prerequisite for actual storage of the commands, entry of the start address into the internal management table, and initiating a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 94</a>. Otherwise (the input pointer is altered without a preceding <code>list_return</code>), 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" has not been previously configured for a sufficient size beyond "List 1" and "List 2".</li> <li>If <code>list_return</code> is the next command after <code>load_char</code>, 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 107</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 correction file into RTC5 memory. Then calls automatically <b>select_cor_table</b> with the most recently used parameter values (or the default parameter values).																				
<b>Call</b>	<code>ErrorNo = load_correction_file( Name, No, Dim )</code>																				
<b>Parameters</b>	<p><b>Name</b> Name of the correction file. As a pointer to a \0-terminated ANSI string.</p> <p><b>No</b> Number of the correction table. Allowed values: [1...4]. As an unsigned 32-bit value. See also <b>number_of_correction_tables</b>.</p> <p><b>Dim</b> 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.</p> <ul style="list-style-type: none"> <li>= 2: 2D correction files and 3D correction files are stored as a 2D correction table ( downgrade, if necessary).</li> <li>= 3: If the <b>Option "3D"</b> is enabled, 2D correction files and 3D correction files are stored as a 3D correction table ( upgrade, if necessary). If the <b>Option "3D"</b> is not enabled, like <b>Dim</b> = 2.</li> </ul> <p>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> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Success.</td> </tr> <tr> <td>1</td> <td>File error (file corrupt or incomplete).</td> </tr> <tr> <td>2</td> <td>Memory error (RTC5 DLL-internal, Windows system memory).</td> </tr> <tr> <td>3</td> <td>File-open error (empty string submitted for <b>Name</b> parameter, file not found, etc.).</td> </tr> <tr> <td>4</td> <td>DSP memory error.</td> </tr> <tr> <td>5</td> <td>PCI download error (RTC5 board driver error).</td> </tr> <tr> <td>8</td> <td>RTC5 board driver not found (<b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>).</td> </tr> <tr> <td>10</td> <td>Parameter error (incorrect <b>No</b>).</td> </tr> <tr> <td>11</td> <td>Access error: board reserved for another user program (<b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>) or version check has detected an error (<b>RTC5OUT.out</b> or <b>RTC5RBF.rbf</b> version not compatible to the current RTC5 DLL version, <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>   <b>RTC5_VERSION_MISMATCH</b>).</td> </tr> </tbody> </table>	Value	Description	0	Success.	1	File error (file corrupt or incomplete).	2	Memory error (RTC5 DLL-internal, Windows system memory).	3	File-open error (empty string submitted for <b>Name</b> parameter, file not found, etc.).	4	DSP memory error.	5	PCI download error (RTC5 board driver error).	8	RTC5 board driver not found ( <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b> ).	10	Parameter error (incorrect <b>No</b> ).	11	Access error: board reserved for another user program ( <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b> ) or version check has detected an error ( <b>RTC5OUT.out</b> or <b>RTC5RBF.rbf</b> version not compatible to the current RTC5 DLL version, <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>   <b>RTC5_VERSION_MISMATCH</b> ).
Value	Description																				
0	Success.																				
1	File error (file corrupt or incomplete).																				
2	Memory error (RTC5 DLL-internal, Windows system memory).																				
3	File-open error (empty string submitted for <b>Name</b> parameter, file not found, etc.).																				
4	DSP memory error.																				
5	PCI download error (RTC5 board driver error).																				
8	RTC5 board driver not found ( <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b> ).																				
10	Parameter error (incorrect <b>No</b> ).																				
11	Access error: board reserved for another user program ( <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b> ) or version check has detected an error ( <b>RTC5OUT.out</b> or <b>RTC5RBF.rbf</b> version not compatible to the current RTC5 DLL version, <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>   <b>RTC5_VERSION_MISMATCH</b> ).																				

Ctrl Command	load_correction_file
	<p>12      Warning: 3D correction table or Dim = 3 selected, but the <b>Option "3D"</b> is not enabled. The system subsequently operates as an ordinary 2D system (this warning is only returned, if no other error has occurred).</p> <p>13      Busy error: no download, board is BUSY or <b>INTERNAL-BUSY</b> list execution status (<b>get_last_error</b> return code <b>RTC5_BUSY</b>).</p> <p>14      PCI upload error (RTC5 board driver error, only with download verification).</p> <p>15      Verify error (only with download verification).</p>
Comments	<p>Notes on loading correction tables:</p> <ul style="list-style-type: none"> <li>The RTC5 can store 4 (on request: 8) different correction tables at the same time, for example, for use in a multiple scan head configuration.</li> <li>Correction tables number 3 and 4 must always be loaded only after <b>load_program_file</b>. These tables occupy the upper half of the memory area reserved for data recording by <b>set_trigger</b> or <b>set_trigger4</b> (see comments for those commands). <b>load_program_file</b> automatically initializes this memory area and any already loaded correction tables number 3 and 4 are lost.</li> <li>Before storing a correction file, <b>load_correction_file</b> performs a <b>DSP</b> memory check. In case of an error, error code 4 is returned.</li> <li>If the parameter <b>No</b> is out of range, then no correction table is loaded (return value 10). Users can further restrict the permitted range by <b>number_of_correction_tables</b>.</li> <li>The name of the to-be-loaded correction file must be passed to <b>load_correction_file</b>. As a pointer to a \0-terminated ANSI string. If <b>Name</b> is passed as a <b>NULL</b> pointer, the corresponding table is replaced by a 1-to-1 table (for <b>Dim</b> = 3 and enabled <b>Option "3D"</b> with a 1-to-1 3D correction table, otherwise with a 1-to-1 2D table). An empty string ("") for <b>Name</b> result in an error return code of 3.</li> <li>If the <b>Option "3D"</b> is <i>not</i> enabled (default), then 2D correction files and 3D correction files are stored as 2D correction tables – regardless of the value specified for <b>Dim</b> (3D correction files also include 2D correction tables). The 3D data sections of 3D correction files are then ignored.</li> <li>If, on the other hand, the <b>Option "3D"</b> is <i>enabled</i>, then both 2D and 3D correction tables can be loaded. <ul style="list-style-type: none"> <li>For <b>Dim</b> = 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 <b>Option "3D"</b> has been enabled).</li> <li>For <b>Dim</b> = 3, if the <b>Option "3D"</b> is enabled then both 2D correction files 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 the <b>load_z_table</b> command (see <a href="#">page 159</a>).</li> <li>All other values for <b>Dim</b> do not change the type of the correction file.</li> </ul> </li> </ul>



Ctrl Command	load_correction_file
Comments (cont'd)	<p>Notes 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 RTC5 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 saving 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. See also <a href="#">Notes, page 165</a>.</li> </ul> <p>Other notes:</p> <ul style="list-style-type: none"> <li>RTC5 correction tables contain parameters that formerly (with an RTC4) had to be manually copied into a user program from a supplied <code>*_ReadMe.txt</code> file. To directly integrate these parameters into the user program, the RTC5 can load them by <code>get_table_para</code> from the currently loaded correction table or read them by <code>get_head_para</code> from the assigned correction table.</li> <li>During the runtime of <code>load_correction_file</code>: <ul style="list-style-type: none"> <li>No other control commands are executed</li> <li><code>External Starts</code> are suppressed</li> </ul> </li> <li><code>load_correction_file</code> is not executed (<code>get_last_error</code> return code <code>RTC5_BUSY</code>), if: <ul style="list-style-type: none"> <li>the <code>BUSY</code> list execution status is set</li> <li>the <code>INTERNAL-BUSY</code> list execution status 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> list execution status set)</li> </ul> </li> <li>If an <code>RTC5_VERSION_MISMATCH</code> error occurs (return value 11), a RTC5 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 RTC5 DLL version must be reloaded by the multi-board command <code>n_load_program_file</code> (after <code>RTC5_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 accessed by <code>acquire_RTC</code> or <code>select_RTC</code> and possibly specified as default board by <code>select_RTC</code>) <code>load_correction_file</code> can be normally executed again.</li> </ul>



Ctrl Command	load_correction_file
RTC4→RTC5	<p>Except for the basic function and the first two parameters (loading of correction files), the functionality has substantially changed:</p> <ul style="list-style-type: none"> <li>• <b>load_correction_file</b> 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, at least <b>two</b> 3D correction tables can be stored in RTC5 memory (with <b>Option "3D"</b>). However, two 3D correction tables can <i>not</i> be simultaneously assigned to the scan head connectors (see <b>select_cor_table</b>).</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 <b>set_scale</b>, <b>set_angle</b> and <b>set_offset</b> in addition to <b>set_matrix</b>.</li> <li>• <b>select_cor_table</b> is automatically called, see comments above and <b>Notes, page 165</b>.</li> <li>• <b>load_correction_file</b> does not return to the user program until the correction motion has been completed (see comments above).</li> </ul>
Version info	–
References	<b>number_of_correction_tables</b>



<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 = <b>load_disk( Name, Mode )</b>
<b>Parameters</b>	<p>Name                   File name or <b>NULL</b>.                             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> <p>= 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).</p> <p>&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).</p>
<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 <b>Name = NULL</b>, only the internal management tables are initialized as with <b>Mode = 0</b> (no loading occurs, no "empty" binary file must be provided). Otherwise, the <b>load_disk</b> command 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 pertaining list commands are thereby written to the protected list memory area "List 3" and the 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 <b>Mode &gt; 0</b> 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 the <b>save_disk</b> command, <b>load_disk</b> can be used, for example, to defragment the protected list memory area "List 3" and to subsequently protect subroutines (see <a href="#">Section "Subsequent Protection and Conversion of Non-Indexed Subroutines", page 105</a> and <a href="#">Section "Index Management and Defragmentation", page 104</a>). 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> command, be sure that the memory configuration provides a sufficiently large protected area 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 DLL, not from the board, see also <a href="#">get_config_list</a> and <a href="#">Chapter 6.7.1 "Board Acquisition by a User Program", page 117</a>).</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, if: <ul style="list-style-type: none"> <li>"List 3" has no memory area assigned, that is, <math>Mem1 + Mem2 = 2^{20}</math> (<b>get_last_error</b> return code <code>RTC5_PARAM_ERROR</code>)</li> <li>the <b>BUSY</b> list execution status is set (<b>get_last_error</b> return code <code>RTC5_BUSY</code>)</li> <li>the <b>INTERNAL-BUSY</b> list execution status is set (<b>get_last_error</b>-Returncode <code>RTC5_BUSY</code>)</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> list execution 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>: <ul style="list-style-type: none"> <li>No other commands can be executed</li> <li><b>External Starts</b> are suppressed</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
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 <b>Section "2D Encoder Compensation for xy Positioning Stages", page 234</b> .																						
<b>Call</b>	NoOfDataPoints = load_fly_2d_table( Name, No )																						
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>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.</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 <b>[Fly2DTable&lt;No&gt;]</b> at the beginning of the desired table. As an unsigned 32-bit value.</td> </tr> </table>	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.	No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of <b>[Fly2DTable&lt;No&gt;]</b> at the beginning of the desired table. As an unsigned 32-bit value.																		
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.																						
No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <No> of <b>[Fly2DTable&lt;No&gt;]</b> 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> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>&lt; 0</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="#">page 235</a>).</td> </tr> <tr> <td>0</td> <td>For Name = <b>NULL</b> (see comments).</td> </tr> <tr> <td>1</td> <td>No valid data points found (though Table No found).</td> </tr> <tr> <td>2</td> <td>Out of Memory (not enough Windows system memory).</td> </tr> <tr> <td>3</td> <td>File not found.</td> </tr> <tr> <td>4</td> <td><b>DSP</b> memory error.</td> </tr> <tr> <td>5</td> <td>BUSY error, board is BUSY or <b>INTERNAL-BUSY</b> list execution status, no download (<b>get_last_error</b> return code <b>RTC5_BUSY</b>).</td> </tr> <tr> <td>8</td> <td>Board is locked by another user program (<b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>).</td> </tr> <tr> <td>11</td> <td>PCI error (<b>get_last_error</b> return code <b>RTC5_SEND_ERROR</b>), verify error (<b>get_last_error</b> return code <b>RTC5_VERIFY_ERROR</b>).</td> </tr> <tr> <td>13</td> <td>The specified table number could not be found in the file.</td> </tr> </table>	Value	Description	< 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="#">page 235</a> ).	0	For Name = <b>NULL</b> (see comments).	1	No valid data points found (though Table No found).	2	Out of Memory (not enough Windows system memory).	3	File not found.	4	<b>DSP</b> memory error.	5	BUSY error, board is BUSY or <b>INTERNAL-BUSY</b> list execution status, no download ( <b>get_last_error</b> return code <b>RTC5_BUSY</b> ).	8	Board is locked by another user program ( <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b> ).	11	PCI error ( <b>get_last_error</b> return code <b>RTC5_SEND_ERROR</b> ), verify error ( <b>get_last_error</b> return code <b>RTC5_VERIFY_ERROR</b> ).	13	The specified table number could not be found in the file.
Value	Description																						
< 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="#">page 235</a> ).																						
0	For Name = <b>NULL</b> (see comments).																						
1	No valid data points found (though Table No found).																						
2	Out of Memory (not enough Windows system memory).																						
3	File not found.																						
4	<b>DSP</b> memory error.																						
5	BUSY error, board is BUSY or <b>INTERNAL-BUSY</b> list execution status, no download ( <b>get_last_error</b> return code <b>RTC5_BUSY</b> ).																						
8	Board is locked by another user program ( <b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b> ).																						
11	PCI error ( <b>get_last_error</b> return code <b>RTC5_SEND_ERROR</b> ), verify error ( <b>get_last_error</b> return code <b>RTC5_VERIFY_ERROR</b> ).																						
13	The specified table number could not be found in the file.																						
<b>Comments</b>	<ul style="list-style-type: none"> <li>The text file's data format requirements for reference points of the 2D encoder compensation table are described in <b>Section "For the 2D compensation tables, the following rules apply:", page 235</b>. 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 any 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.</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 (<a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status 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 <a href="#">set_wait</a> (<b>PAUSED</b> list execution status set)</li> </ul> </li> <li>• During the runtime of <b>load_fly_2d_table</b>, <b>External Starts</b> 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→RTC5	New command.
Version info	–
Version info	Available as of DLL 536, OUT 536.
References	<a href="#">set_fly_2d</a> , <a href="#">init_fly_2d</a> , <a href="#">get_fly_2d_offset</a>

<b>Ctrl Command</b>	<b>load_jump_table</b>												
Function	Loads a value table with <b>Jump Delay</b> 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 <b>Jump Delay</b> table for 2D jumps executable in <b>Jump Mode</b> .												
Call	NoOfDataPoints = <code>load_jump_table( Name, No, PosAck, MinDelay, MaxDelay, ListPos )</code>												
Parameters	<table> <tr> <td>Name</td> <td>See <a href="#">load_jump_table_offset</a>.</td> </tr> <tr> <td>No</td> <td>See <a href="#">load_jump_table_offset</a>.</td> </tr> <tr> <td>PosAck</td> <td>See <a href="#">load_jump_table_offset</a>.</td> </tr> <tr> <td>MinDelay</td> <td>See <a href="#">load_jump_table_offset</a>.</td> </tr> <tr> <td>MaxDelay</td> <td>See <a href="#">load_jump_table_offset</a>.</td> </tr> <tr> <td>ListPos</td> <td>See <a href="#">load_jump_table_offset</a>.</td> </tr> </table>	Name	See <a href="#">load_jump_table_offset</a> .	No	See <a href="#">load_jump_table_offset</a> .	PosAck	See <a href="#">load_jump_table_offset</a> .	MinDelay	See <a href="#">load_jump_table_offset</a> .	MaxDelay	See <a href="#">load_jump_table_offset</a> .	ListPos	See <a href="#">load_jump_table_offset</a> .
Name	See <a href="#">load_jump_table_offset</a> .												
No	See <a href="#">load_jump_table_offset</a> .												
PosAck	See <a href="#">load_jump_table_offset</a> .												
MinDelay	See <a href="#">load_jump_table_offset</a> .												
MaxDelay	See <a href="#">load_jump_table_offset</a> .												
ListPos	See <a href="#">load_jump_table_offset</a> .												
Result	See <a href="#">load_jump_table_offset</a> .												
Notes	<ul style="list-style-type: none"> <li>• <b>load_jump_table</b> is identical to <b>load_jump_table_offset</b> with <b>Offset</b> = 0.</li> </ul>												
Version info	–												
RTC4→RTC5	New command.												
Reference	<a href="#">load_jump_table_offset</a>												



<b>Ctrl Command</b>	<b>load_jump_table_offset</b>
<b>Function</b>	Loads a value table with <b>Jump Delay</b> 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 <b>Jump Delay</b> table for 2D jumps executable in <b>Jump Mode</b> .
<b>Call</b>	NoOfDataPoints = load_jump_table_offset( <b>Name</b> , <b>No</b> , <b>PosAck</b> , <b>Offset</b> , <b>MinDelay</b> , <b>MaxDelay</b> , <b>ListPos</b> )
<b>Parameters</b>	<p><b>Name</b>      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><b>No</b>      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><b>PosAck</b>      Position tolerance value in [bits] (only relevant for automatic determination). As an unsigned 32-bit value.</p> <p><b>Offset</b>      Offset in [10 µs] which is added to all automatically determined delay values (only relevant for automatic determination). As a signed 32-bit value.</p> <p><b>MinDelay</b>      Minimum <b>Jump Delay</b> in [10 µs] (only relevant for automatic determination). As an unsigned 32-bit value.</p> <p><b>MaxDelay</b>      Maximum <b>Jump Delay</b> in [10 µs] (only relevant for automatic determination). As an unsigned 32-bit value.</p> <p><b>ListPos</b>      List position (in the area of list 1 or 2) for six list commands that can be overwritten (only relevant for automatic determination).</p>
<b>Result</b>	<p>Error code.</p> <p>As an unsigned 32-bit value.</p> <p>-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="#">page 205</a>).</p> <p>-1024      Success for <b>Name</b> = <b>NULL</b>: Table has been automatically determined.</p> <p>1      No valid data points found (but Table <b>No</b> found).</p> <p>3      File not found.</p> <p>4      Verify error: <b>DSP</b> memory error.</p> <p>5      Busy error, board is <b>BUSY</b> or <b>INTERNAL-BUSY</b> list execution status, no download (<a href="#">get_last_error</a> return code <b>RTC5_BUSY</b>).</p> <p>8      Access error: board reserved for another user program (<a href="#">get_last_error</a> return code <b>RTC5_ACCESS_DENIED</b>).</p>

<b>Ctrl Command</b>	<b>load_jump_table_offset</b>
Result (cont'd)	10        Only if <code>Name</code> = <b>NULL</b> : Param error: HeadNo or ListPos invalid. 11        PCI error during download ( <code>get_last_error</code> return code <b>RTC5_SEND_ERROR</b> ). 13        The specified table number could not be found in the file.
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>See also <a href="#">Chapter 8.1.5 "Jump Mode", page 202</a>.</li> <li>Format requirements for placing the table with <b>Jump Delay</b> data points into the text file are described in <a href="#">Section "Notes on Loading Determined Jump Delay Values", page 205</a>. When loading the table, the RTC5 uses linear interpolation to establish appropriate values for the complete jump length range.</li> <li><b>load_jump_table_offset</b> overwrites any previously loaded <b>Jump Delay</b> tables for <b>Jump Mode</b>.</li> <li>If <code>Name</code> = filename, then the table number <code>No</code> is loaded. The other parameters are not used.</li> <li>If <code>Name</code> = <b>NULL</b>, then the <b>Jump Delay</b> table for head <code>No</code> is automatically determined and loaded (if <b>Jump Mode</b> has been previously enabled and activated successfully by <code>set_jump_mode</code> (Flag = 1)). 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 206</a>).</li> <li><b>Jump Mode</b> takes effect upon the next 2D jump only if it has been activated and enabled by <code>set_jump_mode</code> or activated by <code>set_jump_mode_list</code>.</li> <li><b>load_jump_table_offset</b> is not executed (<code>get_last_error</code> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY</b> list execution status is set</li> <li>the <b>INTERNAL-BUSY</b> list execution status is set</li> </ul> </li> <li><b>load_jump_table_offset</b> is even executed, if: <ul style="list-style-type: none"> <li>a list has been paused by <code>set_wait</code> (<b>PAUSED</b> list execution status set)</li> </ul> </li> <li>During the runtime of <b>load_jump_table_offset</b>, <b>External Starts</b> are suppressed.</li> <li>Before loading a table, <b>load_jump_table_offset</b> performs a <b>DSP</b> memory check that produces error code 4 in case of error.</li> </ul>
RTC4→RTC5	New command. There is no <b>RTC4 Compatibility Mode</b> for this command.
Version info	–
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>

<b>Ctrl Command</b>	<b>load_list</b>
<b>Function</b>	Opens the list memory area for writing with list commands and sets the input pointer to the specified <i>relative</i> position within the desired list ("List 1" or "List 2"), but only if the list is not currently being processed ( <b>BUSY list status</b> is not set) or has already been processed ( <b>USED list status</b> is set).
<b>Call</b>	NoOfOpenedList = load_list( ListNo, Pos )
<b>Parameters</b>	<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> <p>= 0:      The list is opened for which the <b>BUSY list status</b> is not currently set.                     "List 1" is opened, if the <b>BUSY list status</b> is not set for either list.</p> <p>= 1:      "List 1" is opened, if the <b>BUSY list status</b> for it is not set                     (<b>BUSY1</b> not set).</p> <p>= 2:      "List 2" is opened if the <b>BUSY list status</b> for it is not set                     (<b>BUSY2</b> not set).</p> <p>= 3:      The list is opened for which the <b>BUSY list status</b> is not currently set but the <b>USED list status</b>.                     If for both lists the <b>BUSY list status</b> is not set but the <b>USED list status</b>: that list is opened, which would be executed by a following automatic list change.</p> <p>For <b>Mem2</b> = 0 (see <b>config_list</b>) "List 1" is opened, if:</p> <ul style="list-style-type: none"> <li>the <b>BUSY list status</b> for it is not set (ListNo = 0...2)</li> <li>the <b>BUSY list status</b> for it is not set but the <b>USED list status</b> (ListNo = 3)</li> </ul> <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>20</sup>-1)].</p>
<b>Result</b>	Number of the opened list [1 or 2] if successful, otherwise 0. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>If <b>load_list</b> has been successful, the next list command is stored at the specified address and all subsequent list commands at the following addresses in the selected list.</li> <li>If <b>load_list</b> has not been 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 the responsibility of the user to react to a return code of 0. <b>load_list</b> produces <i>no</i> 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>• <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 100</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 <a href="#">set_start_list_pos</a>.</li> <li>• After a successful check of the list number (<b>BUSY list status</b> not set and, if applicable, <b>USED list status</b> set), <b>load_list</b> behaves like <a href="#">set_start_list_pos</a> (see comments there).</li> <li>• If ListNo = 0...2 when using <b>load_list</b>, then note that the <b>BUSY list status</b> of a list can change between opening of the list with <b>load_list</b> and the actual loading of list commands, for example, if in the meantime an automatic list change has occurred that has been previously defined by <a href="#">auto_change</a> or <a href="#">start_loop</a>. 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 ListNo = 3 <b>load_list</b> ensures that a list is actually opened for loading only directly after processing (and after any successful automatic list changes). Particularly, if for no list the <b>BUSY list status</b> is set but for both the <b>USED list status</b> (for example, after initialization, after <a href="#">stop_execution</a> or after an <a href="#">External Stop</a>), the opened list number is synchronized with the following automatic list change.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 189</a> ).																				
<b>Call</b>	NoOfDataPoints = load_position_control( Name, No )																				
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>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.</td> </tr> <tr> <td>No</td> <td>Determines which table in the text file is to be loaded. The parameter corresponds to the extension <b>&lt;No&gt;</b> of <a href="#">[PositionCtrlTable&lt;No&gt;]</a> at the beginning of the desired table. As an unsigned 32-bit value.</td> </tr> </table>	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.	No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <b>&lt;No&gt;</b> of <a href="#">[PositionCtrlTable&lt;No&gt;]</a> at the beginning of the desired table. As an unsigned 32-bit value.																
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.																				
No	Determines which table in the text file is to be loaded. The parameter corresponds to the extension <b>&lt;No&gt;</b> of <a href="#">[PositionCtrlTable&lt;No&gt;]</a> 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> <tr> <td>Value</td> <td>Description</td> </tr> <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 189</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>BUSY error, board is <b>BUSY</b> list execution status or <b>INTERNAL-BUSY</b> list execution status, no download (<a href="#">get_last_error</a> return code <b>RTC5_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>RTC5_ACCESS_DENIED</b>).</td> </tr> <tr> <td>11</td> <td>PCI error (<a href="#">get_last_error</a> return code <b>RTC5_SEND_ERROR</b>), verify error (<a href="#">get_last_error</a> return code <b>RTC5_VERIFY_ERROR</b>).</td> </tr> <tr> <td>13</td> <td>The specified table number could not been found in the file.</td> </tr> </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 189</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	BUSY error, board is <b>BUSY</b> list execution status or <b>INTERNAL-BUSY</b> list execution status, no download ( <a href="#">get_last_error</a> return code <b>RTC5_BUSY</b> ).	8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <b>RTC5_ACCESS_DENIED</b> ).	11	PCI error ( <a href="#">get_last_error</a> return code <b>RTC5_SEND_ERROR</b> ), verify error ( <a href="#">get_last_error</a> return code <b>RTC5_VERIFY_ERROR</b> ).	13	The specified table number could 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 also <a href="#">Section "Notes on Loading a Scaling Function", page 189</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	BUSY error, board is <b>BUSY</b> list execution status or <b>INTERNAL-BUSY</b> list execution status, no download ( <a href="#">get_last_error</a> return code <b>RTC5_BUSY</b> ).																				
8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <b>RTC5_ACCESS_DENIED</b> ).																				
11	PCI error ( <a href="#">get_last_error</a> return code <b>RTC5_SEND_ERROR</b> ), verify error ( <a href="#">get_last_error</a> return code <b>RTC5_VERIFY_ERROR</b> ).																				
13	The specified table number could not been 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 189</a>. When loading the table, the RTC5 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 <a href="#">load_program_file</a>), 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 mark commands or <a href="#">Arc Commands</a> if it has been initialized by <a href="#">set_auto_laser_control</a>. Position-dependent laser control is deactivated by <a href="#">set_auto_laser_control</a> (Ctrl = 0) or by loading <math>Scale(Position)=1.0</math>. See also <a href="#">Section "Position-Dependent Laser Control", page 189</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 (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status 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 <b>set_wait</b> (<b>PAUSED</b> list execution status set)</li> </ul> </li> <li>• During the runtime of <b>load_position_control</b>, <b>External Starts</b> are suppressed.</li> <li>• Before loading a table, <b>load_position_control</b> performs a <b>DSP</b> memory check. In case of an error, error code 4 is returned.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_auto_laser_control</b>

Ctrl Command	load_program_file																																				
Function	<ul style="list-style-type: none"> <li>• Resets the RTC5 (<a href="#">Software reset</a>)</li> <li>• Initializes the list memory</li> <li>• Performs a <b>DSP</b> memory check</li> <li>• Loads <a href="#">RTC5RBF.rbf</a></li> <li>• Loads <a href="#">RTC5OUT.out</a></li> <li>• Loads <a href="#">RTC5DAT.dat</a></li> <li>• Starts the <b>DSP</b></li> </ul>																																				
Call	ErrorNo = load_program_file( Path )																																				
Parameters	Path      Path name of the folder, where <a href="#">RTC5OUT.out</a> , <a href="#">RTC5RBF.rbf</a> and <a href="#">RTC5DAT.dat</a> are. As a pointer to a \0-terminated string.																																				
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>Success.</td> </tr> <tr> <td>1</td> <td>Reset error: the board could not be reset.</td> </tr> <tr> <td>2</td> <td>Unreset error: the board does not restart.</td> </tr> <tr> <td>3</td> <td>File error: file not found or cannot be opened.</td> </tr> <tr> <td>4</td> <td>Format error: <a href="#">RTC5OUT.out</a> has incorrect format.</td> </tr> <tr> <td>5</td> <td>System error: not enough Windows memory.</td> </tr> <tr> <td>6</td> <td>Another user program has already acquired the board.</td> </tr> <tr> <td>7</td> <td>Version error: <b>RTC5 DLL</b> version, <a href="#">RTC5RBF.rbf</a> version and <a href="#">RTC5OUT.out</a> version are not compatible with each other.</td> </tr> <tr> <td>8</td> <td>RTC5 board driver not found (<a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a>).</td> </tr> <tr> <td>9</td> <td>DriverCall error: loading of <a href="#">RTC5OUT.out</a> failed.</td> </tr> <tr> <td>10</td> <td>Configuration error: <b>DSP</b> register initialization failed.</td> </tr> <tr> <td>11</td> <td><b>FPGA</b> firmware error: loading of <a href="#">RTC5RBF.rbf</a> failed.</td> </tr> <tr> <td>12</td> <td>PCI download error: loading of <a href="#">RTC5DAT.dat</a> failed.</td> </tr> <tr> <td>13</td> <td>Busy error: Download locked, board is BUSY or <b>INTERNAL-BUSY</b> list execution status (<a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a>).</td> </tr> <tr> <td>14</td> <td><b>DSP</b> memory error.</td> </tr> <tr> <td>15</td> <td>Verify error (only applicable for download verification).</td> </tr> <tr> <td>16</td> <td>PCI error.</td> </tr> </tbody> </table>	Value	Description	0	Success.	1	Reset error: the board could not be reset.	2	Unreset error: the board does not restart.	3	File error: file not found or cannot be opened.	4	Format error: <a href="#">RTC5OUT.out</a> has incorrect format.	5	System error: not enough Windows memory.	6	Another user program has already acquired the board.	7	Version error: <b>RTC5 DLL</b> version, <a href="#">RTC5RBF.rbf</a> version and <a href="#">RTC5OUT.out</a> version are not compatible with each other.	8	RTC5 board driver not found ( <a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a> ).	9	DriverCall error: loading of <a href="#">RTC5OUT.out</a> failed.	10	Configuration error: <b>DSP</b> register initialization failed.	11	<b>FPGA</b> firmware error: loading of <a href="#">RTC5RBF.rbf</a> failed.	12	PCI download error: loading of <a href="#">RTC5DAT.dat</a> failed.	13	Busy error: Download locked, board is BUSY or <b>INTERNAL-BUSY</b> list execution status ( <a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a> ).	14	<b>DSP</b> memory error.	15	Verify error (only applicable for download verification).	16	PCI error.
Value	Description																																				
0	Success.																																				
1	Reset error: the board could not be reset.																																				
2	Unreset error: the board does not restart.																																				
3	File error: file not found or cannot be opened.																																				
4	Format error: <a href="#">RTC5OUT.out</a> has incorrect format.																																				
5	System error: not enough Windows memory.																																				
6	Another user program has already acquired the board.																																				
7	Version error: <b>RTC5 DLL</b> version, <a href="#">RTC5RBF.rbf</a> version and <a href="#">RTC5OUT.out</a> version are not compatible with each other.																																				
8	RTC5 board driver not found ( <a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a> ).																																				
9	DriverCall error: loading of <a href="#">RTC5OUT.out</a> failed.																																				
10	Configuration error: <b>DSP</b> register initialization failed.																																				
11	<b>FPGA</b> firmware error: loading of <a href="#">RTC5RBF.rbf</a> failed.																																				
12	PCI download error: loading of <a href="#">RTC5DAT.dat</a> failed.																																				
13	Busy error: Download locked, board is BUSY or <b>INTERNAL-BUSY</b> list execution status ( <a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a> ).																																				
14	<b>DSP</b> memory error.																																				
15	Verify error (only applicable for download verification).																																				
16	PCI error.																																				

Ctrl Command	load_program_file
Comments	<ul style="list-style-type: none"> <li>• If <code>Path</code> = <code>NULL</code>, then the path of the user program's current working directory is used. Caution: It is not always the folder from which the user program has been launched. The current working directory can change, for example, when a file from another folder is selected by the Windows Explorer (unless the "NoChangeDir" flag has been set when incorporating the Explorer window into the user program).</li> <li>• After each <b>Hardware reset</b>, the first user program must begin by issuing a <code>load_program_file</code> command during initialization of the RTC5 Board, see <a href="#">Section "Initializing the Board", page 87</a>. <code>load_program_file</code> should also be executed (for example, if another user program acquires the board, see <code>acquire_rtc</code>) when the board needs to be returned to the default state.</li> <li>• If multiple RTC5 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 113</a>.</li> <li>• After execution of <code>load_program_file</code>: <ul style="list-style-type: none"> <li>– The laser focus is in the center of the <code>Image</code> field (0 0)</li> <li>– The laser control is <i>deactivated</i></li> <li>– On the state of the various output ports, see <a href="#">Chapter 7.4.1 "Enabling, Activating and Switching Laser Control Signals", page 173</a> (LASERON, LASER1, LASER2), <a href="#">Chapter 9.1.1 "16-Bit Digital Output Port", page 258</a> (EXTENSION 1 socket connector), <a href="#">Chapter 9.1.2 "8-Bit Digital Output Port", page 259</a> (EXTENSION 2 socket connector), <a href="#">Chapter 9.1.3 "2 Bit Digital Output Port", page 259</a> (LASER connector), <a href="#">Chapter 9.1.4 "12-Bit Analog Output Port 1 and 2", page 259</a> (LASER connector).</li> </ul> </li> <li>• <i>Caution! In general, sporadic <code>load_program_file</code> calls in your user program:</i> <ul style="list-style-type: none"> <li>– <i>Contradict the safe switch-on sequence prescribed in <a href="#">Chapter 5.4 "Installing the RTC5 Software", page 80</a></i></li> <li>– <i>Pose the risk of personal injury and/or property damage (cases have been reported where lasers with poor electrics have emitted) If you absolutely cannot refrain from sporadic <code>load_program_file</code> calls in your user program, you must implement appropriate messages that warn users accordingly and prompt them to take actions that prevent these hazards.</i></li> </ul> </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 <code>load_correction_file</code>. Already-loaded correction tables remain loaded after <code>load_program_file</code> (#1, #2 only, not #3, #4).</li> <li>• During a RTC5 reset, list memory contents are erased and all parameters (for example, memory configuration, internal variables, matrices, offsets and table assignments) previously set with <code>config_list</code> or <code>select_cor_table</code> are deleted or reset to their default values. During a reset, a correction table is assigned as by <code>select_cor_table(1, 0)</code> but no scanner motion to the corrected output position is executed.</li> <li>• <code>load_program_file</code> only returns to the calling user program, when <b>DSP</b> initialization has been completed.</li> </ul>

Ctrl Command	load_program_file
Comments (cont'd)	<ul style="list-style-type: none"> <li>• <b>load_program_file</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if <ul style="list-style-type: none"> <li>– the <b>BUSY list execution status</b> is currently set</li> <li>– the <b>INTERNAL-BUSY list execution status</b> is currently set</li> </ul> </li> <li>• <b>load_program_file</b> is even executed, if: <ul style="list-style-type: none"> <li>– a list has been previously terminated in an orderly manner</li> <li>– a list has been paused by <b>set_wait</b> (<b>PAUSED list execution status</b> set)</li> <li>– a list has been paused by <b>stop_execution</b></li> <li>– a list has been paused by an <b>External Stop</b>.</li> </ul> </li> <li>• The files <b>RTC5OUT.out</b>, <b>RTC5RBF.rbf</b> and <b>RTC5DAT.dat</b> are included in the RTC5 software package. For easy identifying and archiving of different software versions, the files are also delivered zipped (the <b>zip</b> file names <b>RTC5&lt;...&gt;_&lt;Version&gt;.zip</b> include the version numbers). Copy or unzip the three files (of desired version) to the hard drive of your PC.</li> <li>• Assorted versions of the <b>RTC5 DLL</b> and the files <b>RTC5OUT.out</b>, <b>RTC5RBF.rbf</b> and <b>RTC5DAT.dat</b> cannot be arbitrarily combined with another (each <b>zip</b> file in the RTC5 software includes a text file with corresponding version information).</li> </ul> <p><b>load_program_file</b> performs a version check. If there is a version error, then the loaded programs remain in RTC5 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>RTC5_ACCESS_DENIED</b>   <b>RTC5_VERSION_MISMATCH</b>). To then load a correct program version, only the multi-board command <b>n_load_program_file</b> can be called. Hereby, temporary access rights are requested and released after the download (if the board has not been acquired by another user program; <b>n_load_program_file</b> does not perform an <b>acquire_RTC</b> command). In this case, the single-board command <b>load_program_file</b> does not get access rights for the board. Alternatively, the board can be made currentless to unload the program software (afterward also <b>load_program_file</b> can be used again).</p>
RTC4→RTC5	<ul style="list-style-type: none"> <li>• <b>Path</b> specifies a folder name with RTC5 (in contrast a file name with the RTC4).</li> <li>• <b>load_program_file</b> loads three files, with fixed formats and names (<b>RTC5OUT.out</b>, <b>RTC5RBF.rbf</b>, <b>RTC5DAT.dat</b>) (see above).</li> <li>• After execution of <b>load_program_file</b> the laser control is deactivated.</li> </ul>
Version info	–
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 )																						
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>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.</td> </tr> <tr> <td>No</td> <td> <p>As a signed 32-bit value.</p> <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 <b>[StretchTable&lt;No&gt;]</b> at the beginning of the desired table.</li> <li><math>No &lt; 0</math>: Reserved.</li> </ul> </td> </tr> </table>	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.	No	<p>As a signed 32-bit value.</p> <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 <b>[StretchTable&lt;No&gt;]</b> at the beginning of the desired table.</li> <li><math>No &lt; 0</math>: Reserved.</li> </ul>																		
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.																						
No	<p>As a signed 32-bit value.</p> <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 <b>[StretchTable&lt;No&gt;]</b> at the beginning of the desired table.</li> <li><math>No &lt; 0</math>: Reserved.</li> </ul>																						
<b>Result</b>	<p>A positive error code in case of an error. The negative number of found data pairs in case of success. As a signed 32-bit value.</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <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><b>DSP</b> memory error.</td> </tr> <tr> <td>5</td> <td>BUSY error, board is BUSY or <b>INTERNAL-BUSY</b> list execution status, no download (<b>get_last_error</b> return code <b>RTC5_BUSY</b>).</td> </tr> <tr> <td>6</td> <td>Data error: data pairs missing.</td> </tr> <tr> <td>11</td> <td>PCI download error (<b>get_last_error</b> return code <b>RTC5_SEND_ERROR</b>).</td> </tr> <tr> <td>13</td> <td>The specified table number could not be found in the file.</td> </tr> <tr> <td>15</td> <td>Verify error (<b>get_last_error</b> return code <b>RTC5_VERIFY_ERROR</b>, only possible with active download verification, see <b>set_verify</b>).</td> </tr> </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	<b>DSP</b> memory error.	5	BUSY error, board is BUSY or <b>INTERNAL-BUSY</b> list execution status, no download ( <b>get_last_error</b> return code <b>RTC5_BUSY</b> ).	6	Data error: data pairs missing.	11	PCI download error ( <b>get_last_error</b> return code <b>RTC5_SEND_ERROR</b> ).	13	The specified table number could not be found in the file.	15	Verify error ( <b>get_last_error</b> return code <b>RTC5_VERIFY_ERROR</b> , only possible with active download verification, see <b>set_verify</b> ).
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	<b>DSP</b> memory error.																						
5	BUSY error, board is BUSY or <b>INTERNAL-BUSY</b> list execution status, no download ( <b>get_last_error</b> return code <b>RTC5_BUSY</b> ).																						
6	Data error: data pairs missing.																						
11	PCI download error ( <b>get_last_error</b> return code <b>RTC5_SEND_ERROR</b> ).																						
13	The specified table number could not be found in the file.																						
15	Verify error ( <b>get_last_error</b> return code <b>RTC5_VERIFY_ERROR</b> , only possible with active download verification, see <b>set_verify</b> ).																						
<b>Comments</b>	<ul style="list-style-type: none"> <li>Details about enhanced 3D correction (for example, format requirements for the text file's table entries) are described on <a href="#">Section "Enhanced 3D Correction", page 225</a>.</li> <li>A successfully loaded table activates the new enhanced 3D correction. Here, <b>load_stretch_table</b> overwrites any previously loaded table.</li> <li>If <b>Name</b> is not <b>NULL</b>, but no table has been successfully read, then <b>load_stretch_table</b> returns an error code (for example, code 13 if a <b>No</b> value is specified for which no <b>[StretchTable&lt;No&gt;]</b> entry is included in the text file), but otherwise has no effect (that is, any previous successfully downloaded table stays enabled).</li> </ul>																						



Ctrl Command	load_stretch_table
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>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution 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> list execution status set)</li> </ul> </li> <li>• During execution of <code>load_stretch_table</code>, <b>External Starts</b> are suppressed.</li> <li>• Before loading a table, <code>load_stretch_table</code> performs a <b>DSP</b> memory check. In case of an error, error code 4 is returned.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	–

<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 value 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 <code>load_sub</code> is ignored (<code>get_last_error</code> return code <code>RTC5_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 by a <code>list_return</code> call. This is a prerequisite for actual storage of the commands, entry of the start address into the internal management table, and initiating a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 94</a>. Otherwise (the input pointer is altered without a preceding <code>list_return</code>), the subroutine with this index is not available.</li> <li>An indexed subroutine is not stored if the protected list memory area "List 3" has not been previously configured for a sufficient size beyond "List 1" and "List 2".</li> <li>If <code>list_return</code> is the next command after <code>load_sub</code>, then the corresponding subroutine is deleted from the internal management table.</li> <li>Indexed subroutines can be called by the <code>sub_call</code> command along with the corresponding index (see <a href="#">Section "General Information on Calling Subroutines", page 103</a>).</li> <li>Observe all notes in <a href="#">Section "Indexed Subroutines", page 102</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>	Index      Index of the indexed text string. As an unsigned 32-bit value. Allowed value range: [0...41].
<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>RTC5_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> call. This is a prerequisite for actual storage of the commands, entry of the start address into the internal management table, and initiating a flush of the buffered list input, see <b>Chapter 6.4.1 "Loading Lists", page 94</b>. Otherwise (the input pointer is altered without a preceding <b>list_return</b>), 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" has not been 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 107</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 "User-defined "Variable Polygon Delays""</a> , <a href="#">page 141</a> .																				
<b>Call</b>	<code>NoOfDataPoints = load_varpolydelay( Name, No )</code>																				
<b>Parameters</b>	<table> <tr> <td>Name</td> <td>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.</td> </tr> <tr> <td>No</td> <td>Table in the text file which is to be loaded. The parameter corresponds to the extension <code>&lt;No&gt;</code> of <a href="#">[VarPolyTable&lt;No&gt;]</a> at the beginning of the desired table. As an unsigned 32-bit value.</td> </tr> </table>	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.	No	Table in the text file which is to be loaded. The parameter corresponds to the extension <code>&lt;No&gt;</code> of <a href="#">[VarPolyTable&lt;No&gt;]</a> at the beginning of the desired table. As an unsigned 32-bit value.																
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.																				
No	Table in the text file which is to be loaded. The parameter corresponds to the extension <code>&lt;No&gt;</code> of <a href="#">[VarPolyTable&lt;No&gt;]</a> 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> <tr> <td>Value</td> <td>Description</td> </tr> <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="#">page 141</a>).</td> </tr> <tr> <td>-1024</td> <td>For <code>Name = NULL</code>: the table initialized according to <math>1-\cos(\phi)</math> has been internally loaded (as with program start; see <a href="#">Figure 44</a>).</td> </tr> <tr> <td>1</td> <td>No valid data points found (though Table <code>No</code> 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>BUSY error, board is BUSY or <a href="#">INTERNAL-BUSY list execution status</a>, no download (<a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a>).</td> </tr> <tr> <td>8</td> <td>Board is locked by another user program (<a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a>).</td> </tr> <tr> <td>11</td> <td>PCI error (<a href="#">get_last_error</a> return code <a href="#">RTC5_SEND_ERROR</a>), verify error (<a href="#">get_last_error</a> return code <a href="#">RTC5_VERIFY_ERROR</a>).</td> </tr> <tr> <td>13</td> <td>The specified table number could not been found in the file.</td> </tr> </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="#">page 141</a> ).	-1024	For <code>Name = NULL</code> : the table initialized according to $1-\cos(\phi)$ has been internally loaded (as with program start; see <a href="#">Figure 44</a> ).	1	No valid data points found (though Table <code>No</code> found).	3	File not found.	4	DSP memory error.	5	BUSY error, board is BUSY or <a href="#">INTERNAL-BUSY list execution status</a> , no download ( <a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a> ).	8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a> ).	11	PCI error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_SEND_ERROR</a> ), verify error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_VERIFY_ERROR</a> ).	13	The specified table number could 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="#">page 141</a> ).																				
-1024	For <code>Name = NULL</code> : the table initialized according to $1-\cos(\phi)$ has been internally loaded (as with program start; see <a href="#">Figure 44</a> ).																				
1	No valid data points found (though Table <code>No</code> found).																				
3	File not found.																				
4	DSP memory error.																				
5	BUSY error, board is BUSY or <a href="#">INTERNAL-BUSY list execution status</a> , no download ( <a href="#">get_last_error</a> return code <a href="#">RTC5_BUSY</a> ).																				
8	Board is locked by another user program ( <a href="#">get_last_error</a> return code <a href="#">RTC5_ACCESS_DENIED</a> ).																				
11	PCI error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_SEND_ERROR</a> ), verify error ( <a href="#">get_last_error</a> return code <a href="#">RTC5_VERIFY_ERROR</a> ).																				
13	The specified table number could 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 "User-defined "Variable Polygon Delays""</a>, <a href="#">page 141</a>. When loading the table, the RTC5 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 <code>Name = NULL</code> (as during initialization by <a href="#">load_program_file</a>), the internal (default) table for the "Variable Polygon Delay" (<math>1-\cos(\phi)</math>, see <a href="#">Figure 44</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>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY list execution status</b> is set (a list is being processed or has been halted by <b>pause_list</b>)</li> <li>– the <b>INTERNAL-BUSY list execution status</b> 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 list execution status</b> set)</li> </ul> </li> <li>• During execution of <b>load_varpolydelay</b>, <b>External Starts</b> are suppressed.</li> <li>• Before loading a table, <b>load_varpolydelay</b> performs a <b>DSP</b> memory check. In case of an error, error code 4 is returned.</li> </ul>
RTC4→RTC5	<p>Basically unchanged functionality. However:</p> <ul style="list-style-type: none"> <li>• To return to the internal standard <b>Polygon Delay</b> 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>
Version info	–
References	<b>load_program_file, set_delay_mode</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>	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</b> returns the error code 12 or 13 and otherwise has no effect.																										
<b>Call</b>	<code>ErrorNo = load_z_table( A, B, C )</code>																										
<b>Parameters</b>	<table> <tr> <td>A</td> <td>Coefficient <i>A</i> of the parabolic function <math>z_{\text{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>B</td> <td>Like <i>A</i> (analogously). Allowed value range: [-2048.0...2048.0].</td> </tr> <tr> <td>C</td> <td>Like <i>A</i> (analogously). Allowed value range: [-16.0...16.0].</td> </tr> </table>	A	Coefficient <i>A</i> of the parabolic function $z_{\text{out}} = A + BI + CI^2$ 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.	B	Like <i>A</i> (analogously). Allowed value range: [-2048.0...2048.0].	C	Like <i>A</i> (analogously). Allowed value range: [-16.0...16.0].																				
A	Coefficient <i>A</i> of the parabolic function $z_{\text{out}} = A + BI + CI^2$ 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.																										
B	Like <i>A</i> (analogously). Allowed value range: [-2048.0...2048.0].																										
C	Like <i>A</i> (analogously). Allowed value range: [-16.0...16.0].																										
<b>Result</b>	<p>Error code. As an unsigned 32-bit value.</p> <p>Error bits with values 1 to 64 can also occur combined, but not in conjunction with error code 11...14, 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 list execution status</b> or <b>INTERNAL-BUSY list execution status</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> not enabled.</td> </tr> <tr> <td>13</td> <td>No 3D correction table is currently assigned.</td> </tr> <tr> <td>14</td> <td>RTC5 board driver not found.</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 list execution status</b> or <b>INTERNAL-BUSY list execution status</b> error; for exact reason: see <b>get_last_error</b> ).	11	Access denied.	12	<b>Option "3D"</b> not enabled.	13	No 3D correction table is currently assigned.	14	RTC5 board driver not found.
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 list execution status</b> or <b>INTERNAL-BUSY list execution status</b> error; for exact reason: see <b>get_last_error</b> ).																										
11	Access denied.																										
12	<b>Option "3D"</b> not enabled.																										
13	No 3D correction table is currently assigned.																										
14	RTC5 board driver not found.																										

Ctrl Command	load_z_table
Comments	<ul style="list-style-type: none"> <li>• <b>load_z_table</b> is only needed for re-calibrating the z axis in a 3-axis scan system (for adjusting corresponding coefficients see <a href="#">page 159</a>). Both positive and negative coefficients can be specified. The coefficients should preferably be chosen so that all Z output values lie within the range [-32,768... +32,767].</li> <li>• <b>load_z_table</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status 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 <b>set_wait</b> (<b>PAUSED</b> list execution status set)</li> </ul> </li> <li>• <b>load_z_table</b> should always be used after <b>load_correction_file</b>, since <b>load_correction_file</b> sets the three coefficients to the default values of the loaded correction table.</li> <li>• Prior to the next list command that directly follows <b>load_z_table</b>, a smooth transition from the last z position to the changed position is performed at <b>jump_speed</b>. You can also immediately force this by <b>select_cor_table</b>. This way, time delays can be avoided during an <b>External Start</b>.</li> <li>• Coefficients A, B and C can be queried from the loaded 3D correction table by <b>get_table_para</b> (and from the currently assigned 3D correction table by <b>get_head_para</b>).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: changed value ranges and error codes. If the correct calibration factors are used (see <a href="#">Section "3D Commands", page 223</a> ), then the same ABC coefficients can be used on the same 3-axis scan system with the RTC4 and RTC5 boards.
Version info	–
References	<a href="#">get_z_distance</a> , <a href="#">read_abc_from_file</a> , <a href="#">write_abc_to_file</a> , <a href="#">select_cor_table</a>

Ctrl Command	load_zoom_correction_file
Comments	<ul style="list-style-type: none"> <li>• This command is described, for example, in the manual for the intelliWELD II FT.</li> </ul>



Normal List Command	<b>long_delay</b>
Function	Pauses further processing of the list for the specified time.
Call	<code>long_delay( Delay )</code>
Parameters	<p>Delay      Delay time. In bits.            As an unsigned 32-bit value.            1 bit equals 10 <math>\mu</math>s.            Allowed value range: <math>0 \leq \text{delay} \leq (2^{32}-1)</math>.</p>
Comments	<ul style="list-style-type: none"> <li>• <b>long_delay</b> switches off <a href="#">Signals for "Laser Active" Operation</a> after a <a href="#">LaserOff Delay</a>, 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→RTC5	Unchanged functionality (except for the increased range of values).
Version info	–
References	<a href="#">set_defocus_list</a>



Normal List Command	<b>mark_abs</b>
Function	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 <b>Image field</b> .
Call	<code>mark_abs( X, Y )</code>
Parameters	<p>X      Absolute x coordinate of the mark vector end point. In bits.  As a signed 32-bit value.  Allowed value range: [-8,388,608...+8,388,607].  Out-of-range values are clipped to the boundary values.  The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p>Y      Like X (analogously).</p>
Comments	<ul style="list-style-type: none"> <li>If the mark speed has not been previously explicitly set by <b>set_mark_speed</b> or <b>set_mark_speed_ctrl</b>, then the marking is executed at a predefined mark speed of 1,000 <i>bits/ms</i>.</li> <li>The <b>Signals for "Laser Active" Operation</b> are automatically turned on at the beginning of the marking (or remain on after a directly preceding <b>[*]mark[*]</b> Command or <b>"Arc"</b> command). The defined <b>Scanner Delays</b> and <b>Laser Delays</b> are thereby taken into account (see <b>Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control"</b>, page 133). Note that other delays are executed in <b>Sky Writing</b> mode.  Exception: zero-length <b>[*]mark[*]</b> Commands (see notes on page 137).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the values specified for X and Y by 16. The allowed value range decreases accordingly.
Version info	–
References	<b>set_mark_speed</b> , <b>set_scanner_delays</b> , <b>mark_rel</b> , <b>arc_abs</b> , <b>timed_mark_abs</b>

Normal List Command	<b>mark_abs_3d</b>
Function	Moves the laser focus at mark speed along a 3D vector from the current position to the specified position (absolute coordinate values) within a <b>3D image field</b> .
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
Call	<b>mark_abs_3d( X, Y, Z )</b>
Parameters	<p>X      Absolute x coordinate of the mark vector end point. In bits.  As a signed 32-bit value.  Allowed value range: [-8,388,608...+8,388,607].  Out-of-range values are clipped to the boundary values.  The complete value range is only usable as a virtual <b>Image field</b> 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: [-32,768...+32,767].</p>
Comments	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>mark_abs_3d</b> functions similarly to <b>mark_abs</b> (see comments there).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the values specified for X and Y by 16. The allowed value range decreases accordingly. The value range for Z is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .
Version info	–
References	<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>• <b>mark_char</b> reads the indexed character's starting address from the internal management table based on the supplied index and then calls <b>list_call</b> (see also the comments there), which then starts the corresponding command list.</li> <li>• <b>mark_char</b> 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 <b>Section "Normal, Short, Variable and Multiple List Commands"</b>, page 278). <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, <b>get_char_pointer</b> returns the value “-1” (that is, <math>2^{32}-1</math>).</li> <li>• If Char &gt; 1023, then <b>mark_char</b> is, already during loading, replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>• Absolute <b>Vector commands</b> and “<b>Arc</b>” <b>commands</b> execute absolutely after being called with <b>mark_char</b>. If the character needs to execute at various locations within the <b>Image field</b>, then either the command list can only contain relative <b>[*]mark[*] Commands</b>, “<b>Arc</b>” <b>commands</b> or <b>Jump commands</b> 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 <b>Chapter 6.5.2 “Character Sets and Text Strings”</b>, page 107.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 <b>Vector commands</b> and <b>"Arc" commands</b> 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>The <b>mark_char_abs</b> command reads the indexed character'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), 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→RTC5	New command.
Version info	–
References	<b>mark_char</b>



Normal List Command	<b>mark_date</b>
Function	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.
Call	<code>mark_date( Part, Mode )</code>
Parameters	<p>Part      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 (4 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>Mode      Selection of the format.</p> <p>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>

Normal List Command	<b>mark_date</b>
Comments	<ul style="list-style-type: none"> <li>Before marking dates (after every boot-up), the RTC5 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 196</a>).</li> <li>The complete date can be marked by multiple calls of <b>mark_date</b>.</li> <li><b>mark_date</b> 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 <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="#">page 108</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 107</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="#">RTC5_PARAM_ERROR</a>).</li> <li>Absolute <a href="#">Vector commands</a> and <a href="#">"Arc" commands</a> execute absolutely after being called with <b>mark_date</b>. If date markings need to execute at various locations within the <a href="#">Image field</a>, then the corresponding indexed text strings (or characters) can only contain relative <a href="#">[*]mark[*] Commands</a>, <a href="#">"Arc" commands</a> or <a href="#">Jump commands</a> 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 00:00 o'clock. Leap years are represented in both date styles.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>



Normal List Command	<b>mark_date_abs</b>
Function	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 <b>Vector commands</b> and <b>"Arc" commands</b> receive an offset (based on the current coordinates at the time of the call).
Call	<b>mark_date_abs( Part, Mode )</b>
Parameters	Part      See <b>mark_date</b> .
	Mode      See <b>mark_date</b> .
Comments	<ul style="list-style-type: none"> <li>The <b>mark_date_abs</b> command works like <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>.</li> <li>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→RTC5	New command.
Version info	–
References	<b>mark_date</b>

Normal List Command	mark_ellipse_abs						
Function	Moves the laser focus at mark speed along an elliptical arc around the specified midpoint (absolute coordinate values) within a 2D <b>Image field</b> .						
Restriction	For older RTC5 Boards with <b>DSP</b> version numbers < 2 ( <b>get_RTC_version</b> Bit #16...Bit #23), <b>mark_ellipse_abs</b> is neither executed nor replaced by <b>list_nop</b> ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</b> ).						
Call	<code>mark_ellipse_abs( X, Y, Alpha )</code>						
Parameters	<table> <tr> <td>X</td><td>Absolute x coordinate of the ellipse midpoint. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.</td></tr> <tr> <td>Y</td><td>Like X (analogously).</td></tr> <tr> <td>Alpha</td><td>Angle between elliptical half-axis a (defined by <b>set_ellipse</b>) and the x axis (the angle is referenced to the positive x direction, positive angle values correspond to counterclockwise angles). In degrees. As a 64-bit IEEE floating point value. Alpha gets normalized to the value range [0...&lt;360°].</td></tr> </table>	X	Absolute x coordinate of the ellipse midpoint. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.	Y	Like X (analogously).	Alpha	Angle between elliptical half-axis a (defined by <b>set_ellipse</b> ) and the x axis (the angle is referenced to the positive x direction, positive angle values correspond to counterclockwise angles). In degrees. As a 64-bit IEEE floating point value. Alpha gets normalized to the value range [0...<360°].
X	Absolute x coordinate of the ellipse midpoint. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.						
Y	Like X (analogously).						
Alpha	Angle between elliptical half-axis a (defined by <b>set_ellipse</b> ) and the x axis (the angle is referenced to the positive x direction, positive angle values correspond to counterclockwise angles). In degrees. As a 64-bit IEEE floating point value. Alpha gets normalized to the value range [0...<360°].						
Comments	<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 <b>set_ellipse</b>. For descriptions of the individual parameters, see also <b>Section "Ellipse Commands", page 126</b>.</li> <li>If the arc starting point defined by <b>mark_ellipse_abs</b> and <b>set_ellipse</b> does not equal the current position, then a "Hard jump" to the starting point is executed prior to marking, see also notes in <b>Section "Ellipse Commands", page 126</b>.</li> <li>See also all comments for <b>arc_abs</b>.</li> </ul>						
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for X and Y by 16. The allowed value range decreases accordingly.						
Version info	–						
References	<b>set_ellipse</b> , <b>mark_ellipse_rel</b> , <b>set_mark_speed</b> , <b>set_scanner_delays</b> , <b>arc_abs</b>						



Normal List Command	<b>mark_ellipse_rel</b>
Function	Moves the laser focus at mark speed along an elliptical arc around the specified midpoint (relative coordinate values) within a 2D <b>Image field</b> .
Restriction	For older RTC5 Boards with <b>DSP</b> version numbers < 2 ( <b>get_RTC_version</b> Bit #16...Bit #23), <b>mark_ellipse_rel</b> is neither executed nor replaced by <b>list_nop</b> ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</b> ).
Call	<code>mark_ellipse_rel( dx, dy, Alpha )</code>
Parameters	<p><b>dx</b> Relative x coordinate of the ellipse midpoint. In bits. As a signed 32-bit value. Allowed value range: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values.</p> <p><b>dy</b> Like <b>dx</b> (analogously).</p> <p><b>Alpha</b> Angle between elliptical half-axis a (defined by <b>set_ellipse</b>) and the x axis (see <b>mark_ellipse_abs</b>). In degrees. As a 64-bit IEEE floating point value.</p>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the ellipse midpoint are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>mark_ellipse_rel</b> is identical to <b>mark_ellipse_abs</b> (see the comments there).</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified values for <b>dx</b> and <b>dy</b> by 16. The allowed value range decreases accordingly.</p>
Version info	–
References	<b>mark_ellipse_abs, set_ellipse</b>



<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 <b>Image field</b> .
<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: [-8,388,608...+8,388,607]. Out-of-range values are clipped to the boundary values. The complete value range is only usable as a virtual <b>Image field</b> 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 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 the comments there).</li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <code>dx</code> and <code>dy</code> by 16. The allowed value range decreases accordingly.
<b>Version info</b>	–
<b>References</b>	<a href="#">mark_abs</a> , <a href="#">mark_rel_3d</a> , <a href="#">timed_mark_rel</a>

Normal List Command	<b>mark_rel_3d</b>
Function	Moves the laser focus at mark speed along a 3D vector from the current position to the specified position (relative coordinate values) in the <b>3D image field</b> .
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
Call	<b>mark_rel_3d( dx, dy, dz )</b>
Parameters	<p><b>dx</b>      Relative x coordinate of the mark vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> 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: [-32,768...+32,767].</p>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the mark vector 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>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified value for <b>dx</b> and <b>dy</b> by 16. The allowed value range decreases accordingly. The value range for <b>dz</b> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .
Version info	–
References	<b>mark_abs_3d, mark_abs, mark_rel, timed_mark_rel_3d</b>



Normal List Command	mark_serial
Function	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.
Call	<code>mark_serial( Mode, Digits )</code>
Parameters	<p>Mode      Selection of the serial number format and (de)activation of automatic serial number incrementing.            As an unsigned 32-bit value.  <math>Mode = 20 \times M_1 + 10 \times M_2 + M_3</math>            Allowed value 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>Digits      Number [0-12] of to-be-marked digits.            As an unsigned 32-bit value.            Allowed value range: [0-12]. Larger values are clipped.</p>
Comments	<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 has been incremented or not.</li> <li>If the incremented serial number exceeds <math>10^{Digits}</math>, then marking begins again at 0. The control command <b>set_max_counts</b> allows specification of the maximum number of <b>External Starts</b> and thus the maximum number of markings. Here, all markings of all serial-number-sets contribute jointly to the count.</li> </ul>

Normal List Command	mark_serial
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 <i>additional</i> 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 <b>list_call</b> (see also the comments there) an appropriate number of times, which then starts the corresponding command lists. The command lists must contain marking instructions for digits 0...9 (see <a href="#">page 108</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 107</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>RTC5_PARAM_ERROR</b>).</li> <li>Absolute Vector commands 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 <b>Image field</b>, then the corresponding indexed text strings (or characters) can only contain relative [*]mark[*] Commands, "Arc" commands or Jump commands or <b>mark_serial_abs</b> must be used instead.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_serial</a> , <a href="#">set_serial_step</a> , <a href="#">set_serial_step_list</a> , <a href="#">get_list_serial</a> , <a href="#">set_max_counts</a> , <a href="#">get_counts</a> , <a href="#">load_text_table</a> , <a href="#">set_text_table_pointer</a> , <a href="#">mark_serial_abs</a>



Normal List Command	<b>mark_serial_abs</b>
Function	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 <b>Vector commands</b> 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.
Call	<code>mark_serial_abs( Mode, Digits )</code>
Parameters	Mode      See <b>mark_serial</b> .
	Digits      See <b>mark_serial</b> .
Comments	<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>.</li> <li>• 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→RTC5	New command.
Version info	–
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>	Text      PC memory address of the first character (byte) of the to-be-marked text string. As a pointer to a \0-terminated string.
<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> in the list memory (keep this in mind to prevent unintended overflow of the corresponding list memory area). During processing of the individual <b>mark_text</b>, 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 the <b>mark_text</b> by <b>select_char_set</b>. For the default setting, character set 0 is used. If <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>RTC5_REJECTED</b>) and the indexed subroutine is not stored.</li> <li>Absolute <b>Vector commands</b> and "<b>Arc</b>" <b>commands</b> execute absolutely after being called by <b>mark_text</b>. If the text string needs to execute at various locations within the <b>Image field</b>, then either the indexed character definitions can only contain relative <b>[*]mark[*]</b> <b>Commands</b>, "<b>Arc</b>" <b>commands</b> or <b>Jump commands</b> or <b>mark_text_abs</b> must be used instead.</li> <li>The <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→RTC5	New command.
Version info	–
References	<b>mark_text_abs</b>



<b>Variable List Command</b>	<b>mark_text_abs</b>
<b>Function</b>	Marks a \0-terminated string. In the called indexed characters (see below), any absolute <b>Vector commands</b> and <b>"Arc" commands</b> receive an offset (based on the current coordinates at the time of the call).
<b>Call</b>	<code>mark_text_abs( Text )</code>
<b>Parameters</b>	Text      PC memory address of the first character (byte) of the to-be-marked text string. As a pointer to a \0-terminated string.
<b>Comments</b>	<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→RTC5	New command.
Version info	–
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> <p>= 0: Hours (24-h time, no a.m./p.m.)  = 1: Minutes  = 2: Seconds  = 3: Hours (12-h time, no a.m./p.m.)  = 4: a.m./p.m. (automatically switched in accordance with 24-h time)</p> <p>As an unsigned 32-bit value.  Allowed value range: [0...4].</p> <p><b>Mode</b>      Format.</p> <p>As an unsigned 32-bit value.  Allowed value range: [0...3], only affects <b>Part</b> = 0...3).</p> <p>a) Number of leading zeros:  Bit #0 = 0: Leading zeros are suppressed.  Bit #0 = 1: Always two digits.</p> <p>b) Character set choice for marking of digits:  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.  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>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 RTC5 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 <b>Chapter 7.5 "Marking Dates, Times and Serial Numbers"</b>, page 196).</li> <li>The complete time can be marked by multiple calls of <b>mark_time</b>.</li> </ul>

Normal List Command	<b>mark_time</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The <b>mark_time</b> command reads (according to the stored time and according to the selected time part) the starting address of the corresponding indexed text string (or – for <code>Part = 0...3</code> and <code>Mode = 2</code> or <code>3</code> – of the corresponding indexed character) from the internal management table and then calls <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 <code>0...9</code> and for a.m./p.m. (see <a href="#">page 108</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 107</a>).</li> <li>If <code>Part &gt; 4</code> and/or <code>Mode &gt; 3</code>, then <b>mark_time</b> is, already during loading, replaced by a <b>list_nop</b> (<code>get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>Absolute Vector commands and "Arc" commands execute absolutely after being called with <b>mark_time</b>. If time markings need to execute at various locations within the <b>Image field</b>, then the corresponding indexed text strings (or characters) can only contain relative <code>[*]mark[*]</code> Commands, "Arc" commands or Jump commands or <b>mark_time_abs</b> must be used instead.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 <b>Vector commands</b> and <b>"Arc" commands</b> receive an offset (based on the current coordinates at the time of the call).
<b>Call</b>	<b>mark_time_abs( Part, Mode )</b>
<b>Parameters</b>	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>.</li> <li>• 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>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">mark_time</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> (Multi channel Buffered Serial Port, see also <a href="#">page 71</a> ).				
<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 <code>mcbsp_init</code> is <i>not</i> executed (<a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>The initialized values (after program start) are <code>XDelay = RDelay = 1</code>.</li> <li>The signals and operating conditions of the <b>McBSP interface</b> are presented in <a href="#">Chapter 4.6.6 "SPI / I2C Socket Connector", page 71</a>.</li> <li>The <b>McBSP interface</b> ignores the first FrameSync signal after a <code>mcbsp_init</code>. That is, data provided is <i>not</i> transmitted, see <a href="#">page 73</a>.</li> </ul>				
RTC4→RTC5	New command.				
Version info	–				
References	<a href="#">read_mcbsp</a> , <a href="#">set_mcbsp_out</a> , <a href="#">set_mcbsp_out_ptr</a> , <a href="#">set_mcbsp_freq</a> , <a href="#">mcbsp_init_spi</a>				

<b>Ctrl Command</b>	<b>mcbsp_init_spi</b>
<b>Function</b>	Initializes the <b>SPI</b> functionality at the <b>SPI / I2C Socket Connector</b> (instead of McBSP functionality).
<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 "SPI / I2C Socket Connector", page 71</a>.</li> <li>With the <b>SPI</b> protocol a common <b>CLOCK</b> signal, data input and data output occur simultaneously. The RTC5 is the <b>SPI</b> master.</li> <li><b>mcbsp_init_spi</b> can be called at any time. Data transmissions in progress (started but not yet finished) are cancelled.</li> <li>The clock frequency can be set in advance by <a href="#">set_mcbsp_freq</a>.</li> <li><b>mcbsp_init</b> can be used to reestablish the McBSP functionality at the <b>SPI / I2C Socket Connector</b> at any time.</li> <li>Irrespective of the clock frequency the <b>SPI</b> functionality only permits a single transmission every 10 <math>\mu</math>s (for example, see <a href="#">set_mcbsp_in</a>).</li> <li>The <b>McBSP interface</b> ignores the first FrameSync signal after a <b>mcbsp_init_spi</b>. That is, data provided is not transmitted, see <a href="#">page 73</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	Available as of DLL 538, OUT 538.
<b>References</b>	<a href="#">mcbsp_init</a> , <a href="#">set_mcbsp_freq</a>



<b>Ctrl Command</b>	<b>measurement_status</b>	
<b>Function</b>	Returns the status of a measurement session started by <b>set_trigger</b> or <b>set_trigger4</b> and the current position of the measurement counter.	
<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 counter (within the RTC5 measurement storage area) ( $0 \leq \text{Pos} \leq 2^{20}-1$ or $0 \leq \text{Pos} \leq (2^{19}-1)$ or $\text{Pos} = 2^{32}-1$ , see below). 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 <b>set_trigger</b> or <b>set_trigger4</b> is no longer active, then <code>Pos+1</code> indicates the number of recorded data pairs up to termination (maximum <math>2^{20}</math> for <b>set_trigger</b>, maximum <math>2^{19}</math> for <b>set_trigger4</b>).</li> <li><code>Pos = 2^{32}-1</code> indicates that data recording still has not occurred after <b>load_program_file</b>.</li> <li>Stored data can be queried with <b>get_waveform</b>.</li> <li>The status of a measurement session is reset by <b>set_trigger</b>(<code>Period = 0</code>), <b>set_trigger4</b>(<code>Period = 0</code>), <b>stop_trigger</b> or <b>stop_execution</b> (see <b>set_trigger</b> comments).</li> </ul>	
RTC4→RTC5	Unchanged functionality.	
<b>Version info</b>	–	
<b>References</b>	<b>set_trigger</b> , <b>set_trigger4</b>	

Normal List Command	micro_vector_abs
Function	Moves the output point (of the laser focus) by a "Hard jump" (without split-up into <b>Microsteps</b> ) directly from the current position to the specified position (absolute coordinate values) within a 2D <b>Image field</b> .
Call	micro_vector_abs( <b>X</b> , <b>Y</b> , <b>LasOn</b> , <b>LasOff</b> )
Parameters	<p><b>X</b>      Absolute x coordinate of the micro vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>Y</b>      Like <b>X</b> (analogously).</p> <p><b>LasOn</b>      <b>LaserOn Delay</b>. 1 Bit equals 0.5 <math>\mu</math>s.            As a signed 32-bit value.            Allowed value range: [-<math>2^{31}</math>...+(<math>2^{15}</math>-1)].  <math>\geq 0</math>:      Delay is newly set. Values over (<math>2^{15}</math>-1) are clipped.  <math>&lt; 0</math>:      The previously set delay continues unaffected.</p> <p><b>LasOff</b>      <b>LaserOff Delay</b>.            Like <b>LasOn</b>.</p>
Comments	<ul style="list-style-type: none"> <li>See also <a href="#">Chapter 8.8 "micro_vector[*] Commands", page 249</a>.</li> <li>Wobbel 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 <b>Microvector</b> is always executed as a "Hard jump".</li> <li>By <b>LasOn</b> <math>\geq 0</math> and <b>LasOff</b> <math>\geq 0</math>, you can set a new <b>LaserOn Delay</b> or <b>LaserOff Delay</b> for each individual <b>Microvector</b>. Each delay thereby gets set at the end of the clock cycle in which the new position actually gets outputted (this output clock cycle is delayed by a preceding scanner delay).</li> <li>Negative values (<b>LasOn</b> <math>&lt; 0</math> and <b>LasOff</b> <math>&lt; 0</math>) do not affect <b>Laser Delays</b>. Hereby, the laser can remain on or off across multiple clock cycles (mark and jump simulation).</li> <li>Delays set with <b>LasOn</b> and <b>LasOff</b> only apply to the execution of <b>Microvectors</b>. For execution of normal <b>[*]mark[*] Commands</b> and <b>"Arc" commands</b> (such as <b>mark_abs</b>), only the <b>Laser Delays</b> defined by <b>set_laser_delays</b> apply. <b>LasOn</b> and <b>LasOff</b> do not overwrite the laser delay parameter from <b>set_laser_delays</b>.</li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16, those for <b>LasOn</b> and <b>LasOff</b> by 2. The allowed value ranges decrease accordingly.
Version info	–
References	<a href="#">micro_vector_rel</a> , <a href="#">micro_vector_abs_3d</a>

Normal List Command	<b>micro_vector_abs_3d</b>
Function	Moves the output point (of the laser focus) by a “ <b>Hard jump</b> ” (without split-up into <b>Microsteps</b> ) directly from the current position to the specified position (absolute coordinate values) within the <b>3D image field</b> .
Restriction	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> .
Call	<code>micro_vector_abs_3d( X, Y, Z, LasOn, LasOff )</code>
Parameters	<p><b>X</b>      Absolute x coordinate of the micro vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> 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: [-32,768...+32,767].</p> <p><b>LasOn</b>    <b>LaserOn Delay</b>. 1 Bit equals 0.5 <math>\mu</math>s.            As a signed 32-bit value.            Allowed value range: [-2<sup>31</sup>...+(2<sup>15</sup>-1)].  <math>\geq 0</math>:    Delay is newly set. Values over (2<sup>15</sup>-1) are clipped.  <math>&lt; 0</math>:    The previously set delay continues unaffected.</p> <p><b>LasOff</b>   <b>LaserOff Delay</b>. 1 bit equals 1/64 <math>\mu</math>s.            Like <b>LasOn</b>.</p>
Comments	<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>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16, those for <b>LasOn</b> and <b>LasOff</b> by 2. The allowed value range decreases accordingly. The value range for <b>Z</b> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .
Version info	–
References	<b>micro_vector_abs</b> , <b>micro_vector_rel_3d</b>



Normal List Command	<b>micro_vector_rel</b>
Function	Moves the output point (of the laser focus) by a “ <b>Hard jump</b> ” (without split-up into <b>Microsteps</b> ) directly from the current position to the specified position (relative coordinate values) within a 2D <b>Image field</b> .
Call	<code>micro_vector_rel( dX, dY, LasOn, LasOff )</code>
Parameters	<p>dX      Relative x coordinate of the micro vector end point. In bits. Otherwise, like <b>x</b> from <b>micro_vector_abs</b>.</p> <p>dY      Relative y coordinate of the micro vector end point. In bits. Otherwise, like <b>y</b> from <b>micro_vector_abs</b>.</p> <p>LasOn    Like <b>LasOn</b> from <b>micro_vector_abs</b>.</p> <p>LasOff   Like <b>LasOff</b> from <b>micro_vector_abs</b>.</p>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the micro vector end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>micro_vector_rel</b> is identical to <b>micro_vector_abs</b> (see comments there).</li> <li>The <b>Microvector</b> is always executed as a “<b>Hard jump</b>”.</li> </ul>
RTC4→RTC5	New command. <b>RTC4 Compatibility Mode:</b> see <b>micro_vector_abs</b> .
Version info	–
References	<b>micro_vector_abs</b> , <b>micro_vector_rel_3d</b>

Normal List Command	<b>micro_vector_rel_3d</b>
Function	Moves the output point (of the laser focus) by a "Hard jump" (without split-up into <b>Microsteps</b> ) directly from the current position to the specified position (relative coordinate values) within a <b>3D image field</b> .
Restriction	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> .
Call	<code>micro_vector_rel_3d( dx, dy, dz, LasOn, LasOff )</code>
Parameters	<p>dx      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>dy      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>dz      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>LasOn    Like <b>LasOn</b> from <b>micro_vector_abs_3d</b>.</p> <p>LasOff   Like <b>LasOff</b> from <b>micro_vector_abs_3d</b>.</p>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the micro vector 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 <b>Microvector</b> is always executed as a "Hard jump".</li> </ul>
RTC4→RTC5	New command. <b>RTC4 Compatibility Mode:</b> see <b>micro_vector_abs_3d</b> .
Version info	–
References	<b>micro_vector_abs_3d</b> , <b>micro_vector_rel</b>

Ctrl Command	<b>move_to</b>
Comments	<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" (#0112097).</li> <li><b>move_to</b> is not supported by "RTC5/6 varioSCAN 40 FLEX Extension" extension board (#0128683). See also <b>Chapter 2.8.8 "Extension Board "RTC5/6 varioSCAN FLEX Extension"</b>, page 39.</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...4].      Default after <a href="#">load_program_file</a>: 4. See also <a href="#">Chapter 8.5.3 "Using Several Correction Tables", page 226</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For outside the allowed value range, <b>number_of_correction_tables</b> is ignored (<a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</li> <li><b>number_of_correction_tables</b> 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 RTC5 error messages) in the future.</li> <li><b>number_of_correction_tables</b> refers only to subsequent command executions. Existing assignments of correction tables cannot be corrected automatically. This is particularly important, if RTC5 boards that have been initialized by other user programs are subsequently acquired.</li> <li>On request: 8 allowed correction tables, see Footnote, <a href="#">page 164</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 540.
References	<a href="#">load_correction_file</a> , <a href="#">select_cor_table</a> , <a href="#">select_cor_table_list</a>



Normal List Command	<b>para_jump_abs</b>
Function	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 <b>Image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
Call	<code>para_jump_abs( X, Y, P )</code>
Parameters	<p><b>X</b>      Absolute x coordinate of the jump vector end point. In bits.  As a signed 32-bit value.  Allowed value range: [-8,388,608...+8,388,607].  Out-of-range values are clipped to the boundary values.  The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>Y</b>      Like <b>X</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>
Comments	<ul style="list-style-type: none"> <li>If vector-defined 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 <b>P</b> is then ignored.</li> <li>If vector-defined 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 <b>P</b>, see <b>Section "Vector-Defined Laser Control", page 194</b>.</li> <li>There is no abs mechanism for <b>P</b>. <b>P</b> is clipped to the maximum allowed value.</li> <li>If <b>para_jump_abs</b> is used along with position-dependent 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 <b>Section "Vector-Defined Laser Control", page 194</b>).</li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16. The allowed value ranges decrease accordingly. There is no <b>RTC4 Compatibility Mode</b> for the parameter <b>P</b> . The original RTC5 units must be used.
Version info	–
References	<b>jump_abs</b> , <b>set_vector_control</b> , <b>para_jump_abs_3d</b> , <b>para_jump_rel</b>

Normal List Command	para_jump_abs_3d
Function	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 a <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.
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
Call	para_jump_abs_3d( <b>X</b> , <b>Y</b> , <b>Z</b> , <b>P</b> )
Parameters	<p><b>X</b>      Absolute x coordinate of the jump vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> 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: [-32,768...+32,767].</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>
Comments	<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 <b>para_jump_abs</b> (see the comments there).</li> <li>Further comments see <b>jump_abs_3d</b>.</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16. The allowed value range decreases accordingly. The value range for <b>Z</b> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b>.</p> <p>For the parameter <b>P</b> see <b>para_jump_abs</b>.</p>
Version info	–
References	<b>jump_abs_3d</b> , <b>set_vector_control</b> , <b>para_jump_abs</b> , <b>para_jump_rel_3d</b>



Normal List Command	para_jump_rel
Function	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 <b>Image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
Call	para_jump_rel( <i>dx</i> , <i>dy</i> , <i>P</i> )
Parameters	<p><i>dx</i>      Relative x coordinate of the jump vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p><i>dy</i>      Like <i>dx</i> (analogously).</p> <p><i>P</i>      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>
Comments	<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><i>P</i> is not treated on a relative basis.</li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <i>dx</i> and <i>dy</i> by 16. The allowed value range decreases accordingly. For the parameter <i>P</i> , see <b>para_jump_abs</b> .
Version info	–
References	<b>jump_rel</b> , <b>set_vector_control</b> , <b>para_jump_abs</b> , <b>para_jump_rel_3d</b>

Normal List Command	para_jump_rel_3d
Function	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 a <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.
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
Call	para_jump_rel_3d( dx, dy, dz, P )
Parameters	<p>dx      Relative x coordinate of the jump vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p>dy      Like dx (analogously).</p> <p>dz      Like dx (analogously), except            Allowed value range: [-32,768...+32,767].</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>
Comments	<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 the comments there).</li> <li>P is not treated on a relative basis.</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified values for dx and dy by 16. The allowed value range decreases accordingly. The value range for dz is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b>.</p> <p>The parameter P does not have a <b>RTC4 Compatibility Mode</b>. The original RTC5 units must be used.</p>
Version info	–
References	<a href="#">jump_rel_3d</a> , <a href="#">set_vector_control</a> , <a href="#">para_jump_abs_3d</a> , <a href="#">para_jump_rel</a>

Variable List Command	para_laser_on_pulses_list
Function	Turns on the LASERON signal for the specified number of external signal pulses (but for no longer than the specified time interval) and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
Call	para_laser_on_pulses_list( <i>Period</i> , <i>Pulses</i> , <i>P</i> )
Parameters	<p><i>Period</i> Time interval. In bits.      As an unsigned 32-bit value.      1 bit equals 10 <math>\mu</math>s.      Allowed value range: <math>0 \leq \text{Period} \leq (2^{32}-1)</math>.</p> <p><i>Pulses</i> 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><i>P</i> End value of the signal parameter.      As an unsigned 32-bit value. Allowed value range: dependent on the selection made by <b>set_vector_control</b> (<i>Ctrl</i> parameter), identical with <b>set_vector_control</b> (<i>value</i> parameter).</p>
Comments	<ul style="list-style-type: none"> <li>• <b>para_laser_on_pulses_list</b> is useful for marking single dots, see <a href="#">Chapter 7.1.3 "Marking Single Dots", page 129</a>.</li> <li>• If vector-defined 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 <i>Pulses</i> &gt; 65,535 – like <b>laser_on_list</b> (see comments there). The parameter <i>P</i> is then ignored.</li> <li>• If vector-defined laser control is activated, then the signal parameter selected by <b>set_vector_control</b> is linearly varied from the last valid value to <i>P</i> within the <b>para_laser_on_pulses_list</b> duration (<i>Period</i> <math>\times</math> 10 <math>\mu</math>s) (see <a href="#">Section "Vector-Defined Laser Control", page 194</a>).</li> <li>• There is no abs mechanism for <i>P</i>. <i>P</i> is clipped to the maximum allowed value. This maximum value is <math>(2^{31}-1)</math> or a lower value depending on what has been selected with <b>set_vector_control</b> (<i>Ctrl</i> parameter).</li> <li>• If <b>para_laser_on_pulses_list</b> is used along with position-dependent or speed-dependent laser control for the same control parameter, then the current value of <i>P</i> is used as the basis of the 100% value for laser control (see <a href="#">Section "Vector-Defined Laser Control", page 194</a>).</li> <li>• If <i>Period</i> = 0, <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 <i>Period</i> clocks (that is, <i>Period</i> <math>\times</math> 10 <math>\mu</math>s), 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→RTC5	New command. For the parameter $P$ , see <a href="#">para_jump_abs</a> .
Version info	–
References	<a href="#">laser_on_pulses_list</a> , <a href="#">laser_on_list</a>

Normal List Command	para_mark_abs
Function	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 <b>Image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
Call	para_mark_abs( <b>X</b> , <b>Y</b> , <b>P</b> )
Parameters	<p><b>X</b>      Absolute x coordinate of the mark vector end point. In bits.  As a signed 32-bit value.  Allowed value range: [-8,388,608...+8,388,607].  Out-of-range values are clipped to the boundary values.  The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p><b>Y</b>      Like <b>X</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>
Comments	<ul style="list-style-type: none"> <li>If vector-defined 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 <b>P</b> is then ignored.</li> <li>If vector-defined 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 <b>P</b> (see <b>Section "Vector-Defined Laser Control", page 194</b>).</li> <li>There is no abs mechanism for <b>P</b>. <b>P</b> is clipped to the maximum allowed value.</li> <li>For mark vector lengths = 0, no change of signal parameter <b>P</b> must be programmed: <ul style="list-style-type: none"> <li>This change is not outputted</li> <li>The following <b>[*]para[*]</b> Command (only this one) produces incorrect signal parameter outputs</li> </ul> </li> <li>If <b>para_mark_abs</b> is used along with position-dependent 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 <b>Section "Vector-Defined Laser Control", page 194</b>).</li> <li><b>[*]para_mark[*]</b> commands generally do not take <b>Sky Writing</b> into account.</li> </ul>
RTC4→RTC5	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16. The allowed value range decreases accordingly. For the parameter <b>P</b> , see <b>para_jump_abs</b> .
Version info	–
References	<b>mark_abs</b> , <b>set_vector_control</b> , <b>para_mark_abs_3d</b> , <b>para_mark_rel</b>

Normal List Command	para_mark_abs_3d
Function	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.
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
Call	para_mark_abs_3d( <b>X</b> , <b>Y</b> , <b>Z</b> , <b>P</b> )
Parameters	<p><b>X</b>      Absolute x coordinate of the mark vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> 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: [-32,768...+32,767].</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>
Comments	<ul style="list-style-type: none"> <li>Except for the additional motion in the third dimension, <b>para_mark_abs_3d</b> functions similarly to the <b>para_mark_abs</b> command (see the comments there).</li> <li>Further comments see <b>mark_abs_3d</b>.</li> <li>[*]<b>para_mark</b>[*] commands generally do not take <b>Sky Writing</b> into account.</li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16. The allowed value range decreases accordingly. The value range for <b>Z</b> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> . For the parameter <b>P</b> , see <b>para_jump_abs</b> .
Version info	–
References	<b>mark_abs_3d</b> , <b>set_vector_control</b> , <b>para_mark_abs</b> , <b>para_mark_rel_3d</b>

Normal List Command	para_mark_rel
Function	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 <b>Image field</b> . Simultaneously varies the signal parameter selected by <b>set_vector_control</b> linearly to the specified value.
Call	para_mark_rel( dx, dy, p )
Parameters	<p>dx      Relative x coordinate of the mark vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p>dy      Like dx (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>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the mark vector 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>p is not treated on a relative basis.</li> <li>[*]para_mark[*] commands generally do not take <b>Sky Writing</b> into account.</li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for dx and dy by 16. The allowed value range decreases accordingly. For the parameter p see <b>para_jump_abs</b> .
Version info	–
References	<b>mark_rel</b> , <b>set_vector_control</b> , <b>para_mark_abs</b> , <b>para_mark_rel_3d</b>

Normal List Command	para_mark_rel_3d
Function	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.
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
Call	para_mark_rel_3d( dx, dy, dz, P )
Parameters	<p>dx      Relative x coordinate of the mark vector end point. In bits.            As a signed 32-bit value.            Allowed value range: [-8,388,608...+8,388,607].            Out-of-range values are clipped to the boundary values.            The complete value range is only usable as a virtual <b>Image field</b> for example, for (enabled) Processing-on-the-fly applications.</p> <p>dy      Like x (analogously).</p> <p>dz      Like x (analogously), except            Allowed value range: [-32,768...+32,767].</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> (Ctrl parameter), identical with <b>set_vector_control</b> (Value parameter).</p>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the mark vector 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 the comments there).</li> <li>P is not treated on a relative basis.</li> <li>[*]para_mark[*] commands generally do not take <b>Sky Writing</b> into account.</li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified value for dx and dy by 16. The allowed value range decreases accordingly. The value range for dz is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> . For the parameter P, see <b>para_jump_abs</b> .
Version info	–
References	<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 <b>Image field</b> .
<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 <b>Image field</b>. Allowed value range: [-524,288...524287].          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 <b>Image field</b>. During a subsequent list interruption (by <b>wait_for_encoder</b> etc.), the galvanometer scanners remain 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 <b>Image field</b>. Allowed value range: [-8388608...8388607].          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 <b>Image field</b> (subject to Processing-on-the-fly correction and clipped to the real <b>Image field</b>'s boundaries).          During a subsequent list interruption (by <b>wait_for_encoder</b> etc.), the positions of the galvanometer scanners 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 <b>Image field</b>'s boundaries.</p>
	<p>X      Absolute x coordinate of the jump vector end point. In bits.          As a signed 32-bit value.          Allowed value range: see above.          Out-of-range values are clipped to the boundary values.</p>
	<p>Y      Like X (analogously).</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 intermediate 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 <b>Jump command</b>, as does the return jump by <b>park_return</b> (but Processing-on-the-fly remains switched off).</li> </ul>



<b>Variable List Command</b>	<b>park_position</b>
Comments (cont'd)	<ul style="list-style-type: none"><li>• If the laser focus has been 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 <b>Signals for "Laser Active" Operation</b> after a <b>LaserOff Delay</b>, but does not activate a scanner delay afterward.</li></ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>X</b> and <b>Y</b> by 16. The allowed value range decreases accordingly.
Version info	–
References	<b>park_return</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 <b>Image field</b> .
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> is not enabled, then <b>park_return</b> functions as a normal <b>Jump command</b> . 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 x coordinate of the jump vector end point in the virtual <b>Image field</b>.                In bits.                As a signed 32-bit value.                Allowed value range: [-8,388,608...+8,388,607].                Out-of-range values are clipped to the boundary values.</p> <p>Y        Like X (analogously).</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 motion (following list interruption by <b>wait_for_encoder</b> etc.) (see <b>Chapter 8.6.7 "Synchronizing Processing-on-the-fly Applications"</b>, page 237, 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 has been 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 has been 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> has been called, whereas <b>park_return</b>( Mode = 1 ) moves them to the position specified with the command. When calculating the new position, the RTC5 takes the current Processing-on-the-fly correction into account. But if clipping to the boundaries of the virtual <b>Image field</b> occurs (for example, due to a too long intermediate 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>



<b>Variable List Command</b>	<b>park_return</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>If the laser focus <i>has not been</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 has been 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 <b>Image field</b> is active, then the specified or most recent valid position is subsequently appropriately transformed (see <a href="#">page 158</a>).</li> <li>A scanner <b>Jump Delay</b> is activated after a <b>park_return</b>.</li> <li><b>park_return</b> switches off the <b>Signals for "Laser Active" Operation</b> after a <b>LaserOff Delay</b>.</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified values for X and Y by 16. The allowed value range decreases accordingly.</p>
Version info	–
References	<b>park_position</b>



Ctrl Command	<b>pause_list</b>
Function	Pauses execution of the list and disables the <b>Signals for "Laser Active" Operation</b> .
Call	<code>pause_list()</code>
Parameters	–
Comments	<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 <b>Signals for "Laser Active" Operation</b> 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 <b>Microsteps</b>. The <b>PAUSED list execution status</b> (queryable by <b>get_status</b>) is set, but the <b>BUSY list execution status</b> is left unchanged. Continuation by <b>execute_list_pos</b> or <b>release_wait</b> or by an <b>External Start</b> is not possible. However, <b>stop_execution</b> or an <b>External Stop</b> is possible. <b>restart_list</b>, <b>stop_execution</b> or an <b>External Stop</b> 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 motions (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 list execution status</b> is then reset (here too, <b>BUSY list execution status</b> remains unchanged).</li> <li>• If, during calling of <b>pause_list</b>, no list is currently executing (<b>BUSY list execution status</b> not set) or a list has already been halted by <b>pause_list</b> or <b>set_wait</b> (<b>PAUSED list execution status</b> set), then <b>pause_list</b> is ignored (<b>get_last_error</b> return code <b>RTC5_BUSY</b>; the laser is then already off).</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 executed (<code>get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>• See also <a href="#">Chapter 9.4 "Periodical I/O Signals", page 277</a>.</li> </ul>												
RTC4→RTC5	New command.												
Version info	Last change DLL 543, OUT 543: 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 <math>\mu</math>s] (&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 <math>\mu</math>s] (&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 <math>\mu</math>s]. 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 $\mu$ s] (> 0, 16 bit max.). As an unsigned 32-bit value.	P2	Duration of the toggled signal in [10 $\mu$ s] (> 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 $\mu$ s]. 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 $\mu$ s] (> 0, 16 bit max.). As an unsigned 32-bit value.												
P2	Duration of the toggled signal in [10 $\mu$ s] (> 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 $\mu$ s]. 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>RTC5_PARAM_ERROR</code> is generated.</li> <li><code>Mask</code> 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</math>s clock cycles. Then they are toggled once again and kept stable for <math>P2 \times 10 \mu</math>s clock cycles. 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 <code>periodic_toggle_list</code>.</li> <li>The first toggling occurs after <code>Start</code> <math>\times 10 \mu</math>s clock cycles. 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>As of version DLL 543, OUT 543, <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 277</a>.</li> </ul>												



<b>Undelayed Short List Command</b>	<b>periodic_toggle_list</b>
RTC4→RTC5	New command.
Version info	Last change DLL 543, OUT 543: 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→RTC5	Unchanged functionality.
Version info	–
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 scanner exceeds its position limit.	
<b>Call</b>	range_checking( HeadNo, Mode, Data )	
<b>Parameters</b>	HeadNo	<p>Scan head 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 <a href="#">Signals for "Laser Active" Operation</a> are suppressed immediately. List execution continues. 1: The <a href="#">Signals for "Laser Active" Operation</a> are switched-off immediately. List execution is stopped immediately (as with <a href="#">stop_execution</a> or <a href="#">/STOP</a>).</p>
	Data	<p>Data type chosen to be monitored. As an unsigned 32-bit value.</p> <p>0: Sample values. Like <a href="#">set_trigger</a> signal types 7...9. 1: Transformed control values. Like <a href="#">set_trigger</a> signal types 25...30. 2: Corrected control values. Like <a href="#">set_trigger</a> signal types 10...15. 3: Actual output values. Like <a href="#">set_trigger</a> signal types 20...23. 4: Actual position of galvanometer scanners (only with <a href="#">iDRIVE</a> systems, <a href="#">varioSCAN</a> is internally connected to an x axis). Like <a href="#">set_trigger</a> signal types [1, 2 and 4] or [4, 5 and 1]. 5: Actual position of galvanometer scanners (only with <a href="#">iDRIVE</a> systems, <a href="#">varioSCAN</a> is internally connected to a y axis). Like <a href="#">set_trigger</a> signal types [1, 2 and 5] or [4, 5 and 2].</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> </ul>	



<b>Undelayed Short List Command</b>	<b>range_checking</b>
Comments (cont'd)	<ul style="list-style-type: none"> <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> <li>Processing-on-the-fly applications use the data type <code>Data = 0</code> (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 <code>Data &gt; 0</code>.</li> <li>If the laser has been switched-off with <code>Mode = 0</code> 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><code>Data &gt; 5</code> causes a <a href="#">get_last_error</a> <code>RTC5_PARAM_ERROR</code> error code. In this case, <b>range_checking</b> is sent to the RTC5 Board as <a href="#">list_nop</a>.</li> <li><code>Data=2</code> and <code>Data=3</code> have the identical effect for the z axis.</li> <li>For <code>Data = 4</code> and <code>Data = 5</code> users must set the set position as the to-be-returned data type by the scan system. A simultaneous use of a speed-dependent laser control with <code>Mode = 2</code> (see <a href="#">set_auto_laser_control</a>) is not possible.</li> <li><code>Data = 4</code> and <code>Data = 5</code> only differ in regards to the axis onto an varioSCAN is connected. For 2D systems without varioSCAN both selections have the identical effect.</li> <li><code>Data = 4</code> and <code>Data = 5</code> produce nonsensical switch-offs if an intelliSCAN is connected but is either not switched-on or a actual position has not been set as feedback.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_fly_limits</a> , <a href="#">set_fly_limits_z</a>



<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 (RTC5 DLL-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 $z_{out} = A + B/I + C/I^2$ 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 RTC5 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 119</a>, are neither generated nor altered by <b>read_abc_from_file</b>.</li> </ul>	
RTC4→RTC5	New command.	
Version info	Available as of DLL 538, OUT 538.	
References	<a href="#">vwrite_abc_to_file</a>	

<b>Ctrl Command</b>	<b>read_analog_in</b>
<b>Function</b>	<p>Reads in the analog input values:</p> <ul style="list-style-type: none"> <li>• <b>ANALOG IN0</b> and <b>ANALOG IN1</b> from the <b>ADC Add-On Board</b> of the <b>RTC5 PCI Board</b></li> <li>• <b>ANALOG IN0</b> and <b>ANALOG IN1</b> from the <b>SPI/I<sup>2</sup>C</b> socket connector of the <b>RTC5 PCIe Board</b></li> </ul> <p>Returns them as 12-bit digital values.</p>
<b>Call</b>	<code>AnalogValue = read_analog_in()</code>
<b>Result</b>	<p>As an unsigned 32-bit value.</p> <p>Bit #0                   <b>ANALOG IN0</b> input value as 12-bit digital value.</p> <p>...</p> <p>Bit #11</p> <p>Bit #12               = 0. Channel number.</p> <p>Bit #13               = 0.</p> <p>Bit #14               = 0.</p> <p>Bit #15               Old bit for <b>ANALOG IN0</b>.</p> <p>Bit #16               <b>ANALOG IN1</b> input value as 12-bit digital value.</p> <p>...</p> <p>Bit #27</p> <p>Bit #28               = 1. Channel number.</p> <p>Bit #29               = 0.</p> <p>Bit #30               = 0.</p> <p>Bit #31               Old bit for <b>ANALOG IN1</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>read_analog_in</b> can only be used with <b>RTC5 PCIe Boards</b> and <b>RTC5 PCI Boards</b> having the <b>ADC Add-On Board</b> installed (see <b>Chapter 4.6.8 "Analog Inputs (Accessory)"</b>, page 75 and <b>Chapter 16.2.4 "SPI/I<sup>2</sup>C Socket Connector"</b>, page 742).</li> <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>• As of version DLL 543, OUT 543, RBF 524: Analog input values (<b>ANALOG IN0</b> and <b>ANALOG IN1</b>) can be recorded with signal 54 (see <b>set_trigger</b>/<b>set_trigger4</b>). However, old bits are not recorded. The format of the data is <math>((ANALOG IN1) &lt;&lt; 16) + (ANALOG IN0)</math>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<b>set_trigger</b> , <b>set_trigger4</b>



<b>Ctrl Command</b>	<b>read_encoder</b>
<b>Function</b>	Returns the counts of the two RTC5 encoder counters that were stored by <b>store_encoder</b> .
<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      Count.  As a pointer to a signed 32-bit value.  For parameter "Encoder<sub>n</sub>_m":</p> <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 storage position (parameter <i>Pos</i> of <b>store_encoder</b>).</li> </ul> <p>Encoder1_0      Like Encoder0_0 (analogously).</p> <p>Encoder0_1      Like Encoder0_0 (analogously).</p> <p>Encoder1_1      Like Encoder0_0 (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <b>Chapter 8.6 "Processing-on-the-fly", page 227</b> and <b>Chapter 9.3.3 "Synchronization by Encoder Signals", page 274</b>.</li> <li>• If the workpiece motion is registered with an incremental encoder: <ul style="list-style-type: none"> <li>– Encoder counter "Encoder0" is triggered by the signals at encoder input port <b>ENCODER X</b></li> <li>– Encoder counter "Encoder1" is triggered by the signals at encoder input port <b>ENCODER Y</b></li> </ul> </li> <li>• If an encoder simulation has been started by <b>simulate_encoder( 3 )</b>, both encoder counters are triggered by an internal periodic 1 MHz clock signal.</li> <li>• For storage positions in which counts were not previously stored by <b>store_encoder</b>, the value 0 is returned (initialized value).</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>store_encoder, get_encoder, set_fly_x, set_fly_y, set_fly_rot, wait_for_encoder</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	<code>IOPort = read_io_port()</code>
Result	16-bit value (DIGITAL IN0...DIGITAL IN15). As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"><li>• See also <a href="#">Chapter 9.2.1 "16-Bit Digital Input Port", page 264</a>.</li></ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
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 <a href="#">read_io_port_list</a> (input value read at the EXTENSION 1 socket connector's 16-bit digital input; the galvanometer scanner set position and time value that has been current at the time of reading).
<b>Call</b>	CurrentIndex = <code>read_io_port_buffer( Index, &amp;Value, &amp;XPos, &amp;YPos, &amp;Time )</code>
<b>Parameters</b>	<p>Index      Number of the to-be-returned entry in the IOPort buffer.                  As an unsigned 32-bit value.                  Only the 4 least significant bits are evaluated (the IOPort buffer only has 16 entries). Therefore, the user program can use a continuous counter for the index.</p>
<b>Returned parameter values</b>	<p>Value      16-bit value.                  As a pointer to an unsigned 32-bit value.</p> <p>XPos      Set position of the galvanometer scanner (x value).                  As a pointer to a signed 32-bit value.</p> <p>YPos      Set position of the galvanometer scanner (y value).                  As a pointer to a signed 32-bit value.</p> <p>Time      Time. In seconds.                  As a pointer to an unsigned 32-bit value.</p>
<b>Result</b>	The index to which data is written upon the next <a href="#">read_io_port_list</a> . As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also comments for <a href="#">read_io_port_list</a>.</li> <li>• If no <a href="#">read_io_port_list</a> has been called before <a href="#">read_io_port_buffer</a>, then the initialization values (default: 0) are returned.</li> </ul>
<b>Example (C/C++)</b>	<p>Wait until the data under Index 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 Index 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→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">read_io_port_list</a>



<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 <i>relative</i> time in seconds are stored.
Call	<code>read_io_port_list()</code>
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 <b>read_io_port_buffer</b>.</li> <li>• The IOPort buffer holds 16 entries and is circularly organized. It always stores the 16 most recent entries and overwrites older entries without warning. The incremental Index starts after a reset (program start) at Index 0, and after passing 15 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 264</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 has been 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> <p>0      Memory location for Processing-on-the-fly applications and for <b>set_mcbsp_in</b> input with coding Bit #31 = 0.</p> <p>1      Memory locations for <b>Online Positioning</b> and for <b>set_mcbsp_in</b> input.</p> <p>2      Wie 1.</p> <p>3      Memory location for <b>set_mcbsp_in</b> input with coding Bit #31 = 1. 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 2 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 "SPI / I2C Socket Connector", page 71</a>.</li> <li>The interpretation as <ul style="list-style-type: none"> <li>one signed or unsigned 32-bit data word</li> <li>two signed 16-bit data words</li> <li>two signed 15-bit data words</li> </ul> is the responsibility of the user: <ul style="list-style-type: none"> <li>For Processing-on-the-fly applications (<code>No = 0</code>) 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/SPI Signals", page 276</a>.</li> <li>For <b>"Local Online Positioning"</b> (<code>No = 1...2</code>) see <a href="#">Chapter 8.3.1 ""Local Online Positioning"", page 213</a>.</li> <li>For <b>"Global Online Positioning"</b> (<code>No = 1...2</code>) see <a href="#">Chapter 8.3.2 ""Global Online Positioning"", page 216</a>.</li> <li>For <b>set_mcbsp_in</b> input (<code>No = 0...3</code>) 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 594</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> ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>. That is, data provided is not transmitted, see <a href="#">page 73</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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_status</b>																				
<b>Function</b>	Returns the RTC5 list status, see <a href="#">Chapter 6.4.2 "List Status", page 96</a> .																				
<b>Call</b>	Status = <code>read_status()</code>																				
<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 (<a href="#">LSB</a>)</td> <td>LOAD1 = 1:</td> <td>Indicates that the input pointer is currently in "List 1", that is, that all following list commands are stored in "List 1". <a href="#">LOAD1</a> is set if the input pointer is set into "List 1" (for example, by <a href="#">set_start_list_pos</a>). <a href="#">LOAD1</a> is reset if a <a href="#">set_end_of_list</a> command is written to "List 1" (<a href="#">READY1</a> set) or if <a href="#">LOAD2</a> is set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #1</td> <td>LOAD2 = 1:</td> <td>Indicates that the input pointer is currently in "List 2", that is, that all following list commands are stored in "List 2". <a href="#">LOAD2</a> is set if the input pointer is set into "List 2" (for example, by <a href="#">set_start_list_pos</a>). <a href="#">LOAD2</a> is reset if a <a href="#">set_end_of_list</a> command is written to "List 2" (<a href="#">READY2</a> set) or if <a href="#">LOAD1</a> is set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #2</td> <td>READY1 = 1:</td> <td>Indicates that during the loading procedure a <a href="#">set_end_of_list</a> command has been written to "List 1". <a href="#">READY1</a> is reset if <a href="#">LOAD1</a> is newly set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #3</td> <td>READY2 = 1:</td> <td>Indicates that during the loading procedure a <a href="#">set_end_of_list</a> command has been written to "List 2". <a href="#">READY2</a> is reset if <a href="#">LOAD2</a> is newly set (for example, by <a href="#">set_start_list_pos</a>).</td> </tr> <tr> <td>Bit #4</td> <td>BUSY1 = 1:</td> <td>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" has been started). <a href="#">BUSY1</a> 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). <a href="#">BUSY1</a> 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 ( <a href="#">LSB</a> )	LOAD1 = 1:	Indicates that the input pointer is currently in "List 1", that is, that all following list commands are stored in "List 1". <a href="#">LOAD1</a> is set if the input pointer is set into "List 1" (for example, by <a href="#">set_start_list_pos</a> ). <a href="#">LOAD1</a> is reset if a <a href="#">set_end_of_list</a> command is written to "List 1" ( <a href="#">READY1</a> set) or if <a href="#">LOAD2</a> 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". <a href="#">LOAD2</a> is set if the input pointer is set into "List 2" (for example, by <a href="#">set_start_list_pos</a> ). <a href="#">LOAD2</a> is reset if a <a href="#">set_end_of_list</a> command is written to "List 2" ( <a href="#">READY2</a> set) or if <a href="#">LOAD1</a> 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> command has been written to "List 1". <a href="#">READY1</a> is reset if <a href="#">LOAD1</a> 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> command has been written to "List 2". <a href="#">READY2</a> is reset if <a href="#">LOAD2</a> 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" has been started). <a href="#">BUSY1</a> 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). <a href="#">BUSY1</a> 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 ( <a href="#">LSB</a> )	LOAD1 = 1:	Indicates that the input pointer is currently in "List 1", that is, that all following list commands are stored in "List 1". <a href="#">LOAD1</a> is set if the input pointer is set into "List 1" (for example, by <a href="#">set_start_list_pos</a> ). <a href="#">LOAD1</a> is reset if a <a href="#">set_end_of_list</a> command is written to "List 1" ( <a href="#">READY1</a> set) or if <a href="#">LOAD2</a> 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". <a href="#">LOAD2</a> is set if the input pointer is set into "List 2" (for example, by <a href="#">set_start_list_pos</a> ). <a href="#">LOAD2</a> is reset if a <a href="#">set_end_of_list</a> command is written to "List 2" ( <a href="#">READY2</a> set) or if <a href="#">LOAD1</a> 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> command has been written to "List 1". <a href="#">READY1</a> is reset if <a href="#">LOAD1</a> 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> command has been written to "List 2". <a href="#">READY2</a> is reset if <a href="#">LOAD2</a> 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" has been started). <a href="#">BUSY1</a> 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). <a href="#">BUSY1</a> 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" has been started). <b>BUSY2</b> is set by starting execution of "List 2" (for example, by <b>execute_list_pos</b> ) or by a list change to "List 2" (automatic list change or jump). <b>BUSY2</b> is reset if <b>set_end_of_list</b> is executed in "List 2", if a jump into "List 1" is executed (for example, <b>list_jump_pos</b> ), if "List 1" is started (for example, by <b>execute_list_pos</b> ) or if <b>stop_execution</b> is executed.
	Bit #6	USED1	= 1:	Indicates that a <b>set_end_of_list</b> command has been reached during processing of "List 1". <b>USED1</b> is reset when <b>LOAD1</b> is set (for example, by <b>set_start_list_pos</b> ).
	Bit #7	USED2	= 1:	Indicates that a <b>set_end_of_list</b> command has been reached during processing of "List 2". <b>USED2</b> is reset when <b>LOAD2</b> is set (for example, by <b>set_start_list_pos</b> ).
	Bit #8		0	
	...			
	Bit #31			
Comments	<ul style="list-style-type: none"> <li>When interpreting the status values returned by <b>read_status</b>, 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, the <b>LOAD list status</b> can already have been reset and the <b>READY list status</b> set if – after loading of a <b>set_end_of_list</b> into a list – further list commands are loaded into the same list.</li> <li>– The status values remain unchanged if a <b>set_end_of_list</b> command is overwritten with another command (<b>READY list status</b> is not reset).</li> <li>– If – during a loading process – a <b>set_end_of_list</b> command is processed at the same time in the same list, then the list is regarded as already processed (<b>USED list status</b> set), even though it is still newly loaded (<b>USED list status</b> has been then initially reset).</li> <li>– Even if a completely loaded command list (incl. <b>set_end_of_list</b>) has been stored, the <b>READY list status</b> of a list can be reset if the input pointer has been newly set (for example, by <b>set_input_pointer</b>) into the list. Then a list's <b>USED list status</b> 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 <b>list_jump_pos</b>) during execution, the <b>USED list status</b> values of both lists (unlike their <b>BUSY list execution status</b> values) remain unchanged and are therefore not meaningful.</li> </ul>			



Ctrl Command	read_status
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 has been originally called).</li> <li>If list execution is interrupted (by <b>pause_list</b>, <b>stop_list</b> or <b>set_wait</b>), then the above-mentioned status values remains unchanged.</li> <li>The <b>List Execution Status</b> values <b>BUSY</b> list execution status and <b>PAUSED</b> list execution status can be queried by <b>get_status</b>.</li> <li>To read the status signals from the scan heads, use <b>get_head_status</b>.</li> </ul>
RTC4→RTC5	Basically unchanged functionality. However: Additional <b>USED</b> list status.
Version info	–
References	<b>get_status</b> , <b>get_head_status</b>

Ctrl Command	read_user_data
Function	No function.
Comments	<ul style="list-style-type: none"> <li>To date, <b>read_user_data</b> has no effect.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>send_user_data</b>

<b>Ctrl Command</b>	<b>release_rtc</b>
<b>Function</b>	Releases the specified RTC5 for use by other user programs.
<b>Call</b>	NoOfReleasedCard = <code>release_rtc( CardNo )</code>
<b>Parameters</b>	CardNo <b>RTC5 DLL</b> -internal number of the RTC5 Board. As an unsigned 32-bit value.
<b>Result</b>	The return value is <code>CardNo</code> if the user program has been 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_rtc5_dll</b>.</li> <li>If a board released by <b>release_rtc</b> has been 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 <b>select_rtc</b>.</li> <li><b>release_rtc</b> is available even without explicit access rights for a particular RTC5 Board, but <b>release_rtc</b> then has no effect (return value 0).</li> <li><b>release_rtc</b> also has no effect (return value 0), if: <ul style="list-style-type: none"> <li>CardNo exceeds the number of RTC5 boards found during initialization (see <b>rtc5_count_cards</b>)</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→RTC5	New command.
Version info	–
References	<b>acquire_rtc, init_rtc5_dll</b>



Ctrl Command	<b>release_wait</b>
Function	Resumes execution of a list that has been interrupted by a <b>set_wait</b> .
Call	<code>release_wait()</code>
Comments	<ul style="list-style-type: none"> <li>• <b>release_wait</b> is only executed if the RTC5 is actually in a wait state (hence if a wait marker has been previously reached and execution has been halted; the <b>PAUSED list execution status</b> is then set but <i>not</i> the <b>BUSY list execution status</b>). Otherwise, <b>release_wait</b> is ignored (<b>get_last_error</b> return code <b>RTC5_BUSY</b>).</li> <li>• <b>release_wait</b> resets the <b>PAUSED list execution status</b> and newly sets the <b>BUSY list execution status</b> (both queryable with <b>get_status</b>, see also <a href="#">Chapter 6.4.3 "List Execution Status", page 97</a>).</li> <li>• <b>release_wait</b> resets the <b>WaitWord</b> that receives the break point number during an interrupt to zero.</li> <li>• If a home jump (defined with <b>home_position</b> or <b>home_position_xyz</b>) has been executed by <b>set_wait</b>, then <b>release_wait</b> leads to a corresponding home return (the <b>INTERNAL-BUSY list execution status</b> is set while the home return is executed).</li> <li>• The wait state can be queried by <b>get_wait_status</b>.</li> </ul>
RTC4→RTC5	Basically unchanged functionality. However: The RTC5 also provides a <b>PAUSED list execution status</b> . It is set with <b>set_wait</b> and reset with <b>release_wait</b> .
Version info	–
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>Code</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 119</a>.</li> <li>The cumulative error code <code>AccError</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 = RTC5_BUSY   RTC5_REJECTED</code>)</li> <li>Completely (by <code>Code = "-1"</code>, therefore by <code>Code = 2^32-1</code>) The meanings of bit numbers, error types and error constants is described at <a href="#">get_error</a>.</li> </ul> </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 even available without explicit access rights to a specific RTC5 Board.</li> <li>The board-specific error variable <code>LastError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 119</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→RTC5	New command.
Version info	–
References	<a href="#">get_error</a> , <a href="#">get_last_error</a>



<b>Ctrl Command</b>	<b>restart_list</b>
<b>Function</b>	Reenables <b>Signals for “Laser Active” Operation</b> and resumes execution of a list that has been 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>• <b>restart_list</b> is only executed, if a list has been previously halted by <b>pause_list</b> or <b>stop_list</b> and if the <b>BUSY list execution status</b> and <b>PAUSED list execution status</b> (queryable with <b>get_status</b>) are set. Otherwise (for example, if a list has been halted by <b>set_wait</b>), <b>restart_list</b> is ignored (<b>get_last_error</b> return code <b>RTC5_BUSY</b>).</li> <li>• <b>restart_list</b> resets the <b>PAUSED list execution status</b>. The <b>BUSY list execution status</b> is left unchanged.</li> <li>• In general, an interrupted marking cannot be continued without a disruption in the marking result.</li> </ul>
<b>RTC4→RTC5</b>	Basically unchanged functionality. However: The RTC5 also provides a <b>PAUSED list execution status</b> that is set with <b>pause_list</b> and <b>stop_list</b> and reset by <b>restart_list</b> .
<b>Version info</b>	–
<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>BaudRate    Baud rate.            As an unsigned 32-bit value.            Allowed value range: [300...+115200].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Default value: 9600 baud.</li> <li>• The other RS-232 interface parameters cannot be altered:           <ul style="list-style-type: none"> <li>– Data bits: 8</li> <li>– Start bits: 1</li> <li>– Stop bits: 1</li> <li>– Parity: none</li> </ul> </li> <li>• See also <b>Chapter 4.6.5 “RS232 Socket Connector”, page 70</b>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<b>rs232_write_data</b> , <b>rs232_read_data</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="#">page 264</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. (<a href="#">LSB</a>)</p> <p>...</p> <p>Bit #7</p> <p>Bit #8            "New" bit: = 1:    The value is new (has not been previously read). = 0:    The value is old (has been 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).</li> </ul> <p><i>Important: To reset the overflow counter, you must call <code>rs232_read_data</code> correspondingly often!</i></p> <ul style="list-style-type: none"> <li>Example: return value 459098 = 0x0007015A = (0, 7, 1, 90) means: character 90 ('Z') has been read and is new (1), 7 additional characters remain to be read, the buffer has been never overrun (0).</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">rs232_write_data</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 263</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
<b>References</b>	<a href="#">rs232_write_text</a> , <a href="#">rs232_read_data</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><code>rs232_write_text</code> is split into an appropriate number of <a href="#">rs232_write_data</a>.</li> <li>During <code>rs232_write_text</code> execution no further control commands can execute, but the board's list execution is not affected. If an executing list itself contains <a href="#">rs232_write_text_list</a>, <code>rs232_write_text</code> 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 263</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
<b>References</b>	<a href="#">rs232_write_data</a> , <a href="#">rs232_write_text_list</a>



<b>Variable List Command</b>	<b>rs232_write_text_list</b>
<b>Function</b>	Sends a text string (character-by-character) to the RS-232 interface.
<b>Call</b>	<code>rs232_write_text_list( pData )</code>
<b>Parameters</b>	<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.
<b>Comments</b>	<ul style="list-style-type: none"> <li>When <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 list memory area). Processing of the individual <b>rs232_write_text_list</b> is similar to that of <b>rs232_write_text</b> (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 263</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">rs232_write_text</a>



Ctrl Command	rtc5_count_cards
Function	Returns the number of RTC5 Boards detected during initialization.
Call	NoOfCards = rtc5_count_cards()
Result	Number of RTC5 Boards. As an unsigned 32-bit value.
Comments	<ul style="list-style-type: none"><li>Initialization of the installed RTC5 Boards is to be made separately for each user program by calling <a href="#">init_rt5_dll</a>.</li><li>rtc5_count_cards is available even without explicit access rights to a specific RTC5 Board.</li><li>rtc5_count_cards is not available as a multi-board command.</li><li>The board-specific error variables <a href="#">LastError</a> and <a href="#">AccError</a> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by rtc5_count_cards.</li></ul>
RTC4→RTC5	New command. The functionalities of rtc5_count_cards and the RTC4 command <a href="#">rtc4_count_cards</a> are identical.
Version info	–
References	–



<b>Delayed Short List Command</b>	<b>save_and_restart_timer</b>
Function	Stores the current value of the <b>RTC5 Timer</b> and resets it to zero.
Call	<code>save_and_restart_timer()</code>
Comments	<ul style="list-style-type: none"> <li>The stored <b>RTC5 Timer</b> value can be read by <b>get_time</b>.</li> <li><b>save_and_restart_timer</b> is useful for measuring the marking time of a marking process, see <a href="#">Chapter 8.12 "Time Measurements", page 257</a>.</li> <li>The <b>RTC5 Timer</b> only counts list-command clock cycles. Counting is paused during interruptions by <b>set_wait</b> or <b>pause_list</b>.</li> <li>By <b>get_lap_time</b> the number of elapsed list-command clock cycles since the reset to zero can be queried.</li> <li>To compare RTC5-internal <b>save_and_restart_timer</b> time measurements to external time measurements by the BUSY pin, you should insert a <b>list_nop</b> between <b>save_and_restart_timer</b> and <b>set_end_of_list</b>. This ensures that any scanner delay completes before <b>set_end_of_list</b>. Without <b>list_nop</b>, <b>save_and_restart_timer</b> includes the scanner delay in its measurement even though it completes only after <b>set_end_of_list</b> (and therefore the BUSY pin is already low).</li> <li>As of DLL 543, OUT 543, RBF 524 a <b>32-bit "Timestamp Counter"</b> is available in addition to the <b>RTC5 Timer</b>. It starts counting at 0 with <b>load_program_file</b> and counts uninterruptible all 10 <math>\mu</math>s clock periods. See also <a href="#">Chapter 8.12 "Time Measurements", page 257</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
References	<a href="#">get_lap_time</a> , <a href="#">get_time</a>
Version info	–



<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 storage medium, ordered by index, and returns the number of thereby stored list commands.				
<b>Call</b>	<code>NoOfSavedCommands = save_disk( Name, Mode )</code>				
<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. Bit #2...Bit #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. Bit #2...Bit #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. Bit #2...Bit #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>The <b>save_disk</b> command can be used together with <b>load_disk</b>, for example, to perform defragmentation or to apply subsequent protection to subroutines (see <a href="#">page 105</a> and <a href="#">page 104</a>).</li> <li><b>save_disk</b> stores complete sets (no individual characters, text strings or subroutines!) in the specified file. Indexed characters, text strings or subroutines that are referenced multiple times with <b>copy_dst_src</b> are also correspondingly stored multiple times by <b>save_disk</b>. <b>save_disk</b> ignores unreferenced non-indexed (not subsequently referenced by <b>set_char_pointer</b>,...) characters, text strings and subroutines.</li> <li>The number of list commands stored by <b>save_disk</b> can differ from the number of the list commands stored in the protected list memory area "List 3" (= <math>2^{20} - \text{Mem1} - \text{Mem2} - \text{get_list_space}</math>), due to the following: <ul style="list-style-type: none"> <li>Indexed characters/text strings/subroutines are referenced several times</li> <li>Indexed characters/text strings/subroutines reside in the unprotected list memory area "List 1" or "List 2"</li> </ul> Prior a subsequent <b>load_disk</b>, be sure to compare the returned number with the size of the protected list memory area "List 3" (= <math>2^{20} - \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 <a href="#">page 104</a>).</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>. <b>Jump commands</b> (also branches to various <b>list_return</b> commands) are thereby neither evaluated nor executed.</li> <li>• <b>save_disk</b> 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>RTC5_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>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY list execution status</b> is set</li> <li>– the <b>INTERNAL-BUSY list execution status</b> 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 list execution status</b> set)</li> </ul> </li> <li>• During the runtime of <b>save_disk</b>: <ul style="list-style-type: none"> <li>– No further commands are executed</li> <li>– <b>External Starts</b> are suppressed.</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>load_disk</b>



<b>Undelayed Short List Command</b>	<b>select_char_set</b>
<b>Function</b>	Selects one of the available character sets (exclusively) for the execution of subsequent <b>mark_text</b> and <b>mark_text_abs</b> .
<b>Call</b>	<code>select_char_set( No )</code>
<b>Parameters</b>	No      Number of the desired character set. As an unsigned 32-bit value. Allowed value range: [0...3].
<b>Comments</b>	<ul style="list-style-type: none"> <li>The selection remains valid until another is encountered.</li> <li>The default value (after initialization) is <code>No = 0</code>.</li> <li>If <code>No &gt; 3</code>, then <b>select_char_set</b> is replaced by a <b>list_nop</b>. The current character set then remains valid (<b>get_last_error</b> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>A character set change <i>within</i> a <b>mark_text</b> or <b>mark_text_abs</b> command 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>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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 connectors and activates <b>Image field correction</b> .
<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...4: 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...4: 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><b>select_cor_table</b> or <b>select_cor_table_list</b> should be called directly after loading the desired correction table(s) by <b>load_correction_file</b> and/or <b>load_z_table</b> (or after a subsequent <b>load_program_file</b> command, see below) in order to assign the correction table(s) to the corresponding connectors (<b>select_cor_table</b> is automatically called by <b>load_correction_file</b>, see <a href="#">Notes, page 165</a>). <b>select_cor_table</b> and <b>select_cor_table_list</b> issue a jump with <b>jump_speed</b> (no "Hard jump") to the corrected galvanometer scanner position, so that <b>Image field</b> correction is immediately applied to the currently set position of the galvanometer scanners. <b>select_cor_table</b> does this automatically and immediately, whereas <b>select_cor_table_list</b> 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, <b>goto_xy</b>) that sets the scan system's galvanometer scanners in motion, and <b>select_cor_table</b> or <b>select_cor_table_list</b> should only be started <i>after</i> loading of the desired correction table(s). Otherwise, the galvanometer scanners 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 <b>load_correction_file</b>) <i>prior</i> to <b>load_program_file</b>, are not fully effective before <b>select_cor_table</b> or <b>select_cor_table_list</b> is called. <b>load_program_file</b> deletes correction tables number 3 and 4.</li> <li><b>select_cor_table</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY list execution status</b> is set</li> </ul> The list command <b>select_cor_table_list</b> can be used without restrictions.</li> <li><b>select_cor_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 list execution status</b> set)</li> </ul> </li> <li>If the <b>INTERNAL-BUSY list execution status</b> is set, <b>select_cor_table</b> is only executed with a delay (after <b>INTERNAL-BUSY list execution status</b> has been reset again).</li> </ul>

Ctrl Command	select_cor_table
Comments (cont'd)	<ul style="list-style-type: none"> <li>The <b>INTERNAL-BUSY</b> list execution status is set while the jump to the corrected galvanometer scanner position is executed.</li> <li>If the values for parameters <code>HeadA</code> or <code>HeadB</code> are invalid (all 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> has not been 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 <code>(1, 0)</code>, 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 <code>HeadA = 1</code> and <code>HeadB = 0</code>). 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 <code>(0 0)</code>, even if the correction table content is different.</li> <li>If the <b>Option "Second Scan Head Control"</b> is disabled, 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 <b>Chapter 8.5 "Controlling 2D Scan Systems and 3D Scan Systems", page 221</b>.</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: <b>select_cor_table</b><code>(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 RTC5's <b>Option "3D"</b> is enabled, you can load several (2D or 3D) correction tables. If the <b>Option "Second Scan Head Control"</b> is not enabled, then x axis and y 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 RTC5 Boards with enabled <b>Option "3D"</b> are installed in a PC, then that many 3-axis systems can be simultaneously controlled.</li> </ul>



Ctrl Command	select_cor_table
Comments (cont'd)	<ul style="list-style-type: none"> <li>Provided the <a href="#">Option "3D"</a> and the <a href="#">Option "Second Scan Head Control"</a> 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 54</a> and <a href="#">Section "2D Correction Files 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> 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> <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→RTC5	Unchanged functionality. Exception: several 3D correction tables can be stored in RTC5 memory, see <a href="#">Chapter 8.5.3 "Using Several Correction Tables", page 226</a> . However, two 3D correction tables <i>cannot</i> be simultaneously assigned to the scan head connectors.
Version info	–
References	<a href="#">select_cor_table_list</a> , <a href="#">load_correction_file</a> , <a href="#">load_program_file</a> , <a href="#">number_of_correction_tables</a>



<b>Variable List Command</b>	<b>select_cor_table_list</b>
<b>Function</b>	Like <a href="#">select_cor_table</a> , but a list command.
<b>Call</b>	<code>select_cor_table_list( HeadA, HeadB )</code>
<b>Parameters</b>	HeadA      Like <a href="#">select_cor_table</a> .
	HeadB      Like <a href="#">select_cor_table</a> .
<b>Comments</b>	<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 <math>\mu</math>s, even if the correction table does not specify a correction.</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<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 RTC5 Board for a user program. See <a href="#">Chapter 6.6 "Using Several RTC5 PCI Boards in One PC", page 112</a> .
<b>Call</b>	NoOfSelectedCard = select_rtc( CardNo )
<b>Parameters</b>	CardNo <b>RTC5 DLL</b> -internal number of the RTC5 Board. As an unsigned 32-bit value.
<b>Result</b>	<p>The return 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 RTC5 Boards found during initialization 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>RTC5_ACCESS_DENIED</b> and possibly <b>RTC5_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>RTC5 DLL</b> for this user program is initialized (see <a href="#">init_rtc5_dll</a>). The <b>select_rtc</b> command offers the possibility of changing the active board at any time. All the user program's 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>RTC5_ACCESS_DENIED</b>).</li> <li>If no access rights are assigned for the specified board, and it is not in use by another user program, then the board is not only activated but also automatically reserved for this user program and therefore, unavailable to other user programs (return value: CardNo). 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>RTC5_ACCESS_DENIED RTC5_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><b>select_rtc</b> also has no effect if CardNo exceeds the number of RTC5 Boards found during initialization (see <a href="#">rtc5_count_cards</a>) 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><b>select_rtc</b> is available even without explicit access rights to a particular RTC5 Board.</li> <li><b>select_rtc</b> is not available as a multi-board command.</li> </ul>



Ctrl Command	select_rtc
RTC4→RTC5	With the RTC4, <b>select_rtc</b> only activates the specified board (unconditionally). With the RTC5, <b>select_rtc</b> additionally returns a return value and assigns access rights (as by <b>acquire_rtc</b> ) or it has no effect.
Version info	–
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 <a href="#">load_program_file</a>, serial-number-set 0 is selected.</li> <li><code>select_serial_set</code> does <i>not</i> assign a serial-number-set for serial number marking. The list command <a href="#">select_serial_set_list</a> is intended for that purpose.</li> <li>For <code>select_serial_set</code> usage, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 196</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 <a href="#">load_program_file</a>, serial-number-set 0 is selected.</li> <li>For <code>select_serial_set_list</code> usage, see <a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 196</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>
<b>Function</b>	No function.
<b>Comments</b>	<ul style="list-style-type: none"><li>• To date, <b>send_user_data</b> has no effect.</li></ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<b>read_user_data</b>



<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 209</a> ).
<b>Call</b>	<code>set_angle( HeadNo, Angle, 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 first scan head connector.                    = 2: The definition only affects the second scan head connector.                    = 0, 3: The definition affects <i>both</i> scan head connectors.                    = 4: The definition affects the virtual <b>Image field</b> (see also comments).                    Only the 3 least significant bits are evaluated.</p> <p>Angle      Rotation angle. In degrees.                    As a 64-bit IEEE floating point value.                    Positive angles result in a counterclockwise rotation around the centerpoint of the <b>Image field</b>.</p> <p>at_once      Defines when the defined transformation becomes effective.                    As an unsigned 32-bit value.                    Like <b>set_matrix</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See also <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a>.</li> <li>• The coordinate transformation defined by HeadNo = 4 is only in effect during a Processing-on-the-fly application initiated by <b>set_fly_2d</b>, see <a href="#">Section "Coordinate Transformations in the Virtual Image Field", page 158</a>:       <ul style="list-style-type: none"> <li>– <b>set_fly_2d</b></li> <li>– <math>\geq</math> DLL 541 <b>set_fly_x</b></li> <li>– <math>\geq</math> DLL 541 <b>set_fly_y</b></li> </ul>       Here, the at_once parameter is ignored.     </li> </ul>
RTC4→RTC5	New command.
Version info	Last change DLL 545, OUT 545: 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 <b>set_angle</b> , but a list command.
<b>Call</b>	<code>set_angle_list( HeadNo, Angle, at_once )</code>
<b>Parameters</b>	<p>HeadNo      Like <b>set_angle</b>. However, HeadNo = 4 is not available.</p> <p>Angle      Like <b>set_angle</b>.</p> <p>at_once      Determines when the defined transformation becomes effective.</p> <ul style="list-style-type: none"> <li>= 0:      The transformation settings are only collected 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 collected until then) and processed prior to the next list command. If necessary, <b>Signals for "Laser Active" Operation</b> are switched off in advance.</li> <li>= 2:      The transformation settings are only collected and intermediately stored (as with at_once = 0). However, The transformation is immediately calculated (including all transformation settings that were collected and intermediately stored until then) and applied to the current position when the next <b>jump_abs</b> or <b>jump_rel</b> (only if no list is currently being executed: also <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>As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• HeadNo = 4 is not available.</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<b>set_angle</b>

Ctrl Command	set_auto_laser_control	
Function	Initializes or deactivates "Automatic Laser Control", see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a> , <a href="#">page 187</a> .	
Call	ErrorCode = set_auto_laser_control( Ctrl, Value, Mode, MinValue, MaxValue )	
Parameters	Ctrl	<p>Control parameter. As an unsigned 32-bit value.</p> <p>= 1...6: Defines which signal parameter is corrected by "Automatic Laser Control".</p> <ul style="list-style-type: none"> <li>= 1: 12-bit output value at the <b>ANALOG OUT1</b> output port.</li> <li>= 2: 12-bit output value at the <b>ANALOG OUT2</b> output port.</li> <li>= 3: Output value at the 8-bit digital output port.</li> <li>= 4: Pulse length (<b>PulseLength</b>) of the laser signals LASER1 and LASER2.</li> <li>= 5: Output period (<b>HalfPeriod</b>) of the laser signals LASER1 and LASER2.</li> <li>= 6: Output value at the 16-bit digital output.</li> </ul> <p>= 0 or &gt; 6: Deactivates "Automatic Laser Control" (for Ctrl &gt; 6: <b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</p>
Value		<p>Defines the 100% value for the parameter selected by Ctrl. As an unsigned 32-bit value.</p> <p>Allowed values:</p> <ul style="list-style-type: none"> <li>For Ctrl = 1/2: 12-bit values [0...4095]. Bits with higher significance are ignored.</li> <li>For Ctrl = 3: 8-bit values [0...255]. Bits with higher significance are ignored.</li> <li>For Ctrl = 4: [0...(2<sup>32</sup>-1)]. 1 bit equals 1/64 <math>\mu</math>s.</li> <li>For Ctrl = 5: [0...(2<sup>32</sup>-1)]. 1 bit equals 1/64 <math>\mu</math>s.</li> <li>For Ctrl = 6: 16-bit values [0...(2<sup>16</sup>-1)]. Bits with higher significance are ignored.</li> </ul>
Mode		<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>• <code>M</code> = +4 encoder speed-dependent correction</li> </ul> <p>Then the following applies: <code>Mode</code> = 2 + sum of extensions <code>M</code>.</p> <p>For <code>Mode</code> = 0 or none of the above described ones:</p> <p>Disables the speed-dependent or encoder-speed-dependent laser control (<a href="#">get_last_error</a>-Returncode: <code>RTC5_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 189</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>ErrorCode.</p> <p>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 23</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>RTC5_ACCESS_DENIED</code>).</td></tr> </table>		0	No error.	1	No first scan head active.	2	No iDRIVE scan system (see Glossary entry on <a href="#">page 23</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>RTC5_ACCESS_DENIED</code> ).
0	No error.													
1	No first scan head active.													
2	No iDRIVE scan system (see Glossary entry on <a href="#">page 23</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>RTC5_ACCESS_DENIED</code> ).													
Comments	<ul style="list-style-type: none"> <li>• For <code>set_auto_laser_control</code> usage, see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a>, <a href="#">page 187</a>.</li> <li>• <code>MinValue</code> and <code>MaxValue</code> are automatically exchanged if <code>MinValue</code> &gt; <code>MaxValue</code> and <code>MinValue</code> is clipped to <math>\leq</math> <code>Value</code> and <code>MaxValue</code> clipped to <math>\geq</math> <code>Value</code>.</li> <li>• Control data for 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 <code>set_auto_laser_control</code> 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>													

<b>Ctrl Command</b>	<b>set_auto_laser_control</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>For <code>Mode</code> = 2, a functioning <i>iDRIVE</i> scan system must be attached to the first scan head connector. If this is not the case, then error code 2 is returned and <code>Ctrlset</code> to 0. If the <i>iDRIVE</i> scan system has reacted, then the to-be-transmitted data type for the first scan head connector and both axes are set to "actual velocity" by <code>control_command</code> (<code>Data</code> = <code>0506H</code>). As a result, <code>control_command</code> is unavailable for the first scan head connector as long as this <code>Mode</code> selection is in effect. For <code>Mode</code> = 0 or 1, <code>control_command</code> is available without any restriction.</li> <li>Encoder-speed-dependent laser control (<code>Mode</code> = 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. The target encoder speed is set by <code>set_encoder_speed</code>.</li> <li>With the +4 extension, encoder speeds are converted to galvanometer scanner units (<i>bits/ms</i>) by using the scaling factors from <code>set_fly_x</code> and <code>set_fly_y</code> or <code>set_fly_2d</code>. 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 <code>Mode</code> = 5). The current mark speed is used as reference speed. The <code>set_encoder_speed</code> command (which is used for <code>Mode</code> = 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. <code>set_fly_rot</code>, <code>set_fly_rot_pos</code>, <code>set_fly_x_pos</code> and <code>set_fly_y_pos</code> cannot be combined with the galvanometer scanner speed.</li> <li>If the values for <code>Ctrl</code> or <code>Mode</code> are invalid, then <code>set_auto_laser_control</code> is not executed (return value 3 or 4, <code>get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>The signal parameter selected with <code>Ctrl</code> is updated every 10 <math>\mu</math>s. However, for <code>Ctrl</code> = 5 (output period) it is always only effective after an already started laser signal period has elapsed.</li> <li>"Automatic Laser Control" should not be combined with the variable laser power of <code>set_multi_mcbsp_in</code>.</li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> for <code>Ctrl</code> = 1/2, the specified values for <code>Value</code> , <code>MinValue</code> and <code>MaxValue</code> are internally multiplied by 4. For <code>Ctrl</code> = 4/5, the parameter values for <code>Value</code> , <code>MinValue</code> and <code>MaxValue</code> are multiplied by 8. The allowed value ranges decrease accordingly.
Version info	Last change DLL 540, OUT 540: <code>Mode</code> = 6.
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>



<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( <b>Ctrl</b> , <b>Value</b> , <b>MinValue</b> , <b>MaxValue</b> )				
<b>Parameters</b>	<b>Ctrl</b>	The to-be-controlled signal parameter (see <a href="#">set_auto_laser_control</a> ). Allowed value range: [1...6].			
	<b>Value</b>	100% value. See <a href="#">set_auto_laser_control</a> .			
	<b>MinValue</b>	Lower range limit. See <a href="#">set_auto_laser_control</a> .			
	<b>MaxValue</b>	Upper range limit. See <a href="#">set_auto_laser_control</a> .			
<b>Result</b>	ErrorCode.				
	As an unsigned 32-bit value.				
	0	No error.			
	3	Invalid <b>Ctrl</b> value.			
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>set_auto_laser_params</b> usage, see <a href="#">Chapter 7.4.9 ""Automatic Laser Control""</a>, <a href="#">page 187</a>.</li> <li>The parameter's meaning is identical to that of <a href="#">set_auto_laser_control</a> (see also comments there). However, <b>set_auto_laser_params</b> can neither start nor terminate "Automatic Laser Control".</li> <li>If the value for <b>Ctrl</b> is invalid (0 or &gt;6), then <b>set_auto_laser_params</b> is not executed (<a href="#">get_last_error</a> return code <a href="#">RTC5_PARAM_ERROR</a>).</li> </ul>				
RTC4→RTC5	New command. <a href="#">RTC4 Compatibility Mode</a> : see <a href="#">set_auto_laser_control</a> .				
<b>Version info</b>	–				
<b>References</b>	<a href="#">set_auto_laser_control</a>				

<b>Delayed Short List Command</b>	<b>set_auto_laser_params_list</b>				
<b>Function</b>	Like <a href="#">set_auto_laser_params</a> , but a list command.				
<b>Call</b>	set_auto_laser_params_list( <b>Ctrl</b> , <b>Value</b> , <b>MinValue</b> , <b>MaxValue</b> )				
<b>Parameters</b>	<b>Ctrl</b>	Like <a href="#">set_auto_laser_params</a> .			
	<b>Value</b>	Like <a href="#">set_auto_laser_params</a> .			
	<b>MinValue</b>	Like <a href="#">set_auto_laser_params</a> .			
	<b>MaxValue</b>	Like <a href="#">set_auto_laser_params</a> .			
<b>Comments</b>	<ul style="list-style-type: none"> <li>If the value for <b>Ctrl</b> is invalid (0 or &gt;6), then <b>set_auto_laser_params_list</b> is replaced with a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC5_PARAM_ERROR</a>).</li> </ul>				
RTC4→RTC5	New command. <a href="#">RTC4 Compatibility Mode</a> : see <a href="#">set_auto_laser_control</a> .				
<b>Version info</b>	–				
<b>References</b>	<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>	<code>set_char_pointer( Char, Pos )</code>
<b>Parameters</b>	<p>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 <a href="#">load_char</a>:  <code>Char</code> = character set number <math>\times</math> 256 + ASCII number of the character (character sets are numbered 0 to 3).</p> <p>Pos      Absolute start address.            As an unsigned 32-bit value.            Allowed value range: [0...(2<sup>20</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>If <code>Char</code> &gt; 1023 and/or <code>Pos</code> &gt; (2<sup>20</sup>-1), then <b>set_char_pointer</b> is not executed (<a href="#">get_last_error</a> return code <code>RTC5_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 <a href="#">save_disk</a>/<a href="#">load_disk</a> 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 <a href="#">copy_dst_src</a> 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 <a href="#">get_input_pointer</a> before saving 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 <a href="#">save_disk</a>/<a href="#">load_disk</a>.</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 has been finalized by <a href="#">list_return</a> and does not contain <a href="#">set_end_of_list</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">load_char</a>



<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>	<code>set_char_table( 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 ordering applies as for the <b>load_text_table</b> command:            = 0...9: Digits for marking the time and date [0...9].            = 10...21: Months [January...December].            = 22...28: Days-of-the-week [Sunday...Saturday].            = 29: Blank character for marking serial numbers.            = 30...39: Digits for marking serial numbers [0...9].            = 40: Text for "a.m.".      = 41: Text for "p.m.".</p> <p>Pos      Absolute start address.            As an unsigned 32-bit value.            Allowed value range: [0...(2<sup>20</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_char_table</b> is synonymous with <b>set_text_table_pointer</b>.</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<b>set_text_table_pointer</b>

Ctrl Command	set_control_mode		
Function	Enables or disables the external control input for <a href="#">External Starts</a> and resets the counter for <a href="#">External Starts</a> to zero.		
Call	set_control_mode( <a href="#">Mode</a> )		
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:	With an <a href="#">External Stop</a> (/STOP, /STOP2, /Slave-STOP or <a href="#">simulate_ext_stop</a> ) causes explicit cancellation of the external start queue entries (/START, /START2, /Slave-START or <a href="#">simulate_ext_start</a> ).
		= 0:	No effect.
	Bit #2	= 1:	The track delay (defined by <a href="#">simulate_ext_start</a> , <a href="#">set_ext_start_delay</a> or <a href="#">set_ext_start_delay_list</a> ) that postpones execution of the start relative to the triggering input signal or <a href="#">simulate_ext_start</a> or <a href="#">simulate_ext_start_ctrl</a> (see <a href="#">Section "External Start", page 266</a> ) is deactivated.
		= 0:	No effect. To define and activate the track delay (for example, for Processing-on-the-fly applications), use <a href="#">simulate_ext_start</a> , <a href="#">set_ext_start_delay</a> or <a href="#">set_ext_start_delay_list</a> .
	Bit #3	= 1:	The external start input is <i>not</i> disabled by an external stop signal.
		= 0:	The external start input <i>is</i> disabled by an external stop signal.
	Bit #4		Disables <a href="#">simulate_ext_start_ctrl</a> .
	Bit #5		Reserved.
	Bit #6		Reserved.
	Bit #7		Reserved.
	Bit #8		Reserved.
	Bit #9	= 1:	Encoder resets of the 2 internal encoder counters occur by: <ul style="list-style-type: none"> <li>• An 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 <a href="#">Bit #2</a></li> </ul>
		= 0:	Encoder resets occur immediately with each initiating Processing-on-the-fly command.

Ctrl Command	set_control_mode	
Parameters (cont'd)	Bit #10	= 1: Track delay configured by <code>simulate_ext_start</code> , <code>set_ext_start_delay</code> or <code>set_ext_start_delay_list</code> is counted beginning with the most recent externally (but not with <code>execute_list_pos</code> etc.) triggered or simulated External Start. The interval between subsequent External Starts (in encoder pulses) is thus constant (see also Section "Regular (Periodic) External Starts", page 269). For <code>stop_execution</code> or an external stop signal, Bit #10 gets reset to "0". = 0: Track delay configured by <code>simulate_ext_start</code> , <code>set_ext_start_delay</code> or <code>set_ext_start_delay_list</code> is counted beginning with the point in time an External Start has been requested (that is, with the corresponding <code>simulate_ext_start</code> or <code>simulate_ext_start_ctrl</code> or external start signal). The interval between subsequent External Starts (in encoder pulses) can thus vary.
	Bit #12	Reserved.
	...	...
	Bit #31	Reserved.
Comments	<ul style="list-style-type: none"> <li>If execution is aborted by <code>stop_execution</code>, then Bit #0 and Bit #10 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 Bit #9 = 0, then there is generally a small (random) time offset (10 <math>\mu</math>s jitter) between the start signal at /START, /START2 and the actual execution start. If Bit #9 = 1, then this 10 <math>\mu</math>s jitter is not present because the encoder reset then occurs synchronously with the start signal.</li> <li>Bit #9 = 0 is useful, when several separate Processing-on-the-fly sessions are to be started within a list.</li> <li>See also Section "External Stop", page 265 and Section "External Start", page 266.</li> </ul>	
RTC4→RTC5	<p>Bit #8 cannot not be used to lock or unlock the upper 8 bits of the 16-bit digital output port for I/O commands. The RTC5 automatically reserves these bits for varioSCAN FLEX control by <code>move_to</code>. It is not possible to explicitly release these bits (release automatically occurs by <code>load_program_file</code>).</p> <p>Otherwise: unchanged functionality for the bits that are also used on the RTC4.</p>	
Version info	Last change DLL 543, OUT 543: Bit #4.	
References	<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>
Function	Like <a href="#">set_control_mode</a> , but a list command.
Call	<code>set_control_mode_list( Mode )</code>
Parameters	Mode      Like <a href="#">set_control_mode</a> .
Comments	<ul style="list-style-type: none"> <li>The counter for <a href="#">External Starts</a> is <i>not</i> reset by <a href="#">set_control_mode_list</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality for the bits that are also used on the RTC4.
Version info	Last change DLL 543, OUT 543: <a href="#">Bit #4</a> .
References	<a href="#">set_control_mode</a>

<b>Ctrl Command</b>	<b>set_default_pixel</b>
Function	Defines the pulse length default value for the default pixel that terminates <a href="#">Pixel Output Mode</a> .
Call	<code>set_default_pixel( PulseLength )</code>
Parameters	<p>PulseLength      Default pixel pulse length.  As an unsigned 32-bit value.  1 bit equals 1/64 <math>\mu</math>s.  Allowed value range: [0...(2<sup>32</sup>-1)].</p>
Comments	<ul style="list-style-type: none"> <li>In <a href="#">Pixel Output Mode</a>, 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 247</a>).</li> <li>When initializing (by <a href="#">load_program_file</a>), the PulseLength default value is set to 0.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_port_default</a> , <a href="#">set_default_pixel_list</a>

<b>Undelayed Short List Command</b>	<b>set_default_pixel_list</b>
Function	Like <a href="#">set_default_pixel</a> , but a list command.
Call	<code>set_default_pixel_list( PulseLength )</code>
Parameters	PulseLength      Like <a href="#">set_default_pixel</a> .
Comments	<ul style="list-style-type: none"> <li>See <a href="#">set_default_pixel</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_default_pixel</a> , <a href="#">set_port_default</a>

<b>Ctrl Command</b>	<b>set_defocus</b>
<b>Function</b>	Defines a focus shift for 3D vector outputs.
<b>Restriction</b>	If the <b>Option "3D"</b> has not been enabled or if no 3D correction table has been assigned (see <a href="#">select_cor_table</a> ), 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>	<p>Shift      Focus shift. In bits.  As a signed 32-bit value.  Allowed value range: [-32,768...+32,767].  Out-of-range values are clipped to the boundary values.</p>
<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, see <a href="#">get_galvo_controls</a>.</li> <li>If <b>set_defocus</b> 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. Length and duration of the jump depend on the changed z control value. The duration can be calculated by <a href="#">get_galvo_controls</a>.  If no list is currently <b>BUSY list execution status</b>, then the jump executes immediately, whereby no delay occurs at the next start.</li> <li>If the <b>INTERNAL-BUSY list execution status</b> is set, <b>set_defocus</b> is only executed with a delay (after <b>INTERNAL-BUSY list execution status</b> has been reset again).</li> <li><b>set_defocus</b> sets the <b>INTERNAL-BUSY list execution status</b> while the jump to the changed z output is executed.</li> <li>After vector-defined laser control is activated by <b>set_vector_control</b>(<code>Ctrl = 7</code>), the focus shift changes with parameterized <b>[*]mark[*] Command</b> or <b>Jump commands</b>, too.</li> </ul>
RTC4→RTC5	Basically unchanged functionality. But the RTC5 command avoids " <b>Hard jumps</b> ".
Version info	–
References	<a href="#">set_defocus_list</a> , <a href="#">set_offset_xyz</a> , <a href="#">set_defocus_offset</a> , <a href="#">get_galvo_controls</a>

<b>Variable List Command</b>	<b>set_defocus_list</b>
Function	Like <b>set_defocus</b> , but a list command.
Restriction	Like <b>set_defocus</b> .
Call	<code>set_defocus_list( Shift )</code>
Parameters	Shift      Like <b>set_defocus</b> .
Comments	<ul style="list-style-type: none"> <li>See <b>set_defocus</b>.</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 <math>\mu</math>s, even if <code>Shift = 0</code>.</li> <li>After the jump to the changed z output, a <b>Jump Delay</b> previously configured with <b>set_scanner_delays</b> is inserted. Depending on the dynamics of the z axis it may be reasonable to increase the <b>Jump Delay</b> before calling <b>set_defocus_list</b>.</li> </ul>
RTC4→RTC5	See <b>set_defocus</b> .
Version info	–
References	<b>set_defocus, long_delay</b>

<b>Ctrl Command</b>	<b>set_defocus_offset</b>
Function	Defines an offset in addition to the focus shift (see <b>set_defocus</b> ) for 3D vector outputs.
Restriction	Like <b>set_defocus</b> .
Call	<code>set_defocus_offset( Shift )</code>
Parameters	Shift      See <b>set_defocus</b> .
Comments	<ul style="list-style-type: none"> <li><b>set_defocus_offset</b> has the same effect as <b>set_defocus</b>.</li> <li><b>set_defocus_offset</b> enables setting a global defocus shift that is not overwritten by parameterized commands such as <b>para_mark_abs</b> (see <b>set_vector_control</b>(<code>Ctrl = 7</code>)).</li> <li>The <b>set_defocus_offset</b> shift, <ul style="list-style-type: none"> <li>works additively to <b>set_defocus</b>,</li> <li>cannot be used as a control parameter for vector-defined laser control,</li> <li>cannot be recorded via Signal 32 with <b>set_trigger/set_trigger4</b>,</li> <li>is taken into account by <b>get_z_distance</b>, <b>get_galvo_controls</b> and the back transformation (<b>transform</b>, <b>get_transform</b>).</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 540, OUT 540.
References	<b>set_defocus, set_defocus_offset_list</b>



<b>Variable List Command</b>	<b>set_defocus_offset_list</b>
Function	Like <a href="#">set_defocus_offset</a> , but a list command.
Restriction	Like <a href="#">set_defocus</a> .
Call	<code>set_defocus_offset_list( Shift )</code>
Parameters	Shift      Like <a href="#">set_defocus</a> .
Comments	<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 <a href="#">set_defocus_offset_list</a> 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 <math>\mu</math>s, even if Shift = 0.</li></ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 540, OUT 540.
References	<a href="#">set_defocus_offset</a> , <a href="#">set_defocus_list</a>



Ctrl Command	set_delay_mode		
Function	Turns the “Variable Polygon Delays” mode and the “Variable Jump Delays” mode on or off and sets some special scanner-delay related parameters as well as the 3D Z-Move mode.		
Call	set_delay_mode( VarPoly, DirectMove3D, EdgeLevel, MinJumpDelay, JumpLengthLimit )		
Parameters	Name	Allowed Values	Description
	VarPoly	> 0 = 0	Enables “Variable Polygon Delays” mode. Disables “Variable Polygon Delays” mode. Default setting.
	DirectMove3D	> 0 = 0	As an unsigned 32-bit value. This parameter effects only 3D-applications. The z output is changed directly (linearly) to its end value during a jump.
	EdgeLevel	0...(2 <sup>32</sup> –1). 1 bit equals 10 $\mu$ s.	The z output is changed to its end-value in such a way that the focus is kept in one plane during the entire jump. As an unsigned 32-bit value. This parameter defines a maximum “laser on” time for the corners of a Polyline. If the Polygon Delay is longer than or equal to this value (because the angle $\varphi$ is close to 180°, for instance), the laser is switched off (after a LaserOff Delay) and a new Polyline is started. This can be useful for preventing burn-in effects. The EdgeLevel value must be smaller than twice the set value for the Polygon Delay, otherwise it has no effect. See also <a href="#">Figure 44</a> . Note: To disable this feature, the EdgeLevel value must be set to (2 <sup>32</sup> –1) (default value).
	MinJumpDelay	0...(2 <sup>32</sup> –1). 1 bit equals 10 $\mu$ s.	As an unsigned 32-bit value. Minimum Jump Delay that cannot be undercut for jumps shorter than JumpLengthLimit (even for jump vectors of zero length, see <a href="#">Figure 40</a> ). For jumps longer than JumpLengthLimit, the MinJumpDelay has no relevance. To avoid anomalies in the range of MinJumpDelay, define a value for MinJumpDelay 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><b>Jump length limit.</b> In bits. JumpLengthLimit &gt; 0 enables the “<b>Variable Jump Delays</b>” mode. If the jump vector is <i>longer</i> than this value, then the fixed <b>Jump Delay</b> (see <a href="#">set_scanner_delays</a>) is inserted. For all shorter jump lengths, a linearly interpolated “<b>Variable Jump Delay</b>” (page 136) between <b>MinJumpDelay</b> and the <b>Jump Delay</b> is calculated and inserted, see <a href="#">Figure 40</a>. JumpLengthLimit = 0 disables the “<b>Variable Jump Delays</b>” mode.</p> <p>As an unsigned 32-bit value.</p>
RTC4→RTC5	<p>Unchanged functionality. In addition: extended value range. In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified value for <b>JumpLengthLimit</b> by 16. The allowed value range decreases accordingly.</p>
Version info	–
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→RTC5	<p>New command. <b>RTC4 Compatibility Mode:</b> see <a href="#">set_delay_mode</a>.</p>
Version info	–
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>initial_dsp_mode = set_dsp_mode( Mode )</code>
<b>Parameters</b>	Mode      Desired <b>DSP</b> mode. As an unsigned 32-bit value.
<b>Result</b>	<b>DSP</b> mode for which the board has been set during initialization. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_dsp_mode</b> is only needed when multiple RTC5 Boards with differing <b>DSP</b> versions are to be operated synchronously in a master/slave chain (see <a href="#">page 114</a>), and then only if matching of short list counts is not to be performed by <b>sync_slaves</b>.</li> <li>• Even if another mode has been already set by <b>set_dsp_mode</b>, <b>set_dsp_mode</b> always returns the <b>DSP</b> mode that the board has been set to during initialization. This return value (<code>initial_dsp_mode</code>) is identical to Byte #2 of <b>get_RTC_version</b> (<b>DSP</b> version number).</li> <li>• No mode can be set that is larger than <code>initial_dsp_mode</code> (<code>Mode</code> is clipped to <code>initial_dsp_mode</code>). Thus, faster boards can only be matched to slower boards, not the other way around.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">sync_slaves</a> , <a href="#">get_RTC_version</a>



Undelayed Short List Command	<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> .
Restriction	For older RTC5 Boards with DSP version numbers < 2 ( <b>get_RTC_version</b> Bit #16...Bit #23), <b>set_ellipse</b> is neither executed nor replaced by <b>list_nop</b> ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</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> <p>b      See a.</p> <p>Phi0     Beginning phase angle (the arc starting point position relative to the end point of half-axis a). In degrees.             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 (to-be-marked ellipse section). In degrees.             As a 64-bit IEEE floating point value.             A positive sign means "clockwise".             Allowed value range: [-2,880.0°...+2,880.0°] (<math>\pm 8</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 <b>Image</b> field by <b>mark_ellipse_abs</b> or <b>mark_ellipse_rel</b>. For descriptions of the individual parameters, see <b>Section "Ellipse Commands", page 126</b>.</li> <li>For <math>a &lt; 1</math> and/or <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>RTC5_PARAM_ERROR</b>).</li> </ul>
RTC4→RTC5	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>a</b> and <b>b</b> by 16. The allowed value range decreases accordingly.
Version info	Last change DLL 551, OUT 553: Allowed value range for <b>Phi</b> has been reduced (previously: [-3,600.0°...+3,600.0°] = $\pm 10$ full ellipses).
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 laser control.
<b>Call</b>	<code>set_encoder_speed( EncoderNo, Speed, Smooth )</code>
<b>Parameters</b>	<p><b>EncoderNo</b> 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 laser control in <code>Mode=5</code>.          Bits with higher significance are ignored.</p> <p><b>Speed</b> 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].          Out-of-range values are clipped to the boundary values.</p> <p><b>Smooth</b> 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 <math>\mu</math>s) 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 \leq 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>RTC5_PARAM_ERROR</code>).</li> <li>One encoder increment results in 4 counter pulses, see <a href="#">Section "Input Ports for External Encoder Signals", page 275</a>.</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 <math>\mu</math>s), 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 <math>\mu</math>s) are received.</li> <li>See also <a href="#">Chapter "Encoder-Speed-Dependent Laser Control", page 192</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_encoder_speed_ctrl</a> , <a href="#">set_auto_laser_control</a>



Ctrl Command	<code>set_encoder_speed_ctrl</code>
Function	Like <code>set_encoder_speed</code> , but a control command.
Call	<code>set_encoder_speed_ctrl( EncoderNo, Speed, Smooth )</code>
Parameters	EncoderNo      Like <code>set_encoder_speed</code> .
	Speed      Like <code>set_encoder_speed</code> .
	Smooth      Like <code>set_encoder_speed</code> .
Comments	<ul style="list-style-type: none"> <li>• <code>set_encoder_speed_ctrl</code> is not executed (<code>get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>), if:           <ul style="list-style-type: none"> <li>– Speed <math>\leq 0.0</math></li> <li>– Smooth <math>&lt; 0.0</math></li> </ul> </li> <li>• <code>set_encoder_speed_ctrl</code> is not executed (<code>get_last_error</code> return code <code>RTC5_BUSY</code>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</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> list execution status set)</li> <li>– the <b>INTERNAL-BUSY</b> list execution status is set</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<code>set_encoder_speed</code> , <code>set_auto_laser_control</code>

Normal List Command	<b>set_end_of_list</b>
Function	Ends execution of a list.
Call	<code>set_end_of_list()</code>
Comments	<ul style="list-style-type: none"> <li>If, during processing of a list, <code>set_end_of_list</code> is encountered and no automatic list change has been previously activated (see <a href="#">Chapter 6.4.6 "Automatic List Changing", page 99</a>), then list execution ends. The <a href="#">Signals for "Laser Active" Operation</a> are then switched off and a home jump, if defined by <a href="#">home_position</a> or <a href="#">home_position_xyz</a>, is executed (the <a href="#">INTERNAL-BUSY list execution status</a> 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 has been previously activated. The other list can also be "List 1" if "List 2" has not been configured (<code>Mem2 = 0</code>, see <a href="#">config_list</a>).</li> <li>Upon processing of a <code>set_end_of_list</code>, the <a href="#">USED list status</a> of the respective list (<a href="#">USED1</a> or <a href="#">USED2</a>) is always set and the <a href="#">BUSY list status</a> (<a href="#">BUSY1</a> or <a href="#">BUSY2</a>) of the list is reset, see also <a href="#">Chapter 6.4.3 "List Execution Status", page 97</a>. The <a href="#">BUSY list execution status</a>, on the other hand, is only reset if no automatic list change has been previously activated.</li> <li>An automatic list change of the input pointer never occurs during <i>loading</i> of <code>set_end_of_list</code> (in contrast to an automatic list change of the <i>output pointer</i> during <i>execution</i> 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 <a href="#">READY list status</a> (<a href="#">READY1</a> or <a href="#">READY2</a>) is set and <a href="#">LOAD list status</a> (<a href="#">LOAD1</a> or <a href="#">LOAD2</a>) is reset. Additionally, flushing of the buffered list input is triggered, see <a href="#">Chapter 6.4.1 "Loading Lists", page 94</a>.</li> <li><code>set_end_of_list</code> is ignored during loading and execution, that is, replaced with <a href="#">list_nop</a> if an indexed subroutine is currently being loaded or executed (<a href="#">get_last_error</a> return code <a href="#">RTC5_IGNORED</a>).</li> </ul>
RTC4→RTC5	Basically unchanged functionality. However: Additional <a href="#">USED list status</a> .
Version info	–
References	<a href="#">set_start_list</a>



<b>Ctrl Command</b>	<b>set_extstartpos</b>
<b>Function</b>	Defines the start address (within the range of "List 1" or "List 2") where execution should continue upon <b>External Starts</b> .
<b>Call</b>	<code>set_extstartpos( 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>20</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>By default, an <b>External Start</b> results in a continuation or start at the beginning of "List 1" (<code>Pos = 0</code>).</li> <li><code>Pos</code> must be within the range of "List 1" or "List 2". Otherwise, <code>Pos = 0</code> is set.</li> <li>The specified start address is used for <b>External Starts</b> 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 <code>Pos</code> before each <b>External Start</b>; <code>Pos</code> might therefore become set to 0 at a later point (and remain at this value) if the configuration has been correspondingly changed in the meantime (see <b>config_list</b>).</li> <li>See also <b>Section "External Start", page 266</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_extstartpos_list, set_control_mode</b>

<b>Undelayed Short List Command</b>	<b>set_extstartpos_list</b>
<b>Function</b>	Like <b>set_extstartpos</b> , but a list command.
<b>Call</b>	<code>set_extstartpos_list( Pos )</code>
<b>Parameters</b>	Pos      Like <b>set_extstartpos</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>This list command can be used within a list, to "link" it to the list that follows.</li> <li>See <b>set_extstartpos</b>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range.
Version info	–
References	<b>set_extstartpos</b>

<b>Ctrl Command</b>	<code>set_ext_start_delay</code>
<b>Function</b>	Sets a track delay for <b>External Starts</b> , 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> .
<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:                  = 0:    Encoder counter "Encoder0".                  = 1:    Encoder counter "Encoder1".</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>External Starts</b>, see <a href="#">Section "External Start", page 266</a>.</li> <li>The track delay is specified in <i>relative</i> counting units of the selected encoder counter (the RTC5 encoder counter is triggered by an external or simulated encoder signal, see <a href="#">page 274</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 and is always positive with simulated encoders).</li> <li>For <code>Delay</code> = 0, 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> list execution status or <b>INTERNAL-BUSY</b> list execution 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 an <b>External Stop</b> and by <b>stop_execution</b>. They can also be deactivated with <b>set_control_mode/set_control_mode_list</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</code> &gt; 1, then <b>set_ext_start_delay</b> is ignored (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> </ul>



<b>Ctrl Command</b>	<b>set_ext_start_delay</b>
RTC4→RTC5	<p>Unchanged functionality. In addition: increased value range.</p> <p>The RTC5 allows this command to be used not only together with an external encoder, but also with an encoder simulation (see <a href="#">Section "Encoder Simulation", page 275</a>) started by <a href="#">simulate_encoder</a>.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC5 multiplies the specified value for <code>Delay</code> by 16. The allowed value range decreases accordingly.</p>
Version info	–
References	<a href="#">simulate_ext_start</a> , <a href="#">set_ext_start_delay_list</a> , <a href="#">set_extstartpos</a> , <a href="#">set_extstartpos_list</a> , <a href="#">set_control_mode</a> , <a href="#">set_control_mode_list</a>

<b>Normal List Command</b>	<b>set_ext_start_delay_list</b>
Function	Like <a href="#">set_ext_start_delay</a> , but a list command.
Call	<code>set_ext_start_delay_list( Delay, EncoderNo )</code>
Parameters	<p>Delay      Like <a href="#">set_ext_start_delay</a>.</p> <p>EncoderNo   Like <a href="#">set_ext_start_delay</a>.</p>
Comments	<ul style="list-style-type: none"> <li>If <code>EncoderNo &gt; 1</code>, then <a href="#">set_ext_start_delay_list</a> is replaced by a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</li> </ul>
RTC4→RTC5	See <a href="#">set_ext_start_delay</a> .
Version info	–
References	<a href="#">set_ext_start_delay</a> .

<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.      1 bit equals 1/64 <math>\mu</math>s.      Allowed value range: [0...+(2<sup>26</sup>-1)].      Out-of-range values are clipped to the boundary values.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The time base for the FirstPulseKiller signal is 64 MHz.      1 bit equals 1/64 <math>\mu</math>s.</li> <li>The signal level is defined by <a href="#">set_laser_control</a>.</li> <li>For YAG Mode 2, the Q-Switch delay is also correspondingly altered, see <a href="#">Section "Differences Between the YAG Modes", page 180</a>.</li> <li>In <a href="#">CO<sub>2</sub> Mode</a>, <a href="#">set_firstpulse_killer</a> has no effect.</li> <li>The laser control mode is set by <a href="#">set_laser_mode</a>, see <a href="#">Chapter 7.4 "Laser Control", page 173</a>.</li> <li>Q-Switch pulse length and Q-Switch period can be 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→RTC5	Basically unchanged functionality. In addition: increased value range. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified value by 8. The allowed value range decreases accordingly.
Version info	–
References	<a href="#">set_firstpulse_killer_list</a> , <a href="#">set_laser_mode</a> , <a href="#">set_laser_timing</a>

<b>Undelayed Short List Command</b>	<b>set_firstpulse_killer_list</b>
<b>Function</b>	Like <a href="#">set_firstpulse_killer</a> , but a list command.
<b>Call</b>	<code>set_firstpulse_killer_list( Length )</code>
<b>Parameters</b>	Length      Like <a href="#">set_firstpulse_killer</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">set_firstpulse_killer</a>.</li> </ul>
RTC4→RTC5	See <a href="#">set_firstpulse_killer</a>
Version info	–
References	<a href="#">set_firstpulse_killer</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-motion in 2 dimensions (based on the encoder values transferred to the RTC5 by encoder counters "Encoder0" and "Encoder1"). Sets the corresponding scaling factors.				
<b>Restriction</b>	If the <b>Option Processing-on-the-fly</b> 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>	<b>set_fly_2d( ScaleX, ScaleY )</b>				
<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 <b>Vector command</b> or "<b>Arc</b>" command (without "set_fly_2d" 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 <b>External Start</b> 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→RTC5	<p>New command.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC5 multiplies the specified values for <code>ScaleX</code> and <code>ScaleY</code> by 16. The allowed value range decreases accordingly.</p>
Version info	–
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>

Undelayed Short List Command	<code>set_fly_limits</code>
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><code>Xmin</code> Range boundary.  As a signed 32-bit value.  Allowed value range: <math>[-524,288 \dots +524,287]</math>.  Out-of-range values are clipped to the boundary values.</p> <p><code>Xmax</code> Like <code>Xmin</code> (analogously).</p> <p><code>Ymin</code> Like <code>Xmin</code> (analogously).</p> <p><code>Ymax</code> Like <code>Xmin</code> (analogously).</p>
Comments	<ul style="list-style-type: none"> <li>For <code>set_fly_limits</code> usage, see <a href="#">Section "Customer-Defined Monitoring Area", page 241</a>.</li> <li>During initialization (with <a href="#">load_program_file</a>), the boundary limits get set to the following default values: <code>Xmin = Ymin = -524,288</code> and <code>Xmax = Ymax = +524,287</code>.</li> <li>Boundary limits specified using the parameter value 0 also get set to the above-mentioned default values.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>	<p><code>Zmin</code> Range boundary.            As a signed 32-bit value.            Allowed value range: [-32,768...+32,767].            Out-of-range values are clipped to the boundary values.</p> <p><code>Zmax</code> Like <code>Zmin</code> (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <code>set_fly_limits_z</code> usage, see also <a href="#">Chapter 8.6.9 "Monitoring Processing-on-the-fly Corrections", page 240</a>.</li> <li>During initialization (with <code>load_program_file</code>), the boundary limits get set to the following default values: <code>Zmin</code> = -32,768 and <code>Zmax</code> = +32,767.</li> <li>Boundary limits specified using the parameter value 0 also get set to the above-mentioned default values.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<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 motion (based on angular position values transferred to the RTC5 by encoder counter "Encoder0") and sets the corresponding <code>Resolution</code> parameter.
<b>Restriction</b>	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <code>set_fly_rot</code> terminates the Processing-on-the-fly process (even though it could not have been activated).
<b>Call</b>	<code>set_fly_rot( Resolution )</code>
<b>Parameters</b>	<code>Resolution</code> 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><code>Resolution</code> 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 <code>Resolution</code> parameter, see <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>.</li> <li>Before executing <code>set_fly_rot</code>, you should define the rotation center for Processing-on-the-fly correction by <code>set_rot_center</code> or <code>set_rot_center_list</code>. Otherwise, the default value (0 0) is applied.</li> <li>If unallowed parameter values are supplied (for example, for <code>Resolution = 0</code>), then <code>set_fly_rot</code> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <code>set_fly_rot</code> (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 <a href="#">Vector command</a> or <a href="#">"Arc" command</a> (without "set_fly_rot" 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 Corrections", page 236</a>.</li> <li>By <code>set_control_mode( Bit #9 )</code>, it can be set in advance when the encoder counter "Encoder0" is reset by <code>set_fly_rot</code>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
References	<a href="#">set_rot_center</a> , <a href="#">set_rot_center_list</a> , <a href="#">fly_return</a> , <a href="#">get_encoder</a> , <a href="#">set_fly_rot_pos</a>

Normal List Command	<code>set_fly_rot_pos</code>
Function	Activates Processing-on-the-fly correction for compensation of workpiece or scan system rotary motion (based on angular position values transferred to the RTC5 by the <a href="#">McBSP interface</a> ). It thereby sets the corresponding <code>Resolution</code> parameter.
Restriction	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <code>set_fly_rot_pos</code> terminates the Processing-on-the-fly process (even though it could not have been activated).
Call	<code>set_fly_rot_pos( Resolution )</code>
Parameters	<code>Resolution</code> Number of steps per revolution. As a 64-bit IEEE floating point value. Allowed value range: $ Resolution  > 100.0$ .
Comments	<ul style="list-style-type: none"> <li><code>Resolution</code> 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 <code>Resolution</code> parameter, see <a href="#">Chapter 8.6 "Processing-on-the-fly", page 227</a>.</li> <li>Before calling <code>set_fly_rot_pos</code>, you should define the rotation center for Processing-on-the-fly correction by <a href="#">set_rot_center</a> or <a href="#">set_rot_center_list</a>. Otherwise, the default value (0 0) is applied.</li> <li>If unallowed parameter values are supplied (for example, for <code>Resolution = 0</code>), then <code>set_fly_rot_pos</code> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <code>set_fly_rot_pos</code> (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 <a href="#">Vector command</a> or <a href="#">"Arc" command</a> (without "<code>set_fly_rot_pos</code>" Processing-on-the-fly correction).</li> <li>The various Processing-on-the-fly corrections cannot be arbitrarily combined (see the <a href="#">page 227</a>).</li> <li>For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Corrections", page 236</a>.</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 215</a>.</li> <li>The <a href="#">McBSP interface</a> ignores the first FrameSync signal after a <a href="#">load_program_file</a> or <a href="#">mcbsp_init</a>. That is, data provided is not transmitted, see <a href="#">page 73</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
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 <b>Tracking Error Compensation of Encoder Values for Processing-on-the-fly Applications</b> .
<b>Call</b>	<code>set_fly_tracking_error( TrackingErrorX, TrackingErrorY )</code>
<b>Parameters</b>	<p>TrackingErrorX    <b>Tracking error</b> for the x axis. In units of <math>10 \mu\text{s}</math>.            As a signed 32-bit value.            Allowed value range: [0...65,535]. Excessive bits are ignored.</p> <p>TrackingErrorY    <b>Tracking error</b> for the y axis.            Otherwise, like TrackingErrorX.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>Tracking error</b> compensation is a first-approximation correction:  <math display="block">\text{EC} = \text{E} + (\text{E} - \text{EP}) * \text{TrackingError},</math>           whereby E is the current encoder value, EP is that of the previous cycle and EC is the compensated encoder value (used for Processing-on-the-fly correction).</li> <li>• To avoid step-like galvanometer scanner motions, the used encoders should have a sufficiently high resolution (counts per <math>10 \mu\text{s}</math> cycle).</li> <li>• If <math>\text{TrackingErrorX, Y} = 0</math> (initialization value), then compensation is switched off.</li> <li>• <b>store_encoder</b>, <b>read_encoder</b> and <b>get_encoder</b> still use the original, uncorrected encoder values.</li> <li>• This compensation is practical only for linear motions (<a href="#">Chapter 8.6.2 "Compensation of Linear Motions", page 228</a>), not for rotary motions (<a href="#">Chapter 8.6.3 "Compensating Rotary Motions", page 232</a>). For the latter, <math>\text{TrackingErrorX, Y}</math> should be set to 0.</li> <li>• If the encoders get reset by the start trigger (<b>set_control_mode</b>(Bit #9 = 1)), then the output value of the first <math>10 \mu\text{s}</math> cycle might be incorrect to an unforeseen extent. Resets by the <b>set_fly_x</b> and <b>set_fly_y</b> commands automatically wait internally for an empty cycle.</li> <li>• This compensation only functions for Processing-on-the-fly applications with encoders, but not with position signals by the <b>McBSP interface</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	–

Normal List Command	<code>set_fly_x</code>
Function	Activates Processing-on-the-fly correction for compensation of a linear workpiece-motion in the x direction (based on position values transferred to the RTC5 by encoder counter "Encoder0") and sets the corresponding scaling factor.
Restriction	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <code>set_fly_x</code> terminates the Processing-on-the-fly process (even though it could not have been activated).
Call	<code>set_fly_x( ScaleX )</code>
Parameters	<p>ScaleX      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>.</p>
Comments	<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 <code>set_fly_x</code> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <code>set_fly_x</code> (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 <a href="#">Vector command</a> or <a href="#">"Arc" command</a> (without "set_fly_x" Processing-on-the-fly correction).</li> <li>The various Processing-on-the-fly corrections cannot be arbitrarily combined (see the <a href="#">page 227</a>).</li> <li>For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Corrections", page 236</a>.</li> <li>By <code>set_control_mode( Bit #9 )</code>, it can be set in advance when the encoder counter "Encoder0" is reset by <code>set_fly_x</code>.</li> <li>You can also switch on <code>set_fly_x/set_fly_y</code> Processing-on-the-fly correction by <a href="#">activate_fly_xy/activate_fly_xy_encoder</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified value for ScaleX by 16. The allowed value range decreases accordingly.
Version info	–
References	<a href="#">fly_return</a> , <a href="#">set_fly_y</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>

Normal List Command	<b>set_fly_x_pos</b>
Function	Activates Processing-on-the-fly correction for compensation of a linear workpiece or scan system motion in the x direction (based on position values transferred to the RTC5 by the <a href="#">McBSP interface</a> ); thereby sets the corresponding scaling factor.
Restriction	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).
Call	<code>set_fly_x_pos( ScaleX )</code>
Parameters	ScaleX     Scaling factor for the x direction in <i>(RTC5)bits/(McBSP)bit</i> . As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleX}  \leq 16000.0$ .
Comments	<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 <a href="#">Vector command</a> or <a href="#">"Arc" command</a> (without "set_fly_x_pos" Processing-on-the-fly correction).</li> <li>The various Processing-on-the-fly corrections cannot be arbitrarily combined (see the <a href="#">page 227</a>).</li> <li>For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Corrections", 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 215</a>.</li> <li>The <a href="#">McBSP interface</a> ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>. That is, data provided is not transmitted, see <a href="#">page 73</a>.</li> </ul>
RTC4→RTC5	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified value for ScaleX by 16. The allowed value range decreases accordingly.
Version info	–
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>

Normal List Command	<code>set_fly_y</code>
Function	Activates Processing-on-the-fly correction for compensation of a linear workpiece-motion in the y direction (based on position values transferred to the RTC5 by encoder counter "Encoder1") and sets the corresponding scaling factor.
Restriction	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <code>set_fly_y</code> terminates the Processing-on-the-fly process (even though it could not have been activated).
Call	<code>set_fly_y( ScaleY )</code>
Parameters	<code>ScaleY</code> Scaling factor for the y direction (encoder counter "Encoder1"). In bits/count. As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleY}  \leq 16000.0$ .
Comments	<ul style="list-style-type: none"> <li><code>ScaleY</code> 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 <code>ScaleY = 0</code>), then <code>set_fly_y</code> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <code>set_fly_y</code> (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 <a href="#">Vector command</a> or <a href="#">"Arc" command</a> (without "set_fly_y" Processing-on-the-fly correction).</li> <li>The various Processing-on-the-fly corrections cannot be arbitrarily combined (see the <a href="#">page 227</a>).</li> <li>For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Corrections", page 236</a>.</li> <li>By <code>set_control_mode( Bit #9 )</code>, it can be set in advance when the encoder counter "Encoder0" is reset by <code>set_fly_y</code>.</li> <li>You can also switch on <code>set_fly_x</code>/<code>set_fly_y</code> Processing-on-the-fly correction by <a href="#">activate_fly_xy</a>/<a href="#">activate_fly_xy_encoder</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range.  In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified value for <code>ScaleY</code> by 16. The allowed value range decreases accordingly.
Version info	–
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>

Normal List Command	<b>set_fly_y_pos</b>
Function	Activates Processing-on-the-fly correction for compensation of a linear workpiece or scan system motion in the y direction (based on position values transferred to the RTC5 by the <a href="#">McBSP interface</a> ); thereby sets the corresponding scaling factor.
Restriction	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).
Call	<code>set_fly_y_pos( ScaleY )</code>
Parameters	ScaleY      Scaling factor for the y direction in (RTC5) <i>bits</i> /(McBSP) <i>bit</i> . As a 64-bit IEEE floating point value. Allowed value range: $1/256 \leq  \text{ScaleY}  \leq 16000.0$ .
Comments	<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 <a href="#">Vector command</a> or <a href="#">"Arc" command</a> (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="#">page 227</a>).</li> <li>For deactivating Processing-on-the-fly correction, see <a href="#">Chapter 8.6.5 "Deactivating Processing-on-the-fly Corrections", 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 215</a>.</li> <li>The <a href="#">McBSP interface</a> ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>. That is, data provided is not transmitted, see <a href="#">page 73</a>.</li> </ul>
RTC4→RTC5	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified value for ScaleY by 16. The allowed value range decreases accordingly.
Version info	–
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>

Normal List Command	<code>set_fly_z</code>				
Function	Activates Processing-on-the-fly correction for compensation of a linear workpiece-motion in the z direction (based on position values transferred to the RTC5 by the specified encoder counter) and sets the corresponding scaling factor.				
Restriction	If the <a href="#">Option Processing-on-the-fly</a> is not enabled, then <code>set_fly_z</code> terminates the Processing-on-the-fly process (even though it could not have been activated).				
Call	<code>set_fly_z( ScaleZ, EncoderNo )</code>				
Parameters	<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".				
Comments	<ul style="list-style-type: none"> <li><code>ScaleZ</code> 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 <code>ScaleZ</code> parameter values are supplied (for example, for <code>ScaleZ = 0</code>), then <code>set_fly_z</code> does not activate a Processing-on-the-fly correction or deactivates a Processing-on-the-fly correction previously activated by <code>set_fly_z</code> (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 <a href="#">Vector command</a> or <a href="#">"Arc" command</a> (without "set_fly_z" Processing-on-the-fly correction).</li> <li>For deactivating Processing-on-the-fly correction, see <a href="#">Section "Notes on Usage", page 243</a>.</li> <li>By <a href="#">set_control_mode( Bit #9 )</a>, it can be set in advance when the encoder counter "Encoder0" or "Encoder1" is reset by <code>set_fly_z</code>.</li> <li>If <code>EncoderNo &gt; 1</code>, <code>set_fly_z</code> is replaced by a <a href="#">list_nop (get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</li> </ul>				
RTC4→RTC5	<p>New command.</p> <p>In <a href="#">RTC4 Compatibility Mode</a>, the RTC5 multiplies the specified value for <code>ScaleZ</code> by 16. The allowed value range decreases accordingly.</p>				
Version info	–				
References	<a href="#">fly_return_z</a>				



<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>	<p>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.</p> <p>Value      Desired variable value.            As an unsigned 32-bit value.            Allowed value range: [0...(2<sup>32</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">Chapter 6.9.1 "Free Variables", page 123.</a></li> </ul>
RTC4→RTC5	New command.
Version info	Last change DLL 539, OUT 539: value range of parameter No increased to [0...7].
References	<a href="#">set_free_variable_list</a> , <a href="#">get_free_variable</a> , <a href="#">set_trigger</a>

<b>Undelayed Short List Command</b>	<b>set_free_variable_list</b>
<b>Function</b>	Like <a href="#">set_free_variable</a> , but a list command.
<b>Call</b>	<code>set_free_variable_list( No, Value )</code>
<b>Parameters</b>	<p>No      Like <a href="#">set_free_variable</a>.</p> <p>Value      Like <a href="#">set_free_variable</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">set_free_variable</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_free_variable</a>

<b>Ctrl Command</b>	<b>set_hi</b>
<b>Function</b>	Defines gain values 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    x gain value.                     As a 64-bit IEEE floating point value.                     Allowed value range: [0.01...100].</p> <p>GalvoGainY    Like GalvoGainX (analogously).</p> <p>GalvoOffsetX   x offset value. In bits.                     As a signed 32-bit value.                     Allowed value range: [-524,288...+524,287].</p> <p>GalvoOffsetY   Like GalvoOffsetX (analogously).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>set_hi</b> usage, see <a href="#">Section "Customer-Specific Calibration", page 254</a>.</li> <li>The specified gain values 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 values 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, if: <ul style="list-style-type: none"> <li>A parameter value is invalid  <code>(get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>)</li> <li>the <b>BUSY</b> list execution status is set  <code>(get_last_error</code> return code <code>RTC5_BUSY</code>)</li> <li>the <b>INTERNAL-BUSY</b> list execution status is set  <code>(get_last_error</code> return code <code>RTC5_BUSY</code>)</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> list execution status set)</li> </ul> </li> <li>For <code>RTC5_PARAM_ERROR</code>, the <b>BUSY</b> list execution status is not checked. Therefore the return codes <code>RTC5_BUSY</code> and <code>RTC5_PARAM_ERROR</code> 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→RTC5	New command.
Version info	–
References	<a href="#">auto_cal</a> , <a href="#">get_hi_pos</a> , <a href="#">write_hi_pos</a>



<b>Ctrl Command</b>	<b>set_input_pointer</b>
<b>Function</b>	Opens the list memory area for writing with list commands and sets the input pointer to the specified <i>absolute</i> address in list memory ("List 1" or "List 2").
<b>Call</b>	<code>set_input_pointer( Pos )</code>
<b>Parameters</b>	<p>Pos      Position (absolute memory address) of the input pointer.  <math>[0\dots(2^{20}-1)]</math>.      As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The next list command is stored at the specified address and all further list commands at the subsequent addresses in the selected list.</li> <li><code>set_input_pointer</code> performs basically like <code>set_start_list_pos</code> (see the comments there). But <code>set_input_pointer</code> sets the input pointer based on a specified <i>absolute</i> memory address, whereas <code>set_start_list_pos</code> uses a specified list number and a <i>relative</i> memory address.</li> <li>For <code>Pos <math>\geq</math> Mem1 + Mem2</code> (see <code>config_list</code>), <code>Pos</code> is set to 0.</li> </ul>
RTC4→RTC5	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<a href="#">set_start_list_pos</a> , <a href="#">get_input_pointer</a>

<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 <code>MaskSet</code> , if the current <code>IOvalue</code> at the 16-bit digital <i>input</i> port on the EXTENSION 1 socket connector meets the following condition: $((IOvalue \text{ AND } Mask1) = Mask1) \text{ AND } (((\text{not } IOvalue) \text{ AND } Mask0) = Mask0)$ (= if the bits specified in <code>Mask1</code> are 1 and the bits specified in <code>Mask0</code> are 0).
Call	<code>set_io_cond_list( Mask1, Mask0, MaskSet )</code>
Parameters	<p><code>Mask1</code> 16-bit mask.  As an unsigned 32-bit value.  Only the lower 16 bits are evaluated.</p> <p><code>Mask0</code> Like <code>Mask1</code>.</p> <p><code>MaskSet</code> Like <code>Mask1</code>.</p>
Comments	<ul style="list-style-type: none"> <li>• <code>set_io_cond_list</code> sets only those bits of the digital output port that are set in the parameter <code>MaskSet</code> and leaves the other bits unchanged.</li> <li>• See also <a href="#">Section "16-Bit Digital Input Port and 16-Bit Digital Output Port", page 65</a> and <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 271</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 to 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→RTC5	Unchanged functionality.
Version info	–
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 <b>Jump Mode</b> for 2D jumps and sets the related parameters.
<b>Prerequisite</b>	Enabling is only possible if a jump-tuning-equipped <b>intelliSCAN</b> (with scan system firmware version $\geq$ 2078) 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 = <b>set_jump_mode( Flag, Length, VA1, VA2, VB1, VB2, JA1, JA2, JB1, JB2 )</b>
<b>Parameters</b>	<p><b>Flag</b>      Switching flag.            As a signed 32-bit value.            Allowed values:            = -1: <b>Jump Mode</b> is disabled. Afterward, switching the <b>Flag</b> by <b>set_jump_mode_list</b> is <i>no longer</i> possible.            = 0: <b>Jump Mode</b> is enabled but deactivated.            Afterwards it is also possible to switch the flag by <b>set_jump_mode_list</b>.            = 1: <b>Jump Mode</b> is enabled and activated.            Afterwards it is also possible to switch the flag by <b>set_jump_mode_list</b>.</p> <p><b>Length</b>      Limit of the jump length (per axis) under which 2D jumps – even with activated <b>Jump Mode</b> – are performed in vector mode.            As an unsigned 32-bit value.</p> <p><b>VA1</b>      Numbers of the tunings that should be used for jump execution in <b>Jump Mode</b>:  <b>VA2</b>      V: <b>Vector tuning</b> that is set at the end of a 2D jump.  <b>VB1</b>      J: <b>Jump tuning</b> that is set at the beginning of a 2D jump.  <b>VB2</b>      A      First scan head connector.  <b>JA1</b>      B      Second scan head connector.  <b>JA2</b>      1: x axis (<b>STATUS</b> channel, <b>Galvanometer scanner 2</b>).  <b>JA2</b>      2: y axis (<b>STATUS1</b> channel, <b>Galvanometer scanner 1</b>).  <b>JB1</b>      Allowed values:  <b>JB2</b>      = -1: Tuning is neither checked nor used.            = 0...3: After passing a check, tuning is used for <b>Jump Mode</b>.            The allowed value range also depends on the number of tunings with which the attached scan system is equipped.            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 (<b>Jump Mode</b> deactivated, vector mode activated).</p> <p>1      No error: Flag successfully switched to 1 (<b>Jump Mode</b> activated, vector mode deactivated).</p> <p>-1     Flag successfully switched to -1 (<b>Jump Mode</b> deactivated and disabled).</p> <p>-2     Busy error: board <b>BUSY</b> list execution status or <b>INTERNAL-BUSY</b> list execution status (<b>get_last_error</b> return code <b>RTC5_BUSY</b>).</p> <p>-3     Board not responding: no program loaded or PCI error (<b>get_last_error</b> return code <b>RTC5_TIMEOUT</b>).</p> <p>-4     Access error: board reserved for another user program (<b>get_last_error</b> return code <b>RTC5_ACCESS_DENIED</b>).</p> <p>&gt; 1    Did not pass the check (see also notes). Flag is set to -1 (<b>Jump Mode</b> 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 (<b>Galvanometer scanner 2</b>) not responding or no <b>intelliSCAN</b> (with scan system firmware version <math>\geq</math> 2078) attached.</li> <li>=2: y axis (<b>Galvanometer scanner 1</b>) not responding or no <b>intelliSCAN</b> (with scan system firmware version <math>\geq</math> 2078) 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	set_jump_mode
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 202</a>.</li> <li>A check (see also <a href="#">Section "Requirements and Activation", page 203</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> list execution status or <b>INTERNAL-BUSY</b> list execution status, because meanwhile the data type to-be-returned by the scan system changes and "Automatic Laser Control" is deactivated (both get restored at the end of the command). Depending on results of the check, different error codes are returned (see above). In case of error, <code>Flag</code> gets set to -1 (<b>Jump Mode</b> deactivated and disabled). 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 (<b>Jump Mode</b> activated, vector mode deactivated) and <code>Flag</code> = 0 (<b>Jump Mode</b> 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 – <b>Vector tuning</b>, for example, is not needed in pure drilling applications) or if, after switching to <b>Jump tuning</b>, it is not desirable to return to <b>Vector tuning</b>. As a result, such tunings are neither checked nor switched on. If the <b>Option "Second Scan Head Control"</b> is not enabled, then the tuning numbers for the second scan head connector (B) is automatically set to -1 (even if others were supplied). If all tuning numbers are -1, then <code>Flag</code> is set to -1 (return value -1).</li> <li>Even after successful activation of <b>Jump Mode</b> (<code>Flag</code> = 1), the first servo switching only occurs after the first subsequent 2D jump (see <a href="#">Section "Functional Principle", page 202</a>). If the currently set tuning then does not match the <b>Jump tuning</b> or <b>Vector tuning</b> 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 <code>control_command</code> before the first 2D jump to set the tuning to one that has been supplied by <code>set_jump_mode</code>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_jump_mode_list</a> , <a href="#">load_jump_table_offset</a> , <a href="#">set_jump_table</a>



Normal List Command	<b>set_jump_mode_list</b>
Function	Activates or deactivates and disables <b>Jump Mode</b> for 2D jumps.
Prerequisite	Like <b>set_jump_mode</b> .
Call	<code>set_jump_mode_list( Flag )</code>
Parameters	Flag      See <b>set_jump_mode</b> .
Notes	<ul style="list-style-type: none"> <li>For <b>set_jump_mode_list</b> usage, see <a href="#">Chapter 8.1.5 "Jump Mode", page 202</a>.</li> <li><b>set_jump_mode_list</b> functions like the control command <b>set_jump_mode</b> (see notes there) but, as a list command, has the following differences: <ul style="list-style-type: none"> <li>No tunings or jump length limit can be specified by <b>set_jump_mode_list</b>.</li> <li><b>set_jump_mode_list</b> does not perform a check. <code>Flag</code> must have previously been successfully set to 0 or 1 by <b>set_jump_mode</b>.</li> <li>Though <b>Jump Mode</b> can be deactivated and disabled by setting <code>Flag</code> to -1 by <b>set_jump_mode</b> or <b>set_jump_mode_list</b>, it can only be reactivated again by <b>set_jump_mode</b> (if the check is successful). If <code>Flag</code> has been -1 prior to calling <b>set_jump_mode_list</b>, then <b>set_jump_mode_list</b> has no effect.</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
Reference	<b>set_jump_mode</b>



<b>Undelayed Short List Command</b>	<b>set_jump_speed</b>
Function	Defines the jump speed for <b>Vector commands</b> .
Call	<code>set_jump_speed( Speed )</code>
Parameters	Speed      Jump speed. In <i>bits/ms</i> . As a 64-bit IEEE floating point value. Allowed value range: [1.6...800000.0]. Out-of-range values are clipped to the boundary values.
Comments	<ul style="list-style-type: none"> <li>By default a jump speed of 10000 <i>bits/ms</i> is preset.</li> <li>The specified jump speed is used for all <b>Jump commands</b> 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 [<i>bits/ms</i>] 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 <b>get_table_para</b> or <b>get_head_para</b>.</p> <ul style="list-style-type: none"> <li><b>set_jump_speed</b> is also available as control command <b>set_jump_speed_ctrl</b>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified Speed value by 16. The allowed value range decreases accordingly.
Version info	–
References	<b>jump_abs, jump_rel, set_mark_speed, set_jump_speed_ctrl</b>

<b>Ctrl Command</b>	<b>set_jump_speed_ctrl</b>
Function	Like <b>set_jump_speed</b> , but a control command.
Call	<code>set_jump_speed_ctrl( Speed )</code>
Parameters	Speed      Like <b>set_jump_speed</b> .
Comments	<ul style="list-style-type: none"> <li><b>set_jump_speed_ctrl</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY list execution status</b> is set</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 list execution status</b> set)</li> <li>the <b>INTERNAL-BUSY list execution status</b> is set</li> </ul> </li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> : like <b>set_jump_speed</b> .
Version info	–
References	<b>set_jump_speed, set_mark_speed, set_mark_speed_ctrl</b>



<b>Ctrl Command</b>	<b>set_jump_table</b>
<b>Function</b>	Reads the <b>Jump Delay</b> table with 1024 unsigned 16-bit values stored at the supplied PC address and loads them onto the board as the <b>Jump Delay</b> table (see notes for <b>get_jump_table</b> ).
<b>Call</b>	ErrorCode = <b>set_jump_table( Addr )</b>
<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> <p>0      No error.</p> <p>1      Busy error: board <b>BUSY</b> or <b>INTERNAL-BUSY</b> list execution status (<b>get_last_error</b> return code <b>RTC5_BUSY</b>).</p> <p>4      Verify error: <b>DSP</b> memory error.</p> <p>11     RTC5 board driver error.</p>
<b>Notes</b>	<ul style="list-style-type: none"> <li>• <b>set_jump_table</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status 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 <b>set_wait</b> (<b>PAUSED</b> list execution status set)</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
Reference	<b>get_jump_table</b> , <b>load_jump_table_offset</b>



<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 <a href="#">Laser Mode 4</a> or <a href="#">Laser Mode 6</a>).  <b>(LSB)</b>   The setting only affects those LASER1 or LASER2 "laser active" modulation pulses in <a href="#">CO<sub>2</sub> Mode</a> 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 59</a> and <a href="#">Figure 60</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="#">Section "Pulse Completion", page 175</a>.</p> <p>Bit #1    Phase shift of the laser control signals (does not concern <a href="#">Laser Mode 4</a> or <a href="#">Laser Mode 6</a>).  = 0:   No phase shift.  = 1:   <a href="#">CO<sub>2</sub> Mode</a>: The LASER1 signal is exchanged with the LASER2 signal.  YAG modes: The LASER1 is shifted back half a signal period.</p> <p>Bit #2    Enabling or disabling of <a href="#">Signals for "Laser Active" Operation</a>.  = 0:   The <a href="#">Signals for "Laser Active" Operation</a> are enabled.  = 1:   The <a href="#">Signals for "Laser Active" Operation</a> are disabled  (the signals are set to their respective "Off" level).</p> <p>Bit #3    Level of LASERON signal.  = 0:   Is set to active-HIGH.  = 1:   Is set to active-LOW.  If there is no change of the pin assignment with <a href="#">config_laser_signals</a>, the LASERON signal corresponds to the signal at the LASERON pin, see <a href="#">Figure 17</a>.</p> <p>Bit #4    Levels of LASER1 signal and LASER2 signal.  = 0:   Are set to active-HIGH.  = 1:   Are set to active-LOW.  If there is no change of the pin assignment with <a href="#">config_laser_signals</a>, the LASER1 signal (LASER2 signal) corresponds to the signal at the LASER1 pin (LASER2 pin), see <a href="#">Figure 17</a>.</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>



<b>Ctrl Command</b>	<b>set_laser_control</b>
Parameters (cont'd)	<p>Bit #6 = 0: Synchronization is switched off (default setting). See <a href="#">Chapter 7.4.10 "Output Synchronization", page 195</a>.</p> <p>= 1: Synchronization is switched on.</p> <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 186</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 Reserved.</p> <p>Bit #30 Reserved.</p> <p>Bit #31 Reserved.</p>
Comments	<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 <a href="#">Hardware reset</a>, 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>. Even 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	set_laser_control
Comments (cont'd)	<ul style="list-style-type: none"> <li>For the RTC5 predecessors, the laser signal levels are defined by solder jumpers. The RTC5 lets users control them completely by software. <a href="#">get_startstop_info</a> queries the current status of the laser control signals (Bit #9) and whether the signals are set to active-HIGH or active-LOW (Bit #13).</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> </ul> <p>As of DLL 546, OUT 546: By <a href="#">get_startstop_info</a> ( Bit #14 ) the current state can be queried.</p> <ul style="list-style-type: none"> <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 (Bit #1 = 1) can be set for better synchronization of an analog output, for example, in <a href="#">Softstart Mode</a>, see <a href="#">Chapter 7.4.7 "Softstart Mode", page 184</a> or the <a href="#">Pixel Output Mode</a>, see <a href="#">Chapter 8.7 "Pixel Output Mode", page 244</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	Last change: DLL 546, OUT 546: <a href="#">get_startstop_info</a> ( Bit #14 ) support.
References	<a href="#">set_laser_mode</a> , <a href="#">config_laser_signals</a>

<b>Undelayed Short List Command</b>	<b>set_laser_delays</b>				
<b>Function</b>	Sets the <b>LaserOn Delay</b> and the <b>LaserOff Delay</b> .				
<b>Call</b>	<code>set_laser_delays( LaserOnDelay, LaserOffDelay )</code>				
<b>Parameters</b>	<table> <tr> <td>LaserOnDelay</td> <td><b>LaserOn Delay</b>. As a signed 32-bit value. 1 Bit equals <math>0.5 \mu\text{s}</math>. Allowed value range: <math>[-2^{31} \dots + (2^{21}-1)]</math>. Values over <math>(2^{21}-1)</math> are clipped.</td> </tr> <tr> <td>LaserOffDelay</td> <td><b>LaserOff Delay</b>. As an unsigned 32-bit value. 1 Bit equals <math>0.5 \mu\text{s}</math>. Allowed value range: <math>[0 \dots + (2^{21}-1)]</math>. Values over <math>(2^{21}-1)</math> are clipped.</td> </tr> </table>	LaserOnDelay	<b>LaserOn Delay</b> . As a signed 32-bit value. 1 Bit equals $0.5 \mu\text{s}$ . Allowed value range: $[-2^{31} \dots + (2^{21}-1)]$ . Values over $(2^{21}-1)$ are clipped.	LaserOffDelay	<b>LaserOff Delay</b> . As an unsigned 32-bit value. 1 Bit equals $0.5 \mu\text{s}$ . Allowed value range: $[0 \dots + (2^{21}-1)]$ . Values over $(2^{21}-1)$ are clipped.
LaserOnDelay	<b>LaserOn Delay</b> . As a signed 32-bit value. 1 Bit equals $0.5 \mu\text{s}$ . Allowed value range: $[-2^{31} \dots + (2^{21}-1)]$ . Values over $(2^{21}-1)$ are clipped.				
LaserOffDelay	<b>LaserOff Delay</b> . As an unsigned 32-bit value. 1 Bit equals $0.5 \mu\text{s}$ . Allowed value range: $[0 \dots + (2^{21}-1)]$ . Values over $(2^{21}-1)$ are clipped.				
<b>Comments</b>	<ul style="list-style-type: none"> <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 143</a>). The <code>LaserOffDelay &gt; LaserOnDelay</code> condition required by the RTC5's predecessor boards is therefore unnecessary.</li> <li>Observe the notes in <a href="#">Chapter 7.2.1 "Laser Delays", page 133</a>.</li> <li>If the <b>LaserOn Delay</b> is negative, the total marking time is extended.</li> <li>The <a href="#">XY2-100 Converter (Accessory)</a> introduces a <math>10 \mu\text{s}</math> runtime latency to scan system control. This runtime latency can be compensated by increasing the <b>LaserOn Delay</b> and <b>LaserOff Delay</b> by <math>10 \mu\text{s}</math> each.</li> <li>The default setting after <code>load_program_file</code> corresponds to <code>set_laser_delays( 20, 20 )</code>.</li> </ul>				
<b>RTC4→RTC5</b>	Basically unchanged functionality. In addition: increased value range. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified delay values by 2. The allowed value ranges decrease accordingly.				
<b>Version info</b>	–				
<b>References</b>	<a href="#">set_scanner_delays</a>				

<b>Ctrl Command</b>	<b>set_laser_mode</b>
<b>Function</b>	Sets the laser mode of the RTC5.
<b>Call</b>	<code>set_laser_mode( Mode )</code>
<b>Parameters</b>	<p>Mode      As an unsigned 32-bit value.</p> <p>= 0: <b>CO<sub>2</sub> Mode.</b>  = 1: <b>YAG Mode 1.</b>  = 2: <b>YAG Mode 2.</b>  = 3: <b>YAG Mode 3.</b>  = 4: <b>Laser Mode 4.</b>  = 5: <b>YAG Mode 5.</b>  = 6: <b>Laser Mode 6.</b></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>
<b>RTC4→RTC5</b>	Additional laser modes, for example, YAG Mode 5, <b>Laser Mode 6</b> and Pulse Picking. Otherwise, unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<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 <b>ANALOG OUT1</b> and <b>ANALOG OUT2</b> output ports, as well as for the 8-bit digital output port of the RTC5.
<b>Call</b>	<code>set_laser_off_default( AnalogOut1, AnalogOut2, DigitalOut )</code>
<b>Parameters</b>	<p>AnalogOut1      12-bit value for analog output port <b>ANALOG OUT1</b>, see also <a href="#">Section "12-Bit Analog Output Port 1 and 2", page 61</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 <b>ANALOG OUT2</b>.  See <a href="#">AnalogOut1</a>.</p> <p>DigitalOut      8-bit value for the 8-bit digital output port, see also <a href="#">Section "8-Bit Digital Output Port", page 68</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>
<b>RTC4→RTC5</b>	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for AnalogOut1 and AnalogOut2 by 4.
<b>Version info</b>	–
<b>References</b>	<a href="#">set_port_default</a>



<b>Ctrl Command</b>	<b>set_laser_pin_out</b>
<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.</p> <p>Bit # 1: DIGITAL OUT2.</p> <p>Bit # 2: Reserved.</p> <p>...</p> <p>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 259</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_laser_pin_out_list</a> , <a href="#">get_laser_pin_in</a>

<b>Undelayed Short List Command</b>	<b>set_laser_pin_out_list</b>
<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→RTC5	New command.
Version info	–
References	<a href="#">set_laser_pin_out</a>

<b>Delayed Short List Command</b>	<b>set_laser_pulses</b>
<b>Function</b>	Defines the output period and the pulse lengths for the signals LASER1 and LASER2 for "laser active" operation.
<b>Call</b>	<code>set_laser_pulses( HalfPeriod, PulseLength )</code>
<b>Parameters</b>	<p><code>HalfPeriod</code> <i>Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</i></p> <p><code>PulseLength</code> <i>Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</i></p>
	<ul style="list-style-type: none"> <li>• Note that <i>half</i> the period length must be specified for <code>HalfPeriod</code> (see <a href="#">Figure 59</a> and <a href="#">Figure 60</a>).</li> <li>• If <code>HalfPeriod</code> = 0 and/or <code>PulseLength</code> = 0, no laser signals are outputted.</li> <li>• If the <a href="#">Softstart Mode</a> is used (see <a href="#">Chapter 7.4.7 "Softstart Mode", page 184</a>), <code>HalfPeriod</code> should not be smaller than 104 (this is not automatically checked). This value corresponds to a laser pulse frequency of approx. 308 kHz.</li> <li>• If <code>PulseLength</code> is larger than the output period (<math>2 \times \text{HalfPeriod}</math>), the laser is permanently on.</li> <li>• The signal level is defined by <a href="#">set_laser_control</a>.</li> <li>• The <a href="#">set_laser_pulses</a> command 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>
<b>RTC4→RTC5</b>	New command. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified values for <code>HalfPeriod</code> and <code>PulseLength</code> by 8. The allowed value ranges decrease accordingly.
<b>Version info</b>	–
<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 <a href="#">set_laser_pulses</a> , but a control command.				
<b>Call</b>	<code>set_laser_pulses_ctrl( HalfPeriod, PulseLength )</code>				
<b>Parameters</b>	<table> <tr> <td>HalfPeriod</td> <td>Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</td> </tr> <tr> <td>PulseLength</td> <td>Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</td> </tr> </table>	HalfPeriod	Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].	PulseLength	Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].
HalfPeriod	Half of the output period. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].				
PulseLength	Pulse length of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. 1 bit equals 1/64 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">set_laser_pulses</a></li> </ul>				
RTC4→RTC5	New command. In RTC4 Compatibility Mode: like <a href="#">set_laser_pulses</a> .				
<b>Version info</b>	–				
<b>References</b>	<a href="#">set_laser_pulses</a> , <a href="#">set_laser_timing</a>				

<b>Delayed Short List Command</b>	<b>set_laser_timing</b>	
<b>Function</b>	Defines the output period and the pulse lengths for the signals LASER1 and LASER2 for "laser active" operation.	
<b>Call</b>	set_laser_timing( <code>HalfPeriod</code> , <code>PulseLength1</code> , <code>(PulseLength2)</code> , <code>TimeBase</code> )	
<b>Parameters</b>	<code>HalfPeriod</code> Half of the output period. In bits. As an unsigned 32-bit value. <ul style="list-style-type: none"> <li>• In <b>RTC5 Standard Mode</b>:            1 bit equals 1/64 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</li> <li>• In <b>RTC4 Compatibility Mode</b>:            1 bit equals 1/8 <math>\mu</math>s or 1 <math>\mu</math>s, depending on the selected clock frequency. With respect to the specified <code>TimeBase</code>, the value is converted to an integer-multiple of 1/64 <math>\mu</math>s. The allowed range is correspondingly smaller.</li> </ul>	
	<code>PulseLength1</code> Pulse lengths of the laser signals LASER1 and LASER2. In bits. As an unsigned 32-bit value. <ul style="list-style-type: none"> <li>• In <b>RTC5 Standard Mode</b>:            1 bit equals 1/64 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</li> <li>• In <b>RTC4 Compatibility Mode</b>:            1 bit equals 1/8 <math>\mu</math>s or 1 <math>\mu</math>s, depending on the selected clock frequency. With respect to the specified <code>TimeBase</code>, the value is converted to an integer-multiple of 1/64 <math>\mu</math>s. The allowed range is correspondingly smaller.</li> </ul>	
	<code>(PulseLength2)</code> Value is not used. (As an unsigned 32-bit value.) With the RTC5, the pulse lengths of laser signals LASER1 and LASER2 are always identical and are definable by <code>PulseLength1</code> .	
	<code>TimeBase</code> As an unsigned 32-bit value. <ul style="list-style-type: none"> <li>• In <b>RTC5 Standard Mode</b>, the value is ignored and the clock frequency is fixed at 64 MHz. 1 bit equals 1/64 <math>\mu</math>s.</li> <li>• In <b>RTC4 Compatibility Mode</b>, the <code>TimeBase</code> value is handled as follows:            =0: sets the time base to 1 MHz. 1 bit equals 1 <math>\mu</math>s.            ≠0: sets the time base to 8 MHz. 1 bit equals 1/8 <math>\mu</math>s.         </li> </ul>	



<b>Delayed Short List Command</b>	<b>set_laser_timing</b>
Comments	<ul style="list-style-type: none"> <li>In <b>RTC5 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>For <b>RTC4 Compatibility Mode</b>, SCANLAB generally recommends setting the time base to 8 MHz. In this mode, a time base of 1 MHz should only be chosen if necessary.</li> <li>Refer to <b>Chapter 7.4 "Laser Control", page 173</b>, for details.</li> <li>In <b>RTC4 Compatibility Mode</b>, in <b>Laser Mode 4</b> and <b>Laser Mode 6</b>, the time base for signals LASER1 and LASER2 is independently of <code>TimeBase</code> always 1/8 <math>\mu</math>s.</li> </ul>
RTC4→RTC5	<ul style="list-style-type: none"> <li>With the RTC5, the pulse lengths of laser signals LASER1 and LASER2 are always identical.</li> <li>Clock frequency: <ul style="list-style-type: none"> <li>In <b>RTC5 Standard Mode</b>: fixed clock frequency of 64 MHz.</li> <li>In <b>RTC4 Compatibility Mode</b>: 1 MHz or 8 MHz.</li> </ul> </li> </ul>
Version info	–
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	Execution produces an unconditional jump to the specified address within the list memory area. The next command there is executed immediately without delay.
Call	<code>set_mark_speed_ctrl( Speed )</code>
Parameters	Speed      Like <b>set_mark_speed</b> .
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→RTC5	Unchanged functionality. In addition: increased value range.
Version info	–
References	<b>list_jump_pos</b>

<b>Delayed Short List Command</b>	<b>set_mark_speed</b>
Function	Sets the mark speed for the <b>Vector commands</b> and <b>"Arc" commands</b> .
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]. Out-of-range values are clipped to the boundary values.
Comments	<ul style="list-style-type: none"> <li>By default a mark speed of 1,000 <i>bits/ms</i> is preset.</li> <li>The specified mark speed is used for all <b>[*]mark[*] Commands</b> and <b>"Arc" commands</b> 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 [<i>bits/ms</i>] 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 <b>get_table_para</b> or <b>get_head_para</b>.</p> <ul style="list-style-type: none"> <li><b>set_mark_speed</b> is also available as control command <b>set_mark_speed_ctrl</b>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified Speed value by 16. The allowed value range decreases accordingly.
Version info	–
References	<b>mark_abs, mark_rel, set_jump_speed, set_mark_speed_ctrl</b>

<b>Ctrl Command</b>	<b>set_mark_speed_ctrl</b>
Function	Like <b>set_mark_speed</b> , but a control command.
Call	<code>set_mark_speed_ctrl( Speed )</code>
Parameters	Speed      mark 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><b>set_mark_speed_ctrl</b> is not executed (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if: <ul style="list-style-type: none"> <li>the <b>BUSY list execution status</b> is set</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 list execution status</b> set)</li> <li>the <b>INTERNAL-BUSY list execution status</b> is set</li> </ul> </li> </ul>
RTC4→RTC5	New command. In <b>RTC4 Compatibility Mode</b> : like <b>set_mark_speed</b>
Version info	–
References	<b>set_mark_speed, set_jump_speed, set_jump_speed_ctrl</b>



<b>Ctrl Command</b>	<b>set_matrix</b>
<b>Function</b>	Sets the coefficients of the general transformation matrix $M_T$ for all subsequent coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a> .
<b>Call</b>	<code>set_matrix( HeadNo, M11, M12, M21, M22, 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.                      Requires <a href="#">Option "Second Scan Head Control"</a>.              = 0, 3: The definition affects <i>both</i> scan head connectors.              = 4: The definition affects the virtual <a href="#">Image field</a> (see also comments).                      Only the three least significant bits are evaluated.</p> <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 <a href="#">Image field</a> and [-1.5...+1.5] in the virtual <a href="#">Image field</a>. With invalid value, <code>set_matrix</code> 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.                      For HeadNo = 0...3, the following applies:                  = 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.                  = 1: The new total transformation is calculated immediately (or prior the next list command, if currently the <a href="#">BUSY list execution status</a> or <a href="#">INTERNAL-BUSY list execution status</a> is set) and applied to the current position.                      Signals for "Laser Active" Operation are switched off in advance.                  = 2: The new total transformation (total matrix and offset) is only calculated 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.                  = 3: Like at_once = 1, but the laser control signals remain unchanged.                  &gt; 3: Like at_once = 2.</p>



Ctrl Command	set_matrix
Comments	<ul style="list-style-type: none"> <li>See <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a>.</li> <li>The coordinate transformation defined by <code>HeadNo = 4</code> is only in effect during a Processing-on-the-fly application initiated by <a href="#">set_fly_2d</a>, see <a href="#">Section "Coordinate Transformations in the Virtual Image Field", page 158</a>: <ul style="list-style-type: none"> <li>– <a href="#">set_fly_2d</a></li> <li>– <math>\geq</math> DLL 541 <a href="#">set_fly_x</a></li> <li>– <math>\geq</math> DLL 541 <a href="#">set_fly_y</a></li> </ul> </li> </ul> <p>Here, the <code>at_once</code> parameter is ignored.</p>
RTC4→RTC5	<p>Unchanged primary functionality (matrix definition), however:</p> <ul style="list-style-type: none"> <li>The parameters <code>HeadNo</code> 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 212</a>.</li> </ul>
Version info	Last change DLL 541: also available with <a href="#">set_fly_x</a> , <a href="#">set_fly_y</a> .
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 one of the 4 coefficients of the general transformation matrix $M_T$ during execution of a list, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a> .
<b>Call</b>	<code>set_matrix_list( HeadNo, Ind1, Ind2, Mij, at_once )</code>
<b>Parameters</b>	<p>HeadNo      See <a href="#">set_matrix</a>. However, HeadNo = 4 is not available.</p> <p>Ind1      Row index and column index of the matrix coefficient to be changed. As an unsigned 32-bit value. Allowed values: [Index uneven: 1, Index even: 2].</p> <p>Ind2</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, <code>set_matrix_list</code> is replaced by a <a href="#">list_nop</a>.</p> <p>at_once      Determines when the defined transformation becomes effective: = 0: The transformation settings are only collected 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). = 1: The transformation is immediately calculated (including all transformation settings that were collected until then) and processed prior to the next list command. <a href="#">Signals for "Laser Active" Operation</a> are switched off in advance. = 2: The transformation settings are only collected and intermediately stored (as with <code>at_once = 0</code>). However, The transformation is immediately calculated (including all transformation settings that were collected and intermediately stored until then) and applied to the current position when the next <code>jump_abs</code> or <code>jump_rel</code> (only if no list is currently being executed: also <code>goto_xy</code> or <code>goto_xyz</code>) is executed. = 3: As with <code>at_once = 1</code>, but the laser control signals remain unaffected. &gt; 3: See <code>at_once = 2</code>. As an unsigned 32-bit value.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <code>set_matrix_list</code> only allows changing one of the 4 coefficients at a time. To change several coefficients during execution of a list, <code>set_matrix_list</code> 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 209</a>.</li> <li>• HeadNo = 4 is not available.</li> </ul>
RTC4→RTC5	See <a href="#">set_matrix</a> .
Version info	–
References	<a href="#">set_matrix</a>

<b>Ctrl Command</b>	<b>set_max_counts</b>
<b>Function</b>	Defines the maximum number of <b>External Starts</b> .
<b>Call</b>	<code>set_max_counts( Counts )</code>
<b>Parameters</b>	Counts      Maximum number of <b>External Starts</b> . 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 <b>External Starts</b> has been reached, the external start input is disabled (see <b>set_control_mode</b>, Bit #0 = 0).</li> <li>If Counts = 0, the number of <b>External Starts</b> is unlimited.</li> <li>When the RTC5 is initialized (by <b>load_program_file</b>), Counts is set to 0.</li> <li>The current number of successful <b>External Starts</b> can be read from the corresponding internal counter with <b>get_counts</b>. The counter can be reset by <b>set_control_mode</b>.</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range.
<b>Version info</b>	–
<b>References</b>	<b>get_counts, set_control_mode</b>

<b>Ctrl Command</b>	<b>set_mcbsp_freq</b>
<b>Function</b>	Sets the transmission frequency of the <b>McBSP interface</b> .
<b>Call</b>	<code>mcbsp_freq = set_mcbsp_freq( Freq )</code>
<b>Parameters</b>	Freq      Desired transmission frequency of the <b>McBSP interface</b> 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 <b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b> 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, <b>set_mcbsp_freq</b> 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 <b>McBSP interface</b> by <b>mcbsp_init</b>.</li> <li>The signals and operating conditions of the <b>McBSP interface</b> are presented in the <b>Chapter 4.6.6 "SPI / I2C Socket Connector", page 71</b>.</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→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<b>mcbsp_init, set_mcbsp_out, set_mcbsp_out_ptr</b>

<b>Ctrl Command</b>	<a href="#">set_mcbsp_global_matrix</a>
<b>Function</b>	Activates matrix correction for <b>“Global Online Positioning”</b> by the <b>McBSP</b> interface.
<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 216</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 <a href="#">set_mcbsp_global_matrix</a>. Subsequent activation of other options (by <a href="#">set_mcbsp_global_x</a>, <a href="#">set_mcbsp_global_y</a> or <a href="#">set_mcbsp_global_rot</a>) deactivates the matrix option.</li> <li>The following restrictions apply to the matrix coefficients transferred over the <b>McBSP interface</b> (as with <a href="#">set_matrix</a>(<code>HeadNo = 4, ...</code>)): <ul style="list-style-type: none"> <li>The allowed value range for matrix coefficients is <math>[-1.5\dots+1.5]</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 = \{ M_{00}, M_{01}, M_{10}, M_{11} \} = \{ m_{11}, m_{12}, m_{21}, m_{22} \}</math>.  Conversely, the RTC5 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 <a href="#">read_mcbsp(1)</a>.</li> <li>The <b>McBSP interface</b> cannot be simultaneously used for an <b>Online Positioning</b> and <b>Processing-on-the-fly</b> applications.</li> </ul> <p>See also <a href="#">Section “Notes”, page 215</a>.</p>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	Available as of DLL 545, OUT 545.
<b>References</b>	<a href="#">set_mcbsp_global_matrix_list</a> , <a href="#">set_mcbsp_matrix</a>

<b>Undelayed Short List Command</b>	<a href="#">set_mcbsp_global_matrix_list</a>
<b>Function</b>	Like <a href="#">set_mcbsp_global_matrix</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>Like <a href="#">set_mcbsp_global_matrix</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	Available as of DLL 545, OUT 545.
<b>References</b>	<a href="#">set_mcbsp_global_matrix</a>

<b>Ctrl Command</b>	<b>set_mcbsp_global_rot</b>
<b>Function</b>	Activates or deactivates rotation correction for <b>“Global Online Positioning”</b> by the <b>McBSP interface</b> .
<b>Call</b>	set_mcbsp_global_rot( Resolution )
<b>Parameters</b>	Resolution As a 64-bit IEEE floating point value. $2^{-26} < \text{Resolution} < 2^{26}$ : scaling factor (correction is activated), otherwise: correction is deactivated. Resolution = McBSP/SPI bits per full circle.
<b>Comments</b>	<ul style="list-style-type: none"> <li>See also <b>Chapter 8.3.2 “Global Online Positioning”</b>, page 216.</li> <li>With an McBSP/SPI rotation correction input value of <math>\text{Rot}_{\text{in}}</math>, <b>set_mcbsp_global_rot</b> functions like <b>set_angle(4, Angle × 360°, ...)</b>, 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 <b>McBSP interface</b> cannot be simultaneously used for an <b>Online Positioning</b> and Processing-on-the-fly applications.  See also <b>Section “Notes”</b>, page 215.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 545, OUT 545.
References	<b>set_mcbsp_global_rot_list</b> , <b>set_mcbsp_rot</b>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_global_rot_list</b>
<b>Function</b>	Like <b>set_mcbsp_global_rot</b> , but a list command.
<b>Call</b>	set_mcbsp_global_rot_list( Resolution )
<b>Parameters</b>	Resolution Like <b>set_mcbsp_global_rot</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <b>set_mcbsp_global_rot</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 545, OUT 545.
References	<b>set_mcbsp_global_rot</b>

<b>Ctrl Command</b>	<a href="#"><b>set_mcbsp_global_x</b></a>
<b>Function</b>	Activates or deactivates x offset correction for “ <a href="#">Global Online Positioning</a> ” by the <a href="#">McBSP interface</a> .
<b>Call</b>	<code>set_mcbsp_global_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>• See also <a href="#">Chapter 8.3.2 “Global Online Positioning”</a>, page 216.</li> <li>• With an <a href="#">McBSP</a> input value of <math>X_{in}</math> for the x offset correction, <a href="#"><code>set_mcbsp_global_x</code></a> functions like <code>set_offset_xyz(4, XOffset,...)</code>, whereby <math>XOffset</math> (in bits) = <math>\text{Scale} \times X_{in}</math>.  <math>XOffset</math> values outside of <math>[-524,288\dots+524,287]</math> are clipped to the boundary value as long as <math>\text{Scale} \times X_{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 215.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 545, OUT 545.
References	<a href="#"><b>set_mcbsp_global_x_list</b></a> , <a href="#"><b>set_mcbsp_x</b></a>

<b>Undelayed Short List Command</b>	<a href="#"><b>set_mcbsp_global_x_list</b></a>
<b>Function</b>	Like <a href="#"><b>set_mcbsp_global_x</b></a> , but a list command.
<b>Call</b>	<code>set_mcbsp_global_x_list( Scale )</code>
<b>Parameters</b>	Scale      Like <a href="#"><b>set_mcbsp_global_x</b></a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#"><b>set_mcbsp_global_x</b></a>.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 545, OUT 545.
References	<a href="#"><b>set_mcbsp_global_x</b></a>

<b>Ctrl Command</b>	<a href="#">set_mcbsp_global_y</a>
<b>Function</b>	Activates or deactivates y offset correction for “ <a href="#">Global Online Positioning</a> ” by the <a href="#">McBSP interface</a> .
<b>Call</b>	<code>set_mcbsp_global_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>See also <a href="#">Chapter 8.3.2 “Global Online Positioning”</a>, page 216.</li> <li>With an <a href="#">McBSP</a> input value of <math>Y_{in}</math> for the y offset correction, <a href="#">set_mcbsp_global_y</a> functions like <code>set_offset_xyz(4, ..., YOffset,...)</code>, whereby <math>Y_{Offset}</math> (in bits) = <math>\text{Scale} \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 <math>\text{Scale} \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 215.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 545, OUT 545.
References	<a href="#">set_mcbsp_global_y_list</a> , <a href="#">set_mcbsp_y</a>

<b>Undelayed Short List Command</b>	<a href="#">set_mcbsp_global_y_list</a>
<b>Function</b>	Like <a href="#">set_mcbsp_global_y</a> , but a list command.
<b>Call</b>	<code>set_mcbsp_global_y_list( Scale )</code>
<b>Parameters</b>	Scale      Like <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→RTC5	New command.
Version info	Available as of DLL 545, OUT 545.
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 motion (based on position values transferred to the RTC5 by 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:</p> <ul style="list-style-type: none"> <li>= 0:      Processing-on-the-fly correction is switched off.</li> <li>= 1:      Compensation of linear motion in the x direction.</li> <li>= 2:      Compensation of linear motion in the y direction.</li> <li>= 3:      Compensation of linear motion in the x direction and y direction.</li> <li>= 4:      Compensation of rotary motion.</li> <li>= 5:      Processing-on-the-fly correction is switched off.</li> </ul> <ul style="list-style-type: none"> <li>• Mode = 0...5: All <b>McBSP</b> input values are alternatingly copied to internal memory locations 1 and 2.</li> <li>• 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.</li> <li>• Mode = 1...4: Values copied to internal memory location 0 are applied for Processing-on-the-fly correction.</li> </ul> <p>Scale      Scaling factor or rotation resolution. As a 64-bit IEEE floating point value.</p> <ul style="list-style-type: none"> <li>• Mode = 1...3: scaling factor in <math>(RTC5)bits/(McBSP)bit</math>. Allowed value range: <math>1/256 \leq  Scale  \leq 16000.0</math> (if Mode = 3, Scale applies to both axes).</li> <li>• Mode = 4: number of steps (counts) per revolution. Allowed value range: <math> Scale  &gt; 100.0</math>.</li> </ul>
<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 the <b>page 227</b>).</li> <li>• For deactivating Processing-on-the-fly correction, see <b>Chapter 8.6.5 "Deactivating Processing-on-the-fly Corrections", page 236</b>.</li> <li>• 15 bits per axis (with sign) are effectively available for 2D correction (Mode = 3), 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 <b>Section "Notes", page 215</b>.</li> </ul>



<b>Ctrl Command</b>	<b>set_mcbsp_in</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The <b> McBSP interface</b> ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>. That is, data provided is not transmitted, see <a href="#">page 73</a>.</li> <li>If Mode &gt; 5, <b>set_mcbsp_in</b> is <i>not</i> executed (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>If an unallowed Scale parameter value is supplied (for example, Scale = 0), <b>set_mcbsp_in</b> behaves as with Mode = 0.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_mcbsp_in_list</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_in_list</b>				
Function	Like <b>set_mcbsp_in</b> , but a list command.				
Restriction	If the <b>Option Processing-on-the-fly</b> 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).				
Call	<code>set_mcbsp_in_list( Mode, Scale )</code>				
Parameters	<table> <tr> <td>Mode</td> <td>Like <b>set_mcbsp_in</b>.</td> </tr> <tr> <td>Scale</td> <td>Like <b>set_mcbsp_in</b>.</td> </tr> </table>	Mode	Like <b>set_mcbsp_in</b> .	Scale	Like <b>set_mcbsp_in</b> .
Mode	Like <b>set_mcbsp_in</b> .				
Scale	Like <b>set_mcbsp_in</b> .				
Comments	<ul style="list-style-type: none"> <li>If Mode &gt; 5, then <b>set_mcbsp_in_list</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>See <b>set_mcbsp_in</b>.</li> </ul>				
RTC4→RTC5	New command.				
Version info	–				
References	<a href="#">set_mcbsp_in</a>				

Ctrl Command	<b>set_mcbsp_matrix</b>
Function	Activates matrix correction for “Local Online Positioning” by the McBSP interface.
Call	<code>set_mcbsp_matrix()</code>
Comments	<ul style="list-style-type: none"> <li>For “Local Online Positioning”, see <a href="#">Chapter 8.3.1 “Local Online Positioning”</a>, <a href="#">page 213</a>.</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 option.</li> <li>The following restrictions apply to the matrix coefficients transferred over the <a href="#">McBSP interface</a> (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 <a href="#">McBSP interface</a> 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> </ul> $M_{in} = (\text{integer}(M_{ij} * 2^{24}) << 2) + (i << 1) + j$ <p>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 RTC5 determines a coefficient from the input value as follows:</p> $M_T[M_{in} \& 0x3] = (M_{in} >> 2) / 2^{24}.$ <ul style="list-style-type: none"> <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 <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 215</a>.</p>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_mcbsp_matrix_list</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_matrix_list</b>
Function	Like <a href="#">set_mcbsp_matrix</a> , but a list command.
Call	<code>set_mcbsp_matrix_list()</code>
Comments	<ul style="list-style-type: none"> <li>Like <a href="#">set_mcbsp_matrix</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 533, OUT 534, RBF 524.
References	<a href="#">set_mcbsp_matrix</a>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_out</b>				
Function	Defines two signal types for output at the <a href="#">McBSP interface</a> , see also <a href="#">Chapter 4.6.6 "SPI / I2C Socket Connector", page 71</a> .				
Call	<code>set_mcbsp_out( Signal1, Signal2 )</code>				
Parameters	<table> <tr> <td>Signal1</td><td>To-be-outputted signal type. As an unsigned 32-bit value.</td></tr> <tr> <td>Signal2</td><td>To-be-outputted signal type. As an unsigned 32-bit value.</td></tr> </table>	Signal1	To-be-outputted signal type. As an unsigned 32-bit value.	Signal2	To-be-outputted signal type. As an unsigned 32-bit value.
Signal1	To-be-outputted signal type. As an unsigned 32-bit value.				
Signal2	To-be-outputted signal type. As an unsigned 32-bit value.				
Comments	<ul style="list-style-type: none"> <li>The selectable signal types are identical to those of <a href="#">set_trigger</a> (refer to the comments there for the allowed value range, signal types and other information). If the value for Signal1/Signal2 is unallowed, then <code>set_mcbsp_out</code> is replaced by <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC5_PARAM_ERROR</a>).</li> <li>Both selected data signals are continuously transmitted (once per 10 µs 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 <a href="#">Image field</a>) are ignored.</li> <li>For the other data types (Signal1, Signal2 = 0, 24, 31...54), 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> <li>–</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 <a href="#">set_mcbsp_freq</a>).</li> <li>The signals and operating conditions of the <a href="#">Chapter 4.6.6 "SPI / I2C Socket Connector", page 71</a> interface are presented in <a href="#">Chapter 4.6.6 "SPI / I2C Socket Connector", page 71</a>.</li> </ul>				
RTC4→RTC5	New command.				
Version info	–				
References	<a href="#">read_mcbsp</a> , <a href="#">mcbsp_init</a> , <a href="#">set_mcbsp_freq</a> , <a href="#">set_mcbsp_out_ptr</a> , <a href="#">set_trigger</a>				

<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> (Multi channel Buffered Serial Port, see also <a href="#">page 71</a> ).
<b>Call</b>	<code>set_mcbsp_out_ptr( Number, SignalPtr )</code>
<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 <a href="#">set_mcbsp_out</a>.</p> <p>SignalPtr    Pointer (in C and C++ data type <code>ULONG_PTR</code>, an unsigned 32-bit value or unsigned 64-bit value) to an array of <code>Number</code> unsigned 32-bit values, where the to-be-outputted <code>Number</code> signal type numbers are specified.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The memory area for the <code>SignalPtr</code> array must be provided by the user program.</li> <li>If <code>Number &gt; 8</code> and/or <code>SignalPtr = NULL</code>, <a href="#">set_mcbsp_out_ptr</a> is not executed (<a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>The selectable signal types are identical to those of <a href="#">set_trigger</a> (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 cycle). Each individual signal belongs to a different clock cycle. Transmitted values are always from the previous clock cycle.</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...54):                        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 <a href="#">Section "McBSP Interface", page 71</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_mcbsp_out_ptr list</a> , <a href="#">set_mcbsp_out</a> , <a href="#">set_trigger</a>



<b>Multiple List Command</b>	<b>set_mcbsp_out_ptr_list</b>
Function	Like <a href="#">set_mcbsp_out_ptr</a> , but a list command.
Call	<code>set_mcbsp_out_ptr_list( Number, SignalPtr )</code>
Parameters	Number      Like <a href="#">set_mcbsp_out_ptr</a> . SignalPtr      Like <a href="#">set_mcbsp_out_ptr</a> .
Comments	<ul style="list-style-type: none"><li>• <a href="#">set_mcbsp_out_ptr_list</a> requires two list memory positions for <code>Number</code> <math>\geq 1</math>. Sequence of execution:<ol style="list-style-type: none"><li>1. Pending delayed short list commands</li><li>2. First command part as an undelayed short list command</li><li>3. Second command part as a normal list command</li></ol></li><li>• See <a href="#">set_mcbsp_out_ptr</a>.</li></ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 550, OUT 550.
References	<a href="#">set_mcbsp_out_ptr</a>

<b>Ctrl Command</b>	<b>set_mcbsp_rot</b>
<b>Function</b>	Activates or deactivates rotation correction for <b>"Local Online Positioning"</b> by the <b>McBSP interface</b> .
<b>Call</b>	<code>set_mcbsp_rot( Resolution )</code>
<b>Parameters</b>	<p>Resolution    Value.            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/SPI bits per full circle.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>"Local Online Positioning"</b>, see <b>Chapter 8.3.1 ""Local Online Positioning""</b>, <b>page 213</b>.</li> <li>For an McBSP/SPI rotation correction input value of <math>\text{Rot}_{\text{in}}</math>, <b>apply_mcbsp</b> functions like <code>set_angle( ..., Angle <math>\times 360^\circ</math>, ... )</code>, whereby</li> </ul> $\text{Angle in full circles} = \text{Rot}_{\text{in}} / \text{Resolution}$ <p>Only <code>Angle</code> values in the range <math>[0.0 \dots +20.0 \text{ full circles}]</math> are allowed.</p> <ul style="list-style-type: none"> <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>, <b>page 215</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_mcbsp_rot_list</b> , <b>set_mcbsp_x</b> , <b>set_mcbsp_y</b> , <b>apply_mcbsp</b>

<b>Undelayed Short List Command</b>	<b>set_mcbsp_rot_list</b>
<b>Function</b>	Like <b>set_mcbsp_rot</b> , but a list command.
<b>Call</b>	<code>set_mcbsp_rot_list( Resolution )</code>
<b>Parameters</b>	Resolution    Like <b>set_mcbsp_rot</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>Like <b>set_mcbsp_rot</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_mcbsp_rot</b>



<b>Ctrl Command</b>	<b>set_mcbsp_x</b>
<b>Function</b>	Activates or deactivates x offset correction for “ <a href="#">Local Online Positioning</a> ” by the <a href="#">McBSP interface</a> .
<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 “<a href="#">Local Online Positioning</a>”, see <a href="#">Chapter 8.3.1 ““Local Online Positioning””, page 213</a>.</li> <li>With an <a href="#">McBSP</a> input value of <math>X_{in}</math> for the x offset correction, <a href="#">apply_mcbsp</a> 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 <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 215</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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      Like <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>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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 “ <a href="#">Local Online Positioning</a> ” by the <a href="#">McBSP interface</a> .
<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 “<a href="#">Local Online Positioning</a>”, see <a href="#">Chapter 8.3.1 ““Local Online Positioning””, page 213</a>.</li> <li>With an <a href="#">McBSP</a> input value of <math>Y_{in}</math> for the y offset correction, <a href="#">apply_mcbsp</a> 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 <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 215</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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      Like <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>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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 has been 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>RTC5_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    Mode.              As an unsigned 32-bit value.              = 0:      The transmitted value is used directly.              = 1:      The transmitted value is multiplied by P/16384 and then put out.              = 2:      The transferred value is only stored but not used. The Processing-on-the-fly correction is switched off.</p> <p>Ctrl, P and Mode are only relevant for Type 3 (= laser power) of the transmitted data word, see <b>read_multi_mcbsp</b>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Any other previously activated <b>McBSP</b> transmission for <b>Online Positioning</b> and any other previously activated Processing-on-the-fly application with encoder signals or positional values is terminated first.</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 <math>\mu</math>s. Each 10 <math>\mu</math>s, 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 call the command.</li> <li>The <b>McBSP interface</b> ignores the first FrameSync signal after a <b>load_program_file</b> or <b>mcbsp_init</b>. That is, data provided is not transmitted, see <b>page 73</b>.</li> <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 RTC5 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 <b>read_multi_mcbsp</b>.</li> </ul>

<b>Ctrl Command</b>	<b>set_multi_mcbsp_in</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>The remaining (upper) 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 4 extra parameters are not the same as the free variables (see <a href="#">Chapter 6.9.1 "Free Variables", page 123</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 4 McBSP/SPI transfers complete within a <math>10\ \mu\text{s}</math> clock cycle, 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. Even z is used in 20-bit resolution.</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 = 0</code>, subsequent type-3 transfers are outputted directly at the port assigned by <code>Ctrl</code>. For <code>Mode = 1</code>, 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", but without the Softstart suppression.</li> <li>These laser power variation method can be combined with the vector-defined 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 = 0</code>, then laser power variation is switched off. The values transmitted by McBSP/SPI 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 (except for <code>Mode = 2</code>) regarded 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 = 0</code>) to avoid doubled or meaningless output.</li> </ul>
RTC4→RTC5	New command.
Version info	Last change DLL 550, OUT 550: <code>Mode = 2</code> switches off the Processing-on-the-fly correction.
Comments	<a href="#">set_multi_mcbsp_in_list</a> , <a href="#">read_multi_mcbsp</a> , <a href="#">set_wobble_control</a> , <a href="#">set_wobble_vector</a>



Normal List Command	<code>set_multi_mcbsp_in_list</code>						
Function	Like <code>set_multi_mcbsp_in</code> , but a list command.						
Restriction	Like <code>set_multi_mcbsp_in</code> .						
Call	<code>set_multi_mcbsp_in_list( Ctrl, P, Mode )</code>						
Parameters	<table><tr><td>Ctrl</td><td>Like <code>set_multi_mcbsp_in</code>.</td></tr><tr><td>P</td><td>Like <code>set_multi_mcbsp_in</code>.</td></tr><tr><td>Mode</td><td>Like <code>set_multi_mcbsp_in</code>.</td></tr></table>	Ctrl	Like <code>set_multi_mcbsp_in</code> .	P	Like <code>set_multi_mcbsp_in</code> .	Mode	Like <code>set_multi_mcbsp_in</code> .
Ctrl	Like <code>set_multi_mcbsp_in</code> .						
P	Like <code>set_multi_mcbsp_in</code> .						
Mode	Like <code>set_multi_mcbsp_in</code> .						
Comments	<ul style="list-style-type: none"><li>• See <code>set_multi_mcbsp_in</code>.</li></ul>						
RTC4→RTC5	New command.						
Version info	–						
References	<code>set_multi_mcbsp_in</code> , <code>read_multi_mcbsp</code> , <code>set_wobbel_control</code> , <code>set_wobbel_vector</code>						



<b>Variable List Command</b>	<b>set_n_pixel</b>						
<b>Function</b>	In <b>Pixel Output Mode</b> , executes <b>PortOutValue1</b> and <b>PortOutValue2</b> assigned to the <b>set_pixel</b> command <b>Number</b> times in immediate succession.						
<b>Call</b>	<b>set_n_pixel( PortOutValue1, PortOutValue2, Number )</b>						
<b>Parameters</b>	<table> <tr> <td>PortOutValue 1</td> <td>Like <b>set_pixel</b>.</td> </tr> <tr> <td>PortOutValue 2</td> <td>Like <b>set_pixel</b>.</td> </tr> <tr> <td>Number</td> <td>           Number of pixels.            As an unsigned 32-bit value.            Allowed value range: [1...(2<sup>32</sup>-1)]. 0 is automatically set to 1.         </td> </tr> </table>	PortOutValue 1	Like <b>set_pixel</b> .	PortOutValue 2	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.
PortOutValue 1	Like <b>set_pixel</b> .						
PortOutValue 2	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", page 244</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 <b>Number</b> (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 <b>Pixel Output Mode</b> (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 278</a>.</li> </ul>						
RTC4→RTC5	New command. For <b>RTC4 Compatibility Mode</b> : see <b>set_pixel</b> .						
Version info	–						
References	<b>set_pixel</b> , <b>set_pixel_line</b> , <b>set_pixel_line_3d</b>						

<b>Ctrl Command</b>	<b>set_offset</b>
Function	Defines an offset for all subsequent coordinate transformations (see <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a> ).
Call	<code>set_offset( HeadNo, XOffset, YOffset, at_once )</code>
Call	<code>set_offset( HeadNo, XOffset, YOffset, at_once )</code>
Parameters	<p>HeadNo      See <a href="#">set_offset_xyz</a>.</p> <p>XOffset      See <a href="#">set_offset_xyz</a>.</p> <p>YOffset      See <a href="#">set_offset_xyz</a>.</p> <p>at_once      See <a href="#">set_offset_xyz</a>.</p>
Comments	<ul style="list-style-type: none"> <li>• See <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a>.</li> <li>• <code>set_offset</code> has the same effect as <a href="#">set_offset_xyz</a>, but leaves <code>ZOffset</code> unchanged.</li> </ul>
RTC4→RTC5	Unchanged primary functionality (offset definition). However: <ul style="list-style-type: none"> <li>• The parameters <code>HeadNo</code> and <code>at_once</code> are new.</li> <li>• See <a href="#">set_offset_xyz</a>.</li> </ul>
Version info	–
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	<p>HeadNo      Like <a href="#">set_offset_xyz</a>. However, <code>HeadNo = 4</code> is not available.</p> <p>XOffset      Like <a href="#">set_offset_xyz</a>.</p> <p>YOffset      Like <a href="#">set_offset_xyz</a>.</p> <p>at_once      Like <a href="#">set_offset_xyz_list</a>.</p>
RTC4→RTC5	See <a href="#">set_offset_xyz</a> .
Version info	–
References	<a href="#">set_offset</a> , <a href="#">set_offset_xyz_list</a>

<b>Ctrl Command</b>	<b>set_offset_xyz</b>
<b>Function</b>	Defines an offset for all subsequent coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a> .
<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 first scan head connector.                     = 2: The definition only affects the second scan head connector.                     = 0, 3: The definition affects <i>both</i> scan head connectors.                     = 4: The definition affects the virtual <b>Image field</b> (see also comments).                     Only the three least significant bits are evaluated.</p> <p>XOffset      Offsets for the x direction and y direction (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>YOffset      Offsets for the z direction (coordinate translation relative to the origin of the Cartesian coordinate system). In bits.                     As a signed 32-bit value.                     Allowed value range: [-32,768...+32,767].                     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>See <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a>.</li> <li><code>ZOffset</code> is stored only once, applying jointly for both scan head connectors (<code>HeadNo</code> is therefore ignored).</li> <li><math>ZOffset \neq 0</math> 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 and #5 in <a href="#">Chapter 7.3.6 "Output Values to the Scan System", page 170</a>.</li> <li>The coordinate transformation defined by <code>HeadNo = 4</code> is only in effect during a Processing-on-the-fly application initiated by <a href="#">set_fly_2d</a>, see <a href="#">Section "Coordinate Transformations in the Virtual Image Field", page 158</a>:           <ul style="list-style-type: none"> <li>– <a href="#">set_fly_2d</a></li> <li>– <math>\geq</math> DLL 541 <a href="#">set_fly_x</a></li> <li>– <math>\geq</math> DLL 541 <a href="#">set_fly_y</a></li> </ul>           Here, the <code>at_once</code> parameter is ignored.         </li> <li>If <code>HeadNo = 4</code>, then <code>ZOffset</code> is ignored.</li> </ul>
RTC4→RTC5	New command.  In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <code>XOffset</code> and <code>YOffset</code> by 16. The allowed value ranges decrease accordingly. The value range for <code>ZOffset</code> is identical in <b>RTC5 Standard Mode</b> and <b>RTC4 Compatibility Mode</b> .
<b>Version info</b>	Last change DLL 541, OUT 541: also available with <a href="#">set_fly_x</a> and <a href="#">set_fly_y</a> .
<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      Like <a href="#">set_offset_xyz</a> . However, HeadNo = 4 is not available.
	XOffset      Like <a href="#">set_offset_xyz</a> .
	YOffset      Like <a href="#">set_offset_xyz</a> .
	ZOffset      Like <a href="#">set_offset_xyz</a> .
	at_once      Like <a href="#">set_matrix_list</a> .
<b>Comments</b>	<ul style="list-style-type: none"><li>• HeadNo = 4 is not available.</li></ul>
RTC4→RTC5	See <a href="#">set_offset_xyz</a> .
<b>Version info</b>	–
<b>References</b>	<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 271</a>):  <math>((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})</math>                     (= if the bits in <code>IOvalue</code> specified in <code>Mask1</code> are 1 and the bits specified in <code>Mask0</code> are 0), then automatically a <b>pause_list</b> is executed.                     The condition is checked once per 10 <math>\mu\text{s}</math> clock cycle.</li> <li>The paused list can only be resumed by <b>restart_list</b>.</li> <li><code>Mask1 = Mask0 = 0</code> 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>As of DLL 544, OUT 544, the following applies: a conditional <b>pause_list</b> takes precedence over a simultaneously present /STOP signal.</li> </ul>
RTC4→RTC5	New command.
Version info	Last change DLL 544, OUT 544: precedence reversed in favor of <b>pause_list</b> .
References	<b>pause_list</b> , <b>restart_list</b> , <b>set_pause_list_not_cond</b>



<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      Like 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 (see also <a href="#">Chapter 9.3.2 “Conditional Command Execution”, page 271</a>):           <math display="block">((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } (((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0})</math>           (= if the bits in <code>IOvalue</code> specified in <code>Mask1</code> are 1 and the bits specified in <code>Mask0</code> are 0), then automatically a <b>pause_list</b> is executed.           The condition is checked once per <math>10 \mu\text{s}</math> clock cycle.</li> <li>The paused list can only be resumed by <b>restart_list</b>.</li> <li><code>Mask1 = Mask0 = 0</code> 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→RTC5	New command.
Version info	Available as of DLL 544, OUT 544.
References	<a href="#">pause_list</a> , <a href="#">restart_list</a> , <a href="#">set_pause_list_cond</a>



Variable List Command	set_pixel
Function	In the <b>Pixel Output Mode</b> , defines the laser control parameters (pulse length and analog voltage level) for <b>one</b> pixel in an image line.
Call	set_pixel( PortOutValue1, PortOutValue2 )
Parameters	<p>PortOutValue 1 Pixel pulse length.            As an unsigned 32-bit value.            1 bit equals <math>1/64 \mu\text{s}</math>.            Allowed value range: <math>[0 \dots (2^{32}-1)]</math>.</p> <p>PortOutValue 2 12-bit output value (analog voltage level at the analog output port selected with <b>set_pixel_line</b>) for the pixel.            As an unsigned 32-bit value. Higher bits are ignored (see also <b>write_da_x</b>).</p>
Comments	<ul style="list-style-type: none"> <li>• <b>set_pixel</b> is synonymous with <b>set_n_pixel</b> with <b>Number = 1</b> (see comments there).</li> </ul>
RTC4→RTC5	<ul style="list-style-type: none"> <li>• The RTC5 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>• <b>RTC4-Pixel mode 0</b> is no longer supported.</li> <li>• The RTC5 controls the pixel pulses (at the <b>LASER1</b> port) directly by the <b>PulseLength</b> parameter, no longer by the <b>LASERON</b> signal.</li> <li>• <b>"Classic Mode"</b> (compatible to RTC4 <b>Pixel Output Mode 1</b>):            In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified value for <b>PulseLength</b> (<b>PortOutValue1</b>) by 8 and the one for <b>AnalogOut</b> (<b>PortOutValue2</b>) by 4.            The allowed value ranges decrease accordingly.            The other <b>Pixel Output Modes</b> are not RTC4 compatible.</li> </ul>
Version info	–
References	<b>set_n_pixel</b> , <b>set_pixel_line</b> , <b>set_pixel_line_3d</b>

<b>Normal List Command</b>	<b>set_pixel_line</b>								
<b>Function</b>	Activates the <b>Pixel Output Mode</b> and defines various pixel output parameters.								
<b>Call</b>	<code>set_pixel_line( Channel, HalfPeriod, dx, dy )</code>								
<b>Parameters</b>	<table> <tr> <td>Channel</td><td>Number of the analog output port that should output the analog voltage levels defined by subsequent <b>set_pixel</b>/<b>set_n_pixel</b> calls. As an unsigned 32-bit value. Allowed value range: [1, 2] (1: <b>ANALOG OUT1</b>, 2: <b>ANALOG OUT2</b>).</td></tr> <tr> <td>HalfPeriod</td><td><i>Half</i> pixel output period. In bits. As an unsigned 32-bit value. 1 bit equals <math>1/64 \mu\text{s}</math>. Allowed value range: [104...(<math>2^{32}-1</math>)].</td></tr> <tr> <td>dx</td><td>Distance in the x direction and y direction between adjacent pixels. In bits.</td></tr> <tr> <td>dy</td><td>Each: As a 64-bit IEEE floating point value.</td></tr> </table>	Channel	Number of the analog output port that should output the analog voltage levels defined by subsequent <b>set_pixel</b> / <b>set_n_pixel</b> calls. As an unsigned 32-bit value. Allowed value range: [1, 2] (1: <b>ANALOG OUT1</b> , 2: <b>ANALOG OUT2</b> ).	HalfPeriod	<i>Half</i> pixel output period. In bits. As an unsigned 32-bit value. 1 bit equals $1/64 \mu\text{s}$ . Allowed value range: [104...( $2^{32}-1$ )].	dx	Distance in the x direction and y direction between adjacent pixels. In bits.	dy	Each: As a 64-bit IEEE floating point value.
Channel	Number of the analog output port that should output the analog voltage levels defined by subsequent <b>set_pixel</b> / <b>set_n_pixel</b> calls. As an unsigned 32-bit value. Allowed value range: [1, 2] (1: <b>ANALOG OUT1</b> , 2: <b>ANALOG OUT2</b> ).								
HalfPeriod	<i>Half</i> pixel output period. In bits. As an unsigned 32-bit value. 1 bit equals $1/64 \mu\text{s}$ . Allowed value range: [104...( $2^{32}-1$ )].								
dx	Distance in the x direction and y direction between adjacent pixels. In bits.								
dy	Each: As a 64-bit IEEE floating point value.								
<b>Comments</b>	<ul style="list-style-type: none"> <li>Each image line must start with <b>set_pixel_line</b>. <b>set_pixel_line</b> should be preceded by a jump or mark 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.</li> <li>The first list command after <b>set_pixel_line</b> that is <i>not</i> a <b>set_pixel</b>/<b>set_n_pixel</b> turns off the <b>Pixel Output Mode</b> (<b>set_pixel_line</b>, too, ends the <b>Pixel Output Mode</b> before starting it again). In the process, a default pixel is inserted.</li> <li>If <b>Channel &lt; 1</b> or <b>Channel &gt; 2</b>, then <b>set_pixel_line</b> is replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>) and the <b>Pixel Output Mode</b> is <i>not</i> activated.</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", page 244</b>). <b>HalfPeriod</b> must not be smaller than 104 (otherwise it is clipped). This value corresponds to a pixel frequency of approx. 308 kHz.</li> <li>For pixel output frequencies above around 100 kHz (that is, for a <b>HalfPeriod</b> &lt; approx. 320) digital-to-analog conversion cannot always be fully completed. With such pixel output frequencies users must carefully verify whether the results are as expected.</li> <li>The <b>Pixel Output Mode</b> is incompatible with the <b>Softstart Mode</b> (see <b>Chapter 7.4.7 "Softstart Mode", page 184</b>).</li> <li>The <b>Pixel Output Mode</b> <i>should not</i> be used in conjunction with "Automatic Laser Control" (see <b>Chapter 7.4.9 ""Automatic Laser Control""</b>, page 187) if "Automatic Laser Control" readjusts the 12-bit output values at the <b>ANALOG OUT1</b> or <b>ANALOG OUT2</b> output port or pulse lengths (<b>PulseLength</b>) or output periods (<b>HalfPeriod</b>) of the laser signals <b>LASER1</b> and <b>LASER2</b>.</li> <li>The <b>Pixel Output Mode</b> can be combined with Processing-on-the-fly (see <b>Chapter 8.6 "Processing-on-the-fly", page 227</b>).</li> </ul>								



Normal List Command	<code>set_pixel_line</code>
RTC4→RTC5	<ul style="list-style-type: none"> <li>The <b>Pixel Output Mode</b> <i>cannot</i> be combined with <b>Sky Writing</b> (see <a href="#">Chapter 7.2.4 "Sky Writing", page 149</a>).</li> <li>The <b>Pixel Output Mode</b> <i>cannot</i> be combined with <b>Wobbel</b> (see <a href="#">Chapter 8.4 "Wobbel Mode", page 217</a>).</li> <li>See also <a href="#">Chapter 8.7 "Pixel Output Mode", page 244</a>.</li> <li>The RTC5 only provides the <b>RTC4-Pixel mode</b> = 1.</li> <li>The RTC5 allows selection of the output channel <code>Channel</code> for analog signals.</li> <li>For the RTC5, the pixel output period is specified as the half output period <code>HalfPeriod</code>.</li> <li>With the RTC5, <code>set_pixel_line</code> requires only one list entry, as with other normal list commands.</li> <li>In <b>RTC4 Compatibility Mode</b>, the RTC5 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>
Version info	–
References	<a href="#">set_pixel</a> , <a href="#">set_n_pixel</a> , <a href="#">set_pixel_line_3d</a>



Multiple List Command	<code>set_pixel_line_3d</code>
Function	Activates the <b>Pixel Output Mode</b> and defines various pixel output parameters.
Call	<code>set_pixel_line_3d( Channel, HalfPeriod, dX, dY, dZ )</code>
Parameters	Channel      Like <a href="#">set_pixel_line</a> .
	HalfPeriod    Like <a href="#">set_pixel_line</a> .
	dX            Like <a href="#">set_pixel_line</a> .
	dY            Like <a href="#">set_pixel_line</a> .
	dZ            Distance in the z direction between adjacent pixels. In bits. As a 64-bit IEEE floating point value. As with all z coordinates, dZ too is scaled 16x smaller than dX and dY.
Comments	<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 dZ 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→RTC5	New command. <b>RTC4 Compatibility Mode:</b> see <a href="#">set_pixel_line</a> . In <b>RTC4 Compatibility Mode</b> , the RTC5 does not multiply the specified value for dZ by 16.
Version info	–
References	<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 <code>Value</code> is to be defined. As an unsigned 32-bit value. Allowed values: = 0: <b>ANALOG OUT1</b> output port. See also <a href="#">Section "12-Bit Analog Output Port 1 and 2", page 61</a>. = 1: <b>ANALOG OUT2</b> output port. See also <a href="#">Section "12-Bit Analog Output Port 1 and 2", page 61</a>. = 2: 8-bit digital output port. See also <a href="#">Section "8-Bit Digital Output Port", page 68</a>. = 3: 16-bit digital output port. See also <a href="#">Section "16-Bit Digital Input Port and 16-Bit Digital Output Port", page 65</a>. = 4: 2-bit digital output port. See also <a href="#">Section "2-Bit Digital Output Port", page 61</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]. Bits with higher significance are ignored. For Port = 2: 8-bit values [0...255]. Bits with higher significance are ignored. For Port = 3: 16-bit values [0...(2<sup>16</sup>-1)]. Bits with higher significance are ignored. For Port = 4: 2-bit values [0...3]. Bits with higher significance 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 RTC5 (by <b>load_program_file</b>), 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 <b>stop_execution</b> or by an <b>External Stop</b>. 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 0/1 (<b>ANALOG OUT1</b> or <b>ANALOG OUT2</b>) are also set at the end of <b>Pixel Output Mode</b> (see <a href="#">Chapter 8.7 "Pixel Output Mode", page 244</a>).</li> <li>After initialization of position-dependent or speed-dependent laser control by <b>set_auto_laser_control</b>, the corresponding default value (for output port 0, 1, 2 or 3, depending on the selected laser signal parameter <code>Ctrl</code>), is also outputted each time the laser is switched off after marking or when position-dependent or speed-dependent laser control is set to another <code>Ctrl</code> parameter by <b>set_auto_laser_control</b> or deactivated by <b>set_auto_laser_control</b> (<code>Ctrl</code> = 0) (see <a href="#">Section "General Notes", page 188</a>). If the default value is then defined as "-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>RTC5_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→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified Value for <code>Port</code> = 0/1 by 4.</p>
Version info	–
References	<a href="#">set_laser_off_default</a> , <a href="#">set_default_pixel</a> , <a href="#">set_port_default_list</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>Like <code>set_port_default</code>.</td> </tr> <tr> <td>Value</td> <td>Like <code>set_port_default</code>.</td> </tr> </table>	Port	Like <code>set_port_default</code> .	Value	Like <code>set_port_default</code> .
Port	Like <code>set_port_default</code> .				
Value	Like <code>set_port_default</code> .				
Comments	<ul style="list-style-type: none"> <li>See <code>set_port_default</code>.</li> </ul>				
RTC4→RTC5	New command.				
Version info	Available as of DLL 544, OUT 544.				
References	<a href="#">set_port_default</a>				

<b>Ctrl Command</b>	<b>set_pulse_picking</b>
<b>Function</b>	Switches on <b>Pulse Picking Laser Mode</b> .
<b>Call</b>	<code>set_pulse_picking( No )</code>
<b>Parameters</b>	<p>No                    As an unsigned 32-bit value.                          Allowed value range: [0...63].</p> <p>= 0:                LASER2 puts out the LASERON signal.</p> <p>= 1...63:           LASER2 puts out each <math>No^{th}</math> LASER1 pulse.</p> <p><math>No &gt; 63</math>:        No is clipped to 63                          (<b>get_last_error</b> return code <code>RTC5_PARAM_ERROR</code>).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>Pulse Picking Laser Mode</b>, see <b>Chapter 7.4.8 "Pulse Picking Laser Mode", page 186</b>.</li> <li>The pulse picking signals are outputted until a different laser mode gets set by <b>set_laser_mode</b> (the <b>Pulse Picking Laser Mode</b> gets switched off if any other laser mode switches on by <b>set_laser_mode</b>).</li> <li>You can <i>not</i> switch on <b>Pulse Picking Laser Mode</b> by <b>set_laser_mode</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_laser_mode</b> , <b>set_pulse_picking_list</b>

<b>Ctrl Command</b>	<b>set_pulse_picking_length</b>
<b>Function</b>	Defines a constant pulse length for the "laser active" LASER2 signal in <b>Pulse Picking Laser Mode</b> .
<b>Call</b>	<code>set_pulse_picking_length( Length )</code>
<b>Parameters</b>	<p>Length              Pulse length.                          As an unsigned 32-bit value.                          Allowed value range: [0...65,535]. Bits with higher significance are ignored.                          1 bit equals <math>1/64 \mu s</math>.                          The default value after <b>load_program_file</b> is 0.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For the value to take effect, you must activate <b>Pulse Picking Laser Mode</b> by <b>set_pulse_picking</b> and constant pulse length mode by <b>set_laser_control</b> (Bit #7 = 1). The value then takes immediate effect, even if a marking is carried out.</li> <li>If <b>Pulse Picking Laser Mode</b> has been activated, but not constant pulse length mode (<b>set_laser_control</b> (Bit #7 = 0)), then the pulse-picking signal uses the pulse length of the LASER1 signal (see <b>Chapter 7.4.8 "Pulse Picking Laser Mode", page 186</b>).</li> <li>If neither <b>Pulse Picking Laser Mode</b> nor constant pulse length mode has been activated, then the value <code>Length</code> is irrelevant (<b>set_pulse_picking</b>, <b>set_laser_control</b> and <b>set_pulse_picking_length</b> can be activated in any desired order).</li> <li>After <b>set_pulse_picking</b>( 0 ), LASER2 is continuously output the LASERON signal, but no constant pulse length signal.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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→RTC5	New command.
Version info	–
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 <a href="#">Figure 60</a> ).
Call	<code>set_qswitch_delay( Delay )</code>
Parameters	Delay      Q-Switch delay. As an unsigned 32-bit value. 1 bit equals 1/64 $\mu$ s. 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→RTC5	New command. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified value for <code>Delay</code> by 8. The allowed value range decreases accordingly.
Version info	–
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 <b>set_qswitch_delay</b> , but a list command.
Call	<code>set_qswitch_delay_list( Delay )</code>
Parameters	Delay      Like <b>set_qswitch_delay</b> .
RTC4→RTC5	New command.
Version info	–
References	<b>set_qswitch_delay</b>

<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>	<table> <tr> <td>X</td> <td>Position of the rotation center referenced to the zero point (0 0) of the <a href="#">Image field</a>. As a signed 32-bit value. Allowed value range: [-2<sup>24</sup>...(2<sup>24</sup>-1)].</td> </tr> <tr> <td>Y</td> <td>Like X (analogously).</td> </tr> </table>	X	Position of the rotation center referenced to the zero point (0 0) of the <a href="#">Image field</a> . As a signed 32-bit value. Allowed value range: [-2 <sup>24</sup> ...(2 <sup>24</sup> -1)].	Y	Like X (analogously).
X	Position of the rotation center referenced to the zero point (0 0) of the <a href="#">Image field</a> . As a signed 32-bit value. 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 "Compensating Rotary Motions"</a>, page 232.</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 <a href="#">Image field</a> (the allowed range equals the 32 fold of the <a href="#">Image field</a>).</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→RTC5	Unchanged functionality. In addition: increased value range. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified values for X and Y by 16. The allowed value range decreases accordingly.				
<b>Version info</b>	–				
<b>References</b>	<a href="#">set_rot_center_list</a> , <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>	<table> <tr> <td>X</td> <td>Like <a href="#">set_rot_center</a>.</td> </tr> <tr> <td>Y</td> <td>Like <a href="#">set_rot_center</a>.</td> </tr> </table>	X	Like <a href="#">set_rot_center</a> .	Y	Like <a href="#">set_rot_center</a> .
X	Like <a href="#">set_rot_center</a> .				
Y	Like <a href="#">set_rot_center</a> .				
RTC4→RTC5	New command.				
<b>Version info</b>	–				
<b>References</b>	<a href="#">set_rot_center</a>				



Ctrl Command	<b>set_RTC4_mode</b>
Function	Sets <b>RTC4 Compatibility Mode</b> as the current <b>RTC5 DLL</b> operation mode.
Call	<code>set_RTC4_mode()</code>
Comments	<ul style="list-style-type: none"> <li>• <b>RTC4 Compatibility Mode</b> is an optional operation mode of the <b>RTC5 DLL</b>. It has been made available so that user programs written for the RTC4 can also be processed by the RTC5 (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 RTC5 commands. This user manual's list of commands, when applicable, notes such changes in the "RTC4→RTC5" row.</li> <li>• <b>RTC5 Standard Mode</b> is pre-defined as the default <b>RTC5 DLL</b> operation mode and can also be specified subsequently by <b>set_RTC5_mode</b>.</li> <li>• The current <b>RTC5 DLL</b> 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>• This command's scope is not board-specific, but rather global to the <b>RTC5 DLL</b> and all RTC5 Boards to which the user program has access rights.</li> <li>• The board-specific error variables <b>LastError</b> and <b>AccError</b> (see <b>Chapter 6.8 "Error Handling", page 119</b>) are neither generated nor altered by <b>set_RTC4_mode</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>get_RTC_mode</b> , <b>set_RTC5_mode</b>



<b>Ctrl Command</b>	<b>set_RTC5_mode</b>
<b>Function</b>	Sets <b>RTC5 Standard Mode</b> as the current <b>RTC5 DLL</b> operation mode (default setting).
<b>Call</b>	<code>set_RTC5_mode()</code>
<b>Comments</b>	<ul style="list-style-type: none"><li>The current <b>RTC5 DLL</b> operation mode can be queried by <a href="#">get_RTC_mode</a>.</li><li><b>set_RTC5_mode</b> is available even without explicit access rights to a particular RTC5 Board.</li><li><b>set_RTC5_mode</b> is not available as a multi-board command.</li><li><b>set_RTC5_mode</b> is not board-specific, but rather global to the <b>RTC5 DLL</b> and all RTC5 Boards to which the user program has access rights.</li><li>The board-specific error variables <code>LastError</code> and <code>AccError</code> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <b>set_RTC5_mode</b>.</li></ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">get_RTC_mode</a> , <a href="#">set_RTC4_mode</a>



<b>Ctrl Command</b>	<b>set_scale</b>
<b>Function</b>	Uses a specified scaling factor common to the x axis and y axis to define the scaling matrix $M_S$ for all subsequent coordinate transformations, see <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</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.                     = 1: The definition only affects the first scan head connector.                     = 2: The definition only affects the second scan head connector.                     = 0, 3: The definition affects <i>both</i> scan head connectors.                     Only the two least significant bits are evaluated.</p> <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 <a href="#">set_matrix( HeadNo = 0...3 )</a>.                     As an unsigned 32-bit value.</p>
<b>Comments</b>	• See <a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a> .
RTC4→RTC5	New command.
Version info	–
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→RTC5	New command.
Version info	–
References	<a href="#">set_scale</a>

<b>Delayed Short List Command</b>	<b>set_scanner_delays</b>						
Function	Sets the <b>Scanner Delays</b> .						
Call	<code>set_scanner_delays( Jump, Mark, Polygon )</code>						
Parameters	<table> <tr> <td>Jump</td> <td>Scanner delay of type: <b>Jump Delay</b>. As an unsigned 32-bit value. 1 bit equals 10 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</td> </tr> <tr> <td>Mark</td> <td>Scanner delay of type: <b>Mark Delay</b>. As an unsigned 32-bit value. 1 bit equals 10 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</td> </tr> <tr> <td>Polygon</td> <td>Scanner delay of type: <b>Polygon Delay</b>. As an unsigned 32-bit value. 1 bit equals 10 <math>\mu</math>s. Allowed value range: [0...(2<sup>32</sup>-1)].</td> </tr> </table>	Jump	Scanner delay of type: <b>Jump Delay</b> . As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].	Mark	Scanner delay of type: <b>Mark Delay</b> . As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].	Polygon	Scanner delay of type: <b>Polygon Delay</b> . As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].
Jump	Scanner delay of type: <b>Jump Delay</b> . As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].						
Mark	Scanner delay of type: <b>Mark Delay</b> . As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].						
Polygon	Scanner delay of type: <b>Polygon Delay</b> . As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>32</sup> -1)].						
Comments	<ul style="list-style-type: none"> <li>See also <b>Chapter 7.2.2 "Scanner Delays"</b>, page 135.</li> <li>Variable Delays for jumps and <b>Polylines</b> can be set by <b>set_delay_mode</b>.</li> <li>The specified delays are automatically adjusted by the RTC5 to avoid laser control errors (see <b>Section "Automatic Delay Adjustments"</b>, page 143).</li> <li>The default setting after <b>load_program_file</b> corresponds to <code>set_scanner_delays( 9, 6, 3 )</code>.</li> </ul>						
RTC4→RTC5	Unchanged functionality. In addition: increased value range.						
Version info	–						
References	<b>set_delay_mode</b> , <b>set_laser_delays</b>						

<b>Ctrl Command</b>	<b>set_serial</b>
Function	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.
Call	<code>set_serial( No )</code>
Parameters	No      Serial number. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].
Comments	<ul style="list-style-type: none"> <li><b>set_serial</b> is synonymous with <b>set_serial_step</b> with <b>Step</b> = 1 (see comments there).</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>	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> has been called with Mode M<sub>2</sub> = 1 (that is, automatic serial-number incrementing has been 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 <b>set_serial_step</b> usage, see <b>Chapter 7.5.2 "Marking Serial Numbers", page 196</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>	Like <b>set_serial_step</b> , but a list command.
<b>Call</b>	<code>set_serial_step_list( No, Step )</code>
<b>Parameters</b>	No      Like <b>set_serial_step</b> .
	Step      Like <b>set_serial_step</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <b>set_serial_step</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_serial_step</b> , <b>select_serial_set_list</b> , <b>get_list_serial</b> , <b>mark_serial</b> , <b>mark_serial_abs</b>



<b>Ctrl Command</b>	<b>set_sky_writing</b>
<b>Function</b>	Activates <b>Sky Writing Mode 1</b> and sets the corresponding parameters or switches off <b>Sky Writing</b> .
<b>Call</b>	<code>set_sky_writing( Timelag, LaserOnShift )</code>
<b>Parameters</b>	Timelag      Like <b>set_sky_writing_para</b> .
	LaserOnShift      Like <b>set_sky_writing_para</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_sky_writing</b> is identical to  <code>set_sky_writing_para( Timelag, LaserOnShift, Nprev, Npost )</code>  with <code>Nprev</code> = approx. <math>(0.15 \times \text{Timelag})</math> and <code>Npost</code> = approx. <math>(0.1 \times \text{Timelag})</math>.</li> <li>• See also information about <b>set_sky_writing_para</b> and <b>Chapter 7.2.4 "Sky Writing", page 149</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>set_sky_writing_para</b> , <b>set_sky_writing_list</b>

<b>Ctrl Command</b>	<b>set_sky_writing_limit</b>		
<b>Function</b>	Defines the limit for <b>Sky Writing</b> switching in <b>Sky Writing Mode 3</b> .		
<b>Call</b>	<code>set_sky_writing_limit( Limit )</code>		
<b>Parameters</b>	<table> <tr> <td>Limit</td> <td>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.</td> </tr> </table>	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.
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.		
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>set_sky_writing_limit</b> usage, see <a href="#">Chapter 7.2.4 "Sky Writing", page 149</a>.</li> <li>Limit is the cosine of the angular limit (for angular change between consecutive vectors or arcs within a <b>Polyline</b>) for which a <b>Sky Writing</b> motion should be performed.</li> <li>The initialized value (after program start) is Limit = 0 (angular limit = 90°).</li> </ul>		
RTC4→RTC5	New command.		
Version info	–		
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 <b>set_sky_writing_limit</b> , but a list command.
<b>Call</b>	<code>set_sky_writing_limit_list( Limit )</code>
<b>Parameters</b>	Limit Like <b>set_sky_writing_limit</b> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <b>set_sky_writing_limit</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">set_sky_writing_limit</a>

<b>Normal List Command</b>	<b>set_sky_writing_list</b>				
<b>Function</b>	Like <b>set_sky_writing</b> , but a list command.				
<b>Call</b>	<code>set_sky_writing_list( Timelag, LaserOnShift )</code>				
<b>Parameters</b>	<table> <tr> <td>Timelag</td> <td>Like <b>set_sky_writing_para</b>.</td> </tr> <tr> <td>LaserOnShift</td> <td>Like <b>set_sky_writing_para</b>.</td> </tr> </table>	Timelag	Like <b>set_sky_writing_para</b> .	LaserOnShift	Like <b>set_sky_writing_para</b> .
Timelag	Like <b>set_sky_writing_para</b> .				
LaserOnShift	Like <b>set_sky_writing_para</b> .				
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <b>set_sky_writing</b>, <b>set_sky_writing_para</b> and <b>set_sky_writing_para_list</b>.</li> </ul>				
RTC4→RTC5	New command.				
Version info	–				
References	<a href="#">set_sky_writing</a> , <a href="#">set_sky_writing_para</a> , <a href="#">set_sky_writing_para_list</a> .				





Normal List Command	<code>set_sky_writing_mode_list</code>
Function	Like <code>set_sky_writing_mode</code> , but a list command.
Call	<code>set_sky_writing_mode_list( Mode )</code>
Parameters	Mode      Like <code>set_sky_writing_mode</code> .
Comments	<ul style="list-style-type: none"> <li>By <code>set_sky_writing_mode_list</code>, <b>Sky Writing Mode 2</b> and <b>Sky Writing Mode 3</b> 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 <b>Sky Writing Mode 1</b> and the next is started with it.</li> <li>Deactivation of <b>Sky Writing Mode 1</b> by <code>set_sky_writing_mode_list</code> (switching from Mode = 1 to 0) results in the addition of a <b>Mark Delay</b> defined prior to activation of <b>Sky Writing</b> – provided that no other delay is in effect (for example, a <b>Jump Delay</b>).</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<code>set_sky_writing_mode</code>

Ctrl Command	set_sky_writing_para																		
Function	Activates <b>Sky Writing Mode 1</b> and sets the corresponding parameters or switches <b>Sky Writing</b> off.																		
Call	set_sky_writing_para( Timelag, LaserOnShift, Nprev, Npost )																		
Parameters	<p>Timelag      <b>Sky Writing</b> parameter.  <math>1.0</math> equals <math>1 \mu\text{s}</math>.  The <b>Timelag</b> value is used with an accuracy of <math>1/64 \mu\text{s}</math>.  As a 64-bit IEEE floating point value.</p> <p><math>\geq 1/4</math>: <b>Sky Writing Mode 1</b> is activated.  From <math>1/8</math> to <math>1/4</math>, <b>Sky Writing</b> is activated,  but <b>Timelag</b> = 0 is internally applied.</p> <p><math>&lt; 1/8</math>: <b>Sky Writing</b> is deactivated.</p> <p>LaserOnShift      Shift (for positive values: delay) of the point of time, where the are switched on (see <a href="#">Figure 53</a>).  1 Bit equals <math>0.5 \mu\text{s}</math>.  As a signed 32-bit value.  Negative values smaller than the complete run-in phase are clipped at runtime, therefore the following applies automatically:</p> <table> <thead> <tr> <th>LaserOnShift</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td><math>\geq \text{ca. } (-40) \times \text{Nprev}</math></td> <td>1</td> </tr> <tr> <td><math>\geq \text{ca. } (-20) \times \text{Nprev}</math></td> <td>2 or 3</td> </tr> </tbody> </table> <p>Nprev      Defines the duration of the run-in phase (see <a href="#">Figure 53</a>).  1 bit equals <math>10 \mu\text{s}</math>.  As an unsigned 32-bit value.  Allowed value range: <math>0 \leq \text{Nprev} \leq (2^{32}-1)</math>.</p> <table> <thead> <tr> <th>Run-in duration [<math>\mu\text{s}</math>]</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td><math>20 \times \text{Nprev}</math></td> <td>1</td> </tr> <tr> <td><math>10 \times \text{Nprev}</math></td> <td>2 or 3</td> </tr> </tbody> </table> <p>Npost      Defines the duration of the run-out phase (see <a href="#">Figure 53</a>).  1 bit equals <math>10 \mu\text{s}</math>.  As an unsigned 32-bit value.  Allowed value range: <math>0 \leq \text{Npost} \leq (2^{32}-1)</math>.</p> <table> <thead> <tr> <th>Run-out duration [<math>\mu\text{s}</math>]</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td><math>20 \times \text{Npost}</math></td> <td>1</td> </tr> <tr> <td><math>10 \times \text{Npost}</math></td> <td>2 or 3</td> </tr> </tbody> </table>	LaserOnShift	Mode	$\geq \text{ca. } (-40) \times \text{Nprev}$	1	$\geq \text{ca. } (-20) \times \text{Nprev}$	2 or 3	Run-in duration [ $\mu\text{s}$ ]	Mode	$20 \times \text{Nprev}$	1	$10 \times \text{Nprev}$	2 or 3	Run-out duration [ $\mu\text{s}$ ]	Mode	$20 \times \text{Npost}$	1	$10 \times \text{Npost}$	2 or 3
LaserOnShift	Mode																		
$\geq \text{ca. } (-40) \times \text{Nprev}$	1																		
$\geq \text{ca. } (-20) \times \text{Nprev}$	2 or 3																		
Run-in duration [ $\mu\text{s}$ ]	Mode																		
$20 \times \text{Nprev}$	1																		
$10 \times \text{Nprev}$	2 or 3																		
Run-out duration [ $\mu\text{s}$ ]	Mode																		
$20 \times \text{Npost}$	1																		
$10 \times \text{Npost}$	2 or 3																		
Comments	<ul style="list-style-type: none"> <li>For information on <b>Sky Writing</b> mode and a description of the parameters, see <a href="#">Chapter 7.2.4 "Sky Writing", page 149</a>.</li> <li>If <math>\text{Nprev} \geq 65,535</math>, then <b>set_sky_writing_para</b> behaves similarly to <b>set_sky_writing</b>: <b>Nprev</b> is set to a value of approx. <math>(0.15 \times \text{Timelag})</math>.</li> <li>If <math>\text{Npost} \geq 65,535</math>, then <b>set_sky_writing_para</b> behaves similarly to <b>set_sky_writing</b>: <b>Npost</b> is set to a value of approx. <math>(0.1 \times \text{Timelag})</math>.</li> <li><b>set_sky_writing_para</b> cannot be used to switch back and forth between <b>Sky Writing Mode 1</b>, <b>Sky Writing Mode 2</b> and <b>Sky Writing Mode 3</b> (the proper command for this is <b>set_sky_writing_mode</b>).</li> </ul>																		



Ctrl Command	set_sky_writing_para
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 <math>\text{Timelag} \geq 1/8</math>, <b>Sky Writing Mode 1</b> is activated, if <b>Sky Writing</b> has not been active before. <b>Sky Writing Mode 2</b> or <b>Sky Writing Mode 3</b> remain, but the next <b>Mark command</b> is always started in <b>Sky Writing Mode 1</b></li> <li>With <math>\text{Timelag} &lt; 1/8</math>, <b>Sky Writing</b> 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 <b>Sky Writing Mode 2</b> or <b>Sky Writing Mode 3</b> is active, then <code>set_sky_writing_para</code> is not executed (<code>get_last_error</code> return code <code>RTC5_BUSY</code>)</li> <li>If no <b>Sky Writing</b> or <b>Sky Writing Mode 1</b> 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 <b>Sky Writing Mode 1</b>, an already started <b>Mark command</b> is executed with the previous parameters and ended with <b>Sky Writing Mode 1</b>.</li> <li>In <b>Sky Writing Mode 1</b>, the next <b>Mark command</b> is started in <b>Sky Writing Mode 1</b>, if <math>\text{Timelag} \geq 1/8</math>. Otherwise, it is terminated and a <b>Mark Delay</b> is inserted in <b>Sky Writing Mode 1</b>.</li> </ul> </li> <li>For <b>Sky Writing Mode 2</b> or <b>Sky Writing Mode 3</b>, <math>\text{Nprev} = 0</math> and/or <math>\text{Npost} = 0</math> automatically get(s) internally corrected to 1 (this is not treated as an error). If you subsequently activate <b>Sky Writing Mode 1</b> (by <code>set_sky_writing_mode</code>), then <math>\text{Nprev} = 0</math> and/or <math>\text{Npost} = 0</math> is/are reused.</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
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 <b>set_sky_writing_para</b> , but a list command.
<b>Call</b>	set_sky_writing_para_list( Timelag, LaserOnShift, Nprev, Npost )
<b>Parameters</b>	Timelag      Like <b>set_sky_writing_para</b> .
	LaserOnShift      Like <b>set_sky_writing_para</b> .
	Nprev      Like <b>set_sky_writing_para</b> .
	Npost      Like <b>set_sky_writing_para</b> .
<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 <b>Sky Writing Mode 2</b> or <b>Sky Writing Mode 3</b> is active. Here, an already-begun <b>Mark command</b> is finished with <b>Sky Writing Mode 1</b> and the next is started with it.</li> <li>• By <b>set_sky_writing_para_list</b>, you cannot switch from <b>Sky Writing Mode 2</b> or <b>Sky Writing Mode 3</b> to <b>Sky Writing Mode 1</b> permanently. For this, use <b>set_sky_writing_mode_list</b>.</li> <li>• Deactivation of <b>Sky Writing Mode 1</b>, <b>Sky Writing Mode 2</b> or <b>Sky Writing Mode 3</b> by <b>set_sky_writing_para_list</b> results in the addition of a <b>Mark Delay</b> defined prior to activation of <b>Sky Writing</b> – provided that no other delay is in effect (for example, a <b>Jump Delay</b>).</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<b>set_sky_writing_para</b>

<b>Ctrl Command</b>	<b>set_softstart_level</b>		
<b>Function</b>	Sets the softstart values.		
<b>Call</b>	set_softstart_level( Index, Level )		
<b>Parameters</b>	Parameter	Allowed Values	Description
	Index	0...1023	Index number of the softstart table value. As an unsigned 32-bit value.
	Level	0... $2^{12}-1$ 0...( $2^{32}-1$ )	For <b>Softstart Mode</b> = 1, 2, 11 and 12 (see <a href="#">set_softstart_mode</a> ). Value 0 corresponds to an analog voltage value of 0 V. Value $2^{12}-1$ corresponds to 10 V. For <b>Softstart Mode</b> = 3 and 13 (see <a href="#">set_softstart_mode</a> ). 1 bit corresponds to a pulse length of $1/64 \mu\text{s}$ (in <a href="#">RTC5 Standard Mode</a> ) or $1/8 \mu\text{s}$ (in <a href="#">RTC4 Compatibility Mode</a> ). As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_softstart_level</b> defines the <code>Level</code> value for the pulse number <code>Index</code>. <b>set_softstart_level</b> must be called separately for each individual <code>Level</code> value.</li> <li>• <code>Index</code> must be in the range [0...1023], otherwise <b>set_softstart_level</b> is not executed (<a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>• If <code>Index</code> <math>\geq</math> <code>Number</code> (<code>Number</code> is defined by <a href="#">set_softstart_mode</a>) then <b>set_softstart_level</b> is executed but the defined value is not used during softstart.</li> <li>• Be sure to set as many softstart values as you defined by <a href="#">set_softstart_mode</a> (unspecified softstart values have undefined contents).</li> <li>• The <b>Softstart Mode</b> is enabled by <a href="#">set_softstart_mode</a>.</li> <li>• For <b>Softstart Modes</b> 1, 2, 11 or 12, <code>Level</code> is restricted to 12 bits; higher bits are ignored (see also <a href="#">write_da_x</a> or <a href="#">write_da_x_list</a>).</li> <li>• If <b>Softstart Mode</b> = 3 or 13 and the specified softstart pulse length is larger than the laser signal period duration specified by <a href="#">set_laser_pulses</a> or <a href="#">set_laser_timing</a> (<math>2 \times \text{HalfPeriod}</math>), then the laser remains continuously on.</li> <li>• Particularly if softstart values are to be outputted as analog voltage values (<b>Softstart Mode</b> = 1, 2, 11 or 12), the <a href="#">set_softstart_mode</a> command must be called before first-time use of the <b>set_softstart_level</b> command; otherwise the specified value is not correctly converted to an analog value.</li> <li>• Also observe the note in <a href="#">Chapter 7.4.7 "Softstart Mode", page 184</a>.</li> <li>• If the "freely definable wobble shape" wobble mode is activated, then this command has no effect, see <a href="#">Chapter 8.4 "Wobble Mode", page 217</a>.</li> </ul>		



<b>Ctrl Command</b>	<b>set_softstart_level</b>
RTC4→RTC5	<p>Basically unchanged functionality. In addition: increased value range.</p> <p>In <b>RTC4 Compatibility Mode</b>, analog voltage levels are internally multiplied by 4 and pulse length values by 8. The allowed range of values for <code>Level</code> is correspondingly smaller. Due to this difference in the handling of analog voltage and pulse length values, <b>set_softstart_mode</b> must always be called before <b>set_softstart_level</b>.</p>
Version info	–
References	<b>set_softstart_mode, set_softstart_mode_list, set_softstart_level_list</b>

<b>Normal List Command</b>	<b>set_softstart_level_list</b>								
Function	Like <b>set_softstart_level</b> , but a list command.								
Call	<code>set_softstart_level_list( Index, Level1, Level2, Level3 )</code>								
Parameters	<table> <tr> <td>Index</td> <td>Like <b>set_softstart_level</b>.</td> </tr> <tr> <td>Level1</td> <td>Like <b>set_softstart_level</b>.</td> </tr> <tr> <td>Level2</td> <td>Like <b>set_softstart_level</b>.</td> </tr> <tr> <td>Level3</td> <td>Like <b>set_softstart_level</b>.</td> </tr> </table>	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> .
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> .								
Comments	<ul style="list-style-type: none"> <li>• <b>set_softstart_level_list</b> defines three <code>Level</code> values for the pulses number <code>Index</code> through <code>Index+2</code>. <b>set_softstart_level_list</b> must be called separately for each individual <code>Level1/2/3</code> value-triple.</li> <li>• <code>Index</code> must be in the range [0...1023], otherwise <b>set_softstart_level_list</b> is, already during loading, replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>• Softstart does not use level values where the index count exceeds <code>Number</code> (<code>Number</code> is specified by <b>set_softstart_mode</b>).</li> <li>• Otherwise, <b>set_softstart_level_list</b> is identical to the control command <b>set_softstart_level</b> (see also the comments there).</li> <li>• If the “freely defined wobble shape” wobble mode is activated, then this command has no effect, see <b>Chapter 8.4 “Wobble Mode”, page 217</b>.</li> </ul>								
RTC4→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>: see <b>set_softstart_level</b>.</p>								
Version info	–								
References	<b>set_softstart_level, set_softstart_mode, set_softstart_mode_list</b>								

<b>Ctrl Command</b>	<b>set_softstart_mode</b>	
<b>Function</b>	Switches <b>Softstart Mode</b> on or off.	
<b>Call</b>	set_softstart_mode( Mode, Number, Delay )	
<b>Parameters</b>	Mode	= 0 Disables the <b>Softstart Mode</b> , but does not remove previously loaded softstart values.  = 1, 11 The softstart values defined by <b>set_softstart_level</b> are outputted as analog voltage values at the <b>ANALOG OUT1</b> output port.  = 2, 12 The softstart values defined by <b>set_softstart_level</b> are outputted as analog voltage values at the <b>ANALOG OUT2</b> output port.  = 3, 13 The softstart values defined by <b>set_softstart_level</b> specify the pulselenghts of the first <b>Number</b> laser signal pulses at the <b>LASER1</b> port (neither in <b>Laser Mode 4</b> nor in <b>Laser Mode 6</b> ).  As an unsigned 32-bit value.
	Number	1...1024 Number of softstart values to be transmitted. As an unsigned 32-bit value.
	Delay	0...(2 <sup>32</sup> –1). 1 bit equals 10 $\mu$ s. Defines the time period for which the laser must have been switched off before softstart is activated at the next switch-on. As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>set_softstart_mode</b> must be called <i>before</i> first-time use of the <b>set_softstart_level</b> command, particularly if softstart values are to be outputted as analog voltage values (default setting for softstart value handling: Mode = 3).</li> <li>• The individual softstart values Level(0) – Level(Number – 1) must be provided individually by separate calls of <b>set_softstart_level</b>. Indices for which no <b>Level</b> value has been supplied are undefined.</li> <li>• If the values for <b>Mode</b> or <b>Number</b> are invalid, then <b>Softstart Mode</b> is switched off.</li> <li>• When <b>Softstart Mode</b> is switched off, the already transmitted values are not deleted and can be further used after a renewed switch-on. Here, you should not unintentionally switch between analog voltage values (<b>Mode</b> = 1, 2, 11 or 12) and pulselenghts (<b>Mode</b> = 3 or 13). In contrast, you can switch at any time between <b>ANALOG OUT1</b> (<b>Mode</b> = 1 or 11) and <b>ANALOG OUT2</b> (<b>Mode</b> = 2 or 12).</li> <li>• The <b>Softstart Mode</b> is incompatible with the <b>Pixel mode</b> (see <b>Chapter 8.7 "Pixel Output Mode"</b>, page 244). <b>set_softstart_mode</b> is executed, but softstart is not executed while <b>Pixel mode</b> is operational.</li> <li>• The <b>Softstart Mode</b> should <i>not</i> be used in conjunction with "Automatic Laser Control" (see <b>page 187</b>) if "Automatic Laser Control" readjusts the 12-bit output values at the <b>ANALOG OUT1</b> or <b>ANALOG OUT2</b> output ports or pulse lengths (<b>PulseLength</b>) or output periods (<b>HalfPeriod</b>) of the laser signals <b>LASER1</b> and <b>LASER2</b>. <b>Softstart Mode</b> cannot be combined with "freely definable wobble shapes". If one is defined and activated, <b>set_softstart_mode</b> has no effect.</li> <li>• Observe the notes in <b>Chapter 7.4.7 "Softstart Mode"</b>, page 184.</li> </ul>	



<b>Ctrl Command</b>	<b>set_softstart_mode</b>
RTC4→RTC5	<p>Basically unchanged functionality. In addition: increased value range.</p> <p>Instead of acquiring analog softstart values with the trailing edge of the laser signal pulse, a 180° phase-shifted laser signal can be set as a trigger signal for realizing phase-shifted data acquisition with the RTC5 (see notes in <a href="#">Chapter 7.4.7 "Softstart Mode", page 184</a>).</p> <p>An index shift no longer exists under any circumstances.</p>
Version info	–
References	<a href="#">set_softstart_level</a> , <a href="#">set_softstart_level_list</a> , <a href="#">set_softstart_mode_list</a>

<b>Variable List Command</b>	<b>set_softstart_mode_list</b>						
Function	Like <a href="#">set_softstart_mode</a> , but a list command.						
Call	<code>set_softstart_mode_list( Mode, Number, Delay )</code>						
Parameters	<table> <tr> <td>Mode</td> <td>Like <a href="#">set_softstart_mode</a>.</td> </tr> <tr> <td>Number</td> <td>Like <a href="#">set_softstart_mode</a>.</td> </tr> <tr> <td>Delay</td> <td>Like <a href="#">set_softstart_mode</a>.</td> </tr> </table>	Mode	Like <a href="#">set_softstart_mode</a> .	Number	Like <a href="#">set_softstart_mode</a> .	Delay	Like <a href="#">set_softstart_mode</a> .
Mode	Like <a href="#">set_softstart_mode</a> .						
Number	Like <a href="#">set_softstart_mode</a> .						
Delay	Like <a href="#">set_softstart_mode</a> .						
Comments	<ul style="list-style-type: none"> <li>• <a href="#">set_softstart_mode_list</a> switches off the <a href="#">Signals for "Laser Active" Operation</a> by executing <a href="#">list_nop</a> and waits until the <a href="#">LaserOff Delay</a> has expired.</li> <li>• Otherwise, <a href="#">set_softstart_mode_list</a> is identical to the control command <a href="#">set_softstart_mode</a> (see also the comments there).</li> </ul>						
RTC4→RTC5	New command.						
Version info	–						
References	<a href="#">set_softstart_mode</a> , <a href="#">set_softstart_level</a> , <a href="#">set_softstart_level_list</a>						

<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 <b>Laser Mode 4</b> and <b>Laser Mode 6</b> – 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><b>HalfPeriod</b>      <i>Half of the standby output period.</i>                              As an unsigned 32-bit value.                              1 bit equals 1/64 <math>\mu</math>s.                              Allowed value range: [0...+(2<sup>32</sup>-1)].</p> <p><b>PulseLength</b>      <i>Pulse length of the standby pulses.</i>                              As an unsigned 32-bit value.                              1 bit equals 1/64 <math>\mu</math>s.                              Allowed value range: [0...(2<sup>32</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>After a <b>Hardware reset</b>, (and after <b>load_program_file</b>), the LASER1, LASER2 and LASERON ports must be first-time-activated with <b>set_laser_control</b> before standby pulses can be activated by <b>set_standby</b> or <b>set_standby_list</b> (see <b>Chapter 7.4 "Laser Control", page 173</b>). At the same time, the signal level of the standby pulses can be defined with <b>set_laser_control</b>.</li> <li>The standby pulses are available in <i>all</i> laser modes (<b>CO<sub>2</sub> Mode</b>, YAG Mode 1, YAG Mode 2, YAG Mode 3, <b>Laser Mode 4</b>, YAG Mode 5, <b>Laser Mode 6</b>). They can be deactivated (turned off) by setting the standby pulse length and/or the standby output period to zero (default).</li> <li>With <b>HalfPeriod</b>, <i>half</i> the standby output period must be specified, see <b>Figure 59</b> and <b>Figure 61</b>.</li> <li>If the <b>Softstart Mode</b> is used (see <b>Chapter 7.4.7 "Softstart Mode", page 184</b>), <b>HalfPeriod</b> should not be smaller than 104 (this is not automatically checked). This value corresponds to a laser pulse frequency of approx. 308 kHz.</li> <li>If <b>PulseLength</b> is larger than the output period (2 <math>\times</math> <b>HalfPeriod</b>), the laser is permanently on.</li> <li>The laser control mode has to be set with <b>set_laser_mode</b> (see also <b>Chapter 7.4 "Laser Control", page 173</b>). To set the active output period and pulse length for the "laser active" laser control signals (beside for <b>Laser Mode 4</b> and <b>Laser Mode 6</b>), use <b>set_laser_pulses_ctrl</b>, <b>set_laser_pulses</b> or <b>set_laser_timing</b>.</li> </ul>
<b>RTC4→RTC5</b>	Basically unchanged functionality. In addition: increased value range. However: In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for <b>HalfPeriod</b> and <b>PulseLength</b> by 8. The allowed value ranges decrease accordingly. The smallest reasonable value for <b>HalfPeriod</b> with <b>Softstart Mode</b> is then 13.
<b>Version info</b>	–
<b>References</b>	<b>set_standby_list</b> , <b>get_standby</b>



<b>Delayed Short List Command</b>	<b>set_standby_list</b>
Function	Like <b>set_standby</b> , but a list command.
Call	<code>set_standby_list( HalfPeriod, PulseLength )</code>
Parameters	HalfPeriod      Like <b>set_standby</b> . PulseLength      Like <b>set_standby</b> .
RTC4→RTC5	Siehe <b>set_standby</b>
Version info	–
References	<b>set_standby</b>



<b>Ctrl Command</b>	<b>set_start_list</b>
<b>Function</b>	Opens the list memory area for writing of list commands and sets the input pointer to the start of the desired list ("List 1" or "List 2").
<b>Call</b>	<code>set_start_list( ListNo )</code>
<b>Parameters</b>	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"].
<b>Comments</b>	<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, the commands <code>set_start_list_1</code> and <code>set_start_list_2</code> (with no parameter) can be used.</li> </ul>
RTC4→RTC5	Unchanged functionality.
<b>Version info</b>	–
<b>References</b>	<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>	<code>set_start_list_1()</code>
RTC4→RTC5	Unchanged functionality.
<b>Version info</b>	–
<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>	<code>set_start_list_2()</code>
RTC4→RTC5	Unchanged functionality.
<b>Version info</b>	–
<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 area for writing of list commands and sets the input pointer to the specified <i>relative</i> position in the specified list ("List 1" or "List 2").
<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>20</sup>-1)].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>The next list command is stored at the specified address and all further list commands at the subsequent addresses in the selected list.</li> <li>The specified 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 94</a> and <a href="#">load_list</a>.</li> <li>The input pointer <i>cannot</i> be set to the protected 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 <code>config_list( Mem1, -1 )</code>. After writing, configuration of the area's protection should be restored: <code>config_list( Mem1, Mem2 )</code>.</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 <code>Pos</code> 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 <code>Pos</code> is set to 0.</li> <li>The status values of the selected list (see also <a href="#">read_status</a>) are set as follows:  <code>LOAD</code> list status set, <code>READY</code> list status reset, <code>USED</code> list status reset. The <code>LOAD</code> list status of the other corresponding list is reset.</li> <li><code>set_start_list_pos</code> triggers a flush of the buffered list input, see <a href="#">Chapter 6.4.1 "Loading Lists", page 94</a>.</li> <li><code>set_start_list_pos</code> also covers the specialized variants <code>set_start_list_1</code>, <code>set_start_list_2</code>, <code>set_start_list</code> and <code>set_input_pointer</code>.</li> <li><i>Notice! If the end of the respective list memory area is reached, the list input pointer is automatically reset to the start of the same list memory area.</i>  <i>Make sure not to overwrite any commands still needed by your user program.</i></li> </ul>
RTC4→RTC5	New command.
Version info	–
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>20</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>20</sup>-1)</code>, then <b>set_sub_pointer</b> is not executed (<b>get_last_error</b> return code <code>RTC5_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>.</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 has been finalized by <b>list_return</b> and does not contain <b>set_end_of_list</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 ordering applies as for <b>load_text_table</b>:      = 0...9: Digits for marking the time and date [0...9].      = 10...21: Months [January...December].      = 22...28: Days-of-the-week [Sunday...Saturday].      = 29: Blank character for marking serial numbers.      = 30...39: Digits for marking serial numbers [0...9].      = 40: Text for "a.m.".      = 41: Text for "p.m.".</p> <p>Pos      Absolute start address.      As an unsigned 32-bit value.      Allowed value range: [0...(2<sup>20</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 109</a>).</li> <li>If <code>Index &gt; 41</code> and/or <code>Pos &gt; (2<sup>20</sup>-1)</code>, then <b>set_text_table_pointer</b> is not executed (<a href="#">get_last_error</a> return code <code>RTC5_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 <a href="#">save_disk/load_disk</a> 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 <a href="#">copy_dst_src</a> 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 <a href="#">get_input_pointer</a> 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 <a href="#">save_disk/load_disk</a>.</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 has been finalized by <a href="#">list_return</a> and does not contain <a href="#">set_end_of_list</a>.</li> </ul>
RTC4→RTC5	New command. <b>set_text_table_pointer</b> is synonymous with <b>set_char_table</b> , which is available for the RTC SCANalone Board (standalone version of the RTC4 board).
Version info	–
References	<a href="#">load_text_table</a> , <a href="#">mark_date</a> , <a href="#">mark_serial</a> , <a href="#">mark_time</a>



<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>	<table> <tr> <td>Period</td> <td>Measurement period (time interval between 2 data pair recordings). As an unsigned 32-bit value. 1 bit equals 10 <math>\mu</math>s. Allowed value range: [0...(2<sup>31</sup>-1)].</td> </tr> <tr> <td>Signal1</td> <td>Signal type for measurement channel 1. As an unsigned 32-bit value. Allowed value range: see below.</td> </tr> <tr> <td>0</td> <td>LASERON signal 1 = laser control signal on. 0 = laser control signal off.</td> </tr> <tr> <td>1</td> <td>StatusAX Status signal of x axis, first scan head connector.</td> </tr> <tr> <td>2</td> <td>StatusAY Status signal of y axis, first scan head connector.</td> </tr> <tr> <td>3</td> <td>Reserved.</td> </tr> <tr> <td>4</td> <td>StatusBX Status signal of x axis, second scan head connector.</td> </tr> <tr> <td>5</td> <td>StatusBY Status signal of y axis, second scan head connector.</td> </tr> <tr> <td>6</td> <td>Reserved.</td> </tr> <tr> <td>7</td> <td>SampleX Cartesian control value for the x axis (= up to #4 in <a href="#">7.3.6, page 170</a>).</td> </tr> <tr> <td>8</td> <td>SampleY Cartesian control value for the y axis (= up to #4 in <a href="#">7.3.6, page 170</a>).</td> </tr> <tr> <td>9</td> <td>SampleZ Cartesian control value for the z axis (= up to #4 in <a href="#">7.3.6, page 170</a>).</td> </tr> </table>	Period	Measurement period (time interval between 2 data pair recordings). As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>31</sup> -1)].	Signal1	Signal type for measurement channel 1. As an unsigned 32-bit value. Allowed value range: see below.	0	LASERON signal 1 = laser control signal on. 0 = laser control signal off.	1	StatusAX Status signal of x axis, first scan head connector.	2	StatusAY Status signal of y axis, first scan head connector.	3	Reserved.	4	StatusBX Status signal of x axis, second scan head connector.	5	StatusBY Status signal of y axis, second scan head connector.	6	Reserved.	7	SampleX Cartesian control value for the x axis (= up to #4 in <a href="#">7.3.6, page 170</a> ).	8	SampleY Cartesian control value for the y axis (= up to #4 in <a href="#">7.3.6, page 170</a> ).	9	SampleZ Cartesian control value for the z axis (= up to #4 in <a href="#">7.3.6, page 170</a> ).
Period	Measurement period (time interval between 2 data pair recordings). As an unsigned 32-bit value. 1 bit equals 10 $\mu$ s. Allowed value range: [0...(2 <sup>31</sup> -1)].																								
Signal1	Signal type for measurement channel 1. As an unsigned 32-bit value. Allowed value range: see below.																								
0	LASERON signal 1 = laser control signal on. 0 = laser control signal off.																								
1	StatusAX Status signal of x axis, first scan head connector.																								
2	StatusAY Status signal of y axis, first scan head connector.																								
3	Reserved.																								
4	StatusBX Status signal of x axis, second scan head connector.																								
5	StatusBY Status signal of y axis, second scan head connector.																								
6	Reserved.																								
7	SampleX Cartesian control value for the x axis (= up to #4 in <a href="#">7.3.6, page 170</a> ).																								
8	SampleY Cartesian control value for the y axis (= up to #4 in <a href="#">7.3.6, page 170</a> ).																								
9	SampleZ Cartesian control value for the z axis (= up to #4 in <a href="#">7.3.6, page 170</a> ).																								

<b>Delayed Short List Command</b>	<b>set_trigger</b>
Parameters (cont'd)	<p>Signal 10 (cont'd) SampleAX_Corr          Korrigierter Ansteuerwert für die x axis, erster scan head-Anschluss.          (= bis #7 in 7.3.6, page 170).</p> <p>11 SampleAY_Corr          Korrigierter Ansteuerwert für die y axis, erster scan head-Anschluss          (= bis #7 in 7.3.6, page 170).</p> <p>12 SampleAZ_Corr          Korrigierter Ansteuerwert für die z axis, wenn xy am ersten scan head-Anschluss angeschlossen sind          (= bis #10 in 7.3.6, page 170).          Identisch zum tatsächlichen Ausgabewert für die z axis.</p> <p>13 SampleBX_Corr          Korrigierter Ansteuerwert für die x axis, zweiter scan head-Anschluss          (= bis #7 in 7.3.6, page 170).</p> <p>14 SampleBY_Corr          Korrigierter Ansteuerwert für die y axis, zweiter scan head-Anschluss          (= bis #7 in 7.3.6, page 170).</p> <p>15 SampleBZ_Corr          Korrigierter Ansteuerwert für die z axis, wenn xy am zweiten scan head-Anschluss angeschlossen sind          (= bis #10 in 7.3.6, page 170).          Identisch zum tatsächlichen Ausgabewert für die z axis.</p> <p>16 StatusAX+LASERON signal  <i>StatusAX</i> wenn laser control signal an.          –524,288 wenn laser control signal aus.</p> <p>17 StatusAY+LASERON signal  <i>StatusAY</i> wenn laser control signal an.          –524,288 wenn laser control signal aus.</p> <p>18 StatusBX+LASERON signal  <i>StatusBX</i> wenn laser control signal an.          –524,288 wenn laser control signal aus.</p> <p>19 StatusBY+LASERON signal  <i>StatusBY</i> wenn laser control signal an.          –524,288 wenn laser control signal aus.</p>

<b>Delayed Short List Command</b>	<b>set_trigger</b>		
<b>Parameters (cont'd)</b>	Signal1 (cont'd)	20	SampleAX_Out Tatsächlicher Ausgabewert für die x axis, erster scan head-Anschluss (= bis #11 in 7.3.6, page 170). Nicht für z axisn-Ausgabewerte verwendbar.
		21	SampleAY_Out Tatsächlicher Ausgabewert für die y axis, erster scan head-Anschluss (= bis #11 in 7.3.6, page 170). Nicht für z axisn-Ausgabewerte verwendbar.
		22	SampleBX_Out Tatsächlicher Ausgabewert für die x axis, zweiter scan head-Anschluss (= bis #11 in 7.3.6, page 170). Nicht für z axisn-Ausgabewerte verwendbar.
		23	SampleBY_Out Tatsächlicher Ausgabewert für die y axis, zweiter scan head-Anschluss (= bis #11 in 7.3.6, page 170). Nicht für z axisn-Ausgabewerte verwendbar.
		24	Lasersteuerparameter der "Automatischen Laser-Steuerung" Siehe <b>set_auto_laser_control</b> .
		25	SampleAX_Trans Transformierter Ansteuerwert für die x axis, erster scan head-Anschluss (= bis #6 in 7.3.6, page 170).
		26	SampleAY_Trans Transformierter Ansteuerwert für die y axis, erster scan head-Anschluss (= bis #6 in 7.3.6, page 170).
		27	SampleAZ_Trans Transformierter Ansteuerwert für die z axis, wenn xy am ersten scan head-Anschluss angeschlossen sind (= bis #6 in 7.3.6, page 170).
		28	SampleBX_Trans Transformierter Ansteuerwert für die x axis, zweiter scan head-Anschluss (= bis #6 in 7.3.6, page 170).
		29	SampleBY_Trans Transformierter Ansteuerwert für die y axis, zweiter scan head-Anschluss (= bis #6 in 7.3.6, page 170).
		30	SampleBZ_Trans Transformierter Ansteuerwert für die z axis, wenn xy am zweiten scan head-Anschluss angeschlossen sind (= bis #6 in 7.3.6, page 170).



<b>Delayed Short List Command</b>	<b>set_trigger</b>
Parameters (cont'd)	<p>Signal1 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 <a href="#">ANALOG OUT1</a> output port  See <a href="#">set_vector_control</a> and <a href="#">Chapter 9.1.4 "12-Bit Analog Output Port 1 and 2"</a>, page 259.</p> <p>34      12-bit output value at the <a href="#">ANALOG OUT2</a> output port  See <a href="#">set_vector_control</a> and <a href="#">Chapter 9.1.4 "12-Bit Analog Output Port 1 and 2"</a>, page 259.</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>, page 258.</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>, page 259.</p> <p>37      Pulse length (PulseLength) of the <a href="#">Laser Control Signals LASER1</a> and <a href="#">LASER2</a>  See <a href="#">set_vector_control</a>.</p> <p>38      Output period (HalfPeriod) of the <a href="#">Laser Control Signals LASER1</a> and <a href="#">LASER2</a>  See <a href="#">set_vector_control</a>.</p> <p>39      FreeVariable0</p> <p>40      FreeVariable1</p> <p>41      FreeVariable2</p> <p>42      FreeVariable3</p>



<b>Delayed Short List Command</b>	<b>set_trigger</b>
Parameters (cont'd)	<p>Signal1 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 port (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 32-bit "Timestamp Counter" See <a href="#">Chapter 8.12 "Time Measurements", page 257</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 217</a>.</p> <p>54 ReadAnalogIn See <a href="#">read_analog_in</a>.</p>
	Signal2 Like (analogously) 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> (except for Period = 0) is replaced with a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>• After <b>set_trigger</b> with Period &gt; 0, a data pair is stored immediately and subsequently at intervals defined by the specified measurement period.</li> <li>• The measurement value recording ends automatically with 2<sup>20</sup> data pairs. 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> <li>• If <b>set_trigger</b> has been preceded by a <b>load_correction_file</b> which loaded a file as correction table №=3 or №=4, then the memory area's upper half reserved for data recording by <b>set_trigger</b> is already occupied. Then, only 2<sup>19</sup> data pairs are recordable with <b>set_trigger</b> and measurement value recording automatically terminates at this value. If loading of correction table №=3 or №=4 occurs after more than 2<sup>19</sup> data pairs had been recorded by <b>set_trigger</b>, then the values beyond 2<sup>19</sup> are no longer available. <b>load_program_file</b> reestablishes the original state of up to 2<sup>20</sup> possible measurement pairs. See also <b>number_of_correction_tables</b>( 2 ).</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 reset by <b>set_trigger</b>( Period = 0 ), <b>set_trigger4</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 RTC5 by <b>set_trigger</b> (Signal1, Signal2 = 1-23, 25-30) are always in the RTC5 20-bit range, even if the <b>RTC5 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. It is generally recommended to end the measurement explicitly before reading out the data. The measured values are transferred to the PC by <b>get_waveform</b> as 32-bit values (see comments for <b>get_value</b>).</li> <li>• The current status of the measurement value recording can be queried by <b>measurement_status</b>.</li> <li>• For aborting a measurement session 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>(<i>Period</i> = 0) or <b>set_trigger4</b>(<i>Period</i> = 0), then previously recorded measurement values are <i>not</i> lost and the measurement counter halts at its most recent value. This allows subsequent querying by <b>measurement_status</b> of the number of successful data entries. In contrast, if a measurement session is newly started with <b>set_trigger</b> and <i>Period</i> &gt; 0 or <b>set_trigger4</b>(<i>Period</i> &gt; 0), the measurement counter 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. <b>control_command</b> can be used with iDRIVE scan systems (see Glossary entry on <a href="#">page 23</a>) to specify which information is returned on the status channels.</li> <li>Only X measurement signals contain meaningful data with single-status channel scan systems.</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>( <i>HeadA</i>, 0 )), then its status signal can be queried by StatusBX (Signal1/2 = 4).</li> <li>The signals 12 and 15 as well as 27 and 30 are identical, each: SampleAZ_Corr = SampleBZ_Corr and SampleAZ_Trans = SampleBZ_Trans.</li> <li>Status signals Status&lt;...&gt; are a few 10 µs clock cycles later than control signals Sample&lt;...&gt;. See also <a href="#">Section "Reading Out Data", page 199</a>.</li> <li>Signal1, Signal2 = 0, 24, 31...42 and 47...51 enables logging of values that were outputted during the previous clock cycle. Data recording of different signals is not synchronous because of the internal processing chain. Depending on how the signals are combined, you must later correct the data by a fixed time offset.</li> <li>Signal1, Signal2 = 24 enables logging of the signal parameter that is outputted by "Automatic Laser Control" (but not with vector-defined laser control; see <b>set_auto_laser_control</b>). Logging only occurs when "Automatic Laser Control" has been activated and the laser is on. When the laser is switched off, 0 is recorded.</li> <li>Signal1, Signal2 = 31 enables logging of the signal parameter specified for vector-defined laser control (see <b>set_vector_control</b>). Logging is also possible without executing [*]para[*] Commands, but then the signal values remain unchanged.</li> </ul>



<b>Delayed Short List Command</b>	<code>set_trigger</code>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Pulse length and output period (Signal1, Signal2 = 37, 38) are only logged for <b>Signals for "Laser Active" Operation</b> (not for standby signals). In <b>Softstart Mode</b> and in <b>Pixel Output Mode</b>, the pulse length and output period might possibly change faster than the 10 <math>\mu</math>s clock cycle (time resolution of logging by <code>set_trigger</code>). These values cannot be recorded.</li> <li>For Signal1, Signal2 = 39...42 and 48...51, see <b>Chapter 6.9.1 "Free Variables", page 123</b>.</li> </ul>
RTC4→RTC5	<p>Basically unchanged functionality. However:</p> <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> (Signal1/Signal2 = 3, 6) 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>
Version info	Last change DLL 543, OUT 543, RBF 524: Signal1, Signal2 = 52...54.
References	<code>get_waveform</code> , <code>measurement_status</code> , <code>control_command</code> , <code>get_value</code> , <code>get_values</code> , <code>set_mcbsp_out</code> , <code>set_mcbsp_out_ptr</code> , <code>set_trigger4</code>

<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>Unlike <b>set_trigger</b>, only <math>2^{19}</math> entries per measurement channel are available with <b>set_trigger4</b> (automatic termination upon the maximum <math>2^{19}</math> data quadruplet).</li> <li><b>set_trigger</b>(Period = 0) and <b>set_trigger4</b>(Period = 0) both terminate active data recording regardless of which command started the data recording. If <b>set_trigger</b>(Period &gt; 0) or <b>set_trigger4</b>(Period &gt; 0) is used to start a new measurement, then the measurement value counters are reset and existing data overwritten with new measurement values. See also related comments for <b>set_trigger</b>.</li> <li>Data channels 3 and 4 occupy the measurement memory's upper half, as do correction files <math>No=3</math> and <math>No=4</math>. Therefore, both options cannot be used simultaneously. If <b>set_trigger4</b> has been preceded by loading correction file <math>No=3</math> or <math>No=4</math> with <b>load_correction_file</b>, then measurement values overwrite this area and the correction file is no longer available. If loading of correction file <math>No=3</math> or <math>No=4</math> succeeded after any data quadruplets were recorded with <b>set_trigger4</b> (or more than <math>2^{19}</math> data pairs with <b>set_trigger</b>), then the values for channels 3 and 4 (or beyond <math>2^{19}</math> data pairs) are no longer available.</li> <li><b>load_program_file</b> reestablishes the initial state of measurement memory. See also <b>number_of_correction_tables</b>( 2 ).</li> <li>As with <b>set_trigger</b>, <b>get_waveform</b> can be used to transfer recorded values to the PC.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	Last change DLL 543, OUT 543, RBF 524: Signal1, Signal2 = 52...54.
<b>References</b>	<b>set_trigger</b> , <b>stop_trigger</b> , <b>get_waveform</b>

<b>Undelayed Short List Command</b>	<b>set_vector_control</b>	
<b>Function</b>	Initializes or deactivates vector-defined laser control (see <a href="#">page 194</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-defined laser control (for <i>Ctrl</i> = 1...6: identical with <a href="#">set_auto_laser_control</a>).</p> <p>As an unsigned 32-bit value.</p> <p>= 1...8: Defines which signal parameter is to be varied by vector-defined laser control.</p> <ul style="list-style-type: none"> <li>= 1: 12-bit output value at the <a href="#">ANALOG OUT1</a> output port.</li> <li>= 2: 12-bit output value at the <a href="#">ANALOG OUT2</a> output port.</li> <li>= 3: Output value at the 8-bit digital output port.</li> <li>= 4: Pulse length (<a href="#">PulseLength</a>) of the laser signals LASER1 and LASER2.</li> <li>= 5: Output period (<a href="#">HalfPeriod</a>) 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> <li>= 8: Reserved for intelliWELD.</li> </ul> <p>= 0 or &gt; 8: deactivates vector-defined laser control.</p>
	<i>Value</i>	<p>Defines the starting value for the parameter selected by <i>Ctrl</i>.</p> <p>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 <math>\mu</math>s.</li> <li>For <i>Ctrl</i> = 5: [0...(2<sup>32</sup>-1)]. 1 bit equals 1/64 <math>\mu</math>s.</li> <li>For <i>Ctrl</i> = 6: 16-bit values [0...(2<sup>16</sup>-1)].</li> <li>For <i>Ctrl</i> = 7: [0...65,535], <i>Value</i> = focus shift + 32,768.</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 <code>Ctrl = 7</code>, the focus shift is linearly varied as with <code>set_defocus</code> and <code>set_defocus_list</code>, see comments there) with <code>[*]para[*]</code> 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 <code>[*]para[*]</code> Command, the currently set focus shift does not match the initial value (<code>Value</code>) specified by <code>set_vector_control</code>, then the command begins with a "Hard jump" (in the z output) to <code>Value</code>. The setting defined by <code>set_defocus</code> or <code>set_defocus_list</code> is lost.</li> <li>Unlike signed values in the range <math>[-32,768...+32,767]</math> for <code>set_defocus</code> and <code>set_defocus_list</code>, the focus shift (<code>Value</code>) specified for <code>set_vector_control</code> must be an unsigned number shifted upward by 32,768: <code>Value</code> = focus shift + 32,768.</li> <li>"Automatic Laser Control" should not be combined with the variable laser power of <code>set_multi_mcbsp_in</code> or the "freely definable wobble shape".</li> </ul>
RTC4→RTC5	New command. <code>set_vector_control</code> has no RTC4 Compatibility Mode for <code>Value</code> .
Version info	Last change DLL 538: <code>ctrl = 8</code> .
References	<a href="#">set_auto_laser_control</a>



<b>Ctrl Command</b>	<b>set_verify</b>
<b>Function</b>	Activates or deactivates a download verification, see <a href="#">Chapter 6.8.1 "Download Verification", page 120</a> .
<b>Call</b>	<code>OldVerify = set_verify( Verify )</code>
<b>Parameters</b>	<p><code>Verify</code>      Setting parameter.            As an unsigned 32-bit value.            = 0:      Verification is deactivated.            &gt; 0:      Verification is activated.</p>
<b>Result</b>	The <code>Verify</code> parameter that has been active before calling <b>set_verify</b> . As an unsigned 32-bit value.
<b>Comments</b>	<ul style="list-style-type: none"> <li>If verification is activated, the download times are extended.</li> <li>Complete download verification as described in <a href="#">Chapter 6.8.1 "Download Verification", page 120</a> requires RTC5 board driver V1.0.4.0 or higher.            &gt; DLL 511 used in conjunction with an older driver version, then it returns error code 9 (DriverCall error) during <b>load_program_file</b>.</li> <li>Verification of correction file downloads only works if the correction file contains a checksum (otherwise the error code <code>RTC5_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 <b>verify_checksum</b>. If the correction file does not contain a checksum, then <b>verify_checksum</b> creates and inserts a checksum into the file.</li> <li><b>set_verify</b> is available even without explicit access rights to a specific RTC5 board. <b>set_verify</b> only change settings in the <b>RTC5 DLL</b> of the specified (or default) board. It has 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 119</a>) are neither generated nor altered by <b>set_verify</b>.</li> <li>Activation of the download verification by <b>set_verify</b> can generate the <b>get_last_error</b> return code <code>RTC5_OUT_OF_MEMORY</code>, if a <b>RTC5 DLL</b>-internal Windows memory request fails. In this case, the download verification remains deactivated.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>	Sets a numbered wait marker (break point) in 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 <b>Signals for "Laser Active" Operation</b> 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 <b>INTERNAL-BUSY</b> list execution status is set while the home jump is executed). Continuation by <code>execute_list_pos</code>, <code>restart_list</code> or an <b>External Stop</b> is consequently suppressed. However, <code>stop_execution</code> or an <b>External Stop</b> is possible. <code>release_wait</code>, <code>stop_execution</code> or an <b>External Stop</b> removes suppression of the start.</li> <li><code>set_wait</code> sets the <b>PAUSED</b> list execution status and resets the <b>BUSY</b> list execution status (both queriable with <code>get_status</code>). The opposite occurs with a subsequent <code>release_wait</code> (see also <b>Chapter 6.4.3 "List Execution Status", page 97</b>).</li> <li>If processing has been stopped at a wait marker, <code>get_wait_status</code> returns the number of this marker.</li> </ul>
RTC4→RTC5	Basically unchanged functionality. However: Additional <b>PAUSED</b> list execution status which is set by <code>set_wait</code> .
<b>Version info</b>	–
<b>References</b>	<code>get_wait_status</code> , <code>release_wait</code> , <code>pause_list</code> , <code>stop_list</code>

<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 motion. See <a href="#">Chapter 8.4 "Wobbel Mode", page 217</a> .						
<b>Call</b>	<code>set_wobbel( Transversal, Longitudinal, Freq )</code>						
<b>Parameters</b>	<table> <tr> <td>Transversal</td> <td>Amplitude of the elliptical motion <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.</td> </tr> <tr> <td>Longitudinal</td> <td>Amplitude of the elliptical or figure-8 motion <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.</td> </tr> <tr> <td>Freq</td> <td>Frequency of the wobbel motion. In Hz. (number of ellipses per second). As a 64-bit IEEE floating point value. Allowed value range: [-6000...6000]. Larger values are clipped.</td> </tr> </table>	Transversal	Amplitude of the elliptical motion <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 (= $2^{17}$ -1)]. Larger values are clipped.	Longitudinal	Amplitude of the elliptical or figure-8 motion <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 (= $2^{17}$ -1)]. Larger values are clipped.	Freq	Frequency of the wobbel motion. In Hz. (number of ellipses per second). As a 64-bit IEEE floating point value. Allowed value range: [-6000...6000]. Larger values are clipped.
Transversal	Amplitude of the elliptical motion <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 (= $2^{17}$ -1)]. Larger values are clipped.						
Longitudinal	Amplitude of the elliptical or figure-8 motion <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 (= $2^{17}$ -1)]. Larger values are clipped.						
Freq	Frequency of the wobbel motion. In Hz. (number of ellipses per second). As a 64-bit IEEE floating point value. Allowed value range: [-6000...6000]. Larger values are clipped.						
<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 motion is added to the linear marking motion, resulting in a spiral motion of the laser focus in the <a href="#">Image field</a>. 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 219</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 \text{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 motions 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 motion.</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 219.</li> <li>As of version DLL 543, OUT 543, RBF 524 the present wobbel amplitude can be recorded with trigger signal 53 (see <b>set_trigger</b> or <b>set_trigger4</b>). The format of the data is ((transversal &lt;&lt; 16) + longitudinal).</li> </ul>
RTC4→RTC5	<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 RTC5 multiplies the specified value for Longitudinal and Transversal by 16. The allowed value ranges decrease accordingly.</p>
Version info	–
References	<b>set_mark_speed, set_wobbel_mode</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 <code>Ctrl &gt; 6</code>: <a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</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 <code>Ctrl</code> parameter and on <code>Value</code>.</p> <p><b>MaxValue</b> See <code>MinValue</code>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Any still-pending delayed short list command executes first.</li> <li>For <code>Ctrl = 0</code> and <code>Ctrl &gt; 6</code>, laser power variation is switched off (initialization state after <a href="#">load_program_file</a>). The values for McBSP/SPI multi transfers are then not used.</li> <li>The conditions for <code>Value</code>, <code>MinValue</code>, <code>MaxValue</code> 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 <code>MinValue = 0</code> and <code>MaxValue = 0xFFFFFFFF</code>, that is, no restrictions.</li> <li>The laser power can be combined with the vector-defined laser control, if the corresponding <code>Ctrl</code> parameter has 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". Otherwise, laser power is not varied, even if you make a change in the data sets.</li> <li>Special case: if <code>Value = 0xFFFFFFFF</code>, 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 point in time set by other normal commands, for example, <a href="#">write_da_1_list</a> for <code>Ctrl = 1</code>. These are executed prior to <a href="#">set_wobbel_control</a>. But beware: with execution of a "freely definable wobbel shape", the value at the port can change at any time. Subsequent calls of <a href="#">set_wobbel_control</a> then return other values for the nominal laser power.</li> </ul>



<b>Undelayed Short List Command</b>	<b>set_wobbel_control</b>
RTC4→RTC5	New command.
Version info	–
References	<b>set_wobbel_vector, set_multi_mcbsp_in, set_multi_mcbsp_in_list</b>

<b>Undelayed Short List Command</b>	<b>set_wobbel_direction</b>				
Function	Defines a direction vector for wobbel motions.				
Call	<code>set_wobbel_direction( dx, dy )</code>				
Parameters	<table> <tr> <td><code>dx</code></td> <td>x component of the direction vector. In bits. As a signed 32-bit value.</td> </tr> <tr> <td><code>dy</code></td> <td>y component of the direction vector. In bits. As a signed 32-bit value.</td> </tr> </table>	<code>dx</code>	x component of the direction vector. In bits. As a signed 32-bit value.	<code>dy</code>	y component of the direction vector. In bits. As a signed 32-bit value.
<code>dx</code>	x component of the direction vector. In bits. As a signed 32-bit value.				
<code>dy</code>	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 (<code>dx</code> and/or <code>dy</code> 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 RTC5 normalizes the direction vector.</li> <li>If <code>dx = dy = 0</code>, 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→RTC5	New command.				
Version info	–				
References	<b>set_wobbel, set_wobbel_direction, set_wobbel_mode</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 217</a> .	
<b>Call</b>	set_wobbel_mode( Transversal, Longitudinal, Freq, Mode )	
<b>Parameters</b>	<p>Transversal      Amplitude of the elliptical or figure-8 motion <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 motion <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 motion 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 motion.            &lt; 0:      Figure-of-8 wobbel motion perpendicular to the motion direction (vertical 8).            = 1:      Figure-of-8 wobbel motion 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 <a href="#">set_wobbel_mode</a> 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 217</a>), then Mode &gt; 1 selects this shape, otherwise the horizontal figure-8 shape is selected similarly to Mode = 1. If you <i>subsequently</i> 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 wobbel 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 (Longitudinal = Transversal), 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 motions 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 motion.</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 &gt; 1</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 219.</li> <li>As of version DLL 543, OUT 543, RBF 524 the present wobbel amplitude can be recorded with trigger signal 53 (see <b>set_trigger</b> or <b>set_trigger4</b>). 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>– <b>Sky Writing</b></li> <li>– <b>Pixel Output Mode</b></li> <li>– <b>Jumps</b></li> <li>– <b>laser_on_list</b></li> </ul> </li> </ul>
RTC4→RTC5	<p>New command.</p> <p>In <b>RTC4 Compatibility Mode</b>, the RTC5 multiplies the specified value for <b>Longitudinal</b> and <b>Transversal</b> by 16. The allowed value ranges decrease accordingly.</p>
Version info	–
References	<b>set_wobbel, set_wobbel_control, set_wobbel_vector</b>

<b>Delayed Short List Command</b>	<b>set_wobble_offset</b>
Function	Defines a wobble shape shift in the direction of motion or perpendicular to the direction of motion.
Call	<code>set_wobble_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 <a href="#">load_program_file</a>: (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” wobble shapes (circle, ellipse, sine, figure-8) as well as for “freely definable wobble shapes”, see <a href="#">Chapter 8.4 “Wobble Mode”, page 217</a>.</li> <li>The summed up wobble amplitude including offsets may never exceed <math>\pm(2^{17}-1)</math>.</li> <li>At the beginning of the wobble marking offsets are set as “<a href="#">Hard jumps</a>”. This applies also in case of switching-off or non-using the Wobble mode, for example, when using a <a href="#">Jump command</a> (analogously to “classical” wobble shapes longitudinal amplitudes).</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>In <a href="#">RTC4 Compatibility Mode</a> the RTC5 multiplies the specified values for OffsetTrans and OffsetLong by 16. The allowed value ranges decrease accordingly.</p>
Version info	–
References	<a href="#">set_wobble_mode</a> , <a href="#">set_wobble_vector</a> , <a href="#">set_wobble_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><b>dTrans</b> <b>Microstep</b> of a linear wobbel shape section. In bits.  <b>dLong</b> As a 64-bit IEEE floating point value.          Allowed value range: [-256.0...+255.0].  <b>dTrans</b> is the wobbel excursion perpendicular to the direction of motion (which is either the laser <b>Trajectory</b> or the direction of motion defined by <b>set_wobbel_direction</b>).  <b>dLong</b> is correspondingly longitudinal to it.</p> <p><b>Period</b> As an unsigned 32-bit value.          Allowed value range: [0...65,535].          = 1...65,535: number of <b>Microsteps</b>.          = 0: The wobbel shape is switched off.</p> <p><b>dPower</b> <b>Microstep</b> 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>• <b>set_wobbel_vector</b> cannot be used together with the Softstart function because both require a shared memory area on the RTC5. To avoid (unintentional) mutual overwriting, you absolutely must explicitly deactivate Softstart. Otherwise, <b>set_wobbel_vector</b> has no effect. <b>set_wobbel_vector</b> itself blocks subsequent memory changes by <b>set_softstart_level</b> or <b>set_softstart_level_list</b>. These commands have no effect as long as the wobbel shape has not been explicitly switched off.</li> <li>• <b>Period</b> = 0 is not allowed as a shape section. <b>Period</b> = 0 means explicitly switch off the “freely definable wobbel shape” (not the wobbel mode itself, see <b>set_wobbel_mode</b>). Subsequent calls of <b>set_wobbel_vector</b> begin a new wobbel shape.</li> <li>• Each call of <b>set_wobbel_vector</b> adds a new section at the end of the previous section independently of when the call is performed.</li> <li>• Up to 512 wobbel shape sections can be defined. After 512 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 “<b>Hard jump</b>” of “classic” wobbel shapes is eliminated if the longitudinal amplitude ≠ 0) and also ends there without requiring explicit declaration.</li> </ul>

<b>Undelayed Short List Command</b>	<b>set_wobbel_vector</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Each wobbel shape section consists of a vector defined by <b>Microsteps</b> and their number. The parameters <b>dTrans</b>, <b>dLong</b> get internally rounded to 7 bit decimal places. At runtime each <b>Microsteps</b> is executed within 10 <math>\mu</math>s. For large <b>Period</b> values, you should therefore take rounding error into account. Positive <b>dTrans</b> values cause that in respect to the direction of motion the start is to the right (which applies to circular wobbel shapes as well). Positive <b>dLong</b> values cause that in respect to the direction of motion 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 might get executed as a "<b>Hard jump</b>". 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 <math>\pm(2^{17}-1)</math>.</li> <li>If activated by <b>set_wobbel_control</b>, the RTC5 varies the current laser power <b>P</b> within the wobbel shape section as per <math>P = P_0 \times (1 + dPower \times n)</math>, whereby <math>0 \leq n \leq \text{Period}</math>. <b>n</b> represents the current number of each <b>Microstep</b>. You can define the nominal laser power <b>P0</b> 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 <b>P0</b> is multiplicatively varied by the laser power parameter <b>P</b> across multiple McBSP/SPI transfers: <math>P_{\text{McBSP}} = P_0 \times P / 16384</math>.</li> <li>Beware here that for each both corrections above a maximum laser power of only <math>P_0 \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 <b>Ctrl</b> = 5 (half period), <b>dPower</b> 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 <b>P0</b> 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 <b>P0</b> is emitted at the port assigned by <b>Ctrl</b> if <b>set_wobbel_control</b> has been last called with <b>Value</b> = 0xFFFFFFFF. Otherwise, the laser power <b>P0</b> multiplied by the external factor from <b>set_multi_mcbsp_in</b> is emitted.</li> <li>If the wobbel shape gets switched off while the wobbel mode remains active, then the shape automatically switches to <b>Mode</b> = 1 (horizontal figure-8).</li> <li>The wobbel mode with "freely definable wobbel shapes" also needs to be switched on by <b>set_wobbel_mode</b>.</li> </ul>

<b>Undelayed Short List Command</b>	<b>set_wobbel_vector</b>
Comments (cont'd)	<ul style="list-style-type: none"> <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 <b>Ctrl</b> parameters are identical. Though unidentical <b>Ctrl</b> parameters are allowed, they serve no practical purpose.</li> <li>When defining "freely definable wobbel shapes" make sure that the <b>Microsteps</b> of each individual wobbel vector does not exceed the maximum positioning speed significantly! Otherwise, a galvanometer scanner overheating may occur, see also <b>Chapter 8.4.1 "Wobbel Shapes – Important Notes on Choosing Appropriate Parameter Values"</b>, page 219</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#"><b>set_multi_mcbsp_in</b></a> , <a href="#"><b>set_multi_mcbsp_in_list</b></a> , <a href="#"><b>set_wobbel_control</b></a>

<b>Ctrl Command</b>	<b>set_zoom</b>
Comments	<ul style="list-style-type: none"> <li>This command is described, for example, in the manual for the intelliWELD II FT.</li> </ul>

<b>Undelayed Short List Command</b>	<b>set_zoom_list</b>
Comments	<ul style="list-style-type: none"> <li>This command is described, for example, in the manual for the intelliWELD II FT.</li> </ul>



<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:    <b>ENCODER X</b> pulses are simulated and encoder counter "Encoder0" thereby incremented.</li> <li>= 2:    <b>ENCODER Y</b> pulses are simulated and encoder counter "Encoder1" thereby incremented.</li> <li>= 3:    Pulses for <b>ENCODER X</b> and <b>ENCODER Y</b> 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 275</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>RTC5_PARAM_ERROR</code>).</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
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 <b>External Start</b> .
<b>Call</b>	<code>simulate_ext_start( Delay, EncoderNo )</code>
<b>Parameters</b>	<p><code>Delay</code>      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><code>EncoderNo</code>      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>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <b>External Starts</b>, see <a href="#">Section "External Start", page 266</a>.</li> <li>The track delay is specified in <i>relative</i> counting units of the selected encoder counter (the RTC5 encoder counters are triggered by an external or simulated encoder signal, see <a href="#">Chapter 9.3.3 "Synchronization by Encoder Signals", page 274</a>). The start trigger for the start occurs only after the internal encoder counter has reached the specified track delay. <b>External Starts</b> initiated by <b>simulate_ext_start</b> or by an external start signal, but whose execution has been postponed according to the specified track delay, are placed in a queue that accommodates up to 8 starts. <b>simulate_ext_start</b> cancels a previous queue and starts a new one.</li> <li>A start trigger initiated by <b>simulate_ext_start</b> or an external start signal only triggers a start if it does not occur when the <b>BUSY</b> list execution status is set (for example, when outputting a list), when the <b>INTERNAL-BUSY</b> list execution status is set (for example, during <b>goto_xy</b>) and/or when the <b>PAUSED</b> list execution status is set (after <b>pause_list</b>, <b>stop_list</b> or <b>set_wait</b>). Otherwise, Bit #11 of the <b>get_startstop_info</b> return value is set. Therefore, if an unsuitable track delay is specified (for example, <code>Delay = 0</code>), no start is triggered. If <b>simulate_ext_start</b> is the first command in a list, then <b>get_startstop_info</b> 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 <b>set_ext_start_delay</b> or <b>set_ext_start_delay_list</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 by <b>set_control_mode</b> (Bit #2).</li> <li>The <b>simulate_ext_start</b> command alone <i>does not</i> cause an encoder reset. But if accordingly set with <b>set_control_mode</b> (Bit #9), a start trigger initiated by <b>simulate_ext_start</b>, <b>simulate_ext_start_ctrl</b> 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 <b>set_fly_x</b>, <b>set_fly_y</b>, <b>set_fly_2d</b>, <b>set_fly_2d</b> or <b>set_fly_rot</b>.</li> <li>If <code>EncoderNo &gt; 1</code>, then <b>simulate_ext_start</b> is replaced with a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> </ul>

Normal List Command	<code>simulate_ext_start</code>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified value for <code>Delay</code> by 16. The allowed value range decreases accordingly.
Version info	–
References	<code>simulate_ext_start_ctrl</code> , <code>set_ext_start_delay</code> , <code>set_ext_start_delay_list</code> , <code>set_extstartpos</code> , <code>set_extstartpos_list</code> , <code>set_control_mode</code> , <code>simulate_ext_stop</code>

Ctrl Command	<code>simulate_ext_start_ctrl</code>
Function	Causes a simulated <b>External Start</b> .
Call	<code>simulate_ext_start_ctrl()</code>
Comments	<ul style="list-style-type: none"> <li>For <b>External Starts</b>, see <b>Section "External Start", page 266</b>.</li> <li>The start trigger for the start occurs only after a track delay previously set by <code>set_ext_start_delay</code>, <code>set_ext_start_delay_list</code> or <code>simulate_ext_start</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). External Starts initiated by <code>simulate_ext_start_ctrl</code> or by an external start signal, but whose execution has been postponed according to the specified track delay, are placed in a queue that accommodates up to 8 starts. In contrast to <code>simulate_ext_start</code>, <code>simulate_ext_start_ctrl</code> does <i>not</i> cancel the previous queue.</li> <li>A start trigger initiated by <code>simulate_ext_start_ctrl</code> 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 <code>get_startstop_info</code> return value gets set).</li> <li>The <code>simulate_ext_start_ctrl</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_ctrl</code>, <code>simulate_ext_start</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>As of version DLL 544, OUT 544, the following applies: <code>simulate_ext_start_ctrl</code> can be disabled by <code>set_control_mode</code> (Bit #4 = 1) or <code>set_control_mode_list</code> (Bit #4 = 1).</li> <li>As of version DLL 544, OUT 544, the following applies: provided that an execution start is allowed, <code>simulate_ext_start_ctrl</code> waits 30 µs before it returns. This closes a potential timing gap (with <code>get_status</code>) between command call and actual start.</li> </ul>
RTC4→RTC5	New command.
Version info	Last change DLL 544, OUT 544: see comment.
References	<code>simulate_ext_start</code>



<b>Ctrl Command</b>	<b>simulate_ext_stop</b>
<b>Function</b>	Causes a simulated <b>External Stop</b> .
<b>Call</b>	<code>simulate_ext_stop()</code>
<b>Comments</b>	<ul style="list-style-type: none"><li>For external stops, see <a href="#">Section "External Stop", page 265</a>.</li><li><code>simulate_ext_stop</code> simultaneously halts the master board and all <i>downstream</i> slave boards. See also <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 113</a>.</li><li>In contrast, <a href="#"><b>stop_execution</b></a> only halts the master board.</li></ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#"><b>simulate_ext_start</b></a> , <a href="#"><b>simulate_ext_start_ctrl</b></a> , <a href="#"><b>stop_execution</b></a> , <a href="#"><b>sync_slaves</b></a>

Ctrl Command	<b>start_loop</b>
Function	Starts a repeating automatic list change.
Call	<code>start_loop()</code>
Comments	<ul style="list-style-type: none"> <li>A list change or execution restart activated with <b>start_loop</b> repeats until execution is ended by calling <b>quit_loop</b>.</li> <li><b>start_loop</b> can be called at any desired point in 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 beginning of the list).</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 "Automatic List Changing", page 99</a>.</li> <li>So that the <b>start_loop</b> command 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, the <b>start_loop</b> and <b>auto_change_pos( Pos &gt; 0 )</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> <li>The current <b>List Status</b> can be queried by <b>read_status</b>.</li> <li>The current <b>List Execution Status</b> can be queried by <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 94</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
References	<b>quit_loop</b>

<b>Ctrl Command</b>	<b>stepper_abs</b>						
<b>Function</b>	Triggers set-position motions to the specified absolute set positions by both stepper motor outputs.						
<b>Restriction</b>	For older RTC5 Boards with <b>DSP</b> version numbers < 2 ( <b>get_RTC_version</b> Bit #16...Bit #23), <b>stepper_abs</b> is not executed ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</b> ).						
<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 260</a>.</li> <li><b>stepper_abs</b> sets the new set-position values even if a previously started set-position motion is still in progress: <ul style="list-style-type: none"> <li>If Pos1/Pos2 in the current direction of motion lies in front of the internal position variable's value, then the motion continues and the Busy status remains set.</li> <li>If Pos1/Pos2 equals the current value of the internal position variable, then the motion stops. The Busy status gets reset.</li> <li>If Pos1/Pos2 in the current direction of motion already lies past the internal position variable's value, then the corresponding stepper motor's direction of motion reverses (see note on <a href="#">page 260</a>). The Busy status remains set.</li> </ul> </li> <li>If no set-position motion is in progress, then one starts and the Busy status gets set.</li> <li>During performance of a reference motion (Init status set, see <b>stepper_init</b>), <b>stepper_abs</b> does not execute (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</li> <li>If the CLOCK pulse period has been set to 0 by <b>stepper_init</b>, <b>stepper_control</b> or <b>stepper_control_list</b>, then no stepper motor motion 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→RTC5	New command.						
Version info	–						
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>				
<b>Function</b>	Like <b>stepper_abs</b> , but a list command and without <code>WaitTime</code> parameter.				
<b>Restriction</b>	For older RTC5 Boards with <b>DSP</b> version numbers < 2 ( <code>get_RTC_version</code> Bit #16...Bit #23), <b>stepper_abs_list</b> is neither executed nor replaced by <b>list_nop</b> ( <code>get_last_error</code> return code <b>RTC5_TYPE_REJECTED</b> ).				
<b>Call</b>	<code>stepper_abs_list( Pos1, Pos2 )</code>				
<b>Parameters</b>	<table> <tr> <td>Pos1</td> <td>Like <b>stepper_abs</b>.</td> </tr> <tr> <td>Pos2</td> <td>Like <b>stepper_abs</b>.</td> </tr> </table>	Pos1	Like <b>stepper_abs</b> .	Pos2	Like <b>stepper_abs</b> .
Pos1	Like <b>stepper_abs</b> .				
Pos2	Like <b>stepper_abs</b> .				
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <b>stepper_abs</b>.</li> <li>• During performance of a reference motion (see <b>stepper_init</b>), execution of <b>stepper_abs_list</b> is delayed until the reference motion completes.</li> </ul>				
<b>RTC4→RTC5</b>	New command.				
<b>Version info</b>	–				
<b>References</b>	<b>stepper_abs</b>				

<b>Ctrl Command</b>	<b>stepper_abs_no</b>						
<b>Function</b>	Triggers a set-position motion to the specified absolute set position at <i>one</i> stepper motor output.						
<b>Call</b>	<code>stepper_abs_no( No, Pos, WaitTime )</code>						
<b>Parameters</b>	<table> <tr> <td>No</td> <td>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 (<code>get_last_error</code> return code <b>RTC5_PARAM_ERROR</b>).</td> </tr> <tr> <td>Pos</td> <td>Absolute set position in CLOCK pulse units. As a signed 32-bit value. Allowed value range: <math>[-2^{31} \dots + (2^{31}-1)]</math>.</td> </tr> <tr> <td>WaitTime</td> <td>Determines when <b>stepper_abs_no</b> returns at the latest. <i>1 bit equals 1 s.</i> As an unsigned 32-bit value. Allowed value range: <math>[0 \dots + (2^{32}-1)]</math>.</td> </tr> </table>	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 ( <code>get_last_error</code> return code <b>RTC5_PARAM_ERROR</b> ).	Pos	Absolute set position in CLOCK pulse units. As a signed 32-bit value. Allowed value range: $[-2^{31} \dots + (2^{31}-1)]$ .	WaitTime	Determines when <b>stepper_abs_no</b> returns at the latest. <i>1 bit equals 1 s.</i> As an unsigned 32-bit value. Allowed value range: $[0 \dots + (2^{32}-1)]$ .
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 ( <code>get_last_error</code> return code <b>RTC5_PARAM_ERROR</b> ).						
Pos	Absolute set position in CLOCK pulse units. As a signed 32-bit value. Allowed value range: $[-2^{31} \dots + (2^{31}-1)]$ .						
WaitTime	Determines when <b>stepper_abs_no</b> returns at the latest. <i>1 bit equals 1 s.</i> As an unsigned 32-bit value. Allowed value range: $[0 \dots + (2^{32}-1)]$ .						
<b>Comments</b>	<ul style="list-style-type: none"> <li>A set-position motion is only performed at the stepper motor output port specified by <i>No</i>. Otherwise, <b>stepper_abs_no</b> is identical to <b>stepper_abs</b> (see comments there).</li> </ul>						
<b>RTC4→RTC5</b>	New command.						
<b>Version info</b>	–						
<b>References</b>	<b>stepper_abs, 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.
Restriction	For older RTC5 Boards with DSP version numbers < 2 ( <b>get_RTC_version</b> Bit #16...Bit #23), <b>stepper_abs_no_list</b> is neither executed nor replaced by <b>list_nop</b> ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</b> ).
Call	<b>stepper_abs_no_list( No, Pos )</b>
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>RTC5_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 motion (see <b>stepper_init</b>), execution of <b>stepper_abs_no_list</b> is delayed until the reference motion completes.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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>Restriction</b>	For older RTC5 Boards with DSP version numbers < 2 ( <a href="#">get_RTC_version</a> Bit #16...Bit #23), <b>stepper_control</b> is not executed ( <a href="#">get_last_error</a> return code <a href="#">RTC5_TYPE_REJECTED</a> ).
<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.  1 bit equals 10 <math>\mu</math>s.  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.  = 0: An already-running CLOCK pulse period aborts at each stepper motor output (the corresponding stepper motor motion gets stopped, the Init- and/or Busy statuses for the respective stepper motor outputs get reset).  &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 260</a>.</li> <li>Period1 or Period2= 0 can be used as an emergency stop (see also <a href="#">Section "Terminating Infinite Motions", page 262</a>).</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">stepper_control_list</a>

<b>Undelayed Short List Command</b>	<b>stepper_control_list</b>
<b>Function</b>	Like <a href="#">stepper_control</a> , but a list command.
<b>Restriction</b>	Like <a href="#">stepper_control</a> .
<b>Call</b>	<code>stepper_control_list( Period1, Period2 )</code>
<b>Parameters</b>	<p>Period1      Like <a href="#">stepper_control</a>.</p> <p>Period2      Like <a href="#">stepper_control</a>.</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">stepper_control</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">stepper_control</a>



<b>Ctrl Command</b>	<b>stepper_disable_switch</b>
<b>Function</b>	Controls the usage of the stepper motor control SWITCH signals.
<b>Call</b>	<code>stepper_disable_switch( Disable1, Disable2 )</code>
<b>Parameters</b>	<p>Disable1      Instruction how the SWITCH signals from stepper motor input 1 are to be used. As a signed 32-bit value.            Allowed value range: <math>[-2^{32} \dots + (2^{32}-1)]</math>.</p> <p>&gt; 0:      The SWITCH signal at the stepper motor input is not used.            = 0:      The SWITCH signal at the stepper motor input is used.            &lt; 0:      The use of the stepper motor input signal remains unchanged.</p> <p>Disable2      Like Disable1 (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 260</a>.</li> <li>The limit switch can be ignored during normal motions. For example, this may make sense for continuously rotating axes.</li> <li>The SWITCH signals are always used with motions initiated by <a href="#">stepper_init</a>.</li> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	Available as of DLL 542, OUT 542.
<b>References</b>	<a href="#">stepper_init</a>

<b>Ctrl Command</b>	<b>stepper_enable</b>
<b>Function</b>	Changes the stepper motor control's ENABLE signals.
<b>Restriction</b>	For older RTC5 Boards with DSP version numbers < 2 (get_rtc_version Bit #16...Bit #23), <b>stepper_enable</b> is not executed (get_last_error return code <b>RTC5_TYPE_REJECTED</b> ).
<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 Enable1 (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 260</a>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<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>Restriction</b>	Like <a href="#">stepper_enable</a> .
<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>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">stepper_enable</a>

<b>Ctrl Command</b>	<b>stepper_init</b>
<b>Function</b>	Initializes a stepper motor.
<b>Restriction</b>	For older RTC5 Boards with <b>DSP</b> version numbers < 2 ( <b>get_RTC5_Version</b> Bit #16...Bit #23), <b>stepper_init</b> is not executed ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</b> ).
<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              (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</p> <p>Period    Pulse period of the CLOCK signal.              As an unsigned 32-bit value.              1 bit equals 10 <math>\mu</math>s.              Allowed value range: [0...+(2<sup>24</sup>-1)]. Larger values are clipped.              If <b>Period</b> = 0, then no reference motion is performed and the function              immediately returns (see comments).</p> <p>Dir        Direction of stepper motor motion during the reference motion.              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 motion the                  internal position variable increments.              = 0: The DIRECTION signal gets reset; during the reference motion the                  internal position variable decrements.              &lt; 0: The DIRECTION signal remains unchanged, no reference motion 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 motion (see comments).              As an unsigned 32-bit value.              Allowed value range: [0...+(2<sup>32</sup>-1)].              If <b>Dir</b> &lt; 0 and/or <b>Period</b> = 0, then the value <b>Tol</b> = 0 is irrelevant, otherwise              <b>Tol</b> = 0 causes <b>stepper_init</b> not to be executed              (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</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).              1 bit equals 1 s.              As an unsigned 32-bit value.              Allowed value range: [0...+(2<sup>32</sup>-1)].</p>

Ctrl Command	stepper_init
Comments	<ul style="list-style-type: none"> <li>For programming the stepper motor signals, see <a href="#">Chapter 9.1.5 "Controlling Stepper Motors", page 260</a>.</li> <li><b>stepper_init</b> immediately stops all previously started motions 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 ≥ 0</code>, then stepper motor <code>No</code> starts a reference motion 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_stepper_status</a>).</li> <li>If a limit switch gets activated during a motion, then the reference motion stops there. Afterward, the limit switch position is crossed 4x to arrive at an averaged value for this position. Finally, the stepper motor is driven in the opposite direction by a normal set-position motion (<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 motions 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 motion. <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 motion continues until it is aborted by a new <b>stepper_init</b> command or <b>stepper_control</b>( <code>Period1/Period2 = 0</code>). For more on this, see the <a href="#">Section "Terminating Infinite Motions", page 262</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_stepper_status</a> to check whether the reference motion has completed. During the seeking motion the status "Init" (see <a href="#">page 360</a>) is set and during the terminating set-position motion to (limit switch + <code>Tol</code>) the status "Busy" (see <a href="#">page 360</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 motion gets aborted, but no reference motion 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 360</a>) and status "Busy" (see <a href="#">page 360</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→RTC5	New command.
Version info	–
References	–

<b>Ctrl Command</b>	<b>stepper_rel</b>
Function	Triggers set-position motions to the specified relative set positions by both stepper motor outputs.
Restriction	For older RTC5 Boards with DSP version numbers < 2 ( <a href="#">get_RTC_version</a> Bit #16...Bit #23), <b>stepper_rel</b> is not executed ( <a href="#">get_last_error</a> return code <a href="#">RTC5_TYPE_REJECTED</a> ).
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 desired set positions should be specified relative to the current internal set-position values. Otherwise, <b>stepper_rel</b> is identical to <a href="#">stepper_abs</a> (see comments there).</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 <a href="#">WaitTime</a> parameter.
Restriction	Like <a href="#">stepper_rel</a> .
Call	<code>stepper_rel_list( dPos1, dPos2 )</code>
Parameters	dPos1,      Like <a href="#">stepper_rel</a> . dPos2
Comments	<ul style="list-style-type: none"> <li>See <a href="#">stepper_rel</a>.</li> <li>During performance of a reference motion (see <a href="#">stepper_init</a>), execution of <a href="#">stepper_rel_list</a> is delayed until the reference motion completes.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">stepper_rel</a>



<b>Ctrl Command</b>	<b>stepper_rel_no</b>						
<b>Function</b>	Triggers a set-position motion to the specified relative position at one stepper motor output port.						
<b>Restriction</b>	For older RTC5 Boards with <b>DSP</b> version numbers < 2 ( <b>get_RTC_version</b> Bit #16...Bit #23), <b>stepper_rel_no</b> is not executed ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</b> ).						
<b>Call</b>	<b>stepper_rel_no( No, dPos, WaitTime )</b>						
<b>Parameters</b>	<table> <tr> <td>No</td> <td>Like <b>stepper_abs_no</b>.</td> </tr> <tr> <td>dPos</td> <td>Relative position. In CLOCK pulse units. As a signed 32-bit value. Allowed value range: [-2<sup>31</sup>...+(2<sup>31</sup>-1)].</td> </tr> <tr> <td>WaitTime</td> <td>Like <b>stepper_abs_no</b>.</td> </tr> </table>	No	Like <b>stepper_abs_no</b> .	dPos	Relative position. In CLOCK pulse units. As a signed 32-bit value. Allowed value range: [-2 <sup>31</sup> ...+(2 <sup>31</sup> -1)].	WaitTime	Like <b>stepper_abs_no</b> .
No	Like <b>stepper_abs_no</b> .						
dPos	Relative position. In CLOCK pulse units. As a signed 32-bit value. Allowed value range: [-2 <sup>31</sup> ...+(2 <sup>31</sup> -1)].						
WaitTime	Like <b>stepper_abs_no</b> .						
<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 motion is only performed at the stepper motor output specified by <b>No</b>. Otherwise, <b>stepper_rel_no</b> is identical to <b>stepper_abs</b> (see comments there).</li> </ul>						
RTC4→RTC5	New command.						
Version info	–						
References	<b>stepper_abs_no, stepper_abs, stepper_rel_no_list</b>						

<b>Undelayed Short List Command</b>	<b>stepper_rel_no_list</b>				
<b>Function</b>	Like <b>stepper_rel_no</b> , but a list command and without <b>WaitTime</b> parameter.				
<b>Restriction</b>	Like <b>stepper_rel_no</b> .				
<b>Call</b>	<b>stepper_rel_no_list( No, dPos )</b>				
<b>Parameters</b>	<table> <tr> <td>No</td> <td>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_rel_no_list</b> is, already during loading, replaced by a <b>list_nop</b> (<b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b>).</td> </tr> <tr> <td>dPos</td> <td>Like <b>stepper_rel_no</b>.</td> </tr> </table>	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_rel_no_list</b> is, already during loading, replaced by a <b>list_nop</b> ( <b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b> ).	dPos	Like <b>stepper_rel_no</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. If the value is invalid, then <b>stepper_rel_no_list</b> is, already during loading, replaced by a <b>list_nop</b> ( <b>get_last_error</b> return code <b>RTC5_PARAM_ERROR</b> ).				
dPos	Like <b>stepper_rel_no</b> .				
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <b>stepper_rel_no</b>.</li> <li>During performance of a reference motion (see <b>stepper_init</b>), execution of <b>stepper_rel_no_list</b> is delayed until the reference motion completes.</li> </ul>				
RTC4→RTC5	New command.				
Version info	–				
References	<b>stepper_rel_no</b>				

<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 motion completes.
<b>Restriction</b>	For older RTC5 Boards with <b>DSP</b> version numbers < 2 ( <b>get_RTC_version</b> Bit #16...Bit #23), <b>stepper_wait</b> is neither executed nor replaced by <b>list_nop</b> ( <b>get_last_error</b> return code <b>RTC5_TYPE_REJECTED</b> ).
<b>Call</b>	<b>stepper_wait( No )</b>
<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 outputs. 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 <b>Chapter 9.1.5 "Controlling Stepper Motors", page 260</b>.</li> <li>If no stepper motor motion had been previously started at the specified stepper motor output, then <b>stepper_wait</b> still needs 10 <math>\mu</math>s 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 <b>Signals for "Laser Active" Operation</b></li> <li>the <b>List Status</b></li> <li>the <b>List Execution Status</b></li> </ul> </li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<b>stepper_rel_no</b>



<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 <b>Signals for “Laser Active” Operation</b> even if no list is active (here <b>stop_execution</b> has no other effects; here too: <b>get_last_error</b> return code <b>RTC5_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 <b>goto_xy</b>.</li> <li>• The external start inputs are disabled, see <b>Section “External Start”, page 266</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-List Execution Status value (see <b>get_status</b>) are reset.</li> <li>• A list that has been interrupted by <b>stop_execution</b> cannot be resumed. It 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 RTC5 board. In a master/slave chain, <b>stop_execution</b> is not passed on to the <i>downstream</i> slave boards. If all RTC5 boards of a master/slave chain shall be synchronously stopped, then <b>simulate_ext_stop</b> or an external stop signal must be applied to the master board, see also <b>Chapter 6.6.3 “Master/Slave Operation”, page 113</b>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
References	<b>get_startstop_info</b>

<b>Ctrl Command</b>	<b>stop_list</b>
<b>Function</b>	Pauses execution of the list and deactivates the <b>Signals for “Laser Active” Operation</b> .
<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→RTC5	Basically unchanged functionality. However: Additional <b>PAUSED</b> list execution status which is set by <b>stop_list</b> .
Version info	–
References	<b>pause_list</b>



Ctrl Command	<b>stop_trigger</b>
Function	Resets (to <code>Busy = 0</code> ) a measurement session's status that can be queried by <code>measurement_status</code> .
Call	<code>stop_trigger()</code>
Comments	<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 status even when no list is active (see <code>set_trigger</code> comments).</li> <li>• <code>stop_trigger</code> is not executed (<code>get_last_error</code> return code <code>RTC5_BUSY</code>), if: <ul style="list-style-type: none"> <li>– the <code>BUSY</code> list execution status is set</li> <li>– the <code>INTERNAL-BUSY</code> list execution status 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> (<code>PAUSED</code> list execution status set)</li> </ul> </li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<code>measurement_status</code> , <code>set_trigger</code> , <code>set_trigger4</code>

<b>Undelayed Short List Command</b>	<b>sub_call</b>
Function	Causes an unconditional jump to an indexed subroutine.
Call	<code>sub_call( Index )</code>
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 <b>list_call</b> (see also the comments there). <b>list_call</b> 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.  <code>get_sub_pointer( Index )</code> 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>).</li> <li>• If <code>Index &gt; 1023</code>, then <b>sub_call</b> is, already during loading, replaced by a <b>list_nop</b> (<code>get_last_error</code> return code <code>RTC5_PARAM_ERROR</code>).</li> <li>• Absolute <b>Vector commands</b> and “<b>Arc</b>” <b>commands</b> execute absolutely after being called with <b>sub_call</b>. If the subroutine needs to execute at various locations within the <b>Image field</b>, then either the subroutine can only contain relative <b>[*]mark[*] Commands</b>, <b>arc commands</b> or <b>Jump commands</b> or <b>sub_call_abs</b> must be used instead.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">list_call</a> , <a href="#">sub_call_abs</a> , <a href="#">sub_call_cond</a>

<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 <b>Vector commands</b> and <b>"Arc" commands</b> 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>• <b>sub_call_abs</b> 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→RTC5	New command.
Version info	–
References	<b>sub_call</b> , <b>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 IOvalue at the 16-bit digital input port of the EXTENSION 1 socket connector meets the following condition: $((IOvalue \text{ AND } Mask1) = Mask1) \text{ AND } (((\text{not } IOvalue) \text{ AND } Mask0) = Mask0)$ (= 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 <b>Chapter 9.3.2 "Conditional Command Execution", page 271</b>.</li> </ul>						
RTC4→RTC5	New command.						
Version info	–						
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>	<p>Index      Index of the to be called indexed subroutine (as with <a href="#">sub_call_abs</a>).</p> <p>Number      Number of repetitions. 0 is treated as 1. As an unsigned 32-bit value.</p>
<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→RTC5	New command.
Version info	Available as of DLL 538, OUT 538.
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: sub_call_cond executes sub_call( Index ), if the current IOvalue at the 16-bit digital input port of the EXTENSION 1 socket connector meets the following condition:</i> $((\text{IOvalue AND Mask1}) = \text{Mask1}) \text{ AND } ((\text{not IOvalue}) \text{ AND Mask0}) = \text{Mask0}$ $(= \text{if the bits specified in Mask1 are 1 and the bits specified in Mask0 are 0}). \text{ Otherwise, the directly following list command is immediately executed.}$
<b>Call</b>	<code>sub_call_cond( Mask1, Mask0, Index )</code>
<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> <p>Index      Index of the to-be-called indexed subroutine. As an unsigned 32-bit value. Allowed value range: [0...1023].</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• See <a href="#">sub_call</a>.</li> <li>• See also <a href="#">Chapter 9.3.2 "Conditional Command Execution", page 271</a>.</li> </ul>
RTC4→RTC5	New command.
Version info	–
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 25</a>) from micro vector commands for runup curves and coast down curves can be seamlessly joined together with shapes from subroutines.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 538, OUT 538.
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 (as an unsigned 32-bit value) of the 16-bit digital input port to be evaluated for determining the jump distance. 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>, <b>switch_ioport</b> is replaced by a <b>list_nop (get_last_error</b> return code <b>RTC5_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 Port and 16-Bit Digital Output Port", page 65 and Chapter 9.3.2 "Conditional Command Execution", page 271.</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( \$0016 )</b>, that is, a relative list jump of length 22 list positions.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<b>list_jump_rel, list_jump_rel_cond</b>

<b>Ctrl Command</b>	<b>sync_slaves</b>
<b>Function</b>	Stably synchronizes (with the addressed RTC5 Board's $10\ \mu s$ clock) all slave boards connected in a master/slave chain to the addressed RTC5 Board's SLAVE connector.
<b>Call</b>	<code>sync_slaves()</code>
<b>Comments</b>	<ul style="list-style-type: none"> <li>For <code>sync_slaves</code> usage, see <a href="#">Chapter 6.6.3 "Master/Slave Operation", page 113</a>.</li> <li>SCANLAB recommends executing synchronization immediately after all boards have been initialized by <code>load_program_file</code> and <code>load_correction_file</code>. 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 <code>sync_slaves</code>. To avoid irregularities during execution of this command, you should neither apply external stop signals to the boards nor trigger <a href="#">External Starts</a> (these do not get automatically suppressed).</li> <li>During the course of <code>sync_slaves</code>, a <code>simulate_ext_stop</code> 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). It is the responsibility of users to ensure that a running process is not disrupted by <code>sync_slaves</code>.</li> <li>After execution of <code>sync_slaves</code>, the scan system axes of all involved boards are in either the coordinate center position <math>(0, 0 [,0])</math> or the HomeJump position (possibly shifted by an offset set by <code>set_offset</code>, <code>set_defocus</code> or <code>set_hi</code>).</li> <li><code>sync_slaves</code> (relating to synchronization) only affects RTC5 Boards connected in a master/slave chain to the addressed RTC5 Board's MASTER connector. It does not affect the addressed board itself or any boards connected to the addressed board's SLAVE connector. Therefore, if all slave boards of a master/slave chain shall be synchronized with the master board, then <code>sync_slaves</code> must address the master board of the master/slave chain. <code>sync_slaves</code> has no effect, if no board is connected to the Master connector of the addressed board.</li> <li>Synchronization of downstream slave boards by <code>sync_slaves</code> occurs even when the <a href="#">BUSY list execution status</a> of the board or <a href="#">INTERNAL-BUSY list execution status</a> 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 <code>sync_slaves</code> has no effect.</li> <li>During execution of <code>sync_slaves</code>, all <code>get_startstop_info</code> error bits are cleared on all boards (including upstream boards) allocated to the user program.</li> <li>For each downstream slave board, the <math>10\ \mu s</math> clock is interrupted for a short term and newly synchronized. Therefore, data transmission between the RTC5 and scan system may be disrupted for up to <math>20\ \mu s</math> per downstream board. Such disturbances could induce hard galvanometer scanner jumps if the actual output values do not correspond to <math>(0,0,[0])</math>.</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>RTC5_BUSY</b>), if:<ul style="list-style-type: none"><li>– the <b>BUSY</b> list execution status of the addressed board is currently set</li><li>– the <b>INTERNAL-BUSY</b> list execution 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> list execution status is set)</li></ul></li></ul>
RTC4→RTC5	New command.
Version info	–
References	<b>get_master_slave, get_sync_status</b>



Normal List Command	time_fix
Function	Stores the current time and date of the RTC clock/calender in a buffer for use with <b>mark_date</b> and <b>mark_time</b> .
Call	time_fix()
Comments	<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>
RTC4→RTC5	New command.
Version info	–
References	<b>time_fix_f</b> , <b>time_fix_f_off</b>

Normal List Command	time_fix_f
Function	Stores the current time and date of the RTC clock/calender in a buffer for use with <b>mark_date</b> and <b>mark_time</b> .
Call	time_fix_f( FirstDay )
Parameters	FirstDay      See <b>time_fix_f_off</b> .
Comments	<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>
RTC4→RTC5	New command.
Version info	–
References	<b>time_fix_f</b> , <b>time_fix_f_off</b> , <b>time_update</b> , <b>mark_time</b> , <b>mark_date</b>

Normal List Command	time_fix_f_off	
Function	Stores in a buffer 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> .	
Call	time_fix_f_off( FirstDay, Offset )	
Parameters	FirstDay      Defines the starting number for determining the Julian calendar day from the current date of the RTC calendar: counting proceeds from FirstDay to FirstDay + 364 (+1 for leap years). As an unsigned 32-bit value.	
	Offset      Forward dating. In seconds. As an unsigned 32-bit value. Allowed value range: [0...(2 <sup>32</sup> -1)].	
Comments	<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 RTC5 and PC time should be performed (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 Offset = 0, then the current date and current time are fixed. One practical use of forward dating (Offset &gt; 0) is for setting a date of expiry based on the current date. Backdating (Offset &lt; 0) is not possible.</li> </ul>	
RTC4→RTC5	New command.	
Version info	–	
References	<a href="#">time_fix</a> , <a href="#">time_fix_f</a> , <a href="#">time_update</a> , <a href="#">mark_time</a> , <a href="#">mark_date</a>	



<b>Ctrl Command</b>	<b>time_update</b>
<b>Function</b>	Sets the RTC5's 24-hour clock and calendar 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 <b>load_program_file</b>, if the 24-hour clock and calendar of the RTC5 are to be calibrated.</li> <li>• The base value for internal time counting is set to January 1, 2000, 00:00 by <b>load_program_file</b> whereas to the PC time by <b>time_update</b>. An internal seconds counter is set to 0 by <b>load_program_file</b> or by <b>time_update</b>, but is always driven by the quartz-controlled 10 <math>\mu</math>s clock.</li> <li>• Before marking with <b>mark_date</b> or <b>mark_time</b>, you must call <b>time_fix</b>, <b>time_fix_f</b> or <b>time_fix_f_off</b> 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> </ul>
RTC4→RTC5	New command.
<b>Version info</b>	–
<b>References</b>	<b>time_fix</b> , <b>time_fix_f</b> , <b>time_fix_f_off</b> , <b>mark_date</b> , <b>mark_time</b>



Normal List Command	timed_arc_abs								
Function	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 <b>Image field</b> .								
Call	timed_arc_abs( X, Y, Angle, T )								
Parameters	<table> <tr> <td>X</td><td>Like <b>arc_abs</b>.</td></tr> <tr> <td>Y</td><td>Like <b>arc_abs</b>.</td></tr> <tr> <td>Angle</td><td>Like <b>arc_abs</b>.</td></tr> <tr> <td>T</td><td>           Duration of the complete arc marking process. In <math>\mu\text{s}</math>.            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 <b>arc_abs</b>.         </td></tr> </table>	X	Like <b>arc_abs</b> .	Y	Like <b>arc_abs</b> .	Angle	Like <b>arc_abs</b> .	T	Duration of the complete arc marking process. In $\mu\text{s}$ . 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 <b>arc_abs</b> .
X	Like <b>arc_abs</b> .								
Y	Like <b>arc_abs</b> .								
Angle	Like <b>arc_abs</b> .								
T	Duration of the complete arc marking process. In $\mu\text{s}$ . 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 <b>arc_abs</b> .								
Comments	<ul style="list-style-type: none"> <li>Unlike <b>arc_abs</b>, <b>timed_arc_abs</b> 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 <b>Microsteps</b>) is adjusted so that the arc lasts as long as specified (see <a href="#">Chapter 8.9 "Timed Commands", page 250</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 <b>arc_abs</b>.</li> </ul>								
RTC4→RTC5	New command. See <b>arc_abs</b> .								
Version info	–								
References	<a href="#">arc_abs</a> , <a href="#">timed_arc_rel</a>								



Normal List Command	<b>timed_arc_rel</b>
Function	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 <b>Image field</b> .
Call	<code>timed_arc_rel( dX, dY, Angle, T )</code>
Parameters	<code>dX</code> Like <b>arc_rel</b> .
	<code>dY</code> Like <b>arc_rel</b> .
	<code>Angle</code> Like <b>arc_rel</b> .
	<code>T</code> Duration of the complete arc marking process. In $\mu\text{s}$ . 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_rel</b> behaves like <b>arc_rel</b> .
Comments	<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, <b>timed_arc_rel</b> is analogous to <b>timed_arc_abs</b> (see the comments there).</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>timed_arc_abs</b> , <b>arc_rel</b>



Normal List Command	<b>timed_jump_abs</b>						
Function	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 <b>Image field</b> .						
Call	<code>timed_jump_abs( X, Y, T )</code>						
Parameters	<table> <tr> <td>X</td><td>Like <b>jump_abs</b>.</td></tr> <tr> <td>Y</td><td>Like <b>jump_abs</b>.</td></tr> <tr> <td>T</td><td>Duration of the complete jump vector. In <math>\mu\text{s}</math>. 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 <b>jump_abs</b>.</td></tr> </table>	X	Like <b>jump_abs</b> .	Y	Like <b>jump_abs</b> .	T	Duration of the complete jump vector. In $\mu\text{s}$ . 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</b> behaves like <b>jump_abs</b> .
X	Like <b>jump_abs</b> .						
Y	Like <b>jump_abs</b> .						
T	Duration of the complete jump vector. In $\mu\text{s}$ . 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</b> behaves like <b>jump_abs</b> .						
Comments	<ul style="list-style-type: none"> <li>Unlike <b>jump_abs</b>, <b>timed_jump_abs</b> 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 <b>Microsteps</b>) is adjusted so that the vector lasts as long as specified, see <a href="#">Chapter 8.9 "Timed Commands", page 250</a>.</li> <li>The total jump 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 <b>jump_abs</b>.</li> </ul>						
RTC4→RTC5	Unchanged functionality. In addition: increased value range.  In <b>RTC4 Compatibility Mode</b> , the RTC5 multiplies the specified values for X and Y by 16. The allowed value ranges decrease accordingly.						
Version info	–						
References	<a href="#">jump_abs</a> , <a href="#">timed_jump_rel</a> , <a href="#">timed_jump_abs_3d</a>						



Normal List Command	<b>timed_jump_abs_3d</b>								
Function	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 a <b>3D image field</b> .								
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective jump speed in the xy plane.								
Call	<code>timed_jump_abs_3d( X, Y, Z, T )</code>								
Parameters	<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 <math>\mu\text{s}</math>. 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 $\mu\text{s}$ . 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 $\mu\text{s}$ . 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> .								
Comments	<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→RTC5	New command. See <b>jump_abs_3d</b> .								
Version info	–								
References	<b>timed_jump_abs</b> , <b>jump_abs_3d</b> , <b>timed_jump_rel_3d</b>								



Normal List Command	<b>timed_jump_rel</b>
Function	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 <b>Image field</b> .
Call	<code>timed_jump_rel( dX, dY, T )</code>
Parameters	<p><code>dX</code> Like <b>jump_rel</b>.</p> <p><code>dY</code> Like <b>jump_rel</b>.</p> <p><code>T</code> Duration of the complete jump vector. In <math>\mu\text{s}</math>. 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</b> behaves like <b>jump_rel</b>.</p>
Comments	<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>timed_jump_rel</b> is identical to <b>timed_jump_abs</b> (see the comments there).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. See <b>jump_rel</b> .
Version info	–
References	<b>timed_jump_abs</b> , <b>jump_rel</b> , <b>timed_jump_rel_3d</b>

Normal List Command	<b>timed_jump_rel_3d</b>
Function	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 a <b>3D image field</b> .
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
Call	<b>timed_jump_rel_3d( dx, dy, dz, T )</b>
Parameters	<p>dx      Like <b>jump_rel_3d</b>.</p> <p>dy      Like <b>jump_rel_3d</b>.</p> <p>dz      Like <b>jump_rel_3d</b>.</p> <p>T      Duration of the complete jump vector. In <math>\mu\text{s}</math>. 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>
Comments	<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>timed_jump_rel_3d</b> is identical to <b>timed_jump_abs_3d</b> (see the comments there).</li> </ul>
RTC4→RTC5	New command. See <b>jump_rel_3d</b> .
Version info	–
References	<b>timed_jump_abs_3d, jump_rel_3d, timed_jump_rel</b>



Normal List Command	<b>timed_mark_abs</b>						
Function	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 <b>Image field</b> .						
Call	<code>timed_mark_abs( X, Y, T )</code>						
Parameters	<table> <tr> <td>X</td><td>Like <b>mark_abs</b>.</td></tr> <tr> <td>Y</td><td>Like <b>mark_abs</b>.</td></tr> <tr> <td>T</td><td>Duration of the complete mark vector. In <math>\mu\text{s}</math>. 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 <b>mark_abs</b>.</td></tr> </table>	X	Like <b>mark_abs</b> .	Y	Like <b>mark_abs</b> .	T	Duration of the complete mark vector. In $\mu\text{s}$ . 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 <b>mark_abs</b> .
X	Like <b>mark_abs</b> .						
Y	Like <b>mark_abs</b> .						
T	Duration of the complete mark vector. In $\mu\text{s}$ . 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 <b>mark_abs</b> .						
Comments	<ul style="list-style-type: none"> <li>Unlike <b>mark_abs</b>, the <b>timed_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 <b>Microsteps</b>) is adjusted so that the vector lasts as long as specified (see <b>Chapter 8.9 "Timed Commands", page 250</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>See also comments on <b>mark_abs</b>.</li> </ul>						
RTC4→RTC5	Unchanged functionality. In addition: increased value range. See <b>mark_abs</b> .						
Version info	–						
References	<b>mark_abs</b> , <b>timed_mark_rel</b> , <b>timed_mark_abs_3d</b>						



Normal List Command	<a href="#">timed_mark_abs_3d</a>								
Function	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 a <b>3D image field</b> .								
Restriction	If the <b>Option "3D"</b> is not enabled or no 3D correction table has been assigned (see <a href="#">select_cor_table</a> ), then <b>timed_mark_abs_3d</b> has the same effect as <b>timed_mark_abs</b> . However, split-up into <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.								
Call	<code>timed_mark_abs_3d( X, Y, Z, T )</code>								
Parameters	<table> <tr> <td>X</td><td>Like <a href="#">mark_abs_3d</a>.</td></tr> <tr> <td>Y</td><td>Like <a href="#">mark_abs_3d</a>.</td></tr> <tr> <td>Z</td><td>Like <a href="#">mark_abs_3d</a>.</td></tr> <tr> <td>T</td><td>Duration of the complete mark vector. In <math>\mu\text{s}</math>. 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 <a href="#">mark_abs_3d</a>.</td></tr> </table>	X	Like <a href="#">mark_abs_3d</a> .	Y	Like <a href="#">mark_abs_3d</a> .	Z	Like <a href="#">mark_abs_3d</a> .	T	Duration of the complete mark vector. In $\mu\text{s}$ . 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 <a href="#">mark_abs_3d</a> .
X	Like <a href="#">mark_abs_3d</a> .								
Y	Like <a href="#">mark_abs_3d</a> .								
Z	Like <a href="#">mark_abs_3d</a> .								
T	Duration of the complete mark vector. In $\mu\text{s}$ . 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 <a href="#">mark_abs_3d</a> .								
Comments	<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 <a href="#">mark_abs_3d</a>.</li> </ul>								
RTC4→RTC5	New command. See <a href="#">mark_abs_3d</a> .								
Version info	–								
References	<a href="#">timed_mark_abs</a> , <a href="#">mark_abs_3d</a> , <a href="#">timed_mark_rel_3d</a>								



Normal List Command	<b>timed_mark_rel</b>
Function	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 <b>Image field</b> .
Call	<code>timed_mark_rel( dX, dY, T )</code>
Parameters	<p><code>dX</code> Like <b>mark_rel</b>.</p> <p><code>dY</code> Like <b>mark_rel</b>.</p> <p><code>T</code> Duration of the complete mark vector. In <math>\mu\text{s}</math>. 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 <b>mark_rel</b>).</p>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the mark vector end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>timed_mark_rel</b> is analogous to <b>timed_mark_abs</b> (see the comments there).</li> </ul>
RTC4→RTC5	Unchanged functionality. In addition: increased value range. See <b>mark_rel</b> .
Version info	–
References	<b>timed_mark_abs</b> , <b>mark_rel</b> , <b>timed_mark_rel_3d</b>



Normal List Command	<b>timed_mark_rel_3d</b>
Function	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 a <b>3D image field</b> .
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
Call	<code>timed_mark_rel_3d( dx, dy, dz, T )</code>
Parameters	<p>dx      Like <b>mark_rel_3d</b>.</p> <p>dy      Like <b>mark_rel_3d</b>.</p> <p>dz      Like <b>mark_rel_3d</b>.</p> <p>T      Duration of the complete mark vector. In <math>\mu\text{s}</math>. 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>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the mark vector end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>timed_mark_rel_3d</b> is identical to <b>timed_mark_abs_3d</b> (see the comments there).</li> </ul>
RTC4→RTC5	New command.. See <b>mark_rel_3d</b> .
Version info	–
References	<b>timed_mark_abs_3d</b> , <b>timed_mark_abs</b> , <b>mark_rel_3d</b> , <b>timed_mark_rel</b>

Normal List Command	<b>timed_para_jump_abs</b>								
Function	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 <b>Image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_ctrl</b> to the specified value.								
Call	<code>timed_para_jump_abs( X, Y, P, T )</code>								
Parameters	<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 <math>\mu\text{s}</math>. 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 $\mu\text{s}$ . 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 $\mu\text{s}$ . 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> .								
Comments	<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 <b>Microsteps</b>) is adjusted so that the vector lasts as long as specified (see <b>Chapter 8.9 "Timed Commands", page 250</b>). 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\text{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→RTC5	New command. See <b>para_jump_abs</b> .								
Version info	–								
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>								

Multiple List Command	<b>timed_para_jump_abs_3d</b>										
Function	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 a <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.										
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective jump speed in the xy plane.										
Call	<b>timed_para_jump_abs_3d( X, Y, Z, P, T )</b>										
Parameters	<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 <math>\mu\text{s}</math>. 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 $\mu\text{s}$ . 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 $\mu\text{s}$ . 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> .										
Comments	<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→RTC5	New command. See <b>para_jump_abs_3d</b> .										
Version info	–										
References	<b>timed_para_jump_abs</b> , <b>para_jump_abs_3d</b> , <b>jump_abs_3d</b> , <b>timed_para_jump_rel_3d</b>										



Normal List Command	<a href="#">timed_para_jump_rel</a>
Function	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 <b>Image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
Call	<code>timed_para_jump_rel( dx, dy, p, T )</code>
Parameters	<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 <math>\mu\text{s}</math>. 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>
Comments	<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>timed_para_jump_rel</b> is analogous to <a href="#">timed_para_jump_abs</a> (see the comments there).</li> </ul>
RTC4→RTC5	New command. See <a href="#">para_jump_rel</a> .
Version info	–
References	<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>

Multiple List Command	timed_para_jump_rel_3d
Function	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 a <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.
Restriction	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_rel_3d</b> has the same effect as <b>timed_para_jump_rel</b> . However, split-up into <b>Microsteps</b> is calculated like a 3D command and hence influences the effective jump speed in the xy plane.
Call	timed_para_jump_rel_3d( dX, dY, dZ, P, T )
Parameters	<p>dX      Like <b>para_jump_rel_3d</b>.</p> <p>dY      Like <b>para_jump_rel_3d</b>.</p> <p>dZ      Like <b>para_jump_rel_3d</b>.</p> <p>P      Like <b>para_jump_rel_3d</b>.</p> <p>T      Duration of the complete jump vector. In <math>\mu\text{s}</math>. 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_3d</b> behaves like <b>para_jump_rel_3d</b>.</p>
Comments	<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>timed_para_jump_rel_3d</b> is analogous to <b>timed_para_jump_abs_3d</b> (see comments there).</li> <li>For <math>T \geq 5</math>, <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>
RTC4→RTC5	New command. See <b>para_jump_rel_3d</b> .
Version info	–
References	<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>

Normal List Command	<code>timed_para_mark_abs</code>								
Function	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 <b>Image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.								
Call	<code>timed_para_mark_abs( X, Y, P, T )</code>								
Parameters	<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 <math>\mu\text{s}</math>. 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 $\mu\text{s}$ . 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 $\mu\text{s}$ . 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> .								
Comments	<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 <b>Microsteps</b>) is adjusted so that the vector lasts as long as specified (see <b>Chapter 8.9 "Timed Commands"</b>, page 250). 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 for <math>T \geq 5</math>. 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\text{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_mark_abs</b>.</li> </ul>								
RTC4→RTC5	New command. See <b>para_mark_abs</b> .								
Version info	–								
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>								



Multiple List Command	timed_para_mark_abs_3d										
Function	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 a <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.										
Restriction	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, split-up into <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.										
Call	timed_para_mark_abs_3d( X, Y, Z, P, T )										
Parameters	<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 <math>\mu\text{s}</math>. 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 $\mu\text{s}$ . 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 $\mu\text{s}$ . 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> .										
Comments	<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 for <math>T \geq 5</math>. 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→RTC5	New command. See <b>para_mark_abs_3d</b> .										
Version info	–										
References	<b>timed_para_mark_abs</b> , <b>para_mark_abs_3d</b> , <b>mark_abs_3d</b> , <b>timed_para_mark_rel_3d</b>										



Normal List Command	timed_para_mark_rel
Function	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 <b>Image field</b> and, simultaneously as well as linearly, changes the signal parameter selected by <b>set_vector_control</b> to the specified value.
Call	timed_para_mark_rel( dx, dy, p, T )
Parameters	<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 <math>\mu</math>s. 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</b> behaves like <a href="#">para_mark_rel</a>.</p>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the mark vector end point are to be supplied as relative coordinates with respect to the current position. Otherwise, <b>timed_para_mark_rel</b> 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→RTC5	New command. See <a href="#">para_mark_rel</a> .
Version info	–
References	<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>

Multiple List Command	timed_para_mark_rel_3d
Function	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 a <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.
Restriction	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 <b>Microsteps</b> is calculated like a 3D command and hence influences the effective mark speed in the xy plane.
Call	timed_para_mark_rel_3d( dX, dY, dZ, P, T )
Parameters	<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 <math>\mu\text{s}</math>. 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>
Comments	<ul style="list-style-type: none"> <li>The coordinates for the mark vector 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 for <math>T \geq 5</math>. 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→RTC5	New command. See <b>para_mark_rel_3d</b> .
Version info	–
References	<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>	transform	
<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>	Sig1	Parameter: to-be-transformed position value. As a pointer to a signed 32-bit value.  Returned parameter value: transformed position value. As a signed 32-bit value. The input values are overwritten.
	Sig2	Like Sig1 (analogously).
<b>Parameters</b>	Ptr	Pointer (in C and C++ data type ULONG_PTR, an unsigned 32-bit value or unsigned 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> .
	Code	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.  The parameter's meaning is similar to that of <b>get_transform</b> (Sig1 corresponds to <b>Ptr1</b> and Sig2 to <b>Ptr2</b> ).  If Bit #0 = 0, then both supplied position values (Sig1 and Sig2) are backward transformed as xy coordinates:  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 <b>Image field</b> 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 <code>Sig1</code> or <code>Sig2</code>) is backward transformed as the z coordinate:</p> <p>Bit #1 = 0: The value supplied by <code>Sig1</code> is backward transformed as the z coordinate (<code>Sig2</code> remains unchanged).          Bit #1 = 1: The value supplied by <code>Sig2</code> is backward transformed as the z coordinate (<code>Sig1</code> remains unchanged).</p> <p>Bit #2 = 0: The offset to the focal length defined by <code>set_defocus</code> or <code>set_defocus_list</code> is backward transformed.</p> <p>Bit #3 = 0: The ABC correction is backward transformed.</p> <p>Bit #4 = 0: The offset to the z coordinate defined by <code>set_offset_xyz</code> or <code>set_offset_xyz_list</code> is backward transformed.</p> <p>Bit #5 Reserved.</p> <p>...</p> <p>Bit #31</p>												
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>Success.</td> </tr> <tr> <td>1</td> <td><code>Ptr</code> = <code>NULL</code> (no memory area specified).</td> </tr> <tr> <td>2</td> <td>No valid data at <code>Ptr</code> (<code>upload_transform</code> did not execute).</td> </tr> <tr> <td>3</td> <td>Erroneous data at <code>Ptr</code> (a corresponding error indication has been stored by <code>upload_transform</code>).</td> </tr> <tr> <td>4</td> <td>z axis inversion not possible.</td> </tr> </tbody> </table>	Value	Description	0	Success.	1	<code>Ptr</code> = <code>NULL</code> (no memory area specified).	2	No valid data at <code>Ptr</code> ( <code>upload_transform</code> did not execute).	3	Erroneous data at <code>Ptr</code> (a corresponding error indication has been stored by <code>upload_transform</code> ).	4	z axis inversion not possible.
Value	Description												
0	Success.												
1	<code>Ptr</code> = <code>NULL</code> (no memory area specified).												
2	No valid data at <code>Ptr</code> ( <code>upload_transform</code> did not execute).												
3	Erroneous data at <code>Ptr</code> (a corresponding error indication has been stored by <code>upload_transform</code> ).												
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", page 200</a>.</li> <li>The execution of <code>transform</code> must be preceded by a call to <code>upload_transform</code>. Additionally, position values should have been requested by <code>get_values</code>.</li> <li>If execution of <code>transform</code> results in an error (returned error code &gt; 0), then no transformation occurs (<code>Sig1</code> and <code>Sig2</code> then remain unchanged). Errors also include <code>Ptr</code> = <code>NULL</code> (error code = 1) or errors resulting from prior, erroneous execution of <code>upload_transform</code> (error code = 3).</li> <li>If backward transformation of z values is requested (Code Bit #0 = 1), but only a 2D correction table has been assigned at the point in time 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 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>												



<b>Ctrl Command</b>	<b>transform</b>
Comments (cont'd)	<ul style="list-style-type: none"> <li>Because <b>transform</b> does not access any RTC5 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 RTC5 Board on the PCI bus).</li> <li><b>transform</b> is not available as a multi-board command.</li> <li>The board-specific error variables <a href="#">LastError</a> and <a href="#">AccError</a> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <b>transform</b>.</li> </ul>
RTC4→RTC5	<p>New command.</p> <p>In the <a href="#">RTC4 Compatibility Mode</a>, all back transformed values (including z values) are in the RTC5 20-bit range.</p>
Version info	–
References	<a href="#">upload_transform</a> , <a href="#">get_transform</a> , <a href="#">get_values</a>

<b>Ctrl Command</b>	<b>upload_transform</b>
<b>Function</b>	Transfers from the RTC5 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 value or unsigned 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> <p>Bit #0      =1:    X gain = 0 (gain of automatic self-calibration for <a href="#">Galvanometer scanner 2</a>).</p> <p>Bit #1      =1:    Y gain = 0 (gain of automatic self-calibration for <a href="#">Galvanometer scanner 1</a>).</p> <p>Bit #2      =1:    The total matrix of the defined coordinate transformation is noninvertable.</p> <p>Bit #3      =1:    No correction table assigned.</p> <p>Bit #4      =1:    The ABC values (z axis) are noninvertable.</p> <p>Bit #5      =1:    Error querying correction table.</p> <p>Bit #6      =1:    Parameter error: invalid <code>HeadNo</code> or <code>Ptr</code> = 0.</p> <p>Bit #7      =1:    Busy error, board has <a href="#">BUSY list execution status</a> or <a href="#">INTERNAL-BUSY list execution status</a> (<a href="#">get_last_error</a> return code <code>RTC5_BUSY</code>).</p> <p>Bit #8      Reserved.</p> <p>...</p> <p>Bit #31</p>
<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 <a href="#">transform</a> or <a href="#">get_transform</a> (see also <a href="#">Chapter 8.1.3 "Monitoring the Positioning", page 200</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. <a href="#">transform</a> and <a href="#">get_transform</a> recognize this error information and ensure that the backward transformation is not executed (<a href="#">transform</a> then generates a corresponding error code, <a href="#">get_transform</a> generates a <a href="#">get_last_error</a> return code <code>RTC5_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 (<b>get_last_error</b> return code <b>RTC5_BUSY</b>), if:           <ul style="list-style-type: none"> <li>– the <b>BUSY</b> list execution status is set</li> <li>– the <b>INTERNAL-BUSY</b> list execution status 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 <b>set_wait</b> (<b>PAUSED</b> list execution status set)</li> </ul> </li> <li>• During the runtime of <b>upload_transform</b>, <b>External Starts</b> are suppressed.</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<b>transform</b> , <b>get_transform</b>



<b>Ctrl Command</b>	<b>verify_checksum</b>
<b>Function</b>	Checks or creates the checksum of a correction file.
<b>Call</b>	<code>verify_checksum( Name )</code>
<b>Parameters</b>	<p>Name      Name of the correction file.  As a pointer to a \0-terminated ANSI string.</p>
<b>Result</b>	<p>Result of the test.  As an unsigned 32-bit value.  = 0:    No error (the tested checksum is OK.).  = 1:    A checksum has been newly created (no info about file integrity).  = 2:    The tested checksum is incorrect.  = 3:    A checksum could not be determined (file error, etc.).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>Verification of correction file downloads only works for files that contain a checksum (see <a href="#">Loading of correction files, page 120</a> and <a href="#">set_verify</a>).</li> <li><b>verify_checksum</b> is available even without explicit access rights to a particular RTC5 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 119</a>) are neither generated nor altered by <b>verify_checksum</b>.</li> </ul>
<b>RTC4→RTC5</b>	New command.
<b>Version info</b>	–
<b>References</b>	<a href="#">set_verify</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>	<table> <tr> <td>Value</td> <td>Count. As a signed 32-bit value. Allowed value range: <math>[-2^{31} \dots + (2^{31}-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> </table>	Value	Count. As a signed 32-bit value. Allowed value range: $[-2^{31} \dots + (2^{31}-1)]$	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".
Value	Count. As a signed 32-bit value. Allowed value range: $[-2^{31} \dots + (2^{31}-1)]$				
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".				
<b>Comments</b>	<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→RTC5	New command.				
Version info	–				
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 value. As a signed 32-bit value. 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 274</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 <a href="#">Signals for "Laser Active" Operation</a> (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→RTC5	New command.
Version info	–
References	<a href="#">wait_for_encoder_mode</a> , <a href="#">park_position</a> , <a href="#">park_return</a>

<b>Normal 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^{31} \dots + (2^{31}-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: $[-2^{31} \dots + (2^{31}-1)]$	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: $[-2^{31} \dots + (2^{31}-1)]$						
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 274</a>.</li> <li>If <code>Mode</code> = 0, 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</code> &gt; 0, <code>wait_for_encoder_mode</code> 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</code> ≠ 0, 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</code> &gt; 1, then <code>wait_for_encoder_mode</code> is replaced with a <a href="#">list_nop</a> (<code>get_last_error</code> return code <code>RTC5_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 <a href="#">External Starts</a>, <a href="#">set_wait</a> or <a href="#">if_cond</a>, 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</code> = 0, <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 <a href="#">Signals for "Laser Active" Operation</a> (for example, a <code>list_nop</code>).</li> </ul>						



Normal List Command	<code>wait_for_encoder_mode</code>
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 <code>set_fly_2d</code>, but otherwise remain stationary.</li></ul>
Version info	–
RTC4→RTC5	New command.
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>

Normal List Command	<b>wait_for_mcbsp</b>
Function	Waits until the input value at the <b>McBSP interface</b> has reached, overstepped or understepped the specified value for the first time.
Call	<code>wait_for_mcbsp( Axis, Value, Mode )</code>
Parameters	<p>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, <b>Galvanometer scanner 2</b>)                = 1: upper half-word (y axis, <b>Galvanometer scanner 1</b>)</p> <p>Value      Limit value.                As a signed 32-bit value.</p> <p>Mode      Mode.                As a signed 32-bit value.                = 0: Waits for equality.                &gt; 0: Waits for overstepping.                &lt; 0: Waits for understepping.</p>
Comments	<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/SPI Signals", page 276</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 (get_last_error</b> return code <b>RTC5_PARAM_ERROR</b><b>).</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 <b>External Starts</b>, <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 <b>Signals for "Laser Active" Operation</b> (for example, a <b>list_nop</b>).</li> </ul>
RTC4→RTC5	New command.
Version info	–
References	<a href="#">wait_for_encoder_mode</a>



<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 259.</a></li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
<b>References</b>	<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      Like <a href="#">write_8bit_port</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>• Like <a href="#">write_8bit_port</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
<b>References</b>	<a href="#">write_8bit_port</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>	ErrorNo = write_abc_to_file( Name, A, B, C )	
<b>Parameters</b>	<p>Name      Name of the correction file. As a pointer to a \0-terminated ANSI string.</p> <p>A      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 value range:: see <a href="#">load_z_table</a>.</p> <p>B      Like A (analogously).</p> <p>C      Like A (analogously).</p>	
<b>Result</b>	ErrorNo	<p>Error code. As an unsigned 32-bit value.</p> <p>0      No error.</p> <p>1      A exceeded the maximum allowed value.</p> <p>2      A undercut the minimum allowed value.</p> <p>4      B exceeded the maximum allowed value.</p> <p>8      B undercut the minimum allowed value.</p> <p>16     C exceeded the maximum allowed value.</p> <p>32     C undercut the minimum allowed value.</p> <p>3      File-open error (empty string, file not found etc.).</p> <p>12     File error (checksum could not be determined, file corrupt).</p>
<b>Comments</b>	<ul style="list-style-type: none"> <li>• <b>write_abc_to_file</b> is available even without explicit access rights to a specific RTC5 board.</li> <li>• <b>write_abc_to_file</b> is not available as a multi-board command.</li> <li>• The board-specific error variables <a href="#">LastError</a> and <a href="#">AccError</a> (see <a href="#">Chapter 6.8 "Error Handling", page 119</a>) are neither generated nor altered by <b>write_abc_to_file</b>.</li> <li>• If the error code is not 0 or 12, the ABC values are not outputted.</li> </ul>	
RTC4→RTC5	New command.	
Version info	Available as of DLL 538, OUT 538.	
References	<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 <b>ANALOG OUT1</b> analog output port.      As an unsigned 32-bit value.      Bits with higher significance 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→RTC5	Unchanged functionality. In <b>RTC4 Compatibility Mode</b> : like <a href="#">write_da_x</a>
<b>Version info</b>	–
<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>	Value      Like <a href="#">write_da_1</a> .
RTC4→RTC5	Unchanged functionality. In <b>RTC4 Compatibility Mode</b> : like <a href="#">write_da_x</a>
<b>Version info</b>	–
<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 <b>ANALOG OUT2</b> analog output port.  As an unsigned 32-bit value.  Bits with higher significance 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→RTC5	Unchanged functionality. In <b>RTC4 Compatibility Mode</b> : like <a href="#">write_da_x</a>
<b>Version info</b>	–
<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>	Value      Like <a href="#">write_da_2</a> .
RTC4→RTC5	Unchanged functionality. In <b>RTC4 Compatibility Mode</b> : like <a href="#">write_da_x</a> .
<b>Version info</b>	–
<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 RTC5.
<b>Call</b>	<code>write_da_x( x, Value )</code>
<b>Parameters</b>	<p><b>x</b> Number of the analog output port. As an unsigned 32-bit value. Allowed value range: [1, 2] (1: ANALOG OUT1, 2: ANALOG OUT2)</p> <p><b>Value</b> 12-bit output value for the selected analog output port. As an unsigned 32-bit value. Bits with higher significance 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 Port 1 and 2", page 259</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 (<a href="#">get_last_error</a> return code <code>RTC5_PARAM_ERROR</code>).</li> <li><code>write_da_1</code> / <code>write_da_2</code> can be used alternatively to <code>write_da_x</code> (without parameter <math>x</math>).</li> </ul>
<b>RTC4→RTC5</b>	Unchanged functionality. In <a href="#">RTC4 Compatibility Mode</a> , the RTC5 multiplies the specified <code>value</code> by 4.
<b>Version info</b>	–
<b>References</b>	<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 value range: [1, 2] (1: <a href="#">ANALOG OUT1</a>, 2: <a href="#">ANALOG OUT2</a>)</p> <p>Value      12-bit output value for the selected analog output port.            As an unsigned 32-bit value. Bits with higher significance 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>, <code>write_da_x_list</code> is replaced with a <a href="#">list_nop</a> (<a href="#">get_last_error</a> return code <a href="#">RTC5_PARAM_ERROR</a>).</li> </ul>
RTC4→RTC5	Unchanged functionality. In <a href="#">RTC4 Compatibility Mode</a> : like <a href="#">write_da_x</a> .
Version info	–
References	<a href="#">write_da_x</a>

<b>Ctrl Command</b>	<b>write_hi_pos</b>
<b>Function</b>	Writes the specified values to the <b>EEPROM</b> 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> <p>0           Kein Fehler.</p> <p>Bit #0      =1:     Wrong HeadNo.</p> <p>Bit #1      =1:     Wrong sensor position for X1.</p> <p>Bit #2      =1:     Wrong sensor position for X2.</p> <p>Bit #3      =1:     Wrong sensor position for Y1.</p> <p>Bit #4      =1:     Wrong sensor position for Y2.</p> <p>Bit #5      =1:     Invalid ASC version.</p>
<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 RTC5 Board "A" to RTC5 Board "B" in order to transfer its Home-In reference positions from RTC5 Board "A" to RTC5 Board "B".</li> <li>• Use <code>get_hi_pos( HeadNo )</code> to read out the Home-In reference positions of RTC5 Board "A" (only after <code>init_rtc5_dll</code>, see <b>get_hi_pos</b>).</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, <code>auto_cal( Command = 4 )</code> (or <code>auto_cal( Command = 0 )</code> because this command implicitly executes <code>auto_cal( Command = 4 )</code> must have been successfully executed at last on the RTC5 Board "B".                    A cross-check with <code>get_auto_cal( HeadNo )</code> needs to return 100. Otherwise, <b>write_hi_pos</b> would return the error <b>Bit #5 = 1</b>.                    If required, connect the scan head to RTC5 Board "B" and execute <code>auto_cal( HeadNo, 4 )</code> before executing <b>write_hi_pos</b>.</li> </ul>
RTC4→RTC5	New command.
Version info	Available as of DLL 538, OUT 538.
References	<b>get_hi_pos, get_auto_cal, auto_cal</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>	<p>Value      16-bit output value (DIGITAL OUT0...DIGITAL OUT15).      As an unsigned 32-bit value.      Only the least significant 16 bits are evaluated.</p>
<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 digital <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 <a href="#">Chapter 9.1.1 "16-Bit Digital Output Port", page 258</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
<b>References</b>	<a href="#">write_io_port_list</a> , <a href="#">write_io_port_mask</a> , <a href="#">write_io_port_mask_list</a> , <a href="#">set_io_cond_list</a> , <a href="#">clear_io_cond_list</a> , <a href="#">get_io_status</a>

<b>Delayed Short List Command</b>	<b>write_io_port_list</b>
<b>Function</b>	Like <a href="#">write_io_port</a> , but a list command.
<b>Call</b>	<code>write_io_port_list( Value )</code>
<b>Parameters</b>	Value      Like <a href="#">write_io_port</a> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <a href="#">write_io_port</a>.</li> </ul>
RTC4→RTC5	Unchanged functionality.
Version info	–
<b>References</b>	<a href="#">write_io_port</a>

<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> = 0xFFFF, <code>write_io_port_mask</code> behaves like <code>write_io_port</code>.</li> <li>See also Chapter 9.1.1 "16-Bit Digital Output Port", page 258.</li> </ul>	
RTC4→RTC5	New command.	
Version info	–	
References	<a href="#">write_io_port</a> , <a href="#">write_io_port_mask_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>	Like <code>write_io_port_mask</code> .
	<code>Mask</code>	Like <code>write_io_port_mask</code> .
<b>Comments</b>	<ul style="list-style-type: none"> <li>See <code>write_io_port_mask</code>.</li> </ul>	
RTC4→RTC5	New command.	
Version info	–	
References	<a href="#">write_io_port_mask</a> , <a href="#">write_io_port_list</a>	



## 10.3 Unsupported RTC2/RTC3/RTC4 Commands

Some RTC3/RTC4 commands and a few RTC2 commands emulated by the RTC3/RTC4 are not supported by the RTC5. But most of these can be replaced by RTC5 commands. The following table lists all unsupported commands and appropriate RTC5 replacements.

<b>Ctrl Command</b>	<b>aut_change</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>auto_change</b>

<b>Ctrl Command</b>	<b>dsp_start</b>
Support status	This RTC2/RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	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>List Command</b>	<b>field_jump</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>home_position</b> , the “field jump” is automatically executed.

<b>Ctrl Command</b>	<b>get_RTC2_mode</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	The RTC5 has no need to query the mode, which is now be set by the software command <b>set_laser_mode</b> instead of hardware.

<b>Ctrl Command</b>	<b>get_RTC2_version</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>get_RTC_version</b>

<b>Ctrl Command</b>	<b>get_version</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>get_hex_version</b>



<b>Ctrl Command</b>	<b>get_xy_pos</b>
Support status	This RTC2/RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	Replaced by <b>get_value</b>

<b>Ctrl Command</b>	<b>get_xyz_pos</b>
Support status	This RTC2/RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	Replaced by <b>get_value</b>

<b>List Command</b>	<b>home_jump</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>home_position</b> , the “home jump” is automatically executed.

<b>List Command</b>	<b>laser_on</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>laser_on_list</b>

<b>Ctrl Command</b>	<b>load_cor</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>load_correction_file</b>

<b>Ctrl Command</b>	<b>load_pro</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>load_program_file</b>

<b>List Command</b>	<b>pola_abs, polb_abs, polc_abs</b>
Support status	These RTC2 commands are not supported by the RTC5.
Replaced by	<b>mark_abs</b>

<b>Ctrl Command</b>	<b>read_pixel_ad</b>
Support status	This RTC2/RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	There is no equivalent command for the RTC5.



<b>Ctrl Command</b>	<b>rtc3_count_cards</b>
Support status	This RTC3 command is not supported by the RTC5.
Replaced by	<b>rtc5_count_cards</b>

<b>Ctrl Command</b>	<b>rtc4_count_cards</b>
Support status	This RTC4 command is not supported by the RTC5.
Replaced by	<b>rtc5_count_cards</b>

<b>Ctrl Command</b>	<b>select_list</b>
Support status	This RTC2/RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	<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, <b>Mem1</b> is the memory size of "List 1" (and the absolute start address of "List 2"). After initialization (with <b>load_program_file</b> ), <b>Mem1</b> is 4000, otherwise as set by <b>config_list</b> . <b>Mem1</b> 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_base</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	There is no equivalent command for the RTC5 (because it is a plug-and-play board whose base address is automatically set).

<b>Ctrl Command</b>	<b>set_co2_standby</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>set_standby</b>

<b>List Command</b>	<b>set_co2_standby_list</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>set_standby_list</b>

<b>List Command</b>	<b>set_delays</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>set_laser_delays, set_scanner_delays</b>

<b>List Command</b>	<b>set_encoder_values</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>set_fly_x, set_fly_y</b>



<b>Ctrl Command</b>	<b>set_gain</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>set_matrix, set_offset</b>

<b>Ctrl Command</b>	<b>set_list_mode</b>
Support status	This RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	See <a href="#">Chapter 6.5.4 "RTC4 Circular Queue Mode", page 110.</a>

<b>Ctrl Command</b>	<b>set_mode</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	There is no equivalent command for the RTC5, because its <b>Image field</b> correction algorithm is always enabled. Instead, a 1to1 correction file can be loaded.

<b>Ctrl Command</b>	<b>set_piso_control</b>
Support status	This RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	<p><b>set_piso_control</b> is not needed for operation of the RTC5.</p> <p>For data transfer in accordance with the SL2-100 protocol, the bi-directional communication between the scan system and the RTC5 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 <a href="#">XY2-100 Converter (Accessory)</a>.</p>

<b>Ctrl Command</b>	<b>set_speed</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>set_jump_speed, set_mark_speed</b>

<b>Ctrl Command</b>	<b>set_wobble_xy</b>
Support status	This RTC4 command is not supported by the RTC5.
Replaced by	<b>set_wobble, set_wobble_mode</b>

<b>Ctrl Command</b>	<b>set_yag_parameter</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	There is no equivalent command for the RTC5.



<b>Ctrl Command</b>	<b>write_da</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>wwrite_da_1</b>

<b>List Command</b>	<b>write_da_list</b>
Support status	This RTC2 command is not supported by the RTC5.
Replaced by	<b>wwrite_da_1_list</b>

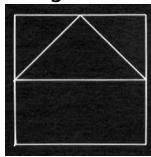
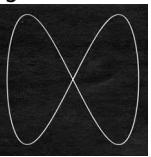
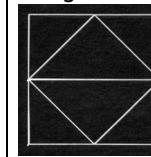
<b>Ctrl Command</b>	<b>z_out</b>
Support status	This RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	For setting the z value in a 3-axis scan system, <b>set_defocus</b> can be used (only together with an RTC5 with enabled <b>Option "3D"</b> ). After <b>load_z_table</b> (0.0, 1.0, 0.0), <b>set_defocus(z)</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 RTC3/RTC4 command is not supported by the RTC5.
RTC4→RTC5	After <b>load_z_table</b> (0.0, 1.0, 0.0), <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

The RTC5 software package contains various program code samples in the **Folder DemoFiles**. They demonstrate **RTC5 DLL** and RTC5 initialization, error handling and usage of the diverse control commands and list commands.

They contain the required calling sequences of RTC5 commands, which you can easily translate into your preferred programming language. The following table summarizes the characteristics of the demo programs.

File name	Demo1	Demo2	Demo3	Demo4	Demo5	Demo6	Demo7
Marking Task	Square and triangle 	Lissajous figures 	Archimedean spirals 	Raster image reproduction 	Squares and triangles 	Raster image reproduction, Processing-on-the-fly	Text output, intelliSCAN
Laser type	CO <sub>2</sub>	YAG	YAG	CO <sub>2</sub>	YAG	CO <sub>2</sub>	YAG
Marking method	vector	vector	vector	Pixel mode	vector	Pixel mode, fly mode	vector
Additional features			home jump		home jump	data recording, time measurement, 16-bit IO port	data recording, time measurement, iDRIVE functions
DLL linking	implicit	explicit	explicit	explicit	explicit	implicit	implicit
List handling	use of a single list	single list, continuous transfer (circular queue)	two alternating lists, continuous transfer	two alternating lists, continuous transfer	use of two lists	two alternating lists, continuous transfer	two lists (default), protected area
External control inputs					/START, /STOP	allowed	
Exception handling	load_list	get_status, set_wait, release_wait, pause_list, restart_list, stop_execution	get_status, load_list, pause_list, restart_list, stop_execution	load_list	get_status, load_list, stop_execution	get_status, load_list, stop_execution	get_status
Other commands	config_list, set_end_of_list, execute_list_pos	config_list, set_start_list_pos, execute_list_pos, set_end_of_list	config_list, set_start_list_pos, set_end_of_list, execute_list_pos, auto_change	config_list, set_end_of_list, execute_list_pos, auto_change, set_pixel_line, set_n_pixel	config_list, set_end_of_list, execute_list_pos, set_extstartpos, set_control_mode, get_startstop_info, bounce_supp	config_list, set_end_of_list, execute_list_pos, auto_change, set_control_mode, simulate_encoder, set_pixel_line, set_n_pixel, set_fly_x, fly_return, set_trigger, save_and_restart_timer, write_io_port_list, get_io_status	control_command, load_char, mark_text, mark_text_abs, set_start_list_pos, set_trigger, measurement_stat us, get_waveform, set_start_list_pos, save_and_restart_timer, set_end_of_list, execute_list_pos

## 12 Troubleshooting

Problem	Remedy
PC does not boot	<p>Switch off the PC and check the following:</p> <ul style="list-style-type: none"> <li>Check if the RTC5 Board is correctly seated in the PCI 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 of the RTC5.</li> <li>Check for loose cables or connectors</li> </ul>
RTC5 does not respond	<ul style="list-style-type: none"> <li>Check the RTC5 board driver installation. See <a href="#">Chapter 5.4 "Installing the RTC5 Software", page 80</a>.</li> <li>Use the included HPGL converter program to check if the board can be properly accessed.</li> </ul> <p>If not: Check the cable type and the cable length. If the scan system is controlled by an <a href="#">XY2-100 Converter (Accessory)</a>, then check, whether the solder jumpers of the <a href="#">XY2-100 Converter (Accessory)</a> are set appropriate for the cable length. See <a href="#">Figure 14</a>.</p> <p>If yes: Check the <a href="#">RTC5 DLL</a> import declarations in your user program. See <a href="#">Chapter 6.2.2 "Importing Commands", page 84</a>.</p>
User program fails	<ul style="list-style-type: none"> <li>Check the driver installation. See <a href="#">Chapter 5.4 "Installing the RTC5 Software", page 80</a>.</li> <li>Check the RTC5 initialization in the user program.</li> <li>Check the <a href="#">RTC5 DLL</a> import declarations in your user program. See <a href="#">Chapter 6.2.2 "Importing Commands", page 84</a>.</li> </ul>
Scan head control fails	<ul style="list-style-type: none"> <li>Check if the scan head is properly connected to the RTC5 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 58</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>
Laser control fails	<ul style="list-style-type: none"> <li>Check the interface between the RTC5 and the laser.</li> </ul>
Irregular marking results	<ul style="list-style-type: none"> <li>Check the laser and <a href="#">Scanner Delays</a>. See <a href="#">Chapter 7.2.3 "Notes on Optimizing the Delays", page 143</a>.</li> </ul>
Laser does not switch off during <a href="#">Jump commands</a>	<ul style="list-style-type: none"> <li>Check the laser and <a href="#">Scanner Delays</a>. See <a href="#">Chapter 7.2 "Delay Settings – Coordinating Scan Head Control and Laser Control", page 133</a>.</li> </ul>

Additionally, the following commands can be 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 119](#)).
- With `set_verify`, you can verify that all downloads (commands, tables) were performed error-free (see [Chapter 6.8.1 "Download Verification", page 120](#)).
- 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 [Chapter 7.3.7 "Status Monitoring and Diagnostics", page 172](#); for iDRIVE scan systems see also [Chapter 8.1 "iDRIVE Functions", page 198](#)).
- 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 cycle 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 [page 278](#)) are typically used for this, one command might overwrite the other's output value within the same 10 µs clock cycle. 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 cycle 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 143](#)).

If the problems persist, contact SCANLAB.



## 13 Customer Service

### 13.1 Servicing and Repairs

All servicing and repairs should be performed only at SCANLAB. The warranty expires if the 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 repairs not performed by SCANLAB
- if the RTC5 Board has been altered
- for damage resulting from improper packaging of a product returned to SCANLAB
- 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 RTC5 can be returned to SCANLAB for a fee to be properly disposed of.



## 14 Legal

### 14.1 EU Declaration of Conformity – RTC5 PCI Board


<p><b>EU-Konformitätserklärung</b></p> <p>für das Produkt:</p> <p><b>RTC4 (PCI); RTC4 PCI-Express; RTC5 (PCI); RTC5 PCI-Express; RTC6 PCI-Express</b></p> <p>Der Hersteller</p> <p><b>SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Deutschland</b></p> <p>erklärt, dass das genannte Produkt die einschlägigen Harmonisierungsrechtsvorschriften der Union erfüllt:</p> <ul style="list-style-type: none"><li>• <b>2014/30/EU</b> - Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Harmonisierung der Rechtsvorschriften der Mitgliedstaaten über die elektromagnetische Verträglichkeit (EMV-Richtlinie).</li><li>• <b>2011/65/EU</b> - Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Beschränkung der Verwendung bestimmter gefährlicher Stoffe in Elektro- und Elektronikgeräten (RoHS-Richtlinie).</li></ul> <p>Folgende harmonisierte Normen wurden angewandt:</p> <ul style="list-style-type: none"><li>• <b>EN IEC 61000-6-2:2005</b> – Elektromagnetische Verträglichkeit (EMV) - Teil 6-2: Fachgrundnormen - Störfestigkeit für Industriebereiche</li><li>• <b>DIN EN 55011:2018-05</b> - Industrielle, wissenschaftliche und medizinische Geräte - Funkstörungen - Grenzwerte und Messverfahren</li><li>• <b>DIN EN IEC 63000:2019-05</b> - Technische Dokumentation zur Beurteilung von Elektro- und Elektronikgeräten hinsichtlich der Beschränkung gefährlicher Stoffe</li></ul> <p>Die alleinige Verantwortung für die Ausstellung dieser Konformitätserklärung trägt der Hersteller.</p> <p>Dieses Dokument zur Kundeninformation wurde elektronisch ausgestellt und ist daher ohne Unterschrift gültig. Eine unterschriebene Ausführung ist Bestandteil der SCANLAB-internen technischen Dokumentation.</p> <p><b><i>EU Declaration of Conformity</i></b></p> <p>for the product:</p> <p><b>RTC4 (PCI); RTC4 PCI-Express; RTC5 (PCI); RTC5 PCI-Express; RTC6 PCI-Express</b></p> <p>The manufacturer</p> <p><b>SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Germany</b></p> <p>declares that the product mentioned above complies with the relevant Union harmonization legislation:</p> <ul style="list-style-type: none"><li>• <b>2014/30/EU</b> - Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the harmonisation of the laws of the Member States relating to electromagnetic compatibility (EMC Directive)</li><li>• <b>2011/65/EU</b> - Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the restriction of the use of certain hazardous substances in electrical and electronic equipment (RoHS Directive).</li></ul> <p>The following harmonized standards have been applied:</p> <ul style="list-style-type: none"><li>• <b>EN IEC 61000-6-2:2005</b> - Electromagnetic compatibility (EMC) - Part 6-2: Generic standards - Immunity for industrial environments</li><li>• <b>DIN EN 55011:2018-05</b> - Industrial, scientific and medical equipment – Radio-frequency disturbance characteristics - Limits and methods of measurement</li><li>• <b>DIN EN IEC 63000:2019-05</b> - Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances</li></ul> <p>This declaration of conformity is issued under the sole responsibility of the manufacturer.</p> <p>This document, used for customer information, was issued electronically and is therefore valid without a signature. A signed version is part of the SCANLAB internal technical documentation.</p> <p>SCANLAB GmbH Puchheim 2021-07-19</p> <p>© SCANLAB GmbH – 2021/07</p>



## 14.2 Compliance with FCC Rules

The RTC5 PCI 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 RTC5 Board is operated in a commercial environment. The RTC5 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 RTC5 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 Technical Specifications – RTC5 PCI Board

System Requirements		Interfaces	
Windows PC with PCI bus interface			
Operating system	Microsoft Windows 10, 8, 7 as 32-bit or 64-bit version	<b>SCANHEAD Connector</b> Connector	9-pin D-SUB connector (female)
Dimensions		<b>2. SCANHEAD Socket Connector</b>	
Length	160 mm	Connector	10-pin socket connector.
Height	100 mm		<ul style="list-style-type: none"> <li>• xy output values only with enabled <b>Option "Second Scan Head Control"</b></li> <li>• z output values only with enabled <b>Option "3D"</b></li> </ul>
Scan System Control			
Number of list memory areas	Up to 3 (configurable)		
Capacity of list memory area	1000000 commands	Signal transmission	SL2-100 protocol. Via <b>XY2-100 Converter (Accessory)</b> : XY2-100 protocol.
Output interval of Microsteps	10 $\mu$ s		
Maximum range for the <b>Image field</b> coordinates	-524,288... +524,287 (20-bit signed = 21 bit)		
Virtual <b>Image field</b> (for example, for Processing-on-the-fly)	-8,388,608 to +8,388,607 (24-bit signed)		

<b>LASER Connector</b>		Inputs for external start and stop signals	TTL active-LOW, internally connected to +3.3 V by pull-up resistors (4.7 kΩ)
Connector	15-pin D-SUB connector, female	• /START	edge sensitive HIGH level > 2.3 V LOW level < 0.6 V
Laser control signals	LASER1, LASER2, LASERON	• /STOP	level sensitive HIGH level > 2.3 V LOW level < 0.6 V
• TTL level	5 V, active-HIGH or active-LOW (programmable)	• Reference	GND <sup>(a)</sup>
• Max. current load	20 mA	Other signals	
• Reference	GND <sup>(a)</sup> (GND2 with Option "DC/DC Converter")	• BUSY OUT	5 V, TTL active-HIGH, max. 10 mA
Analog outputs	ANALOG OUT1, ANALOG OUT2	• +5 V	max. 100 mA
• Output voltage range	0 V...10 V	• Reference	GND <sup>(a)</sup>
• 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 60](#).

### EXTENSION 1 Socket Connector

Connector	40-pin socket connector, output signal level configurable (by JP1)
Digital output	16 bits, buffered <ul style="list-style-type: none"> <li>• LOW level &lt; 0.4 V</li> <li>• HIGH level &gt; 2.0 V (3.3 V or 5 V)</li> <li>• Max. current load <math>\pm 8</math> mA</li> <li>• Reference GND<sup>(a)</sup></li> <li>• LATCH signal 3.3 V or 5 V, TTL active-HIGH, 5 <math>\mu</math>s pulse, max. 10 mA</li> </ul>
Digital input	16 bits <ul style="list-style-type: none"> <li>• LOW level &lt; 0.5 V</li> <li>• HIGH level &gt; 2.6 V...24 V</li> <li>• Max. input voltage range <math>-0.5</math> V...+26 V</li> <li>• Input resistance <math>&gt; 10</math> k<math>\Omega</math></li> <li>• Reference GND<sup>(a)</sup></li> <li>• SYNC signal 3.3 V or 5 V, TTL active-HIGH, square wave signal (5 <math>\mu</math>s pulse, 10 <math>\mu</math>s period) max. 10 mA</li> </ul>
Other signals	<ul style="list-style-type: none"> <li>• BUSY OUT 3.3 V or 5 V, TTL active-HIGH, max. 10 mA</li> <li>• VCC 3.3 V or 5 V, max. 100 mA</li> <li>• +5 V max. 100 mA</li> <li>• Reference GND<sup>(a)</sup></li> </ul>

### EXTENSION 2 Socket Connector

Connector	26-pin socket connector, configurable (by JP2...JP8)
Digital output	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<sup>(a)</sup></li> <li>• LATCH signal 5 V, TTL active-HIGH, 5 <math>\mu</math>s pulse, max. 10 mA</li> </ul>
Laser signals	LASERON, LASER1, LASER2 (see LASER connector)
Other signals	<ul style="list-style-type: none"> <li>• +5 V max. 100 mA</li> <li>• Reference GND<sup>(a)</sup></li> </ul>

### MARKING ON THE FLY Socket Connector

Connector	16-pin socket connector
2 encoder input ports for incremental encoders	<b>ENCODER X</b> (1 $\pm$ ,2 $\pm$ ) and <b>ENCODER Y</b> (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 outputs	<b>ANALOG OUT2</b> (see LASER connector)
Inputs for external start and stop signals	/START2, /STOP2 (see /START, /STOP of the LASER connector)
Other signals	<ul style="list-style-type: none"> <li>• BUSY OUT 5 V, TTL active-HIGH, max. 10 mA</li> <li>• +5 V max. 100 mA</li> <li>• Reference GND<sup>(a)</sup></li> </ul>



#### 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

#### STEPPER MOTOR Socket Connector

Connector	10-pin socket connector
Signals for controlling two stepper motors:	
	• ENABLE and DIRECTION outputs
	• CLOCK output
	5 V, TTL
	5 V, TTL active-HIGH, 5 $\mu$ s pulse
	• SWITCH input
	TTL active-LOW, internally connected to +3.3 V by pull-up resistors (10 k $\Omega$ )
	• Max. current load
	25 mA
	• Reference
	GND <sup>(a)</sup>

#### SPI / I2C Socket Connector

Connector	10-pin socket connector
-----------	-------------------------

In McBSP configuration, see [Section "McBSP Interface", page 71](#):

- Transmitter signal 3.3 V TTL level
- Receiver signal 3.3 V or 5 V TTL level
- McBSP mode Single Phase Frame  
Single Element per Frame  
32 bits per Element  
Data delay  $X_{Delay}$  bits  
Data delay  $R_{Delay}$  bits
- Reference GND<sup>(a)</sup>

In SPI configuration, see [Section "SPI Interface", page 73](#):

- Transmitter signal 3.3 V TTL level
- Receiver signal 3.3 V or 5 V TTL level
- SPI mode Single Phase Frame  
Single Element per Frame  
32 bits per Element  
Data delay 1 bit
- Reference GND<sup>(a)</sup>

## 16 Appendix A: RTC5 PCIe Board

### 16.1 Product Overview

#### 16.1.1 Intended Use – Comparison to the RTC5 PCI Board

Just like the RTC5 PCI Board, the SCANLAB RTC5 PCIe Board is intended for synchronous real-time control of scan systems, lasers and peripheral equipment. It differs from the RTC5 PCI Board in the following hardware-related details:

- The RTC5 PCIe Board provides a PCI-Express interface and must be connected to a PCI-Express slot of the PC.
- The SPI/I<sup>2</sup>C socket connector provides two 10 V analog inputs instead of an I<sup>2</sup>C interface (see [Chapter 16.2.4 "SPI/I<sup>2</sup>C Socket Connector", page 742](#)).

Otherwise, the RTC5 PCIe Board provides the same functionality for controlling scan systems, lasers and peripheral equipment as the RTC5 PCI Board. Neither the number, type and positioning of connectors for controlling scan systems, lasers and peripheral equipment nor the jumpers for customized configuration differ from that of the RTC5 PCI Board. Specifications for signal inputs and outputs are the same as with the RTC5 PCI Board (with above-mentioned exception), as are software installation and developing user programs. Both the RTC5 PCIe Board and the RTC5 PCI Board use the same RTC5 software package (driver, DLL, import declarations, etc.) and same command set.

#### 16.1.2 System Requirements

##### Hardware

The RTC5 PCIe Board requires a Windows PC with a PCI-Express bus interface and at least one free PCI-Express slot. The dimensions of the RTC5 Board are shown in [Figure 75](#).

RTC5 PCIe Boards intended for synchronized master/slave operation must be installed in adjacent PCI-Express slots.

##### Software

The RTC5 PCIe Board uses the same driver and [RTC5 DLL](#) file as the RTC5 PCI Board (about the supported operating systems see [Chapter 2.5.2 "Software", page 32](#)).

#### Notice!

- The RTC5 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 33](#).

#### 16.1.3 Optional Functionality

The RTC5 PCIe Board can be configured for the same functionality as the RTC5 PCI Board (see [Chapter 2.6 "Options", page 34](#)). Optional functionality must be enabled by SCANLAB or installed.

#### 16.1.4 Labeling

The serial number of the RTC5 PCIe Board is printed on a label attached to the board (format: "RTC SN...").

The article number and configuration are described in the packaging list (see also [Chapter 2.8 "Accessories for the RTC5 PCI Board", page 37](#) and [Chapter 16.1.5 "Type Identification", page 739](#)).

#### 16.1.5 Type Identification

The ID number of the RTC5 PCIe Board reveals the board's factory-equipped jumper configuration (JP1...JP8). The jumper configuration is additionally encoded in a three-digit type code scheme. The type code scheme is identical to that of the RTC5 PCI Board (see [Chapter 2.7 "Jumper Settings and Type Identification", page 35](#)).

#### 16.1.6 Unpacking Instructions and Typical Scope of Delivery

- (1) Carefully remove the RTC5 PCIe Board from the package.
- (2) Keep the packaging, including the antistatic bag the RTC5 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 package typically includes an RTC5 PCIe Board and a data CD (containing the corresponding software (see below) and this manual). Some product versions may also supply additional components, see [Chapter 16.1.7 "Accessories", page 739](#).

#### Delivered Software

The delivered software package is identical to the RTC5 software package (see [Chapter 2.3 "Delivered RTC5 Software Package", page 27](#)). It contains drivers for Microsoft's Windows 10, 8, 7 (32-bit or 64-bit) operating systems, correction, **RTC5 DLL** and program files, as well as utility files for C, C++ and C#.

#### 16.1.7 Accessories

In addition to the RTC5 PCIe Board and its RTC5 software package, the following accessories – as with the RTC5 PCI Board – can be obtained from SCANLAB (only hardware extensions from SCANLAB should be used in combination with the RTC5 PCIe Board):

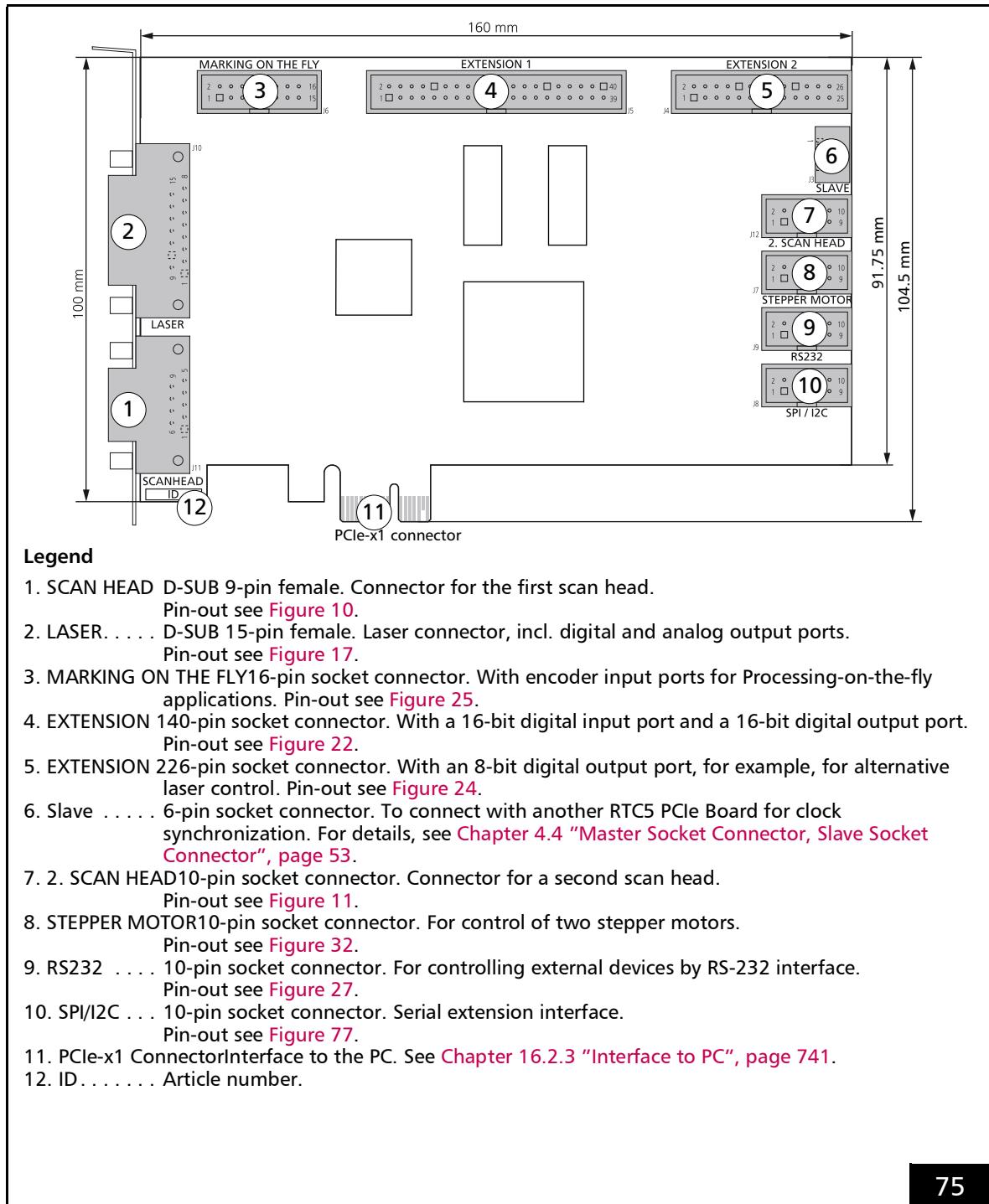
- XY2-100 Converter (Accessory)
- Data cables, see [Chapter 2.8.3 "Data Cables", page 37](#)
- Laser adapter, see [Chapter 2.8.2 "Laser Adapter", page 37](#)
- Slot cover with a 9-pin D-SUB connector for using the 2. SCAN HEAD socket connector, see [Section "Second Scan Head Slot Cover \(Accessory\)", page 55](#)
- Slot cover with a 15-pin D-SUB connector for using the MARKING ON THE FLY socket connector, see [Section "MARKING ON THE FLY Slot Cover \(Accessory\)", page 70](#)

#### 16.1.8 Supplementary Software

To facilitate customizing RTC correction files basing on your own test measurements, SCANLAB offers its correXion line of software and related documentation, see also [Section "Image Field Correction Algorithm", page 163](#).

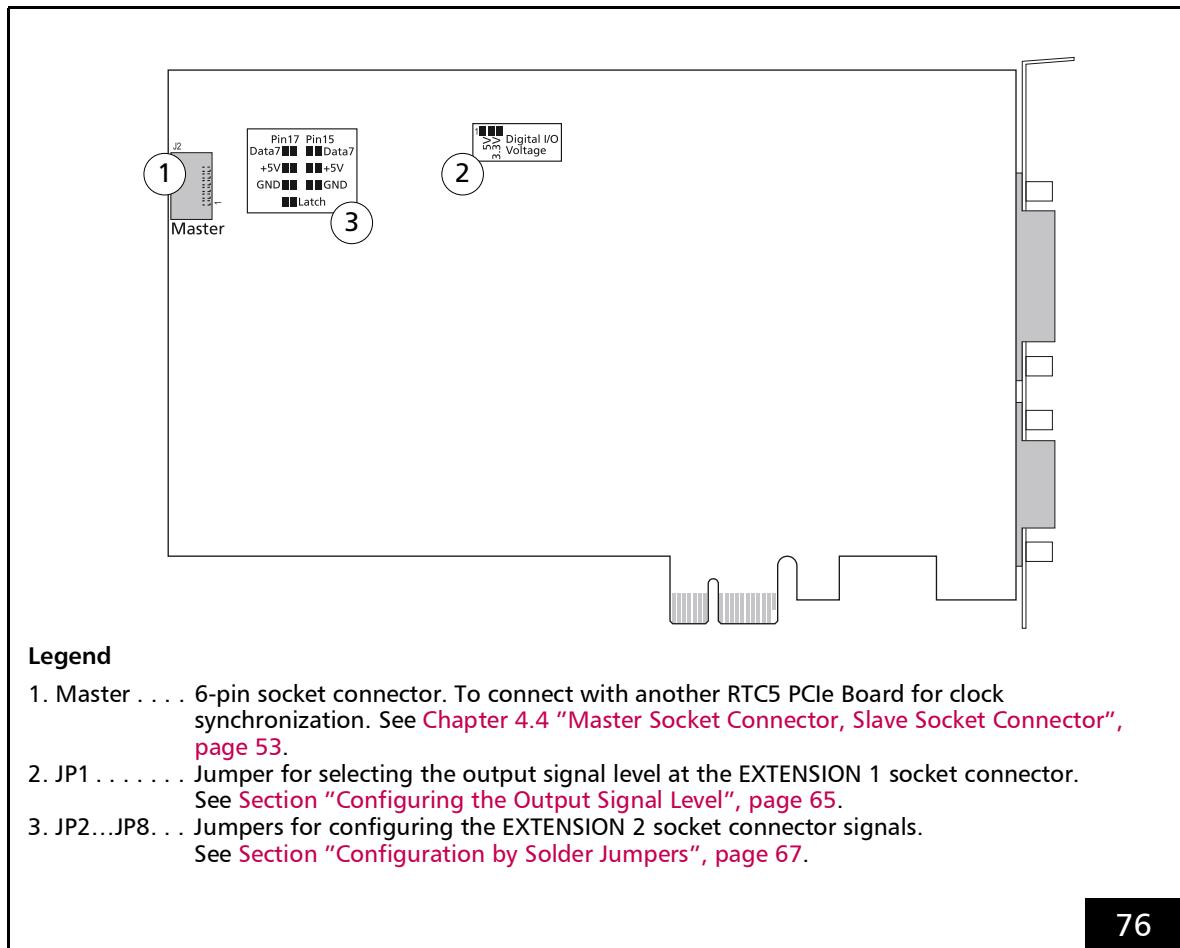
## 16.2 RTC5 PCIe Board – Layout and Interfaces

### 16.2.1 Layout – Upper Side



RTC5 PCIe Board: upper side.

### 16.2.2 Layout – Lower Side



76

RTC5 PCIe Board: lower side.

### 16.2.3 Interface to PC

Interface to the PC is the PCIe-x1 connector of the RTC5 PCIe Board.

The RTC5 PCIe Board can be installed into any Windows PC with PCI-Express bus and at least one free PCI-Express slot.

#### Notice!

- The RTC5 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 33](#).

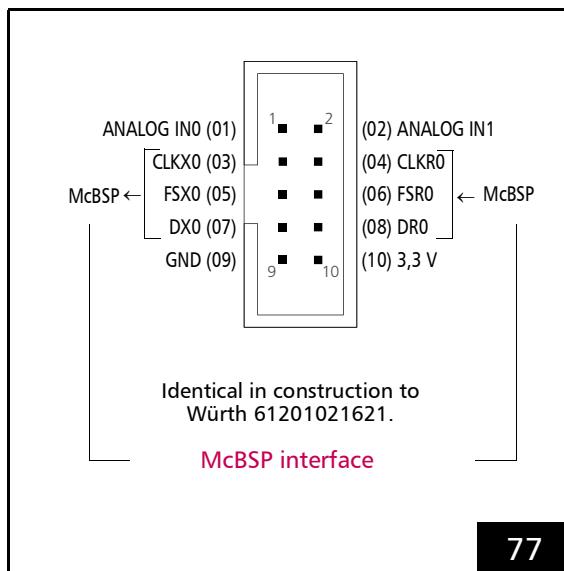
### 16.2.4 SPI/I2C Socket Connector

The SPI/I2C socket connector has 10 pins. It is located on the upper side of the RTC5 PCIe Board, see [Figure 75](#).

The SPI/I2C socket connector is a hardware interface that

- is designed for bidirectional data exchange ([McBSP](#)), see [Section "McBSP Interface", page 742](#) 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 742](#).

The pin-out is shown in [Figure 77](#).



### McBSP Interface

The [McBSP interface](#) and the corresponding commands are identical to that of the RTC5 PCI Board (see [Section "McBSP Interface", page 71](#)).

### Analog Input Ports

The RTC5 PCIe Board provides two 10 V analog inputs at the SPI/I2C socket connector (instead of the I<sup>2</sup>C interface which is provided by the RTC5 PCI Board; similar to the RTC5 PCI Board's optional [ADC Add-On Board](#)):

- ANALOG IN0
- ANALOG IN1

For using these analog input ports, see [Chapter 4.6.8 "Analog Inputs \(Accessory\)", page 75](#).

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

For technical specifications, see [Analog input ports, page 746](#).



## 16.3 Installation and Start-Up

For installing and starting-up an RTC5 PCIe Board and its software proceed as with an RTC5 PCI Board, see [Chapter 5 "Installation and Start-Up", page 77](#).

### Notice!

- The RTC5 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). See also [Section "Notes", page 33](#).



## 16.4 Legal

### 16.4.1 EU Declaration of Conformity – RTC5 PCIe Board


<p><b>EU-Konformitätserklärung</b></p> <p>für das Produkt:</p> <p><b>RTC4 (PCI); RTC4 PCI-Express; RTC5 (PCI); RTC5 PCI-Express; RTC6 PCI-Express</b></p> <p>Der Hersteller</p> <p><b>SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Deutschland</b></p> <p>erklärt, dass das genannte Produkt die einschlägigen Harmonisierungsrechtsvorschriften der Union erfüllt:</p> <ul style="list-style-type: none"><li>• <b>2014/30/EU</b> - Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Harmonisierung der Rechtsvorschriften der Mitgliedstaaten über die elektromagnetische Verträglichkeit (EMV-Richtlinie).</li><li>• <b>2011/65/EU</b> - Richtlinie des EUROPÄISCHEN PARLAMENTS UND DES RATES zur Beschränkung der Verwendung bestimmter gefährlicher Stoffe in Elektro- und Elektronikgeräten (RoHS-Richtlinie).</li></ul> <p>Folgende harmonisierte Normen wurden angewandt:</p> <ul style="list-style-type: none"><li>• <b>EN IEC 61000-6-2:2005</b> – Elektromagnetische Verträglichkeit (EMC) - Teil 6-2: Fachgrundnormen - Störfestigkeit für Industriebereiche</li><li>• <b>DIN EN 55011:2018-05</b> - Industrielle, wissenschaftliche und medizinische Geräte - Funkstörungen - Grenzwerte und Messverfahren</li><li>• <b>DIN EN IEC 63000:2019-05</b> - Technische Dokumentation zur Beurteilung von Elektro- und Elektronikgeräten hinsichtlich der Beschränkung gefährlicher Stoffe</li></ul> <p>Die alleinige Verantwortung für die Ausstellung dieser Konformitätserklärung trägt der Hersteller.</p> <p>Dieses Dokument zur Kundeninformation wurde elektronisch ausgestellt und ist daher ohne Unterschrift gültig. Eine unterschriebene Ausführung ist Bestandteil der SCANLAB-internen technischen Dokumentation.</p> <p><b><i>EU Declaration of Conformity</i></b></p> <p>for the product:</p> <p><b>RTC4 (PCI); RTC4 PCI-Express; RTC5 (PCI); RTC5 PCI-Express; RTC6 PCI-Express</b></p> <p>The manufacturer</p> <p><b>SCANLAB GmbH, Siemensstr. 2a, 82178 Puchheim, Germany</b></p> <p>declares that the product mentioned above complies with the relevant Union harmonization legislation:</p> <ul style="list-style-type: none"><li>• <b>2014/30/EU</b> - Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the harmonisation of the laws of the Member States relating to electromagnetic compatibility (EMC Directive)</li><li>• <b>2011/65/EU</b> - Directive of the EUROPEAN PARLIAMENT AND OF THE COUNCIL on the restriction of the use of certain hazardous substances in electrical and electronic equipment (RoHS Directive).</li></ul> <p>The following harmonized standards have been applied:</p> <ul style="list-style-type: none"><li>• <b>EN IEC 61000-6-2:2005</b> - Electromagnetic compatibility (EMC) - Part 6-2: Generic standards - Immunity for industrial environments</li><li>• <b>DIN EN 55011:2018-05</b> - Industrial, scientific and medical equipment – Radio-frequency disturbance characteristics - Limits and methods of measurement</li><li>• <b>DIN EN IEC 63000:2019-05</b> - Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances</li></ul> <p>This declaration of conformity is issued under the sole responsibility of the manufacturer.</p> <p>This document, used for customer information, was issued electronically and is therefore valid without a signature. A signed version is part of the SCANLAB internal technical documentation.</p> <p>SCANLAB GmbH Puchheim 2021-07-19</p> <p>© SCANLAB GmbH – 2021/07</p>



### **16.4.2 Compliance with FCC Rules**

The RTC5 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 RTC5 PCIe Board is operated in a commercial environment. The RTC5 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 RTC5 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.



## 16.5 Technical Specifications – RTC5 PCIe Board

### System Requirements

Windows PC with PCI-Express bus interface

Operating system Microsoft Windows 10, 8,  
7 as 32-bit or 64-bit  
version

### Dimensions

Length 160 mm

Height 104.5 mm

### Interface to PC

PCI-Express x1, version 1.0

### SPI/I2C Socket Connector

Connector 10-pin socket connector

McBSP interface Like RTC5 PCI Board (see  
page 71)

Analog input ports ANALOG IN0,  
ANALOG IN1

- Input voltage 0 V...10 V
- range 4 kΩ
- Input resistance 12 bit
- ADC resolution GND
- Reference

### Other Connectors and Specifications

Identical to RTC5 PCI Board

## 17 Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards

- For usage, see [control\\_command](#).

- Example:

```
control_command( Data = 0501H )
that is, CodeHIGH = 05 (SetMode) and
CodeLOW = 01 requests that the
iDRIVE scan system should change the data type
to be transmitted to "actual position". The
iDRIVE scan system evaluates the request and – if
successful – subsequently transmits the
"actual position" to the RTC-control board
(notation in this manual is "Signal 0501H" in this
case).
```

- The following reference is used in several
RTC manuals and is thus generic.

Code <sub>HIGH</sub> (hex)	
05	<p><b>SetMode</b></p> <p><b>SetMode</b> selects the data signal to be returned from the scan system for the specified axis. See also <a href="#">Chapter 8.1.2 "Configuring the Data Signal Transmission Behavior of the Scan System", page 225</a>.</p> <p>Each <b>Code<sub>LOW</sub></b> parameter value corresponds to a particular data type.</p> <p>Default setting: Code<sub>LOW</sub> = 00<sub>H</sub> (XY2-100 status word).</p> <p>See also "On SetMode", page 1122.</p> <p>See also "On SetMode, SetControlDefinitionMode and SetEchoMode", page 1122.</p> <p>iDRIVE scan systems with XY2-100 interface transmit a signed 16-bit-value.</p> <p>iDRIVE scan systems with SL2-100 interface transmit an signed 20-bit value. For example, excelliSCAN-scan heads.</p> <p>iDRIVE scan systems with XY2-100 Enhanced interface transmit a signed 16-bit-value. With these, the <b>XY2-100 Converter (Accessory)</b> converts (by multiplying by 16) the value to a signed 20-bit value.</p> <p>In this document:</p> <p>0500<sub>H</sub>; 0501<sub>H</sub>; 0502<sub>H</sub>; 0503<sub>H</sub>; 0504<sub>H</sub>; 0505<sub>H</sub>; 0506<sub>H</sub>; 0507<sub>H</sub>; 0508<sub>H</sub>; 0509<sub>H</sub>; 0510<sub>H</sub>; 0511<sub>H</sub>; 0512<sub>H</sub>; 0513<sub>H</sub>; 0514<sub>H</sub>; 0515<sub>H</sub>; 0516<sub>H</sub>; 0517<sub>H</sub>; 0518<sub>H</sub>; 0519<sub>H</sub>; 051A<sub>H</sub>; 051B<sub>H</sub>; 051C<sub>H</sub>; 051D<sub>H</sub>; 051E<sub>H</sub>; 051F<sub>H</sub>; 0520<sub>H</sub>; 0521<sub>H</sub>; 0522<sub>H</sub>; 0523<sub>H</sub>; 0524<sub>H</sub>; 0525<sub>H</sub>; 0526<sub>H</sub>; 0527<sub>H</sub>; 0528<sub>H</sub>; 0529<sub>H</sub>; 052A<sub>H</sub>; 052B<sub>H</sub>; 052C<sub>H</sub>; 052D<sub>H</sub>; 052E<sub>H</sub>; 052F<sub>H</sub>; 0530<sub>H</sub>; 0531<sub>H</sub>; 0532<sub>H</sub>; 0533<sub>H</sub>; 0534<sub>H</sub>; 0535<sub>H</sub>; 0536<sub>H</sub>; 0537<sub>H</sub>; 0538<sub>H</sub>; 0539<sub>H</sub>; 053A<sub>H</sub>; 053B<sub>H</sub>; 053C<sub>H</sub>; 053D<sub>H</sub>; 053E<sub>H</sub>; 053F<sub>H</sub>; 0549<sub>H</sub>; 055E<sub>H</sub>; 0564<sub>H</sub>; 0565<sub>H</sub>; 0574<sub>H</sub>;</p>

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

<b>Code<sub>HIGH</sub> (hex)</b>	<b>Code<sub>LOW</sub> (hex)</b>	Data type transmitted by the scan system	
<b>05</b>	<b>00</b>	As of scan system firmware $\geq$ 2001. XY2-100 status word	
	16-bit protocol	20-bit protocol	
	Bit #15 (MSB)	Bit #19 (MSB)	Like [Bit #7   Bit #11]. = 1: The scan system servo control is ready for input data ("PowerOK", actually corresponds to a "ServoOK"). For scan systems with sensors as well as scan systems with interlock: there is also no interlock error.
	Bit #14	Bit #18	Like [Bit #6   Bit #10]. = 1: The system temperature within optimal range ("TempOK"). For scan systems with sensors: there is also no warning.
	Bit #13	Bit #17	= 1.
	Bit #12	Bit #16	Like [Bit #4   Bit #8]. = 1: The y axis position errors are within allowed range (PosAck Signal).
	Bit #11	Bit #15	Like [Bit #3   Bit #7]. = 1: The x axis position errors are within allowed range (PosAck Signal).
	Bit #10	Bit #14	Like [Bit #2   Bit #6]. = 1. Only intelliWELD: Reserved. For scan systems with sensors for automatic self calibration: Reserved.
	Bit #9	Bit #13	= 0.
	Bit #8	Bit #12	= 1.
	Bit #7	Bit #11	Like [Bit #15   Bit #19].
	Bit #6	Bit #10	Like [Bit #14   Bit #18].
	Bit #5	Bit #9	= 1.
	Bit #4	Bit #8	Like [Bit #12   Bit #16].
	Bit #3	Bit #7	Like [Bit #11   Bit #15].
	Bit #2	Bit #6	Like [Bit #10   Bit #14].
	Bit #1	Bit #5	= 0.
	Bit #0	Bit #4	= 1.
–	Bit #3		= 0.
–	Bit #2		= 0.
–	Bit #1		= 0.
–	Bit #0		= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	01	As of scan system firmware $\geq$ 2001. Actual position (angular position of galvanometer scanners)		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In bits. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In bits. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	02	As of scan system firmware $\geq$ 2001. Set position (angular position of galvanometer scanners) As of scan system firmware $\geq$ 5050. With excelliSCAN scan heads: Precalculated set position (angular position of galvanometer scanners). Refer to "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual, Chapter 2.1.6 "SCANAhead System Timing", page 20 and Figure 7.		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In bits. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In bits. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system						
05	03	<p>As of scan system firmware <math>\geq</math> 2001.</p> <p>Position error</p> <p>(= set position <math>0502_H</math> – actual position <math>0501_H</math>)</p> <p>As of scan system firmware <math>\geq</math> 5050.</p> <p>With excelliSCAN scan heads:</p> <p><b>Trajectory Error.</b> See Glossary entry.</p>						
		<table border="1"> <thead> <tr> <th>16-bit protocol</th> <th>20-bit protocol</th> <th></th> </tr> </thead> <tbody> <tr> <td>Bit #0 ... Bit #15</td> <td>Bit #0 ... Bit #19</td> <td>           Unit with 16-bit protocol: In bits.            Value range with 16-bit protocol: [-32.768...+32.767].            Unit with 20-bit protocol: In bits.            Value range with 20-bit protocol: -524.288...+524.287.         </td> </tr> </tbody> </table>	16-bit protocol	20-bit protocol		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In bits. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In bits. Value range with 20-bit protocol: -524.288...+524.287.
16-bit protocol	20-bit protocol							
Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In bits. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In bits. Value range with 20-bit protocol: -524.288...+524.287.						
		<p>Notice! Generic information. May be incomplete or even incorrect for your scan system.</p> <p>Always consult the dedicated scan system manual!</p> <p>Module Rev.1.0.6 en-US</p>						

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system						
05	04	<p>As of scan system firmware <math>\geq</math> 2001.</p> <p>Actual current (output stage current)</p>						
		<table border="1"> <thead> <tr> <th>16-bit protocol</th> <th>20-bit protocol</th> <th></th> </tr> </thead> <tbody> <tr> <td>Bit #0 ... Bit #15</td> <td>Bit #0 ... Bit #19</td> <td>           Unit with 16-bit protocol: In mA.            Value range with 16-bit protocol: [-32.768...+32.767].            Unit with 20-bit protocol: In <math>10^{-1}</math> mA.            Value range with 20-bit protocol: -524.288...+524.287.         </td> </tr> </tbody> </table>	16-bit protocol	20-bit protocol		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In mA. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $10^{-1}$ mA. Value range with 20-bit protocol: -524.288...+524.287.
16-bit protocol	20-bit protocol							
Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In mA. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $10^{-1}$ mA. Value range with 20-bit protocol: -524.288...+524.287.						

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
 Always consult the dedicated scan system manual!  
 Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	05	As of scan system firmware $\geq$ 2001. Relative galvanometer scanner control Not intellicube.		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In per mille. Value range with 16-bit protocol: [-1,000...+1,000]. Unit with 20-bit protocol: In $16^{-1}$ per mille. Value range with 20-bit protocol: [-16,000...+16,000]. The return value 16 entspricht 1%.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	06	As of scan system firmware $\geq$ 2001. Actual velocity (angular velocity)		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In Bit/ms. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In Bit/ms. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	07	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	08	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	09	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	10	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	11	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	12	As of scan system firmware $\geq$ 2001. Only intelliWELD with varioSCAN de FCi. Not: intelliWELD with varioSCAN FC: Actual z axis position Can only be read out from the channel of the scan head connector to which the z axis is connected.		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In bits. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In bits. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
 Always consult the dedicated scan system manual!  
 Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	13	Reserved.		
		16-bit protocol	20-bit protocol	-
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	-

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
 Always consult the dedicated scan system manual!  
 Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	14	As of scan system firmware $\geq$ 2001. Temperature of galvanometer scanner		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In $10^{-1}$ °C. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $160^{-1}$ °C. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
 Always consult the dedicated scan system manual!  
 Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system						
05	15	<p>As of scan system firmware <math>\geq</math> 2001.</p> <p>Servo board temperature</p> <p>As of scan system firmware <math>\geq</math> 5050.</p> <p>Axis 1 (X-axis): Temperature of the servo board</p> <p>Axis 2 (Y-axis): Temperature of the servo add-on board for evaluating the encoder signals</p>						
		<table border="1"> <tr> <td>16-bit protocol</td> <td>20-bit protocol</td> <td></td> </tr> <tr> <td>Bit #0 ... Bit #15</td> <td>Bit #0 ... Bit #19</td> <td> <p>Unit with 16-bit protocol: In <math>10^{-1}</math> °C.</p> <p>Value range with 16-bit protocol: [-32.768...+32.767].</p> <p>Unit with 20-bit protocol: In <math>160^{-1}</math> °C.</p> <p>Value range with 20-bit protocol: -524.288...+524.287.</p> </td></tr> </table>	16-bit protocol	20-bit protocol		Bit #0 ... Bit #15	Bit #0 ... Bit #19	<p>Unit with 16-bit protocol: In <math>10^{-1}</math> °C.</p> <p>Value range with 16-bit protocol: [-32.768...+32.767].</p> <p>Unit with 20-bit protocol: In <math>160^{-1}</math> °C.</p> <p>Value range with 20-bit protocol: -524.288...+524.287.</p>
16-bit protocol	20-bit protocol							
Bit #0 ... Bit #15	Bit #0 ... Bit #19	<p>Unit with 16-bit protocol: In <math>10^{-1}</math> °C.</p> <p>Value range with 16-bit protocol: [-32.768...+32.767].</p> <p>Unit with 20-bit protocol: In <math>160^{-1}</math> °C.</p> <p>Value range with 20-bit protocol: -524.288...+524.287.</p>						
		<p>Notice! Generic information. May be incomplete or even incorrect for your scan system.</p> <p>Always consult the dedicated scan system manual!</p> <p>Module Rev.1.0.6 en-US</p>						

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system						
05	16	<p>As of scan system firmware <math>\geq</math> 2001.</p> <p>AGC voltage (position detector supply voltage)</p>						
		<table border="1"> <tr> <td>16-bit protocol</td> <td>20-bit protocol</td> <td></td> </tr> <tr> <td>Bit #0 ... Bit #15</td> <td>Bit #0 ... Bit #19</td> <td> <p>Unit with 16-bit protocol: In <math>10^{-2}</math> V.</p> <p>Value range with 16-bit protocol: [-32.768...+32.767].</p> <p>Unit with 20-bit protocol: In <math>1600^{-1}</math> V.</p> <p>Value range with 20-bit protocol: -524.288...+524.287.</p> </td></tr> </table>	16-bit protocol	20-bit protocol		Bit #0 ... Bit #15	Bit #0 ... Bit #19	<p>Unit with 16-bit protocol: In <math>10^{-2}</math> V.</p> <p>Value range with 16-bit protocol: [-32.768...+32.767].</p> <p>Unit with 20-bit protocol: In <math>1600^{-1}</math> V.</p> <p>Value range with 20-bit protocol: -524.288...+524.287.</p>
16-bit protocol	20-bit protocol							
Bit #0 ... Bit #15	Bit #0 ... Bit #19	<p>Unit with 16-bit protocol: In <math>10^{-2}</math> V.</p> <p>Value range with 16-bit protocol: [-32.768...+32.767].</p> <p>Unit with 20-bit protocol: In <math>1600^{-1}</math> V.</p> <p>Value range with 20-bit protocol: -524.288...+524.287.</p>						

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	17	As of scan system firmware $\geq$ 2001. <b>DSP core supply voltage</b>		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In $10^{-2}$ V. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $1600^{-1}$ V. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	18	As of scan system firmware $\geq$ 2001. <b>DSP IO voltage</b>		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In $10^{-2}$ V. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $1600^{-1}$ V. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	19	As of scan system firmware $\geq$ 2001. Analog section voltage		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In $10^{-2}$ V. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $1600^{-1}$ V. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	1A	As of scan system firmware $\geq$ 2001. Analog-digital converter supply voltage		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In $10^{-2}$ V. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $1600^{-1}$ V. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	1B	As of scan system firmware $\geq$ 2001. AGC current (PD supply current)		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	Unit with 16-bit protocol: In mA. Value range with 16-bit protocol: [-32.768...+32.767]. Unit with 20-bit protocol: In $16^{-1}$ mA. Value range with 20-bit protocol: -524.288...+524.287.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	1C	As of scan system firmware $\geq$ 2066. Hardware version of servo board		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	16-bit protocol: 2nnn = DSCB 5021 = ISB1 5031 = ISB2 6011 = microISB  20-bit protocol: 2nnn $\times$ 16 = DSCB 5021 $\times$ 16 = ISB1 5031 $\times$ 16 = ISB2 6011 $\times$ 16 = microISB

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system									
05	1D	<p>As of scan system firmware <math>\geq</math> 2001.</p> <p>Relevant for scan systems with galvanometer scanner heating.</p> <p>Relative heating output of the corresponding galvanometer scanner heater</p> <p>Not relevant for intellicube.</p> <p>Not relevant for dynAXIS XS.</p> <p>Not relevant for scan systems with water-cooled galvanometer scanners.</p>									
		<table border="1"> <tr> <td>16-bit protocol</td> <td>20-bit protocol</td> <td></td> </tr> <tr> <td>Bit #0 ... Bit #15</td> <td>Bit #4 ... Bit #19</td> <td>Unit with 16-bit protocol: In per mille. Value range with 16-bit protocol: [0...1,000]. Unit with 20-bit protocol: In per mille. Value range with 20-bit protocol: [0...1,000].</td> </tr> <tr> <td>–</td> <td>Bit #0 ... Bit #3</td> <td>= 0.</td> </tr> </table>	16-bit protocol	20-bit protocol		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In per mille. Value range with 16-bit protocol: [0...1,000]. Unit with 20-bit protocol: In per mille. Value range with 20-bit protocol: [0...1,000].	–	Bit #0 ... Bit #3	= 0.
16-bit protocol	20-bit protocol										
Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In per mille. Value range with 16-bit protocol: [0...1,000]. Unit with 20-bit protocol: In per mille. Value range with 20-bit protocol: [0...1,000].									
–	Bit #0 ... Bit #3	= 0.									

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system									
05	1E	<p>As of scan system firmware <math>\geq</math> 2001.</p> <p>Serial number, lower 16 bits</p>									
		<table border="1"> <tr> <td>16-bit protocol</td> <td>20-bit protocol</td> <td></td> </tr> <tr> <td>Bit #0 ... Bit #15</td> <td>Bit #4 ... Bit #19</td> <td>Unit with 16-bit protocol: None. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: None. Value range with 20-bit protocol: [0...65.535].</td> </tr> <tr> <td>–</td> <td>Bit #0 ... Bit #3</td> <td>= 0.</td> </tr> </table>	16-bit protocol	20-bit protocol		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: None. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: None. Value range with 20-bit protocol: [0...65.535].	–	Bit #0 ... Bit #3	= 0.
16-bit protocol	20-bit protocol										
Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: None. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: None. Value range with 20-bit protocol: [0...65.535].									
–	Bit #0 ... Bit #3	= 0.									

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	1F	As of scan system firmware ≥ 2001. Serial number, upper 16 bits		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: None. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: None. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	20	As of scan system firmware ≥ 2001. SCANLAB article number, lower 16 bits		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: None. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: None. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	21	As of scan system firmware $\geq$ 2001. SCANLAB article number, upper 16 bits		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: None. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: None. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	22	As of scan system firmware $\geq$ 2001. Version number of scan system firmware		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: None. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: None. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	23	As of scan system firmware ≥ 2001. Calibration angle Mechanical scan angle ( $\pm$ ) for 96% of the maximum or minimum control value. With 16-bit protocol $\pm 31457$ bit. With 20-bit protocol $\pm 503316$ bit.		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In $10^{-3}$ degrees. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: In $10^{-3}$ degrees. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	24	As of scan system firmware ≥ 2001. Aperture		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In mm. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: In mm. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	25	As of scan system firmware ≥ 2001.  Wavelength		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In nm. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: In nm. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	26	As of scan system firmware ≥ 2078.  Tuning number		
		16-bit protocol	20-bit protocol	
		Bit #8 ... Bit #15	Bit #12 ... Bit #19	Start setting.
		Bit #0 ... Bit #7	Bit #4 ... Bit #11	Current setting.
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	27	As of scan system firmware $\geq$ 2078. Only after <code>control_command(Data = 17FF<sub>H</sub>)</code> : number of the temporarily stored data type		
		16-bit protocol	20-bit protocol	
		Bit #8	Bit #12	= 0.
		...	...	
		Bit #15	Bit #19	
		Bit #0	Bit #4	Cached setting.
		...	...	
		Bit #7	Bit #11	
		—	Bit #0	= 0.
			...	
			Bit #3	

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	28	As of scan-system firmware ≥ 2060. Diagnostic flags, lower 16 bits		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	29	As of scan-system firmware ≥ 2060. Diagnostic flags, upper 16 bits		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system																																																									
05	2A	<p>As of scan system firmware <math>\geq</math> 2001.</p> <p>Stop event code</p> <p>Value range with 16-bit protocol: [0...65.535].</p> <p>Value range with 20-bit protocol: [0...65.535].</p>																																																									
		<table border="1"> <thead> <tr> <th>16-bit protocol (hex)</th><th>20-bit protocol (hex)</th><th></th></tr> </thead> <tbody> <tr><td>0001</td><td>00010</td><td>The galvanometer scanner has reached a critical edge position.</td></tr> <tr><td>0002</td><td>00020</td><td>AD converter error.</td></tr> <tr><td>0003</td><td>00030</td><td>Temperature in scan system above max. allowed value.</td></tr> <tr><td>0004</td><td>00040</td><td>External power supply voltages have dropped below the allowed value.</td></tr> <tr><td>0005</td><td>00050</td><td>Flags are not valid.</td></tr> <tr><td>0006</td><td>00060</td><td>Reserved.</td></tr> <tr><td>...</td><td>...</td><td>...</td></tr> <tr><td>000C</td><td>000C0</td><td>Reserved.</td></tr> <tr><td>000D</td><td>000D0</td><td>Reserved.</td></tr> <tr><td>000E</td><td>000E0</td><td>Position Acknowledge time out (set position not reached for long time).</td></tr> <tr><td>000F</td><td>000F0</td><td>Reserved.</td></tr> <tr><td>0014</td><td>00140</td><td>Reserved.</td></tr> <tr><td>0015</td><td>00150</td><td>Reserved.</td></tr> <tr><td>0016</td><td>00160</td><td>Reserved.</td></tr> <tr><td>0017</td><td>00170</td><td>Reserved.</td></tr> <tr><td>0018</td><td>00180</td><td>Reserved.</td></tr> <tr><td>0019</td><td>00190</td><td>Reserved.</td></tr> <tr><td>001A</td><td>001A0</td><td>Reserved.</td></tr> </tbody> </table>	16-bit protocol (hex)	20-bit protocol (hex)		0001	00010	The galvanometer scanner has reached a critical edge position.	0002	00020	AD converter error.	0003	00030	Temperature in scan system above max. allowed value.	0004	00040	External power supply voltages have dropped below the allowed value.	0005	00050	Flags are not valid.	0006	00060	Reserved.	...	...	...	000C	000C0	Reserved.	000D	000D0	Reserved.	000E	000E0	Position Acknowledge time out (set position not reached for long time).	000F	000F0	Reserved.	0014	00140	Reserved.	0015	00150	Reserved.	0016	00160	Reserved.	0017	00170	Reserved.	0018	00180	Reserved.	0019	00190	Reserved.	001A	001A0	Reserved.
16-bit protocol (hex)	20-bit protocol (hex)																																																										
0001	00010	The galvanometer scanner has reached a critical edge position.																																																									
0002	00020	AD converter error.																																																									
0003	00030	Temperature in scan system above max. allowed value.																																																									
0004	00040	External power supply voltages have dropped below the allowed value.																																																									
0005	00050	Flags are not valid.																																																									
0006	00060	Reserved.																																																									
...	...	...																																																									
000C	000C0	Reserved.																																																									
000D	000D0	Reserved.																																																									
000E	000E0	Position Acknowledge time out (set position not reached for long time).																																																									
000F	000F0	Reserved.																																																									
0014	00140	Reserved.																																																									
0015	00150	Reserved.																																																									
0016	00160	Reserved.																																																									
0017	00170	Reserved.																																																									
0018	00180	Reserved.																																																									
0019	00190	Reserved.																																																									
001A	001A0	Reserved.																																																									

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	2B	As of scan-system firmware $\geq$ 2060. Flags from last stop event, lower 16 bits		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	2C	As of scan-system firmware $\geq$ 2060. Flags from last stop event, upper 16 bits		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	2D	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	2E	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	2F	As of scan system firmware ≥ 2061. Running time (seconds)		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In s. Value range with 16-bit protocol: [0...59]. Unit with 20-bit protocol: In s. Value range with 20-bit protocol: [0...59].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	30	As of scan system firmware ≥ 2061. Running time (minutes)		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In min. Value range with 16-bit protocol: [0...59]. Unit with 20-bit protocol: In min. Value range with 20-bit protocol: [0...59].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	31	As of scan system firmware ≥ 2061. Running time (hours)		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In h. Value range with 16-bit protocol: [0...23]. Unit with 20-bit protocol: In h. Value range with 20-bit protocol: [0...23].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	32	As of scan system firmware ≥ 2061. Running time (days)		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In d. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: In d. Value range with 20-bit protocol: [0...65.535].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	33	As of scan system firmware $\geq$ 206x. Only intelliWELD: 3.3 V sensor board operating voltage		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In $10^{-3}$ V. Value range with 16-bit protocol: [0...7,000]. Unit with 20-bit protocol: In $10^{-3}$ V. Value range with 20-bit protocol: [0...7,000].
		—	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	34	As of scan system firmware $\geq$ 206x. Only intelliWELD: Sensor board operating temperature		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0.1 °C. Value range with 16-bit protocol: [0...65.535]. Unit with 20-bit protocol: In 0.1 °C. Value range with 20-bit protocol: [0...65.535].
		—	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	35	As of scan system firmware $\geq$ 206x. <b>Only intelliWELD:</b> Purge gas flow rate		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: $[(\text{purge gas flow rate} - 4 \text{ l/min}) * 109,89 \text{ min/l}]$ . Value range with 16-bit protocol: [0...2,250]. Unit with 20-bit protocol: $[\text{purge gas flow rate} - 4 \text{ l/min}] * 109,89 \text{ min/l}$ . Value range with 20-bit protocol: [0...2,250].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	36	As of scan system firmware $\geq$ 206x. [Temperature of mirror 2 + 26.6 °C]		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0.05 °C. Value range with 16-bit protocol: [0...1,992]. Unit with 20-bit protocol: In 0.05 °C. Value range with 20-bit protocol: [0...1,992].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	37	As of scan system firmware $\geq$ 206x. [Temperature of mirror 1 + 26.6 °C]		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0.05 °C. Value range with 16-bit protocol: [0...1,992]. Unit with 20-bit protocol: In 0.05 °C. Value range with 20-bit protocol: [0...1,992].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	38	As of scan system firmware $\geq$ 206x. Only intelliWELD: Protective-window temperature		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0,044 °C. Value range with 16-bit protocol: [0...2,250]. Unit with 20-bit protocol: In 0,044 °C. Value range with 20-bit protocol: [0...2,250].
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	39	As of scan system firmware $\geq$ 206x.  Only intelliWELD: Collimator temperature		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0.05 °C. Value range with 16-bit protocol: [0...2,250]. Unit with 20-bit protocol: In 0.05 °C. Value range with 20-bit protocol: [0...2,250].
		—	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	3A	As of scan system firmware $\geq$ 206x.  Only intelliWELD: Galvanometer scanner mount temperature		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0.05 °C. Value range with 16-bit protocol: [0...2,250]. Unit with 20-bit protocol: In 0.05 °C. Value range with 20-bit protocol: [0...2,250].
		—	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	3B	As of scan system firmware $\geq$ 206x. Only intelliWELD: [Coolant-flow rate + 0.93 l $\times$ min $^{-1}$ ]		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0,00525 l $\times$ min-1. Value range with 16-bit protocol: [0...2,250]. Unit with 20-bit protocol: [0...2,250]. Value range with 20-bit protocol: In 0,00525 l $\times$ min-1.
		—	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	3C	As of scan system firmware $\geq$ 206x. Only intelliWELD: Protective-window scattered light value		
		16-bit protocol	20-bit protocol	
		Bit #0 ... Bit #15	Bit #4 ... Bit #19	Unit with 16-bit protocol: In 0.00444 V. Value range with 16-bit protocol: [0...2,250]. Unit with 20-bit protocol: [0...2,250]. Value range with 20-bit protocol: In 0.00444 V.
		—	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	3D	As of scan system firmware $\geq$ 206x. <b>Only intelliWELD:</b> Flags sensor board Value range with 16-bit protocol: [0...65,535]. Value range with 20-bit protocol: [0...65,535].		
		16-bit protocol	20-bit protocol	
		Bit #15 (MSB)	Bit #19 (MSB)	= 1: Protective-window temperature value not in [0...1,882].
		Bit #14	Bit #18	= 1: Temperature of mirror 1 value not in [0...2,250].
		Bit #13	Bit #17	= 1: Temperature of mirror 2 value not in [0...2,250].
		Bit #10	Bit #14	= 1: Emerging-beam-opening temperature value not in [0...1,200]. Additionally, an INTERLOCK error is initiated.
		Bit #3	Bit #7	= 1: Protective-window scattered light value not in [0...2,250].
		Bit #2	Bit #6	= 1: Coolant-flow rate value not in [750...2,250].
		Bit #1	Bit #5	= 1: Galvanometer-mount temperature value not in [0...1,600]. Additionally, an INTERLOCK error is initiated.
		Bit #0	Bit #4	= 1: Collimator temperature value not in [0...2,000]. Additionally, an INTERLOCK error is initiated.
		–	Bit #0	= 0.
			...	...
			Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	3E	Reserved.		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	3F	As of scan system firmware ≥ 2072. Set scaling factor for position values		
		16-bit protocol	20-bit protocol	–
		Bit #2 ... Bit #15	Bit #6 ... Bit #19	Reserved.
		Bit #0 ... Bit #1	Bit #4 ... Bit #5	Scaling factor = $1/2^n$ . 16-bit protocol $n = 2 \times \text{Bit } \#1 + \text{Bit } \#0$ . 20-bit protocol $n = 2 \times \text{Bit } \#5 + \text{Bit } \#4$ .
		–	Bit #0 ... Bit #3	= 0.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	49	As of scan system firmware $\geq$ 5000. Supply voltage (usually 30 V or 48 V)		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
05	5E	As of scan system firmware $\geq$ 5000. Number of transitions “regular operation $\rightarrow$ fault condition”		
		16-bit protocol	20-bit protocol	–
		Bit #0 ... Bit #15	Bit #0 ... Bit #19	–

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system
05	64	<p>As of scan system firmware <math>\geq</math> 5100.</p> <p>Refer to "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual, Chapter 2.3 "Advanced Settings for "Spot Distance Control""", page 24: TimeShift relative to the SCANLAB default offset.</p>

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system
05	65	<p>As of scan system firmware <math>\geq</math> 5100.</p> <p>Refer to "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual, Chapter 2.3 "Advanced Settings for "Spot Distance Control""", page 24: TimeShift in total.</p>

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system												
05	74	<p>As of scan system firmware <math>\geq</math> 51*7 + &lt; 5090.</p> <p>For <b>Return Channel Multiplexing</b>. For use in conjunction with Code<sub>HIGH</sub> (hex) = 65<sub>H</sub> and Code<sub>HIGH</sub> (hex) = 66<sub>H</sub>. See corresponding Chapter incl. example code.</p>												
		<table border="1"> <tr> <td>16-bit protocol</td> <td>20-bit protocol</td> <td>–</td> </tr> <tr> <td>Bit #0</td> <td>Bit #0</td> <td>–</td> </tr> <tr> <td>...</td> <td>...</td> <td></td> </tr> <tr> <td>Bit #15</td> <td>Bit #19</td> <td></td> </tr> </table>	16-bit protocol	20-bit protocol	–	Bit #0	Bit #0	–	...	...		Bit #15	Bit #19	
16-bit protocol	20-bit protocol	–												
Bit #0	Bit #0	–												
...	...													
Bit #15	Bit #19													

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)	
0A	<p>As of scan system firmware <math>\geq</math> 2078.</p> <p><b>UpdatePermanentMemory</b></p> <p><b>UpdatePermanentMemory</b> causes the scan system to set the current servo behavior as the startup behavior for subsequent restarts or resets.</p> <p>See also <a href="#">Chapter 8.1.8 "Configuring the Start Behavior", page 235</a>.</p> <p>As of scan system firmware <math>\geq</math> 5050.</p> <p>Only excelliSCAN:</p> <p>excelliSCAN scan heads do not support <b>UpdatePermanentMemory</b>.</p>
Code <sub>LOW</sub> (hex)	
00	Only allowed value.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)				
OE	<p>As of scan system firmware <math>\geq</math> 206x.</p> <p><b>SetControlDefinitionMode</b></p> <p><b>SetControlDefinitionMode</b> causes the scan system to transmit information about a specific tuning (or the corresponding control algorithm) over the selected status channel.</p> <p>See also "On SetMode, SetControlDefinitionMode and SetEchoMode", page 1122.</p>			
	Code <sub>LOW</sub> (hex)	Data type transmitted by the scan system		
	00	As of scan system firmware $\geq$ 206x.		
	01	Tuning number		
	02	16-bit protocol	20-bit protocol	
	03	Bit #15 (MSB)	Bit #19 (MSB)	= 0: Tuning available. = 1: Tuning not available.
		Bit #4	Bit #8	Reserved.
		...	...	
		Bit #14	Bit #18	
		Bit #0	Bit #4	Tuning type
		...	...	= 0: Vector tuning.
		Bit #3	Bit #7	= 1: Jump tuning.
				= 2: Reserved.
				= 3: Reserved.
				= 4: As of scan system firmware $\geq$ 5050. SCANahead servo control.
		—	Bit #0	= 0.
			...	
			Bit #3	

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
Always consult the dedicated scan system manual!  
Module Rev.1.0.6 en-US



Code <sub>HIGH</sub> (hex)											
11	<p>As of scan system firmware <math>\geq</math> 2078.</p> <p><b>SelectControlDefinition</b></p> <p>Only scan systems equipped with several tunings (or corresponding servo algorithms).</p> <p><b>SelectControlDefinition</b> causes the scan systems to switch to a certain tuning. See also <a href="#">Chapter 8.1.4 "Selecting the Tuning (Dynamics Setting)", page 228</a>.</p> <p><b>Code<sub>LOW</sub></b> = 00...03 specifies the tuning number.</p> <p>As of scan system firmware <math>\geq</math> 5050.</p> <p>Only <b>excelliSCAN</b>:</p> <p>excelliSCAN scan heads do not support <b>SelectControlDefinition</b>. With excelliSCAN scan heads, there is only one tuning.</p>										
	<table border="1"> <thead> <tr> <th>Code<sub>LOW</sub> (hex)</th> <th></th> </tr> </thead> <tbody> <tr> <td>00</td><td>Tuning 0 (= default setting)</td></tr> <tr> <td>01</td><td>Tuning 1</td></tr> <tr> <td>02</td><td>Tuning 2</td></tr> <tr> <td>03</td><td>Tuning 3</td></tr> </tbody> </table>	Code <sub>LOW</sub> (hex)		00	Tuning 0 (= default setting)	01	Tuning 1	02	Tuning 2	03	Tuning 3
Code <sub>LOW</sub> (hex)											
00	Tuning 0 (= default setting)										
01	Tuning 1										
02	Tuning 2										
03	Tuning 3										

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	
12	<p>As of scan system firmware <math>\geq</math> 2072.</p> <p><b>SetPositionScale</b></p> <p><b>SetPositionScale</b> causes the scan system to switch to a certain scaling factor. The scan system servo electronics multiplies the actual position values received from the RTC-control board by this scaling factor. The actual position values sent back to the RTC-control board by the scan system are automatically divided by the same scaling factor. In this way, set position and actual position are always scaled to match each other. Important: You cannot detect a scaling factor <math>\neq 1</math> by comparing set position and actual position.</p> <p>See also <a href="#">Chapter 8.1.7 "Configuring the Effective Calibration", page 234</a>.</p> <p>Important: The scaling factor needs to be changed in 2 steps: 1. Execute <b>SetPositionScale</b> with <b>Code<sub>LOW</sub></b> = <math>83_{\text{H}}</math>. 2. Execute <b>SetPositionScale</b> with <b>Code<sub>LOW</sub></b> = [<math>00_{\text{H}}</math>...<math>03_{\text{H}}</math>] (= the desired scaling factor). See also <a href="#">"On SetPositionScale", page 1123</a>.</p> <p>As of scan system firmware <math>\geq</math> 5050.</p> <p>Only <b>SCANAhead</b> systems:</p> <p><b>SCANAhead</b> systems do not support <b>SetPositionScale</b>. With <b>SCANAhead</b> systems, the effective calibration cannot be changed.</p>
Code <sub>LOW</sub> (hex)	
00	The scaling factor is set to the value 1/1 (= no scaling; default setting)
01	The scaling factor is set to the value 1/2
02	The scaling factor is set to the value 1/4
03	The scaling factor is set to the value 1/8
83	Make ready for scaling factor change.

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	
15	<p>As of scan system firmware <math>\geq</math> 2066.</p> <p><b>SetPosAcknowledgeLevel</b></p> <p><b>SetPosAcknowledgeLevel</b> causes the scan system to switch to a certain <b>PosAck</b> limit value.</p> <p>See also <a href="#">Chapter 8.1.6 "Configuring the PosAck Limit Value", page 234</a>.</p> <p>See also <a href="#">"On SetPosAcknowledgeLevel", page 1123</a>.</p>
Code <sub>LOW</sub> (hex)	
00	Desired <b>PosAck</b> limit value in <b>LSB16</b> , 16-bit protocol. In bits.
...	
FF	

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	
17	<p>As of scan system firmware <math>\geq</math> 2078.</p> <p><b>Store/RestoreTransmissionMode</b></p> <p><b>Store/RestoreTransmissionMode</b> causes with <b>Code<sub>LOW</sub></b> = <b>FF<sub>H</sub></b> the scan system to cache the data type currently selected for transmission. It can be reinstated at a later time then with <b>Code<sub>LOW</sub></b> = <b>00<sub>H</sub></b>.</p> <p>See also <b>Data</b> = <b>0527<sub>H</sub></b>.</p> <p>See also <a href="#">"On Store/RestoreTransmissionMode", page 1123</a>.</p>
Code <sub>LOW</sub> (hex)	
FF	Store temporarily
00	Reinstate

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	
21	<p>As of scan system firmware <math>\geq</math> 2078.</p> <p><b>SetEchoMode</b></p> <p><b>SetEchoMode</b> verifies whether data transfer is intact.</p> <p>See also <a href="#">Chapter 8.1.9 "Fault Diagnosis and Functional Test", page 235</a>.</p> <p><b>SetEchoMode</b> passes an 8-bit value to the scan system with <b>Code<sub>LOW</sub></b>. As a consequence, the scan system transmits a 16-bit value (16-bit protocol) or 20-bit value (20-bit protocol) on the corresponding status channel to the RTC-control board. If the data transmission has been error-free, with this:</p> <ul style="list-style-type: none"> <li>• The upper 8 bits and <b>Code<sub>LOW</sub></b> are identical</li> <li>• With the 16-bit value, the lower 8 bits / with the 20-bit value, the next lower 8 bits and the complementary value of <b>Code<sub>LOW</sub></b> (NOT <b>Code<sub>LOW</sub></b>) are identical</li> </ul> <p>Example with 16-bit protocol: For <code>control_command(1, 1, 0x210A)</code> and if data transfer is error-free, <code>(get_value(1) AND 0xFFFF)</code> returns <code>0x0AF5</code>.</p> <p>Example with 20-bit protocol: For <code>control_command(1, 1, 0x210A)</code> and if data transfer is error-free, <code>(get_value(1) AND 0xFFFF0)</code> returns <code>0x0AF50</code>.</p> <p>See also <a href="#">"On SetMode, SetControlDefinitionMode and SetEchoMode", page 1122</a>.</p>

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	
40	<p>As of scan system firmware <math>\geq 5102 + \geq 5112</math>.</p> <p>Only excelliSCAN:</p> <p><b>ResetTrAck</b></p> <p><b>ResetTrAck</b> causes the scan system to reset the TrAck Signal bits.</p> <p>Refer to "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual, Chapter 2.1.3 "TrAck Signal", page 16. For more information, for example, about configuring the TrAck limit value, see the corresponding scan system manual.</p> <p>Note: The TrAck limit value cannot be saved for subsequent new starts or resets. excelliSCAN scan heads do not support <b>UpdatePermanentMemory</b>.</p>
Code <sub>LOW</sub> (hex)	
3A	<p>Only excelliSCAN:</p> <p>Reset TrAck Signal bits.</p>

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	
50	<p>As of scan system firmware <math>\geq 5100</math>.</p> <p>Only excelliSCAN:</p> <p>Refer to "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual, Chapter 2.3 "Advanced Settings for "Spot Distance Control"", page 24: <math>dT_{high}</math>.</p>

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)	
51	<p>As of scan system firmware <math>\geq 5100</math>.</p> <p>Only excelliSCAN:</p> <p>Refer to "excelliSCAN Scan Heads – Functional Principle of SCANAhead Servo Control and Operation by RTC6 Boards" Manual, Chapter 2.3 "Advanced Settings for "Spot Distance Control"", page 24: <math>dT_{low}</math>.</p>

Notice! Generic information. May be incomplete or even incorrect for your scan system.

Always consult the dedicated scan system manual!

Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)		
65	As of scan system firmware $\geq 51*7 + < 5090$ . For <b>Return Channel Multiplexing</b> . For use in conjunction with Code <sub>HIGH</sub> (hex) = $66_H$ and $0574_H$ . See corresponding Chapter incl. example code.	
Code <sub>LOW</sub> (hex)		
ON	Sets <b>Return Channel Multiplexing</b> block length to $N+1$ that is, highest <b>Return Channel Multiplexing</b> index = $N$ . In other words: number of <b>Return Channel Multiplexing</b> values the iDRIVE scan system has to transmit. Allowed values: 1; 2; 3; 4; 6; 8; 12; 16 (= integer divisor of $192 \geq 16$ ).	
8N	Sets the next to-be-configured <b>Return Channel Multiplexing</b> index to $N$ . In other words: Defines the position where the signal is transmitted.	

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
 Always consult the dedicated scan system manual!  
 Module Rev.1.0.6 en-US

Code <sub>HIGH</sub> (hex)		
66	As of scan system firmware $\geq 51*7 + < 5090$ . For <b>Return Channel Multiplexing</b> . For use in conjunction with Code <sub>HIGH</sub> (hex) = $65_H$ and $0574_H$ . See corresponding Chapter incl. example code.	
Code <sub>LOW</sub> (hex)		
NN	Sets the return value for the current index to $NN$ = writes the signal value to this index. For $NN$ , you can insert the same Code <sub>LOW</sub> values as for <b>SetMode</b> , see Code <sub>LOW</sub> = $00_H$ (XY2-100 status word), Code <sub>LOW</sub> = $01_H$ (Actual position (angular position of galvanometer scanners)) etc.	

Notice! Generic information. May be incomplete or even incorrect for your scan system.  
 Always consult the dedicated scan system manual!  
 Module Rev.1.0.6 en-US

Change Index		
Module Rev.1.0.5 en-US	$0536_H$	Not only with intelliWELD.
Module Rev.1.0.5 en-US	$0537_H$	Not only with intelliWELD.
Module Rev.1.0.6 en-US	$0500_H$	[Bit #15 Bit #19]. Editorial enhancement.
Module Rev.1.0.6 en-US	$0500_H$	[Bit #14 Bit #18]. Editorial enhancement.

### Comments (RTC5)

- The scan system firmware a.bb.c is read as abbc, for example, 5100 corresponds to version 5.10.0, and 5054 to version 5.05.4.
- On **SetMode**, **SetControlDefinitionMode**, **SetEchoMode** and **Store/RestoreTransmissionMode**
  - The data type selected by the `control_command(CodeHIGH = 05H, 0EH or 21H or Data = 1700H)` is transmitted until another data type is selected.
  - Data returned from the scan system to the RTC5 can be queried by **get\_value**, **get\_values**, **set\_trigger/set\_trigger4** and **get\_waveform**. The first data after switching the data source are transmitted only after a short time delay due to the serial transmission times. Therefore, a delay time of up to 60 s may occur between the switchover time and the first output of the new data type. **control\_command** always automatically inserts a waiting time of 60  $\mu$ s after the data source is switched (so that the previously mentioned commands now always return correct values).
  - All data returned from the scan system to the RTC5 are passed over as signed 20-bit values. This applies even if the **RTC5 DLL** is set to **RTC4 Compatibility Mode** and even for scan systems without SL2-100 interface, controlled by an **XY2-100 Converter (Accessory)**. Queried data returned by **get\_value**, **get\_values** or **get\_waveform** are nevertheless generally transferred to the PC as signed 32-bit values (for data evaluation, see comments for **get\_value**).
  - For scan systems with integrated SL2-100 interface, **get\_head\_status** queries the **XY2-100 status word** regardless of settings made by `control_command(CodeHIGH = 05H, 0EH or 21H or Data = 1700H)`. It is returned in addition to the selected data. In contrast, if iDRIVE scan systems (*without* an integrated SL2-100 interface) are controlled by an **XY2-100 Converter (Accessory)**, **get\_head\_status** returns the **XY2-100 status word** only if this

signal has been previously selected to be returned from the scan system by **control\_command** (see also [Section "Reading Out Data", page 199](#)).

- 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 **get\_value**). After a reset or power-up of the scan system, always the **XY2-100 status word** is returned.
- On **SetMode**
  - Set position (angular position) values returned by the scan system correspond to the effective output values `Sample<AX..BY>_Out` with **set\_trigger/set\_trigger4**. Additionally, the 20-bit set position values returned from a z axis (and in **RTC4 Compatibility Mode** from the x axis and y axis, too) are scaled by a factor 16 relating to the therefore specified 16-bit coordinate values.
  - The angle bit-values (actual position, set position, position error) or angle bit/ms-values (actual speed) returned to the RTC5 can be converted into °-values or °/ms-values by the scan system's calibration angle. The calibration angle can be read out by `Data = 0523H`.
  - Exact values for the internal voltages referred to in the table can vary for different versions of the scan system.
  - intelliWELD scan systems can be supplied with various number of sensors. Therefore, observe the manual of the respective scan system. This manual lists the command codes for currently available sensors.



- On **SetPositionScale**
  - If the scaling factor for scaling the position values is changed, then the user program calibration factor for calculating RTC5 control values (in bits) from the desired **Image field** position values (in mm) (see [Chapter 7.3.2 "Image Field Size and Image Field Calibration", page 156](#)) needs to be subjected to an inverse proportional scaling. In addition, the jump speed and mark speed should be adjusted.
  - The currently set scaling factor can be queried with **Data = 053F<sub>H</sub>**.
- On **SetPosAcknowledgeLevel**
  - The limit value must be specified related to a 16-bit position range.
  - The default start behavior is for the scan system to set the limit value to 0.28% of the *full* position range (16 bit, **Code<sub>LOW</sub>** = 183 = B7<sub>H</sub>) after every power-up or reset.  
If other limit values are desired, they must be separately set for each axis (**Code<sub>HIGH</sub>** = 15<sub>H</sub>). SCANLAB recommends setting only limit values (**Code<sub>LOW</sub>**) > 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).
- On **Store/RestoreTransmissionMode**
  - **control\_command( Data = 1700<sub>H</sub>** ) has no effect if a power-up or reset was executed after the most recent execution of **control\_command( Data = 17FF<sub>H</sub>** ).



## 18 Appendix C: Change Index

The following are changes in this manual due to the technical evolution of the product as well as significant editorial changes.

In this Chapter:

- Changes to Document Revision 1.9 en-US from Document Revision 1.8 en-US, page 791
- Changes to Document Revision 1.10 en-US from Document Revision 1.9 en-US, page 793
- Changes to Document Revision 1.11 en-US from Document Revision 1.10 en-US, page 795
- Changes to Document Revision 1.12 en-US from Document Revision 1.11 en-US, page 796
- Changes to Document Revision 1.13 en-US from Document Revision 1.12 en-US, page 797
- Changes to Document Revision 1.14.2 en-US from Document Revision 1.13 en-US, page 798
- Changes to Document Revision 1.14.3 en-US from Document Revision 1.14.2 en-US, page 800
- Changes to Document Revision 1.14.4 en-US from Document Revision 1.14.3 en-US, page 800
- Changes to Document Revision 1.14.5 en-US from Document Revision 1.14.4 en-US, page 801
- Changes to Document Revision 1.14.6 en-US from Document Revision 1.14.5 en-US, page 801
- Changes to Document Revision 1.14.7 en-US from Document Revision 1.14.6 en-US, page 802
- Changes to Document Revision 1.14.8 en-US from Document Revision 1.14.7 en-US, page 802
- Changes to Document Revision 1.14.9 en-US from Document Revision 1.14.8 en-US, page 803
- Changes to Document Revision 1.15.0 en-US from Document Revision 1.14.9 en-US, page 804
- Changes to Document Revision 1.15.1 en-US from Document Revision 1.15.0 en-US, page 805

### Changes to Document Revision 1.9 en-US from Document Revision 1.8 en-US

Where	Why & What
<a href="#">Chapter 7.5.2 "Marking Serial Numbers", page 196</a>	Expanded (as of DLL 537, OUT 537): Serial number marking now with selectable serial-number-sets.
<a href="#">Chapter 8.4 "Wobble Mode", page 217</a>	Expanded: specify direction of motion and "freely definable wobble shapes".
<a href="#">Chapter 8.4.1 "Wobble Shapes – Important Notes on Choosing Appropriate Parameter Values", page 219</a>	Added.
<a href="#">Section "Correction via McBSP Interface with Enhanced McBSP Input", page 231</a>	Processing-on-the-fly correction in z direction and laser power variation is now possible.
<a href="#">clear_fly_overflow, page 309</a>	Changed with version DLL 531, OUT 532: bits #4, 5.
<a href="#">config_laser_signals_list, page 312</a>	Added (as of DLL 537, OUT 537).
<a href="#">fly_return_z, page 324</a>	Added (as of DLL 531, OUT 532).
<a href="#">get_list_serial, page 346</a>	Added (as of DLL 537, OUT 537).
<a href="#">if_fly_z_overflow, page 379</a>	Added (as of DLL 531, OUT 532).
<a href="#">if_not_fly_z_overflow, page 383</a>	Added (as of DLL 531, OUT 532).
<a href="#">load_program_file, page 430</a>	Expanded (as of DLL 537, OUT 537): PCI error now has its own error code 16.
<a href="#">range_checking, page 487</a>	Added (as of DLL 537, OUT 537).
<a href="#">read_multi_mcbsp, page 496</a>	Added (as of DLL 537, OUT 537).
<a href="#">select_serial_set, page 518</a>	Added (as of DLL 537, OUT 537).
<a href="#">select_serial_set_list, page 518</a>	Added (as of DLL 537, OUT 537).
<a href="#">set_serial_step_list, page 616</a>	Added (as of DLL 537, OUT 537).
<a href="#">set_fly_limits_z, page 547</a>	Added (as of DLL 531, OUT 532).
<a href="#">set_multi_mcbsp_in, page 594</a>	Added (as of DLL 537, OUT 537).
<a href="#">set_multi_mcbsp_in_list, page 596</a>	Added (as of DLL 537, OUT 537).
<a href="#">set_sky_writing_para, page 621</a>	Changed parameter <code>Timelag</code> (as of DLL 537, OUT 537).



Where	Why & What
<a href="#">set_wobbel_control, page 649</a>	Added (as of DLL 537, OUT 537).
<a href="#">set_wobbel_direction, page 650</a>	Added (as of DLL 537, OUT 537).
<a href="#">set_wobbel_mode, page 651</a>	Changed with version DLL 537, OUT 537: Modes > 0.
<a href="#">set_wobbel_offset, page 653</a>	Added (as of DLL 537, OUT 537).
<a href="#">set_wobbel_vector, page 654</a>	Added (as of DLL 537, OUT 537).
Appendix C: Change Index, page 790	

### Changes to Document Revision 1.10 en-US from Document Revision 1.9 en-US

Where	Why & What
Chapter 2.8.8 "Extension Board "RTC5/6 varioSCAN FLEX Extension"", page 39	Added.
Chapter 2.9 "Supplementary Software", page 40	Expanded.
Chapter 4.6.6 "SPI / I <sup>2</sup> C Socket Connector", page 71	Expanded (as of DLL 538, OUT 538): The SPI/I <sup>2</sup> C socket connector now can also be initialized with SPI functionality: "SPI interface". "McBSP interface" is an established term since long times in this manual is extended with "/SPI" and used as such.
Section "Repeatedly Executed Calls", page 103	Added.
Chapter 6.9.1 "Free Variables", page 123	Changed: 8 free variables are possible as of DLL 539, OUT 539.
Section "Sky Writing Mode 3", page 151	Expanded.
Figure 52, page 150	Corrected.
Section "Precalculation and Diagnosis", page 171	Added.
Chapter 8.11 "Camming", page 255	Added.
Chapter 9.4 "Periodical I/O Signals", page 277	Added.
auto_cal, page 301	Text added: error code 9, 90.
camming, page 307	Added (as of DLL 512, OUT 511).
get_free_variable, page 334	Changed (as of DLL 539, OUT 539): value range of parameter <i>No</i> increased.
get_galvo_controls, page 335	Added (as of DLL 539, OUT 539).
get_head_status, page 338	Expanded.
mcbsp_init_spi, page 462	Added (as of DLL 538, OUT 538).
periodic_toggle, page 484	Added (as of DLL 538, OUT 538).
periodic_toggle_list, page 485	Added (as of DLL 538, OUT 538).
read_abc_from_file, page 489	Added (as of DLL 538, OUT 538).
set_free_variable, page 556	Changed (as of DLL 539, OUT 539): value range of parameter <i>No</i> increased.



Where	Why & What
<a href="#">set_trigger, page 634</a>	Changed (as of DLL 538, OUT 538): Signal1, Signal2 = 45 and 46.
<a href="#">set_trigger, page 634</a>	Changed (as of DLL 539, OUT 539): Signal1, Signal2 = 47...51.
<a href="#">set_wobbel_vector, page 654</a>	Corrected: formula for variation of the current laser power P within the wobbel shape section.
<a href="#">sub_call_abs_repeat, page 677</a>	Added (as of DLL 538, OUT 538).
<a href="#">sub_call_repeat, page 678</a>	Added (as of DLL 538, OUT 538).
<a href="#">write_abc_to_file, page 715</a>	Added (as of DLL 538, OUT 538).
<a href="#">write_hi_pos, page 720</a>	Added (as of DLL 538, OUT 538).
<a href="#">Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734</a>	Section SPI/I <sup>2</sup> C Connector: SPI configuration values added.
"Compliance with EU Directive for Electromagnetic Compatibility (EMC)"	Added.
<a href="#">Appendix C: Change Index, page 790</a>	



### Changes to Document Revision 1.11 en-US from Document Revision 1.10 en-US

Where	Why & What
Chapter 2.3 "Delivered RTC5 Software Package", page 27 and others	Expanded (Windows 10 is supported).
Section "Non-Indexed Subroutines", page 101	Expanded.
Section "Example Code (Delphi)", page 116	Added (identification of master/slave boards).
Section "Speed-Dependent Laser Control", page 191	Expanded (note on <code>set_auto_laser_control</code> Mode = 6).
<code>control_command</code> , page 315	Changed: formula for coolant-flow rate at <code>053B</code> <sub>H</sub> .
<code>control_command</code> , page 315	Changed: formula for protective-window scattered light value at <code>053C</code> .
<code>list_call_abs_repeat</code> , page 400	Added (as of DLL 540, OUT 540).
<code>list_call_repeat</code> , page 402	Added (as of DLL 540, OUT 540).
<code>move_to</code> , page 467	Command description has been moved to the manual "Installation and Operation RTC Step Motor Extension for the RTC4 and RTC5 PC interface boards" (available in English only).
<code>set_auto_laser_control</code> , page 522	Changed (as of DLL 540, OUT 540): Mode = 6.
<code>stepper_enable</code> , page 667	Corrected: <code>stepper_enable</code> does not have a result.
Chapter 17.4 "Technical Specifications – RTC5 PCIe/104 Board"	Expanded (note on <code>ANALOG OUT1</code> and <code>ANALOG OUT2</code> ).
Appendix C: Change Index, page 790	



## Changes to Document Revision 1.12 en-US from Document Revision 1.11 en-US

Where	Why & What
<b>Chapter 2.3 "Delivered RTC5 Software Package", page 27</b>	Expanded (the RTC5 software package now also contains files for CMake).
<b>Chapter 5.3.2 "Exchange of RTC Boards and Update of RTC Board Driver", page 79</b>	Added. For further information, refer to the (updated as of version DLL 542) file <a href="#">ReadMe_ScanlabClassChecker.pdf</a> .
<b>Section "Coordinate Transformations in the Virtual Image Field", page 158</b>	Expanded (coordinate transformations in the virtual <b>Image field</b> are now also available with <b>set_fly_x</b> and/or <b>set_fly_y</b> ; as of version DLL 541, OUT 541).
<b>Chapter 8.6.11 "Processing-on-the-fly Correction for the z Axis", page 242</b>	Added.
<b>Chapter 8.12 "Time Measurements", page 257</b>	Added.
<b>control_command, page 315</b>	Changed: value range decreased (from 2250 to 1992) for temperature of mirror 2 at <b>0536<sub>H</sub></b> .
<b>control_command, page 315</b>	Changed: value range decreased (from 2250 to 1992) for temperature of mirror 1 at <b>0537<sub>H</sub></b> .
<b>get_lap_time, page 343</b>	Added (as of DLL 541, OUT 541).
<b>list_next, page 409</b>	Added (as of DLL 541, OUT 541).
<b>set_fly_z, page 555</b>	Added (as of DLL 530, OUT 531).
<b>set_pulse_picking_list, page 610</b>	Corrected: this command is an undelayed short list command.
<b>set_trigger4, page 642</b>	Corrected: this command is a delayed short list command.
<b>stepper_disable_switch, page 666</b>	Added (as of DLL 542, OUT 542).
<b>Appendix C: Change Index, page 790</b>	

### Changes to Document Revision 1.13 en-US from Document Revision 1.12 en-US

Where	Why & What
Chapter 4.6.3 "EXTENSION 2 Socket Connector", page 67	Corrected: signal at pin (26) is LASERON (not: NOT CONNECTED). <a href="#">Figure 24</a> .
Chapter 4.6.8 "Analog Inputs (Accessory)", page 75	Expanded: The analog input values (ANALOG IN0 and ANALOG IN1, see <a href="#">read_analog_in</a> ) can be recorded with signal 54 (see <a href="#">set_trigger</a> or <a href="#">set_trigger4</a> ; as of version DLL 543, OUT 543, RBF 524).
Chapter 8.4 "Wobbel Mode", page 217	Expanded: The present wobbel amplitude (from <a href="#">set_wobbel</a> or <a href="#">set_wobbel_mode</a> ) can be recorded with signal 53 (see <a href="#">set_trigger</a> or <a href="#">set_trigger4</a> ; as of version DLL 543, OUT 543, RBF 524).
Chapter 8.12 "Time Measurements", page 257	Expanded: 32-bit "Timestamp Counter". Can be recorded with trigger signal 52 (see <a href="#">set_trigger</a> or <a href="#">set_trigger4</a> ; as of version DLL 543, OUT 543, RBF 524).
Chapter 9.3.2 "Conditional Command Execution", page 271	Expanded: control command <a href="#">set_pause_list_cond</a> (as of DLL 543, OUT 543).
<a href="#">periodic_toggle</a> , page 484	Changed: endless toggling with <code>Count = 2<sup>32</sup>-1</code> (as of DLL 543, OUT 543).
<a href="#">periodic_toggle_list</a> , page 485	Changed: endless toggling with <code>Count = 2<sup>32</sup>-1</code> (as of DLL 543, OUT 543).
<a href="#">set_control_mode</a> , page 528	Expanded: Bit #4 (as of DLL 543, OUT 543).
<a href="#">set_control_mode_list</a> , page 530	Expanded: Bit #4 (as of DLL 543, OUT 543).
<a href="#">set_pause_list_cond</a> , page 601	Added (as of DLL 543, OUT 543).
<a href="#">set_trigger</a> , page 634	Changed: <code>Signal1, Signal2 = 52...54</code> (as of DLL 543, OUT 543, RBF 524).
<a href="#">set_trigger4</a> , page 642	Changed: <code>Signal1, Signal2 = 52...54</code> (as of DLL 543, OUT 543, RBF 524).
Section "EXTENSION 2 Socket Connector", page 736	Corrected: missing signal LASERON added.
Appendix C: Change Index, page 790	

**Changes to Document Revision 1.14.2 en-US from Document Revision 1.13 en-US**

Where	Why & What
Global	Document Revision 1.14.2 en-US applies to RTC5 software package RTC5_Software_2020_11_13.zip <sup>(a)</sup>
Global	Editorial change. The previously used term "Online Positioning" (due to the introduction of "Global Online Positioning") now reads - where applicable - "Local Online Positioning", for example, in <a href="#">Chapter 8.3.1</a> "Local Online Positioning", page 213.
Global	All information explicitly referring to DLL 537, OUT 537 and earlier has been removed from this document. However, this information can still be taken from <a href="#">RTC5_Software_RevisionHistory_&lt;Date&gt;.pdf</a> .
<a href="#">Chapter 1 "About this Manual", page 21</a>	Software change.
<a href="#">Chapter 4.5.2 "XY2-100 Converter (Accessory)", page 56</a>	Corrected: SCANLAB recommends that cables for data transmission via XY2-100 should be less than 20 m long.
<a href="#">Chapter 4.5.3 "Data Cables (Accessories)", page 58</a>	Corrected: SCANLAB recommends that cables for data transmission via XY2-100 should be less than 20 m long.
<a href="#">Chapter 4.6.1 "LASER Connector", page 60</a>	Note for laserDESK users on external documents, <a href="#">page 60</a> .
<a href="#">Chapter 8.2 "Coordinate Transformations", page 209</a>	Editorial enhancement. Notes on data recording, <a href="#">page 211</a> .
<a href="#">Chapter 8.3.2 "Global Online Positioning", page 216</a>	Added.
<a href="#">activate_fly_2d_encoder</a> , page 293	Added (as of DLL 544, OUT 544).
<a href="#">activate_fly_xy_encoder</a> , page 294	Added (as of DLL 544, OUT 544).
<a href="#">control_command</a> , page 315	Editorial change. Description of "PowerOK" and "TempOK" has been changed. Furthermore, the description of $0528_H$ , $0529_H$ , $052B_H$ and $052C_H$ has been removed (these data signal types are not intended for users).
<a href="#">control_command</a> , page 315	Changed: in the formula for $053B_H$ , the value is now "0.00525 $I \times \text{min}^{-1}$ " (additional decimal place, before: "0.0052 $I \times \text{min}^{-1}$ ").
<a href="#">control_command</a> , page 315	Changed significance of $0535_H$ for intelliWELD-systems delivered as of October 2018. For these, the previous meaning "emerging-beam-opening temperature" no longer applies. The new meaning is now "purge air flow rate".
<a href="#">get_startstop_info</a> , page 356	Changed: <a href="#">Bit #14, page 357</a> (as of DLL 546, OUT 546).
<a href="#">load_program_file</a> , page 430	Editorial enhancement. Links to the state of the various output ports inserted, <a href="#">page 431</a> .

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).

Where	Why & What
<a href="#">number_of_correction_tables, page 468</a>	Editorial enhancement. Command has not been documented earlier due to the limited number of users.
<a href="#">set_defocus_offset, page 532</a>	Added (as of DLL 540, OUT 540).
<a href="#">set_defocus_list, page 532</a>	Editorial enhancement. New comment.
<a href="#">set_defocus_offset_list, page 533</a>	Added (as of DLL 540, OUT 540).
<a href="#">set_mcbsp_global_matrix, page 581</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_mcbsp_global_matrix_list, page 581</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_mcbsp_global_rot, page 582</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_mcbsp_global_rot_list, page 582</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_mcbsp_global_x, page 583</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_mcbsp_global_x_list, page 583</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_mcbsp_global_y, page 584</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_mcbsp_global_y_list, page 584</a>	Added (as of DLL 545, OUT 545).
<a href="#">set_pause_list_cond, page 601</a>	A conditional <b>pause_list</b> now takes precedence over a simultaneously present /STOP signal (as of DLL 544, OUT 544).
<a href="#">set_pause_list_not_cond, page 602</a>	Added (as of DLL 544, OUT 544).
<a href="#">set_port_default_list, page 608</a>	Added (as of DLL 544, OUT 544).
<a href="#">wait_for_encoder_mode, page 711</a>	Corrected: This command is not a normal list command but a multiple list command.
<a href="#">Chapter 15 "Technical Specifications – RTC5 PCI Board", page 734</a>	Added: HIGH level and LOW level of /START and /STOP.
"Compliance with EU Directive for Electromagnetic Compatibility (EMC)"	Editorial change. EU Directive 2014/30/EU.
"Compliance with EU Directive for Electromagnetic Compatibility (EMC)"	Editorial change. EU Directive 2014/30/EU.
<a href="#">Chapter 1.3 "Glossary and Abbreviations", page 23</a>	Added.
<a href="#">Appendix C: Change Index, page 790</a>	

**Changes to Document Revision 1.14.3 en-US from Document Revision 1.14.2 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"> <li>• 1.14.3 en-US</li> </ul> applies to RTC5 software package <ul style="list-style-type: none"> <li>• RTC5_Software_2020_11_13.zip<sup>(a)</sup></li> </ul>
Chapter 14.1 "EU Declaration of Conformity – RTC5 PCI Board", page 732	Editorial change. Replaces Chapter "Compliance with EC Directive for Electromagnetic Compatibility (EMC)".
Chapter 16.4.1 "EU Declaration of Conformity – RTC5 PCIe Board", page 744	Editorial change. Replaces Chapter "Compliance with EC Directive for Electromagnetic Compatibility (EMC)".
Appendix C: Change Index, page 790	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).

**Changes to Document Revision 1.14.4 en-US from Document Revision 1.14.3 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"> <li>• 1.14.4 en-US</li> </ul> applies to RTC5 software package <ul style="list-style-type: none"> <li>• RTC5_Software_2020_11_13.zip<sup>(a)</sup></li> </ul>
Chapter 2.10 "Notes for RTC4 Users", page 40	Editorial enhancement. Encoder counter direction of RTC4 boards, see <a href="#">Notice!</a> , page 49.
<a href="#">list_repeat</a> , page 410	Editorial enhancement. New comment, <a href="#">page 410</a> .
<a href="#">clear_fly_overflow</a> , page 309	Editorial change. Mode = 63 (not: = 15).
<a href="#">control_command</a> , page 315	Editorial change. Content moved to Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747.
Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747	Editorial change. <a href="#">052A<sub>H</sub></a> : Stop Event Code <a href="#">000D0</a> is not intended for users.
Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747	Editorial enhancement.
Appendix C: Change Index, page 790	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).



**Changes to Document Revision 1.14.5 en-US from Document Revision 1.14.4 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"> <li>• 1.14.5 en-US</li> </ul> applies to RTC5 software package <ul style="list-style-type: none"> <li>• RTC5_Software_2021_10_22.zip<sup>(a)</sup></li> </ul>
<a href="#">Chapter 1 "About this Manual", page 21</a>	Software change.
<a href="#">clear_fly_overflow_ctrl, page 309</a>	Added (as of DLL 548, OUT 548).
<a href="#">Appendix C: Change Index, page 790</a>	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).

**Changes to Document Revision 1.14.6 en-US from Document Revision 1.14.5 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"> <li>• 1.14.6 en-US</li> </ul> applies to RTC5 software package <ul style="list-style-type: none"> <li>• RTC5_Software_2021_10_22.zip<sup>(a)</sup></li> </ul>
<a href="#">Chapter 2.8.6 "Slot Cover with 15-pin D-SUB Connector and 9-pin D-SUB Connector", page 39</a>	Editorial enhancement. #0130209.
<a href="#">Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747</a>	Editorial change. Module Rev.1.0.6 en-US.
<a href="#">Appendix C: Change Index, page 790</a>	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).



**Changes to Document Revision 1.14.7 en-US from Document Revision 1.14.6 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"> <li>• 1.14.7 en-US</li> </ul> applies to RTC5 software package <ul style="list-style-type: none"> <li>• RTC5_Software_2021_12_30.zip<sup>(a)</sup></li> </ul>
<a href="#">Chapter 1 "About this Manual", page 21</a>	Software change.
<a href="#">set_multi_mcbsp_in, page 594</a>	Software change. <code>Mode = 2</code> .
<a href="#">Appendix C: Change Index, page 790</a>	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).

**Changes to Document Revision 1.14.8 en-US from Document Revision 1.14.7 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"> <li>• 1.14.8 en-US</li> </ul> applies to RTC5 software package <ul style="list-style-type: none"> <li>• RTC5_Software_2022_01_28.zip<sup>(a)</sup></li> </ul>
<a href="#">Chapter 1 "About this Manual", page 21</a>	Software change.
<a href="#">Chapter 7.3.4 "3D Image Field", page 159</a>	Editorial enhancement. ABC values from the <code>*_ReadMe.txt</code> file of correction file are to be interpreted as 16-bit values.
<a href="#">set_mcbsp_out_ptr_list, page 590</a>	Software change. <sup>(a)</sup> New command.
<a href="#">set_multi_mcbsp_in, page 594</a>	Software change. <sup>(a)</sup> <code>Mode = 2</code> changed.
<a href="#">Appendix C: Change Index, page 790</a>	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).



**Changes to Document Revision 1.14.9 en-US from Document Revision 1.14.8 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"><li>• 1.14.9 en-US</li></ul> applies to RTC5 software package <ul style="list-style-type: none"><li>• RTC5_Software_2022_03_09.zip<sup>(a)</sup></li></ul>
<a href="#">Chapter 1 "About this Manual", page 21</a>	Software change. <a href="#">Appendix C: Change Index, page 790</a>
<a href="#">Appendix C: Change Index, page 790</a>	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).



**Changes to Document Revision 1.15.0 en-US from Document Revision 1.14.9 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"> <li>• 1.15.0 en-US</li> </ul> applies to RTC5 software package <ul style="list-style-type: none"> <li>• RTC5_Software_2022-11-11<sup>(a)</sup></li> </ul>
Global	Software change. <sup>(a)</sup> Microsoft Vista, XP SP2, XP SP3 and earlier are no longer supported.
Global	RTC5 PC/104-Plus-Boards: Last support was 2023-12-31. Information removed.
Global	RTC5 PCIe/104-Boards: Last support was 2023-12-31. Information removed.
Chapter 1 "About this Manual", page 21	Software change. <sup>(a)</sup>
Chapter 4.4 "Master Socket Connector, Slave Socket Connector", page 53	Hardware change. #0149963 (previously: #0117241).
get_list_pointer, page 345	Editorial enhancement. New comment, <a href="#">page 345</a> .
set_auto_laser_params, page 525	Editorial enhancement. <b>Result.</b>
set_laser_control, page 566	Editorial enhancement. Further clarification with <b>Bit #3</b> and <b>Bit #4</b> .
set_vector_control, page 643	Software change. <sup>(a)</sup> <b>Ctrl</b> = 8.
wait_for_encoder_mode, page 711	Editorial change. <b>wait_for_encoder_mode</b> is category "Normal List Command" (not: "Multiple List Command").
Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747	Editorial change. Module Rev.1.0.6 en-US.
<a href="#">Appendix C: Change Index, page 790</a>	

(a) See also [RTC5\\_Software\\_RevisionHistory\\_<Date>.pdf](#).



**Changes to Document Revision 1.15.1 en-US from Document Revision 1.15.0 en-US**

Where	Why & What
Global	Document Revision <ul style="list-style-type: none"><li>• 1.15.1 en-US</li></ul> applies to RTC5 software package <ul style="list-style-type: none"><li>• RTC5_Software_2024-09-27<sup>(a)</sup></li></ul>
Chapter 1 "About this Manual", page 21	Software change. <sup>(a)</sup>
get_marking_info, page 347	Editorial change. Text of Bit #17 and Bit #18 was mixed up.
set_ellipse, page 537	Software change. <sup>(a)</sup> See Version info, page 537.
Chapter 17 "Appendix B: iDRIVE Scan Systems – Control Commands and Signals Transmitted to RTC Control Boards", page 747	Editorial change. Module Rev.1.0.6 en-US.
Appendix C: Change Index, page 790	

(a) See also RTC5\_Software\_RevisionHistory\_<Date>.pdf.



## Notes