



syncAXIS-DLL – Schnittstelle zur Anwendungsprogrammierung

syncAXIS control **V1.8.0**

SCANLAB GmbH
Siemensstr. 2a
82178 Puchheim
Deutschland

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

info@scanlab.de
www.scanlab.de

© SCANLAB GmbH

(Doc. Rev. 1.9.20 de-DE - 2022-11-16)

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

Alle erwähnte Marken unterliegen dem Markenschutz der jeweiligen Markeninhaber.

Inhalt

1	Über dieses Handbuch	9
1.1	Weiterführende Dokumente	9
1.2	Hersteller	9
1.3	Überblick	10
1.4	Glossar	11
2	Softwareentwicklung mit der syncAXIS-DLL	16
2.1	Sicherheit	16
2.2	Über die SICHERE Verwendung von syncAXIS control – Allgemeines Vorgehen	18
	syncAXIS control Sicherheitsfeatures	18
2.2.1	Systemgrenzen ermitteln	19
2.2.2	Sicherheitsmechanismen einrichten	19
2.2.3	Sichere syncAXIS control Instanzen konfigurieren	20
2.2.4	Jobs simulieren und nachbessern	24
2.3	Über die Hauptstrukturen eines syncAXIS-DLL-basierten Anwenderprogramms (Beispielhaft)	25
2.3.1	Einzuhaltende Struktur bei der Job-Definition	25
2.4	Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen	26
2.5	Über den syncAXIS control Simulationsmodus	31
2.6	Über das Optimieren von syncAXIS control-basierten Anwenderprogrammen	36
2.6.1	Optimierungsmöglichkeiten	36
2.6.2	Iteratives Vorgehen	38
2.7	Über Vorgänge bei der Ausführung des Anwenderprogramms	41
2.7.1	Über die Puffer der syncAXIS control-Instanzen	42
	Pufferunterläufe vermeiden	42
2.7.2	Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden	45
2.8	Über das Logging in syncAXIS control	47
2.9	Über das automatische Steuern des Lasers durch syncAXIS control (“Automatische Lasersteuerung”)	48
2.9.1	Aktivierung der “Automatischen Lasersteuerung”	48
2.9.2	Definition der Channel und ActiveChannel	48
2.9.3	Über die Berechnung der ActiveChannel-Werte entlang einer Kontur	51
2.9.4	Über Rampen	53
	Beispiel – lineare (einfache) Rampe	54
	Beispiel – mehrteilige (komplexere) Rampe	57
2.9.5	Über die “Konturabhängige Geschwindigkeitsberechnung”	60
2.10	Über Heuristik und Kennlinien für Geschwindigkeits-Verminderungen	64
2.11	Über das Arbeiten mit “Modulen”	65
2.12	Über den Modus “Manuelle Positionierung”	70
2.12.1	Erlaubte/nicht erlaubte syncAXIS control-Funktionen	71
2.12.2	Beispiel – Verfahrtsch vorübergehend freigeben und Ziel-Verfahrtsch wechseln	74
3	In der API bereitgestellte Funktionen	76
3.1	Funktionale Übersicht	76
3.1.1	Konfigurations-Funktionen (slsc_cfg_*)	76
	syncAXIS control-Instanz-bezogene Funktionen	78
	Funktionenzur Konfigurationsänderung der bestehenden syncAXIS control-Instanz	80
	Funktionenzum Erfassen von “Callback Events”	81
3.1.2	Job-Funktionen (slsc_list_*)	85
	Funktionen zum Definieren von Job-Anfängen/Enden	85
	Funktionen zum Definieren von Sprüngen	85

Funktionen zum Definieren von Markierungen	87
[*]dashed[*]-Funktionen	91
Funktionen zum Ändern der Zielpunkt-Koordinaten	92
Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)	92
Funktionen zum Setzen von Signalen	93
Funktionen zum Ändern der Geschwindigkeiten	93
Funktion zum Ändern der Minimalen Geschwindigkeiten	93
Funktionen zum Ändern von Trajektorienplanungs-Werten	94
Funktionen zum Ändern des Verhaltens von Blending-Kurven	94
Funktion zur "Konturabhängigen Geschwindigkeitsberechnung"	94
Funktion zum Setzen des Werts einer freien Variablen auf der RTC6	95
Funktion zum Beeinflussen der Laserpulsausgabe über HalfPeriod/PulseLength	95
Funktionen für "Module"	95
3.1.3 Kontroll-Funktionen (slsc_ctrl_*)	96
Laser-bezogene Funktionen	96
Ausführungs-bezogene Funktionen	97
Korrekturdatei-bezogene Funktionen	99
Fehler-bezogene Funktionen	99
Funktionen zur Abfrage von Messwerten	99
Funktionen nur für den Modus "Manuelle Positionierung"	99
Funktionen zum Verwalten des Werts einer freien Variablen auf der RTC6	99
Funktionen zur Parameterwert-Optimierung	100
Funktionen zum Starten/Beenden des Modus "Manuelle Positionierung"	100
Funktionen zur Abfrage von Positionen	100
Simulationseinstellungs-bezogene Funktion	100
Funktionen zum Setzen von Signalen	100
Funktion zum Beeinflussen der Laserpulsausgabe über HalfPeriod/PulseLength	100
3.1.4 Utility-Funktionen (slsc_util_*)	101
RTC6-Karten-bezogene Funktion	101
3.2 Alphabetische Übersicht	102
3.3 Funktionsreferenz	114
3.3.1 Allgemeiner Aufbau der Referenztabelle	114
3.3.2 Datentypen der syncAXIS-DLL-Funktionen	115
3.3.3 Referenztabelle	117
slsc_cfg_acquire_stage (veraltet)	117
slsc_cfg_delete	118
slsc_cfg_delete_trajectory_config	119
slsc_cfg_get_calculation_dynamics_jump_scan_device	120
slsc_cfg_get_calculation_dynamics_mark_scan_device	121
slsc_cfg_get_calculation_dynamics_stage	122
slsc_cfg_get_dynamic_limits_scan_device	123
slsc_cfg_get_dynamic_limits_stage	124
slsc_cfg_get_dynamic_violation_reaction	125
slsc_cfg_get_field_limits_scan_device	126
slsc_cfg_get_field_limits_stage	127
slsc_cfg_get_jump_time	128
slsc_cfg_get_mode	130
slsc_cfg_get_operation_status	131
slsc_cfg_get_scan_device_dynamic_monitoring_level	132
slsc_cfg_get_simulation_setting	133
slsc_cfg_get_stage_dynamic_monitoring_level	134
slsc_cfg_get_sync_axis_version	135

slsc_cfg_get_trajectory_config	135
slsc_cfg_initialize_copy	136
slsc_cfg_initialize_from_file	137
slsc_cfg_register_callback_job_end_planned	139
slsc_cfg_register_callback_job_finished_executing	140
slsc_cfg_register_callback_job_is_executing	141
slsc_cfg_register_callback_job_loaded_enough	142
slsc_cfg_register_callback_job_progress_planned	143
slsc_cfg_register_callback_job_start_planned	144
slsc_cfg_reinitialize	145
slsc_cfg_reinitialize_from_file	147
slsc_cfg_release_stage (veraltet)	149
slsc_cfg_select_heuristic	152
slsc_cfg_select_stage	153
slsc_cfg_select_stage_axis (veraltet)	154
slsc_cfg_set_bandwidth	155
slsc_cfg_set_calculation_dynamics_jump_scan_device	156
slsc_cfg_set_calculation_dynamics_mark_scan_device	158
slsc_cfg_set_calculation_dynamics_stage	160
slsc_cfg_set_contour_dependent_speed_control_2d	162
slsc_cfg_set_dynamic_limits_scan_device	164
slsc_cfg_set_dynamic_limits_stage	165
slsc_cfg_set_dynamic_violation_reaction	166
slsc_cfg_set_field_limits_scan_device	167
slsc_cfg_set_field_limits_stage	168
slsc_cfg_set_jump_speed	170
slsc_cfg_set_list_handling_mode	171
slsc_cfg_set_list_handling_mode_with_context	173
slsc_cfg_set_mark_speed	174
slsc_cfg_set_matrix_and_offset	175
slsc_cfg_set_mode	176
slsc_cfg_set_part_displacement	177
slsc_cfg_set_rot_and_offset_2d	178
slsc_cfg_set_scan_device_dynamic_monitoring_level	179
slsc_cfg_set_simulation_setting	180
slsc_cfg_set_stage_dynamic_monitoring_level	181
slsc_cfg_set_trajectory_config	182
slsc_ctrl_disable_laser	183
slsc_ctrl_enable_laser	184
slsc_ctrl_follow	185
slsc_ctrl_get_error	186
slsc_ctrl_get_error_count	187
slsc_ctrl_get_exec_state	188
slsc_ctrl_get_free_variable	189
slsc_ctrl_get_job_characteristic	190
slsc_ctrl_get_scan_device_position	191
slsc_ctrl_get_simulation_filename	192
slsc_ctrl_get_stage_position	193
slsc_ctrl_get_syncaxis_simulation_filename	194
slsc_ctrl_get_value	195
slsc_ctrl_is_list_input_buffer_full	196
slsc_ctrl_laser_signal_off	197

slsc_ctrl_laser_signal_on	198
slsc_ctrl_move_scanner_abs	199
slsc_ctrl_move_stage_abs	200
slsc_ctrl_refresh_correction_file	201
slsc_ctrl_select_correction_file	202
slsc_ctrl_set_free_variable	203
slsc_ctrl_set_laser_pulses	204
slsc_ctrl_start_execution	205
slsc_ctrl_stop	206
slsc_ctrl_stop_controlled	207
slsc_ctrl_unfollow	208
slsc_ctrl_write_analog_x	209
slsc_ctrl_write_digital_out	210
slsc_ctrl_write_digital_out_mask	211
slsc_list_arc_abs	212
slsc_list_begin	214
slsc_list_begin_absolute	215
slsc_list_begin_module	217
slsc_list_begin_relative	218
slsc_list_circle_2d_abs	220
slsc_list_dashed_arc_abs	221
slsc_list_dashed_circle_2d_abs	223
slsc_list_dashed_mark_abs	225
slsc_list_end	226
slsc_list_jump_abs	227
slsc_list_jump_abs_min_time	228
slsc_list_mark_abs	229
slsc_list_multi_para_arc_abs	230
slsc_list_multi_para_circle_2d_abs	231
slsc_list_multi_para_dashed_arc_abs	232
slsc_list_multi_para_dashed_circle_2d_abs	234
slsc_list_multi_para_dashed_mark_abs	236
slsc_list_multi_para_mark_abs	238
slsc_list_para_arc_abs	239
slsc_list_para_circle_2d_abs	241
slsc_list_para_dashed_arc_abs	243
slsc_list_para_dashed_circle_2d_abs	245
slsc_list_para_dashed_mark_abs	247
slsc_list_para_disable	249
slsc_list_para_enable	250
slsc_list_para_jump_abs	251
slsc_list_para_jump_abs_min_time	252
slsc_list_para_mark_abs	253
slsc_list_para_playback_module	254
slsc_list_playback_module	255
slsc_list_set_approx_blend_limit	257
slsc_list_set_calculation_dynamics_jump_scan_device	258
slsc_list_set_calculation_dynamics_mark_scan_device	259
slsc_list_set_contour_dependent_speed_control_2d	260
slsc_list_set_free_variable	261
slsc_list_set_jump_speed	262
slsc_list_set_laser_on_move	263

slsc_list_set_laser_pulses	264
slsc_list_set_mark_speed	265
slsc_list_set_matrix_and_offset	266
slsc_list_set_min_mark_speed	267
slsc_list_set_rot_and_offset_2d	268
slsc_list_suppress_spotdistance_control	270
slsc_list_unsuppress_spotdistance_control	271
slsc_list_wait_with_laser_off	272
slsc_list_wait_with_laser_on	273
slsc_list_write_analog_x	274
slsc_list_write_digital_out	276
slsc_list_write_digital_out_mask	277
slsc_util_reset_pcie	278
4 Standard-Rückgabewerte der syncAXIS-DLL Funktionen	279
5 Fehlercodes (Error Codes) bei slsc_ctrl_get_error, Log-Datei und Konsole	282
6 Strukturen	289
slsc_GeometryConfig	289
slsc_MarkConfig	293
slsc_MultiParaTarget	297
slsc_ParaSection	298
slsc_TrajectoryConfig	299
VersionInfo	299
7 Aufzählungstypen enum	300
slsc_AnalogOutput	300
slsc_BlendModes	301
slsc_DynamicsMonitoringLevel	303
slsc_DynamicViolationReaction	304
slsc_ExecState	305
slsc_JobCharacteristic	306
slsc_ListHandlingMode	312
slsc_MeasurementSignal	313
slsc_OperationMode	314
slsc_OperationStatus	315
slsc_PositionType	315
slsc_ScanDevice	316
slsc_SimulationSetting	317
slsc_SplineModes	318
slsc_Stage	320
8 Anhang A: syncAXIS control V1.2.4 und höher mit XL SCAN-Multi-Head-Systemen nutzen	321
8.1 Über diesen Anhang	321
8.2 Verwendung von syncAXIS control V1.2.4 und höher	324
8.2.1 Voraussetzungen für diesen Anhang	324
8.2.2 syncAXISConfig.xml für syncAXIS control V1.2.4 und höher anpassen	324
Schritt 1 von 2: syncAXISConfig.xml technisch anpassen	324
Schritt 2 von 2: syncAXISConfig.xml inhaltlich anpassen	324
8.2.3 Weitere Anmerkungen zur Verwendung von syncAXIS control V1.2.4 und höher	331
8.3 Über Transformationen in syncAXIS control V1.2.4 und höher	332
9 Anhang B: Anwendungshinweis – Das Handling von Listen mit syncAXIS control	335
9.1 Listen-Handling-Modus "ReturnAtOnce"	337
9.2 Listen-Handling-Modus "RepeatWhileBufferFull"	339

9.3 Listen-Handling-Modus "RepeatWhilePredicate"	343
10 Anhang C: Anwendungshinweis – Texte markieren mittels Modulen	344
11 Anhang D: Anwendungshinweis – Pufferunterlauf vermeiden mittels Modulen	347
12 Anhang E: Anwendungshinweis – Benutzung der syncAXIS-DLL unter C#	350
12.1 Unterschiede in den syncAXIS-DLL-Funktionssignaturen	350
12.1.1 Notation der Datentypen	350
12.1.2 Zeiger-Ersetzungen für C#	350
12.2 Unterschiede bei der Verwendung von Callback-Funktionen	351
12.3 Code-Beispiel 1 (C#)	352
12.4 Code-Beispiel 2 (C#)	357
13 Anhang F: Referenz der syncAXISConfig.xml-Tags	358
13.1 xml-Struktur-Übersicht	358
13.2 xml-Tags	361
14 Änderungsindex	476

1 Über dieses Handbuch

Dieses Handbuch beschreibt die SCANLAB syncAXIS-DLL und ihre Verwendung zur Entwicklung und Ausführung von Anwenderprogrammen für die synchrone Ansteuerung eines Lasers, eines Scan-Kopfs (nur excelliSCAN) und eines Verfahrtschis in einem Laser-Scan-System.

Die syncAXIS-DLL ist (32-Bit-Version und 64-Bit-Version) Teil des syncAXIS control-Software-Pakets .

Zur Entwicklung von Anwenderprogrammen stellt sie eine Programmierschnittstelle (**API**) in Form von Funktionen bereit. Diese ermöglichen u.a.:

- Umfangreiche Konfigurationsmöglichkeiten für Systemparameter sowie der syncAXIS-DLL selber (Berechnung der Bewegungsdaten nur für den Scan-Kopf, nur den Verfahrtsch, oder beides)
- Die Definition von Bearbeitungs-**Jobs** (z. B. Beschriftungsmuster)
- Das Laden, Ausführen und Stoppen von **Jobs** – auch in einem Simulationsmodus als Datei-ausgabe
- Die Statusüberwachung von **Jobs**
- Erfassen von **Callback-Events**

Voraussetzungen für die Verwendung der syncAXIS-DLL:

- Installierte, konfigurierte und kalibrierte Hardware (siehe **Abbildung 1, Seite 10**): Scan-Kopf⁽¹⁾ (nur excelliSCAN), Verfahrtsch, **ACS**-Komponenten, RTC6 PCI-Express-Karte, Netzteile, Kabel, PC, **Dongle**
- Betriebssystem: MS Windows 7, 8, 10 (32-Bit und 64-Bit-Varianten)
- Installiertes syncAXIS control-Software-Paket⁽²⁾
- Installiertes **ACS**-Software-Paket

(1) syncAXIS-DLL kann zur Scan-Kopf-Kalibrierung genutzt werden.

(2) Das Paket enthält u.a. RTC6-Dateien (RTC6DAT.dat, RTC6DLL.dll, RTC6OUT.out, RTC6RBF.rbf). Diese dürfen nicht mit denjenigen RTC6-Dateien ersetzt werden, die bei der Auslieferung der RTC6-Karte mitgeliefert oder von der SCANLAB-Website heruntergeladen wurden.

Achtung!

Lesen Sie das Dokument "syncAXIS control Lizenzvertrag" sorgfältig durch, bevor Sie syncAXIS control installieren und verwenden. Diese Vereinbarung definiert Themen wie Nutzungsbedingungen, Garantieinformationen und Haftungsausschlüsse. Wenn Sie dazu Fragen haben, wenden Sie sich an SCANLAB.



Vorsicht!

Lesen und befolgen Sie alle Sicherheitshinweise in diesem Handbuch!

SCANLAB übernimmt keine Haftung für Schäden oder Folgeschäden aufgrund Nichtbeachtung dieses Handbuchs, insbesondere der hierin genannten Sicherheitshinweise.

1.1 Weiterführende Dokumente

- RTC6-Handbuch
- Handbuch "Installation der SCANLAB XL SCAN-Komponenten und Erstinbetriebnahme des XL SCAN-Systems"
- Handbuch "syncAXIS Viewer"
- Handbuch "syncAXIS Configurator"
- Handbuch "syncAXIS Master-Slave-Synchronizer"

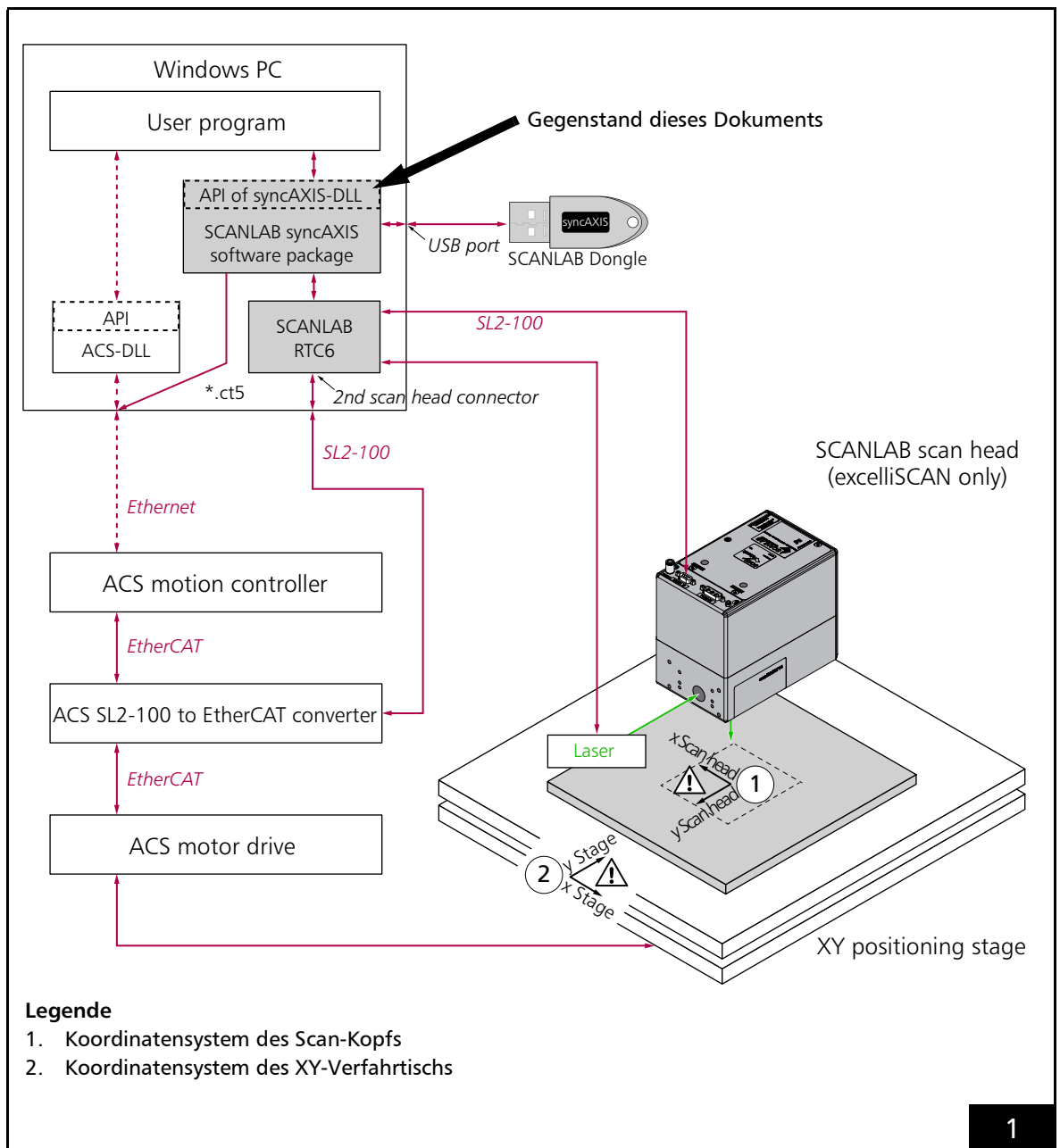
1.2 Hersteller

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

1.3 Überblick

Ein Szenario zur Anwendung der syncAXIS-DLL (als Teil des syncAXIS control-Software-Pakets) zeigt **Abbildung 1, Seite 10**.

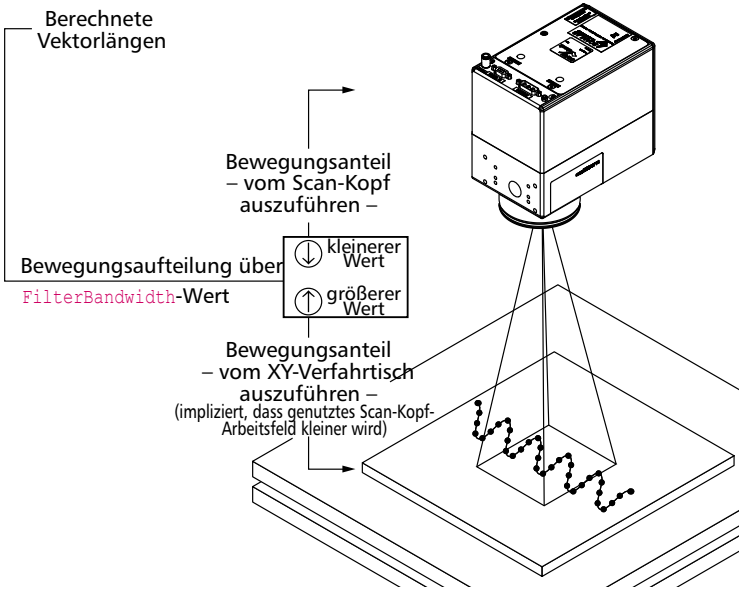
Veranschaulicht ist die gleichzeitige Steuerung eines 2D-Scan-Kopfs und eines mechanischen XY-Verfahrtischs mit zwei Servo-Achsen. Dieses kombinierte System vergrößert im Bezug zum Scan-Kopf-Arbeitsfeld den Arbeitsbereich und die Markiierungsergebnisse zeigen keinen Stitching-Fehler.



Anwendungsszenario der syncAXIS-DLL. ACS: siehe **Seite 11**.

Das gezeigte System hat nur 1 Verfahrtisch. Wahlweise kann ein weiterer Verfahrtisch hinzugefügt werden (eine weitere RTC6-Karte ist dazu nicht nötig und auch kein weiterer SL2-100 to EtherCAT-Konverter).

1.4 Glossar

ACS	Bezeichnet den Hersteller von Maschinensteuerungssystemen, dessen Komponenten aktuell verwendet werden müssen.
ACS Motion-Controller	XL SCAN-Komponente von ACS, siehe Abbildung 1, Seite 10 .
API	Abkürzung für Application Programming Interface ("Schnittstelle zur Anwendungsprogrammierung"). Programmteil (hier: der syncAXIS-DLL), der anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird (hier: Funktionen der syncAXIS-DLL). Siehe Kapitel 3 "In der API bereitgestellte Funktionen" , Seite 76 .
Ausgangspuffer	Der syncAXIS-DLL-interne Zwischenspeicher am Ende der Verarbeitungskette. Siehe Puffer und Abbildung 11, Seite 41 in Kapitel 2.7 "Über Vorgänge bei der Ausführung des Anwenderprogramms" , Seite 41 .
Benutzer	Bezeichnet eine Person (= "Anlagen-Programmierer"), die mittels syncAXIS control-Softwarepaket Anwenderprogramme entwickelt. Nicht gemeint ist der "Benutzer oder Betreiber einer syncAXIS control-Anlage".
Betriebsmodus	Siehe enum slsc_OperationMode .
Bewegungsaufteilung	<p>Betrifft nur den Betriebsmodus "ScannerAndStage". Anschließend an die Trajektorienplanung (= nachdem der Laser-Spot-Pfad berechnet ist; siehe Abbildung 11, Seite 41) folgt eine Zerlegung/Aufspaltung gemäß FilterBandwidth-Wert in einen durch den Scan-Kopf auszuführenden Anteil und einen durch den Verfahrtisch auszuführenden Anteil.</p> 
Callback-Event	Eines von mehreren syncAXIS-DLL-internen Ereignissen in Abbildung 12, Seite 43 und Tabelle auf Seite 81 , die auftreten, wenn ein Job seinen Berechnungs-, Transfer- und Ausführungs-Status durchläuft.

Callback-Funktion (Rückruffunktion)	Bezeichnet eine benutzerdefinierte Funktion, die ausgeführt werden soll, wenn ein bestimmtes “Callback-Event“ auftritt. Eine “Callback-Funktion“ wird über eine “Funktion zum Erfassen von “Callback-Events““ bei der syncAXIS control-Instanz angemeldet. Sie muss deshalb einer vorgeschriebenen Funktions-Signatur (slsc_JobCallback oder slsc_ExecTimeCallback) entsprechen.
Dongle	Kurz für “SCANLAB-USB-Dongle für XL SCAN“ . USB-Kopierschutzstecker, der individuelle Lizenzierungsinformationen enthält, siehe Multi-Instanz und Multi-Stage . Kein Speichermedium! Ab syncAXIS control ≥ V1.2 ist eine weitere Variante erhältlich (#139941 “Office-Dongle“). Damit können Benutzer die Software ohne Hardware-Anbindung am lokalen PC nutzen, z.B. um Simulationsdateien zu erstellen (d. h. Ansteuerwerte generieren).
Eingangspuffer	Der syncAXIS-DLL-interne Zwischenspeicher am Beginn der Verarbeitungskette. Siehe Puffer und Abbildung 11, Seite 41 in Kapitel 2.7 “Über Vorgänge bei der Ausführung des Anwenderprogramms“, Seite 41 .
Execution Layer	syncAXIS-DLL-interne Teilmodul, das für die Kommunikation mit der RTC6-Karte und deren Überwachung zuständig ist. Sein Zustand kann mit slsc_ctrl_get_exec_state abgefragt werden. Bei einem slsc_ctrl_start_execution -Aufruf triggert es die Job-Ausführung .
Funktion zum Erfassen von “Callback-Events“	Eine der syncAXIS-DLL-Konfigurations-Funktionen mit Präfix slsc_cfg_register_callback_ , siehe Seite 81 . Für jedes “Callback-Event“ gibt es eine entsprechende “Funktion zum Erfassen von “Callback-Events““ , mit der die auszuführende “Callback-Funktion“ angegeben wird. So können Job -bezogene Informationen erfasst und darauf reagiert werden.
Handle	Begriff aus der Programmierung: Abstrakte Referenz auf eine Ressource. Bezieht sich in diesem Handbuch (meistens) auf eine bestimmte syncAXIS control-Instanz . Deren Handle-Wert wird durch eine Initialisierungsfunktion zugeteilt. Bei den (meisten) syncAXIS control-Funktionen muss der Handle-Wert der gewünschten Ziel- syncAXIS control-Instanz angegeben werden.
Heuristik	<p>Für ≥ V1.5.0 gilt:</p> <ul style="list-style-type: none"> Ist ein syncAXIS-DLL-interner Algorithmus Ist nur im Betriebsmodus ScannerAndStage wirksam Setzt mindestens 1 definierte Kennlinie voraus, siehe DynamicReductionFunction Wertet Sprung-Segmente und Markier-Segmente aus <ul style="list-style-type: none"> Wahlweise nur Sprung-Segmente wenn in der syncAXISConfig.xml eingestellt ist: <code><cfg:HeuristicForJumpsOnly>true</cfg:HeuristicForJumpsOnly></code> Siehe auch Kapitel 2.10 “Über Heuristik und Kennlinien für Geschwindigkeits-Verminderungen“, Seite 64. <p>Die Heuristik:</p> <ol style="list-style-type: none"> (1) Ermittelt die räumliche Ausdehnung jedes Segments = Distanz zwischen Anfangspunkt und Endpunkt (Ausnahme: Durchmesser des Kreises bei kreisförmigen Segmente ([*]arc[*], [*]circle[*]), die mehr als einen Halbkreis überstreichen). (2) Prüft die Kennlinie (= DynamicReductionFunction), ob für das Ergebnis aus (1) (= Length-Wert) ein entsprechender Velocity-Wert vorhanden ist. (3) Nur wenn der Velocity-Wert aus (2) kleiner als die ursprüngliche Markiergeschwindigkeit ist: die Heuristik verringert die Markiergeschwindigkeit des Segments auf diesen Velocity-Wert.

Initialisierungsfunktion	Sammelbegriff für syncAXIS-DLL-Funktionen, die eine syncAXIS control-Instanz aufbauen: slsc_cfg_initialize_copy und slsc_cfg_initialize_from_file , sowie slsc_cfg_reinitialize und slsc_cfg_reinitialize_from_file . Siehe auch Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen", Seite 26.
Job	Bezeichnet eine zwingend einzuhaltende Abfolge von Job-Funktionen . Siehe auch Abschnitt "Einzuhaltende Struktur bei der Job-Definition", Seite 25.
Job-ID	Wird von slsc_list_begin , slsc_list_begin_absolute und slsc_list_begin_relative zurückgegeben.
Job-Queue	Bezeichnet eine Abfolge (unbegrenzte Anzahl) von Jobs . Eine syncAXIS control-Instanz verwaltet genau eine Job-Queue . Sie wird bei der Initialisierung angelegt und ist zunächst leer. Siehe auch Abbildung 13, Seite 44.
Job-Funktion (slsc_list_*)	Bezeichnet eine Funktion in der API mit dem Präfix slsc_list_ . Diese Funktionen dienen zum Definieren von Jobs (z. B. für Markierungen, Sprünge, sowie zum Schalten von Signalen an Ausgabe-Ports). Siehe Kapitel 3.1.2 "Job-Funktionen (slsc_list_*)", Seite 85.
Job-Status	Siehe Abbildung 12, Seite 43.
Konfigurations-Funktion (slsc_cfg_*)	Bezeichnet eine Funktion in der API mit dem Präfix slsc_cfg_ . Diese Funktionen dienen z. B. zur Konfigurationsteuerung der syncAXIS control-Instanz und um Event Callbacks zu konfigurieren. Siehe Kapitel 3.1.1 "Konfigurations-Funktionen (slsc_cfg_*)", Seite 76.
Kontroll-Funktion (slsc_ctrl_*)	Bezeichnet eine Funktion in der API mit dem Präfix slsc_ctrl_ . Diese Funktionen dienen zur Steuerung des Anwenderprogrammablaufs, z. B. Ausführung starten/stoppen, Fehler abfragen, Sprunggeschwindigkeit und Markiergeschwindigkeit setzen etc. Siehe Kapitel 3.1.3 "Kontroll-Funktionen (slsc_ctrl_*)", Seite 96.
"Laser active"-Betrieb	Siehe Abbildung 42 sowie RTC6-Handbuch, Kapitel 7.4 "Laser-Steuerung", Seite 183, insbesondere Signale für den "Laser active"-Betrieb , Seite 185. Im Endeffekt emittiert der Laser.
"Laser standby"-Betrieb	Siehe Abbildung 42 sowie RTC6-Handbuch, Kapitel 7.4 "Laser-Steuerung", Seite 183, insbesondere Signale für den "Laser standby"-Betrieb , Seite 185. Im Endeffekt emittiert der Laser <i>nicht</i> .
Liste	Dieser Begriff ist reserviert für einen direkten Kontext mit RTC6-Karten. Daher wird er in diesem Handbuch nicht im <i>direkten Zusammenhang mit der syncAXIS-DLL</i> verwendet. Vgl. das Präfix slsc_list_ von Job-Funktionen .
LSB	Least Significant Bit. Das niedrigstwertige Bit.
Markier-Funktion	syncAXIS-DLL-Funktion die dazu dient, um eine Markierbewegung bei eingeschaltetem Laser zu veranlassen. Beispiele: [*]mark[*] , [*]arc[*] , [*]ellipse[*] .

Modul	Ein Job der im Simulationsmodus mit slsc_list_begin_module aufgezeichnet wurde, siehe Kapitel 2.11 "Über das Arbeiten mit "Modulen"" , Seite 65.
MSB	Most Significant Bit. Das höchstwertige Bit.
Multi-Head	Bezeichnet ein XL SCAN-System, bei dem 1 syncAXIS control-Instanz mehr als einen excelliSCAN-Scan-Kopf und 1 Verfahrtsch ansteuert.
Multi-Instanz	Bezeichnet die (optionale) Fähigkeit, mehr als eine syncAXIS control-Instanz auf einem PC gleichzeitig ausführen zu können. Erfordert einen Dongle , der diese Option explizit unterstützt ^(a) . Multi-Instanz und Multi-Head sind grundsätzlich ^(b) kompatibel.
Multi-Stage	Bezeichnet die (optionale) Funktionalität der syncAXIS control-Instanz , den Verfahrtsch wechseln zu können. Erfordert einen Dongle , der diese Option explizit unterstützt.
Puffer	Bezeichnet die syncAXIS-DLL-internen Zwischenspeicher der Verarbeitungskette. Siehe Kapitel 2.7 "Über Vorgänge bei der Ausführung des Anwenderprogramms" , Seite 41.
Pufferunterlauf	"Buffer underrun". Nach dem Ausführungs-Start hat die RTC6-Karte alle RTC6-Mikrovektorbefehle in ihrem Listenspeicher abgearbeitet, weil die syncAXIS-DLL in der Berechnung und Übertragung von weiteren RTC6-Mikrovektorbefehlen zu langsam war ("Berechnung langsamer als Ausführung"). Siehe auch Abbildung 11 , Seite 41. Ein Pufferunterlauf kommt beispielsweise vor, wenn der Eingangspuffer relativ zur Ausführungsdauer mit vielen Daten gefüllt wird (sprich kurze Vektorabschnitte und hohe Markiergeschwindigkeit). Neue Daten werden nicht rechtzeitig in den Ausgangspuffer geschrieben. In der Folge kann die RTC6-Karte ihre Liste nicht weiter ausführen und sendet fort-dauernd eine statische Position ("harter Stop"). Dadurch kommt es zur Überschreitung einer System-Dynamikgrenze. Ein Pufferunterlauf versetzt XL SCAN in einen Fehlerzustand. Im Regelfall wird außerdem ein Fehler an den ACS -Achsen detektiert, da es zu einer plötzlichen (Beschleunigungs- und Ruck-unbeschränkten) Geschwindigkeitsänderung kam (die zuletzt angefahrte Position wird ausgegeben). Mögliche Maßnahmen zur Vermeidung eines Pufferunterlaufs siehe Kapitel 2.7.1 "Über die Puffer der syncAXIS control-Instanzen" , Seite 42.
Rampe	Bezeichnet eine Variation der Ausgabewerte an "ActiveChannels" entlang von Kurven (Kurve = Markiermuster bestehend aus geraden und/oder gekrümmten Anteilen – vgl. Trajektorie). Siehe Kapitel 2.9.4 "Über Rampen" , Seite 53.
Segment	Abschnitte der Trajektorie , die der Benutzer unter Angabe von Ziel-Koordinaten mit den folgenden Job-Funktionen (slsc_list_*) definiert hat: <ul style="list-style-type: none"> • Siehe Definieren von Sprüngen zu Ziel-Koordinaten, Seite 86 • Siehe Definieren von Markierungen zu Ziel-Koordinaten, Seite 86 Entsprechend gibt es den Segment -Typ: <ul style="list-style-type: none"> • Sprung-Segment • Markier-Segment

Simulationseinstellung	Zustand der syncAXIS control-Instanz im Bezug auf Simulationsmodus/Hardwaremodus, siehe slsc_SimulationSetting . Siehe auch Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31 .
Sprung-Funktion	syncAXIS-DLL-Funktion die dazu dient, um die Achsen des Scan-Systems bei <i>ausgeschaltetem</i> Laser zu einer neuen Position zu bewegen. Beispiel: [*]jump[*] .
Subcycle Switching	Leistungsmerkmal von aktuellen RTC6-Software-Paketen, das auch durch die [*]dashed[*]-Funktionen genutzt wird. Damit kann die RTC6-Karte bei jedem Mikroschritt (10 μ s; siehe RTC6-Handbuch, Kapitel 7.1.2 "Zerlegung in Mikroschritte", Seite 139) bis zu 20 \times den Laser schalten.
syncAXIS control-Instanz	Softwareobjekt, das beim Aufruf einer gültigen syncAXISConfig.xml durch ein syncAXIS control-basiertes Anwenderprogramms im PC-RAM angelegt wird. Eine syncAXIS control-Instanz ist entweder für den Hardwaremodus oder den Simulationsmodus konfiguriert, siehe Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen", Seite 26 . Jede syncAXIS control-Instanz kann durch ein eindeutiges Handle angesprochen werden.
syncAXISConfig.xml	XML-Konfigurationsdatei. Obwohl der Dateiname frei gewählt werden kann, wird sie in diesem Dokument durchgehend als " syncAXISConfig.xml " bezeichnet. Die vollständige Beschreibung aller Tags finden Sie in Kapitel 13 "Anhang F: Referenz der syncAXISConfig.xml-Tags", Seite 358 .
syncAXISSysConfig.xml	Entfällt für syncAXIS control-Software-Paket \geq V1.2: XML-System-Konfigurationsdatei.
Trajektorie	Kurve mit Zeitparameterisierung.
Trajektorienplanung	Siehe Abbildung 11, Seite 41 .
Utility-Funktion (slsc_util_*)	Bezeichnet eine Funktion in der API mit dem Präfix slsc_util_ . Diese Funktionen dienen zu Sonderzwecken außerhalb des regulären syncAXIS control-Betriebs. Siehe Kapitel 3.1.4 "Utility-Funktionen (slsc_util_*)", Seite 101 .

- (a) Die Anzahl der zulässigen **syncAXIS control-Instanzen** ist auf dem **Dongle** codiert.
- (b) Nicht kompatibel mit bestimmten Sonder-Systemen, bei denen 2 **syncAXIS control-Instanzen** auf einem PC 2 Master/Slave-verbundene RTC6-Karten ansteuern, die in das gleiche EtherCAT-Netzwerk einspeisen.

2 Softwareentwicklung mit der syncAXIS-DLL

2.1 Sicherheit

Berücksichtigen Sie zur Entwicklung von syncAXIS control-basierten Anwenderprogrammen das Sicherheitskonzept Ihrer Anlagen-Steuerung.

Achten Sie insbesondere darauf, dass weder der Laser unvorhergesehen eingeschaltet noch der Verfahrtsch unvorhergesehen bewegt wird.



Warnung!

Verletzungsgefahr durch Laserstrahlung! Lasersicherheitsbestimmungen einhalten!



Warnung!

Verletzungsgefahr durch Laserstrahlung! **slsc_cfg_initialize_from_file** und **slsc_util_reset_pcie** können zu undefinierten Zuständen der RTC6-Karte(n) führen, bei denen unvorhergesehen der Laser eingeschaltet werden könnte! Stellen Sie vor dem Aufruf dieser Funktionen sicher, dass der Laser ausgeschaltet ist!



Vorsicht!

Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! Berücksichtigen Sie im Sicherheitskonzept Ihrer Anlagen-Steuerung, dass die Lasersteuersignale der RTC durch **slsc_cfg_initialize_from_file** und **slsc_ctrl_enable_laser** freigegeben werden.



Vorsicht!

Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! Berücksichtigen Sie im Sicherheitskonzept Ihrer Anlagen-Steuerung, dass bei der **Job**-Ausführung der Laser angeschaltet wird.



Warnung!

Verletzungsgefahr durch Verfahrtschbewegung! Keine Personen im Gefahrenbereich!



Vorsicht!

Gefahr der Sachbeschädigung durch Verfahrtschbewegung! Keine Fremdgegenstände im Gefahrenbereich!



Vorsicht!

Von einem bewegten Verfahrtsch geht eine mechanische Gefahr aus. Es besteht Quetschgefahr für die Finger und Hände. Berücksichtigen Sie im Sicherheitskonzept Ihrer Anlagen-Steuerung, dass **slsc_ctrl_start_execution** u.U. den Verfahrtsch (eventuell auch verzögert) in Bewegung setzen kann. Achten Sie darauf, dass alle umstehenden Personen während der Ausführung ausreichend Abstand zu dem Gerät halten.



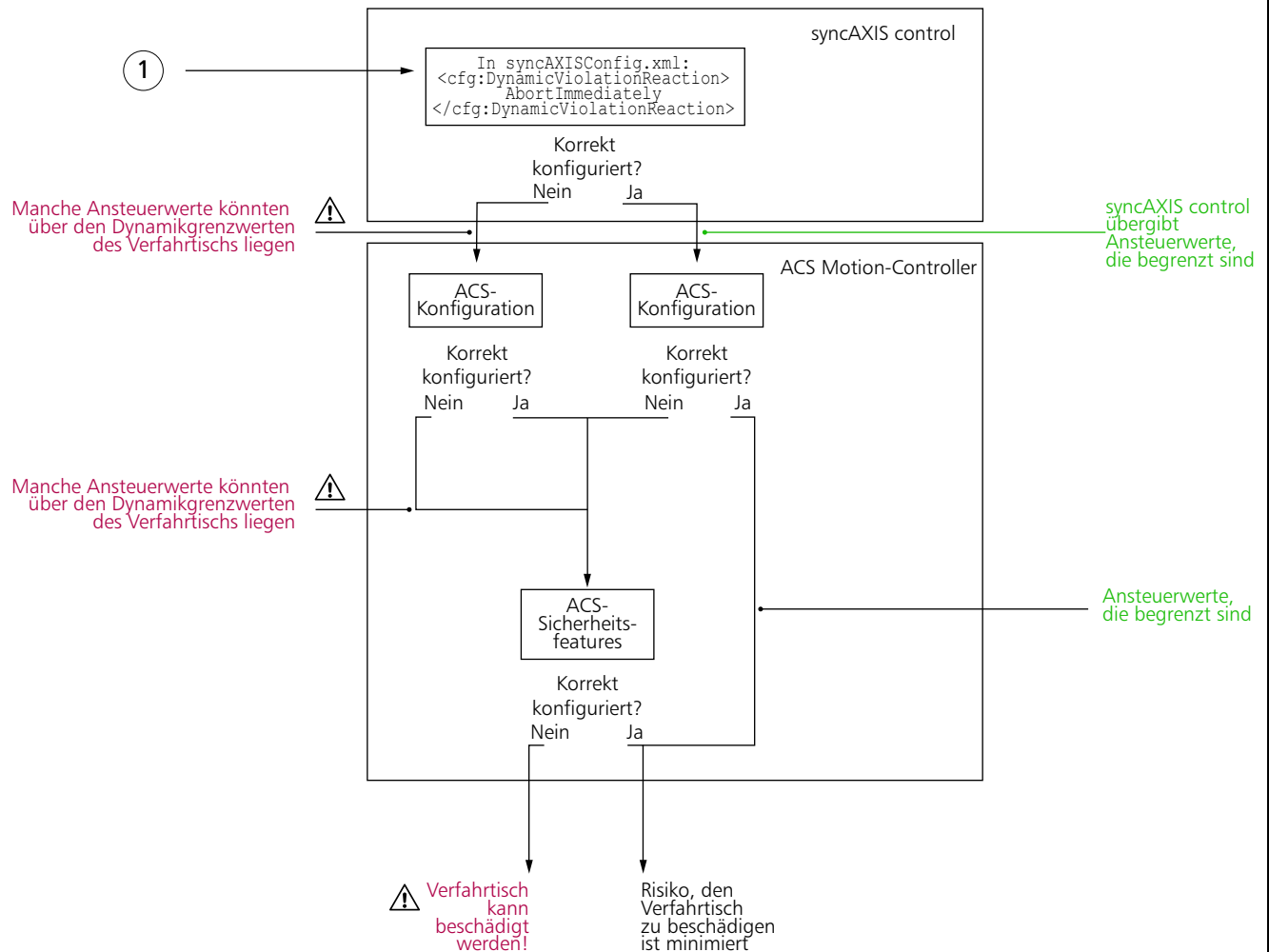
Warnung!

Die Code-Abschnitte dieses Handbuchs dürfen niemals ohne vorhergehende Anpassung und Simulation auf echten XL SCAN-Systemen ausgeführt werden. Anderenfalls drohen Personenschäden und Sachschäden.

Haftungsausschluss: SCANLAB übernimmt keine Haftung für Schäden oder Folgeschäden, die sich aus der Nichtbeachtung dieser Warnung ergeben. SCANLAB übernimmt keine Verantwortung für die Richtigkeit oder Funktionalität dieser Code-Abschnitte.

Hinweise

- Für einen schnellen Einstieg enthält das syncAXIS control-Software-Paket das "Installation_Project" (Quellcode in C++). Siehe auch **Handbuch "Installation der SCANLAB XL SCAN-Komponenten und Erstinbetriebnahme des XL SCAN-Systems"**.



Legende

- Benutzereingaben in `syncAXISConfig.xml`. syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als:
 - Scan-Device-Dynamikgrenzen die `DynamicLimits`-Werte, [Seite 388](#)
 - Scan-Device-Arbeitsfeldgrenzen die `FieldLimits`-Werte, [Seite 395](#)
 - Scan-Device-Überwachungskriterium den `MonitoringLevel`-Wert, [Seite 397](#)
 - Verfahrenstisch-Arbeitsfeldgrenzen die `FieldLimits`-Werte, [Seite 454](#)
 - Verfahrenstisch-Dynamikgrenzen die `DynamicLimits`-Werte, [Seite 456](#)
 - Verfahrenstisch-Überwachungskriterium den `MonitoringLevel`-Wert, [Seite 452](#)
 - Reaktion auf Überschreitungen den `DynamicViolationReaction`-Wert, [Seite 365](#)

Stellen Sie die Voraussetzungen sicher, dass syncAXIS control \geq V1.2 begrenzte Ansteuerwerte an den ACS Motion-Controller übergeben kann!

2.2 Über die SICHERE Verwendung von syncAXIS control – Allgemeines Vorgehen

Die syncAXIS control-Software kann mit Konfigurationsparametern flexibel auf unterschiedliche Laser-Scan-System-Hardware-Komponenten angepasst werden. syncAXIS control-basierte Anwenderprogramme können zusätzlich mit weiteren Parametern auf Durchsatz und Genauigkeit optimiert werden.

Diese Offenheit gegenüber Anpassbarkeit und Optimierbarkeit hat allerdings umgekehrt zur Konsequenz, dass Benutzer syncAXIS control sehr sorgfältig konfigurieren müssen. Nur dann kann der eingebaute Sicherheitsmechanismus wirksam werden, bei dem die an den **ACS Motion-Controller** übergebenen Ansteuerwerte so begrenzt sind, dass weitere nachfolgende Mechanismen Schäden verhindern können, siehe **Abbildung 2, Seite 17** und **Abschnitt "syncAXIS control Sicherheitsfeatures", Seite 18**.

Schäden durch eine solche fehlerhafte Ansteuerung können u.a. sein:

- Personenschäden (z.B. durch Laser-Strahlung oder durch Verfahrtschbewegung)
- Schäden am Laser-Scan-System (z.B. durch Vignettierung am Scan-Kopf oder Objektiv oder durch ungebremstes Fahren des Verfahrtschis an den Anschlag)
- Erhöhte Produktionskosten (z.B. durch Unterbrechung des Produktionsprozesses oder durch unbrauchbare Werkstücke wegen fehlender oder abgeschnittener Markierungen)

Um solche Schäden zu vermeiden, führen Sie daher unbedingt die folgenden Schritte durch, bevor Sie erste Anwenderprogramme an Ihrem Laser-Scan-System ausführen:

- **Systemgrenzen ermitteln, Seite 19**
- **Sicherheitsmechanismen einrichten, Seite 19**
- **Sichere syncAXIS control Instanzen konfigurieren, Seite 20**
- **Jobs simulieren und nachbessern, Seite 24**

syncAXIS control Sicherheitsfeatures

Ab syncAXIS control \geq V1.2 steht als wesentliches Sicherheitsfeature der **syncAXISConfig.xml**-Tag **DynamicViolationReaction** zur Verfügung⁽¹⁾.

Der **DynamicViolationReaction**-Wert legt die Reaktion auf Grenzwertüberschreitungen fest.

Diese Grenzwertüberschreitungen können sowohl

- Arbeitsfeldverletzungen als auch
- Dynamikverletzungen von
- Scan-Devices und
- Verfahrtschis

einschließen.

Welche Grenzwertüberschreitungen konkret überwacht werden sollen ("Überwachungskriterien"), wird in der **syncAXISConfig.xml** für Scan-Devices und Verfahrtschis getrennt eingestellt unter:

- **<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:MonitoringLevel>**
- **<cfg:Configuration> → <cfg:StageConfig> → <cfg:MonitoringLevel>**

Die genaue Reaktion auf solche Grenzwertüberschreitungen wird durch den Wert von **DynamicViolationReaction** bestimmt. Mögliche Reaktionen sind das Erzeugen von **[WARN]-Log-Dateizeilen** (**WarningOnly**), ein sofortiger Abbruch der Bewegung ("Not-Halt"; **AbortImmediately**) oder der Versuch einer kontrollierten Abbremsbewegung (**StopAndReport**).

(1) Anmerkung zu syncAXIS control \leq V1.1:
Diese Versionen erzeugen lediglich **[WARN]-Log-Dateizeilen**. Der Bewegungsabbruch erfolgt abrupt, sofern der **ACS Motion-Controller** tatsächlich der Trajektorie aufgrund der Grenzwertüberschreitung nicht folgen konnte. Dann werden beide Systeme, syncAXIS control und **ACS Motion-Controller**, in einen Fehlerzustand versetzt.

2.2.1 Systemgrenzen ermitteln

Ermitteln Sie – z.B. durch Lesen der entsprechenden Handbücher oder durch Rückfrage bei den jeweiligen Herstellern – die realen Grenzen Ihres XL SCAN-Systems:

- nutzbare Verfahrtisch-Arbeitsbereich(e)
- die max. Verfahrtischgeschwindigkeit
- die max. Verfahrtischbeschleunigung
- den max. Verfahrtischruck
- das max. nutzbare Scan-Kopf-Arbeitsfeld (d.h. den max. Auslenkwinkel und das max. Bildfeld, abhängig vom verwendeten Objektiv)
- die max. Galvanometerscannergeschwindigkeit⁽¹⁾
- die max. Galvanometerscannerbeschleunigung⁽¹⁾
- den max. Galvanometerscannerruck⁽¹⁾
- die max. erlaubte Laser-Leistung (und Laserleistungsdichte)

Die Verfahrtisch-Grenzen hängen sowohl vom verwendeten Motor-Modell als auch von den verwendeten Achs-Reglern ab.

2.2.2 Sicherheitsmechanismen einrichten

Richten Sie Mechanismen ein, die Personenschäden und Sachschäden verhindern:

- Sorgen Sie durch angemessene Sicherheitseinrichtungen, Sicherheitswarnhinweise und Schutzausrüstung für ausreichenden Schutz gegen die Laserstrahlung des verwendeten Bearbeitungslasers.
- Sorgen Sie durch angemessene Sicherheitseinrichtungen für ausreichenden Schutz gegen Personenschäden durch Verfahrtischbewegungen.
- Richten Sie bei Bedarf – ggf. mit Unterstützung der Hersteller des Verfahrtischs und der zugehörigen Achs-Regler – Mechanismen ein, die dafür sorgen, dass die Grenzen des Verfahrtischs sicher eingehalten werden (z.B. durch automatisches Abbremsen oder Abschalten des Verfahrtischs). Beugen Sie auf diese Weise einer Beschädigung des Verfahrtischs und entsprechenden Folgeschäden vor.
- Richten Sie bei Bedarf Mechanismen ein, die dafür sorgen, dass die Laser-Leistung nicht den erlaubten Bereich überschreitet.

(1) Dieser Grenzwert ist typischerweise von SCANLAB in der `syncAXISConfig.xml` voreingestellt und muss dort i.d.R. nicht geändert werden.

2.2.3 Sichere syncAXIS control Instanzen konfigurieren

Hinweise

- Für syncAXIS control \geq V1.2 entfällt die XML-System-Konfigurationsdatei `syncAXISSysConfig.xml`.
- Alle Details zur `syncAXISConfig.xml` finden Sie in Kapitel 13 "Anhang F: Referenz der syncAXIS-Config.xml-Tags", Seite 358.

In der `syncAXISConfig.xml` müssen die korrekten Werte für Planungs-Parameter-Werte und Überwachungs-Grenzwerte für die vorgesehenen Scan-Devices und Verfahrtsche eingetragen sein.

- (1) Planungs-Parameter für die Scan-Devices, Seite 20
- (2) Überwachungs-Grenzwerte für die Scan-Devices, Seite 20
 - a. Scan-Device-Arbeitsfelder, Seite 20
 - b. Scan-Device-Dynamikgrenzen, Seite 21
- (3) Planungs-Parameter für die Verfahrtsche, Seite 21
- (4) Überwachungs-Grenzwerte für die Verfahrtsche, Seite 21
 - a. Verfahrtsch-Arbeitsfelder, Seite 21
 - b. Verfahrtsch-Dynamikgrenzen, Seite 22

(1) Planungs-Parameter für die Scan-Devices

⚠ Vorsicht! syncAXIS control verwendet diese Werte zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte für Beschleunigung und Ruck korrekt sind. Siehe Seite 390 zu Geschwindigkeits-Werten.

```
<cfg:Configuration>
  <cfg:ScanDeviceConfig>
    <cfg:CalculationDynamics>
      <cfg:MarkDynamics>
        <cfg:Acceleration>
        <cfg:Jerk>
      <cfg:JumpDynamics>
        <cfg:Acceleration>
        <cfg:Jerk>
```

(2) Überwachungs-Grenzwerte für die Scan-Devices

a. Scan-Device-Arbeitsfelder

```
<cfg:Configuration>
  <cfg:ScanDeviceConfig>
    <cfg:FieldLimits>
      <cfg:XDirection ...>(1)
      <cfg:YDirection ...>(1)
```

Diese Werte werden nur zum Überwachen verwendet. Ist als Scan-Device-MonitoringLevel das Überwachungskriterium Position, Velocity, Acceleration oder Jerk eingetragen (Jerk subsumiert auch Acceleration, Acceleration subsumiert auch Velocity etc.),

```
<cfg:Configuration>
  <cfg:ScanDeviceConfig>
    <cfg:MonitoringLevel>
      Position ODER Velocity ODER
      Acceleration ODER Jerk
    </cfg:MonitoringLevel>
```

dann lösen Überschreitungen automatisch die in DynamicViolationReaction festgelegte Reaktion (WarningOnly ODER AbortImmediately ODER StopAndReport) aus.

(1) Der Tag hat einen Unit -Attributwert, d.h. die Einheit ist einstellbar.

b. Scan-Device-Dynamikgrenzen

```
<cfg:Configuration>
  <cfg:ScanDeviceConfig>
    <cfg:DynamicLimits>
      <cfg:Velocity ...>(1)
      <cfg:Acceleration ...>(1)
      <cfg:Jerk ...>(1)
```

Diese Werte werden nur zum Überwachen verwendet. Ist als Scan-Device-MonitoringLevel das Überwachungskriterium **Position**, **Velocity**, **Acceleration** oder **Jerk** eingetragen (**Jerk** subsumiert auch **Acceleration**, **Acceleration** subsumiert auch **Velocity** etc.),

```
<cfg:Configuration>
  <cfg:ScanDeviceConfig>
    <cfg:MonitoringLevel>
      Position ODER Velocity ODER
      Acceleration ODER Jerk
    </cfg:MonitoringLevel>
```

dann lösen Überschreitungen automatisch die in **DynamicViolationReaction** festgelegte Reaktion (**WarningOnly** ODER **AbortImmediately** ODER **StopAndReport**) aus.

(3) Planungs-Parameter für die Verfahrtische

```
<cfg:Configuration>
  <cfg:StageConfig>
    <cfg:StageList>
      <cfg:Stage>
        <cfg:CalculationDynamics>
          <cfg:Velocity ...>(1)
          <cfg:Acceleration ...>(1)
          <cfg:Jerk ...>(1)
```

⚠ Vorsicht! syncAXIS control verwendet die Werte bei **Velocity**, **Acceleration** und **Jerk** zur Planung von Trajektorien für den **Betriebsmodus** "StageOnly" sowie für die Endbewegung an **Job**-Enden. Stellen Sie sicher, dass die eingetragenen Werte korrekt sind.

(4) Überwachungs-Grenzwerte für die Verfahrtische

a. Verfahrtisch-Arbeitsfelder

```
<cfg:Configuration>
  <cfg:StageConfig>
    <cfg:StageList>
      <cfg:Stage>
        <cfg:FieldLimits>
          <cfg:XDirection ...>(1)
          <cfg:YDirection ...>(1)
```

Diese Werte werden nur zum Überwachen verwendet. Ist als Verfahrtisch-MonitoringLevel das Überwachungskriterium **Position** eingetragen,

```
<cfg:Configuration>
  <cfg:StageConfig>
    <cfg:MonitoringLevel>
      Position
    </cfg:MonitoringLevel>
```

dann lösen Überschreitungen automatisch die in **DynamicViolationReaction** festgelegte Reaktion (**WarningOnly** ODER **AbortImmediately** ODER **StopAndReport**) aus.

b. Verfahrtsch-Dynamikgrenzen

```
<cfg:Configuration>
  <cfg:StageConfig>
    <cfg:StageList>
      <cfg:Stage>
        <cfg:DynamicLimits>
          <cfg:Velocity>
          <cfg:Acceleration>
          <cfg:Jerk>
```

Diese Werte werden nur zum Überwachen verwendet. Ist als Verfahrtsch-MonitoringLevel das Überwachungskriterium *Velocity*, *Acceleration* oder *Jerk* eingetragen (*Acceleration* subsumiert auch *Velocity*, *Jerk* subsumiert auch *Acceleration*),

```
<cfg:Configuration>
  <cfg:StageConfig>
    <cfg:MonitoringLevel>
      Velocity ODER Acceleration ODER Jerk
    </cfg:MonitoringLevel>
```

dann lösen Überschreitungen automatisch die in *DynamicViolationReaction* festgelegte Reaktion (*WarningOnly* ODER *AbortImmediately* ODER *StopAndReport*) aus.

Stellen Sie – im Sinne einer hohen System-Performance – sicher, dass die eingetragenen Grenzwerte unter (1) *Planungs-Parameter für die Scan-Devices*, Seite 20 und (3) *Planungs-Parameter für die Verfahrtsche*, Seite 21 mit den realen Systemgrenzen übereinstimmen.

Stellen Sie – im Sinne einer hohen System-Sicherheit – sicher, dass unter (2) *Überwachungs-Grenzwerte für die Scan-Devices*, Seite 20 und (4) *Überwachungs-Grenzwerte für die Verfahrtsche*, Seite 21 keine Grenzwerte eingetragen sind, die außerhalb der realen Systemgrenzen liegen.

Stellen Sie außerdem sicher, dass syncAXIS control-basierte Anwenderprogramme bei nachfolgenden Simulationen und realen Markierungen nur *syncAXISConfig.xml* verwenden, in denen auch korrekte Grenzwerte eingetragen sind (eine *syncAXIS control-Instanz* wird mit derjenigen *syncAXISConfig.xml* initialisiert, die bei *slsc_cfg_initialize_from_file* angegeben ist).

Hinweise

- Prüfen Sie auch bei jeder nachträglichen Hardware-Änderung am verwendeten XL SCAN-System (z. B. bei einem Wechsel zu einem Objektiv mit anderer Brennweite oder zu einem anderen Verfahrtisch) unbedingt auch die Einstellungen in allen zugehörigen XML-Konfigurationsdateien. Passen Sie die eingetragenen Grenzwerte bei Bedarf entsprechend der dann gültigen Systemgrenzen an.
- syncAXIS control verwendet die in `syncAXISConfig.xml` eingetragenen Werte für
 - (1) Planungs-Parameter für die Scan-Devices, Seite 20 und
 - (3) Planungs-Parameter für die Verfahrtische, Seite 21
 u.a. bei der Trajektorienberechnung von Ansteuerwerten für Scan-Kopf und Verfahrtisch⁽¹⁾ – allerdings nur im Sinne einer Berechnung von möglichst effizienten Trajektorien.

Wichtig: syncAXIS control führt kein automatisches Clipping der Ansteuerwerte und Prozessgeschwindigkeiten auf die Systemgrenzen durch.

Insbesondere bei den schnellen Sprung-Bewegungen, die Scan-Kopf und Verfahrtisch zwischen den eigentlichen Markierungen ausführen, könnte es zu Ansteuerwerten und Prozessgeschwindigkeiten außerhalb der Systemgrenzen kommen.

Die von syncAXIS control berechneten Ansteuerwerte und Prozessgeschwindigkeiten müssen daher unbedingt durch eine Simulation auf Einhaltung der Systemgrenzen geprüft werden und die jeweiligen **Jobs** entsprechend nachgebesert werden, bevor reale Markierungen mit Laser, Scan-Kopf und Verfahrtisch ausgeführt werden, siehe Kapitel 2.2.4 "Jobs simulieren und nachbesern", Seite 24.

(1) ...und am Ende des **Jobs** zu einer Verfahrtisch-Ausgleichsbewegung. Außerdem schreibt syncAXIS control Warnungen in die Log-Datei, wenn eine der Systemgrenzen verletzt wird (sofern in `syncAXISConfig.xml` konfiguriert, siehe "Über das Logging in syncAXIS control", Seite 47).

- syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als:
 - Scan-Device-Dynamikgrenzen die `DynamicLimits`-Werte, Seite 388
 - Scan-Device-Arbeitsfeldgrenzen die `FieldLimits`-Werte, Seite 395
 - Scan-Device-Überwachungskriterium den `MonitoringLevel`-Wert, Seite 397
 - Verfahrtisch-Arbeitsfeldgrenzen die `FieldLimits`-Werte, Seite 454
 - Verfahrtisch-Dynamikgrenzen die `DynamicLimits`-Werte, Seite 456
 - Verfahrtisch-Überwachungskriterium den `MonitoringLevel`-Wert, Seite 452
 - Reaktion auf Überschreitungen den `DynamicViolationReaction`-Wert, Seite 365

Achtung!

Für Multi-Kopf-Systeme gilt:

- syncAXIS control überprüft nicht auf Arbeitsfeldüberlappungen der Scan-Devices. Stellen Sie sicher, dass jedes Scan-Device nur sein "eigenes" Werkstück bearbeiten kann.
- syncAXIS control kann nicht überprüfen, ob Scan-Kopf-Arbeitsfeldgrenzen aufgrund von `slsc_cfg_set_part_displacement`-Transformationen verletzt werden. Ein kontrolliertes Abbremsen als `DynamicViolationReaction` ist nicht möglich.



Vorsicht!

Trotz der in den `syncAXISConfig.xml` eingetragenen Grenzwerten berechnet syncAXIS control u.U. Ansteuerwerte und Prozessgeschwindigkeiten *außerhalb* der Systemgrenzen.

Führen Sie deshalb syncAXIS control-basierte Anwenderprogramme immer erst im Simulationsmodus aus (um eine Prüfung auf Überschreitung der Systemgrenzen durchzuführen), bevor Sie mit ihm erstmalig ein reale Markierung mit Laser, Scan-Kopf und Verfahrtisch ausführen.

2.2.4 Jobs simulieren und nachbessern

Bevor Sie erstmalig mit einem syncAXIS control-basierten Anwenderprogramm reale Markierungen mit Laser, Scan-Kopf und Verfahrtsch ausführen, gehen Sie wie folgt vor:

- (1) Führen Sie das Anwenderprogramm im Simulationsmodus aus und lassen Sie hierbei die von syncAXIS control berechneten Ansteuerwerte aufzeichnen⁽¹⁾, siehe [Kapitel 2.5 "Über den syncAXIS control Simulationsmodus"](#), Seite 31. Siehe auch den Hinweis rechts, Seite 24.
- (2) Analysieren Sie die aufgezeichneten Ansteuerwerte – z.B. durch grafische Darstellung⁽²⁾ – auf Überschreitung der Systemgrenzen. Achten Sie hierbei u.a. auf Folgendes:
 - Sind Ansteuerwerte an den Verfahrtsch vorgegeben, die nicht von diesem angefahren werden können, die den Verfahrtsch an den Anschlag fahren lassen würden oder eine Notabschaltung des Verfahrtschs auslösen könnten?
 - Sind Ansteuerwerte an den Scan-Kopf vorgegeben, bei denen Vignettierung am Scan-Kopf oder am Scan-Objektiv auftreten könnten?
 - Sind Geschwindigkeiten vorgegeben, bei denen der Verfahrtsch die tatsächlich realisierbaren Geschwindigkeiten, Beschleunigungen oder Rucke überschreiten müsste?
 - Wahlweise können Sie hierbei auch gleich eine Analyse zu Optimierungszwecken durchführen:
 - Wie groß ist das vom Scan-Kopf tatsächlich genutzte Arbeitsfeld?
 - Wie lang ist die Gesamtprozesszeit des syncAXIS control-basierten Anwenderprogramms?
- (3) Bessern Sie das syncAXIS control-basierte Anwenderprogramm so lange nach, bis Sie in nachfolgenden Simulations- und Analyseschritten keine Grenzüberschreitung mehr finden.

Hinweise

- Für syncAXIS control \geq V1.2.4 gilt: Die Extrema des aktuellen **Jobs** können mit **slsc_ctrl_get_job_characteristic** abgefragt werden. Die Dateiausgabe⁽¹⁾ muss dazu *nicht* aktiviert sein.
- Für syncAXIS control \geq V1.6.0 siehe auch [Abschnitt "Funktionen zum Ändern von Trajektorienplanungs-Werten"](#), Seite 94.
- Für syncAXIS control \geq V1.7.0 gilt: Bei **Job**-Voranalysen zur Optimierung kann mit **slsc_cfg_get_jump_time** die Dauer von Sprüngen berechnet werden.

(1) Siehe [DisableFileOutput](#).

(2) z.B. im syncAXIS Viewer.

2.3 Über die Hauptstrukturen eines syncAXIS-DLL-basierten Anwenderprogramms (Beispielhaft)

Im Folgenden sind beispielhaft die wesentlichen Teile eines Anwenderprogramms genannt, das die syncAXIS-DLL verwendet.

- (1) Programmteil zum Initialisieren und Konfigurieren der **syncAXIS control-Instanz**
 - **slsc_cfg_initialize_from_file** aufrufen, siehe Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen", Seite 26.
 - Prüfung, ob der **syncAXIS control-Instanz-Status** "grün" ist: **slsc_cfg_get_operation_status** aufrufen.
 - Hinweis: An dieser Stelle (wie im Installationshandbuch beschrieben) könnte auch der **Betriebsmodus** schon geändert werden (**slsc_cfg_set_mode**). Beispielsweise müssen bei der Kalibrierung des Systems u.a. **Jobs** (Markieren der Referenzgitter) in unterschiedlichen Betriebsmodi "**ScannerOnly**" und "**StageOnly**" ausgeführt werden.
- (2) Programmteil (Code), der sich um die **Job-Definitionen** und **Job-Abfolgen** "kümmert"
 - **Jobs** aus den geeigneten **Markier-Funktionen** und **Sprung-Funktionen** zusammensetzen (jeder Einzel-**Job** mit **slsc_list_begin** am Anfang und **slsc_list_end** am Ende, siehe Kapitel 2.3.1 "Einzuhaltende Struktur bei der Job-Definition", Seite 25).
 - Bedarfsweise: Funktionen einfügen (**slsc_list_write_analog_x**, **slsc_list_write_digital_out**, **slsc_list_write_digital_out_mask**), die bei der Programmausführung Signale (entsprechend den RTC-Befehlen **list_write_da***, **list_write_io***) an Ausgabe-Ports setzen
 - Alle Funktionsparameter mit den gewünschten Werten versorgen (z. B. Koordinaten)
 - Hinweis: Die Reihenfolge der **Jobs** im Quellcode entspricht der späteren Reihenfolge der Ausführung (Prinzip "first in – first out").

- (3) Programmteil, der die Ausführung von **Jobs** startet und überwacht
 - Voraussetzung prüfen (**slsc_ctrl_get_exec_state**): Ist die RTC6-Karte bereit zur Ausführung (**slsc_ExecState_ReadyForExecution**)
 - Ausführungsstart auslösen (**slsc_ctrl_start_execution**)
 - Ausführungsstatus monitoren z. B. Events über Callbacks auswerten (z. B. ist die Ausführung beendet)
- (4) Programmteil zur Fehlerdetektion und durch den Benutzer definierte Reaktion
 - Ob Fehler aufgetreten sind abfragen (**slsc_ctrl_get_error_count**).
 - Auf Fehler reagieren (gemäß Wissen des Benutzers)
- (5) Programmteil zum Abbauen der **syncAXIS control-Instanz**
 - **slsc_cfg_delete** aufrufen

2.3.1 Einzuhaltende Struktur bei der Job-Definition

Für **Jobs** ist die vorgeschriebene Abfolge der Job-Funktionen (**slsc_list_***) ist wie folgt:

- (1) **slsc_list_begin***
- (2) 0...unbegrenzte Anzahl von Job-Funktionen **slsc_list_*** – außer **slsc_list_begin***⁽¹⁾.
- (3) **slsc_list_end**.

Zur Laufzeit werden **Jobs** durch das Anwenderprogramm Funktion-um-Funktion an die **syncAXIS control-Instanz** übertragen, siehe Abbildung 11, Seite 41.

Die syncAXIS-DLL prüft bei den eingehenden **Jobs**, ob die vorgeschriebene Abfolge der Job-Funktionen (**slsc_list_***) eingehalten wird (Konsistenzprüfung). Im Fehlerfall ist hier beim Rückgabewert **Bit #07** gesetzt (**JobStructureNotValid**).

(1) **Jobs** können also nicht verschachtelt werden.

2.4 Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen

Achtung!

Um auf einem PC mehrere **syncAXIS control-Instanzen** im Hardwaremodus gleichzeitig betreiben zu können, muss ein entsprechend konfigurierter **Dongle** verwendet werden!

Sonst ist beim Rückgabewert Bit #30 gesetzt (**MaxInstancesReached**).

Eine **syncAXIS control-Instanz** ist konfiguriert entweder für den

- Hardwaremodus (USE CASE 1 in **Abbildung 3, Seite 27**) oder den
- Simulationsmodus (USE CASE 2 in **Abbildung 3, Seite 27**).

Hinweise für den Hardwaremodus

- Jede **syncAXIS control-Instanz** erfordert 1 RTC6-Karte (im PC installiert und nicht von einem anderen Anwenderprogramm übernommen).



Warnung!

Verletzungsgefahr durch Laserstrahlung! Lasersicherheitsbestimmungen einhalten!



Warnung!

Verletzungsgefahr durch Verfahrtschbewegung! Keine Personen im Gefahrenbereich!



Vorsicht!

Gefahr der Sachbeschädigung durch Verfahrtschbewegung! Keine Fremdgegenstände im Gefahrenbereich!

Um die **syncAXIS control-Instanz** anzulegen, erfolgt aus dem Anwenderprogramm ("Executable") heraus der Aufruf von **slsc_cfg_initialize_from_file**.

In der angegebenen **syncAXISConfig.xml** muss stehen:

- Für den Hardwaremodus:

```
<cfg:SimulationMode>>false
</cfg:SimulationMode>
```
- Für den Simulationsmodus:

```
<cfg:SimulationMode>>true
</cfg:SimulationMode> .
```

Die **syncAXIS control-Instanz** wird aufgebaut und erhält durch **slsc_cfg_initialize_from_file** (auch: **slsc_cfg_initialize_copy**) einen **Handle** zugeteilt. Über den **Handle** kann sie durch die (meisten) syncAXIS-DLL-Funktionen angesprochen werden (**Handle**-Wert als Parameter). Während des Aufbaus⁽¹⁾ finden folgende Vorgänge statt:

- Scan-Kopf: wird übernommen. Die Scan-Kopf-Spiegel werden in die Nullposition gebracht.
- RTC6: wird übernommen. Die RTC6-Lasersteuerung wird aktiv (scharf) geschaltet. Die Lasersteuersignale werden an der RTC6 bereits freigegeben.
- Verfahrtsch: wird übernommen. Die Verfahrtschposition wird nicht verändert.
- Output-Signale: Es werden die Werte angewendet, die in der **syncAXISConfig.xml** unter `<cfg:IOConfig><cfg:DefaultOutputs>` eingetragen sind (= **LaserPinOut, AnalogOut1, AnalogOut2**).

Welche der verfügbaren RTC6-Karten (sofern überhaupt mehrere in einem PC eingebaut sind) verwendet wird, bestimmen in der **syncAXISConfig.xml** die entsprechenden Einträge unter `<cfg:RTCCConfig>` zur Festlegung deren Identifizierung mittels Seriennummer:

```
<cfg:BoardIdentificationMethod>BySerialNumber
</cfg:BoardIdentificationMethod>
```

sowie

```
<cfg:SerialNumber>123456
</cfg:SerialNumber>.
```

Ist nur eine einzige RTC6-Karte installiert, ist die Angabe von

```
<cfg:BoardIdentificationMethod>UseFirstFound
</cfg:BoardIdentificationMethod>
```

ausreichend.

(1) Siehe **Seite 118** zu Vorgängen beim Abbauen.







Sowohl für den Hardwaremodus als auch den Simulationsmodus gilt, dass die **syncAXIS control-Instanz** in dem **Betriebsmodus** gestartet wird, der in der **syncAXISConfig.xml** hinterlegt ist:

```
<cfg:InitialOperationMode>
ScannerOnly | StageOnly | ScannerAndStage
</cfg:InitialOperationMode> (1)
```

(1) Nach der Initialisierung kann der **Betriebsmodus** mit **slsc_cfg_set_mode** geändert werden.

Zum Reinitialisieren einer **syncAXIS control-Instanz** steht **slsc_cfg_reinitialize_from_file** zur Verfügung, siehe USE CASE 1a in **Abbildung 3, Seite 27, mitte**. Diese Funktion löscht die angegebene **syncAXIS control-Instanz** und baut sie wieder auf, ändert den **Handle-Wert** der **syncAXIS control-Instanz** jedoch nicht.

Zum Simulationsmodus finden Sie weitere Informationen in **Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31**. Im Arbeitsspeicher eines PCs kann es nur 1 einzige **syncAXIS control-Instanz** im Simulationsmodus geben, siehe auch **Abbildung 6, Seite 30, unten**.

syncAXIS control-Executable	Aufruf	Argument(e)	Resultierendes Softwareobjekt	RTC6-Status (hier: 3 RTC6 installiert)
USE CASE 1 	slsc_cfg initialize_from_file	 Simulation=FALSE BoardIdentificationMethod=BySerialNumber	syncAXIS control-Instanz - im "Hardwaremodus" - Handle-Wert = Handle#1	RTC6#1 = akquiriert d. Handle#1 RTC6#2 = nicht akquiriert RTC6#3 = nicht akquiriert
USE CASE 1a 	slsc_cfg reinitialize_from_file	 Simulation=FALSE BoardIdentificationMethod=BySerialNumber	-Handle bleibt -löscht -Re-erstellt syncAXIS control-Instanz - im "Hardwaremodus" - Handle-Wert = Handle#1	RTC6#1 = akquiriert d. Handle#1 RTC6#2 = nicht akquiriert RTC6#3 = nicht akquiriert
USE CASE 2 	slsc_cfg initialize_from_file	 Simulation=TRUE BoardIdentificationMethod=<Eintrag ignoriert>	syncAXIS control-Instanz - im "Simulationsmodus" - Handle-Wert = Handle#2	RTC6#1 = nicht akquiriert RTC6#2 = nicht akquiriert RTC6#3 = nicht akquiriert

3

Überblick zu Initialisierungen: USE CASE 1, 1a, 2. Nicht dargestellt: Zusammen mit der RTC6-Karte wird auch der Verfahrtsch übernommen (betrifft auch den **Betriebsmodus** "ScannerOnly").

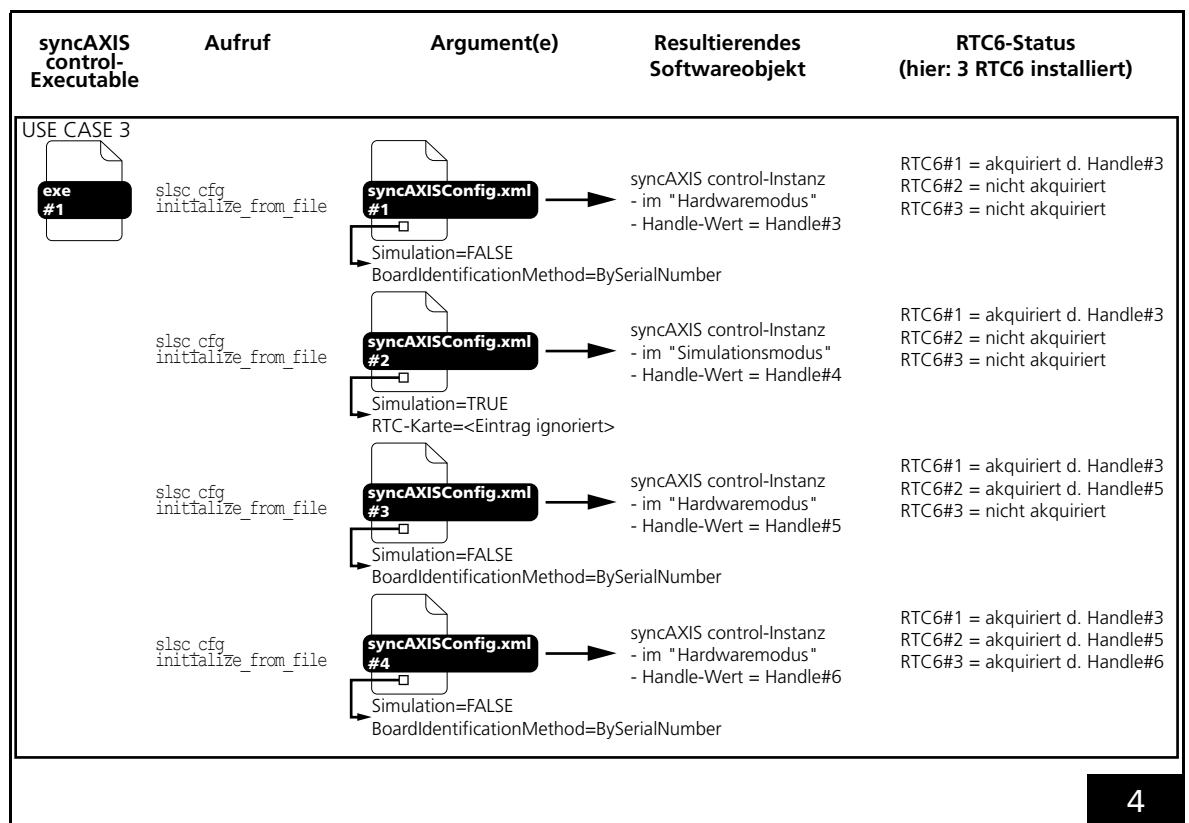
Abbildung 4, Seite 28, USE CASE 3, veranschaulicht, wie ein einziges Anwenderprogramm nacheinander 4 unterschiedliche (!) `syncAXISConfig.xml` aufruft.

Achtung!

Um auf einem PC mehrere `syncAXIS control-Instanzen` im Hardwaremodus gleichzeitig betreiben zu können, muss ein entsprechend konfigurierter `Dongle` verwendet werden!

Sonst ist beim Rückgabewert Bit #30 gesetzt (`MaxInstancesReached`).

Es ergeben sich 3 separate `syncAXIS control-Instanzen` im Hardwaremodus und 1 `syncAXIS control-Instanz` im Simulationsmodus. Anschließend sind alle 3 verfügbare RTC6-Karten übernommen.



Überblick zu Initialisierungen: USE CASE 3. Nicht dargestellt: Zusammen mit der RTC6-Karte wird auch der Verfahrtschirm übernommen (betrifft auch den `Betriebsmodus` "ScannerOnly").

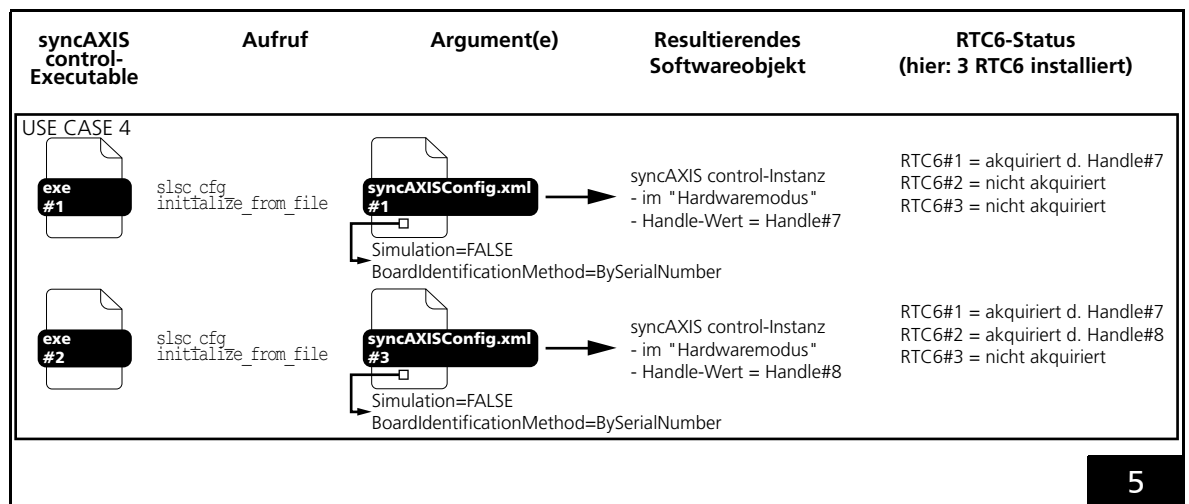
Abbildung 5, Seite 29, USE CASE 4, veranschaulicht, wie zwei unterschiedliche Anwenderprogramme 2 unterschiedliche (!) `syncAXISConfig.xml` aufrufen.

Achtung!

Um auf einem PC mehrere `syncAXIS control-Instanzen` gleichzeitig betreiben zu können, muss ein entsprechend konfigurierter `Dongle` verwendet werden!

Sonst ist beim Rückgabewert Bit #30 gesetzt (`MaxInstancesReached`).

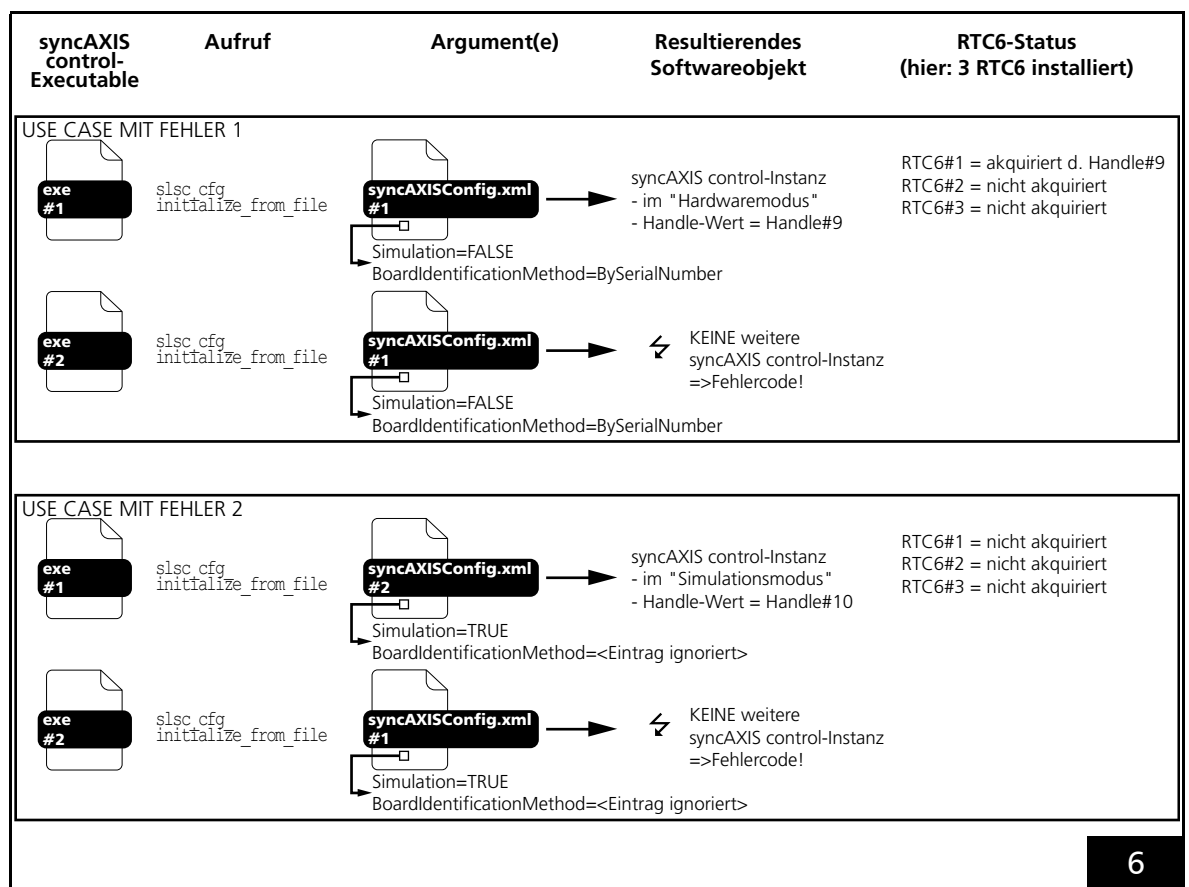
Es ergeben sich 2 separate `syncAXIS control-Instanzen` im Hardwaremodus. Anschließend sind 2 (der 3) verfügbaren RTC6-Karten übernommen.



Überblick zu Initialisierungen: USE CASE 4. Nicht dargestellt: Zusammen mit der RTC6-Karte wird auch der Verfahrtschirm übernommen (betrifft auch den `Betriebsmodus` "ScannerOnly").

Abbildung 6, Seite 30, oben,
USE CASE MIT FEHLER 1, veranschaulicht das
Ergebnis, wenn zwei unterschiedliche
Anwenderprogramme die gleiche (!)
`syncAXISConfig.xml` aufrufen. Es ergibt sich nur 1
syncAXIS control-Instanz im Hardwaremodus, eine
zweite wird nicht aufgebaut. Anschließend ist nur 1
(der 3) verfügbaren RTC6-Karten übernommen.

Abbildung 6, Seite 30, unten,
USE CASE MIT FEHLER 2, veranschaulicht, wie zwei
unterschiedliche Anwenderprogramme 2 unter-
schiedliche `syncAXISConfig.xml` aufrufen, aber beide
den gleichen Eintrag
`<cfg:SimulationMode>true</cfg:SimulationMode>`
aufweisen.
Es ergibt sich nur 1 **syncAXIS control-Instanz** im
Simulationsmodus, eine zweite wird nicht aufgebaut.
Anschließend ist keine (der 3) verfügbaren
RTC6-Karten übernommen.



Überblick zu Initialisierungen: USE CASE MIT FEHLER 1 und USE CASE MIT FEHLER 2. Nicht dargestellt:
Zusammen mit der RTC6-Karte wird auch der Verfahrtsch übernommen (betrifft auch den **Betriebsmodus**
"ScannerOnly").

2.5 Über den syncAXIS control Simulationsmodus

syncAXIS control-basierte Anwenderprogramme können mit der **syncAXIS control-Instanz** im Simulationsmodus ausgeführt werden (≥ V1.5.0: **Simulationseinstellung** abfragen mit **slsc_cfg_get_simulation_setting** und ändern mit **slsc_cfg_set_simulation_setting**).

Der Simulationsmodus:

- Erfordert den **Dongle**
- Erfordert und verwendet *keine* Hardware (wie eine RTC6 etc.)

Während der Ausführung im Simulationsmodus werden u.a. die von syncAXIS control berechneten Ansteuerwerte in einer Textdatei aufgezeichnet ("Simulationsdatei").

Anwender (z. B. **Job-Designer**) können dann durch Analysieren der aufgezeichneten Werte – bereits vor einem "Realtest" mit Laser und Werkstück – ermitteln, an welchen Stellen ein **Job** optimiert werden kann oder muss:

- Ein **Job** *muss* optimiert werden, falls berechnete Ansteuerwerte die Systemgrenzen überschreiten, siehe **Kapitel 2.2.1 "Systemgrenzen ermitteln"**, **Seite 19**.
- Ein **Job** *kann* hinsichtlich Prozesszeit (Durchsatz) und Ausführgenauigkeit (Markiierungsergebnisqualität) optimiert werden, siehe **Kapitel 2.6 "Über das Optimieren von syncAXIS control-basierten Anwenderprogrammen"**, **Seite 36**.

Um syncAXIS control-basierte Anwenderprogramme im Simulationsmodus auszuführen:

- muss der **Dongle** eingesteckt sein
- muss zuvor in der **syncAXISConfig.xml** (die als Argument in **slsc_cfg_initialize_from_file** angegeben ist) – im Tag `<cfg:SimulationMode>` – der Simulationsmodus eingestellt werden:

```
<cfg:SimulationMode>true
</cfg:SimulationMode>
```

(zur Initialisierung siehe auch **Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen"**, **Seite 26**)
- muss in der gleichen XML-Konfigurationsdatei als `<cfg:SimOutputFileDirectory>`-Wert der Ziel-Ordnerpfad für die Simulationsdatei angegeben werden

Mit den obigen Einstellungen initialisiert das syncAXIS control-basierte Anwenderprogramm die **syncAXIS control-Instanz** im Simulationsmodus.

Sobald es bei der Ausführung **slsc_ctrl_start_execution** erreicht, wird im angegebenen Ziel-Ordnerpfad 1 (eine) (≥ V1.5.0) Simulationsdatei erzeugt.

Mit syncAXIS control ≥ V1.3 wird dabei folgende Dateinamenskonvention angewendet:

- `Simulation_ID_<Job-ID>_TS_<13 Ziffern>.txt`⁽¹⁾

(1) Beispiel: `Simulation_ID_2_TS_1546938743472.txt`.
TS = timestamp. Anmerkung: in V1.2 noch
* _<Scan-Device>_TS_* statt *_TS_*.

Eine Simulationsdatei (\geq V1.5) enthält:

- (1) Die für den **Job** *tatsächlich* verwendete Konfiguration im xml-Format⁽¹⁾⁽²⁾
 - Diese Parameter wurden aus der `syncAXISConfig.xml` bei der Initialisierung ausgelesen
 - Die Parameter-Werte können seitdem unverändert geblieben oder mittlerweile (durch vor der **Job**-Ausführung aufgerufene Konfigurations-Funktionen (`slsc_cfg_*`), siehe **Kapitel 3.1.1 "Konfigurations-Funktionen (`slsc_cfg_*`)"**, **Seite 76**) geändert worden sein
- (2) Eine Zeile (beginnend mit "`<!--Simulation`
`output: "`") mit den Spaltenkopfnamen, siehe Tabelle **"Simulationsdateien \geq V1.5: Mögliche Werte pro Datensatz (siehe 3, Seite 32)", Seite 33**
- (3) Nachfolgende Zeilen = die einzelnen Datensätze
 - Jeder Datensatz
 - entspricht einem RTC6-Mikrovektorbefehl ($10 \mu s$)
 - enthält mehrere semikolonseparierte Werte
 - Die Anzahl der Werte in den Datensätzen von Simulationsdateien \geq V1.5 ist nicht mehr konstant, sondern können variieren, siehe Tabelle **"Simulationsdateien \geq V1.5: Mögliche Werte pro Datensatz (siehe 3, Seite 32)", Seite 33**. Die Zeilen können also bei \geq V1.5 unterschiedlich lang sein
- (4) Letzte Zeile: "`-->`"

Zur Überprüfung der berechneten Ansteuerwerte auf Überschreiten der Systemgrenzen können die Daten aus den Simulationsdateien mit einem passenden Software-Tool (wie dem SCANLAB syncAXIS Viewer, siehe **Abbildung 8, Seite 35**) automatisch ausgelesen und ausgewertet oder zur Auswertung grafisch dargestellt werden.

Beispielhaft zeigt **Abbildung 7, Seite 34** eine solche grafische Darstellung:

- Hier sind zum einen die Positionswerte (X,Y) von Scan-Kopf und Verfahrtsch und **"Toggle-Signal" am Ende des Mikrovektors** aufgetragen, die bei Ausführung eines Beispiel-Jobs im Simulationsmodus in einer Simulationsdatei aufgezeichnet wurden.
- Zusätzlich sind hier Verfahrtsch-Geschwindigkeit, -Beschleunigung und -Ruck aufgetragen, die durch einfaches bzw. mehrfaches numerisches Ableiten aus den Verfahrtsch-Positionswerten berechnet wurden.
- Die Abbildung zeigt, dass die Y-Positionswerte des Scan-Kopfs an einer Stelle des Beispiel-Jobs in einen grenzwertigen Bereich kommen.
- Für den **Betriebsmodus `ScannerAndStage`** gilt: Zu Beginn sind die Kurvenwerte noch bei $0^{(3)}$, da die Algorithmen zur **Bewegungsaufteilung** erst nach einigen Zehntelsekunden ("Anfangs-Antwortzeit des Filters") wirksam werden. Außerdem veranlasst die syncAXIS control-Software nach dem letzten Bewegungskommando Bewegungen, um das System in einen stabilen Zustand zu bringen.

(1) Auswertungswerkzeuge können die Systemgrenzen direkt auslesen.

(2) Kann mit syncAXIS Viewer \geq V1.5 als xml-Datei exportiert werden.

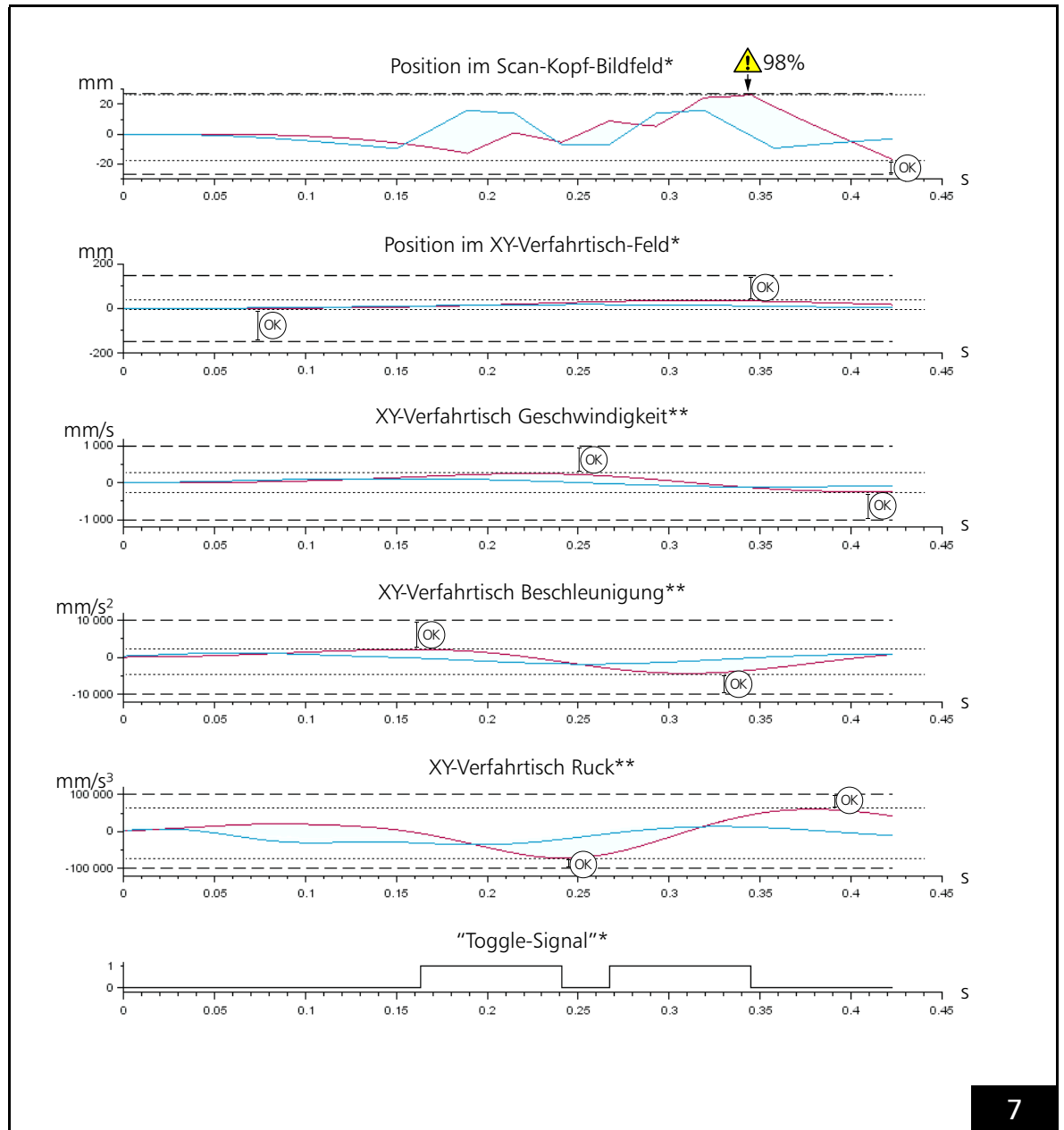
(3) Die absolute Position muss nicht bei 0 sein, die relative dagegen schon.

Simulationsdateien ≥ V1.5: Mögliche Werte pro Datensatz (siehe 3, Seite 32)		
Auftreten pro Datensatz (siehe 3, Seite 32)	Name (siehe 2, Seite 32)	Bedeutung
1	ScanDevice1X	y-Wert für Scan-Device 1. In mm.
1	ScanDevice1Y	y-Wert für Scan-Device 1. In mm.
0...1	ScanDevice2X	x-Wert für Scan-Device 2. In mm.
0...1	ScanDevice2Y	y-Wert für Scan-Device 2. In mm.
0...1	ScanDevice3X	x-Wert für Scan-Device 3. In mm.
0...1	ScanDevice3Y	y-Wert für Scan-Device 3. In mm.
0...1	ScanDevice4X	x-Wert für Scan-Device 4. In mm.
0...1	ScanDevice4Y	y-Wert für Scan-Device 4. In mm.
0...1	StageX	x-Wert für Verfahrtsch. In mm.
0...1	StageY	y-Wert für Verfahrtsch. In mm.
1	NumLaserOn	Anzahl nachfolgender LaserOnDelays-Werte [0...10].
0...10 (siehe NumLaserOn)	LaserOnDelays	"Laser active"-Betrieb-Start-Zeitpunkt. ^(a) Relativ zum Beginn des Mikrovektors. In μ s.
1	NumLaserOff	Anzahl nachfolgender LaserOffDelays-Werte [0...10].
0...10 (siehe NumLaserOff)	LaserOffDelays	"Laser standby"-Betrieb-Start-Zeitpunkt. ^(a) Relativ zum Beginn des Mikrovektors. In μ s.
1	LaserToggle	"Toggle-Signal" am Ende des Mikrovektors. 1: "Laser active"-Betrieb. 0: "Laser standby"-Betrieb.
1	ActiveChannel0	1. ActiveChannel-Wert ^(b) : Einheit ist die des eingestellten Kanals. ^(c) Wenn kein ActiveChannel definiert ist: 0.
1	ActiveChannel1	2. ActiveChannel-Wert ^(b) . Siehe 1. ActiveChannel.
1	CommandCount	Laufende Nummer der Job-Funktion (slsc_list_*) im Job.

(a) Falls keine "Laser active"-Betriebe oder "Laser standby"-Betriebe stattfinden, ist das in ≥ V1.5 anhand von NumLaserOn/NumLaserOff zu erkennen.

(b) Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.

(c) Wenn als ActiveChannel SpotDistance eingestellt ist: Laserspotgeschwindigkeit. In mm/s.



Grafische Darstellung von Simulationsdaten eines Beispiel-Jobs:

* Daten aus der Simulationsdatei (Positionswerte und "Toggle-Signal" am Ende des Mikrovektors.)

** Daten, die aus Simulationsdateiwerten abgeleitet sind (Geschwindigkeit, Beschleunigung, Ruck)

blaue Linie: x-Werte

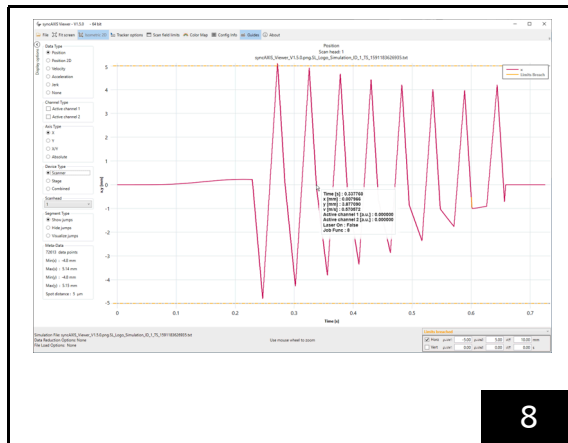
rote Linie: y-Werte

— — — — — : Systemgrenzen

- - - - - : Hilfslinie (auf Höhe der Kurven-Maximalwerte zur leichteren optischen Orientierung eingezeichnet)

Hinweise

- Daten aus Simulationsdateien können mit dem Software-Tool SCANLAB syncAXIS Viewer visualisiert werden, das als Teil des syncAXIS control-Software-Pakets mitgeliefert wird, siehe [Abbildung 8, Seite 35](#).



SCANLAB syncAXIS Viewer.

- Über Simulationen kann die reale Markierergebnisqualität nicht vorhergesagt werden. Deshalb muss nach der Evaluation der Simulation in jedem Fall das Markiermuster tatsächlich ausgeführt werden ("Realtest"⁽¹⁾) um das Ergebnis auf die erzielte Qualität zu bewerten. Ab syncAXIS control \geq V1.0 gibt es auch im Hardwaremodus die Möglichkeit, nach Berechnung eines Jobs (mit [slsc_ctrl_get_job_characteristic](#)) bestimmte Charakteristika abzufragen (z. B. die max. Scan-Kopf-Position im Job). Unter Umständen muss das syncAXIS control-basierte Anwenderprogramm dann noch mal nachgebessert und nochmals im Simulationsmodus ausgeführt werden (letzteres um erneut auf Einhaltung der Systemgrenzen zu prüfen).

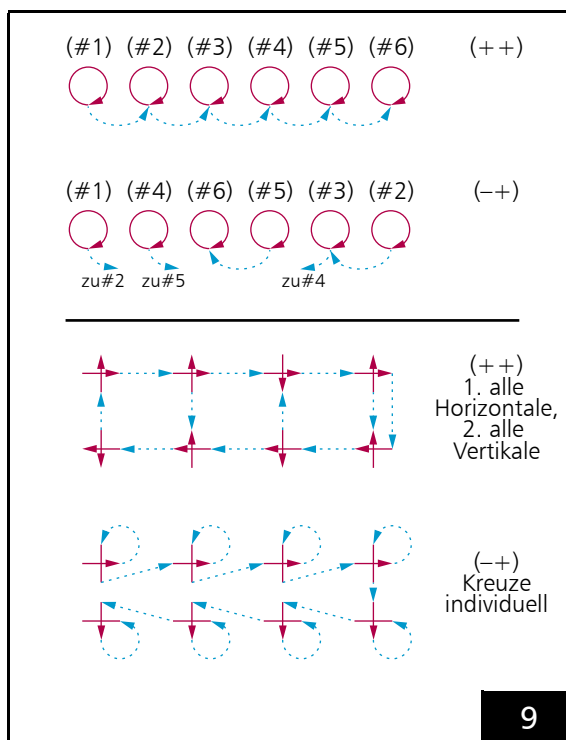
(1) Dazu muss der Simulationsmodus in der `syncAXISConfig.xml` wieder ausgeschaltet werden:

```
<cfg:SimulationMode>false
</cfg:SimulationMode>.
```

2.6 Über das Optimieren von syncAXIS control-basierten Anwenderprogrammen

2.6.1 Optimierungsmöglichkeiten

In der Regel muss ein syncAXIS control-basiertes Anwenderprogramm nicht nur (wie beschrieben in Kapitel 2.2.4 "Jobs simulieren und nachbessern", Seite 24 und Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31) auf Einhaltung der Systemgrenzen geprüft und nachgebessert, sondern auch hinsichtlich Prozesszeit (Durchsatz) und Ausführngenauigkeit (Markiierungsergebnisqualität) optimiert werden. So eine Optimierung kann einerseits im syncAXIS control-basierten Anwenderprogramm selber erfolgen, andererseits aber auch durch Ändern von Initialisierungswerten (in der verwendeten `syncAXISConfig.xml`) für die **syncAXIS control-Instanz**. Ein Ansatz zur Steigerung des Durchsatzes ist auch die Reduzierung der Markiermuster-Ausführzeiten durch eine geschickte Reihenfolge der **Markier-Funktionen** und **Sprung-Funktionen** im Quellcode (z.B. wegen kürzerer Sprünge wie in Abbildung 9 illustriert).



Unterschiedliche Funktionsabfolgen im Quellcode können für das gleiche Markiierungsergebnis unterschiedliche Ausführzeiten zur Folge haben.

Eine weitere Möglichkeit zur Durchsatzsteigerung ist eine Erhöhung der Prozessgeschwindigkeiten.

- Hierzu kann zum einen in der `syncAXISConfig.xml` – in den Tags `<cfg:JumpSpeed>` und `<cfg:MarkSpeed>` – eine "globale" Standardeinstellung für die Sprunggeschwindigkeit und Markiergeschwindigkeit für alle **Jobs** vorgegeben werden (diese gelten, sofern in einem **Job** keine "lokalen" Werte angegeben werden, s.u.). Beide Geschwindigkeiten beeinflussen die Bewegung von sowohl Scan-Kopf als auch Verfahrtschritt.
 - Die Markiergeschwindigkeit ist oft vom Laserprozess vorgegeben.
 - Die Sprunggeschwindigkeit kann typischerweise höher als die Markiergeschwindigkeit gesetzt werden. Eine höhere Sprunggeschwindigkeit verringert zwar die Prozesszeit (höherer Durchsatz). Hohe Sprunggeschwindigkeiten führen allerdings u.U. auch zu Ungenauigkeiten. Hohe Sprunggeschwindigkeiten können u.U. auch zu einer Überschreitung von Systemgrenzen führen.
- Zur Feinoptimierung von Prozesszeit und Ausführngenauigkeit können zusätzlich innerhalb eines **Jobs** (d. h. innerhalb des syncAXIS control-basierten Anwenderprogramms) – mit `slsc_list_set_jump_speed` und `slsc_list_set_mark_speed` – die "lokale" Sprunggeschwindigkeit und Markiergeschwindigkeit angegeben werden:
 - im Prinzip auch nur für einen einzelnen Markier- oder Sprungvektor,
 - die Änderungen gelten ab der Einfügestelle aber nur bis höchstens zum Ende des **Jobs**.

In manchen Fällen kann eine Durchsatzsteigerung auch durch ein Aufteilen des **Jobs** (im Quellcode) auf mehrere Teil-**Jobs** erreicht werden:

- zum einen auf Teil-**Jobs**, die nur diejenigen Markiermuster-Anteile enthalten, die nur vom (i.d.R. schnelleren) Scan-Kopf erzeugt werden sollen (spätere Ausführung dann im **Betriebsmodus ScannerOnly**)
- zum anderen auf Teil-**Jobs**, die nur diejenigen Markiermuster-Anteile enthalten, die mit **Bewegungsaufteilung** von Scan-Kopf und Verfahrtschritt erzeugt werden sollen (spätere Ausführung dann im **Betriebsmodus ScannerAndStage**).

Für diejenige Teil-Jobs, die bei gleichzeitiger Bewegung von Scan-Kopf und Verfahrtisch (d.h. im Betriebsmodus "ScannerAndStage") markiert werden sollen, kann der FilterBandwidth-Wert vorgegeben werden⁽¹⁾. Dieser Wert wirkt sich in diesem Betriebsmodus auf die Berechnung der Bewegungsaufteilung zwischen Scan-Kopf und Verfahrtisch durch syncAXIS control aus:

- Eine Erhöhung des FilterBandwidth-Werts führt zu einem höheren Bewegungsanteil des Verfahrtischs (und damit umgekehrt zu einer Verringerung des genutzten Scan-Kopf-Arbeitsfelds).
 - Mögliche Vorteile dieser Einstellung:
 - Der Verfahrtisch ist in manchen Situationen genauer als der Scan-Kopf.
 - Der Verfahrtisch hat ein größeres Arbeitsfeld als der Scan-Kopf und kann daher auch größere Vektorlängen allein fahren.
 - Mögliche Nachteile dieser Einstellung:
 - Die Prozessgeschwindigkeit muss ggf. verringert werden (Der Verfahrtisch ist langsamer als der Scan-Kopf und der Verfahrtisch kann der Trajektorie vielleicht nicht folgen).
 - Der Verfahrtisch muss mehr Masse bewegen. Daher können bei hohen Rucken vom Verfahrtisch Schwingungen ausgehen.
 - Der Verfahrtisch hat einen höheren Stromverbrauch als der Scan-Kopf.
 - Systemgrenzen könnten verletzt werden, insbesondere die Dynamiken des Verfahrtisches.
- Eine Verringerung des FilterBandwidth-Werts führt zu einem kleineren Bewegungsanteil des Verfahrtischs (und damit umgekehrt zu einer Vergrößerung des Bewegungsanteils und genutzten Arbeitsfelds des Scan-Kopfs).
 - Mögliche Vorteile dieser Einstellung:
 - Der Scan-Kopf ist schneller als der Verfahrtisch. Dadurch kann eine höhere Geschwindigkeit eingestellt werden.
 - Der Scan-Kopf hat einen geringeren Stromverbrauch als der Verfahrtisch.
 - Mögliche Nachteile dieser Einstellung:
 - Der Scan-Kopf ist in manchen Situationen ungenauer als der Verfahrtisch. Der Scan-Kopf erzielt z. B. an seinem Arbeitsfeldrand weniger genaue Ergebnisse. Dementsprechend ist die Gesamtgenauigkeit schlechter.
 - Der Scan-Kopf hat ein kleineres Arbeitsfeld als der Verfahrtisch und kann daher auch größere Vektorlängen nicht allein fahren.
 - Systemgrenzen könnten verletzt werden, insbesondere die Ansteuerwerte für den Scan-Kopf.
 - Beispielsweise könnten Anwender, die feststellen, dass der Verfahrtisch der Bewegung nicht mehr nachkommt (dies kann sich in einer Fehlermeldung der Tischansteuerung zeigen), die genutzte Verfahrtischdynamik reduzieren (d.h. die Verfahrtischbewegungen vermindern), indem sie den FilterBandwidth-Wert verringern.
- Der Durchsatz wird durch das alleinige Ändern des FilterBandwidth-Werts nur geringfügig verändert (die Jobdauer wird hauptsächlich von den Markier- und Sprunglängen und den dabei verwendeten Geschwindigkeiten bestimmt).
- Als weitere grobe Zusammenhänge für den FilterBandwidth-Wert gelten:
 - $1/\langle \text{c}fg:\text{FilterBandwidth} \rangle\text{-Wert} \sim \text{genutztes Scan-Kopf-Arbeitsfeld}$
 - $\langle \text{c}fg:\text{FilterBandwidth} \rangle\text{-Wert} \sim \text{höchste Verfahrtischbeschleunigung}$
 - $\langle \text{c}fg:\text{FilterBandwidth} \rangle\text{-Wert}^2 \sim \text{höchster Verfahrtischruck}$

(1) In der syncAXISConfig.xml, Tag `<cfg:FilterBandwidth>`. Dieser Wert ist die "globale" Standardeinstellung für alle Jobs.

2.6.2 Iteratives Vorgehen

Leider können Durchsatz und Genauigkeit nicht beliebig gesteigert werden. Maximaler Durchsatz und maximale Genauigkeit sind i.d.R. sogar konkurrierende Ziele.

Realistisches Ziel eines Optimierungsprozesses ist es, in einem i.d.R. iterativen Vorgehen, die für den jeweiligen Prozess und das jeweilige Markiermuster optimalen Prozessparameter zu finden, um einen möglichst hohen Durchsatz bei gleichzeitig noch akzeptabler Genauigkeit (oder umgekehrt eine möglichst hohe Genauigkeit bei noch akzeptablem Durchsatz) zu erreichen.

Als Unterstützung für solch einen Optimierungsprozess können syncAXIS control-basierte Anwenderprogramme im Simulationsmodus ausgeführt werden, siehe [Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31](#). Auch ohne (aufwändigere) Realtests kann damit u.a. festgestellt werden, wie sich Parameteränderungen auf die Prozesszeit, auf die **Bewegungsaufteilung** zwischen Scan-Kopf und Verfahrtschritt und auf die Einhaltung von Systemgrenzen auswirken.

SCANLAB empfiehlt das folgende Vorgehen:

- Erstellen und kompilieren Sie ein syncAXIS control-basiertes Anwenderprogramm (Quellcode) mit dem gewünschtem Markiermuster.
- Stellen Sie sicher, dass in der verwendeten `syncAXISConfig.xml` die realen Systemgrenzen als Grenzwerte eingetragen sind.
- Prüfen Sie zusätzlich vor dem ersten Iterationsschritt die folgenden Einstellungen in `syncAXISConfig.xml`:
 - Die Prozessgeschwindigkeiten sollten so konservativ (niedrig) eingestellt sein, dass die Systemgrenzen möglichst nicht überschritten werden.
 - **Betriebsmodus:**

```
<cfg:InitialOperationMode>
ScannerAndStage
</cfg:InitialOperationMode>
```
 - **Simulationsmodus Ein**

```
<cfg:SimulationMode>true
</cfg:SimulationMode>
```

- Ausgabepfad für die Simulationsdateien angeben

```
<cfg:SimOutputFileDirectory>...
</cfg:SimOutputFileDirectory>
```

- Führen Sie folgende Iterationsschritte aus:
 - Programmparameter ändern:
 - z. B. die Sprunggeschwindigkeit in `syncAXISConfig.xml` erhöhen, um den Durchsatz zu erhöhen
 - z. B. den `FilterBandwidth`-Wert verringern, um den Bewegungsanteil des Verfahrtschritts zu verringern und damit umgekehrt den Bewegungsanteil und das genutzte Arbeitsfeld des Scan-Kopfs zu erhöhen
 - z. B. den Quellcode ändern, um andere Optimierungen (z. B. Funktionsabfolge) vorzunehmen und dann bei Bedarf erneut kompilieren
 - syncAXIS control-basiertes Anwenderprogramm im Simulationsmodus ausführen: siehe [Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31](#)
 - Daten (u.a. Positionswerte und **"Toggle-Signal" am Ende des Mikrovektors**.) aus der erzeugten Simulationsdatei auslesen und Geschwindigkeit, Beschleunigung und Ruck daraus ableiten.⁽¹⁾⁽²⁾
 - Simulationsdaten analysieren
 - u.a. jeden Parameter auf Überschreitung der Systemgrenzen prüfen (ggf. hierzu auch die erzeugte Log-Datei inspizieren, siehe Fußnote auf [Seite 23](#)). Ist eine Grenzwertüberschreitung erkennbar, dann muss auf jeden Fall in einem weiteren Iterationsschritt der betreffende Parameter entsprechend angepasst werden und erneut eine Simulation und Ergebnisanalyse durchgeführt werden.
 - Die Prozesszeit lässt sich aus den Daten der erzeugten Simulationsdatei ablesen. Beachten Sie dabei, dass die Berechnungszeit (= Dauer, bis ein **Job** startbereit ist) im Simulationsmodus signifikant länger ist als im Hardwaremodus.

(1) syncAXIS Viewer berechnet diese abgeleiteten Werte.

(2) Für syncAXIS control \geq V1.2.4 gilt: Die Extrema des aktuellen **Jobs** können mit `slsc_ctrl_get_job_characteristic` abgefragt werden.

- Wahlweise können Sie das syncAXIS control-basierte Anwenderprogramm immer dann, wenn die Simulationsergebnisse keine Grenzwertüberschreitungen zeigen, auch nachträglich zusätzlich in einem Realtest mit Laser, Scan-Kopf und Verfahrtisch ausführen, um eine reale Markierung auf einem Werkstück zu erzeugen. Überprüfen Sie dann die erzeugte Markierung auf die erreichte Qualität. Führen Sie bei Bedarf auch auf Basis dieser Qualitätsprüfung weitere Iterationsschritte durch (auch hier anfangs wieder inkl. Simulation und Analyse auf Überschreitung von Systemgrenzen).

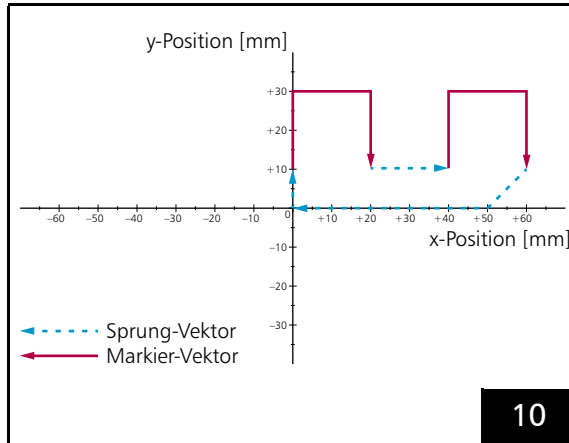
Beispiel

Die folgende Tabelle zeigt beispielhaft einige eingestellte Parameter und Simulationsergebnisse für vier Schritte eines solchen Optimierungsprozesses (für ein einfaches Markiermuster, siehe **Abbildung 10, Seite 40**).

Hinter den Schritten 2 bis 4 stecken eigentlich jeweils mehrere Teil-Iterationsschritte, weil anfangs nicht klar ist, mit welchen Einstellparametern im Ergebnis die Systemgrenzen nur angerissen, aber nicht überschritten werden.

- Im 1. Iterationsschritt sind die eingestellten Parameter so konservativ gewählt, dass in der Simulation alle Systemgrenzen einfach eingehalten werden.
- Im 2. Iterationsschritt werden Sprunggeschwindigkeit und Markiergeschwindigkeit erhöht (und der **FilterBandwidth**-Wert unverändert gelassen). In der Simulation ist zu erkennen, dass sich dadurch die Prozesszeit erheblich reduziert, aber der Scan-Kopf (u.U. grenzwertig) sein Arbeitsfeld nahezu komplett ausschöpft.
- Im 3. Iterationsschritt wird der **FilterBandwidth**-Wert erhöht, um den Bewegungsanteil des Scan-Kopfs zu verringern (Sprunggeschwindigkeit und Markiergeschwindigkeit werden unverändert gelassen). In der Simulation ist zu erkennen, dass dadurch das verwendete Scan-Kopf-Arbeitsfeld verringert wird – bei gleichbleibender Prozesszeit. Allerdings erhöht sich dadurch der Verfahrtisdruck in einen u.U. grenzwertigen Bereich.
- Im 4. Iterationsschritt werden Sprunggeschwindigkeit und Markiergeschwindigkeit erhöht (was zu einer Reduzierung der Prozesszeit führt) und der **FilterBandwidth**-Wert verringert (was zwar den Verfahrtisdruck verringert, aber auch das verwendete Scan-Kopf-Arbeitsfeld in einen u.U. grenzwertigen Bereich erhöht).

Ob es von Vorteil ist, eher das komplette Scan-Kopf-Arbeitsfeld auszunutzen oder ob umgekehrt eher die Dynamik auf den Verfahrtisdruck zu verlagern, muss letztlich anhand von Realtests nachgeprüft werden.



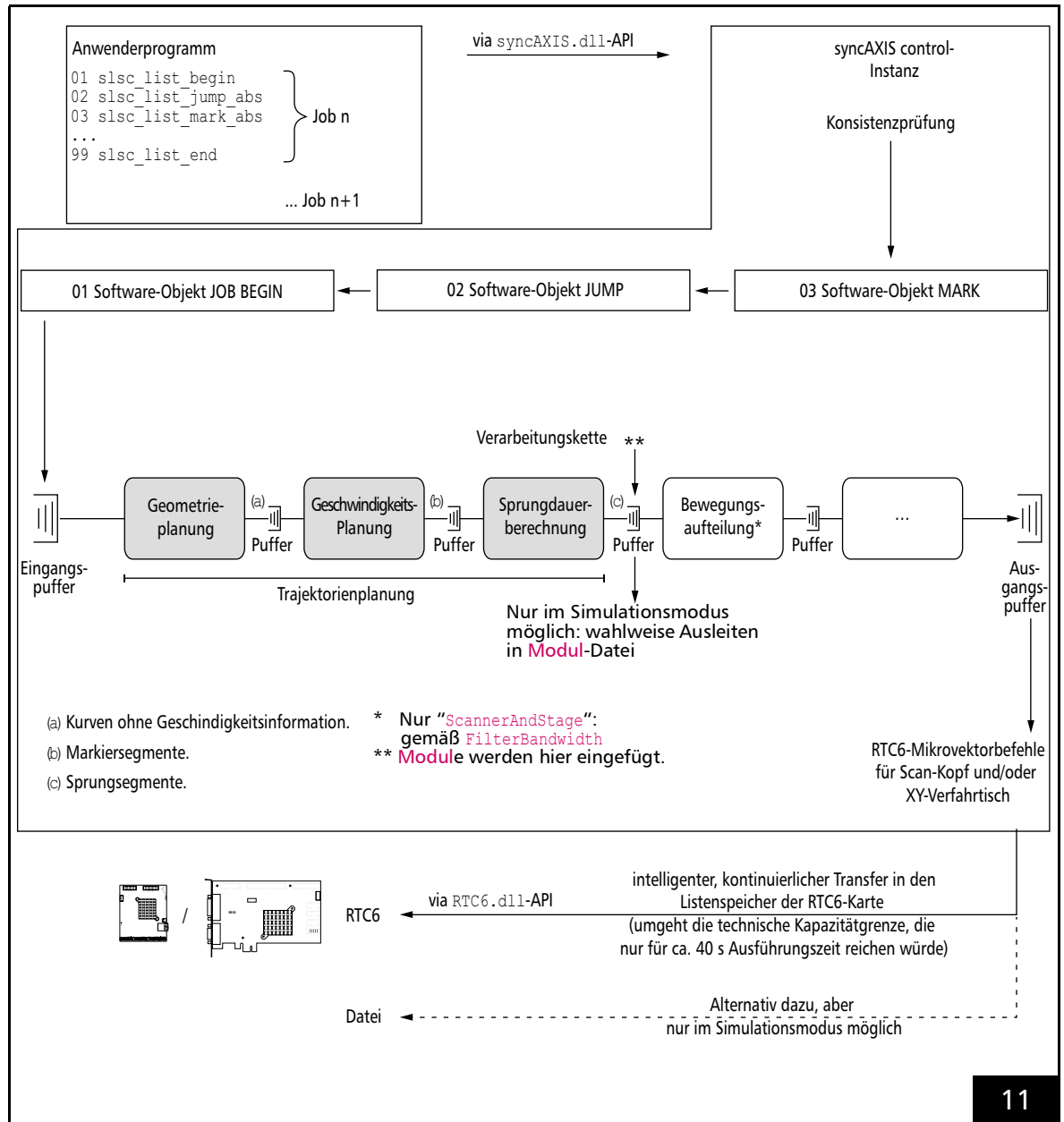
Markiermuster: zwei Quadrate ohne Grundlinien.

		Schritt 1	Schritt 2	Schritt 3	Schritt 4
Eingestellte Parameter					
	Markiergeschwindigkeit [mm/s]	480	770	770	840
	Sprunggeschwindigkeit [mm/s]	480	770	770	840
	FilterBandwidth-Wert [Hz]	1,95	1,95	2,25	2,10
Simulationsergebnis					
	Prozesszeit, relativ zur Prozesszeit 600 ms von Schritt 1	100%	71%↓ ^(a)	71%	68%↓ ^(a)
	Benutztes Scan-Kopf-Arbeitsfeld, relativ zur Systemgrenze (± 27 mm) x (± 27 mm)	63%	98%↑ ^(b)	87%↓ ^(a)	99%↑ ^(b)
	max. Verfahrtischgeschwindigkeit, relativ zur Systemgrenze 1.000 mm/s	27%	25%	29%	27%
	max. Verfahrtischbeschleunigung, relativ zur Systemgrenze 10.000 mm/s ²	38%	44%	58%	51%
	max. Verfahrtischruck, relativ zur Systemgrenze 100.000 mm/s ³	52%	72%	98%↑ ^(b)	90%

(a) grün: Wertänderung relativ zum vorherigen Schritt wird als "vorteilhaft" interpretiert.

(b) gelb: Wert wird als "grenzwertig" interpretiert.

2.7 Über Vorgänge bei der Ausführung des Anwenderprogramms



Ein **Job** wird Funktion-um-Funktion der **syncAXIS control-Instanz** übergeben. Diese werden in einzelne **Job-Elemente** (Software-Objekte) umgewandelt. Jedes **Job-Element** durchläuft sequenziell eine Verarbeitungskette. Zu unterschiedlichen Verarbeitungszeiten kommen auch Pufferzeiten hinzu. Deshalb ist der Status eines **Jobs** ein aus den Status der einzelnen **Job-Elemente** kombinierter Status, siehe auch **Abbildung 12, Seite 43** und **Abbildung 13, Seite 44**.

2.7.1 Über die Puffer der syncAXIS control-Instanzen

Puffer sind syncAXIS-DLL-interne Zwischenspeicher der Verarbeitungskette, siehe [Abbildung 11](#), [Seite 41](#). Die Software-Objekte (wie "JOB BEGIN", "JUMP", "MARK etc.) durchlaufen diese Verarbeitungskette. Puffer sind notwendig, weil Verarbeitungszeiten in den einzelnen Gliedern der Verarbeitungskette anfallen und diese für unterschiedliche Software-Objekte unterschiedlich lang sind. Am Beginn der Verarbeitungskette steht der **Eingangspuffer**.

- Aus Benutzersicht ist beim Beladen der Puffer Folgendes zu beachten: wenn der **Eingangspuffer** voll ist, dann kann syncAXIS-DLL die aufgerufene Job-Funktion (`slsc_list_*`) nicht ausführen. Abhängig vom `slsc_ListHandlingMode`, kann dies unterschiedliche Folgen haben.
 - `slsc_ListHandlingMode_ReturnAtOnce`
Die zuletzt aufgerufene Job-Funktion (`slsc_list_*`) *ignoriert* (es wird *keine* Exception geworfen). Im Markierungsergebnis können u.U. dadurch Teile fehlen. Um diesen Fall zu vermeiden, können im Anwenderprogramm folgende Maßnahmen ergriffen werden:
 - Mit `slsc_ctrl_is_list_input_buffer_full` kann vor dem Beladen der Puffer abgefragt werden, ob der **Eingangspuffer** voll ist.
 - Über den Rückgabewert einer Job-Funktion (`slsc_list_*`) kann geprüft werden, ob sie abgelehnt wurde (**Bit #04** gesetzt (`BUFFER_FULL`)).
 - `slsc_ListHandlingMode_RepeatWhileBufferFull` und `slsc_ListHandlingMode_RepeatWhilePredicate`
syncAXIS-DLL schließt die Ausführung von Job-Funktionen (`slsc_list_*`) erst dann ab, sobald wieder ausreichender **Eingangspuffer** frei ist. Erst danach wird das Anwenderprogramm fortgesetzt. Benutzer müssen also sich im Klaren sein, dass das Anwenderprogramm (oder der entsprechende Thread) möglicherweise blockiert wird. Sollen während der Blockierung weitere Funktionen (z.B. ein **Job-Start** mit `slsc_ctrl_start_execution`) ausgeführt werden, muss dies durch asynchrone Programmierung (oder über **Callback-Funktionen**) in einem anderen Thread des Anwenderprogramms stattfinden.

- Aus Benutzersicht ist beim Beladen der Puffer – während die RTC6-Karte eine **Liste** ausführt – darauf zu achten, dass kein **Pufferunterlauf** (siehe [Glossareintrag Seite 14](#)) erzeugt wird. Siehe [Abschnitt "Pufferunterläufe vermeiden", Seite 42](#).

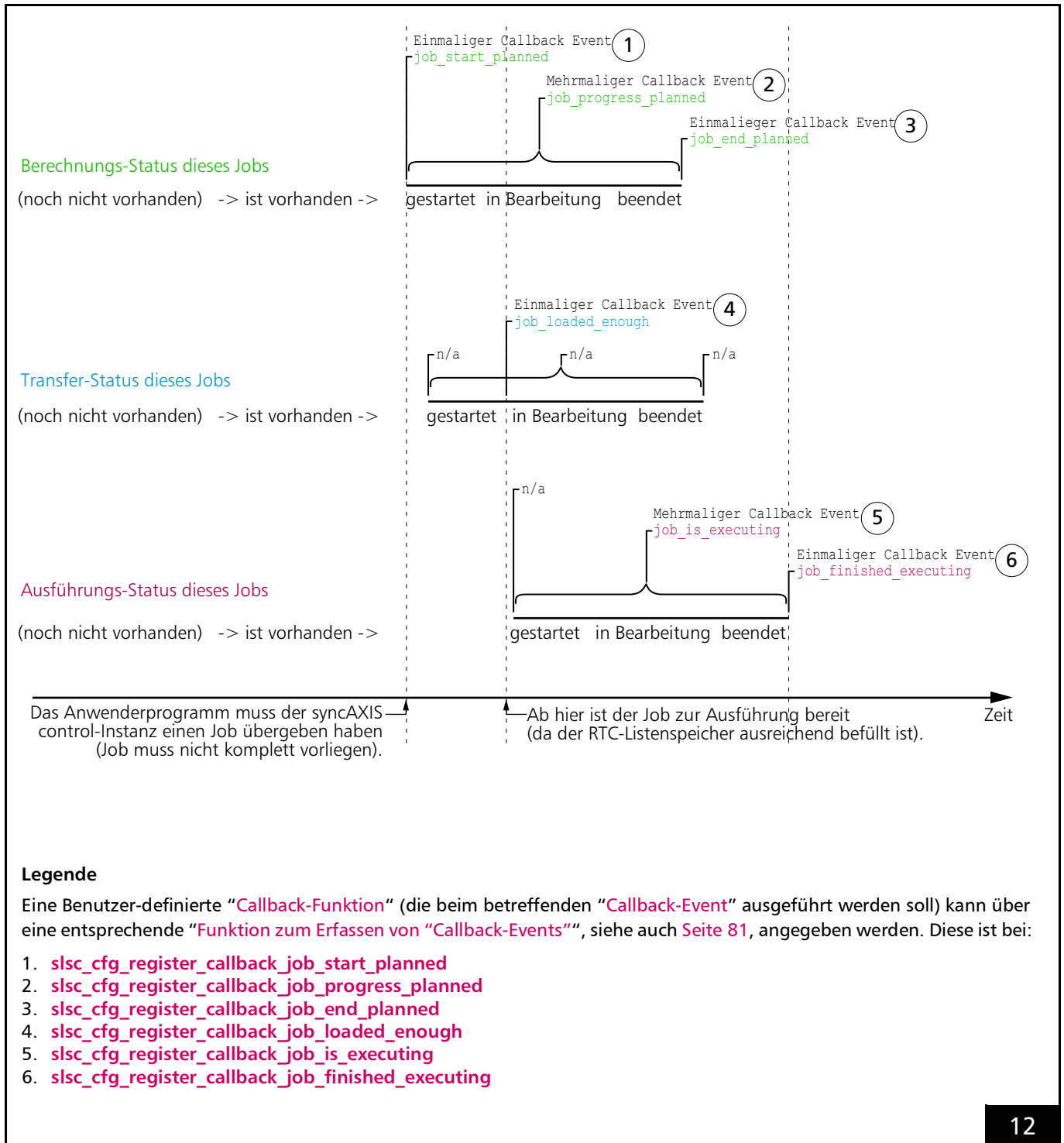
Pufferunterläufe vermeiden

Im Fall von `0x 00 00 00 02 00 00 00 02`

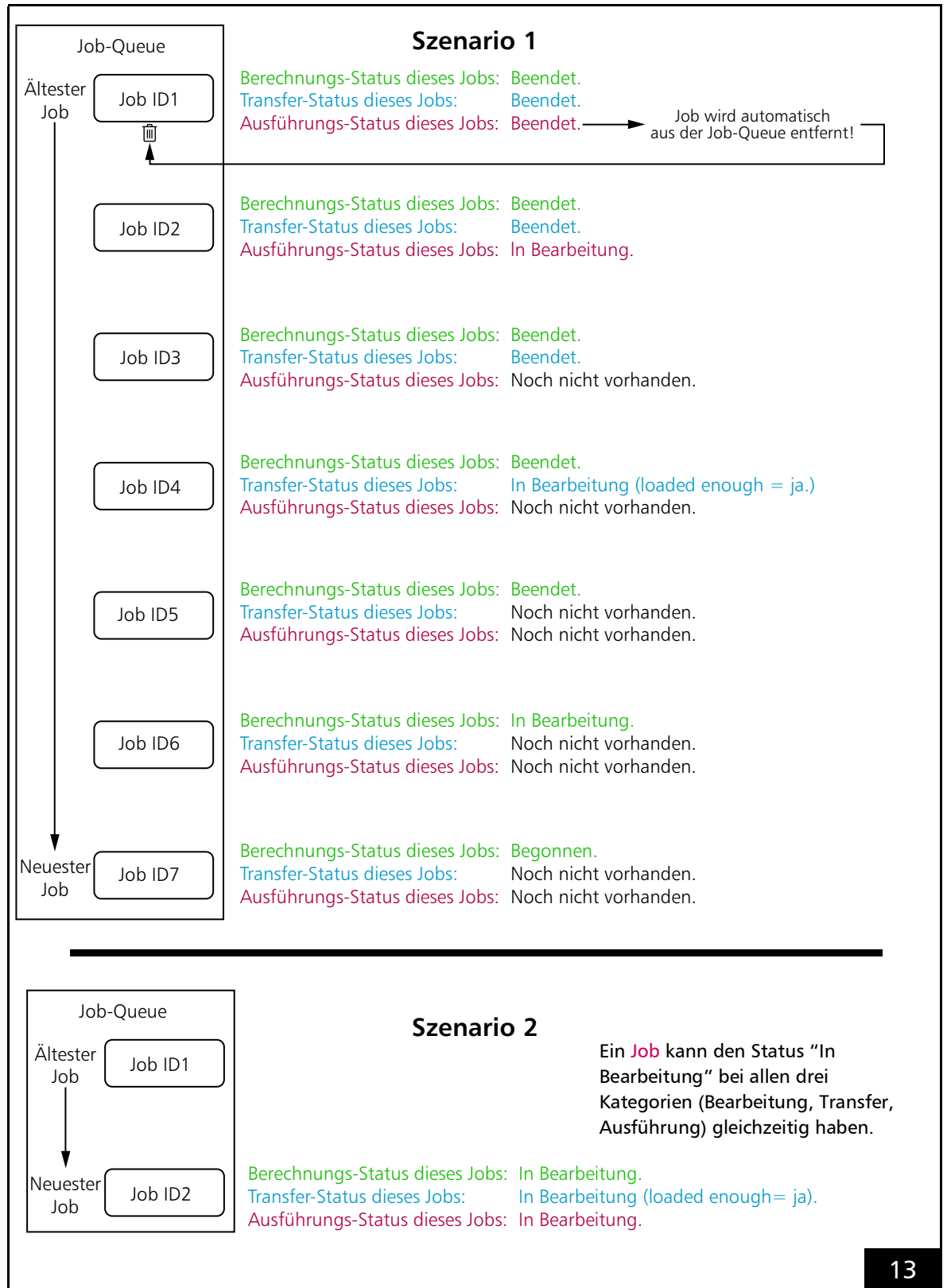
`EXEC_BUFFER_UNDERRUN` können Sie folgende

Maßnahmen zur Behebung des Fehlers treffen:

- (1) Benutzen Sie **Module**, siehe [Kapitel 2.11 "Über das Arbeiten mit "Modulen""](#), [Seite 65](#).
- (2) Geben Sie PC-Ressourcen frei, indem Sie z.B. nicht mehr benötigte Programme schließen.
- (3) Räumen Sie mehr Vorberechnungszeit ein. Natürlich ist es die schnellste Art einen **Job** auszuführen, seine Ausführung zu starten, sobald sein Ausführungszustand zur Ausführung bereit = "`job_loaded_enough`" ist. Wenn Sie jedoch Probleme mit einem **Pufferunterlauf** haben, können Sie syncAXIS control mehr Zeit für die Berechnung zur Verfügung stellen. In solchen Fällen ist es sinnvoll, die **Trajektorie** von syncAXIS control so weit wie möglich vorberechnen zu lassen, d.h. bis der **Eingangspuffer** voll ist. Wir empfehlen daher, den **Job** erst dann zu starten, wenn dieser Puffer ausreichend gefüllt ist.
- (4) Arbeiten Sie mit kürzeren **Jobs**
Um Punkt (3) besonders viel Wirkung zu verleihen, ist es ratsam, bei Problemen mit einem **Pufferunterlauf** lange **Jobs** in mehrere, kürzere **Jobs** zu zerlegen. So wird Rechenzeit zwischen den einzelnen **Jobs** hinzugefügt.
- (5) Parameter `VectorResolution`
Achten Sie darauf, dass der Parameter `VectorResolution` nicht zu fein definiert ist. Eine hinreichend grob definierte `VectorResolution` kann helfen, sehr feine Eingangsdaten in längere Vektoren umzuwandeln.
- (6) Testen Sie Ihre **Jobs** vorab
Sie können einen **Job** mit deaktiviertem Laser mehrmals ausführen, um sicher zu stellen, dass kein Risiko für einen **Pufferunterlauf** besteht.



Ein **Job** ist durch 3 unterschiedliche Status von Berechnung, Transfer und Ausführung gekennzeichnet (die zunächst aber noch gar nicht vorhanden sind). An bestimmten Stellen werden "Callback-Events" erzeugt. Bei der Berechnung ("In Bearbeitung") werden aus den Job-Funktionen (`slsc_list_*`) laufend RTC6-Mikrovektorbefehle erstellt. Diese werden an die RTC6-Karte übertragen (Transfer). Bei Start der Ausführung beginnt die RTC6-Karte mit der Ausführung der RTC6-Mikrovektorbefehle.



2.7.2 Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden

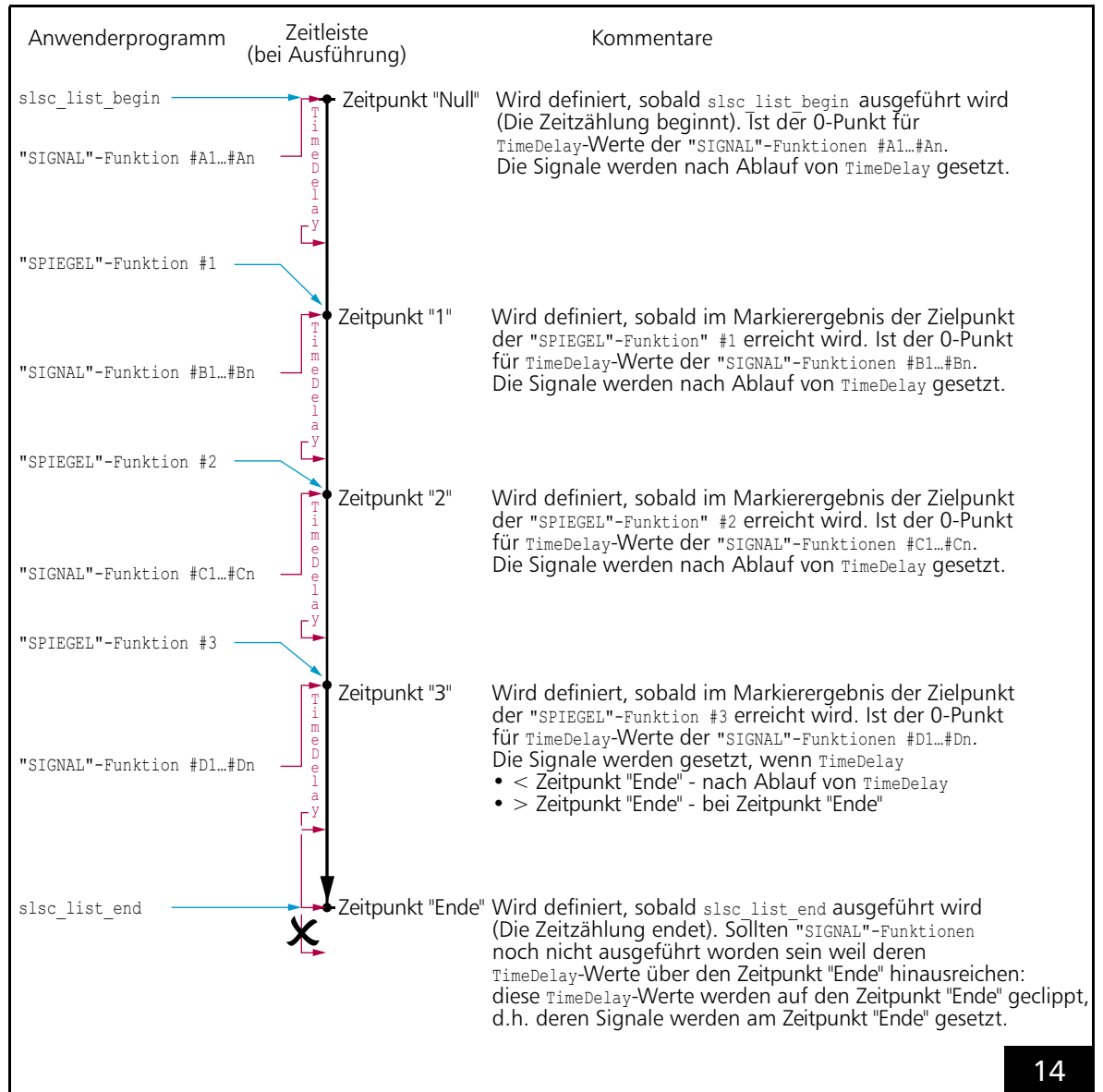
Benutzer können nicht exakt vorhersagen, wann ein (mit **"SIGNAL"-Funktionen**) angegebenes Output-Signal tatsächlich gesetzt wird (die Ausführungszeit-dauern der einzelnen **"SPIEGEL"-Funktionen** werden erst bei der Berechnung bestimmt). **Abbildung 14**, **Seite 46** beschreibt beispielhaft die Zusammenhänge.

"SIGNAL"-Funktion

- **slsc_list_set_free_variable**
- **slsc_list_set_laser_pulses**
- **slsc_list_suppress_spotdistance_control**
- **slsc_list_unsuppress_spotdistance_control**
- **slsc_list_write_analog_x**
- **slsc_list_write_digital_out**
- **slsc_list_write_digital_out_mask**

"SPIEGEL"-Funktion

- | | |
|---|--|
| • slsc_list_arc_abs | • slsc_list_dashed_arc_abs |
| • slsc_list_multi_para_arc_abs | • slsc_list_multi_para_dashed_arc_abs |
| • slsc_list_para_arc_abs | • slsc_list_para_dashed_arc_abs |
| • slsc_list_circle_2d_abs | • slsc_list_dashed_circle_2d_abs |
| • slsc_list_multi_para_circle_2d_abs | • slsc_list_multi_para_dashed_circle_2d_abs |
| • slsc_list_para_circle_2d_abs | • slsc_list_para_dashed_circle_2d_abs |
| • slsc_list_jump_abs | • slsc_list_jump_abs_min_time |
| • slsc_list_para_jump_abs | • slsc_list_para_jump_abs_min_time |
| • slsc_list_mark_abs | • slsc_list_dashed_mark_abs |
| • slsc_list_multi_para_mark_abs | • slsc_list_multi_para_dashed_mark_abs |
| • slsc_list_para_mark_abs | • slsc_list_para_dashed_mark_abs |
| • slsc_list_wait_with_laser_off | |
| • slsc_list_wait_with_laser_on | |



Die syncAXIS control-Instanz setzt Output-Signale bezogen auf vorausgehende Zeitpunkte.

2.8 Über das Logging in syncAXIS control

Zur erweiterten Problemanalyse können Sie den syncAXIS control Logging-Mechanismus verwenden. Dieser ist nicht nur ein effizientes Überwachungswerkzeug für den Benutzer, sondern ermöglicht auch eine bessere und einfachere Unterstützung durch SCANLAB bei Problemen.

Das Logging kann in der `syncAXISConfig.xml` eingestellt werden, siehe **Abbildung 15**.

Die Log-Datei⁽¹⁾ wird in einem Textdateiformat an der im Parameter `LogfilePath` angegebenen Stelle gespeichert.

Der eingestellte `Loglevel` definiert die Art der geloggten Meldungen:

- **Error**
 - Triggert gegebenenfalls **[ERROR]-Log-Dateizeilen**
 - Diese nennen Fehler, die das System tatsächlich stoppen und in einen Fehlerzustand versetzen
- **Warn**
 - Subsumiert **Error**
 - Triggert gegebenenfalls zusätzlich **[WARN]-Log-Dateizeilen**
 - Diese könnten ein potenzielles Problem beschreiben, das System wird jedoch *nicht* in einen Fehlerzustand versetzt. Beispiele: Überschreitungen der Dynamikgrenzwerte (in schweren Fällen kommt vom ACS-Subsystem zusätzlich ein Fehler) oder eine Markiergeschwindigkeits-Verminderung bei einer Nur-Verfahrtisch-Bewegung. Diese Mel-

(1) syncAXIS-DLL ≥ V1.5.0: nur noch 1 Log-Datei.

dungen weisen nicht unbedingt auf ein fehlerhaftes Verhalten hin. Daher ist es Aufgabe des Benutzers, auf diese angemessen zu reagieren.

- **Info**
 - Subsumiert **Warn** und **Error**
 - Triggert gegebenenfalls zusätzlich **[INFO]-Log-Dateizeilen**
 - Diese beinhalten alle für den Benutzer sichtbaren Meldungen. Sowohl die Kommunikation als auch **Job**-Informationen werden geloggt.

Hinweise

- Siehe auch **Kapitel 5 "Fehlercodes (Error Codes) bei slsc_ctrl_get_error, Log-Datei und Konsole", Seite 282**.
- Nachdem die erste **syncAXIS control-Instanz** aufgebaut ist, werden beim Aufbauen weiterer **syncAXIS control-Instanzen** die Logging-Einstellungen aus den jeweiligen `syncAXISConfig.xmls` *nicht* berücksichtigt. Daher:
 - Verwenden weitere **syncAXIS control-Instanzen** die gleichen Logging-Einstellungen wie die erste **syncAXIS control-Instanz**
 - Schreiben weitere **syncAXIS control-Instanzen** ihre Log-Dateizeilen in die Log-Datei der ersten **syncAXIS control-Instanz**
 - In diesen Log-Dateizeilen ist auch angegeben, von welchem Thread eine Nachricht stammt.
 - In einer Log-Dateizeile – die geschrieben wird, weil eine syncAXIS-DLL-Funktion mit einem Fehlercode $\neq 0$ zurückkehrt, siehe **Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279** – ist

```
<cfg:LogConfig>
  <cfg:LogfilePath>Log/Log.txt</cfg:LogfilePath>
  <cfg:Loglevel>Info</cfg:Loglevel>
  <cfg:EnableConsoleLogging>true</cfg:EnableConsoleLogging>
  <cfg:EnableFileLogging>true</cfg:EnableFileLogging>
  <cfg:MaxLogfileSize>26214400</cfg:MaxLogfileSize>
  <cfg:MaxBackupFileCount>0</cfg:MaxBackupFileCount>
</cfg:LogConfig>
```

15

Beispiel: Abschnitt aus einer `syncAXISConfig.xml`.

2.9 Über das automatische Steuern des Lasers durch syncAXIS control (“Automatische Lasersteuerung”)

Achtung!

Die “Automatische Lasersteuerung” kann nur dann sinnvoll eingesetzt werden, wenn Ihr Laser auf die Änderungen der Signalparameter reagieren kann (s.u.).

Eine **syncAXIS control-Instanz** kann so eingestellt werden, dass während der **Job**-Ausführung durch die RTC6 periodisch Werte erzeugt und ausgegeben (alle 10 μ s, für **SpotDistance** – siehe rechte Spalte – auch häufiger) werden.

Dieses Leistungsmerkmal ist primär dazu vorgesehen, um den eingesetzten Laser automatisch durch die **syncAXIS control-Instanz** steuern zu lassen. Es wird in syncAXIS control deswegen als “Automatische Lasersteuerung” bezeichnet.

Die mögliche “Art” der Ausgabe wird in syncAXIS control als “Channel” (Kanal) bezeichnet (im **RTC6-Handbuch** wird der Begriff “Signalparameter” verwendet).

Die tatsächliche “Art” der Ausgabe wird als “ActiveChannel” (aktiver Kanal) bezeichnet.

2.9.1 Aktivierung der “Automatischen Lasersteuerung”

Beim Initialisieren der **syncAXIS control-Instanz** wird die “Automatische Lasersteuerung” angeschaltet, wenn in der **syncAXISConfig.xml**

- mindestens ein Channel definiert (siehe **Kapitel 2.9.2 “Definition der Channel und ActiveChannel”, Seite 48**) und
- dieser Kanal als “ActiveChannel” eingetragen ist, siehe **Abbildung 16, Seite 49**.

2.9.2 Definition der Channel und ActiveChannel

Die Einstellungen zu Channel und ActiveChannel werden in der **syncAXISConfig.xml** vorgenommen und sind in **Abbildung 16, Seite 49**, gezeigt (zu Einstellungen, die über das Anwenderprogramm beeinflusst werden können, siehe **Abbildung 17, Seite 52**).

Definiert werden können bis zu 5 Channel:

- **AnalogOut1**
[VoltageFraction]
Entspricht dem RTC6-Signalparameter ANALOG OUT1. Das Signal wird am entsprechenden RTC6-Pin alle 10 μ s ausgegeben.
- **AnalogOut2**
[VoltageFraction]
Entspricht dem RTC6-Signalparameter ANALOG OUT2. Das Signal wird am entsprechenden RTC6-Pin alle 10 μ s ausgegeben.
- **PulseLength**
[s]
Entspricht dem RTC6-Signalparameter PulseLength. Das Signal wird – abhängig von den (in der **syncAXISConfig.xml** eingetragenen) Werten für **LaserMode** und **LaserPortCfg** – am entsprechenden RTC6-Pin alle 10 μ s geändert und ausgegeben.
- **HalfPeriod**
[s]
Entspricht dem RTC6-Signalparameter HalfPulsePeriod. Das Signal wird – abhängig von den (in der **syncAXISConfig.xml** eingetragenen) Werten für **LaserMode** und **LaserPortCfg** – am entsprechenden RTC6-Pin alle 10 μ s geändert und ausgegeben.
- **SpotDistance**
[mm]
Das Signal wird – abhängig von den (in der **syncAXISConfig.xml** eingetragenen) Werten für **LaserMode** und **LaserPortCfg** – am gleichen RTC6-Pin wie die **HalfPeriod** – ausgegeben. *Die Änderungsrate ist jedoch viel höher als die 10 μ s von HalfPeriod.*
SpotDistance basiert auf der Laserspotgeschwindigkeit. Diese wird an die RTC6 im 10 μ s-Takt gesendet und dort in passende Laserspotabstände umgerechnet. **Faktor Ir**, **Faktor Iv**, **Faktor Ip** wirken sich auf die Laserspotgeschwindigkeit aus.

<pre> <cfg:AutomaticLaserControl> <cfg:ActiveChannel> <cfg:Channel>AnalogOut2</cfg:Channel> <cfg:Channel>SpotDistance</cfg:Channel> </cfg:ActiveChannel> <cfg:AnalogOut1 DefaultOutput="0.5" Format="VoltageFraction"> <cfg:RadiusFactor Enabled="true"> <cfg:DataPoint Unit="mm" Radius="0" Factor="0.9" /> <cfg:DataPoint Unit="mm" Radius="27" Factor="1" /> </cfg:RadiusFactor> <cfg:VelocityFactor Enabled="true"> <cfg:DataPoint Unit="mm/s" Velocity="0" Factor="0.9" /> <cfg:DataPoint Unit="mm/s" Velocity="400" Factor="1.0" /> <cfg:DataPoint Unit="mm/s" Velocity="4000" Factor="2.0" /> </cfg:VelocityFactor> <cfg:Shift>0</cfg:Shift> </cfg:AnalogOut1> <cfg:AnalogOut2 DefaultOutput="0.5" Format="VoltageFraction"> <cfg:RadiusFactor Enabled="false"> <cfg:DataPoint Unit="mm" Radius="0" Factor="0.9" /> <cfg:DataPoint Unit="mm" Radius="27" Factor="1" /> </cfg:RadiusFactor> <cfg:VelocityFactor Enabled="false"> <cfg:DataPoint Unit="mm/s" Velocity="0" Factor="0.9" /> <cfg:DataPoint Unit="mm/s" Velocity="400" Factor="1.0" /> <cfg:DataPoint Unit="mm/s" Velocity="4000" Factor="2.0" /> </cfg:VelocityFactor> <cfg:Shift>0</cfg:Shift> </cfg:AnalogOut2> <cfg:PulseLength DefaultOutput="1e-5" Unit="s"> <cfg:RadiusFactor Enabled="true"> <cfg:DataPoint Unit="mm" Radius="0" Factor="0.9" /> <cfg:DataPoint Unit="mm" Radius="27" Factor="1" /> </cfg:RadiusFactor> <cfg:VelocityFactor Enabled="true"> <cfg:DataPoint Unit="mm/s" Velocity="0" Factor="0.9" /> <cfg:DataPoint Unit="mm/s" Velocity="400" Factor="1.0" /> <cfg:DataPoint Unit="mm/s" Velocity="4000" Factor="2.0" /> </cfg:VelocityFactor> <cfg:Shift>0</cfg:Shift> </cfg:PulseLength> <cfg:HalfPeriod DefaultOutput="1e-5" Unit="s"> <cfg:RadiusFactor Enabled="true"> <cfg:DataPoint Unit="mm" Radius="0" Factor="0.9" /> <cfg:DataPoint Unit="mm" Radius="27" Factor="1" /> </cfg:RadiusFactor> <cfg:VelocityFactor Enabled="true"> <cfg:DataPoint Unit="mm/s" Velocity="0" Factor="0.9" /> <cfg:DataPoint Unit="mm/s" Velocity="400" Factor="1.0" /> <cfg:DataPoint Unit="mm/s" Velocity="4000" Factor="2.0" /> </cfg:VelocityFactor> <cfg:Shift>0</cfg:Shift> </cfg:HalfPeriod> <cfg:SpotDistance DefaultOutput="0.005" Unit="mm"> <cfg:RadiusFactor Enabled="false"> <cfg:DataPoint Unit="mm" Radius="0" Factor="0.9" /> <cfg:DataPoint Unit="mm" Radius="26" Factor="0" /> <cfg:DataPoint Unit="mm" Radius="27" Factor="1.0" /> </cfg:RadiusFactor> <cfg:VelocityFactor Enabled="false"> <cfg:DataPoint Unit="mm/s" Velocity="0.0" Factor="0.9" /> <cfg:DataPoint Unit="mm/s" Velocity="400.0" Factor="1.0" /> <cfg:DataPoint Unit="mm/s" Velocity="800.0" Factor="1.2" /> </cfg:VelocityFactor> <cfg:Shift>1e5</cfg:Shift> </cfg:SpotDistance> </cfg:AutomaticLaserControl> </pre>	<p>"ActiveChannel" = zu verwendender Kanal [SpotDistance UND 1 Channel] ODER [1 Channel]</p> <p>Kanal AnalogOut1 Definitionen wirksam, wenn: <cfg:Channel>AnalogOut1</cfg:Channel></p> <p>Kanal AnalogOut2 Definitionen wirksam, wenn: <cfg:Channel>AnalogOut2</cfg:Channel></p> <p>Kanal PulseLength Definitionen wirksam, wenn: <cfg:Channel>PulseLength</cfg:Channel></p> <p>Kanal HalfPeriod Definitionen wirksam, wenn: <cfg:Channel>HalfPeriod</cfg:Channel></p> <p>Kanal SpotDistance Definitionen wirksam, wenn: <cfg:Channel>SpotDistance</cfg:Channel></p>
--	--

Legende

1. DefaultOutput-Attributwert
2. Iv soll (true) / nicht (false) verwendet werden.
3. Iv-Kennlinien-Datensätze (für 2.=true).
4. Iv soll (true) / nicht (false) verwendet werden.
5. Iv-Kennlinien-Datensätze (für 4.=true).
6. Shift (wahlweise, ab V1.2), siehe Shift.

syncAXISConfig.xml: Channels und ActiveChannels.

Für jeden Channel kann definiert werden:

- Dessen DefaultOutput-Wert
- Ob in die Ausgabe-Werte-Berechnung für diesen Channel der **Faktor Ir** = "Radius-Faktor" eingehen soll oder nicht. Der **Faktor Ir** wird einberechnet, wenn `enabled = true` ist. Dazu müssen aber auch die entsprechenden Kennlinien-Stützpunkte (**DataPoint**-Tags) eingetragen sein.
- Ob in die Ausgabe-Werte-Berechnung für diesen Channel der **Faktor Iv** = "Velocity-Factor" eingehen soll oder nicht. Der **Faktor Iv** wird einberechnet, wenn `enabled = true` ist. Dazu müssen die entsprechenden Kennlinien-Stützpunkte (**DataPoint**-Tags) eingetragen sein.
- Wenn der Kanal ein "ActiveChannel" ist: ob die Ausgabe-Werte dieses Channels zeitlich vor- oder nach hinten verlegt werden sollen. Zu diesem Zweck gibt es ab V1.2 **Shift**-Tags.

Zur Berechnung der Ausgabe-Werte siehe **Abbildung 17, Seite 52**, Output-Wert.

Von den (bis zu 5) *definierten* Channels können jedoch nur 1 oder 2 tatsächlich *genutzt* werden. Dazu muss er/müssen sie als ActiveChannel eingetragen sein:

- 1 beliebiger Channel
- 1 beliebiger Channel zusammen mit dem Channel **SpotDistance**


Hinweise

- Wenn kein ActiveChannel-Eintrag vorhanden ist, dann wird der Abschnitt AutomaticLaserControl beim Initialisieren der **syncAXIS control-Instanz** nicht ausgewertet. Die "Automatische Lasersteuerung" ist dann nicht aktiv (wird nicht genutzt).
- syncAXIS control bietet zwar ähnliche Funktionalitäten wie der RTC6-Befehl **set_auto_laser_control**, jedoch wird intern dieser RTC6-Befehl nicht genutzt (syncAXIS control hat eine davon völlig unabhängige, eigenständige Implementierung).
- Ist die "Automatische Lasersteuerung" aktiv mit **SpotDistance** als "ActiveChannel", dann wirken von **slsc_ctrl_set_laser_pulses** und **slsc_list_set_laser_pulses** deren Parameter **PulseLength** (d. h. die Pulslängen von Lasersignal LASER1 und LASER2 werden geändert), jedoch **HalfPeriod** *nicht*.

2.9.3 Über die Berechnung der ActiveChannel-Werte entlang einer Kontur

Welchen Wert die **syncAXIS control-Instanz** entlang einer Kontur genau für jeden ActiveChannel auf der RTC6 eingestellt, ist das Ergebnis einer Berechnung, siehe **Abbildung 17, Seite 52**.

Der DefaultOutput-Wert (siehe **Abbildung 16, Seite 49**) wird multipliziert mit bis (wenn diese Enabled = `true` sind) zu 3 Faktoren:

- Faktor **lr**
"Radius-Faktor". Kennlinienwert für den aktuellen Auslenkradius = Abstand vom Scan-Kopf-Arbeitsfeld-0,0.

- Faktor **lv**
"Velocity-Faktor". Kennlinienwert für die aktuelle Geschwindigkeit des Laserspots im Arbeitsfeld.

- Faktor **lp**
Rampen-Faktor. **slsc_list_para_enable** schaltet dessen Verarbeitung frei, sonst wird dieser *nicht* angewendet.
 - A) Mit **slsc_list_***-Funktionen
Faktor lp bleibt konstant. Entweder ist er gleich dem **ParaTargetDefault**-Wert oder dem Zielwert, der durch die vorherige **slsc_list_[multi_para/para]***-Funktion definiert ist.
 - B) Mit **slsc_list_para_***-Funktionen
Faktor lp wird entlang der Vektorlänge/Kreissegmentlänge *linear* variiert. Der Zielwert wird durch das Argument **ParaTarget** der aktuellen **slsc_list_para_***-Funktion definiert. Der Ausgangswert ist der zuletzt erreichte Wert (= **ParaTargetDefault**-Wert oder der Zielwert, der durch die vorherige **slsc_list_[multi_para/para]***-Funktion definiert ist).
 - C) Mit **slsc_list_[multi_para/para]***-Funktionen
Faktor lp wird entlang der Vektorlänge/Kreissegmentlänge *segmentweise linear* variiert.
(die Segmente komplizierterer **Rampen** werden mit **slsc_ParaSection** definiert). Ausgangswert des 1. **Rampenabschnitts** ist der zuletzt erreichte Wert (= **ParaTargetDefault**-Wert oder der Zielwert, der durch die vorherige **slsc_list_[multi_para/para]***-Funktion definiert ist).

Output-Wert alle 10 μ s
Jeder ActiveChannel individuell

=

DefaultOutput-Wert.

Dieser Wert ist in syncAXISConfig.xml für jeden Channel getrennt gesetzt, z.B. `<cfg:AnalogOut1 DefaultOutput="0.5" Format="..."`

×

Radius-Faktor l_r .

Hängt von der Einstellung in syncAXISConfig.xml ab: l_r wird nur verwendet, wenn beim Channel `<cfg:RadiusFaktor Enabled="true">` steht. Dann wird der zum Scan-Kopf-Auslenkradius passende l_r -Wert aus der Kennlinie entnommen (= `cfg:DataPoint`-Tags unter `cfg:RadiusFaktor`).

×

Geschwindigkeits-Faktor l_v .

Hängt von der Einstellung in syncAXISConfig.xml ab: l_v wird nur verwendet, wenn beim Channel `<cfg:VelocityFaktor Enabled="true">` steht. Dann wird der zur Laserspotgeschwindigkeit passende l_v -Wert aus der Kennlinie entnommen (= `cfg:DataPoint` tags unter `cfg:VelocityFaktor`).

×

Rampen-Faktor l_p .

Gesteuert über den Quell-Code: l_p wird nur verwendet nach `slsc_list_para_enable`. Der tatsächlich benutzte l_p -Wert wird aus bei `slsc_list_[multi_para/para]`-Funktionen definierten "Rampen" entnommen.

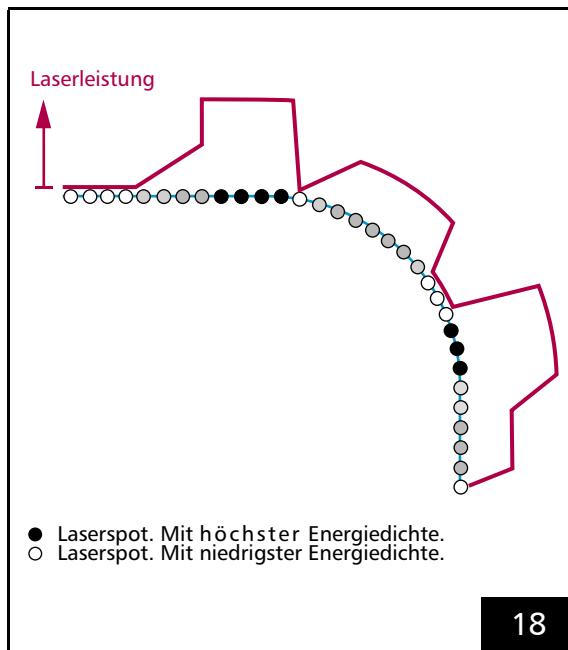
17

Berechnung des Output-Werts. Erfolgt für jeden ActiveChannel getrennt.

2.9.4 Über Rampen

Rampen werden im Quellcode mittels `slsc_list_[para/multi_para]*`-Funktionen definiert, siehe Abschnitt "Funktionen zum Definieren von Rampen (`slsc_list_[para/multi_para]*`-Funktionen)", Seite 92. D.h. die einzelnen Rampenpunkte werden mit `syncAXIS` control zusammen mit Positions-Koordinaten definiert.

Mit `syncAXIS` control können Rampen unterschiedlicher Profile modelliert werden, z.B. \sqcap , \sqcap , \sqcap , \sqcap . Sie werden unter anderem zur automatischen Variation der Laserleistung eingesetzt, wenn das Markiersubstrat entlang der Kurve unterschiedliche Lichtabsorption zeigt (d.h. das Werkstückmaterial besteht aus absorptionsempfindlicheren und absorptionsunempfindlicheren Bereichen), siehe Abbildung 18, Seite 53.

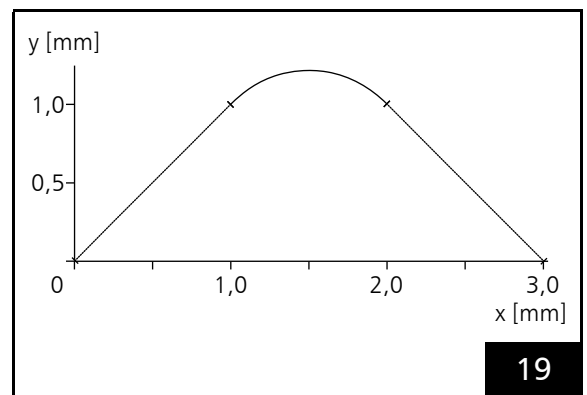


Variieren der Laserleistung.

Im Folgenden sind zwei konkrete Beispiele beschrieben:

- "Beispiel – lineare (einfache) Rampe", Seite 54
- "Beispiel – mehrteilige (komplexere) Rampe", Seite 57

Beide Beispiele verwenden das gleiche Markiermuster: Sprung auf Ausgangsposition – Gerade Linie – Kreisbogen – gerade Linie, siehe Abbildung 19, Seite 53 (Sprünge dort nicht dargestellt).



2D-Positionen der beiden Beispiele in diesem Abschnitt. Sprünge sind nicht dargestellt.

Beispiel – lineare (einfache) **Rampe**

Source-Code

- Siehe **Abbildung 20**.

Erläuterungen zum Source-Code

- Markiermuster: Sprung auf Ausgangsposition – Gerade Linie – Kreisbogen – gerade Linie, siehe **Abbildung 19, Seite 53** (Sprünge dort nicht dargestellt)
- Mit **slsc_list_para_enable**
 - wird die Verarbeitung von `TargetPara`-Argumenten freigeschaltet
 - `TargetParaDefault` wird = 1 gesetzt
- Mit **slsc_list_para_arc_abs** wird definiert
 - der Kreisbogen
 - eine lineare **Rampe**

Erläuterungen zum Simulationsergebnis

- Das Simulationsergebnis zeigt **Abbildung 21, Seite 56**
- Die **Rampe** beginnt am Kreisbogen
- Vor und am Beginn der **Rampe** ist der Ausgabe-Wert 0,5 (weil `DefaultOutput="0.5"` und `TargetParaDefault = 1` "keine Änderung" bedeutet)
- Der Ziel-Ausgabe-Endwert ist der zuletzt erreichte Wert \times **Faktor Ip** ($0,5 \times 0,5 = 0,25$)
- Während des Kreisbogens wird der Ziel-Ausgabe-Startwert 0,5 linear zum Ziel-Ausgabe-Endwert 0,25 variiert
- Andere Faktoren beeinflussen diese Variation nicht, da weder **Faktor Ir** noch **Faktor Iv** definiert sind
- Über das **Rampen**-Ende hinaus bleibt der zuletzt eingestellte Wert bestehen (0,25)

Einstellungen für die Simulation

Zur Simulation des Codes wird eine `syncAXISConfig.xml` mit den folgenden Einstellungen verwendet:

- `ActiveChannel = AnalogOut2`
- `Channel AnalogOut2`
 - `DefaultOutput="0.5"`
 - `RadiusFactor Enabled="false"`
(=es soll kein **Faktor Ir** verwendet werden)
 - `VelocityFactor Enabled="false"`
(=es soll kein **Faktor Iv** verwendet werden)
- Splines sind deaktiviert
(`slsc_SplineModes_Deactivated` oder über **slsc_cfg_set_trajectory_config**)
- Blending-Kurven sind deaktiviert
(`slsc_BlendModes_Deactivated` oder über **slsc_cfg_set_trajectory_config**)

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.
// Beispiel: Lineare (=einfache) Rampe mittels einem slsc_list_para_arc_abs.
// Es sind keine Blending-Kurven (blending curves) eingestellt.
// Pseudo-Code (nicht vollständig)

size_t JobID = 0;
slsc_list_begin(Handle, &JobID);
double TargetPosition_0[2] = { 0.0, 0.0 };
double TargetPosition_1[2] = { 1.0, 1.0 };
double TargetPosition_2[2] = { 1.025, 1.025 };
double TargetPosition_3[2] = { 2.0, 1.0 };
double TargetPosition_4[2] = { 3.0, 0.0 };

double TargetParaDefault[1] = { 1.0 };
double TargetPara_1[1] = { 0.5 };

slsc_list_jump_abs(Handle, TargetPosition_0);
slsc_list_para_enable(Handle, TargetParaDefault);
slsc_list_mark_abs(Handle, TargetPosition_1);
slsc_list_para_arc_abs(Handle, TargetPosition_2, TargetPosition_3, TargetPara_1);
slsc_list_mark_abs(Handle, TargetPosition_4);
slsc_list_jump_abs(Handle, TargetPosition_0);
slsc_list_end(Handle);
```

Beispiel-Code: lineare (einfache) **Rampe**

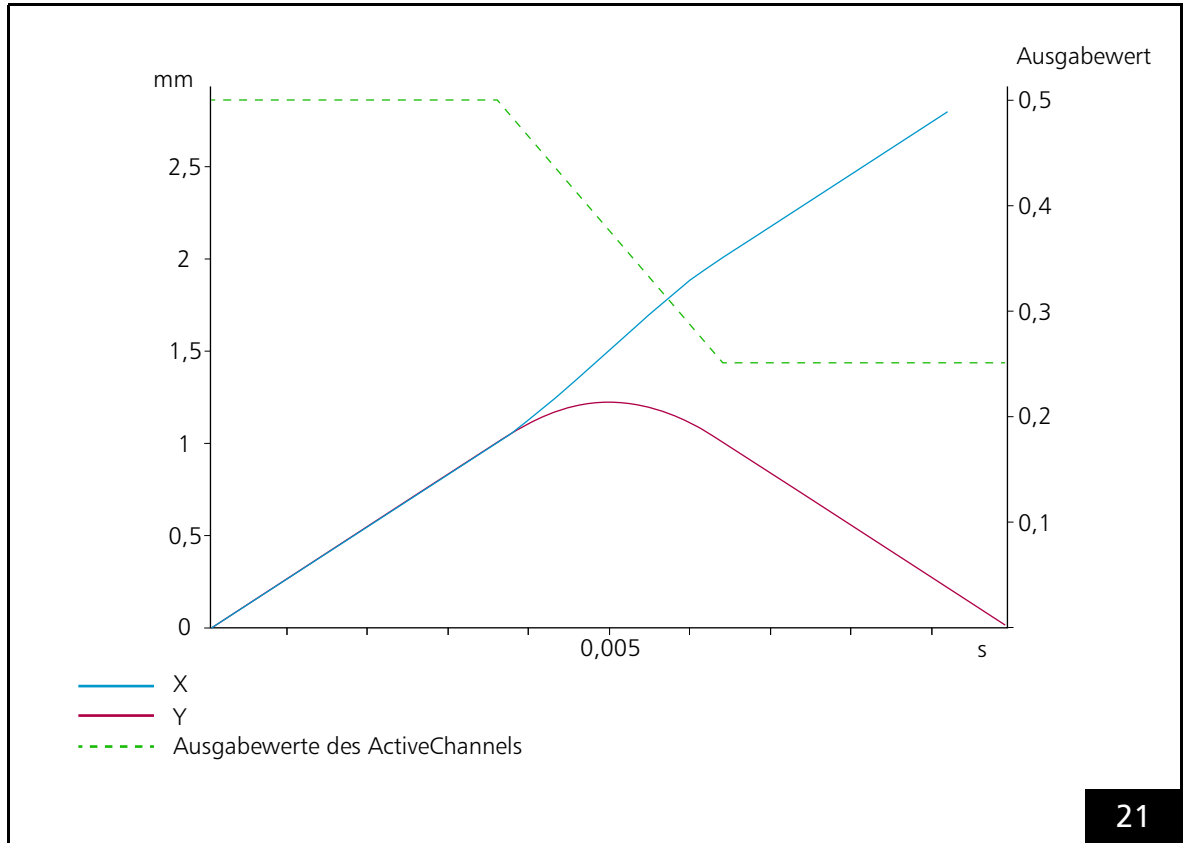


Diagramm zum **Abschnitt "Beispiel – lineare (einfache) Rampe", Seite 54.**

Simulationsergebnis: Zeitlicher Verlauf (Werte aus der **Trajektorienplanung**) der X- und Y-Positionen, sowie der linearen (einfachen) **Rampe**. ActiveChannel = AnalogOut2, kein **Faktor Ir**, kein **Faktor Iv**. Keine Splines, keine Blending-Kurven.

Rampen-Start-Wert: DefaultOutput \times TargetParaDefault ($0,5 \times 1$).

Rampen-End-Wert: [TargetPara_1 (= **Faktor Ip**) \times letzter erreichter Wert] ($0,5 \times 0,5$).

Beispiel – mehrteilige (komplexere) **Rampe**

Source-Code in **Abbildung 22** und Erläuterungen

- Markiermuster: Sprung auf Ausgangsposition – Gerade Linie – Kreisbogen – gerade Linie, siehe **Abbildung 19, Seite 53** (Sprünge dort nicht dargestellt)
- Mit **slsc_list_para_enable**
 - wird die Verarbeitung von TargetPara-Argumenten freigeschaltet
 - TargetParaDefault wird = 1 gesetzt
- Mit **slsc_list_multi_para_arc_abs** wird definiert
 - der Kreisbogen
 - eine **Rampe** aus 3 Abschnitten
 - ParaSection[0]...[2]

Einstellungen für die Simulation

Zur Simulation des Codes wird eine **syncAXISConfig.xml** mit den folgenden Einstellungen verwendet:

- ActiveChannel = AnalogOut2
- Channel AnalogOut2
 - DefaultOutput="0.5"
 - RadiusFactor Enabled="false" (=es soll kein **Faktor Ir** verwendet werden)
 - VelocityFactor Enabled="false" (=es soll kein **Faktor Iv** verwendet werden)
- Splines sind deaktiviert (**slsc_SplineModes_Deactivated** oder über **slsc_cfg_set_trajectory_config**)
- Blending-Kurven sind deaktiviert (**slsc_BlendModes_Deactivated** oder über **slsc_cfg_set_trajectory_config**)

Erläuterungen zum Simulationsergebnis

- Das Simulationsergebnis zeigt **Abbildung 23, Seite 59**.
- Die **Rampe** beginnt am Kreisbogen
- Vor und am Beginn der **Rampe** ist der Ausgabe-Wert 0,5 (weil DefaultOutput="0.5" und TargetParaDefault = 1 "keine Änderung" bedeutet)
- 1. **Rampenabschnitt**
 - Der Ziel-Ausgabe-Endwert (a) ist der DefaultOutput-Wert × **Faktor Ip@ParaSection[0]** ($0,5 \times 0,5 = 0,25$)
 - Dieser Wert soll nach 0,25 mm Markierstrecke erreicht sein
 - Dabei wird linear (abnehmend) variiert
- 2. **Rampenabschnitt**
 - Der Ziel-Ausgabe-Endwert (b) ist der DefaultOutput-Wert × **Faktor Ip@ParaSection[1]** ($0,5 \times 0,5 = 0,25$)
 - Dieser Wert soll nach 0,61(...) mm Markierstrecke erreicht sein
 - Der Ausgabewert bleibt konstant (da Startwert und End-Wert 0,25)
- 3. **Rampenabschnitt**
 - Der Ziel-Ausgabe-Endwert (c) ist der DefaultOutput-Wert × **Faktor Ip@ParaSection[2]** ($0,5 \times 1,0 = 0,5$)
 - Dieser Wert soll nach 0,25 mm Markierstrecke erreicht sein
 - Dabei wird linear (zunehmend) variiert
- Andere Faktoren beeinflussen diese Variation nicht, da weder **Faktor Ir** noch **Faktor Iv** definiert sind
- Über das **Rampen-Ende** hinaus bleibt der zuletzt eingestellte Wert bestehen (0,5)

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.
// Beispiel MultiPara über slsc_list_multi_para_arc_abs
// Es sind keine Blending-Kurven (blending curves) eingestellt.
// Pseudo-Code (nicht vollständig)

size_t JobID = 0;
slsc_list_begin(Handle, &JobID);
double TargetPosition_0[2] = { 0.0, 0.0 };
double TargetPosition_1[2] = { 1.0, 1.0 };
double TargetPosition_2[2] = { 1.025, 1.025 };
double TargetPosition_3[2] = { 2.0, 1.0 };
double TargetPosition_4[2] = { 3.0, 0.0 };

slsc_list_jump_abs(Handle, TargetPosition_0);
slsc_list_mark_abs(Handle, TargetPosition_1);
double TargetParaDefault[1] = { 1.0 };
slsc_list_para_enable(Handle, TargetParaDefault);

// Hier wird erstellt: Array vom Typ slsc_ParaSection mit Dimension 3.
slsc_ParaSection ParaSection[3];
ParaSection[0] = slsc_ParaSection{ 0.25, 0.5 };
ParaSection[1] = slsc_ParaSection{ 0.6186678697087737, 0.5 };
ParaSection[2] = slsc_ParaSection{ 0.25, 1.0 };

// Hier wird erstellt: Array vom Typ slsc_MultiParaTarget mit Dimension 1 (da ActiveChannel == 1).
slsc_MultiParaTarget MultiTargetPara[1];
MultiTargetPara[0].Targets = ParaSection;
MultiTargetPara[0].NumParaTargets = 3;
slsc_list_multi_para_arc_abs(Handle, TargetPosition_2, TargetPosition_3, MultiTargetPara);

slsc_list_mark_abs(Handle, TargetPosition_4);
slsc_list_jump_abs(Handle, TargetPosition_0);
slsc_list_end(Handle);
```

Beispiel-Code: mehrteilige (komplexere) **Rampe**

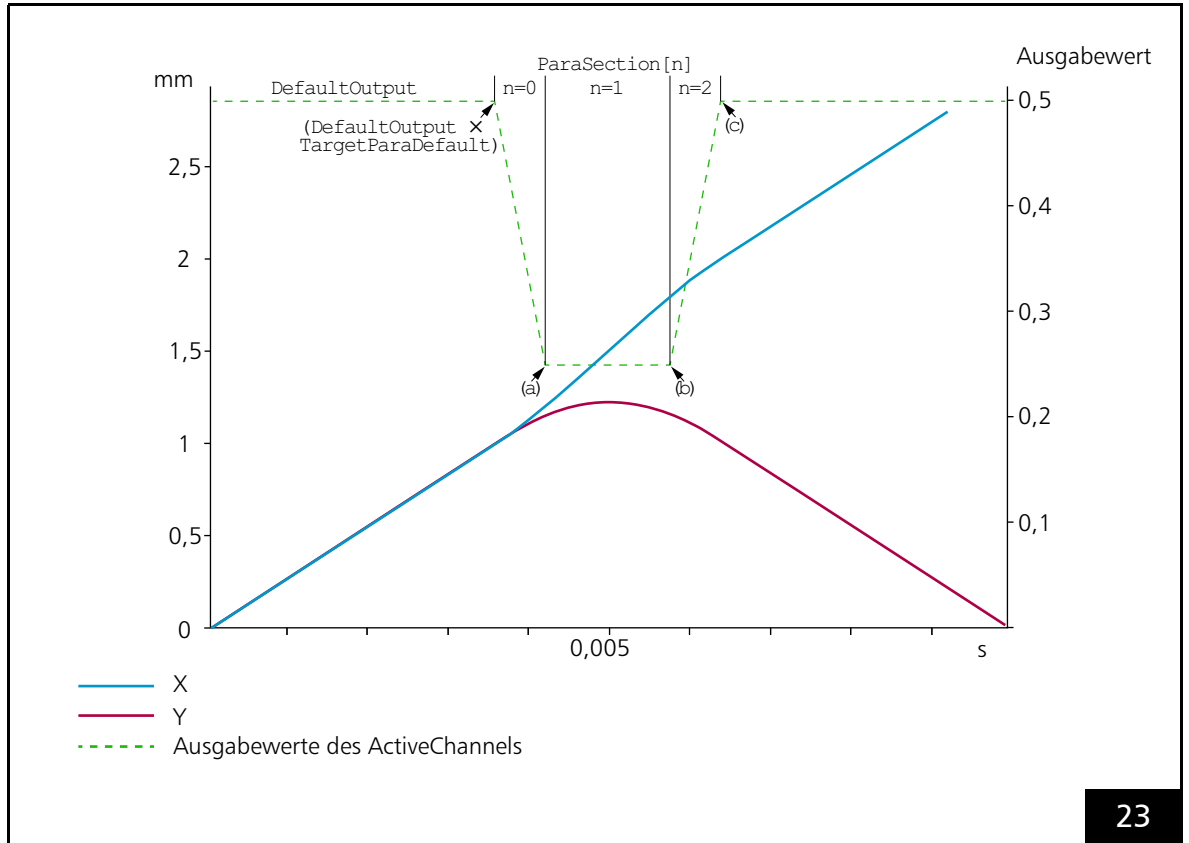


Diagramm zum **Abschnitt "Beispiel – mehrteilige (komplexere) Rampe"**, Seite 57.

Simulationsergebnis: Zeitlicher Verlauf (Werte aus der **Trajektorienplanung**) der X- und Y-Positionen, sowie der mehrteiligen (komplexeren) **Rampe**. ActiveChannel = AnalogOut2, kein **Faktor Ir**, kein **Faktor Iv**. Keine Splines, keine Blending-Kurven. Weitere Details siehe Text.

2.9.5 Über die “Konturabhängige Geschwindigkeitsberechnung”

Die “Konturabhängige Geschwindigkeitsberechnung” ist eine Funktionalität der “Automatischen Laser Steuerung”.

Die “Konturabhängige Geschwindigkeitsberechnung” setzt voraus, dass die “Automatische Lasersteuerung” in der `syncAXISConfig.xml` konfiguriert (mindestens 1 Channel ist definiert und als “ActiveChannel” eingetragen) und eingeschaltet (Initialisieren der `syncAXIS control-Instanz` mit dieser `syncAXISConfig.xml`) ist, siehe Kapitel 2.9.1 “Aktivierung der “Automatischen Lasersteuerung””, Seite 48.

Nach dem Initialisieren der `syncAXIS control-Instanz` mit `slsc_cfg_initialize_from_file` ist die “Konturabhängige Geschwindigkeitsberechnung” aber zunächst *nicht angeschaltet*.

Hauptanwendungsfall der “Konturabhängigen Geschwindigkeitsberechnung” ist, dass Benutzer beeinflussen können, auf welcher Berechnungsgrundlage der Energieeintrag entlang von Kurven erfolgen soll (und damit letztendlich das Markierungsergebnis).

Beispiel: es liegen Einbrände an Kurven (trotz äquidistanter Laserspots und konstanter Markiergeschwindigkeit) vor und manche davon dehnen sich in Bereiche aus, die eigentlich als Werkstück verwendet werden sollten, siehe [Abbildung 24, Seite 61](#).

Um die Gleichmäßigkeit der Signalausgaben (an allen “ActiveChannel”, z. B. Äquidistanz der Laserspotabstände mit `SpotDistance`) zu erreichen, berechnet die `syncAXIS control-Instanz` – ohne “Konturabhängige Geschwindigkeitsberechnung” – deren Ausgabe bezogen auf die Geschwindigkeit entlang der Konturlinien-Mitte. Diesen Fall illustriert [Abbildung 24, Seite 61](#).

Diese Berechnungsgrundlage (für diese Geschwindigkeit) können Benutzer mit den folgenden Funktionen ändern:

- `slsc_cfg_set_contour_dependent_speed_control_2d`
- `slsc_list_set_contour_dependent_speed_control_2d`

Über die Wahl geeigneter Parameter kann damit eingestellt werden, ob die Geschwindigkeit berechnet wird bezogen

- auf einen bestimmten Abstand zur Konturlinien-Mitte (`SpotRadius`) und
- außerdem, ob dieser Abstand
 - rechts von der Kontur (`Direction = +1`), siehe [Abbildung 25, Seite 62](#)
 - links von der Kontur (`Direction = -1`), siehe [Abbildung 26, Seite 63](#) sein soll.

Hinweise

- Die “Konturabhängige Geschwindigkeitsberechnung” wird ausgeschaltet durch:
 - `slsc_cfg_set_contour_dependent_speed_control_2d` mit `Direction=0` oder `SpotRadius=0`
 - `slsc_list_set_contour_dependent_speed_control_2d` mit `Direction=0` oder `SpotRadius=0`
- (Nur) für den ActiveChannel `SpotDistance` ist es mit diesen beiden Funktionen möglich, die Spot-Abstands-berechnung auf die Geschwindigkeit am Innenradius oder Außenradius (statt auf die Konturlinien-Mitte) zu beziehen und dann bei der `Job`-Ausführung die Laserspots häufiger als nur im 10 μ s-Takt tatsächlich auszugeben.
- Für alle ActiveChannel (auch `SpotDistance`) gilt:
 - wenn die “Konturabhängige Geschwindigkeitsberechnung” *angeschaltet* ist: **Faktor lv** bezieht sich auf die oben beschriebene Geschwindigkeit
 - wenn die “Konturabhängige Geschwindigkeitsberechnung” *ausgeschaltet* ist: **Faktor lv** bezieht sich auf die Geschwindigkeit des Laserspots im Arbeitsfeld

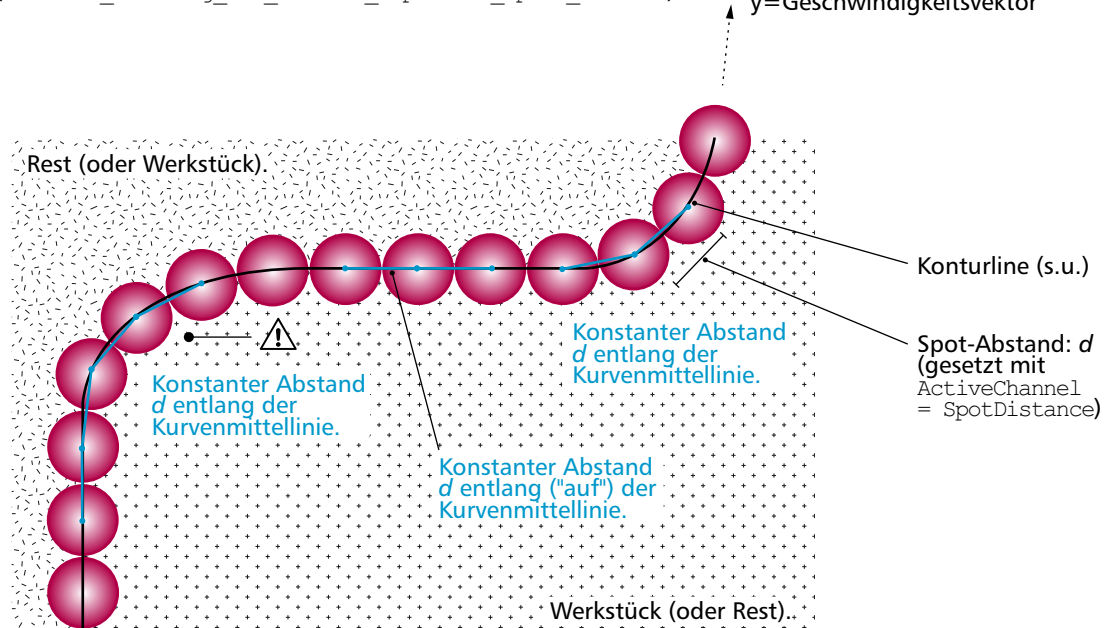
Automatische Laser Steuerung = AN:

- ActiveChannel = SpotDistance

Konturabhängige Geschwindigkeitsberechnung = AUS:

- Direction=0

(von slsc_list/cfg_set_contour_dependent_speed_control)



24

Die "Automatische Lasersteuerung" ist an, weil in der `syncAXISConfig.xml` als "ActiveChannel" (siehe [Abbildung 16, Seite 49](#)) `SpotDistance` eingestellt ist. `SpotDistance` stellt sicher, dass die Laserspots äquidistant sind: In diesem Beispiel ist die "Konturabhängige Geschwindigkeitsberechnung" aber *nicht angeschaltet* (Default-Einstellung, sowie durch `slsc_cfg_set_contour_dependent_speed_control_2d` oder `slsc_list_set_contour_dependent_speed_control_2d` mit `Direction=0` oder `SpotRadius=0`). Deswegen berechnet `syncAXIS control` die Gleichmäßigkeit der Laserspotabstände bezogen auf die Mitte der Konturlinie. Die Konturlinie muss (gilt für `syncAXIS control` \geq V0.11) aus mehreren `slsc_list_[multi_para/para]*`-Funktionen bestehen. Der Geschwindigkeitsvektor kommt aus der [Trajektorienplanung](#) und ergibt sich aus dem definierten Markiermuster.

Hinweis: Im hier dargestellten Fall können potenziell im mit "⚠" bezeichneten Bereich – trotz äquidistanter Laserspots und konstanter Markiergeschwindigkeit dort – Einbrüche zu beobachten sein (weswegen zur Behebung das Leistungsmerkmal "Konturabhängige Geschwindigkeitsberechnung" eingeführt wurde).

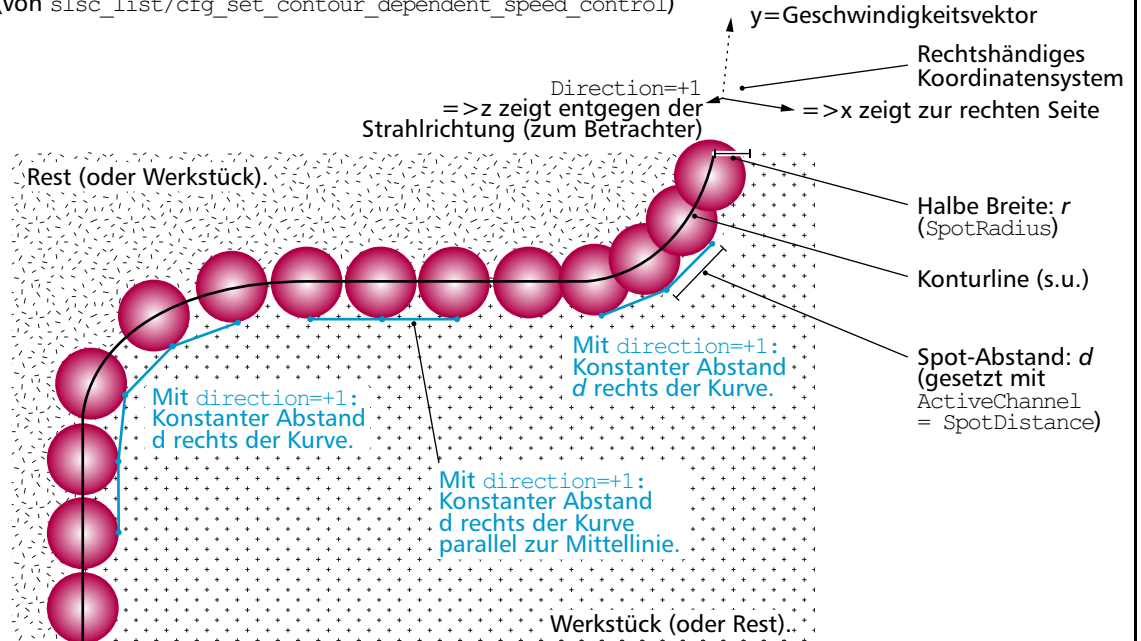
Automatische Laser Steuerung = AN:

- ActiveChannel = SpotDistance

Konturabhängige Geschwindigkeitsberechnung = AN:

- Direction=+1

(von slsc_list/cfg_set_contour_dependent_speed_control)



25

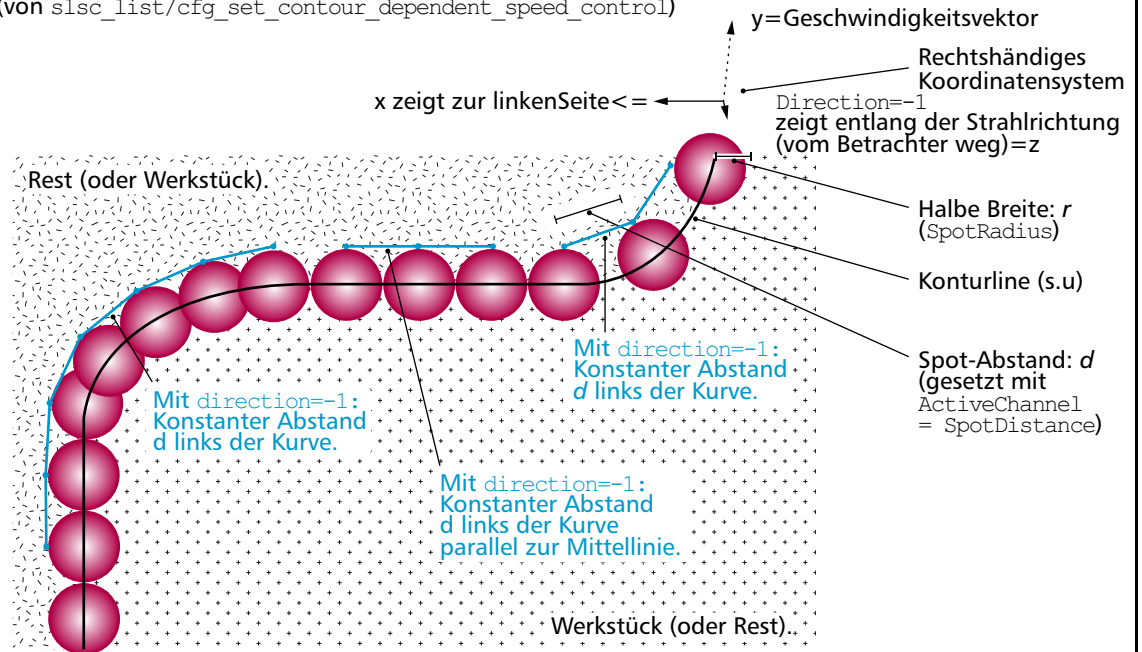
Die "Automatische Lasersteuerung" ist an, weil in der `syncAXISConfig.xml` als "ActiveChannel" (siehe [Abbildung 16, Seite 49](#)) `SpotDistance` eingestellt ist. `SpotDistance` stellt sicher, dass die Laserspots äquidistant sind: In diesem Beispiel ist die "Konturabhängige Geschwindigkeitsberechnung" *angeschaltet*, da bei `slsc_cfg_set_contour_dependent_speed_control_2d` oder `slsc_list_set_contour_dependent_speed_control_2d` `Direction=+1` angegeben ist. Deswegen berechnet `syncAXIS control` die Gleichmäßigkeit der Laserspotsabstände bezogen auf "Rechts" der Konturlinie. Die Konturlinie muss (gilt für `syncAXIS control` $\geq V0.11$) aus mehreren `slsc_list_[multi_para/para]*`-Funktionen bestehen. Der Geschwindigkeitsvektor kommt aus der [Trajektorienplanung](#) und ergibt sich aus dem definierten Markiermuster.

Automatische Laser Steuerung = AN:

- ActiveChannel = SpotDistance

Konturabhängige Geschwindigkeitsberechnung = AN:

- Direction=-1
(von slsc_list/cfg_set_contour_dependent_speed_control)



26

Die "Automatische Lasersteuerung" ist an, weil in der `syncAXISConfig.xml` als "ActiveChannel" (siehe [Abbildung 16, Seite 49](#)) `SpotDistance` eingestellt ist. `SpotDistance` stellt sicher, dass die Laserspots äquidistant sind: In diesem Beispiel ist die "Konturabhängige Geschwindigkeitsberechnung" *angeschaltet*, da bei `slsc_cfg_set_contour_dependent_speed_control_2d` oder `slsc_list_set_contour_dependent_speed_control_2d` `Direction=-1` angegeben ist. Deswegen berechnet `syncAXIS control` die Gleichmäßigkeit der Laserspotabstände bezogen auf "Links" der Konturlinie. Die Konturlinie muss (gilt für `syncAXIS control` $\geq V0.11$) aus mehreren `slsc_list_[multi_para/para]*`-Funktionen bestehen. Der Geschwindigkeitsvektor kommt aus der [Trajektorienplanung](#) und ergibt sich aus dem definierten Markiermuster.

2.10 Über **Heuristik** und Kennlinien für Geschwindigkeits- Verminderungen

Heuristik

- Definition inkl. Voraussetzung, **Segment**-Typen (inkl. Einschränkung nur auf **Sprung-Segmente**), siehe Glossareintrag **Heuristik**

Kennlinien für Geschwindigkeitsverminderungen

- Anwendungsfälle siehe `DynamicReductionFunction` (**Seite 473**)
- Nicht-Anwendungsfall siehe `DynamicReductionFunction` (**Seite 473**)
- Definitions-Ort: Child-Tags von `DynamicReductionFunctions`
 - Eine Kennlinie = ein `DynamicReductionFunction`-Tag. Ihre Stützpunkte = `DataPoint`-Child-Tags (x-Werte räumliche Länge, y-Werte Geschwindigkeit)
 - Beispiel siehe **XML-Abschnitt-Beispiel, Seite 473**
 - Zulässige Kennlinien-Anzahl siehe `DynamicReductionFunction`
 - Default-Kennlinie siehe `DynamicReductionFunction`
 - Zum Wechseln der Kennlinie (`slsc_cfg_select_heuristic`) siehe `DynamicReductionFunction`
 - Keine Kennlinie – keine **Heuristik** möglich (`DynamicReductionFunction` ist kein Pflicht-Tag)

2.11 Über das Arbeiten mit "Modulen"

Wie in [Abbildung 11, Seite 41](#) illustriert, definiert der Benutzer einen **Job** durch eine Serie von **Job-Funktionen** (`slsc_list_*`) als Eingabe für die **Trajektorienplanung** der syncAXIS-DLL. Deren Ergebnis wird in Form von RTC6-Mikrovektorbefehlen⁽¹⁾ in den RTC6-Listenspeicher geladen.

Nachdem die **Job**-Ausführung ausgelöst ist, laufen **Trajektorienplanung** und **Job**-Ausführung typischerweise parallel ab.

Bei einer sehr rechenintensive **Trajektorie** (z. B. im Code sind viele sehr kurze Vektoren⁽²⁾ bei hoher Markiergeschwindigkeit definiert) kann es dazu kommen, dass der RTC6-Listenspeicher schneller abgearbeitet wird, als neue Mikrovektoren berechnet werden können.

Sobald dann der RTC6-Listenspeicher vollständig abgearbeitet ist, wird die **Job**-Ausführung wegen **Pufferunterlaufs** abgebrochen (siehe auch [Kapitel 2.7.1 "Über die Puffer der syncAXIS control-Instanzen", Seite 42](#)) sowie XL SCAN in einen Fehlerzustand versetzt.

Mit `slsc_list_begin_module` kann dieses Problem umgangen werden. Im Simulationsmodus kann mit dieser Funktion "im Voraus" ein **Job** berechnet werden lassen. Das Ergebnis ("Modul") wird als "**Modul-Datei**" aufgezeichnet, siehe [Abschnitt "Modul-Datei", Seite 66](#).

Diese **Modul-Datei** kann dann zu einem späteren Zeitpunkt in anderen **Jobs** wiederverwendet werden: das "Abspielen des **Moduls**" erfolgt (ohne rechenintensive **Trajektorienplanung**) entweder mit

- `slsc_list_playback_module` (Parameter-Werte zu **Rampen** werden nicht angewendet) oder
- `slsc_list_para_playback_module` (Parameter-Werte zu **Rampen** werden angewendet, wenn ein `slsc_list_para_enable` vorausgeht)

Nach dem Einlesen der **Modul-Datei** stehen dem Anwenderprogramm die Positionsdaten zur Verfügung. Diese werden anschließend vor dem Verarbeitungsschritt "Bewegungsaufteilung" eingefügt, siehe [Abbildung 11, Seite 41](#).

Von diesem Schritt an ist die weitere Verarbeitung schnell. Deshalb kommt es nicht mehr zum **Pufferunterlauf**, selbst wenn die Eingangsdaten ursprünglich hoch aufgelöst waren.

Hinweise

- Die **Bewegungsaufteilung** (`FilterBandwidth`) erfolgt erst beim Abspielen des **Moduls** (`slsc_list_playback_module`).
- Siehe auch [Abschnitt "Modul-Abspiel-Verhalten", Seite 66](#).
- Die Dynamik der Soll-**Trajektorie** kann beim Abspielen des **Moduls** nicht geändert werden.
- Dennoch kann das **Modul** mit `slsc_list_set_rot_and_offset_2d` beliebig im Raum positioniert und gedreht werden.
- Aus Benutzersicht verhalten sich **Modul** (aufgezeichneter **Job**) und eine Programmierung aus entsprechenden **Job-Funktionen** (`slsc_list_*`) weitgehend gleich. Zu den Besonderheiten gerade beim Anfang und Ende des **Moduls** siehe `slsc_list_begin_module` und `slsc_list_playback_module`.
- Code-Beispiele:
 - Grundsätzlicher Ablauf mit **Modulen**, siehe [Abbildung 27, Seite 67](#)
 - **Modul** aufzeichnen, siehe [Abbildung 28, Seite 68](#)
 - **Modul** abspielen, siehe [Abbildung 29, Seite 69](#)
 - Siehe auch [Kapitel 10 "Anhang C: Anwendungshinweis – Texte markieren mittels Modulen", Seite 344](#)
 - Siehe auch [Kapitel 11 "Anhang D: Anwendungshinweis – Pufferunterlauf vermeiden mittels Modulen", Seite 347](#)

(1) Zur Berechnungszeit eines einzelnen RTC6-Mikrovektors kann keine allgemeine Aussage getroffen werden. Diese hängt u.a. stark von den eingestellten Parameter-Werten und der Komplexität der **Trajektorie** ab.

(2) Zum Beispiel mit `slsc_list_mark_abs`.

Modul-Datei

- Eigenschaften
 - Dateiendung *.slm
 - Binärdatei
 - Kann nicht in syncAXIS Viewer V1.5 geöffnet werden
 - Wurde aufgezeichnet durch **slsc_list_begin_module**
 - Wird abgespielt mit **slsc_list_playback_module**
 - Jedoch gilt:

Betriebsmodus in dem das Modul aufgezeichnet wurde	Zulässiger Betriebsmodus zum Abspielen des Moduls
StageOnly	StageOnly
ScannerOnly	ScannerOnly ScannerAndStage
ScannerAndStage	ScannerOnly ScannerAndStage

- Inhalt
 - Eine interne Versionsnummer (Kompatibilitätsinformation)
 - Positionsdaten in 10 μ s-Takten (Ergebnis der **Trajektorienplanung**) vor der **Bewegungsaufteilung**⁽¹⁾, siehe **Abbildung 11, Seite 41** samt Laserschaltzeitpunkten (diese aber ohne **LaserPreTriggerTime**- und **LaserSwitchOffsetTime**-Korrektur⁽²⁾)
 - Zur Abspielzeit kann deswegen die Dynamik der aufgezeichneten **Trajektorie** *nicht* mehr verändert werden
 - Parameter-Werte zu **Rampen**
 - Relative Auslösezeiten und Parameter-Werte (vom Benutzer angegeben) der **“SIGNAL“-Funktionen**⁽³⁾
 - Diverse Metadaten, z. B. zu Höchstgeschwindigkeiten

- Modul**-Abspiel-Verhalten

Das Verhalten der Konfigurations-Parameter-Werte beim Abspielen des **Moduls** zeigt **Kapitel 13.1 “xml-Struktur-Übersicht”, Seite 358**, d. h. ob/wie die Parameter-Werte aus

- dem **Modul** oder
- der abspielenden **syncAXIS control-Instanz** angewendet werden:

- Container

Kein Verhalten, da Container-Tag

- STANDARD

Standard-Verhalten: Der Parameter-Wert der abspielenden **syncAXIS control-Instanz** wird angewendet. Sie müssen also *nicht* jedes Mal ein neues **Modul** aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. Wenn ergänzende Informationen verfügbar sind:

STANDARD***

- NON-ST'D

Kein Standard-Verhalten: Der Parameter-Wert der abspielenden **syncAXIS control-Instanz** wird *nicht* angewendet.

Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues **Modul** aufzeichnen. Wenn ergänzende Informationen verfügbar sind:

NON-ST'D***

(1) Das ist die Soll-**Trajektorie** relativ zum Werkstück.

(2) Die Laserschaltzeitpunkte werden um die zur Abspielzeit gültigen Werte für **LaserPreTriggerTime** und **LaserSwitchOffsetTime** korrigiert.

(3) Siehe **Kapitel 2.7.2 “Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden”, Seite 45**.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

// slsc_cfg_initialize_copy verwenden, ein Modul aufzeichnen und abspielen.
// Quell-syncAXIS control-Instanz im Hardwaremodus initialisieren.
size_t HardwareHandle = 0;
const char* XmlConfigFileName = "HardwareMode_syncAXISConfig.xml";
slsc_cfg_initialize_from_file(&HardwareHandle, XmlConfigFileName);

// Ändere die Konfiguration der Quell-syncAXIS control-Instanz (ist im Hardwaremodus).
double NewMarkSpeed = 500.0;
slsc_cfg_set_mark_speed(HardwareHandle, NewMarkSpeed);

// Initialisiere die Ziel-syncAXIS control-Instanz im Simulationsmodus (mit der momentanen Konfiguration der
// Quell-syncAXIS control-Instanz (ist im Hardwaremodus)).
size_t SimulationHandle = 0;
slsc_cfg_initialize_copy(&SimulationHandle, HardwareHandle);

// Vorbereitung der Callback-Funktion, um das Ende der Modulaufzeichnung (= Ende der Jobplanung)
// in der Ziel-syncAXIS control-Instanz (ist im Simulationsmodus) zu kommunizieren.
slsc_ExecTimeCallback setModuleRecordingFinished = static_cast<slsc_ExecTimeCallback>([](size_t JobID, uint64_t,
Progress, double ExecTime, void* Context)
{
    bool* PtrToModuleRecordingFinished = static_cast<bool*>(Context);
    *PtrToModuleRecordingFinished = true;
});
bool ModuleRecordingFinished = false;
slsc_cfg_register_callback_job_end_planned(SimulationHandle, setModuleRecordingFinished,
&ModuleRecordingFinished);

// Zeichne eine Modul-Datei in der Ziel-syncAXIS control-Instanz (ist im Simulationsmodus) auf.
size_t JobID_Sim = 0;
const char* ModuleFileName = "NewModule.slm";
double[2] StartPosition = {0.0, 0.0};
slsc_list_begin_module(SimulationHandle, &JobID_Sim, StartPosition, ModuleFileName);
double Target[2] = {1.0, 2.0};
slsc_list_mark_abs(SimulationHandle, Target);
slsc_list_end(SimulationHandle);

// Warte bis Aufzeichnung beendet ist. Dann Ziel-syncAXIS control-Instanz (ist im Simulationsmodus) löschen.
while(!ModuleRecordingFinished)
{
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}
slsc_cfg_delete(SimulationHandle);

// Abspielen der aufgezeichneten Modul-Datei in der Quell-syncAXIS control-Instanz (ist im Hardwaremodus).
size_t JobID_HW = 0;
slsc_list_begin(HardwareHandle, &JobID_HW);
slsc_list_playback_module(HardwareHandle, ModuleFileName);
slsc_list_end();

// Den Job ausführen. Dann Quell-syncAXIS control-Instanz (ist im Hardwaremodus) löschen.
slsc_ExecState State;
slsc_ctrl_get_exec_state(HardwareHandle, &State);
while (State != slsc_ExecState_ReadyForExecution)
{
    slsc_ctrl_get_exec_state(HardwareHandle, &State);
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}
slsc_ctrl_start_execution(HardwareHandle);
while ((State != slsc_ExecState_Idle) && (State != slsc_ExecState_NotInitOrError))
{
    slsc_ctrl_get_exec_state(HardwareHandle, &State);
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}
slsc_cfg_delete(HardwareHandle);
```

Vereinfachte Codestruktur zum Arbeiten mit **Modulen** – Grundsätzlicher Ablauf (**slsc_cfg_initialize_copy**, ein **Modul** aufzeichnen und abspielen, einen **Job** ausführen).

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

// Ein Modul aufzeichnen.

// Die Quell-syncAXIS control-Instanz im Simulationsmodus initialisieren.
size_t SLHandle = 0;
const char* XmlConfigFileName = "InSimulationMode_syncAXISConfig.xml";
slsc_cfg_initialize_from_file(&SLHandle, XmlConfigFileName);

// Vorbereitung der Callback-Funktion, um das Ende der Modulaufzeichnung (= Ende der Jobplanung) zu kommunizieren.
slsc_ExecTimeCallback setModuleRecordingFinished = static_cast<slsc_ExecTimeCallback>([](size_t JobID, uint64_t
Progress, double ExecTime, void* Context)
{
    bool* PtrToModuleRecordingFinished = static_cast<bool*>(Context);
    *PtrToModuleRecordingFinished = true;
});
bool ModuleRecordingFinished = false;
slsc_cfg_register_callback_job_end_planned(SLHandle, setModuleRecordingFinished, &ModuleRecordingFinished);

// Modul-Aufzeichnung an der Startposition (0,0) beginnen.
size_t JobID = 0;
double StartPosition[2] = {0.0, 0.0};
const char* ModuleFileName = "NewModule.slm";
slsc_list_begin_module(SLHandle, &JobID, StartPosition, ModuleFileName);

// Definiere den aufzuzeichnenden Job mit einer Abfolge von Listenfunktionen.
// Beachten Sie, dass dieses Beispiel nur die Mechanik der Modul-Aufzeichnung veranschaulicht.
// Es macht im Allgemeinen nicht viel Sinn, ein Modul eines so kleinen Jobs aufzuzeichnen.
double Target1[2] = {0.0, 5.0};
double Target2[2] = {5.0, 5.0};
uint16_t DigOutVal = 2;
double DigOutDelay = 0.0;
slsc_list_jump_abs(SLHandle, Target1);
slsc_list_write_digital_out(SLHandle, DigOutVal, DigOutDelay);
slsc_list_mark_abs(SLHandle, Target2);
slsc_list_end(SLHandle);

// Warte bis die Modul-Aufzeichnung beendet ist. Dann syncAXIS control-Instanz löschen.
while(!ModuleRecordingFinished)
{
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}
slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Aufzeichnen eines Moduls.


```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

// Modul abspielen.

// Initialisiere syncAXIS control-Instanz.
size_t SLHandle = 0;
const char* XmlConfigFileName = "syncAXISConfig.xml";
slsc_cfg_initialize_from_file(&SLHandle, XmlConfigFileName);

// Definieren Sie den Job, in den Sie ein aufgezeichnetes Modul aufnehmen möchten. Stellen Sie sich vor, dass das
// Modul ein Markierungsmuster darstellt, das Sie nun auf einem regelmäßigen Raster mehrmals wiederholen möchten.
// Jedes Gitterelement soll auch um einen gewissen Betrag gegenüber dem ursprünglichen Muster gedreht werden.
size_t JobID = 0;
slsc_list_begin(SLHandle, &JobID);
int Nx = 15;
double DeltaX = 1.0;
int Ny = 10;
double DeltaY = 2.0;
const double Pi = 3.14159265359;
double DeltaAngle = 5.0 * Pi/180;
const char* ModuleFileName = "MyMarkingPattern.slm";
for (int i = 0; i < Nx; ++i)
{
    for (int j = 0; j < Ny; ++j)
    {
        double Angle = (i+j) * DeltaAngle;
        double Offset[2] = {i * DeltaX, j * DeltaY};
        slsc_list_set_rot_and_offset_2d(SLHandle, Angle, Offset);
        slsc_list_playback_module(SLHandle, ModuleFileName);
    }
}
slsc_list_end(SLHandle);

// Job ausführen. Dann syncAXIS control-Instanz löschen.
slsc_ExecState State;
slsc_ctrl_get_exec_state(SLHandle, &State);
while (State != slsc_ExecState_ReadyForExecution)
{
    slsc_ctrl_get_exec_state(SLHandle, &State);
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}
slsc_ctrl_start_execution(SLHandle);
while ((State != slsc_ExecState_Idle) && (State != slsc_ExecState_NotInitOrError))
{
    slsc_ctrl_get_exec_state(SLHandle, &State);
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}
slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Abspielen eines Moduls.

2.12 Über den Modus “Manuelle Positionierung”

- Um eine **syncAXIS control-Instanz** automatisch in den Modus “Manuelle Positionierung” zu versetzen⁽¹⁾ rufen Sie auf:
 - **slsc_ctrl_unfollow**
- Um den Modus “Manuelle Positionierung” zu beenden, rufen Sie auf:
 - **slsc_ctrl_follow**
- Der Modus “Manuelle Positionierung” ermöglicht es, die Positionen von Scan-Device und Verfahrtsch aus der **syncAXIS control-Instanz** heraus wie gewünscht einzustellen. Es ergibt sich eine Zeitersparnis, weil die **syncAXIS control-Instanz** dazu *nicht* abgebaut (dann mit einer externen Software, z. B. **ACS-Software** verfahren) und anschließend wiederaufgebaut werden muss.
- Im Modus “Manuelle Positionierung”:
 - bleibt die **Job-Queue** erhalten
 - ist der Aufruf von Job-Funktionen (**slsc_list_***) weiterhin möglich
 - werden Trajektorienberechnungen nicht unterbrochen
 - bleibt die RTC6 übernommen
 - kann der Verfahrtsch nicht nur von extern gesteuert werden, sondern auch aus der **syncAXIS control-Instanz** heraus⁽²⁾ (mit **slsc_ctrl_move_stage_abs**; das Scan-Device auch mit **slsc_ctrl_move_scanner_abs**)
- Allerdings können im Modus “Manuelle Positionierung” keine **Job**-Starts mit **slsc_ctrl_start_execution** getriggert werden
- Eine **syncAXIS-DLL-Funktion** kann im Modus “Manuelle Positionierung”
 - nicht
 - ausschließlich
 - auch
 erlaubt sein, siehe Kapitel 2.12.1 “Erlaubte/nicht erlaubte **syncAXIS control-Funktionen**”, Seite 71.
- Der Modus “Manuelle Positionierung” ist *kein* weiterer **Betriebsmodus** (siehe enum **slsc_OperationMode**). Daher kann er auch nicht in der **syncAXISConfig.xml** als initialer **Betriebsmodus** eingestellt werden.

Anwendungsfall: Überprüfung der Laserstrahlqualität im laufenden Betrieb

- Im Modus “Manuelle Positionierung” werden mit **slsc_ctrl_move_scanner_abs** und **slsc_ctrl_move_stage_abs** (die ähnlich zum RTC6-Befehl **goto_xy** sind) die Galvanometerscanner im Scan-Device und der Verfahrtsch voneinander unabhängig bewegt, damit der Laserstrahlfokus einen Sensor treffen kann. Dann kann der Laser direkt mit **slsc_ctrl_laser_signal_on** und **slsc_ctrl_laser_signal_off** an-/ausgeschaltet werden.

Verfahrtsch vorübergehend freigeben und Ziel-Verfahrtsch wechseln

- Siehe Kapitel 2.12.2 “Beispiel – Verfahrtsch vorübergehend freigeben und Ziel-Verfahrtsch wechseln”, Seite 74.

(1) Ab **syncAXIS control** \geq V1.1.0.

(2) Es muss also nicht die **ACS-Software** benutzt werden, um den Verfahrtsch an eine gewünschte Position zu verfahren.

2.12.1 Erlaubte/nicht erlaubte syncAXIS control-Funktionen

- Ist eine Funktion im Modus "Manuelle Positionierung" nicht erlaubt, dann ist beim Rückgabewert das Bit #11 gesetzt (NotAllowedInCurrentMode).

Funktion	Im Modus "Manuelle Positionierung" erlaubt?
slsc_cfg_acquire_stage (veraltet)	Nein.
slsc_cfg_delete	Ja.
slsc_cfg_delete_trajectory_config	Ja.
slsc_cfg_get_calculation_dynamics_jump_scan_device	Ja.
slsc_cfg_get_calculation_dynamics_mark_scan_device	Ja.
slsc_cfg_get_calculation_dynamics_stage	Ja.
slsc_cfg_get_dynamic_limits_scan_device	Ja.
slsc_cfg_get_dynamic_limits_stage	Ja.
slsc_cfg_get_dynamic_violation_reaction	Ja.
slsc_cfg_get_field_limits_scan_device	Ja.
slsc_cfg_get_field_limits_stage	Ja.
slsc_cfg_get_jump_time	Ja.
slsc_cfg_get_mode	Ja.
slsc_cfg_get_operation_status	Ja.
slsc_cfg_get_scan_device_dynamic_monitoring_level	Ja.
slsc_cfg_get_simulation_setting	Ja.
slsc_cfg_get_stage_dynamic_monitoring_level	Ja.
slsc_cfg_get_sync_axis_version	Ja.
slsc_cfg_get_trajectory_config	Ja.
slsc_cfg_initialize_copy	Nein.
slsc_cfg_initialize_from_file	Nein.
slsc_cfg_register_callback_job_end_planned	Ja.
slsc_cfg_register_callback_job_finished_executing	Ja.
slsc_cfg_register_callback_job_is_executing	Ja.
slsc_cfg_register_callback_job_loaded_enough	Ja.
slsc_cfg_register_callback_job_progress_planned	Ja.
slsc_cfg_register_callback_job_start_planned	Ja.
slsc_cfg_reinitialize	Ja.
slsc_cfg_reinitialize_from_file	Ja.
slsc_cfg_release_stage (veraltet)	Nein.
slsc_cfg_select_heuristic	Ja.
slsc_cfg_select_stage	Ja.
slsc_cfg_set_bandwidth	Ja.

Funktion (Forts.)	Im Modus "Manuelle Positionierung" erlaubt? (Forts.)
slsc_cfg_set_calculation_dynamics_jump_scan_device	Ja.
slsc_cfg_set_calculation_dynamics_mark_scan_device	Ja.
slsc_cfg_set_calculation_dynamics_stage	Ja.
slsc_cfg_set_contour_dependent_speed_control_2d	Ja.
slsc_cfg_set_dynamic_limits_scan_device	Ja.
slsc_cfg_set_dynamic_limits_stage	Ja.
slsc_cfg_set_dynamic_violation_reaction	Ja.
slsc_cfg_set_field_limits_scan_device	Ja.
slsc_cfg_set_field_limits_stage	Ja.
slsc_cfg_set_jump_speed	Ja.
slsc_cfg_set_list_handling_mode	Ja.
slsc_cfg_set_list_handling_mode_with_context	Ja.
slsc_cfg_set_mark_speed	Ja.
slsc_cfg_set_matrix_and_offset	Ja.
slsc_cfg_set_mode	Nein.
slsc_cfg_set_part_displacement	Ja.
slsc_cfg_set_rot_and_offset_2d	Ja.
slsc_cfg_set_scan_device_dynamic_monitoring_level	Ja.
slsc_cfg_set_simulation_setting	Nein.
slsc_cfg_set_stage_dynamic_monitoring_level	Ja.
slsc_cfg_set_trajectory_config	Ja.
slsc_ctrl_disable_laser	Ja.
slsc_ctrl_enable_laser	Ja.
slsc_ctrl_follow	Nur im Modus "Manuelle Positionierung" erlaubt.
slsc_ctrl_get_error	Ja.
slsc_ctrl_get_error_count	Ja.
slsc_ctrl_get_exec_state	Ja.
slsc_ctrl_get_free_variable	Ja.
slsc_ctrl_get_job_characteristic	Ja.
slsc_ctrl_get_scan_device_position	Ja.
slsc_ctrl_get_stage_position	Ja.
slsc_ctrl_get_syncaxis_simulation_filename	Ja.
slsc_ctrl_get_value	Ja.
slsc_ctrl_is_list_input_buffer_full	Ja.
slsc_ctrl_laser_signal_off	Nur im Modus "Manuelle Positionierung" erlaubt.
slsc_ctrl_laser_signal_on	Nur im Modus "Manuelle Positionierung" erlaubt.

Funktion (Forts.)	Im Modus "Manuelle Positionierung" erlaubt? (Forts.)
<code>slsc_ctrl_move_scanner_abs</code>	Nur im Modus "Manuelle Positionierung" erlaubt.
<code>slsc_ctrl_move_stage_abs</code>	Nur im Modus "Manuelle Positionierung" erlaubt.
<code>slsc_ctrl_refresh_correction_file</code>	Ja.
<code>slsc_ctrl_select_correction_file</code>	Ja.
<code>slsc_ctrl_set_laser_pulses</code>	Ja.
<code>slsc_ctrl_set_free_variable</code>	Ja.
<code>slsc_ctrl_start_execution</code>	Nein.
<code>slsc_ctrl_stop</code>	Ja.
<code>slsc_ctrl_stop_controlled</code>	Nein.
<code>slsc_ctrl_unfollow</code>	Nein.
<code>slsc_ctrl_write_analog_x</code>	Nur im Modus "Manuelle Positionierung" erlaubt.
<code>slsc_ctrl_write_digital_out</code>	Nur im Modus "Manuelle Positionierung" erlaubt.
<code>slsc_ctrl_write_digital_out_mask</code>	Nur im Modus "Manuelle Positionierung" erlaubt.
<code>slsc_list_*</code> (siehe Seite 109)	Ja.

2.12.2 Beispiel – Verfahrtsch vorübergehend freigeben und Ziel-Verfahrtsch wechseln

Achtung!

Um **slsc_cfg_select_stage** verwenden zu können, muss ein **Dongle** verwendet werden, der die Verwendung mehrerer Verfahrtsche erlaubt (Standard-Dongle nicht ausreichend)!













Sonst ist beim Rückgabewert Bit #31 gesetzt (**InvalidOrMissingDongle**).













Hinweise

- Für den Anwendungsfall "Verfahrtsch vorübergehend freigeben und (mit **slsc_cfg_select_stage**) den Ziel-Verfahrtsch wechseln" sind die folgenden syncAXIS-DLL-Funktionen veraltet und *nicht* mehr zu verwenden:
 - slsc_cfg_release_stage (veraltet)**
 - slsc_cfg_acquire_stage (veraltet)**

Betreiber von Systemen mit 2 Verfahrtschen können den Modus "Manuelle Positionierung" (**slsc_ctrl_unfollow/slsc_ctrl_follow**) und **slsc_cfg_select_stage** kombinieren.

Die nachfolgende Tabelle zeigt beispielhaft die Vorgänge im Detail.

Zeit	 *.exe^(a) läuft?	 *.exe ruft auf...	 syncAXIS control-Instanz vorhanden?	 syncAXIS control-Instanz im Modus "Manuelle Positionierung"?	 syncAXIS control-Instanz mit Queue?	 syncAXIS control-Instanz berechnet für Job 1?	 syncAXIS control-Instanz berechnet für Job 2?	 syncAXIS control-Instanz führt Job 1 aus?	 syncAXIS control-Instanz führt Job 2 aus?	 RTC6 übernommen? (b)	 Verfahrtsch "1" übernommen? (b)	 Verfahrtsch "2" übernommen? (b)
0	Nein.	–	–	–	–	–	–	–	–	Nein.	Nein.	Nein. ^{(c)(d)}
1	Ja.	–	Nein.	–	–	–	–	–	–	Nein.	Nein.	Nein.
2	Ja.	slsc_cfg_initialize_from_file^(e)	Nein.	–	–	–	–	–	–	Nein.	Nein.	Nein.
3	Ja.	–	Ja.	Nein.	Ja.	–	–	–	–	Ja.	Ja.	Nein.
4	Ja.	slsc_list_begin	Ja.	Nein.	Ja.	Nein.	Nein.	Nein.	Nein.	Ja.	Ja.	Nein.
5	Ja.	slsc_list_*	Ja.	Nein.	Ja.	Ja.	Nein.	Nein.	Nein.	Ja.	Ja.	Nein.
6	Ja.	slsc_list_**	Ja.	Nein.	Ja.	Ja.	Nein.	Nein.	Nein.	Ja.	Ja.	Nein.
7	Ja.	slsc_ctrl_get_exec_state^(f)	Ja.	Nein.	Ja.	Ja.	Nein.	Nein.	Nein.	Ja.	Ja.	Nein.
8	Ja.	slsc_ctrl_start_execution	Ja.	Nein.	Ja.	Ja.	Nein.	Nein.	Nein.	Ja.	Ja.	Nein.
9	Ja.	–	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.
10	Ja.	slsc_list_end	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.
11	Ja.	–	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.
12	Ja.	slsc_list_begin	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.
13	Ja.	slsc_list_*	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.
14	Ja.	slsc_list_**	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.
15	Ja.	–	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.
16	Ja.	–	Ja.	Nein.	Ja.	Nein. ^(g)	Ja.	Ja.	Nein.	Ja.	Ja.	Nein.
17	Ja.	–	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.	Ja.	Nein.
18	Ja.	slsc_ctrl_get_exec_state	Ja.	Nein.	Ja.	Nein.	Ja.	Nein. ^(h)	Nein.	Ja.	Ja.	Nein.
19	Ja.	slsc_ctrl_get_exec_state^(f)	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Ja.	Nein.

Zeit	 *exe ^(a) läuft?	 *exe ruft auf...	 syncAXIS control-Instanz vorhanden?	 syncAXIS control-Instanz im Modus "Manuelle Positionierung"?	 syncAXIS control-Instanz mit Queue?	 syncAXIS control-Instanz berechnet für Job 1?	 syncAXIS control-Instanz berechnet für Job 2?	 syncAXIS control-Instanz führt Job 1 aus?	 syncAXIS control-Instanz führt Job 2 aus?	 RTC6 übernommen? ^(b)	 Verfahr-tisch "1" über-nommen? ^(b)	 Verfahr-tisch "2" über-nommen? ^(b)
20	Ja.	–	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Ja.	Nein.
21	Ja.	slsc_ctrl_unfollow	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Ja.	Nein.
22	Ja.	–	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein. ^(d)	Nein.
23	Ja.	slsc_ctrl_move_scanner_abs	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein. ^(d)	Nein.
24	Ja.	slsc_ctrl_move_stage_abs ^(f)	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein. ^(d)	Nein.
25	Ja.	–	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein. ^(d)	Nein.
26	Ja.	slsc_cfg_select_stage(Stage = 2) ^(f)	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein. ^(d)	Nein.
27	Ja.	–	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein.	Nein. ^(d)
28	Ja.	slsc_ctrl_move_scanner_abs	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein.	Nein. ^(d)
29	Ja.	slsc_ctrl_move_stage_abs ^(k)	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein.	Nein. ^(d)
30	Ja.	–	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein.	Nein. ^(d)
31	Ja.	slsc_ctrl_follow	Ja.	Ja.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein.	Nein. ^(d)
32	Ja.	–	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein.	Ja.
33	Ja.	slsc_ctrl_start_execution	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Nein.	Ja.	Nein.	Ja.
34	Ja.	–	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.
35	Ja.	slsc_list_end	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.
36	Ja.	–	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.
37	Ja.	–	Ja.	Nein.	Ja.	Nein.	Ja.	Nein.	Ja.	Ja.	Nein.	Ja.
38	Ja.	–	Ja.	Nein.	Ja.	Nein.	Nein. ^(f)	Nein.	Ja.	Ja.	Nein.	Ja.
39	Ja.	–	Ja.	Nein.	Ja.	Nein.	Nein.	Nein.	Ja.	Ja.	Nein.	Ja.
40	Ja.	–	Ja.	Nein.	Ja.	Nein.	Nein.	Nein.	Ja.	Ja.	Nein.	Ja.
41	Ja.	slsc_ctrl_get_exec_state	Ja.	Nein.	Ja.	Nein.	Nein.	Nein.	Nein. ^(m)	Ja.	Nein.	Ja.
42	Ja.	–	Ja.	Nein.	Ja.	Nein.	Nein.	Nein.	Nein.	Ja.	Nein.	Ja.
43	Ja.	slsc_cfg_delete	Ja.	Nein.	Ja.	Nein.	Nein.	Nein.	Nein.	Ja.	Nein.	Ja.
44	Ja.	–	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.
45	Nein.	–	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.	Nein.

(a) syncAXIS-DLL-basiertes Anwenderprogramm, z.B. eine GUI.

(b) Durch die **syncAXIS control-Instanz**.

(c) Verfahr-tisch "2" kann durch ein "anderes" Anwenderprogramm übernommen sein.

(d) Dieser Verfahr-tisch kann durch die syncAXIS-DLL positioniert werden. Hierbei wird ein **ACS**-Point-to-Point-Motion-Befehl via TCP/IP an den **ACS Motion-Controller** gesendet. Dies kann für Vorbereitungen zur späteren Werkstückbearbeitung z.B. einrichten, bewegen, Werkstück zuführen, Bildgebung (Imaging), Vermessungen, Positionierungen verwendet werden.

(e) Hier wird angenommen, dass in der **syncAXISConfig.xml** "Verfahr-tisch 1" und sowie **Betriebsmodus** "ScannerAndStage" eingetragen sind.

(f) **slsc_ExecState_ReadyForExecution**.

(g) Event **job_end_planned**, siehe **Abbildung 12**.

(h) Event **job_finished_executing**, siehe **Abbildung 12**.

(i) Sendet einen **ACS**-Point-to-Point-Motion-Befehl für Verfahr-tisch 1 via TCP/IP an den **ACS Motion-Controller**.

(j)  Erfordert einen entsprechend konfigurierten **Dongle**!

(k) Sendet einen **ACS**-Point-to-Point-Motion-Befehl für Verfahr-tisch 2 via TCP/IP an den **ACS Motion-Controller**.

(l) Event **job_end_planned**, siehe **Abbildung 12**.

(m) Event **job_finished_executing**, siehe **Abbildung 12**.

3 In der API bereitgestellte Funktionen

3.1 Funktionale Übersicht

In diesem Kapitel:

- Konfigurations-Funktionen (`slsc_cfg_*`), Seite 76
- Job-Funktionen (`slsc_list_*`), Seite 85
- Kontroll-Funktionen (`slsc_ctrl_*`), Seite 96
- Utility-Funktionen (`slsc_util_*`), Seite 101

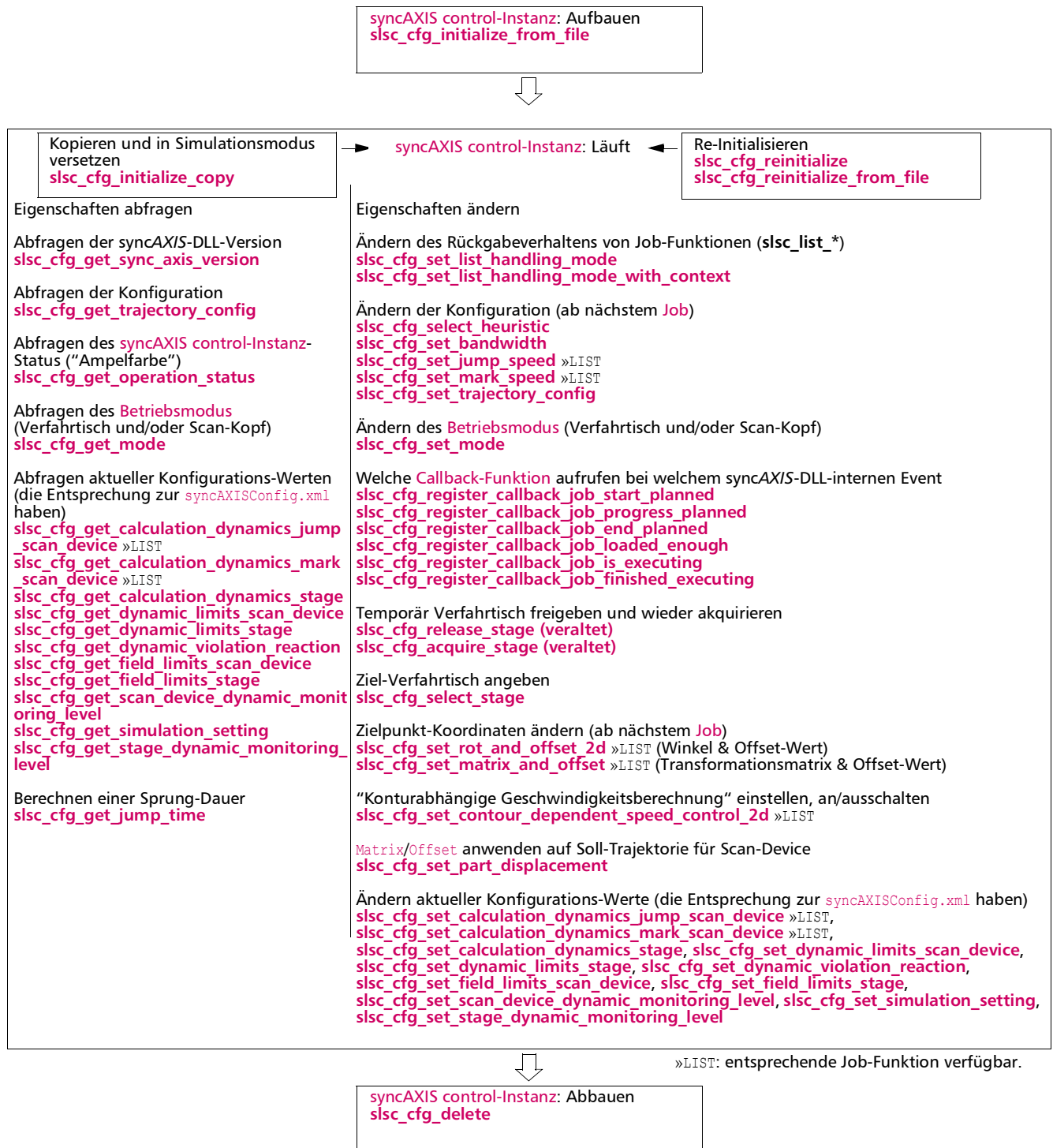
3.1.1 Konfigurations-Funktionen (`slsc_cfg_*`)

Konfigurations-Funktionen (Präfix `slsc_cfg_`) erlauben Aufbau/Abbau/Reinitialisierung von **syncAXIS control-Instanzen**, sowie Abfragen und Setzen ihrer Eigenschaften ("Konfiguration").

Außerdem kann der Verfahrtsch temporär freigeben (z. B. damit er "extern" angesteuert werden kann) und anschließend wieder übernommen werden. Wahlweise kann während der temporären Freigabe aber auch ein anderer Ziel-Verfahrtsch angegeben werden (es wird dann dieser anschließend übernommen). Auf diese Weise kann abwechselnd mit mehreren Verfahrtsch gearbeitet werden. Zu weiteren Informationen siehe **Kapitel 2.12.2 "Beispiel – Verfahrtsch vorübergehend freigeben und Ziel-Verfahrtsch wechseln"**, Seite 74.

Einen grafischen Überblick gibt **Abbildung 30**, Seite 77.

Konfigurations-Funktionen (slsc_cfg_*)



syncAXIS control-Instanz-bezogene Funktionen

- **slsc_cfg_acquire_stage (veraltet)** veraltet
- **slsc_cfg_delete**
- **slsc_cfg_get_calculation_dynamics_jump_scan_device**
- **slsc_cfg_get_calculation_dynamics_mark_scan_device**
- **slsc_cfg_get_calculation_dynamics_stage**
- **slsc_cfg_get_dynamic_limits_scan_device**
- **slsc_cfg_get_dynamic_limits_stage**
- **slsc_cfg_get_dynamic_violation_reaction**
- **slsc_cfg_get_field_limits_scan_device**
- **slsc_cfg_get_field_limits_stage**
- **slsc_cfg_get_jump_time**
- **slsc_cfg_get_mode**
- **slsc_cfg_get_operation_status**
- **slsc_cfg_get_scan_device_dynamic_monitoring_level**
- **slsc_cfg_get_stage_dynamic_monitoring_level**
- **slsc_cfg_get_sync_axis_version**
- **slsc_cfg_get_trajectory_config**
- **slsc_cfg_initialize_copy**
- **slsc_cfg_initialize_from_file**
- **slsc_cfg_reinitialize**
- **slsc_cfg_reinitialize_from_file**
- **slsc_cfg_release_stage (veraltet)** veraltet
- **slsc_cfg_select_stage**
- **slsc_cfg_set_list_handling_mode**
- **slsc_cfg_set_list_handling_mode_with_context**
- **slsc_cfg_set_matrix_and_offset**
- **slsc_cfg_set_mode**
- **slsc_cfg_set_trajectory_config**



Vorsicht!

Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! Berücksichtigen Sie im Sicherheitskonzept Ihrer Anlagen-Steuerung, dass die Lasersteuersignale der RTC durch **slsc_cfg_initialize_from_file** und **slsc_ctrl_enable_laser** freigegeben werden.

Um die syncAXIS-DLL zu nutzen, muss am Anfang des Anwenderprogramms eine Funktion zum Aufbau einer **syncAXIS control-Instanz** stehen. Das erfolgt mit **slsc_cfg_initialize_from_file** unter Angabe einer validen XML-Konfigurationsdatei⁽¹⁾⁽²⁾.

Dabei wird der **syncAXIS control-Instanz** ein eindeutiges **Handle** zugewiesen, über das sie angesprochen werden kann (fast alle Funktionen erfordern die Angabe eines **Handle**-Werts). Weiterhin wird die Hardware übernommen (RTC6-Karte und Verfahrtsch). Zu weiteren Informationen siehe Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen", Seite 26.

Eine **syncAXIS control-Instanz** kann entweder mit **slsc_cfg_reinitialize_from_file** (basierend auf der **syncAXISConfig.xml**) oder **slsc_cfg_reinitialize** (basierend auf dem momentanen Konfigurationszustand) reinitialisiert werden. Dabei bleibt der **Handle**-Wert unverändert.

Der gegenwärtige Status einer **syncAXIS control-Instanz** ("Ampelfarbe") wird mit **slsc_cfg_get_operation_status** abgefragt.

Um die Konfigurationswerte der **Trajektorienplanung** abzufragen wird **slsc_cfg_get_trajectory_config** verwendet (zum Ändern dieser siehe Abschnitt "Funktionen zur Konfigurationsänderung der bestehenden syncAXIS control-Instanz", Seite 80).

(1) xsd-Datei (XML Schema Definition; **syncAXIS_1_8.xsd**) und XML-Konfigurationsdatei-Vorlagen werden mitgeliefert.

(2) In der **syncAXISConfig.xml** sind einige **syncAXIS control-Instanz**-Konfigurations-Parameter passend für die jeweilige Anlage oder Applikation einzustellen. Die Beschreibung dieser Parameter finden Sie als Inline-Kommentare in der **syncAXIS_1_8.xsd** sowie in Kapitel 13 "Anhang F: Referenz der **syncAXISConfig.xml**-Tags", Seite 358. Einige der Parameter-Werte können über syncAXIS-DLL-Funktionen zur Laufzeit im Anwenderprogramm geändert werden.

Um momentane Werte der **syncAXIS control**-Instanz abzufragen, für die es Entsprechungen/dezidierte Tags in der **syncAXISConfig.xml** gibt, stehen zur Verfügung:

- **slsc_cfg_get_calculation_dynamics_jump_scan_device**
- **slsc_cfg_get_calculation_dynamics_mark_scan_device**
- **slsc_cfg_get_calculation_dynamics_stage**
- **slsc_cfg_get_dynamic_limits_scan_device**
- **slsc_cfg_get_dynamic_limits_stage**
- **slsc_cfg_get_dynamic_violation_reaction**
- **slsc_cfg_get_field_limits_scan_device**
- **slsc_cfg_get_field_limits_stage**
- **slsc_cfg_get_scan_device_dynamic_monitoring_level**
- **slsc_cfg_get_stage_dynamic_monitoring_level**

Die Version der **syncAXIS**-DLL kann mit **slsc_cfg_get_sync_axis_version** abgefragt werden.

slsc_cfg_set_mode setzt den Betriebsmodus ("StageOnly"/"ScannerOnly"/"ScannerAndStage"; siehe enum **slsc_OperationMode**), in dem die **syncAXIS control**-Instanz arbeiten soll. Die **syncAXIS control**-Instanz wird dabei reinitialisiert!

slsc_cfg_initialize_copy versetzt die Kopie einer **syncAXIS control**-Instanz in den Simulationsmodus ohne deren Konfiguration zu verändern.

slsc_cfg_initialize_copy kann z.B. im Kontext der Aufzeichnung von Modul-Dateien benutzt werden. Dazu wird **slsc_cfg_initialize_copy** vor **slsc_list_begin_module** (diese Funktion setzt den Simulationsmodus voraus) aufgerufen, siehe Abschnitt "Funktionen für "Module"", Seite 95.

Der aktuell eingestellte Betriebsmodus der **syncAXIS control**-Instanz wird mit **slsc_cfg_get_mode** abgefragt.

Zu **slsc_cfg_select_stage** siehe Kapitel 2.12.2 "Beispiel – Verfahrtsch vorübergehend freigeben und Ziel-Verfahrtsch wechseln", Seite 74. Beachten Sie, dass für diesen Anwendungsfall die veralteten **slsc_cfg_release_stage** (veraltet)/**slsc_cfg_acquire_stage** (veraltet) nicht mehr verwendet werden sollten.

Mit **slsc_cfg_get_simulation_setting** kann die aktuelle **Simulationseinstellung** abgefragt werden, siehe auch Kapitel 2.5 "Über den **syncAXIS control** Simulationsmodus", Seite 31. Zum Ändern ist **slsc_cfg_set_simulation_setting** verfügbar. Die **syncAXIS control**-Instanz wird dabei reinitialisiert!

Mit **slsc_cfg_set_list_handling_mode** kann das Rückgabeverhalten der Job-Funktionen (**slsc_list_***) geändert werden. Bei **slsc_cfg_set_list_handling_mode_with_context** kann zusätzlich ein Kontext angegeben werden.

Um Zielpunkt-Koordinaten (siehe Seite 268) von **slsc_list_arc_abs**, **slsc_list_circle_2d_abs**, **slsc_list_jump_abs**, **slsc_list_mark_abs** und deren entsprechenden **slsc_list_[para/multi_para]*** Funktionen für eine **syncAXIS control**-Instanz ändern zu können, stehen **slsc_cfg_set_rot_and_offset_2d** und **slsc_cfg_set_matrix_and_offset** zur Verfügung. Bei **slsc_cfg_set_rot_and_offset_2d** kann ein Winkel und Offset-Wert angegeben werden, bei **slsc_cfg_set_matrix_and_offset** eine Transformationsmatrix und Offset-Wert. Für beide gibt es entsprechende Job-Funktionen (**slsc_list_***), **slsc_list_set_rot_and_offset_2d** und **slsc_list_set_matrix_and_offset**, siehe auch Abschnitt "Funktionen zum Ändern der Zielpunkt-Koordinaten", Seite 92.

Die "Konturabhängige Geschwindigkeitsberechnung" wird mit **slsc_cfg_set_contour_dependent_speed_control_2d** eingestellt, sowie an- und ausgeschaltet. Zu Voraussetzungen und weiteren Informationen siehe Kapitel 2.9.5 "Über die "Konturabhängige Geschwindigkeitsberechnung"", Seite 60. Für **slsc_cfg_set_contour_dependent_speed_control_2d** gibt es die entsprechende Job-Funktion (**slsc_list_***) **slsc_list_set_contour_dependent_speed_control_2d**.

Mit **slsc_cfg_get_jump_time** kann ein Job analysiert und optimiert werden - ohne ihn im Ganzen simulieren zu müssen. Siehe auch Kapitel 2.2.4 "Jobs simulieren und nachbessern", Seite 24.

Am Ende des Anwenderprogramms muss **slsc_cfg_delete** stehen, um die **syncAXIS control**-Instanz wieder abzubauen und auch die Hardware wieder freizugeben.

Funktionen zur Konfigurationsänderung der bestehenden **syncAXIS control**-Instanz

- **slsc_cfg_select_heuristic**
- **slsc_cfg_set_bandwidth**
- **slsc_cfg_set_calculation_dynamics_jump_scan_device**
- **slsc_cfg_set_calculation_dynamics_mark_scan_device**
- **slsc_cfg_set_calculation_dynamics_stage**
- **slsc_cfg_set_dynamic_limits_scan_device**
- **slsc_cfg_set_dynamic_limits_stage**
- **slsc_cfg_set_dynamic_violation_reaction**
- **slsc_cfg_set_field_limits_scan_device**
- **slsc_cfg_set_field_limits_stage**
- **slsc_cfg_set_jump_speed**
- **slsc_cfg_set_mark_speed**
- **slsc_cfg_set_part_displacement**
- **slsc_cfg_set_scan_device_dynamic_monitoring_level**
- **slsc_cfg_set_stage_dynamic_monitoring_level**
- **slsc_cfg_set_trajectory_config** (mit **slsc_cfg_delete_trajectory_config**)

Diesen Funktionen ist gemeinsam:

- Es erfolgt *keine Re-Initialisierung* der **syncAXIS control**-Instanz.
- Die angegebenen Konfigurations-Änderungen werden erst durch **slsc_list_begin*** wirksam. Diese Konfigurations-Änderungen gelten dann vom nächsten **Job** an für alle folgenden **Jobs**.

Hinweise

- Für **slsc_cfg_set_jump_speed** gibt es die entsprechende Job-Funktion (**slsc_list_***) **slsc_list_set_jump_speed**.
- Für **slsc_cfg_set_mark_speed** gibt es die entsprechende Job-Funktion (**slsc_list_***) **slsc_list_set_mark_speed**.
- **slsc_cfg_set_trajectory_config** stellt die Konfigurationswerte der **Trajektorienplanung** für die angegebene **syncAXIS control**-Instanz ein.
- **slsc_cfg_delete_trajectory_config** ist eine Hilfsfunktion für die Softwareentwicklung. Damit kann das Trajektorienkonfigurations-Objekt (das durch **slsc_cfg_set_trajectory_config** erstellt wird) zur Vermeidung von Speicherlecks wieder gelöscht werden, wenn es nicht mehr gebraucht wird. **slsc_cfg_delete_trajectory_config** verändert keine Konfigurations-Parameter-Werte.
- Weitergehende Informationen zu **slsc_cfg_set_part_displacement** siehe Kapitel 8.3 "Über Transformationen in **syncAXIS control** V1.2.4 und höher", Seite 332.
- Unterschiedliche **Jobs** haben jeweils unterschiedliche "optimale" **FilterBandwidth**-Werte. Neben der Möglichkeit den **FilterBandwidth**-Wert im Anwenderprogramm ändern zu können, wird **slsc_cfg_set_bandwidth** auch bereitgestellt, um eine Optimierungsalgorithmen implementieren zu können.
- **slsc_cfg_select_heuristic** stellt die gewünschte Kennlinie für die Geschwindigkeitsverminderung (**DynamicReductionFunction**) ein.
- Um momentane Werte der **syncAXIS control**-Instanz zu ändern, für die es Entsprechungen/deziierte Tags in der **syncAXISConfig.xml** gibt, stehen zur Verfügung:
 - **slsc_cfg_set_calculation_dynamics_jump_scan_device**
 - **slsc_cfg_set_calculation_dynamics_mark_scan_device**
 - **slsc_cfg_set_calculation_dynamics_stage**
 - **slsc_cfg_set_dynamic_limits_scan_device**
 - **slsc_cfg_set_dynamic_limits_stage**
 - **slsc_cfg_set_dynamic_violation_reaction**
 - **slsc_cfg_set_field_limits_scan_device**
 - **slsc_cfg_set_field_limits_stage**
 - **slsc_cfg_set_scan_device_dynamic_monitoring_level**
 - **slsc_cfg_set_stage_dynamic_monitoring_level**

Funktionen zum Erfassen von "Callback Events"

- `slsc_cfg_register_callback_job_start_planned`
- `slsc_cfg_register_callback_job_progress_planned`
- `slsc_cfg_register_callback_job_end_planned`
- `slsc_cfg_register_callback_job_finished_executing`
- `slsc_cfg_register_callback_job_is_executing`
- `slsc_cfg_register_callback_job_loaded_enough`

Wie in [Abbildung 12, Seite 43](#) und nachfolgender Tabelle gezeigt, treten `syncAXIS control`-Instanzen intern bei jedem `Job` mehrmals unterschiedliche "Callback-Events" auf. Eine entsprechende Funktion zum Erfassen von "Callback-Events" steht für jeden dieser "Callback-Events" zur Verfügung, z. B. `slsc_cfg_register_callback_job_finished_executing` für "job_finished_executing".

Mit diesen Funktionen kann die `syncAXIS control`-Instanz so konfiguriert werden, dass die angegebene benutzerdefinierte Funktion (= "Callback-Funktion") aufgerufen wird, sobald der zugehörige "Callback-Event" auftritt.

Jede "Callback-Funktion" muss der ihr vorgegebenen Konvention entsprechen (Rückgabetypen, Aufrufkonventionen, Argumente), siehe die beiden **Callback-Funktions-Signaturen** `slsc_ExecTimeCallback` und `slsc_JobCallback`.

Name des "Callback-Events"	Häufigkeit pro Job	Entsprechende Funktion zum Erfassen von "Callback-Events"	Für die "Callback-Funktion" vorgeschriebene Signatur
<code>job_start_planned</code>	1 ×	<code>slsc_cfg_register_callback_job_start_planned</code>	<code>slsc_JobCallback</code>
<code>job_progress_planned</code>	Kann mehrmals auftreten ^(a)	<code>slsc_cfg_register_callback_job_progress_planned</code>	<code>slsc_ExecTimeCallback</code>
<code>job_end_planned</code>	1 ×	<code>slsc_cfg_register_callback_job_end_planned</code>	<code>slsc_ExecTimeCallback</code>
<code>job_loaded_enough</code>	1 ×	<code>slsc_cfg_register_callback_job_loaded_enough</code>	<code>slsc_JobCallback</code>
<code>job_is_executing</code>	Kann mehrmals auftreten	<code>slsc_cfg_register_callback_job_is_executing</code>	<code>slsc_ExecTimeCallback</code>
<code>job_finished_executing</code>	1 ×	<code>slsc_cfg_register_callback_job_finished_executing</code>	<code>slsc_ExecTimeCallback</code>

(a) Bei sehr kurzen `Jobs` vielleicht auch gar nicht.

Hinweise

- Einfache Anwendungsbeispiele sind:
 - Eine externe Überwachung (= durch eine GUI) des Fortschritts von **Jobs**
 - Eine "Rote Ampel" in einer GUI, solange ein **Job** ausgeführt wird
 - Eine GUI-Meldung, wenn ein **Job** beendet ist
- Beispiel-Codes:
 - Eine Beispiel-Implementierung unter Verwendung von Funktions-Zeigern zeigt **Abbildung 31, Seite 83**
 - Eine Beispiel-Implementierung unter Verwendung von C++11 Lambda-Funktionen (anonyme Funktionen) zeigt **Abbildung 32, Seite 84**
- Die in **slsc_cfg_register_callback_job_start_planned** angegebene "Callback-Funktion" wird ausgeführt, sobald die **Trajektorienplanung** für den **Job** gestartet wurde. Dies bedeutet, dass **slsc_list_begin[*]** aufgerufen und verarbeitet wurde, die erste Job-Funktion (wie **slsc_list_jump_abs** usw.) aufgerufen und erfolgreich durch die Validierungsprüfung verarbeitet wurde. Die **Puffer** müssen ausreichend gefüllt sein, damit die **Trajektorienplanung** beginnen und die Bewegung für Scan-Kopf und Verfahrtschicht aufgeteilt werden kann.
- Die in **slsc_cfg_register_callback_job_progress_planned** angegebene "Callback-Funktion" wird während der **Job**-Planung wiederholt ausgeführt. Dies bedeutet, dass die in **slsc_cfg_register_callback_job_start_planned** angegebene "Callback-Funktion" bereits ausgeführt wurde. Bei sehr kurzen **Jobs** könnte es sein, dass der "Callback-Event" vom Typ **"job_progress_planned"** gar nicht auftritt.
- Die in **slsc_cfg_register_callback_job_loaded_enough** angegebene "Callback-Funktion" wird ausgeführt, sobald die **Job**-Planung so weit fortgeschritten ist, dass der **Job** gestartet werden könnte. Dies bedeutet, dass die Informationsübertragung zur RTC6-Karte begonnen wurde und schon einige RTC6-Befehle in den RTC6-Listenspeicher geschrieben wurden.

Etwa in diesem Moment geht der Ausführungszustand (**slsc_ExecState**) des **Jobs** auch auf **slsc_ExecState_ReadyForExecution** über (aufgrund der internen Informationsabfrage nicht unbedingt genau der gleiche Zeitpunkt). Zur Prüfung, ob die **Job**-Ausführung gestartet werden kann, empfiehlt SCANLAB sich nicht auf die "Call-back-Funktion" zu verlassen sondern stattdessen **slsc_ctrl_get_exec_state** zu verwenden. Ab diesem Zeitpunkt kann die **Job**-Ausführung mit **slsc_ctrl_start_execution** gestartet werden.

- Die in **slsc_cfg_register_callback_job_end_planned** angegebene "Callback-Funktion" wird nach Beendigung der **Job**-Planung ausgeführt. Dies bedeutet, dass alle Berechnungen durchgeführt wurden und die Berechnung des nächsten **Jobs** beginnt, wenn der Benutzer eine andere **Job**-Planung gestartet hat. syncAXIS control überträgt die restlichen Informationen weiterhin an die RTC6-Karte. Ab diesem Zeitpunkt können Benutzer bei Bedarf die **Job**-Merkmale ("Key") mit **slsc_ctrl_get_job_characteristic** abfragen.
- Die in **slsc_cfg_register_callback_job_is_executing** angegebene "Callback-Funktion" wird während der Ausführung des **Jobs** wiederholt ausgeführt. Dies bedeutet, dass das "Callback-Event" **"job_loaded_enough"** bereits aufgetreten ist und die Ausführung des **Jobs** tatsächlich durch **slsc_ctrl_start_execution** gestartet wurde. Zu diesem Zeitpunkt wechselt der Ausführungszustand (**slsc_ExecState**) des **Jobs** zu **slsc_ExecState_Executing**. Bei sehr kurzen **Jobs** könnte es sein, dass der "Callback-Event" vom Typ **"job_is_executing"** gar nicht auftritt.
- Die in **slsc_cfg_register_callback_job_finished_executing** angegebene "Callback-Funktion" wird nach Beendigung der **Job**-Ausführung ausgeführt. Dies bedeutet, dass Scan-Kopf und Verfahrtschicht ihre Endposition erreicht haben und die Bewegung gestoppt wurde. Zu diesem Zeitpunkt wechselt der Ausführungszustand (**slsc_ExecState**) des **Jobs** auf **slsc_ExecState_Idle**. Es wird kein weiterer "Callback-Event" zu diesem **Job** ausgelöst werden.


```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.
// Es werden Funktionen und Konzepte verwendet, die im "Installation_Project" des syncAXIS control-Software-Pakets
// deklariert und definiert sind, siehe dort.

struct Content
{
    int* Counter;
    std::string Action;
};

void execCallbackFunction(size_t JobID, uint64_t Progress, double ExecTime, void* Context)
{
    Content* CallbackContent = static_cast<Content*>(Context);
    std::cout << "Step number " << (*CallbackContent->Counter)++ << " " << CallbackContent->Action << ". Execution up to now took: "
    << ExecTime << std::endl; return;
};

void jobCallbackFunction(size_t JobID, void* Context)
{
    Content* CallbackContent = static_cast<Content*>(Context);
    std::cout << "Step number " << (*CallbackContent->Counter)++ << " " << CallbackContent->Action << "." << std::endl;
    return;
};

size_t SLHandle = 0;
// Initialize Instance, e.g.
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

int ProcessCounter = 0;

Content ContentIsPlanning = { &ProcessCounter, "is planning" };
Content ContentIsExecuting = { &ProcessCounter, "is executing" };
Content ContentPlanningStarted = { &ProcessCounter, "started the planning" };
Content ContentJobFinished = { &ProcessCounter, "finished the execution" };
Content ContentPlanningFinished = { &ProcessCounter, "finished the planning" };
Content ContentJobReady = { &ProcessCounter, "loaded enough information in the buffers to start the execution." };

slsc_cfg_register_callback_job_finished_executing(SLHandle, static_cast<slsc_ExecTimeCallback>(&execCallbackFunction), &ContentJobFinished);
slsc_cfg_register_callback_job_end_planned(SLHandle, static_cast<slsc_ExecTimeCallback>(&execCallbackFunction), &ContentPlanningFinished);
slsc_cfg_register_callback_job_loaded_enough(SLHandle, static_cast<slsc_JobCallback>(&jobCallbackFunction), &ContentJobReady);
slsc_cfg_register_callback_job_is_executing(SLHandle, static_cast<slsc_ExecTimeCallback>(&execCallbackFunction), &ContentIsExecuting);
slsc_cfg_register_callback_job_progress_planned(SLHandle, static_cast<slsc_ExecTimeCallback>(&execCallbackFunction), &ContentIsPlanning);
slsc_cfg_register_callback_job_start_planned(SLHandle, static_cast<slsc_JobCallback>(&jobCallbackFunction), &ContentPlanningStarted);

// Execute random job, e.g. (from Installation Project)
{
    double ScalingFactor = 10;
    auto ListFilling = std::async(std::launch::async, [&]()
    {
        return writeMarkingPattern1(SLHandle, RetVal, ScalingFactor, Transformation, Offset);
    });
};

startJob(SLHandle, RetVal, true);
}

// Delete Instance, e.g.
slsc_cfg_delete(SLHandle);
```

Beispiel-Code: Implementierung der Funktionen zum Erfassen von "Callback-Events" (siehe Seite 12) und "Callback-Funktionen". Anders als in Abbildung 32, Seite 84 werden hier Funktions-Zeiger verwendet.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.
// Es werden Funktionen und Konzepte verwendet, die im "Installation_Project" des syncAXIS control-Software-Pakets
// deklariert und definiert sind, siehe dort.

struct Content
{
    int* Counter;
    std::string Action;
};

size_t SLHandle = 0;
//Initialize Instance, e.g.
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

int ProcessCounter = 0;

Content ContentIsPlanning= { &ProcessCounter, "is planning" };
Content ContentIsExecuting = { &ProcessCounter, "is executing" };
Content ContentPlanningStarted = { &ProcessCounter, "started the planning" };
Content ContentJobFinished = { &ProcessCounter, "finished the execution" };
Content ContentPlanningFinished = { &ProcessCounter, "finished the planning" };
Content ContentJobReady = { &ProcessCounter, "loaded enough information in the buffers to start the execution." };

slsc_ExecTimeCallback ExecCallback = [](size_t JobID, uint64_t Progress, double ExecTime, void* Context)
{
    Content* CallbackContent = static_cast<Content*>(Context);
    std::cout << "Step number " << (*CallbackContent->Counter)++ << " " << CallbackContent->Action << ". Execution up to now took: "
    << ExecTime << std::endl; return;
};

slsc_JobCallback JobCallback = [](size_t JobID, void* Context)
{
    Content* CallbackContent = static_cast<Content*>(Context);
    std::cout << "Step number " << (*CallbackContent->Counter)++ << " " << CallbackContent->Action << "." << std::endl;
    return;
};

slsc_cfg_register_callback_job_finished_executing(SLHandle, ExecCallback, &ContentJobFinished);
slsc_cfg_register_callback_job_end_planned(SLHandle, ExecCallback, &ContentPlanningFinished);
slsc_cfg_register_callback_job_loaded_enough(SLHandle, JobCallback, &ContentJobReady);
slsc_cfg_register_callback_job_is_executing(SLHandle, ExecCallback, &ContentIsExecuting);
slsc_cfg_register_callback_job_progress_planned(SLHandle, ExecCallback, &ContentIsPlanning);
slsc_cfg_register_callback_job_start_planned(SLHandle, JobCallback, &ContentPlanningStarted);

// execute random job, e.g. (from Installation Project)
{
    double ScalingFactor = 10;
    auto ListFilling = std::async(std::launch::async, [&]()
    {
        return writeMarkingPattern1(SLHandle, RetVal, ScalingFactor, Transformation, Offset);
    });
}

startJob(SLHandle, RetVal, true);

// Delete Instance, e.g.
slsc_cfg_delete(SLHandle);
```

Beispiel-Code: Implementierung der Funktionen zum Erfassen von **"Callback-Events"** (siehe Seite 12) und **"Callback-Funktionen"**. Anders als in Abbildung 31, Seite 83 werden hier C++ 11 Lambda-Funktionen (anonyme Funktionen) verwendet.

3.1.2 Job-Funktionen (slsc_list_*)

Job-Funktionen (Präfix **slsc_list_**) erlauben das Definieren von **Jobs**, d. h. sie sind die einzelnen Elemente von **Jobs**. Wesentlich sind die Funktionen für Markierungen und Sprünge, sowie für das Schalten von Signalen an Ausgabe-Ports (neben den verpflichtenden Funktionen für Anfang und Ende von **Jobs**). Einen grafischen Überblick gibt **Abbildung 33**, **Seite 86**.

Hinweise

- Für alle Job-Funktionen (**slsc_list_***) gilt: Beim Übertragen an die syncAXIS-DLL wird eine Konsistenzprüfung durchgeführt. Im Fehlerfall ist hier beim Rückgabewert **Bit #07** gesetzt (**JobStructureNotValid**).
- Siehe auch **Abschnitt "Einzuhaltende Struktur bei der Job-Definition"**, **Seite 25**.
- Job-Funktionen (Präfix **slsc_list_**) sind technisch völlig anders als die im **RTC6-Handbuch** beschriebenen "RTC-Listenbefehle (Suffix **_list**)" und werden daher in diesem Dokument nicht als solche bezeichnet. So sind z. B. die Ausführdauern von Job-Funktionen aus Benutzersicht nicht genau vorherzusehen (im Gegensatz zu RTC-Listenbefehlen). Auch gibt es Job-Funktionen, die mehrere RTC6-Listenbefehle für die RTC6-Karte generieren.
- Die letzte Job-Funktion im **Job** muss **slsc_list_end** sein. Sie berechnet einen Abschluss für die **Trajektorie**. Danach sind (wie bei **slsc_cfg_initialize_from_file** und **slsc_cfg_reinitialize_from_file**)
 - die Scan-Kopf-Spiegel in der Nullposition und
 - der Verfahrtschicht (nicht relevant im **Betriebsmodus "ScannerOnly"**) verbleibt auf der zuletzt eingestellten Sprungposition bzw. Markierposition.

Funktionen zum Definieren von Job-Anfängen/Enden

- slsc_list_begin**
- slsc_list_begin_absolute**
- slsc_list_begin_relative**
- slsc_list_end**

Die erste Funktion eines **Jobs** muss **slsc_list_begin**, (alternativ auch **slsc_list_begin_absolute** oder **slsc_list_begin_relative**, siehe deren Referenztabellen) sein.

Anderenfalls ist beim Rückgabewert **Bit #07** gesetzt (**JobStructureNotValid**).

Keine der Funktionen **slsc_list_begin**, **slsc_list_begin_absolute** und **slsc_list_begin_relative** darf von **slsc_list_begin**, **slsc_list_begin_absolute** oder **slsc_list_begin_relative** gefolgt werden. Anderenfalls ist beim Rückgabewert **Bit #07** gesetzt (**JobStructureNotValid**).

slsc_list_end muss die letzte Job-Funktion im **Job** sein, siehe Listenpunkt links.

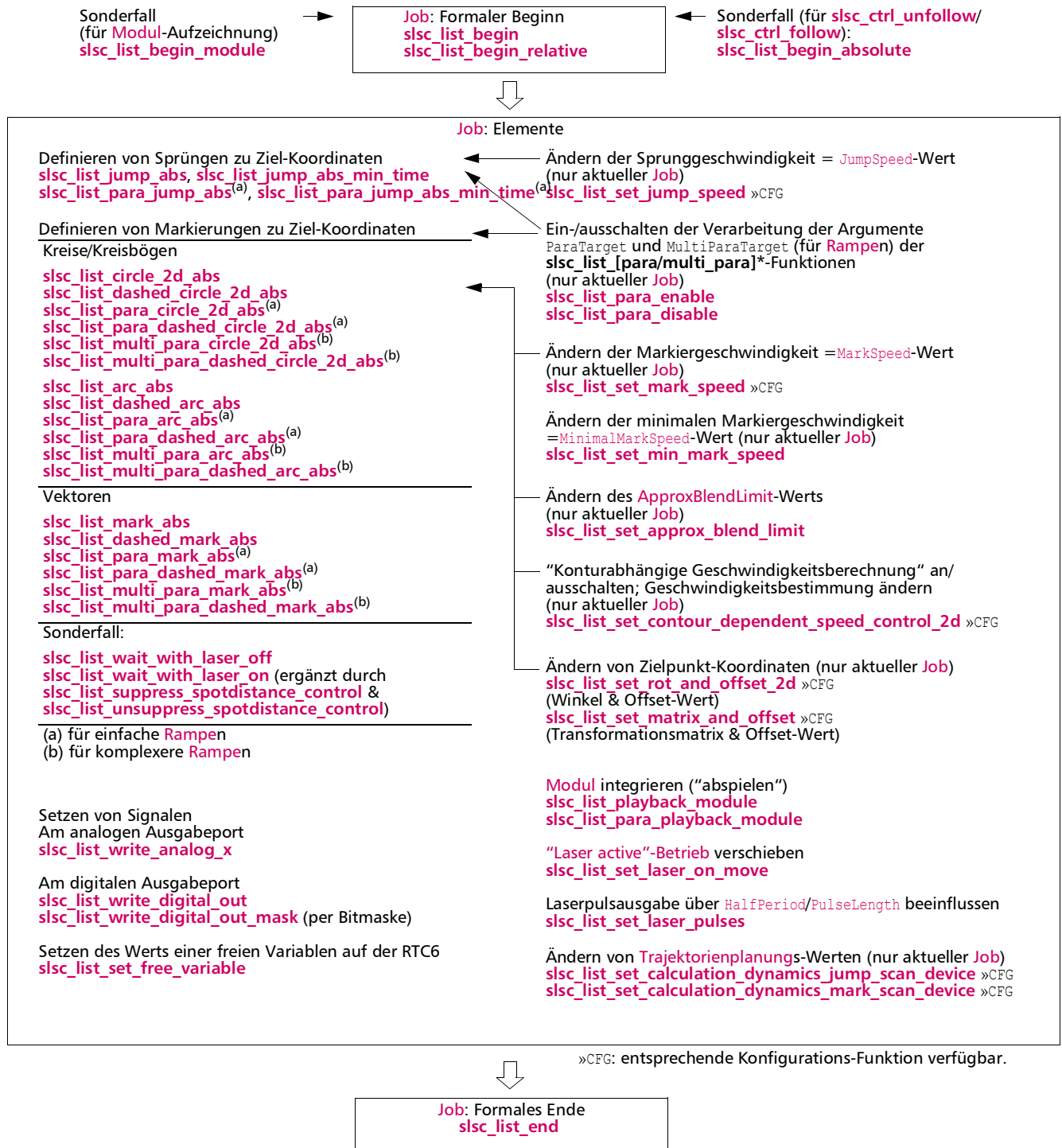
Funktionen zum Definieren von Sprüngen

- slsc_list_jump_abs**
- slsc_list_jump_abs_min_time**

Mit **slsc_list_jump_abs_min_time** kann – wie mit **slsc_list_jump_abs** – ein Sprung definiert werden, erlaubt jedoch zusätzlich die Angabe dessen Mindestdauer.

Die entsprechende **slsc_list_para***-Funktion von **slsc_list_jump_abs** ist **slsc_list_para_jump_abs**, die von **slsc_list_jump_abs_min_time** ist **slsc_list_para_jump_abs_min_time**.

Job-Funktionen (slsc_list_*)



Funktionen zum Definieren von Markierungen

- `slsc_list_arc_abs`
- `slsc_list_circle_2d_abs`
- `slsc_list_mark_abs`
- `slsc_list_wait_with_laser_off`
- `slsc_list_wait_with_laser_on` (mit ergänzenden `slsc_list_suppress_spotdistance_control` und `slsc_list_unsuppress_spotdistance_control`)
- `slsc_list_set_laser_on_move`

`slsc_list_wait_with_laser_on` verhält sich wie ein `slsc_list_mark_abs` mit Geschwindigkeit 0 (d.h. für eine angegebene Zeit verbleiben die Spiegel bei angesaltetem Laser in der zuletzt angefahrenen Position) und kann z.B. genutzt werden, wenn die Qualität des Laserspots durch externe Sensoren gemessen werden soll. Ergänzt wird `slsc_list_wait_with_laser_on` durch `slsc_list_wait_with_laser_off`. Der einzige Unterschied zwischen den beiden Funktionen ist, dass bei `slsc_list_wait_with_laser_off` der Laser ausgeschaltet ist.

Sonderfall: `SpotDistance` als "ActiveChannel"

Nur diejenigen Benutzer, die (in der `syncAXISConfig.xml`) `SpotDistance` als "ActiveChannel" eingestellt haben, siehe Kapitel 2.9.2 "Definition der Channel und ActiveChannel", Seite 48, müssen vor `slsc_list_wait_with_laser_on` zuerst `slsc_list_suppress_spotdistance_control` aufrufen. Bei triggerbaren Lasern werden sonst keine Pulse ausgelöst.

Hintergrund: Mit `SpotDistance` als "ActiveChannel" wird der zeitliche Pulsabstand (d.h. die HalfPeriod für die LASER1-Signale und LASER2-Signale) während der Job-Ausführung so angepasst, dass im Endeffekt die Spotabstände äquidistant sind. Die syncAXIS-DLL-interne Berechnung dafür basiert allerdings auf der momentanen Markiergeschwindigkeit. Je geringer sie ist, um so größer ist der Pulsabstand. Im Grenzfall "Markiergeschwindigkeit = 0" schließlich (ist bei `slsc_list_wait_with_laser_on` der Fall, s.o.) wird gar kein Puls mehr beim Laser getriggert.

Ein (zu `slsc_list_wait_with_laser_on` vorgängiger) Aufruf von

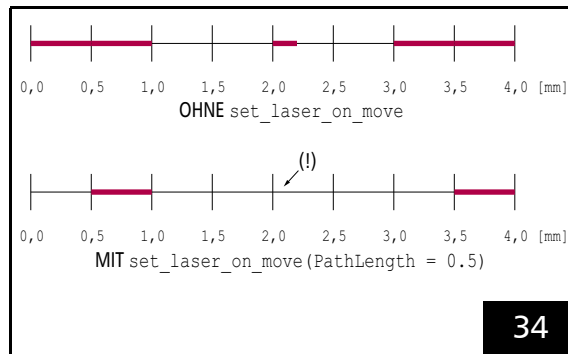
`slsc_list_suppress_spotdistance_control` unterdrückt die Funktionalität zum Erreichen äquidistanter Spotabstände solange, bis dieser Zustand durch `slsc_list_unsuppress_spotdistance_control` wieder aufgehoben wird.

Solange die Unterdrückung besteht, wird zum Auslösen von Laserpulsen verwendet:

- der `HalfPeriod`-Wert von `slsc_ctrl_set_laser_pulses` oder
- der `HalfPeriod`-Wert von `slsc_list_set_laser_pulses` oder
- der `HalfPeriod`-Attributwert aus dem `syncAXISConfig.xml`-Tag `LaserOutput`

Über `slsc_list_set_laser_on_move`

`slsc_list_set_laser_on_move` verzögert den "Laser active"-Betrieb um genau die Zeitdauer, die gebraucht wird, um die angegebene Pfadlänge (`PathLength`) auf dem aktuellen Markierabschnitt abzufahren, siehe [Abbildung 34](#).



`slsc_list_set_laser_on_move`. Erläuterungen siehe Text, [Seite 87](#).

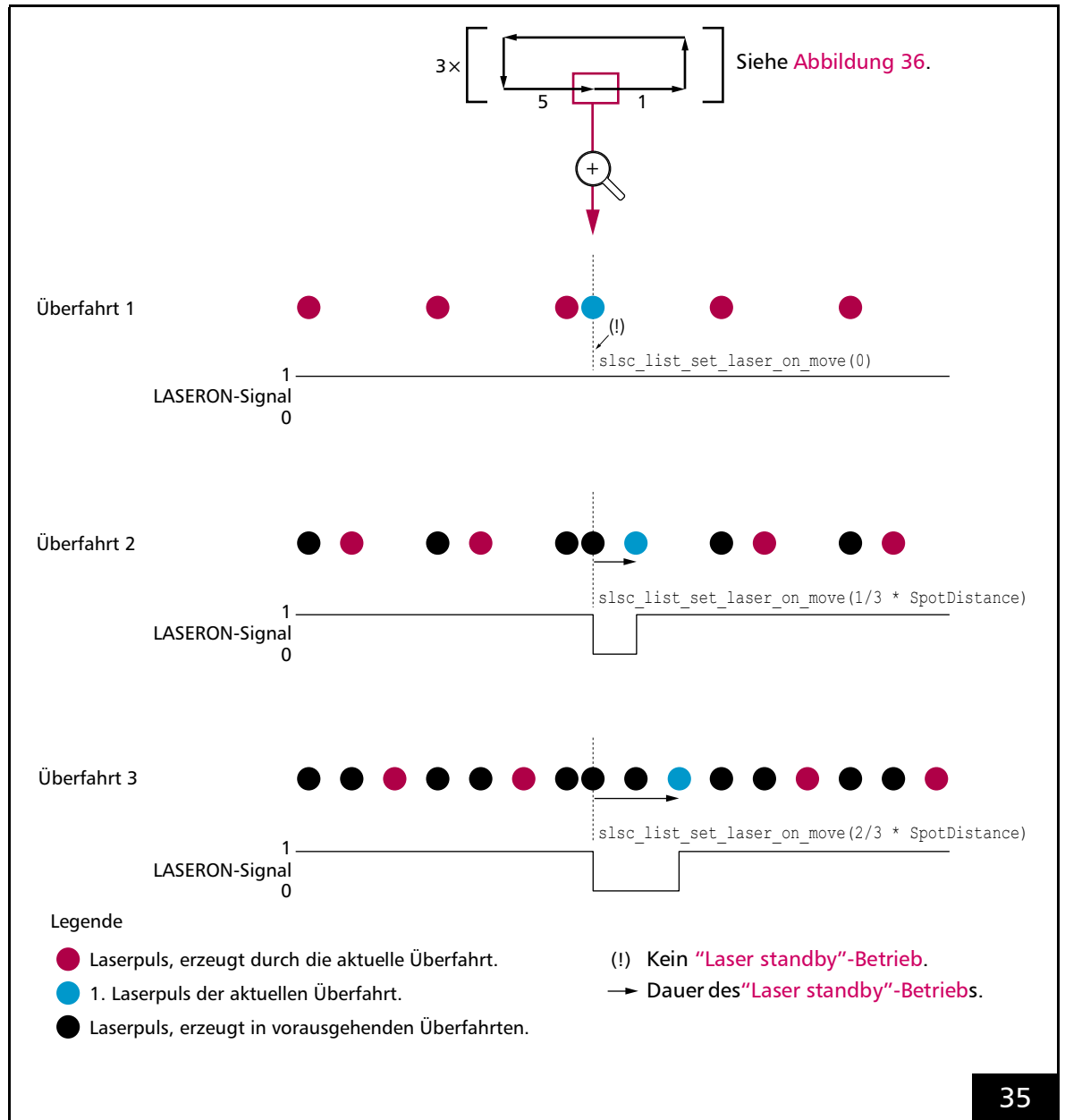
Ein Anwendungsfall ist ein Markiermuster, das mit äquidistanten Laserspotabständen (siehe `SpotDistance` und [Kapitel 2.9.5 "Über die "Konturabhängige Geschwindigkeitsberechnung"', Seite 60](#)) mehrmals markiert werden soll ("mehrfache Überfahrten"). Bei jeder Wiederholung kann mit `slsc_list_set_laser_on_move` der "Laser active"-Betrieb verzögert werden (d. h. der erste Laserpuls wird örtlich versetzt). Auf diese Weise kommen die Laserpulse der einzelnen Überfahrten nicht übereinander zu liegen, siehe [Abbildung 35](#).

Für einen `slsc_list_set_laser_on_move`-Aufruf zwischen zwei Markiermuster-Abschnitten mit Blending-Kurve gilt:

- die Blending-Kurve wird durch eine Sky Writing-ähnliche Bewegung ersetzt

Für einen `slsc_list_set_laser_on_move`-Aufruf zwischen zwei Markiermuster-Abschnitten mit "normaler" Überfahrt (= es wird weder eine Sky Writing-ähnliche Bewegung noch eine Blending-Kurve eingefügt) gilt:

- Am Übergang der beiden Markiermuster-Abschnitte wird kurzzeitig in den "Laser standby"-Betrieb gewechselt (siehe [Abbildung 35](#), fallende Flanke des LASERON-Signals)
- Nach der `PathLength` wird wieder in den "Laser active"-Betrieb gewechselt (siehe [Abbildung 35](#), steigende Flanke an den Pfeilenden). Gegebenenfalls wird eine Sky Writing-ähnliche Bewegung eingefügt, damit die `LaserMinOffTime` nicht unterschritten wird



slsc_list_set_laser_on_move. Erläuterungen siehe Text, **Seite 87**.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

size_t JobID;
slsc_list_begin(SLHandle, &JobID);

// Target points: 2 x 1 mm rectangle
const std::array<double, 2> LowerMid = { 0.0, 0.0 };
const std::array<double, 2> LowerRightCorner = { 1.0, 0.0 };
const std::array<double, 2> UpperRightCorner = { 1.0, 1.0 };
const std::array<double, 2> UpperLeftCorner = { -1.0, 1.0 };
const std::array<double, 2> LowerLeftCorner = { -1.0, 0.0 };

// Say spot distance is 10 µm
const double SpotDistance = 0.01; // mm

// Jump to start position
slsc_list_jump_abs(SLHandle, LowerMid.data());

// Traversing 1
slsc_list_set_laser_on_move(SLHandle, 0.0);

slsc_list_mark_abs(SLHandle, LowerRightCorner.data());
slsc_list_mark_abs(SLHandle, UpperRightCorner.data());
slsc_list_mark_abs(SLHandle, UpperLeftCorner.data());
slsc_list_mark_abs(SLHandle, LowerLeftCorner.data());
slsc_list_mark_abs(SLHandle, LowerMid.data());

// Traversing 2
slsc_list_set_laser_on_move(SLHandle, 1.0 / 3.0 * SpotDistance);

slsc_list_mark_abs(SLHandle, LowerRightCorner.data());
slsc_list_mark_abs(SLHandle, UpperRightCorner.data());
slsc_list_mark_abs(SLHandle, UpperLeftCorner.data());
slsc_list_mark_abs(SLHandle, LowerLeftCorner.data());
slsc_list_mark_abs(SLHandle, LowerMid.data());

// Traversing 3
slsc_list_set_laser_on_move(SLHandle, 2.0 / 3.0 * SpotDistance);


slsc_list_mark_abs(SLHandle, LowerRightCorner.data());
slsc_list_mark_abs(SLHandle, UpperRightCorner.data());
slsc_list_mark_abs(SLHandle, UpperLeftCorner.data());
slsc_list_mark_abs(SLHandle, LowerLeftCorner.data());
slsc_list_mark_abs(SLHandle, LowerMid.data());

slsc_list_end(SLHandle);
```

Code-Beispiel: **slsc_list_set_laser_on_move**.

[*]dashed[*]-Funktionen

- `slsc_list_dashed_arc_abs`
- `slsc_list_dashed_circle_2d_abs`
- `slsc_list_dashed_mark_abs`
- `slsc_list_multi_para_dashed_arc_abs`
- `slsc_list_multi_para_dashed_circle_2d_abs`
- `slsc_list_multi_para_dashed_mark_abs`
- `slsc_list_para_dashed_arc_abs`
- `slsc_list_para_dashed_circle_2d_abs`
- `slsc_list_para_dashed_mark_abs`

[*]dashed[*]-Funktionen sind speziell dafür entwickelt, den Laser entlang von Markiermuster-Abschnitten (siehe **Markier-Funktionen**) räumlich oft ein/ausschalten zu können. Anwendungsfälle sind z. B. Markiermuster mit Schraffuren aus kurzen Strichen .

Für alle [*]dashed[*]-Funktionen gilt:

- Sie verhalten sich prinzipiell wie ihre entsprechenden (ohne “_dashed” im Namen) nicht-[*]dashed[*]-Funktionen
- Sie bieten zusätzlich je 2 Argumente zum Schalten des Lasers. Diese verhalten sich bei allen [*]dashed[*]-Funktionen gleich:
 - `NSwitches`
 - `LaserSwitches`
- `NSwitches` ist die Größe des `LaserSwitches`-Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1.
- `LaserSwitches` ist ein Array von `double`-Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen “Laser standby”-Betrieb und “Laser active”-Betrieb gewechselt wird.
Der erste Wechsel:
 - Ist als Wechsel zum “Laser active”-Betrieb festgelegt
 - Ist für eine Bogenlänge von 0,0 mm erlaubt, d.h. der Laser wird sofort bei Beginn des Markiermuster-Abschnitts eingeschaltet

Die `LaserSwitches`-Werte müssen folgende Voraussetzungen erfüllen:

- Aufsteigende Reihenfolge
- $\geq 0,0$
- \leq Gesamt-Bogenlänge des Markiermuster-Abschnitts
- Dauer des “Laser standby”-Betriebs⁽¹⁾:
 $\geq [\text{LaserMinOffTime} + \text{LaserPreTriggerTime}]$
- Zeit zwischen⁽¹⁾ zwei gleichartigen Wechseln⁽²⁾:
 $\geq 1 \mu\text{s}$
- Ist eine der vorgenannten Voraussetzungen nicht erfüllt, sind bei den Rückgabewerten der aufrufenden [*]dashed[*]-Funktionen dann **Bit #06** gesetzt (`UnplausibleOrUnknownParameter`).
- Wichtig: Unabhängig vom gegenwärtig eingestellten `BlendMode` sind durch [*]dashed[*]-Funktionen definierte Markiermuster-Abschnitte *niemals* Teil einer Blending-Kurve.

Hinweise

- Für [*]dashed[*]-Funktionen geeignete Markiermuster könnten Sie alternativ durch eine Abfolge von kurzen **Markier-Funktionen** und **Sprung-Funktionen** erzielen. Allerdings besteht hier das Risiko, dass die Berechnungsdauer stark ansteigt. Ursächlich dafür wären die geometrischen Berechnungen (siehe **Abbildung 11**), die für jeden dieser **Markier-Funktionen** und **Sprung-Funktionen** einzeln durchgeführt werden müssen. Deren Berechnungsdauern sind unabhängig von der räumlichen Ausdehnung der Markiermuster-Abschnitte. Im Gegensatz dazu werden bei [*]dashed[*]-Funktionen diese Berechnungen – unabhängig vom angegebenen `NSwitches`-Wert – nur $1 \times$ durchgeführt.
- Abfolgen von **Markier-Funktionen** und **Sprung-Funktionen** dagegen:
 - Eignen sich durchaus dann, wenn der Laser nur relativ selten ein/ausgeschaltet werden soll
 - Müssen sogar verwendet werden, wenn Blending-Kurven gewünscht sind

(1) Die Zeitdifferenzen ergeben sich aus den vorgegebenen Bogenlängen und der Markiergeschwindigkeit.

(2) “Laser standby”-Betrieb -> “Laser standby”-Betrieb
“Laser active”-Betrieb -> “Laser active”-Betrieb

Funktionen zum Ändern der Zielpunkt-Koordinaten

- **slsc_list_set_matrix_and_offset**
- **slsc_list_set_rot_and_offset_2d**

slsc_list_set_rot_and_offset_2d ändert (ab der Einfügestelle) Zielpunkt-Koordinaten (siehe Seite 268) um einen Winkel und Offset-Wert nachfolgender **slsc_list_arc_abs**, **slsc_list_circle_2d_abs**, **slsc_list_jump_abs**, **slsc_list_mark_abs** und deren entsprechenden

slsc_list_[para/multi_para]* Funktionen, aber nur bis zum Ende des gerade laufenden **Jobs**. Bei **slsc_list_set_rot_and_offset_2d** kann ein Winkel und Offset-Wert angegeben werden, bei **slsc_list_set_matrix_and_offset** eine Transformationsmatrix und Offset-Wert. Für beide gibt es entsprechende Konfigurations-Funktionen (**slsc_cfg_***), **slsc_cfg_set_rot_and_offset_2d** und **slsc_cfg_set_matrix_and_offset**.

Die bei

slsc_list_set_matrix_and_offset/slsc_cfg_set_matrix_and_offset anzugebende Transformationsmatrix kann z.B. zum Drehen (mathematisch positiver Drehsinn d.h. positive Winkel drehen im Gegenuhrzeigersinn), Skalieren oder Umklappen von Markierungsergebnissen verwendet werden. Einige Beispiele zeigt nachfolgende Tabelle.

Drehung um den Winkel α	$\begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}$
Skalierung um die Faktoren k_x und k_y	$\begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$
Spiegelung an der y-Achse (Umklappen in x-Richtung)	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)

- **slsc_list_multi_para_arc_abs**
- **slsc_list_multi_para_circle_2d_abs**
- **slsc_list_multi_para_dashed_arc_abs⁽¹⁾**
- **slsc_list_multi_para_dashed_circle_2d_abs⁽¹⁾**
- **slsc_list_multi_para_dashed_mark_abs⁽¹⁾**
- **slsc_list_multi_para_mark_abs**
- **slsc_list_para_arc_abs**
- **slsc_list_para_circle_2d_abs**
- **slsc_list_para_dashed_arc_abs⁽¹⁾**
- **slsc_list_para_dashed_circle_2d_abs⁽¹⁾**
- **slsc_list_para_dashed_mark_abs⁽¹⁾**
- **slsc_list_para_disable**
- **slsc_list_para_enable**
- **slsc_list_para_jump_abs**
- **slsc_list_para_jump_abs_min_time**
- **slsc_list_para_mark_abs**

Die **slsc_list_para***-Funktionen

(**slsc_list_para_arc_abs**, **slsc_list_para_circle_2d_abs**, **slsc_list_para_jump_abs**, **slsc_list_para_jump_abs_min_time**, **slsc_list_para_mark_abs**) bieten (im Vergleich zu ihren entsprechenden **slsc_list***-Funktionen **slsc_list_arc_abs**, **slsc_list_circle_2d_abs**, **slsc_list_jump_abs**, **slsc_list_jump_abs_min_time**, **slsc_list_mark_abs**) zusätzlich das Argument **ParaTarget**. Über **ParaTarget** wird eine (einfache) **Rampe** definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert). Damit das Argument **ParaTarget**, ausgewertet wird, muss **slsc_list_para_enable** vorher aufgerufen worden sein. Anderenfalls wirken **slsc_list_para***-Funktionen wie ihre entsprechenden **slsc_list***-Funktionen. Das gleiche gilt, wenn kein "ActiveChannel" definiert ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48, oder **slsc_list_para_disable** vorher aufgerufen wurde.

(1) Siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.

Die `slsc_list_multi_para*`-Funktionen (`slsc_list_multi_para_arc_abs`, `slsc_list_multi_para_circle_2d_abs`, `slsc_list_multi_para_mark_abs`) bieten (im Vergleich zu ihren entsprechenden `slsc_list*`-Funktionen `slsc_list_arc_abs`, `slsc_list_circle_2d_abs`, `slsc_list_mark_abs`) zusätzlich das Argument `MultiParaTarget`. Über `MultiParaTarget` kann (je "ActiveChannel") eine **Rampe** definiert werden, die aus mehreren Abschnitten bestehen kann. Das ermöglicht auch die Definition von **Rampen** mit Sägezahn- und Rechteck-Profil, siehe auch **Abschnitt "Über Rampen"**, Seite 53.

Damit das Argument `MultiParaTarget`, ausgewertet wird, muss `slsc_list_para_enable` vorher aufgerufen worden sein. Anderenfalls wirken `slsc_list_multi_para*`-Funktionen wie ihre entsprechenden `slsc_list*`-Funktionen. Das gleiche gilt, wenn kein "ActiveChannel" definiert ist, siehe **Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")"**, Seite 48, oder `slsc_list_para_disable` vorher aufgerufen wurde.

Funktionen zum Setzen von Signalen

- `slsc_list_write_analog_x`
- `slsc_list_write_digital_out`
- `slsc_list_write_digital_out_mask`

Mit diesen Funktionen werden die Signale zwischen (bezogen auf das Markierungsergebnis im Bildfeld) zwei Job-Funktionen (`slsc_list_*`) an den angegebenen Ausgabe-Ports gesetzt, siehe auch **Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden"**, Seite 45. Sie bleiben über das **Job**-Ende hinaus gesetzt.

Die zuletzt gesetzten Signale werden weiterhin ausgegeben, auch wenn die `syncAXIS control`-Instanz mit `slsc_cfg_delete` gelöscht wird.

Funktionen zum Ändern der Geschwindigkeiten

- `slsc_list_set_jump_speed`
- `slsc_list_set_mark_speed`

Eine Änderung durch `slsc_list_set_jump_speed` und `slsc_list_set_mark_speed` gilt für alle (ab der Einfügestelle) nachfolgenden Job-Funktionen (`slsc_list_*`), aber nur bis zum Ende des gerade laufenden **Jobs**.

Für beide gibt es entsprechende Konfigurations-Funktionen (`slsc_cfg_*`), `slsc_cfg_set_jump_speed` und `slsc_cfg_set_mark_speed`.

Funktion zum Ändern der Minimalen Geschwindigkeiten

- `slsc_list_set_min_mark_speed`

Eine Änderung durch `slsc_list_set_min_mark_speed` gilt für alle (ab der Einfügestelle) nachfolgenden Job-Funktionen (`slsc_list_*`), aber nur bis zum Ende des gerade laufenden **Jobs**.

Für `slsc_list_set_min_mark_speed` gibt es keine entsprechende Konfigurations-Funktion (`slsc_cfg_*`).

Funktionen zum Ändern von Trajektorienplanungs-Werten

- `slsc_list_set_calculation_dynamics_jump_scan_device`
- `slsc_list_set_calculation_dynamics_mark_scan_device`

Eine Änderung durch diese Funktionen gilt für alle (ab der Einfügestelle) nachfolgenden Job-Funktionen (`slsc_list_*`), aber nur bis zum Ende des gerade laufenden **Jobs**.

Mit diesen Funktionen können die zur Planung von **Trajektorien** (für **Betriebsmodus** "`ScannerOnly`" sowie "`ScannerAndStage`") verwendeten Werte für Beschleunigung und Ruck "lokal" (= an einer bestimmten Stelle im **Job**) verändert werden.

Mögliche Verwendungsszenarien:

- (1) Es werden sehr hohe Genauigkeitsanforderungen an das Markierergebnis gestellt. Beim Optimieren kann eine solche "lokale" Verringerung der Scan-Kopf-Beschleunigung das Markierergebnis weiter verbessern, weil dadurch der ohnehin geringe Regelfehler des Scan-Kopfs noch weiter verringert wird.
- (2) Zum punktuellen Optimieren der **Bewegungsaufteilung**, wenn:
 - Mit den (**Job**-weit gültigen) **Bewegungsaufteilungs**-Einstellungen (= `FilterBandwidth`-Wert, `DynamicReductionFunction`-Wert) wird zwar insgesamt schon ein relativ gutes Ergebnis erreicht
 - Allerdings werden an einigen wenigen Stellen die Verfahrtsch-Dynamikgrenzen immer noch überschritten
 Als Abhilfe können (mit $\geq V1.6$) an diesen wenigen Stellen **CalculationDynamics**-Werte⁽¹⁾ gezielt verringert werden (statt die **Bewegungsaufteilungs**-Einstellungen, s.o., ändern zu müssen).

Funktionen zum Ändern des Verhaltens von Blending-Kurven

- `slsc_list_set_approx_blend_limit`

Mit `slsc_list_set_approx_blend_limit` kann der `ApproxBlendLimit`-Wert geändert werden, siehe auch **Abbildung 39, Seite 292** bei `slsc_GeometryConfig`.

Eine Änderung durch `slsc_list_set_approx_blend_limit` gilt für alle (ab der Einfügestelle) nachfolgenden Job-Funktionen (`slsc_list_*`), aber nur bis zum Ende des gerade laufenden **Jobs**.

Funktion zur "Konturabhängigen Geschwindigkeitsberechnung"

- `slsc_list_set_contour_dependent_speed_control_2d`

Die "Konturabhängige Geschwindigkeitsberechnung" kann mit `slsc_list_set_contour_dependent_speed_control_2d` an- und ausgeschaltet sowie geändert werden. Zu Voraussetzungen und weiteren Informationen siehe **Kapitel 2.9.5 "Über die "Konturabhängige Geschwindigkeitsberechnung"', Seite 60**.

Eine Änderung durch `slsc_list_set_contour_dependent_speed_control_2d` gilt für alle (ab der Einfügestelle) nachfolgenden Job-Funktionen (`slsc_list_*`), aber nur bis zum Ende des gerade laufenden **Jobs**.

Für `slsc_list_set_contour_dependent_speed_control_2d` gibt es die entsprechende Konfigurationsfunktion (`slsc_cfg_*`) `slsc_cfg_set_contour_dependent_speed_control_2d`.

(1) `<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics>`

Funktion zum Setzen des Werts einer freien Variablen auf der RTC6

- **slsc_list_set_free_variable**

Siehe Abschnitt "Funktionen zum Verwalten des Werts einer freien Variablen auf der RTC6", Seite 99.

Funktion zum Beeinflussen der Laserpulsausgabe über HalfPeriod/PulseLength

- **slsc_list_set_laser_pulses**

Siehe Abschnitt "Funktion zum Beeinflussen der Laserpulsausgabe über HalfPeriod/PulseLength", Seite 100.

Funktionen für "Module"

- **slsc_list_begin_module**
- **slsc_list_para_playback_module**
- **slsc_list_playback_module**

Siehe Kapitel 2.11 "Über das Arbeiten mit "Modulen"", Seite 65.

3.1.3 Kontroll-Funktionen (slsc_ctrl_*)

Kontroll-Funktionen (Präfix **slsc_ctrl_**) erlauben das Kontrollieren des Anwenderprogrammflusses. Diese Funktionen dienen im Wesentlichen zur Steuerung des Anwenderprogrammablaufs:

- Lasersignale freigeben/nicht freigeben
- Abfrage, ob der syncAXIS-DLL-interne **Eingangspuffer** frei ist
- Ausführungsstatus der RTC6-Karte abfragen
- Abfragen von Fehlern
- Abfragen von Messsignalen
- Starten und Abbrechen der **Job**-Ausführung

Einen grafischen Überblick gibt **Abbildung 37**, **Seite 98**.

Hinweise

- Kontroll-Funktionen sind technisch völlig anders als die im **RTC6-Handbuch** beschriebenen "RTC-Kontrollbefehle" und werden daher in diesem Dokument nicht als solche bezeichnet. So stellt z.B. die **syncAXIS control-Instanz** sicher, dass die Kontroll-Funktion auch tatsächlich ausgeführt werden darf/kann.
- Kontroll-Funktionen (**slsc_ctrl_***) werden immer akzeptiert, wenn der Betriebsstatus "grün" ist (siehe **slsc_cfg_get_operation_status**).
- Für manche Kontroll-Funktionen (**slsc_ctrl_***) bestehen Einschränkungen bezüglich des **Betriebsmodus** (siehe enum **slsc_OperationMode**), siehe **Abbildung 37**, **Seite 98**.

Laser-bezogene Funktionen

- **slsc_ctrl_disable_laser**
- **slsc_ctrl_enable_laser**



Warnung!

Verletzungsgefahr durch Laserstrahlung!
slsc_cfg_initialize_from_file kann zu undefinierten Zuständen der RTC6-Karte(n) führen, bei denen unvorhergesehen der Laser eingeschaltet werden könnte! Stellen Sie vor dem Aufruf von **slsc_cfg_initialize_from_file** sicher, dass der Laser ausgeschaltet ist!



Vorsicht!

Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! Berücksichtigen Sie im Sicherheitskonzept Ihrer Anlagen-Steuerung, dass die Lasersteuersignale der RTC durch **slsc_cfg_initialize_from_file** und **slsc_ctrl_enable_laser** freigegeben werden.

slsc_cfg_initialize_from_file schaltet u.a. auch die Lasersteuerung auf der RTC6-Karte aktiv (scharf) (dazu wird intern der RTC6-Kontrollbefehl **set_laser_control** benutzt). Weiterhin führt **slsc_cfg_initialize_from_file** u.a. **slsc_ctrl_enable_laser** automatisch aus, d.h. die Lasersteuersignale⁽¹⁾ LASERON, LASER1 und LASER2 werden tatsächlich schon an der RTC6-Karte ausgegeben. Diese Ausgabe kann mit **slsc_ctrl_disable_laser** unterbunden werden.

(1) Siehe **RTC6-Handbuch**.

Ausführungs-bezogene Funktionen

Prüfen, ob der syncAXIS-DLL-Eingangspuffer voll ist und deshalb momentan keine weitere Job-Funktion (`slsc_list_*`) mehr aufnehmen kann:

- `slsc_ctrl_is_list_input_buffer_full`

Ausführstatus⁽¹⁾ der RTC6 abfragen:

- `slsc_ctrl_get_exec_state`

Starten und Abbrechen eines Jobs:

- `slsc_ctrl_start_execution`
- `slsc_ctrl_stop_controlled` oder `slsc_ctrl_stop`, siehe Abschnitt "Gegenüberstellung von `slsc_ctrl_stop_controlled` und `slsc_ctrl_stop`", Seite 97



Vorsicht!

Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! Berücksichtigen Sie im Sicherheitskonzept Ihrer Anlagen-Steuerung, dass bei der Job-Ausführung der Laser angeschaltet wird.



Vorsicht!

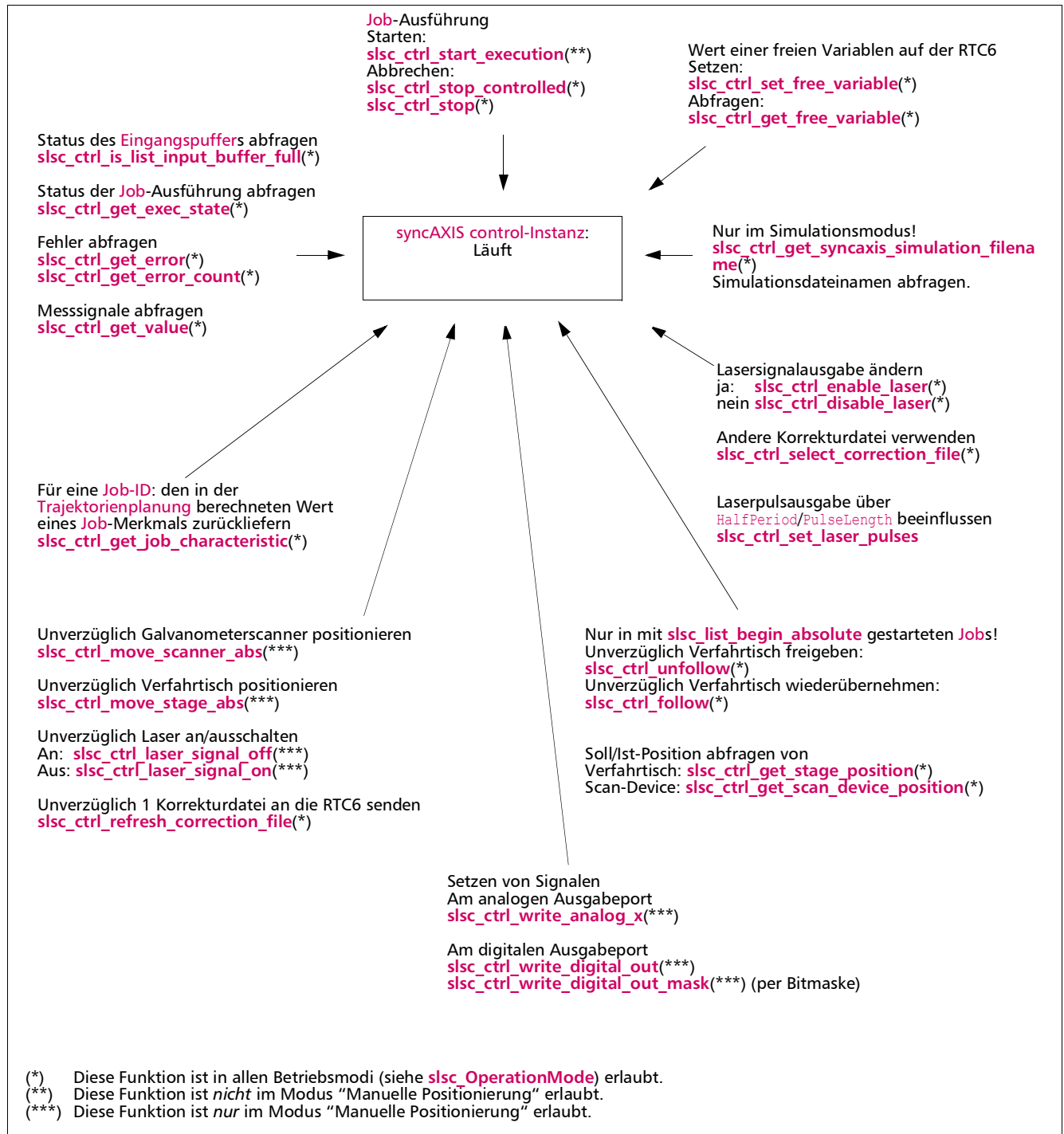
Von einem bewegten Verfahrtsch geht eine mechanische Gefahr aus. Es besteht Quetschgefahr für die Finger und Hände. Berücksichtigen Sie im Sicherheitskonzept Ihrer Anlagen-Steuerung, dass `slsc_ctrl_start_execution` u.U. den Verfahrtsch (eventuell auch verzögert) in Bewegung setzen kann. Achten Sie darauf, dass alle umstehenden Personen während der Ausführung ausreichend Abstand zu dem Gerät halten.

Gegenüberstellung von `slsc_ctrl_stop_controlled` und `slsc_ctrl_stop`

<code>slsc_ctrl_stop_controlled</code>	<code>slsc_ctrl_stop</code>
Mit Ausgleichsbewegung zum Abbremsen	"Not-Halt"
Gewährleistet keinen schnellstmöglichen Stillstand des Systems, hält aber sämtliche vorgegebenen Dynamikgrenzen ein	<ul style="list-style-type: none"> • Schaltet sofort den Laser aus • Bringt Spiegel sowie Verfahrtsch schnellstmöglich zum Stillstand • Bei häufiger Anwendung kann es zu Verschleiß kommen
Nachfolgend muss die <code>syncAXIS control-Instanz</code> neu initialisiert werden	Nachfolgend muss die <code>syncAXIS control-Instanz</code> neu initialisiert werden

(1) `slsc_ExecState_Idle`,
`slsc_ExecState_ReadyForExecution`,
`slsc_ExecState_Executing`,
`slsc_ExecState_NotInitOrError`.

Kontroll-Funktionen (slsc_ctrl_*)



Korrekturdatei-bezogene Funktionen

- **slsc_ctrl_refresh_correction_file**
- **slsc_ctrl_select_correction_file**

In einer **syncAXISConfig.xml** für **syncAXIS control** \geq V1.1.0 können bis zu 4 Korrekturdateien eingetragen sein⁽¹⁾. Beim Aufbauen der **syncAXIS control-Instanz** werden diese an die RTC6 übertragen und dort gespeichert.

Mit **slsc_ctrl_select_correction_file** kann (über die Angabe des entsprechenden Index 0...3 der gewünschten Korrekturdatei) eingestellt werden, welche dieser Korrekturdateien ab sofort verwendet werden soll (z. B. nachdem zu einer anderen Verfahrtsch-Scan-Kopf-Kombination gewechselt wurde und jetzt auf dieser markiert werden soll). **slsc_ctrl_select_correction_file** wählt also (ähnlich zum RTC6-Befehl **select_cor_table**) eine auf der RTC6 bereits vorhandene Korrekturdatei aus.

Im Gegensatz dazu überträgt **slsc_ctrl_refresh_correction_file** (ähnlich zum RTC6-Befehl **load_correction_file**) unverzüglich eine Korrekturdatei auf die RTC6 (dazu wird wie bei **slsc_ctrl_select_correction_file** der entsprechende Index 0...3 angegeben). Beispiel für einen Anwendungsfall: eine **syncAXIS control-Instanz** läuft, damit sind die **syncAXISConfig.xml** angegebenen Korrekturdateien auf der RTC6 gespeichert. Der Benutzer generiert für das Laser-Scan-System nun eine optimierte Korrekturdatei und überschreibt (im Dateisystem) damit die entsprechende Korrekturdatei, die momentan in Verwendung ist. Mit **slsc_ctrl_refresh_correction_file** wird dann (ohne die **syncAXIS control-Instanz** abbauen/neu aufbauen zu müssen) die optimierte Korrekturdatei an die RTC6 gesendet.

(1) **syncAXIS_1_8.xsd** erlaubt unterhalb von **CorrectionFileList** bis zu 4 **CorrectionFilePath** als Child-Tags.

Fehler-bezogene Funktionen

- **slsc_ctrl_get_error**
- **slsc_ctrl_get_error_count**

Zunächst muss mit **slsc_ctrl_get_error_count** die Anzahl n der vorliegenden Fehler festgestellt werden. Dann kann ein sinnvoller Wert für **ErrorNr** bei **slsc_ctrl_get_error** angegeben werden, um Details zum jeweiligen Fehler abzufragen. Der erste aufgetretene Fehler ist der älteste detektierte Fehler. Er hat die Nummer 0.

Funktionen zur Abfrage von Messwerten

- **slsc_ctrl_get_value**

Mit **slsc_ctrl_get_value** können zur Diagnose und Monitoring-Zwecken umfangreiche Messsignale zu den 4 Achsen ausgelesen werden.

Funktionen nur für den Modus "Manuelle Positionierung"

- **slsc_ctrl_laser_signal_off**
- **slsc_ctrl_laser_signal_on**
- **slsc_ctrl_move_scanner_abs**
- **slsc_ctrl_move_stage_abs**

Siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.

Funktionen zum Verwalten des Werts einer freien Variablen auf der RTC6

- **slsc_ctrl_get_free_variable**
- **slsc_ctrl_set_free_variable**

Die Funktionen für freie Variablen **slsc_ctrl_set_free_variable**, **slsc_ctrl_get_free_variable** sowie die ergänzende Job-Funktion (**slsc_list**) **slsc_list_set_free_variable** machen jeweils den direkt entsprechenden RTC6-Befehl in **syncAXIS control** verfügbar. Damit können z. B. Inkremente (innerhalb von **Jobs**) festgestellt und gezählt werden. Allerdings haben **slsc_ctrl_get_free_variable**, **slsc_ctrl_set_free_variable**, sowie **slsc_list_set_free_variable** im Simulationsmodus keine Auswirkung.

Weitere Informationen zu freien Variablen finden Sie im RTC6-Handbuch, Kapitel 6.9.1 "Freie Variablen", Seite 134.

Funktionen zur Parameterwert-Optimierung

- **slsc_ctrl_get_job_characteristic**

slsc_ctrl_get_job_characteristic ist hauptsächlich dazu vorgesehen, um über Algorithmen die Auswirkungen von Parameterpermutationen (im Simulationsmodus) auszuwerten (d.h. die Parameterwert-Optimierung automatisieren).

slsc_ctrl_get_job_characteristic liefert – für eine angegebene *Job-ID* – den in der *Trajektorienplanung* berechneten Wert *eines bestimmten Job-Merkmals* ("Key", siehe enum **slsc_JobCharacteristic**) zurück. Es können die letzten 10 berechneten *Jobs* (mittels *Job-ID*, siehe **slsc_list_begin**) abgefragt werden. Um alle *Job-Merkmale* eines *Jobs* abzufragen, muss **slsc_ctrl_get_job_characteristic** entsprechend oft aufgerufen werden.

Allerdings setzt **slsc_ctrl_get_job_characteristic** voraus, dass der Status der angegebenen *Job-ID* mindestens "Berechnung: Beendet" (siehe *Abbildung 12, Seite 43*) ist. Anderenfalls ist beim Rückgabewert *Bit #06* gesetzt (*UnplausibleOrUnknownParameter*). Deswegen kann **slsc_ctrl_get_job_characteristic** bei kurzen *Job-IDs* (nicht genauer quantifizierbar) auch genutzt werden, um Sicherheitsabfragen in einer GUI zu implementieren, d.h. wenn *Trajektorienplanungs-Berechnungsergebnisse* (z.B. maximale Ansteuerwerte für den Verfahrtsch) außerhalb der definierten Grenzen liegen (z.B. "...do you really want to execute Job...").

Funktionen zum Starten/Beenden des Modus "Manuelle Positionierung"

- **slsc_ctrl_follow**
- **slsc_ctrl_unfollow**

Siehe *Kapitel 2.12 "Über den Modus "Manuelle Positionierung""*, *Seite 70*.

Funktionen zur Abfrage von Positionen

- **slsc_ctrl_get_scan_device_position**
- **slsc_ctrl_get_stage_position**

Die Soll-Position oder Ist-Position des angegebenen Scan-Device gibt

slsc_ctrl_get_scan_device_position zurück, **slsc_ctrl_get_stage_position** entsprechend für den Verfahrtsch.

Simulationseinstellungs-bezogene Funktion

- **slsc_ctrl_get_syncaxis_simulation_filename**

slsc_ctrl_get_syncaxis_simulation_filename ersetzt **slsc_ctrl_get_simulation_filename**, weil Simulationsdateien in V1.3 nicht mehr Scan-Device-spezifisch generiert werden.

slsc_ctrl_get_syncaxis_simulation_filename setzt den Simulationsmodus voraus. Mit dieser Funktion kann der Simulationsdateiname (für eine bestimmte *Job-ID*) festgestellt werden. Das erleichtert die Realisierung von Anwenderprogrammen, die automatisch Simulationsdateien importieren sollen (z.B. um diese auszuwerten oder grafisch dazustellen).

Funktionen zum Setzen von Signalen

- **slsc_ctrl_write_analog_x**
- **slsc_ctrl_write_digital_out**
- **slsc_ctrl_write_digital_out_mask**

Diese Funktionen sind nur im Modus "Manuelle Positionierung" erlaubt. Sie werden nicht akzeptiert, wenn gerade ein *Job* ausgeführt wird. Jede dieser Funktionen wird immer an alle RTC6-Karten geschickt, die Signale unverzüglich gesetzt.

Die zuletzt gesetzten Signale werden weiterhin ausgegeben, auch wenn die *syncAXIS control-Instanz* mit **slsc_cfg_delete** gelöscht wird.

Funktion zum Beeinflussen der Laserpulsausgabe über HalfPeriod/PulseLength

- **slsc_ctrl_set_laser_pulses**

slsc_ctrl_set_laser_pulses sowie die ergänzende Job-Funktion (*slsc_list*) **slsc_list_set_laser_pulses** werden für diejenigen Benutzer bereitgestellt, die (aufgrund ihres eingesetzten Lasers) die "Automatische Lasersteuerung" zum Erreichen äquidistanter Spotabstände nicht nutzen können und stattdessen die Pulsausgabe über *HalfPeriod* und *PulseLength* beeinflussen wollen.

Allerdings haben **slsc_ctrl_set_laser_pulses** sowie **slsc_list_set_laser_pulses** im Simulationsmodus keine Auswirkung.

3.1.4 Utility-Funktionen (slsc_util_*)

Utility-Funktionen (Präfix **slsc_util_**)⁽¹⁾⁽²⁾ ersetzen im Wesentlichen externe Dienstprogramme und erfordern die Einhaltung bestimmter Voraussetzungen. Sie dürfen nicht während des regulären syncAXIS control-Betriebs aufgerufen werden. Einen grafischen Überblick gibt **Abbildung 38**, **Seite 101**.

RTC6 PCI-Express-Karten in einem Zustand befindet, der vom Benutzer als "eigenartig"⁽³⁾ empfunden wird.

RTC6-Karten-bezogene Funktion

- **slsc_util_reset_pcie**



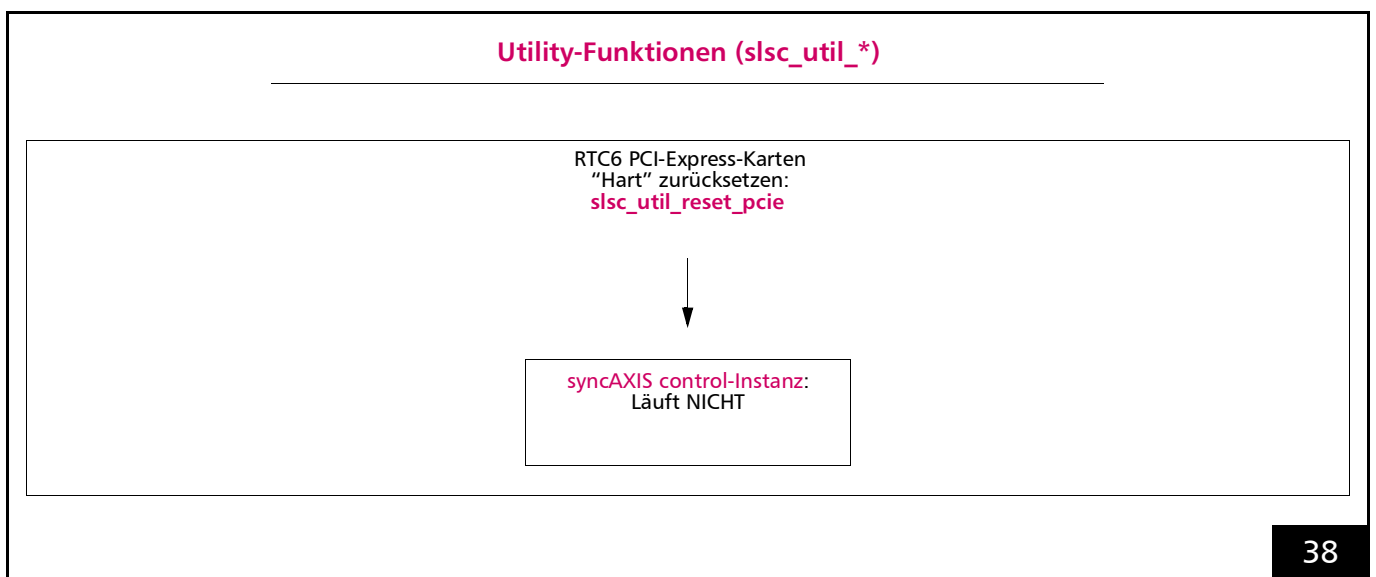
Warnung!

Verletzungsgefahr durch Laserstrahlung!
slsc_util_reset_pcie kann zu undefinierten Zuständen der RTC6-Karte(n) führen, bei denen unvorhergesehen der Laser eingeschaltet werden könnte! Stellen Sie vor dem Aufruf von **slsc_util_reset_pcie** sicher, dass der Laser ausgeschaltet ist!

Außer dem vorhergehenden Sicherheitshinweis ist vor dem Aufruf von **slsc_util_reset_pcie** sicherzustellen, dass gerade keine syncAXIS control-Instanz läuft. **slsc_util_reset_pcie** ist eine "harte" Reset-Funktion, für den Fall, dass sich eine der

- (1) Verfügbar in syncAXIS control ≥ V1.3.0.
(2) Derzeit (V1.3.0) nur eine.

- (3) In dieser Situation ist vom Gebrauch von **iSCANcfg.exe** abzuraten, da meist RTC6-Dateien geladen werden, die nicht aus dem syncAXIS control-Software-Paket stammen und es damit unvorhersehbaren Nebenwirkungen kommt.



Utility-Funktionen (slsc_util_*): Grafischer Überblick.

3.2 Alphabetische Übersicht

In diesem Kapitel:

- Konfigurations-Funktionen (slsc_cfg_*), Seite 102
- Kontroll-Funktionen (slsc_ctrl_*), Seite 107
- Job-Funktionen (slsc_list_*), Seite 109
- Utility-Funktionen (slsc_util_*), Seite 113

Konfigurations-Funktionen (slsc_cfg_*)	Zweck
slsc_cfg_acquire_stage (veraltet)	Veraltet.
slsc_cfg_delete	Baut die angegebene syncAXIS control-Instanz ab. Dabei werden auch die Ressourcen (RTC6-Karte, Verfahrtsch, ...) wieder freigegeben.
slsc_cfg_delete_trajectory_config	Hilfsfunktion für die Softwareentwicklung: Löscht das Trajektorienkonfigurations-Objekt (zur Vermeidung von Speicherlecks), siehe Code-Beispiel.
slsc_cfg_get_calculation_dynamics_stage	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Verfahrtsch-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Verfahrtsch-Bewegung verwendet.
slsc_cfg_get_dynamic_limits_scan_device	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Scan-Device-Typs.
slsc_cfg_get_dynamic_limits_stage	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Die Dynamikgrenzen des vorgesehenen Verfahrtsch-Typs.
slsc_cfg_get_dynamic_violation_reaction	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Die Reaktion bei einer Dynamik-Grenzwertüberschreitung.
slsc_cfg_get_field_limits_scan_device	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Arbeitsfeldgrenzen des vorgesehenen Scan-Device-Typs.
slsc_cfg_get_field_limits_stage	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Die Arbeitsfeldgrenzen des vorgesehenen Verfahrtsch-Typs.
Alphabetische Übersicht, Seite 102	

Konfigurations-Funktionen (slsc_cfg_*) (Forts.)	Zweck (Forts.)
slsc_cfg_get_calculation_dynamics_jump_scan_device	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Sprünge, aber nicht Markierungen.
slsc_cfg_get_calculation_dynamics_mark_scan_device	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Markierungen, aber nicht Sprünge.
slsc_cfg_get_jump_time	Berechnet die Dauer eines Sprungs (außerhalb "normaler" Jobs): <ul style="list-style-type: none"> • Basierend auf der momentanen Einstellung der angegebenen syncAXIS control-Instanz • Sowie in Abhängigkeit der angegebenen Werte für Start-Dynamik und End-Dynamik
slsc_cfg_get_mode	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Den Betriebsmodus (ScannerOnly, StageOnly, ScannerAndStage).
slsc_cfg_get_operation_status	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Den Betriebsstatus ("Ampelfarbe").
slsc_cfg_get_scan_device_dynamic_monitoring_level	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Das Kriterium, auf das die Scan-Devices hin überwacht werden sollen (z.B. slsc_DynamicsMonitoringLevel_Velocity).
slsc_cfg_get_simulation_setting	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Die Simulationseinstellung.
slsc_cfg_get_stage_dynamic_monitoring_level	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Das Kriterium, auf das die Verfahrtische hin überwacht werden sollen (z.B. slsc_DynamicsMonitoringLevel_Velocity).
slsc_cfg_get_sync_axis_version	Gibt Versionsinformationen über die aktuell laufende syncAXIS-DLL zurück.
slsc_cfg_get_trajectory_config	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: Die Konfiguration der Trajektorienplanung.
Alphabetische Übersicht, Seite 102	

Konfigurations-Funktionen (slsc_cfg_*) (Forts.)	Zweck (Forts.)
slsc_cfg_initialize_copy	Initialisierungsfunktion: Baut eine neue (Ziel-)syncAXIS control-Instanz im Simulationsmodus mit der momentanen Konfiguration der angegebenen (Quell-)syncAXIS control-Instanz (entweder im Hardwaremodus oder Simulationsmodus) auf und ordnet ihr einen eindeutigen Handle-Wert zu.
slsc_cfg_initialize_from_file	Initialisierungsfunktion: Baut eine neue syncAXIS control-Instanz (unter Verwendung der angegebenen XML-Konfigurationsdatei) auf und teilt ihr einen eindeutigen Handle-Wert zu.
slsc_cfg_register_callback_job_end_planned	Stellt ein, dass die angegebene "Callback-Funktion" bei einem "Callback-Event" des Typs "job_end_planned" aufgerufen wird.
slsc_cfg_register_callback_job_finished_executing	Stellt ein, dass die angegebene "Callback-Funktion" bei einem "Callback-Event" des Typs "job_finished_executing" aufgerufen wird.
slsc_cfg_register_callback_job_is_executing	Stellt ein, dass die angegebene "Callback-Funktion" bei einem "Callback-Event" des Typs "job_is_executing" aufgerufen wird.
slsc_cfg_register_callback_job_loaded_enough	Stellt ein, dass die angegebene "Callback-Funktion" bei einem "Callback-Event" des Typs "job_loaded_enough" aufgerufen wird.
slsc_cfg_register_callback_job_progress_planned	Stellt ein, dass die angegebene "Callback-Funktion" bei einem "Callback-Event" des Typs "job_progress_planned" aufgerufen wird.
slsc_cfg_register_callback_job_start_planned	Stellt ein, dass die angegebene "Callback-Funktion" bei einem "Callback-Event" des Typs "job_start_planned" aufgerufen wird.
slsc_cfg_reinitialize	Initialisierungsfunktion: Baut die (über das Handle) angegebene syncAXIS control-Instanz ab und dann wieder (unter Verwendung der zuvor ausgelesenen, momentanen Konfigurationseinstellungen und -werte) neu auf. Der Handle-Wert bleibt dabei unverändert.
slsc_cfg_reinitialize_from_file	Initialisierungsfunktion: Baut die (über das Handle) angegebene syncAXIS control-Instanz ab und dann wieder (unter Verwendung der angegebenen XML-Konfigurationsdatei) neu auf. Der Handle-Wert bleibt dabei unverändert.
slsc_cfg_release_stage (veraltet)	Veraltet.
slsc_cfg_select_heuristic	Zum Angeben der Kennlinie für die Geschwindigkeitsverminderung (DynamicReductionFunction).
Alphabetische Übersicht, Seite 102	

Konfigurations-Funktionen (slsc_cfg_*) (Forts.)	Zweck (Forts.)
slsc_cfg_select_stage	Zum Angeben des Ziel-Verfahrtischs ("Verfahrtisch-Wechsel"). slsc_cfg_select_stage ersetzt ab syncAXIS-DLL ≥ V1.2.0 slsc_cfg_select_stage_axis (veraltet).
slsc_cfg_select_stage_axis (veraltet)	Veraltet.
slsc_cfg_set_bandwidth	Ändert den FilterBandwidth-Wert der angegebenen syncAXIS control-Instanz.
slsc_cfg_set_calculation_dynamics_jump_scan_device	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Den Höchstwert von Beschleu- nigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Sprünge, aber nicht Markierungen.
slsc_cfg_set_calculation_dynamics_mark_scan_device	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Den Höchstwert von Beschleu- nigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Markie- rungen, aber nicht Sprünge.
slsc_cfg_set_calculation_dynamics_stage	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Verfahrtisch-Typs. Die Werte werden nur in Trajektorienpla- nungs-Berechnungen der Verfahrtisch-Bewegung verwendet.
slsc_cfg_set_contour_dependent_speed_control_2d	"Konturabhängige Geschwindigkeitsberechnung" an-/ ausschalten. Außerdem wird eingestellt, wie (die syncAXIS control-Instanz intern) die Geschwindigkeiten entlang von Kurven bestimmt ("links" oder "rechts" der Kurven- linien-Mitte; Abstand zu dieser). Bei aktivierter "Automatische Lasersteuerung" werden diese Ergebnisse dazu verwendet, um entsprechend dazu z. B. die Laser-Spot-Abstände äquidistant zu setzen.
slsc_cfg_set_dynamic_limits_scan_device	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Scan-Device-Typs.
slsc_cfg_set_dynamic_limits_stage	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Dynamikgrenzen des vorgese- henen Verfahrtisch-Typs.
slsc_cfg_set_dynamic_violation_reaction	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Reaktion bei einer Dynamik- Grenzwertüberschreitung.
Alphabetische Übersicht, Seite 102	

Konfigurations-Funktionen (slsc_cfg_*) (Forts.)	Zweck (Forts.)
slsc_cfg_set_field_limits_scan_device	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Arbeitsfeldgrenzen des vorgesehenen Scan-Device-Typs.
slsc_cfg_set_field_limits_stage	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Arbeitsfeldgrenzen des vorgesehenen Verfahrtschicht-Typs.
slsc_cfg_set_jump_speed	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Sprunggeschwindigkeit.
slsc_cfg_set_list_handling_mode	Stellt das Handling und das Rückgabeverhalten der Job-Funktionen (slsc_list_*) ein.
slsc_cfg_set_list_handling_mode_with_context	Wie slsc_cfg_set_list_handling_mode. Aber es kann zusätzlich ein Kontext angegeben werden. Stellt das Handling und das Rückgabeverhalten der Job-Funktionen (slsc_list_*) ein.
slsc_cfg_set_mark_speed	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Markiergeschwindigkeit.
slsc_cfg_set_matrix_and_offset	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Zielpunkt-Koordinaten gemäß einer Transformationsmatrix und einem Offset-Wert.
slsc_cfg_set_mode	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Den Betriebsmodus (ScannerOnly, StageOnly, ScannerAndStage).
slsc_cfg_set_part_displacement	Wendet eine Matrix und einen Offset auf die Soll-Trajektorie für das angegebene Scan-Device (Scan-Kopf) an. Siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332.
slsc_cfg_set_rot_and_offset_2d	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Zielpunkt-Koordinaten um einen Winkel und Offset-Wert.
slsc_cfg_set_scan_device_dynamic_monitoring_level	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Das Kriterium, auf das die Scan-Devices hin überwacht werden sollen (z.B. slsc_DynamicsMonitoringLevel_Velocity).
slsc_cfg_set_simulation_setting	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Simulationseinstellung.
slsc_cfg_set_stage_dynamic_monitoring_level	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Das Kriterium, auf das die Verfahrtschichten hin überwacht werden sollen (z.B. slsc_DynamicsMonitoringLevel_Velocity).
slsc_cfg_set_trajectory_config	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: Die Konfiguration der Trajektorienplanung.
Alphabetische Übersicht, Seite 102	

Kontroll-Funktionen (slsc_ctrl_*)	Zweck
slsc_ctrl_disable_laser	Unterbindet, dass die Lasersteuersignale LASERON, LASER1 und LASER2 (siehe RTC6-Handbuch) an der RTC6-Karte ausgegeben werden.
slsc_ctrl_enable_laser	Gibt die Lasersteuersignale LASERON, LASER1 und LASER2 (siehe RTC6-Handbuch) an der RTC6-Karte frei.
slsc_ctrl_follow	Zum Wieder-Übernehmen des Verfahrtisches nach einem slsc_ctrl_unfollow.
slsc_ctrl_get_error	Gibt Informationen zu einem aufgetretenen Fehler (Fehlernummer <code>ErrorNr</code>) zurück.
slsc_ctrl_get_error_count	Gibt die Anzahl der vorliegenden Fehler zurück.
slsc_ctrl_get_exec_state	Gibt den Status des Execution Layer zurück.
slsc_ctrl_get_free_variable	Gibt den aktuellen Wert einer freien Variable der RTC6 zurück.
slsc_ctrl_get_job_characteristic	Gibt – für eine angegebene Job-ID – den in der Trajektorienplanung berechneten Wert eines Job-Merkmals ("Key", siehe enum <code>slsc_JobCharacteristic</code>) zurück.
slsc_ctrl_get_scan_device_position	Gibt die Soll-Position oder Ist-Position des angegebenen Scan-Device (Scan-Kopf) zurück.
slsc_ctrl_get_simulation_filename	Veraltet.
slsc_ctrl_get_stage_position	Gibt die Soll-Position oder Ist-Position des Verfahrtisches zurück.
slsc_ctrl_get_syncaxis_simulation_filename	Nur im Simulationsmodus! Gibt für eine angegebene Job-ID den entsprechenden Simulationsdateinamen zurück.
slsc_ctrl_get_value	Gibt den aktuellen Wert des angegebenen Signals der angegebenen Achse zurück.
slsc_ctrl_is_list_input_buffer_full	Prüft, ob der syncAXIS-DLL-Eingangspuffer voll ist (und deshalb momentan keine weitere Job-Funktion (slsc_list_*) mehr aufnehmen kann).
slsc_ctrl_laser_signal_off	Nur im Modus "Manuelle Positionierung": Schaltet den Laser unverzüglich aus.
slsc_ctrl_laser_signal_on	Nur im Modus "Manuelle Positionierung": Schaltet den Laser unverzüglich an.
Alphabetische Übersicht, Seite 102	

Kontroll-Funktionen (slsc_ctrl_*) (Forts.)	Zweck (Forts.)
slsc_ctrl_move_scanner_abs	Nur im Modus "Manuelle Positionierung": Bringt alle Scan-Devices (ausgehend von der aktuellen Position) mit Sprunggeschwindigkeit in die angegebene Position.
slsc_ctrl_move_stage_abs	Nur im Modus "Manuelle Positionierung": Bringt den Verfahrtsch (ausgehend von der aktuellen Position) mit den über die ACS-API eingestellten Dynamiken (Beschleunigung und Ruck) in die angegebene Position.
slsc_ctrl_refresh_correction_file	Überträgt unverzüglich eine Korrekturdatei an die RTC6-Karte.
slsc_ctrl_select_correction_file	Zum Angeben einer Korrekturdatei, die unverzüglich verwendet werden soll.
slsc_ctrl_set_free_variable	Setzt auf der RTC6 den Wert einer freien Variable.
slsc_ctrl_set_laser_pulses	Definiert die Ausgabeperiode und die Pulslänge der Lasersignale LASER1 und LASER2 für den "Laser active"-Betrieb der RTC6-Karte.
slsc_ctrl_start_execution	Versucht, die Ausführung eines Jobs durch das Execution Layer zu starten.
slsc_ctrl_stop	Beendet die Ausführung des aktuellen Jobs unkontrolliert und sofort über einen direkten Zugriff auf die RTC6-Karte ("Not-Halt").
slsc_ctrl_stop_controlled	Beendet die Ausführung des aktuellen Jobs kontrolliert durch Einfügen einer Ausgleichsbewegung zum Abbremsen.
slsc_ctrl_unfollow	Die angegebene syncAXIS control-Instanz gibt vorübergehend den Verfahrtsch frei. Dieser kann nun extern (z.B. durch ein nicht-syncAXIS control-basiertes Anwenderprogramm) gesteuert werden.
slsc_ctrl_write_analog_x	Nur im Modus "Manuelle Positionierung"! Schreibt einen Ausgabewert auf den 12-Bit-Analogausgang ANALOG OUT1 oder ANALOG OUT2 aller RTC6-Karten.
slsc_ctrl_write_digital_out	Nur im Modus "Manuelle Positionierung"! Schreibt einen 16-Bit-Ausgabewert auf den 16-Bit-Digital-Ausgang DIGITAL OUT 0...DIGITAL OUT 15 aller RTC6-Karten.
slsc_ctrl_write_digital_out_mask	Nur im Modus "Manuelle Positionierung"! Schreibt nur diejenigen Bits des Value-Werts auf den 16-Bit-Digital-Ausgang aller RTC6-Karten, die in der benutzerdefinierten Bitmaske (Parameter Mask) festgelegt sind.
Alphabetische Übersicht, Seite 102	

Job-Funktionen (slsc_list_*)	Zweck
slsc_list_arc_abs	Definiert einen zu markierenden Kreisbogen (nicht: Ellipsenbogen) mittels absoluter Koordinatenwerte.
slsc_list_begin	Definiert den Beginn eines Jobs. Ist eines von 2 verpflichtenden Strukturelementen eines Jobs.
slsc_list_begin_absolute	Definiert (alternativ zu slsc_list_begin, slsc_list_begin_relative) den Beginn eines Jobs, um eine (im Modus "Manuelle Positionierung") durch slsc_ctrl_move_stage_abs verursachte Positionsänderung des Verfahrtschis ausgleichen zu können.
slsc_list_begin_module	Nur im Simulationsmodus erlaubt. Zur "Vorbereitung eines Jobs": Definiert den Anfang eines aufzuzeichnenden Jobs (Modul). Dieser wird wie üblich mit slsc_list_end geschlossen.
slsc_list_begin_relative	Definiert (alternativ zu slsc_list_begin) den Beginn eines Jobs. Ist eines von 2 verpflichtenden Strukturelementen eines Jobs.
slsc_list_circle_2d_abs	Definiert einen Kreis (nicht: Ellipse) über den absoluten Koordinatenwert des Kreismittelpunkts. Der Parameter <i>Angle</i> bestimmt die Richtung, mit der die Markierung ausgeführt wird, sowie die Anzahl der Umdrehungen (z. B. $3,25 \times 2\pi$).
slsc_list_dashed_arc_abs	Wie slsc_list_arc_abs, aber die korrespondierende [*]dashed[*]-Funktion. Bietet daher zusätzlich die Argumente <i>NSwitches</i> und <i>LaserSwitches</i> , siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
slsc_list_dashed_circle_2d_abs	Wie slsc_list_circle_2d_abs, aber die korrespondierende [*]dashed[*]-Funktion. Bietet daher zusätzlich die Argumente <i>NSwitches</i> und <i>LaserSwitches</i> , siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
slsc_list_dashed_mark_abs	Wie slsc_list_mark_abs, aber die korrespondierende [*]dashed[*]-Funktion. Bietet daher zusätzlich die Argumente <i>NSwitches</i> und <i>LaserSwitches</i> , siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
slsc_list_end	Definiert das Ende eines Jobs. Ist eines von 2 verpflichtenden Strukturelementen eines Jobs.
slsc_list_jump_abs	Definiert einen Sprung mittels absoluter Koordinatenwerte.
slsc_list_jump_abs_min_time	Wie slsc_list_jump_abs. Erlaubt aber zusätzlich, eine Mindestdauer für den Sprung anzugeben.
slsc_list_mark_abs	Definiert einen Markiervektor mittels absoluter Koordinatenwerte.
Alphabetische Übersicht, Seite 102	

Job-Funktionen (slsc_list_*) (Forts.)	Zweck (Forts.)
slsc_list_multi_para_arc_abs	Wie slsc_list_arc_abs . Bietet aber zusätzlich das Argument MultiParaTarget , über das (je "ActiveChannel") eine aus mehreren Abschnitten bestehende Rampe definiert wird.
slsc_list_multi_para_circle_2d_abs	Wie slsc_list_circle_2d_abs . Bietet aber zusätzlich das Argument MultiParaTarget , über das (je "ActiveChannel") eine aus mehreren Abschnitten bestehende Rampe definiert wird.
slsc_list_multi_para_dashed_arc_abs	Wie slsc_list_multi_para_arc_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
slsc_list_multi_para_dashed_circle_2d_abs	Wie slsc_list_multi_para_circle_2d_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
slsc_list_multi_para_dashed_mark_abs	Wie slsc_list_multi_para_mark_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
slsc_list_multi_para_mark_abs	Wie slsc_list_mark_abs . Bietet aber zusätzlich das Argument MultiParaTarget , über das (je "ActiveChannel") eine aus mehreren Abschnitten bestehende Rampe definiert wird.
slsc_list_para_arc_abs	Wie slsc_list_arc_abs . Bietet aber zusätzlich das Argument ParaTarget , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
slsc_list_para_circle_2d_abs	Wie slsc_list_circle_2d_abs . Bietet aber zusätzlich das Argument ParaTarget , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
slsc_list_para_dashed_arc_abs	Wie slsc_list_para_arc_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
slsc_list_para_dashed_circle_2d_abs	Wie slsc_list_para_circle_2d_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
slsc_list_para_dashed_mark_abs	Wie slsc_list_para_dashed_mark_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Alphabetische Übersicht, Seite 102	

Job-Funktionen (slsc_list_*) (Forts.)	Zweck (Forts.)
slsc_list_para_disable	Schaltet die Verarbeitung der Argumente ParaTarget (der slsc_list_para *-Funktionen) und MultiParaTarget (der slsc_list_multi_para *-Funktionen) aus.
slsc_list_para_enable	Schaltet die Verarbeitung der Argumente ParaTarget (der slsc_list_para *-Funktionen) und MultiParaTarget (der slsc_list_multi_para *-Funktionen) ein.
slsc_list_para_jump_abs	Wie slsc_list_jump_abs . Bietet aber zusätzlich das Argument ParaTarget , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
slsc_list_para_jump_abs_min_time	Wie slsc_list_jump_abs_min_time . Bietet aber zusätzlich das Argument ParaTarget , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
slsc_list_para_mark_abs	Wie slsc_list_mark_abs . Bietet aber zusätzlich das Argument ParaTarget , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
slsc_list_para_playback_module	Wie slsc_list_playback_module , aber es werden Parameter-Werte zu Rampen angewendet, wenn ein slsc_list_para_enable vorausgeht.
slsc_list_playback_module	Integriert ("spielt ab") ein Modul in den aktuellen Job. Parameter-Werte zu Rampen werden nicht angewendet.
slsc_list_set_approx_blend_limit	Ändert den ApproxBlendLimit -Wert, der in der Konfiguration der Trajektorienplanung festgelegt ist (s.u.). Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_set_calculation_dynamics_jump_scan_device	Ändert: Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Sprünge, aber nicht Markierungen.
slsc_list_set_calculation_dynamics_mark_scan_device	Ändert: Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Markierungen, aber nicht Sprünge.
Alphabetische Übersicht, Seite 102	

Job-Funktionen (slsc_list_*) (Forts.)	Zweck (Forts.)
slsc_list_set_contour_dependent_speed_control_2d	“Konturabhängige Geschwindigkeitsberechnung” an-/ausschalten. Außerdem kann geändert werden, wie (die syncAXIS control-Instanz intern) die Geschwindigkeiten entlang von Kurven bestimmt (“links” oder “rechts” der Kurvenlinien-Mitte; Abstand zu dieser). Bei aktivierter “Automatische Lasersteuerung” werden diese Ergebnisse dazu verwendet, um entsprechend dazu z. B. die Laser-Spot-Abstände äquidistant zu setzen. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_set_free_variable	Wie slsc_ctrl_set_free_variable.
slsc_list_set_jump_speed	Ändert die Sprunggeschwindigkeit. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_set_laser_on_move	Verzögert den “Laser active”-Betrieb um genau die Zeitdauer, die gebraucht wird, um die angegebene Pfadlänge (PathLength) auf dem aktuellen Markierabschnitt abzufahren. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_set_laser_pulses	Wie slsc_ctrl_set_laser_pulses.
slsc_list_set_mark_speed	Ändert die Markiergeschwindigkeit. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_set_matrix_and_offset	Ändert Zielpunkt-Koordinaten entsprechend einer Transformationsmatrix und einem Offset-Wert. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_set_min_mark_speed	Ändert die minimale Markiergeschwindigkeit, siehe MinimalMarkSpeed. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_set_rot_and_offset_2d	Ändert Zielpunkt-Koordinaten um einen Winkel und Offset-Wert. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
slsc_list_suppress_spotdistance_control	Nur wenn die “Automatische Lasersteuerung” aktiv ist mit SpotDistance als “ActiveChannel”: Zusatzfunktion, die slsc_list_wait_with_laser_on vorausgehen muss.
slsc_list_unsuppress_spotdistance_control	Hebt die Wirkung von slsc_list_suppress_spotdistance_control wieder auf.
Alphabetische Übersicht, Seite 102	

Job-Funktionen (slsc_list_*) (Forts.)	Zweck (Forts.)
<code>slsc_list_wait_with_laser_off</code>	Wie <code>slsc_list_wait_with_laser_on</code> , aber der Laser ist ausgeschaltet.
<code>slsc_list_wait_with_laser_on</code>	Definiert eine Wartezeit, mit der der Laserspot am zuletzt definierten Zielpunkt mit angeschaltetem Laser warten soll.
<code>slsc_list_write_analog_x</code>	Schreibt einen Ausgabewert auf den 12-Bit-Analogausgang ANALOG OUT1 oder ANALOG OUT2 aller RTC6-Karten.
<code>slsc_list_write_digital_out</code>	Schreibt einen 16-Bit-Ausgabewert auf den 16-Bit-Digital-Ausgang DIGITAL OUT 0...DIGITAL OUT 15 aller RTC6-Karten.
<code>slsc_list_write_digital_out_mask</code>	Schreibt nur diejenigen Bits des <code>Value</code> -Werts auf den 16-Bit-Digital-Ausgang aller RTC6, die in der benutzerdefinierten Bitmaske (Parameter <code>Mask</code>) festgelegt sind.
Alphabetische Übersicht, Seite 102	

Utility-Funktionen (slsc_util_*)	Zweck
<code>slsc_util_reset_pcie</code>	Setzt alle gefundenen RTC6 PCI-Express-Karten "hart" zurück.
Alphabetische Übersicht, Seite 102	

3.3 Funktionsreferenz

In diesem Kapitel:

- Kapitel 3.3.1 "Allgemeiner Aufbau der Referenztabellen", Seite 114
- Kapitel 3.3.2 "Datentypen der syncAXIS-DLL-Funktionen", Seite 115
- Kapitel 3.3.3 "Referenztabellen", Seite 117

3.3.1 Allgemeiner Aufbau der Referenztabellen

Name der Funktion	<p>präfix_name</p> <p>Das Präfix kennzeichnet die Kategorie der Funktion:</p> <p>slsc_cfg_ – Konfigurations-Funktion</p> <p>slsc_ctrl_ – Kontroll-Funktion</p> <p>slsc_list_ – Job-Funktion</p>
Zweck	Kurze Beschreibung wozu die Funktion geschrieben wurde.
Funktions-Signatur	<pre>datatype präfix name(datatype A, datatype* B, datatype C);</pre> <p> > Zeile Argument(e) C</p> <p> > Zeile Argument(e) B^(a)</p> <p> > Zeile Argument(e) A</p> <p> > Zeilen Name der Funktion, Zweck</p> <p>> Zeile Rückgabewert</p> <p>Beispiel: <code>uint32_t slsc_list_arc_abs(size_t Handle, const double* Mid, const double* Target);</code></p>
Argument(e)	A Datentyp. Kurztext.
	B Datentyp. Kurztext.
	C Datentyp. Kurztext.
Rückgabewert	Verweis auf eine Beschreibung des Rückgabewerts, z.B. "Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279".
Kommentar(e)	<ul style="list-style-type: none"> • Weitere Informationen zu dieser und verwandten Funktionen. • Verweise auf andere Kapitel und Dokumente.
Code-Beispiel	<p>Beispielhafter Quellcode-Schnipsel (nicht kompilierbar).</p> <pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.</pre>
Versionsinfo	Nennt die Version der syncAXIS-DLL, in der die Funktion erstmalig zur Verfügung steht, sowie ggf. auch weitere Informationen zu Änderungen.
Verweise	Links zu verwandten Funktionen: präfix_name_2

(a) 'datatype*' (Adressoperator) kennzeichnet einen Zeiger.

3.3.2 Datentypen der syncAXIS-DLL-Funktionen

Programmiersprache C	Datenformat
bool	Bool-Wert (true, false).
bool*	Zeiger auf einen Bool-Wert (true, false).
char	Ein darstellbares Zeichen (character) aus 1 Byte = 8 Bit.
char*	Zeiger auf einen nullterminierten ANSI-String, 1 Byte pro Zeichen. 4 Byte für Win32 Executables. 8 Byte für Win64 Executables. Synonym: Char-Array, C-String.
double	64-Bit IEEE Gleitkommaformat. Siehe https://de.wikipedia.org/wiki/IEEE_754 .
double*	Zeiger auf einen double-Wert. double* kann auch ein Array sein.
size_t	Wie definiert in der <code>stddef.h</code> . In der Regel <code>uint32_t</code> für Win32 Executables.
size_t*	Zeiger auf einen size_t-Wert.
slsc_ExecTimeCallback	<p>Hilfsdatentyp. Gibt die Signatur für die (vom Benutzer zur Verfügung zu stellende) "Callback-Funktion" vor, die angegeben werden muss bei:</p> <ul style="list-style-type: none"> • <code>slsc_cfg_register_callback_job_end_planned</code> • <code>slsc_cfg_register_callback_job_finished_executing</code> • <code>slsc_cfg_register_callback_job_is_executing</code> • <code>slsc_cfg_register_callback_job_progress_planned</code> <p>Funktions-Signatur:</p> <pre>typedef void(*slsc_ExecTimeCallback)(size_t JobID, uint64_t Progress, double ExecTime, void* Context);</pre> <p>Argument(e):</p> <p>JobID Job-ID.</p> <p>Progress Reserviert.</p> <p>ExecTime Ausführungszeit dieser Job-ID bis hierhin in ms. Diese Informationen über die Dauer geplanter oder ausgeführter Bewegungen können zur Prozessauswertung und Prozessoptimierung verwendet werden. Beachten Sie, dass die gemessenen Zeiten schwanken werden, weil MS Windows kein Echtzeitsystem ist.</p> <p>Context Zeiger auf das Objekt, das in der jeweiligen <code>cfg_register_callback</code>-Funktion referenziert wurde.</p> <p>Kommentar(e):</p> <ul style="list-style-type: none"> • <code>slsc_ExecTimeCallback</code> kann z. B. als Cast für Funktions-Zeiger oder als Typ für Lambda-Funktionen verwendet werden.

Programmiersprache C	Datenformat
<code>slsc_JobCallback</code>	<p>Hilfsdatentyp. Gibt die Signatur für die (vom Benutzer zur Verfügung zu stellende) "Callback-Funktion" vor, die angegeben werden muss bei:</p> <ul style="list-style-type: none"> • <code>slsc_cfg_register_callback_job_loaded_enough</code> • <code>slsc_cfg_register_callback_job_start_planned</code> <p>Funktions-Signatur:</p> <pre>typedef void (*slsc_JobCallback) (size_t JobID, void* Context);</pre> <p>Argument(e):</p> <p>JobID Job-ID.</p> <p>Context Zeiger auf das Objekt, das in der jeweiligen cfg_register_callback-Funktion referenziert wurde.</p> <p>Kommentar(e):</p> <ul style="list-style-type: none"> • <code>slsc_JobCallback</code> kann z.B. als Cast für Funktions-Zeiger oder als Typ für Lambda-Funktionen verwendet werden.
<code>uint16_t</code>	Synonym: unsigned short. 16-Bit-Wert ohne Vorzeichen: $[0 \dots + (2^{16} - 1)]$.
<code>uint32_t</code>	Synonym: unsigned int. 32-Bit-Wert ohne Vorzeichen: $[0 \dots + (2^{32} - 1)]$.
<code>uint64_t</code>	Synonym: unsigned long long. 64-Bit-Wert ohne Vorzeichen: $[0 \dots + (2^{64} - 1)]$.
<code>uint32_t*</code>	Zeiger auf einen 32-Bit-Wert ohne Vorzeichen: $[0 \dots + (2^{32} - 1)]$.
<code>uint64_t*</code>	Zeiger auf einen 64-Bit-Wert ohne Vorzeichen: $[0 \dots + (2^{64} - 1)]$.

Hinweise

- `**`
bedeutet Zeiger auf einen Zeiger, z. B. bei **`slsc_cfg_get_trajectory_config`**.
- `const`
(z. B. bei `const double* Target`) bedeutet, dass der nachfolgend stehende Wert nicht veränderbar ist. D.h. nach dem Aufruf der Funktion ist der Wert gleich wie vorher (im Gegensatz zu `size_t*`). `const` wird verwendet um diese Werte zu Parameterrückgaben abzugrenzen.
- `void`
bedeutet, dass die Funktion keinen Rückgabewert liefert.
- `void*`
bedeutet einen Zeiger auf einen generischen Datentyp.

3.3.3 Referenztabellen

Die Abfolge der Referenztabellen in diesem Kapitel ist alphabetisch.

Name der Funktion	slsc_cfg_acquire_stage (veraltet)
Zweck	Veraltet. Verwenden Sie stattdessen slsc_ctrl_follow/slsc_ctrl_unfollow . Versetzt die angegebene syncAXIS control-Instanz (nach slsc_cfg_release_stage (veraltet)) wieder in einen voll funktionsfähigen Zustand zurück. RTC6-Karte und Verfahrtisch werden wieder übernommen.
Funktions-Signatur	<code>uint32_t slsc_cfg_acquire_stage (veraltet)(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Vor slsc_cfg_acquire_stage (veraltet) sollte slsc_cfg_release_stage (veraltet) aufgerufen worden sein. • slsc_cfg_acquire_stage (veraltet) versetzt die angegebene syncAXIS control-Instanz (nach slsc_cfg_release_stage (veraltet)) wieder in einen voll funktionsfähigen Zustand zurück. Dabei werden RTC6-Karte und Verfahrtisch wieder übernommen. In der Folge kann die syncAXIS control-Instanz ohne Einschränkungen (siehe Seite 150) wieder benutzt werden (= <code>slsc_OperationStatus_Green</code>). Der Vorgang ist typischerweise in unter 0,1 s abgeschlossen. • Vor slsc_cfg_acquire_stage (veraltet) kann slsc_cfg_select_stage aufgerufen werden, um einen anderen Ziel-Verfahrtisch als den bisherigen anzugeben. In diesem Fall übernimmt slsc_cfg_acquire_stage (veraltet) also einen anderen Verfahrtisch als den, der mit slsc_cfg_release_stage (veraltet) freigegeben wird. • Siehe auch Kommentar(e) von slsc_cfg_release_stage (veraltet), Seite 150. • Siehe auch Kapitel 2.12.2 "Beispiel – Verfahrtisch vorübergehend freigeben und Ziel-Verfahrtisch wechseln", Seite 74. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	Siehe slsc_cfg_release_stage (veraltet) .
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0 . Veraltet mit syncAXIS-DLL V1.0.7 .
Verweise	slsc_cfg_release_stage (veraltet)

Name der Funktion	<code>slsc_cfg_delete</code>
Zweck	Baut die angegebene syncAXIS control-Instanz ab. Dabei werden auch die Ressourcen (RTC6-Karte, Verfahrtsch, ...) wieder freigegeben.
Funktions-Signatur	<code>uint32_t slsc_cfg_delete(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_delete wird nicht ausgeführt, wenn der angegebene Handle-Wert nicht existiert. Beim Rückgabewert ist dann Bit #02 gesetzt (NotAllowedWithoutInitialization). • slsc_cfg_delete löscht auch den Handle-Wert der angegebenen syncAXIS control-Instanz. Das ist insbesondere zu berücksichtigen, wenn Sie in Ihrem Anwenderprogramm die Handle-Werte verwalten (siehe Kommentar im Code-Beispiel bei slsc_cfg_initialize_from_file). • Während des Abbaus (siehe Seite 26 zum Aufbau) einer syncAXIS control-Instanz im Hardwaremodus finden folgende Vorgänge statt: <ul style="list-style-type: none"> – Scan-Kopf: wird freigegeben. Die Scan-Kopf-Spiegelposition bleibt, wie sie gerade ist (wird nicht verändert). – RTC6: wird freigegeben. Die RTC6-Lasersteuerung ist nicht mehr aktiv. – Verfahrtsch: wird freigegeben. Die Verfahrtschposition bleibt, wie sie gerade ist (wird nicht verändert). – Output-Signale: es werden die zuletzt gesetzten Signale weiterhin ausgegeben. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_cfg_initialize_from_file

Name der Funktion	slsc_cfg_delete_trajectory_config
Zweck	Hilfsfunktion für die Softwareentwicklung: Löscht das Trajektorienkonfigurations-Objekt (zur Vermeidung von Speicherlecks), siehe Code-Beispiel .
Funktions-Signatur	<code>uint32_t slsc_cfg_delete_trajectory_config(slsc_TrajectoryConfig** TrajConfig);</code>
Argument(e)	TrajConfig Siehe Struktur slsc_TrajectoryConfig .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_delete_trajectory_config verändert keine Konfigurations-Parameter-Werte. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_TrajectoryConfig* TrajConfig = 0; // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file // get configuration parameters slsc_cfg_get_trajectory_config(Handle, &TrajConfig); // change configuration parameters TrajConfig->GeometryConfig.BlendMode = slsc_BlendModes::slsc_BlendModes_Deactivated; slsc_cfg_set_trajectory_config(Handle, TrajConfig); // delete configuration object slsc_TrajectoryConfig // (does not change the configuration parameters) slsc_cfg_delete_trajectory_config(&TrajConfig);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_cfg_get_trajectory_config, slsc_cfg_set_trajectory_config

Name der Funktion	<code>slsc_cfg_get_calculation_dynamics_jump_scan_device</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> • Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Sprünge, aber nicht Markierungen
Funktions-Signatur	<code>uint32_t slsc_cfg_get_calculation_dynamics_jump_scan_device(size_t Handle, double* JumpAngularAcc, double* JumpAngularJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	JumpAngularAcc Wie JumpAngularAcc.
	JumpAngularJerk Wie JumpAngularJerk.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> – <code>slsc_cfg_set_calculation_dynamics_jump_scan_device</code> – <code>slsc_list_set_calculation_dynamics_jump_scan_device</code> • Für <code>slsc_cfg_get_calculation_dynamics_jump_scan_device</code> gibt es: <ul style="list-style-type: none"> – Eine entsprechende Job-Funktion (<code>slsc_list_*</code>) <code>slsc_list_set_calculation_dynamics_jump_scan_device</code> – Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0.
Verweise	<code>slsc_cfg_set_calculation_dynamics_jump_scan_device</code>, <code>slsc_list_set_calculation_dynamics_jump_scan_device</code>

Name der Funktion	<code>slsc_cfg_get_calculation_dynamics_mark_scan_device</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> • Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Markierungen, aber nicht Sprünge
Funktions-Signatur	<code>uint32_t slsc_cfg_get_calculation_dynamics_mark_scan_device(size_t Handle, double* MarkAngularAcc, double* MarkAngularJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	MarkAngularAcc Wie MarkAngularAcc.
	MarkAngularJerk Wie MarkAngularJerk.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> – <code>slsc_cfg_set_calculation_dynamics_mark_scan_device</code> – <code>slsc_list_set_calculation_dynamics_mark_scan_device</code> • Für <code>slsc_cfg_get_calculation_dynamics_mark_scan_device</code> gibt es: <ul style="list-style-type: none"> – Eine entsprechende Job-Funktion (<code>slsc_list_*</code>) <code>slsc_list_set_calculation_dynamics_mark_scan_device</code> – Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0.
Verweise	<code>slsc_cfg_set_calculation_dynamics_mark_scan_device</code>, <code>slsc_list_set_calculation_dynamics_mark_scan_device</code>

Name der Funktion	<code>slsc_cfg_get_calculation_dynamics_stage</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Verfahrtsch-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Verfahrtsch-Bewegung verwendet
Funktions-Signatur	<code>uint32_t slsc_cfg_get_calculation_dynamics_stage(size_t Handle, slsc_Stage Stage, double* StageVel, double* StageAcc, double* StageJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Stage Siehe enum slsc_Stage .
	StageVel Wie StageVel .
	StageAcc Wie StageAcc .
	StageJerk Wie StageJerk .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_calculation_dynamics_stage</code> Für <code>slsc_cfg_get_calculation_dynamics_stage</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_calculation_dynamics_stage</code>

Name der Funktion	<code>slsc_cfg_get_dynamic_limits_scan_device</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Scan-Device-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_get_dynamic_limits_scan_device(size_t Handle, double* AngularVel, double* AngularAcc, double* AngularJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	AngularVel Wie AngularVel.
	AngularAcc Wie AngularAcc.
	AngularJerk Wie AngularJerk.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_dynamic_limits_scan_device</code> Für <code>slsc_cfg_get_dynamic_limits_scan_device</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_dynamic_limits_scan_device</code>

Name der Funktion	<code>slsc_cfg_get_dynamic_limits_stage</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Die Dynamikgrenzen des vorgesehenen Verfahrtsch-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_get_dynamic_limits_stage(size_t Handle, slsc_Stage Stage, double* StageVel, double* StageAcc, double* StageJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Stage Siehe enum slsc_Stage .
	StageVel Wie StageVel .
	StageAcc Wie StageAcc .
	StageJerk Wie StageJerk .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_dynamic_limits_stage</code> Für <code>slsc_cfg_get_dynamic_limits_stage</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. double StageVel; double StageAcc; double StageJerk; slsc_cfg_get_dynamic_limits_stage(Handle, slsc_Stage1, &StageVel, &StageAcc, &StageJerk);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_dynamic_limits_stage</code>

Name der Funktion	<code>slsc_cfg_get_dynamic_violation_reaction</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Die Reaktion bei einer Dynamik-Grenzwertüberschreitung.
Funktions-Signatur	<code>uint32_t slsc_cfg_get_dynamic_violation_reaction(size_t Handle, slsc_DynamicViolationReaction* DynamicViolationReaction);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	DynamicViolationReaction Parameterrückgabe: Zeiger. Siehe enum slsc_DynamicViolationReaction .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_dynamic_violation_reaction</code> Für <code>slsc_cfg_get_dynamic_violation_reaction</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_DynamicViolationReaction DynamicViolationReaction; // Handle: see Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_get_dynamic_violation_reaction(Handle, &DynamicViolationReaction);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_dynamic_violation_reaction</code>

Name der Funktion	<code>slsc_cfg_get_field_limits_scan_device</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Arbeitsfeldgrenzen des vorgesehenen Scan-Device-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_get_field_limits_scan_device(size_t Handle, double* FieldLimitsMin, double* FieldLimitsMax);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	FieldLimitsMin Wie FieldLimitsMin .
	FieldLimitsMax Wie FieldLimitsMax .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_field_limits_scan_device</code> Für <code>slsc_cfg_get_field_limits_scan_device</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_field_limits_scan_device</code>

Name der Funktion	<code>slsc_cfg_get_field_limits_stage</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Die Arbeitsfeldgrenzen des vorgesehenen Verfahrtsch-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_get_field_limits_stage(size_t Handle, slsc_Stage Stage, double* FieldLimitsMin, double* FieldLimitsMax);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Stage Siehe enum slsc_Stage .
	FieldLimitsMin Wie FieldLimitsMin .
	FieldLimitsMax Wie FieldLimitsMax .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_field_limits_stage</code> Für <code>slsc_cfg_get_field_limits_stage</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_field_limits_stage</code>

Name der Funktion	slsc_cfg_get_jump_time
Zweck	<p>Berechnet die Dauer eines Sprungs (außerhalb "normaler" Jobs):</p> <ul style="list-style-type: none"> • Basierend auf der momentanen Einstellung der angegebenen syncAXIS control-Instanz • Sowie in Abhängigkeit der angegebenen Werte für Start-Dynamik und End-Dynamik
Funktions-Signatur	<pre>uint32_t slsc_cfg_get_jump_time(const size_t Handle, const double* SStart, const double* VStart, const double* AStart, const double* SEnd, const double* VEnd, const double* AEnd, double MinimalJumpTime, double* JumpTime);</pre>
Argument(e)	<p>Handle Handle auf eine syncAXIS control-Instanz.</p>
	<p>SStart Zeiger auf ein Array der Dimension 2. Startpunkt-Koordinaten. In mm.</p>
	<p>VStart Zeiger auf ein Array der Dimension 2. Geschwindigkeit im Startpunkt. In mm/s.</p>
	<p>AStart Zeiger auf ein Array der Dimension 2. Beschleunigung im Startpunkt. In mm/s².</p>
	<p>SEnd Zeiger auf ein Array der Dimension 2. Zielpunkt-Koordinaten. In mm.</p>
	<p>VEnd Zeiger auf ein Array der Dimension 2. Geschwindigkeit im Zielpunkt. In mm/s.</p>
	<p>AEnd Zeiger auf ein Array der Dimension 2. Beschleunigung im Zielpunkt. In mm/s².</p>
	<p>MinimalJumpTime Minstdauer für den Sprung. Zulässige Werte: ≥ 0. In s.</p>
	<p>JumpTime Parameterrückgabe: Zeiger. Zeitdauer des Sprungs. In s.</p>
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_get_jump_time ist für die isolierte Verwendung im Rahmen von Job-Voranalysen zur Optimierung gedacht (slsc_cfg_get_jump_time soll also nicht im "normalen" Job aufgerufen werden). Siehe auch Kapitel 2.2.4 "Jobs simulieren und nachbessern", Seite 24. • Das Berechnen der Dauer des Sprungs während der Job-Ausführung und von slsc_cfg_get_jump_time sind identisch. Allerdings kommuniziert slsc_cfg_get_jump_time niemals mit der RTC6-Karte. • slsc_cfg_get_jump_time: <ul style="list-style-type: none"> – Berechnet die Sprung-Dauer mit momentanen Einstellung der angegebenen syncAXIS control-Instanz (wie die Werte für Dynamik, LaserMinOffTime, LaserPreTriggerTime, MotionDecompositionConfig) – Berücksichtigt Änderungen durch Konfigurations-Funktionen (slsc_cfg_*) (z.B. slsc_cfg_set_jump_speed) <i>nach</i> der Initialisierung – Berücksichtigt nicht die Änderungen durch Job-Funktionen (slsc_list_*) (z.B. slsc_list_set_jump_speed) während der Job-Ausführung

Name der Funktion	<code>slsc_cfg_get_jump_time</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • <code>MinimalJumpTime</code> entspricht <code>MinimalJumpTime</code> von <code>slsc_list_jump_abs_min_time</code>. Soll die Dauer eines gewöhnlichen, mit <code>slsc_list_jump_abs</code> kommandierten Sprungs ausgerechnet werden, müssen Sie den <code>MinimalJumpTime</code>-Wert auf 0.0 setzen. • Für <code>slsc_cfg_get_jump_time</code> gibt es: <ul style="list-style-type: none"> – Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) – Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) • <code>slsc_cfg_get_jump_time</code> ist in jedem Betriebsmodus erlaubt. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.7.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_get_mode</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> • Den Betriebsmodus (<code>ScannerOnly</code>, <code>StageOnly</code>, <code>ScannerAndStage</code>)
Funktions-Signatur	<code>uint32_t slsc_cfg_get_mode(size_t Handle, slsc_OperationMode* Mode);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mode Parameterrückgabe: Zeiger. Siehe enum slsc_OperationMode .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Der Betriebsmodus der syncAXIS control-Instanz wird gesetzt mit slsc_cfg_set_mode. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_OperationMode Mode; // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_get_mode(Handle, &Mode);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_cfg_set_mode

Name der Funktion	<code>slsc_cfg_get_operation_status</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Den Betriebsstatus ("Ampelfarbe")
Funktions-Signatur	<code>uint32_t slsc_cfg_get_operation_status(size_t Handle, slsc_OperationStatus* State);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	State Parameterrückgabe: Zeiger. Siehe enum slsc_OperationStatus .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_get_operation_status gibt keine Auskunft über den Status von Jobs. Beispielsweise bedeutet Betriebsstatus "grün" (syncAXIS control-Instanz läuft, keine Fehler aufgetreten) nicht, dass gerade ein Job abgearbeitet wird. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_OperationStatus State; // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_get_operation_status(Handle, &State);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_cfg_initialize_from_file

Name der Funktion	<code>slsc_cfg_get_scan_device_dynamic_monitoring_level</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Das Kriterium, auf das die Scan-Devices hin überwacht werden sollen (z.B. <code>slsc_DynamicsMonitoringLevel_Velocity</code>)
Funktions-Signatur	<code>uint32_t slsc_cfg_get_scan_device_dynamic_monitoring_level(size_t Handle, slsc_DynamicsMonitoringLevel* DynamicMonitoringLevel);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	DynamicMonitoringLevel Parameterrückgabe: Zeiger. Siehe enum slsc_DynamicsMonitoringLevel .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_scan_device_dynamic_monitoring_level</code> Für <code>slsc_cfg_get_scan_device_dynamic_monitoring_level</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_DynamicsMonitoringLevel DynamicMonitoringLevel; // Handle: see Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_get_scan_device_dynamic_monitoring_level(Handle, &slsc_DynamicsMonitoringLevel);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_scan_device_dynamic_monitoring_level</code>

Name der Funktion	<code>slsc_cfg_get_simulation_setting</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Die Simulationseinstellung
Funktions-Signatur	<code>uint32_t slsc_cfg_get_simulation_setting(size_t Handle, slsc_SimulationSetting* SimulationSetting);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	SimulationSetting Parameterrückgabe: Zeiger. Siehe enum slsc_SimulationSetting .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Die Simulationseinstellung der syncAXIS control-Instanz wird geändert mit slsc_cfg_set_simulation_setting. Siehe auch Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31. Für slsc_cfg_get_simulation_setting gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_SimulationSetting SimulationSetting; // Handle: see Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_get_simulation_setting(Handle, &SimulationSetting);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_set_simulation_setting

Name der Funktion	<code>slsc_cfg_get_stage_dynamic_monitoring_level</code>
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Das Kriterium, auf das die Verfahrtschritte hin überwacht werden sollen (z.B. <code>slsc_DynamicsMonitoringLevel_Velocity</code>)
Funktions-Signatur	<code>uint32_t slsc_cfg_get_stage_dynamic_monitoring_level(size_t Handle, slsc_DynamicsMonitoringLevel* DynamicMonitoringLevel);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	DynamicMonitoringLevel Parameterrückgabe: Zeiger. Siehe enum slsc_DynamicsMonitoringLevel .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_set_stage_dynamic_monitoring_level</code> Für <code>slsc_cfg_get_stage_dynamic_monitoring_level</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_DynamicsMonitoringLevel DynamicMonitoringLevel; // Handle: see Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_get_stage_dynamic_monitoring_level(Handle, &DynamicMonitoringLevel);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_set_stage_dynamic_monitoring_level</code>

Name der Funktion	slsc_cfg_get_sync_axis_version
Zweck	Gibt Versionsinformationen über die aktuell laufende syncAXIS-DLL zurück.
Funktions-Signatur	<code>VersionInfo slsc_cfg_get_sync_axis_version(void);</code>
Argument(e)	Diese Funktion hat keine Argumente.
Rückgabewert	Siehe Struktur VersionInfo .
Kommentar(e)	<ul style="list-style-type: none"> Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	–

Name der Funktion	slsc_cfg_get_trajectory_config
Zweck	Gibt die momentane Einstellung der angegebenen syncAXIS control-Instanz zurück für: <ul style="list-style-type: none"> Die Konfiguration der Trajektorienplanung
Funktions-Signatur	<code>uint32_t slsc_cfg_get_trajectory_config(size_t Handle, slsc_TrajectoryConfig** TrajConfig);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	TrajConfig Parameterrückgabe: Zeiger auf einen Zeiger. Siehe Struktur slsc_TrajectoryConfig .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_cfg_delete_trajectory_config, slsc_cfg_set_trajectory_config

Name der Funktion	slsc_cfg_initialize_copy
Zweck	Initialisierungsfunktion: Baut eine neue (Ziel-)syncAXIS control-Instanz im Simulationsmodus mit der momentanen Konfiguration der angegebenen (Quell-)syncAXIS control-Instanz (entweder im Hardwaremodus oder Simulationsmodus) auf und ordnet ihr einen eindeutigen Handle-Wert zu.
Funktions-Signatur	<code>uint32_t slsc_cfg_initialize_copy(size_t* Handle, size_t OriginalHandle);</code>
Argument(e)	Handle Parameterrückgabe: Zeiger. Handle auf die neu erzeugte syncAXIS control-Instanz (diese ist für den Simulationsmodus konfiguriert).
	OriginalHandle Handle auf eine syncAXIS control-Instanz.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_initialize_copy und slsc_cfg_initialize_from_file sind gleich, außer: <ul style="list-style-type: none"> – slsc_cfg_initialize_copy verwendet die Konfiguration einer bereits vorhandenen (Quell-)syncAXIS control-Instanz, statt diese aus einer <code>syncAXISConfig.xml</code> zu lesen. – slsc_cfg_initialize_copy baut die neue (Ziel-)syncAXIS control-Instanz immer nur im Simulationsmodus auf. • slsc_cfg_initialize_copy ist in erster Linie im Kontext der Aufzeichnung von Modul-Dateien vorgesehen ("Vorbereitung eines Jobs"), siehe Code-Beispiel unten: slsc_cfg_initialize_copy geht slsc_list_begin_module voraus, um schnell eine für den Simulationsmodus konfigurierte syncAXIS control-Instanz zu erzeugen (weil slsc_list_begin_module nur im Simulationsmodus erlaubt ist). Im Anschluss wird das Modul in der "originalen" (Quell-)syncAXIS control-Instanz abgespielt. • Siehe auch Abschnitt "Funktionen für "Module"", Seite 95. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	Siehe Abbildung 27, Seite 67 und Kapitel 11 "Anhang D: Anwendungshinweis – Pufferunterlauf vermeiden mittels Modulen", Seite 347.
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0.
Verweise	slsc_list_begin_module

Name der Funktion	slsc_cfg_initialize_from_file	
Zweck	Initialisierungsfunktion: Baut eine neue syncAXIS control-Instanz (unter Verwendung der angegebenen XML-Konfigurationsdatei) auf und teilt ihr einen eindeutigen Handle-Wert zu.	
Funktions-Signatur	<code>uint32_t slsc_cfg_initialize_from_file(size_t* Handle, const char* XmlConfigFileName);</code>	
Argument(e)	Handle	Parameterrückgabe: Zeiger. Handle auf eine syncAXIS control-Instanz .
	XmlConfigFileName	Name der XML-Konfigurationsdatei. Zeiger auf einen \0-terminierten ANSI-String, 1 Byte pro Zeichen.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.	
Kommentar(e)	<ul style="list-style-type: none"> ⚠ Warnung! Verletzungsgefahr durch Laserstrahlung! slsc_cfg_initialize_from_file kann zu Zuständen der RTC6-Karte(n) führen, bei denen unvorhergesehen der Laser emittieren könnte! Stellen Sie vor dem Aufruf von slsc_cfg_initialize_from_file sicher, dass der Laser ausgeschaltet ist! slsc_cfg_initialize_from_file wird nicht ausgeführt, wenn die in der XML-Konfigurationsdatei angegebene RTC6-Karte bereits übernommen ist. Beispiel: slsc_cfg_initialize_from_file wird ein zweites Mal mit der identischen XML-Konfigurationsdatei aufgerufen wie zuvor (und dort ist <code>BySerialNumber</code> als <code>BoardIdentificationMethod</code> eingetragen). Beim Rückgabewert ist dann Bit #06 gesetzt (UnplausibleOrUnknownParameter). slsc_cfg_initialize_from_file schlägt fehl, wenn der FilterBandwidth-Wert kleiner als 0,23 ist. Beim Rückgabewert ist dann Bit #14 gesetzt (XmlLoadError). slsc_cfg_initialize_from_file schaltet u.a. auch die Lasersteuerung auf der RTC6-Karte aktiv (scharf) (dazu wird intern der RTC-Kontrollbefehl set_laser_control benutzt). Weiterhin führt slsc_cfg_initialize_from_file u.a. slsc_ctrl_enable_laser automatisch aus, d.h. die Lasersteuersignale LASERON, LASER1 und LASER2 werden tatsächlich schon an der RTC6-Karte ausgegeben. ⚠ Vorsicht! Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! slsc_cfg_initialize_from_file bringt, wie auch die letzte Job-Funktion (slsc_list_*), <ul style="list-style-type: none"> – die Scan-Kopf-Spiegel in die Nullposition – lässt aber die Verfahrtschposition unverändert Zu den weiteren Vorgängen beim Initialisieren der syncAXIS control-Instanz (Hardwaremodus) siehe Seite 26. Das syncAXIS control-Software-Paket enthält eine xsd-Datei (mit Inline-Kommentaren zur Benutzerunterstützung) und Beispiel-XML-Konfigurationsdateien. Die xsd-Datei legt das Schema für die XML-Konfigurationsdatei fest, die mit slsc_cfg_initialize_from_file zur Initialisierung einer syncAXIS control-Instanz übergeben wird. 	

Name der Funktion	<code>slsc_cfg_initialize_from_file</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • <code>slsc_cfg_reinitialize_from_file</code> und <code>slsc_cfg_initialize_from_file</code> sind gleich, außer: <ul style="list-style-type: none"> – <code>slsc_cfg_initialize_from_file</code> teilt einer <code>syncAXIS control-Instanz</code> einen eindeutigen <code>Handle</code>-Wert zu – Bei <code>slsc_cfg_reinitialize_from_file</code> wird ein <code>Handle</code>-Wert angegeben (der unverändert bleibt). • Das <code>syncAXIS control</code>-Software-Paket bietet "out-of-the-box" keine Möglichkeit zu überprüfen, ob und welche <code>Handle</code>-Werte schon vorhanden sind. Daher sollten Benutzer Code einfügen, mit dem die <code>Handles</code> verwaltet werden. • Zur Zulässigkeit von <code>syncAXIS control</code>-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. size_t Handle = 0; const char* XmlConfigFileName = "syncAXISConfig.xml"; slsc_cfg_initialize_from_file(&Handle, XmlConfigFileName); // Diesen Handle-Wert muss man sich (im Anwenderprogramm) merken, // um auf genau diese Instanz wieder zugreifen zu können. // d.h. Handle-Werte können nicht "out-of-the-box" abgefragt werden.</pre>
Versionsinfo	Verfügbar ab <code>syncAXIS-DLL V0.9.0</code> .
Verweise	<code>slsc_cfg_delete</code> , <code>slsc_ctrl_enable_laser</code> , <code>slsc_cfg_get_operation_status</code> , <code>slsc_cfg_reinitialize_from_file</code>

Name der Funktion	<code>slsc_cfg_register_callback_job_end_planned</code>
Zweck	Stellt ein, dass die angegebene “Callback-Funktion” bei einem “Callback-Event” des Typs “job_end_planned” aufgerufen wird.
Funktions-Signatur	<code>uint32_t slsc_cfg_register_callback_job_end_planned(size_t Handle, slsc_ExecTimeCallback Callback, void* Context);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Callback Funktions-Zeiger auf die “Callback-Funktion” (= benutzerdefinierte Funktion gemäß Signatur <code>slsc_ExecTimeCallback</code>).
	Context Zeiger auf ein benutzerdefiniertes Objekt. In der Funktion <code>Callback</code> kann auf dieses Objekt zugegriffen werden.
Rückgabewert	Siehe Kapitel 4 “Standard-Rückgabewerte der syncAXIS-DLL Funktionen” , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Sobald ein “Callback-Event” des Typs “job_end_planned” (siehe auch Abbildung 12, Seite 43) innerhalb der syncAXIS control-Instanz auftritt, wird automatisch die angegebene “Callback-Funktion” aufgerufen. • Siehe auch Abschnitt “Funktionen zum Erfassen von “Callback Events””, Seite 81. • <code>slsc_cfg_register_callback_job_end_planned</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es <i>keine</i> Default-Werte für die Argumente von <code>slsc_cfg_register_callback_job_end_planned</code>, mit der die syncAXIS control-Instanz initialisiert wird. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus “Manuelle Positionierung” siehe Kapitel 2.12 “Über den Modus “Manuelle Positionierung””, Seite 70.
Code-Beispiel	Siehe <code>slsc_cfg_register_callback_job_finished_executing</code> .
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_register_callback_job_finished_executing</code>
Zweck	Stellt ein, dass die angegebene “Callback-Funktion” bei einem “Callback-Event” des Typs “job_finished_executing” aufgerufen wird.
Funktions-Signatur	<code>uint32_t slsc_cfg_register_callback_job_finished_executing(size_t Handle, slsc_ExecTimeCallback Callback, void* Context);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Callback Funktions-Zeiger auf die “Callback-Funktion” (= benutzerdefinierte Funktion gemäß Signatur <code>slsc_ExecTimeCallback</code>).
	Context Zeiger auf ein benutzerdefiniertes Objekt. In der Funktion <code>Callback</code> kann auf dieses Objekt zugegriffen werden.
Rückgabewert	Siehe Kapitel 4 “Standard-Rückgabewerte der syncAXIS-DLL Funktionen” , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Sobald ein “Callback-Event” des Typs “job_finished_executing” (siehe auch Abbildung 12, Seite 43) innerhalb der syncAXIS control-Instanz auftritt, wird automatisch die angegebene “Callback-Funktion” aufgerufen. • Siehe auch Abschnitt “Funktionen zum Erfassen von “Callback Events””, Seite 81. • <code>slsc_cfg_register_callback_job_finished_executing</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es <i>keine</i> Default-Werte für die Argumente von <code>slsc_cfg_register_callback_job_finished_executing</code>, mit der die syncAXIS control-Instanz initialisiert wird. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus “Manuelle Positionierung” siehe Kapitel 2.12 “Über den Modus “Manuelle Positionierung””, Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. int Counter; slsc_ExecTimeCallback Callback = [](size_t JobID, uint64_t Progress, double ExecTime, void* Context) { int* Counter = static_cast<int*>(Context); Counter++; }; // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_register_callback_job_finished_executing(Handle, Callback, &Counter);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_register_callback_job_is_executing</code>
Zweck	Stellt ein, dass die angegebene “Callback-Funktion” bei einem “Callback-Event” des Typs “job_is_executing” aufgerufen wird.
Funktions-Signatur	<code>uint32_t slsc_cfg_register_callback_job_is_executing(size_t Handle, slsc_ExecTimeCallback Callback, void* Context);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Callback Funktions-Zeiger auf die “Callback-Funktion” (= benutzerdefinierte Funktion gemäß Signatur <code>slsc_ExecTimeCallback</code>).
	Context Zeiger auf ein benutzerdefiniertes Objekt. In der Funktion <code>Callback</code> kann auf dieses Objekt zugegriffen werden.
Rückgabewert	Siehe Kapitel 4 “Standard-Rückgabewerte der syncAXIS-DLL Funktionen” , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Sobald ein “Callback-Event” des Typs “job_is_executing” (siehe auch Abbildung 12, Seite 43) innerhalb der syncAXIS control-Instanz auftritt, wird automatisch die angegebene “Callback-Funktion” aufgerufen. • Siehe auch Abschnitt “Funktionen zum Erfassen von “Callback Events””, Seite 81. • slsc_cfg_register_callback_job_is_executing ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es <i>keine</i> Default-Werte für die Argumente von slsc_cfg_register_callback_job_is_executing, mit der die syncAXIS control-Instanz initialisiert wird. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus “Manuelle Positionierung” siehe Kapitel 2.12 “Über den Modus “Manuelle Positionierung””, Seite 70.
Code-Beispiel	Siehe slsc_cfg_register_callback_job_finished_executing .
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_register_callback_job_loaded_enough</code>
Zweck	Stellt ein, dass die angegebene “Callback-Funktion” bei einem “Callback-Event” des Typs “job_loaded_enough” aufgerufen wird.
Funktions-Signatur	<code>uint32_t slsc_cfg_register_callback_job_loaded_enough(size_t Handle, slsc_JobCallback Callback, void* Context);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Callback Funktions-Zeiger auf die “Callback-Funktion” (= benutzerdefinierte Funktion gemäß Signatur <code>slsc_JobCallback</code>).
	Context Zeiger auf ein benutzerdefiniertes Objekt. In der Funktion <code>Callback</code> kann auf dieses Objekt zugegriffen werden.
Rückgabewert	Siehe Kapitel 4 “Standard-Rückgabewerte der syncAXIS-DLL Funktionen” , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Sobald ein “Callback-Event” des Typs “job_loaded_enough” (siehe auch Abbildung 12, Seite 43) innerhalb der syncAXIS control-Instanz auftritt, wird automatisch die angegebene “Callback-Funktion” aufgerufen. • Siehe auch Abschnitt “Funktionen zum Erfassen von “Callback Events””, Seite 81. • slsc_cfg_register_callback_job_loaded_enough ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es <i>keine</i> Default-Werte für die Argumente von slsc_cfg_register_callback_job_loaded_enough, mit der die syncAXIS control-Instanz initialisiert wird. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus “Manuelle Positionierung” siehe Kapitel 2.12 “Über den Modus “Manuelle Positionierung””, Seite 70.
Code-Beispiel	Siehe slsc_cfg_register_callback_job_finished_executing .
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_register_callback_job_progress_planned</code>
Zweck	Stellt ein, dass die angegebene “Callback-Funktion” bei einem “Callback-Event” des Typs “job_progress_planned” aufgerufen wird.
Funktions-Signatur	<code>uint32_t slsc_cfg_register_callback_job_progress_planned(size_t Handle, slsc_ExecTimeCallback Callback, void* Context);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Callback Funktions-Zeiger auf die “Callback-Funktion” (= benutzerdefinierte Funktion gemäß Signatur <code>slsc_ExecTimeCallback</code>).
	Context Zeiger auf ein benutzerdefiniertes Objekt. In der Funktion <code>Callback</code> kann auf dieses Objekt zugegriffen werden.
Rückgabewert	Siehe Kapitel 4 “Standard-Rückgabewerte der syncAXIS-DLL Funktionen” , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Sobald ein “Callback-Event” des Typs “job_progress_planned” (siehe auch Abbildung 12, Seite 43) innerhalb der syncAXIS control-Instanz auftritt, wird automatisch die angegebene “Callback-Funktion” aufgerufen. • Siehe auch Abschnitt “Funktionen zum Erfassen von “Callback Events””, Seite 81. • <code>slsc_cfg_register_callback_job_progress_planned</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es <i>keine</i> Default-Werte für die Argumente von <code>slsc_cfg_register_callback_job_progress_planned</code>, mit der die syncAXIS control-Instanz initialisiert wird. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus “Manuelle Positionierung” siehe Kapitel 2.12 “Über den Modus “Manuelle Positionierung””, Seite 70.
Code-Beispiel	Siehe <code>slsc_cfg_register_callback_job_finished_executing</code> .
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_register_callback_job_start_planned</code>
Zweck	Stellt ein, dass die angegebene “Callback-Funktion” bei einem “Callback-Event” des Typs “job_start_planned” aufgerufen wird.
Funktions-Signatur	<code>uint32_t slsc_cfg_register_callback_job_start_planned(size_t Handle, slsc_JobCallback Callback, void* Context);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Callback Funktions-Zeiger auf die “Callback-Funktion” (= benutzerdefinierte Funktion gemäß Signatur <code>slsc_JobCallback</code>).
	Context Zeiger auf ein benutzerdefiniertes Objekt. In der Funktion <code>Callback</code> kann auf dieses Objekt zugegriffen werden.
Rückgabewert	Siehe Kapitel 4 “Standard-Rückgabewerte der syncAXIS-DLL Funktionen” , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Sobald ein “Callback-Event” des Typs “job_start_planned” (siehe auch Abbildung 12, Seite 43) innerhalb der syncAXIS control-Instanz auftritt, wird automatisch die angegebene “Callback-Funktion” aufgerufen. • Siehe auch Abschnitt “Funktionen zum Erfassen von “Callback Events””, Seite 81. • <code>slsc_cfg_register_callback_job_start_planned</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es <i>keine</i> Default-Werte für die Argumente von <code>slsc_cfg_register_callback_job_start_planned</code>, mit der die syncAXIS control-Instanz initialisiert wird. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus “Manuelle Positionierung” siehe Kapitel 2.12 “Über den Modus “Manuelle Positionierung””, Seite 70.
Code-Beispiel	Siehe <code>slsc_cfg_register_callback_job_finished_executing</code> .
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_reinitialize</code>
Zweck	Initialisierungsfunktion: Baut die (über das Handle) angegebene syncAXIS control-Instanz ab und dann wieder (unter Verwendung der zuvor ausgelesenen, momentanen Konfigurationseinstellungen und -werte) neu auf. Der Handle -Wert bleibt dabei unverändert.
Funktions-Signatur	<code>uint32_t slsc_cfg_reinitialize(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_reinitialize ist in allen Betriebsmodi (siehe slsc_OperationMode) erlaubt. • slsc_cfg_reinitialize wird nicht ausgeführt, wenn die angegebene syncAXIS control-Instanz gerade einen Job ausführt. Dann ist beim Rückgabewert Bit #03 gesetzt (NotAllowedInExecuting). • slsc_cfg_reinitialize wird nicht ausgeführt, wenn der angegebene Handle-Wert nicht existiert. Beim Rückgabewert ist dann Bit #02 gesetzt (NotAllowedWithoutInitialization). • slsc_cfg_reinitialize schaltet u.a. auch die Lasersteuerung auf der RTC6-Karte aktiv (scharf) (dazu wird intern der RTC-Kontrollbefehl set_laser_control benutzt). Weiterhin führt slsc_cfg_reinitialize u.a. slsc_ctrl_enable_laser automatisch aus, d.h. die Lasersteuersignale LASERON, LASER1 und LASER2 werden tatsächlich schon an der RTC6-Karte ausgegeben. • ⚠ Vorsicht! Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! • slsc_cfg_reinitialize bringt, wie auch die letzte Job-Funktion (slsc_list_*), <ul style="list-style-type: none"> – die Scan-Kopf-Spiegel in die Nullposition – lässt aber die Verfahrtschichtposition unverändert • Zu den weiteren Vorgängen beim Initialisieren der syncAXIS control-Instanz (Hardwaremodus) siehe Seite 26. • slsc_cfg_reinitialize, slsc_cfg_reinitialize_from_file und slsc_cfg_initialize_from_file sind gleich, außer: <ul style="list-style-type: none"> – slsc_cfg_initialize_from_file teilt einer syncAXIS control-Instanz einen eindeutigen Handle-Wert zu – Bei slsc_cfg_reinitialize und slsc_cfg_reinitialize_from_file wird ein Handle-Wert angegeben (der unverändert bleibt). • Anwendungsfall für slsc_cfg_reinitialize: Die syncAXIS control-Instanz befindet sich in einem Fehlerzustand. Um weiterhin Jobs ausführen zu können, muss der Benutzer neu initialisieren. Mit slsc_cfg_reinitialize kann der Benutzer dabei die momentane Konfiguration einfach weiter verwenden, ohne die bisherigen Konfigurationsschritte (syncAXISConfig.xml einlesen, Serie von syncAXIS-DLL-Funktions-Aufrufen) wiederholen zu müssen.

Name der Funktion	slsc_cfg_reinitialize
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Siehe auch Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen", Seite 26 mit Abbildung 3, Seite 27. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.4.0.
Verweise	slsc_cfg_delete, slsc_ctrl_enable_laser, slsc_cfg_initialize_from_file

Name der Funktion	<code>slsc_cfg_reinitialize_from_file</code>
Zweck	Initialisierungsfunktion: Baut die (über das Handle) angegebene syncAXIS control-Instanz ab und dann wieder (unter Verwendung der angegebenen XML-Konfigurationsdatei) neu auf. Der Handle -Wert bleibt dabei unverändert.
Funktions-Signatur	<code>uint32_t slsc_cfg_reinitialize_from_file(size_t Handle, const char* XmlConfigFileName);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	XmlConfigFileName Name der syncAXISConfig.xml . Zeiger auf einen \0-terminierten ANSI-String, 1 Byte pro Zeichen.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_reinitialize ist in allen Betriebsmodi (siehe slsc_OperationMode) erlaubt. • slsc_cfg_reinitialize_from_file wird nicht ausgeführt, wenn die angegebene syncAXIS control-Instanz gerade einen Job ausführt. Dann ist beim Rückgabewert Bit #03 gesetzt (NotAllowedInExecuting). • slsc_cfg_reinitialize_from_file wird nicht ausgeführt, wenn der angegebene Handle-Wert nicht existiert. Beim Rückgabewert ist dann Bit #02 gesetzt (NotAllowedWithoutInitialization). • slsc_cfg_reinitialize_from_file schaltet u.a. auch die Lasersteuerung auf der RTC6-Karte aktiv (scharf) (dazu wird intern der RTC-Kontrollbefehl set_laser_control benutzt). Weiterhin führt slsc_cfg_reinitialize_from_file u.a. slsc_ctrl_enable_laser automatisch aus, d. h. die Lasersteuersignale LASERON, LASER1 und LASER2 werden tatsächlich schon an der RTC6-Karte ausgegeben. • ⚠ Vorsicht! Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! • slsc_cfg_reinitialize_from_file bringt, wie auch die letzte Job-Funktion (slsc_list_*), <ul style="list-style-type: none"> – die Scan-Kopf-Spiegel in die Nullposition – lässt aber die Verfahrtschposition unverändert • Zu den weiteren Vorgängen beim Initialisieren der syncAXIS control-Instanz (Hardwaremodus) siehe Seite 26. • Das syncAXIS control-Software-Paket enthält eine xsd-Datei (mit Inline-Kommentaren zur Benutzerunterstützung) und eine Beispiel-XML-Konfigurationsdatei. Die xsd-Datei legt das Schema für die XML-Konfigurationsdatei fest, die mit slsc_cfg_reinitialize_from_file zur Initialisierung einer syncAXIS control-Instanz übergeben wird. • slsc_cfg_reinitialize, slsc_cfg_reinitialize_from_file und slsc_cfg_initialize_from_file sind gleich, außer: <ul style="list-style-type: none"> – slsc_cfg_initialize_from_file teilt einer syncAXIS control-Instanz einen eindeutigen Handle-Wert zu – Bei slsc_cfg_reinitialize und slsc_cfg_reinitialize_from_file wird ein Handle-Wert angegeben (der unverändert bleibt).

Name der Funktion	slsc_cfg_reinitialize_from_file
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Siehe auch Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen", Seite 26 mit Abbildung 3, Seite 27. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_cfg_delete, slsc_ctrl_enable_laser, slsc_cfg_initialize_from_file, slsc_cfg_reinitialize

Name der Funktion	slsc_cfg_release_stage (veraltet)
Zweck	<p>Veraltet.</p> <p>Verwenden Sie stattdessen slsc_ctrl_follow/slsc_ctrl_unfollow.</p> <p>Die angegebene syncAXIS control-Instanz gibt vorübergehend den Verfahrtsch frei. Dieser kann nun extern (z.B. durch ein nicht-syncAXIS control-basiertes Anwenderprogramm) gesteuert werden.</p>
Funktions-Signatur	<code>uint32_t slsc_cfg_release_stage (veraltet) (size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.

Name der Funktion	slsc_cfg_release_stage (veraltet)
Kommentar(e)	<ul style="list-style-type: none"> Nach dem erfolgreichen Aufbau einer (im Hardwaremodus konfigurierten) syncAXIS control-Instanz (durch slsc_cfg_initialize_from_file) ist (außer der RTC6-Karte auch) der Verfahrtschirm übernommen. Mit slsc_cfg_release_stage (veraltet) kann der Verfahrtschirm <i>temporär</i> freigegeben werden. Das Freigeben erfolgt typischerweise in unter 0,02 s. Die betreffende syncAXIS control-Instanz muss also nicht abgebaut und anschließend wieder aufgebaut (das Aufbauen dauert typischerweise um 1 s) werden. Während der Verfahrtschirm temporär freigegeben ist, kann ihn währenddessen ein externes Anwenderprogramm steuern. Der Handle-Wert der syncAXIS control-Instanz wird durch slsc_cfg_release_stage (veraltet) nicht verändert. slsc_cfg_release_stage (veraltet) wird nicht akzeptiert, wenn gerade ein Job ausgeführt wird. Dann ist beim Rückgabewert Bit #03 gesetzt (NotAllowedInExecuting). Nach slsc_cfg_release_stage (veraltet) werden alle Jobs aus der Job-Queue gelöscht. Nach slsc_cfg_release_stage (veraltet) bleibt die syncAXIS control-Instanz bestehen, ist aber syncAXIS control-Instanz vorübergehend (bis zum Aufruf von slsc_cfg_acquire_stage (veraltet)) nicht mehr durch alle Funktionen ansprechbar: <ul style="list-style-type: none"> Der Betriebsstatus wechselt zu "rot" (siehe slsc_cfg_get_operation_status). Job-Funktionen (slsc_list_*) werden nicht akzeptiert (also auch slsc_list_begin, deshalb kann auch kein Job gestartet werden). Beim Rückgabewert ist dann Bit #02 gesetzt (NotAllowedWithoutInitialization). Kontroll-Funktionen (slsc_ctrl_*) werden nicht akzeptiert. Beim Rückgabewert ist dann Bit #02 gesetzt (NotAllowedWithoutInitialization). Jedoch werden viele Konfigurations-Funktionen (slsc_cfg_*) akzeptiert (z.B. slsc_cfg_delete, slsc_cfg_acquire_stage (veraltet), slsc_cfg_select_stage). Nicht akzeptierte Konfigurations-Funktionen sind: slsc_cfg_get_mode, slsc_cfg_set_list_handling_mode, slsc_cfg_set_mode, sowie alle Funktionen zum Erfassen von Event Callbacks (cfg_register_callback-Funktionen, siehe Abschnitt "Funktionen zum Erfassen von "Callback Events", Seite 81). slsc_cfg_release_stage (veraltet) ändert <i>nicht</i> die aktuelle Lasersteuerungskonfiguration (d.h. z.B. es wird kein slsc_ctrl_disable_laser abgesetzt). Um die syncAXIS control-Instanz wieder in den Ausgangszustand zu versetzen, muss slsc_cfg_acquire_stage (veraltet) aufgerufen werden. Dabei werden RTC6-Karte und Verfahrtschirm wieder übernommen. Dieser Vorgang ist typischerweise in unter 0,1 s abgeschlossen.

Name der Funktion	slsc_cfg_release_stage (veraltet)
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • slsc_cfg_release_stage (veraltet) gibt technisch auch die RTC6-Karte temporär frei. Bei der Wieder-Akquirierung mit slsc_cfg_acquire_stage (veraltet) muss deren interner Zustand unverändert sein. Deshalb darf die RTC6-Karte (anders als der Verfahrtisch) <i>nicht</i> durch ein anderes Anwenderprogramm genutzt werden (weil dieses den internen Zustand RTC6-Karte verändern kann). • Siehe auch Kapitel 2.12.2 "Beispiel – Verfahrtisch vorübergehend freigeben und Ziel-Verfahrtisch wechseln", Seite 74. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. size_t Handle = 0; const char* XmlConfigFileName = "syncAXISConfig.xml"; slsc_cfg_initialize_from_file(&Handle, XmlConfigFileName); // typical duration: about 1 s // [optional] define and execute a Job slsc_cfg_release_stage (veraltet)(Handle); // typical duration: < 0.02 s // [optional] external control of stage slsc_cfg_acquire_stage (veraltet)(Handle); // typical duration: < 0.1 s // [optional] define and execute a Job // afterwards destroy instance slsc_cfg_delete(Handle);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0. Veraltet mit syncAXIS-DLL V1.0.7.
Verweise	slsc_cfg_acquire_stage (veraltet)

Name der Funktion	<code>slsc_cfg_select_heuristic</code>
Zweck	Zum Angeben der Kennlinie für die Geschwindigkeitsverminderung (<code>DynamicReductionFunction</code>).
Funktions-Signatur	<code>uint32_t slsc_cfg_select_heuristic(size_t Handle, uint32_t HeuristicIndex);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	HeuristicIndex Index der gewünschten Kennlinie für die Geschwindigkeitsverminderung (<code>DynamicReductionFunction</code>). Zulässiger Wertebereich: 0...(<code>DynamicReductionFunction</code> – 1).
Rückgabewert	Siehe Kapitel 4 “Standard-Rückgabewerte der syncAXIS-DLL Funktionen”, Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> • <code>slsc_cfg_select_heuristic</code> setzt voraus, dass der Betriebsmodus “ScannerAndStage” aktiv ist. Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). • Für HeuristicIndex-Wert > (<code>DynamicReductionFunction</code> – 1) ist beim Rückgabewert Bit #06 gesetzt (<code>UnplausibleOrUnknownParameter</code>). • <code>slsc_cfg_select_heuristic</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von <code>slsc_cfg_select_heuristic</code>, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> – <code><cfg:Configuration> → <cfg:MotionDecompositionConfig> → <cfg:HeuristicConfig> → <cfg:DynamicReductionFunctions> → <cfg:DynamicReductionFunction ...></code> (= der erste <code>DynamicReductionFunction</code>-Tag; entspricht <code>slsc_cfg_select_heuristic(HeuristicIndex = 0)</code>). • Für <code>slsc_cfg_select_heuristic</code> gibt es keine entsprechende Job-Funktion (<code>slsc_list_*</code>). • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus “Manuelle Positionierung” siehe Kapitel 2.12 “Über den Modus “Manuelle Positionierung””, Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.4.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_select_stage</code>
Zweck	Zum Angeben des Ziel-Verfahrtischs ("Verfahrtisch-Wechsel"). <code>slsc_cfg_select_stage</code> ersetzt ab syncAXIS-DLL ≥ V1.2.0 <code>slsc_cfg_select_stage_axis</code> (veraltet).
Funktions-Signatur	<code>uint32_t slsc_cfg_select_stage(size_t Handle, slsc_Stage Stage, uint32_t CorrectionFileIndex);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Stage Siehe enum <code>slsc_Stage</code>.
	CorrectionFileIndex Index der Korrekturdatei, die verwendet werden soll (Korrekturdateien sind in der <code>syncAXISConfig.xml</code> angegeben; siehe auch Abschnitt "Korrekturdatei-bezogene Funktionen", Seite 99). Erlaubte Werte: 0...3.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Mit <code>slsc_cfg_select_stage</code> wird derjenige Verfahrtisch angegeben, der durch <code>slsc_ctrl_move_stage_abs</code> bewegt werden soll. • Um <code>slsc_cfg_select_stage</code> verwenden zu können, muss ein Dongle verwendet werden, der die Verwendung mehrerer Verfahrtische erlaubt (Standard-Dongle nicht ausreichend)! Sonst ist beim Rückgabewert Bit #31 gesetzt (InvalidOrMissingDongle). • <code>slsc_cfg_select_stage</code> ist nur im Modus "Manuelle Positionierung" möglich. • Jobs können nur nach einer Wiederübernahme des Verfahrtischs mit <code>slsc_ctrl_follow</code> ausgeführt werden. Die nach einem Verfahrtisch-Wechsel erfolgenden Job-Ausführungen werden mit dem zur Laufzeit übernommenen Verfahrtisch ausgeführt, nicht mit dem zur Planungszeit übernommenen Verfahrtisch. Siehe auch Kapitel 2.12.2 "Beispiel – Verfahrtisch vorübergehend freigeben und Ziel-Verfahrtisch wechseln", Seite 74. • <code>slsc_cfg_select_stage</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. • In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von <code>slsc_cfg_select_stage</code>, mit der die syncAXIS control-Instanz initialisiert wird. • Für <code>slsc_cfg_select_stage</code> gibt es keine entsprechende Job-Funktion (<code>slsc_list_*</code>). • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.0.
Verweise	<code>slsc_cfg_acquire_stage</code> (veraltet), <code>slsc_ctrl_follow</code>, <code>slsc_ctrl_unfollow</code>

Name der Funktion	<code>slsc_cfg_select_stage_axis</code> (veraltet)
Zweck	Veraltet. Hilfs-Funktion zum Angeben des Ziel-Verfahrtischs. Ab <code>syncAXIS-DLL</code> ≥ V1.2.0 ist <code>slsc_cfg_select_stage</code> zu verwenden!
Funktions-Signatur	<code>uint32_t slsc_cfg_select_stage_axis (veraltet)(size_t Handle, uint32_t StageAxisX, uint32_t StageAxisY, uint32_t SlecEtherCATNodeID, uint32_t DriveEtherCATNodeID, uint32_t CorrectionFileIndex);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
	StageAxisX X-Achse des Verfahrtischs, die verfahren werden soll. Der anzugebende Parameterwert wird dem Kunden in Zusammenarbeit mit ACS bekannt gegeben.
	StageAxisY Y-Achse des Verfahrtischs, die verfahren werden soll. Der anzugebende Parameterwert wird dem Kunden in Zusammenarbeit mit ACS bekannt gegeben.
	SlecEtherCATNodeID Position des SLEC im EtherCAT-Netzwerk. Der anzugebende Parameterwert wird dem Kunden in Zusammenarbeit mit ACS bekannt gegeben.
	DriveEtherCATNodeID Position des Drive im EtherCAT-Netzwerk. Der anzugebende Parameterwert wird dem Kunden in Zusammenarbeit mit ACS bekannt gegeben.
	CorrectionFileIndex Index der Korrekturdatei, die verwendet werden soll (Korrekturdateien sind in der <code>syncAXISConfig.xml</code> angegeben; siehe auch Abschnitt "Korrekturdatei-bezogene Funktionen", Seite 99). Erlaubte Werte: 0...3.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Um <code>slsc_cfg_select_stage_axis (veraltet)</code> verwenden zu können, muss ein Dongle verwendet werden, der die Verwendung mehrerer Verfahrtische erlaubt (Standard-Dongle nicht ausreichend)! Sonst ist beim Rückgabewert Bit #31 gesetzt (<code>InvalidOrMissingDongle</code>). Dem Aufruf von <code>slsc_cfg_select_stage_axis (veraltet)</code> muss <code>slsc_cfg_acquire_stage (veraltet)</code> folgen, damit die Parameter-Werte angewendet werden. Siehe auch Kapitel 2.12.2 "Beispiel – Verfahrtisch vorübergehend freigeben und Ziel-Verfahrtisch wechseln", Seite 74. <code>slsc_cfg_select_stage_axis (veraltet)</code> ändert die aktuelle Konfiguration der <code>syncAXIS control</code>-Instanz. Die <code>syncAXIS control</code>-Instanz wird dabei nicht reinitialisiert. In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von <code>slsc_cfg_select_stage_axis (veraltet)</code>, mit der die <code>syncAXIS control</code>-Instanz initialisiert wird. Zur Zulässigkeit von <code>syncAXIS control</code>-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab <code>syncAXIS-DLL</code> V0.9.0...1.1.0.
Verweise	<code>slsc_cfg_select_stage</code>

Name der Funktion	<code>slsc_cfg_set_bandwidth</code>
Zweck	Ändert den <code>FilterBandwidth</code> -Wert der angegebenen syncAXIS control-Instanz .
Funktions-Signatur	<code>uint32_t slsc_cfg_set_bandwidth(size_t Handle, double FilterBandwidth);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	<code>FilterBandwidth</code> Gewünschter <code>FilterBandwidth</code> -Wert. In Hz. Werte <0,23 sind nicht zulässig. Anderenfalls ist beim Rückgabewert Bit #06 gesetzt (<code>UnplausibleOrUnknownParameter</code>). Typische Werte liegen zwischen 1...3 Hz, abhängig vom Scan-Kopf-Arbeitsfeld und dem Dynamikbereich des Verfahrtschis.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_cfg_set_bandwidth</code> setzt voraus, dass der Betriebsmodus "ScannerAndStage" aktiv ist. Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). <code>FilterBandwidth</code> und der <code>syncAXISConfig.xml</code>-Tag <code>FilterBandwidth</code> entsprechen einander. Bei der Initialisierung der syncAXIS control-Instanz (durch <code>slsc_cfg_initialize_from_file</code>) wird der <code>FilterBandwidth</code>-Wert aus der <code>syncAXISConfig.xml</code> ausgelesen. <code>slsc_cfg_set_bandwidth</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten <code>slsc_list_begin*</code>. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. Für <code>slsc_cfg_set_bandwidth</code> gibt es keine entsprechende Job-Funktion (<code>slsc_list_*</code>). Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0.
Verweise	–

Name der Funktion	slsc_cfg_set_calculation_dynamics_jump_scan_device
Zweck	<p>Ändert die Einstellung der angegebenen syncAXIS control-Instanz für:</p> <ul style="list-style-type: none"> Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Sprünge, aber nicht Markierungen
Funktions-Signatur	<pre>uint32_t slsc_cfg_set_calculation_dynamics_jump_scan_device(size_t Handle, double JumpAngularAcc, double JumpAngularJerk);</pre>
Argument(e)	<p>Handle Handle auf eine syncAXIS control-Instanz.</p>
	<p>JumpAngularAcc Entspricht Acceleration unter <cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:JumpDynamics>.</p>
	<p>JumpAngularJerk Entspricht Jerk unter <cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:JumpDynamics>.</p>
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> ⚠ Vorsicht! syncAXIS control verwendet den Wert von Acceleration = JumpAngularAcc = JumpAngularAcc zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. ⚠ Vorsicht! syncAXIS control verwendet den Wert von Jerk = JumpAngularJerk = JumpAngularJerk zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. slsc_cfg_set_calculation_dynamics_jump_scan_device ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_calculation_dynamics_jump_scan_device ist in erlaubt im Betriebsmodus "ScannerOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_calculation_dynamics_jump_scan_device, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:JumpDynamics> → <cfg:Acceleration> <cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:JumpDynamics> → <cfg:Jerk> Für slsc_cfg_set_calculation_dynamics_jump_scan_device gibt es: <ul style="list-style-type: none"> Eine entsprechende Job-Funktion (slsc_list_*) slsc_list_set_calculation_dynamics_jump_scan_device Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Abfragen der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_calculation_dynamics_jump_scan_device

Name der Funktion	slsc_cfg_set_calculation_dynamics_jump_scan_device
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0.
Verweise	slsc_cfg_get_calculation_dynamics_jump_scan_device , slsc_list_set_calculation_dynamics_jump_scan_device

Name der Funktion	<code>slsc_cfg_set_calculation_dynamics_mark_scan_device</code>
Zweck	<p>Ändert die Einstellung der angegebenen syncAXIS control-Instanz für:</p> <ul style="list-style-type: none"> Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Markierungen, aber nicht Sprünge
Funktions-Signatur	<pre>uint32_t slsc_cfg_set_calculation_dynamics_mark_scan_device(size_t Handle, double MarkAngularAcc, double MarkAngularJerk);</pre>
Argument(e)	<p>Handle Handle auf eine syncAXIS control-Instanz.</p>
	<p>MarkAngularAcc Entspricht Acceleration unter <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:MarkDynamics></code>.</p>
	<p>MarkAngularJerk Entspricht Jerk unter <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:MarkDynamics></code>.</p>
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> ⚠ Vorsicht! syncAXIS control verwendet den Wert von Acceleration = MarkAngularAcc = MarkAngularAcc zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. ⚠ Vorsicht! syncAXIS control verwendet den Wert von Jerk = MarkAngularJerk = MarkAngularJerk zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. slsc_cfg_set_calculation_dynamics_mark_scan_device ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_calculation_dynamics_mark_scan_device ist in erlaubt im Betriebsmodus "ScannerOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_calculation_dynamics_mark_scan_device, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:MarkDynamics> → <cfg:Acceleration></code> <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:MarkDynamics> → <cfg:Jerk></code> Für slsc_cfg_set_calculation_dynamics_mark_scan_device gibt es: <ul style="list-style-type: none"> Eine entsprechende Job-Funktion (slsc_list_*) slsc_list_set_calculation_dynamics_mark_scan_device Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Abfragen der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_calculation_dynamics_mark_scan_device

Name der Funktion	slsc_cfg_set_calculation_dynamics_mark_scan_device
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0.
Verweise	slsc_cfg_get_calculation_dynamics_mark_scan_device , slsc_list_set_calculation_dynamics_mark_scan_device

Name der Funktion	<code>slsc_cfg_set_calculation_dynamics_stage</code>
Zweck	<p>Ändert die Einstellung der angegebenen syncAXIS control-Instanz für:</p> <ul style="list-style-type: none"> Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Verfahrtsch-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Verfahrtsch-Bewegung verwendet
Funktions-Signatur	<code>uint32_t slsc_cfg_set_calculation_dynamics_stage(size_t Handle, slsc_Stage Stage, double StageVel, double StageAcc, double StageJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Stage Siehe enum slsc_Stage .
	StageVel Entspricht Velocity unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:CalculationDynamics></code> .
	StageAcc Entspricht Acceleration unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:CalculationDynamics></code> .
	StageJerk Entspricht Jerk unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:CalculationDynamics></code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> ⚠ Vorsicht! syncAXIS control verwendet die Werte bei Velocity, Acceleration und Jerk zur Planung von Trajektorien für den Betriebsmodus "StageOnly" sowie für die Endbewegung an Job-Enden. Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. slsc_cfg_set_calculation_dynamics_stage ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_calculation_dynamics_stage ist in erlaubt im Betriebsmodus "StageOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_calculation_dynamics_stage, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:CalculationDynamics></code> Für slsc_cfg_set_calculation_dynamics_stage gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Abfragen der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_calculation_dynamics_stage

Name der Funktion	slsc_cfg_set_calculation_dynamics_stage
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_calculation_dynamics_stage

Name der Funktion	<code>slsc_cfg_set_contour_dependent_speed_control_2d</code>
Zweck	<p>„Konturabhängige Geschwindigkeitsberechnung“ an-/ ausschalten. Außerdem wird eingestellt, wie (die syncAXIS control-Instanz intern) die Geschwindigkeiten entlang von Kurven bestimmt („links“ oder „rechts“ der Kurvenlinien-Mitte; Abstand zu dieser). Bei aktivierter „Automatische Lasersteuerung“ werden diese Ergebnisse dazu verwendet, um entsprechend dazu z.B. die Laser-Spot-Abstände äquidistant zu setzen.</p>
Funktions-Signatur	<pre>uint32_t slsc_cfg_set_contour_dependent_speed_control_2d(size_t Handle, int32_t Direction, double SpotRadius);</pre>
Argument(e)	<p>Handle Handle auf eine syncAXIS control-Instanz.</p>
	<p>Direction 0: „Konturabhängige Geschwindigkeitsberechnung“ = Aus. Die Geschwindigkeiten werden auf der Kurvenlinien-Mitte bestimmt. Ist auch der Default-Zustand nach der Initialisierung der syncAXIS control-Instanz mit slsc_cfg_initialize_from_file.</p> <p>+ 1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden rechts der Kurvenlinien-Mitte bestimmt.</p> <p>– 1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden links der Kurvenlinien-Mitte bestimmt.</p>
	<p>SpotRadius Radius des Laserspots in der Arbeitsebene. In mm. Der Wert legt fest, wie weit rechts oder links von der Kurvenlinien-Mitte entfernt die Geschwindigkeitswerte bestimmt werden.</p>
Rückgabewert	Siehe Kapitel 4 „Standard-Rückgabewerte der syncAXIS-DLL Funktionen“ , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_set_contour_dependent_speed_control_2d ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. • In der syncAXISConfig.xml gibt es <i>keine</i> Default-Werte für die Argumente von slsc_cfg_set_contour_dependent_speed_control_2d, mit der die syncAXIS control-Instanz initialisiert wird. • slsc_cfg_set_contour_dependent_speed_control_2d hat keine Auswirkung (es wird kein Fehler zurückgegeben), wenn die „Automatische Lasersteuerung“ nicht eingeschaltet ist (z. B. es ist kein ActiveChannel in der syncAXISConfig.xml eingetragen). • Siehe auch Kapitel 2.9.5 „Über die „Konturabhängige Geschwindigkeitsberechnung““, Seite 60.

Name der Funktion	<code>slsc_cfg_set_contour_dependent_speed_control_2d</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Die entsprechende Job-Funktion (<code>slsc_list_*</code>) von <code>slsc_cfg_set_contour_dependent_speed_control_2d</code> ist <code>slsc_list_set_contour_dependent_speed_control_2d</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	<code>slsc_list_set_contour_dependent_speed_control_2d</code>

Name der Funktion	<code>slsc_cfg_set_dynamic_limits_scan_device</code>
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die maximalen dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Scan-Device-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_set_dynamic_limits_scan_device(size_t Handle, double AngularVel, double AngularAcc, double AngularJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	AngularVel Entspricht Velocity unter <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:DynamicLimits></code> .
	AngularAcc Entspricht Acceleration unter <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:DynamicLimits></code> .
	AngularJerk Entspricht Jerk unter <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:DynamicLimits></code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_dynamic_limits_scan_device ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_dynamic_limits_scan_device ist in erlaubt im Betriebsmodus "ScannerOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_dynamic_limits_scan_device, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:DynamicLimits></code> Für slsc_cfg_set_dynamic_limits_scan_device gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Abfragen der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_dynamic_limits_scan_device Überschreitungen lösen automatisch die in DynamicViolationReaction oder slsc_cfg_set_dynamic_violation_reaction festgelegte Reaktion aus. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_dynamic_limits_scan_device , slsc_cfg_set_dynamic_violation_reaction , slsc_cfg_set_scan_device_dynamic_monitoring_level

Name der Funktion	slsc_cfg_set_dynamic_limits_stage
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Dynamikgrenzen des vorgesehenen Verfahrtsch-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_set_dynamic_limits_stage(size_t Handle, slsc_Stage Stage, double StageVel, double StageAcc, double StageJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Stage Siehe enum slsc_Stage .
	StageVel Entspricht Velocity unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:DynamicLimits></code> .
	StageAcc Entspricht Acceleration unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:DynamicLimits></code> .
	StageJerk Entspricht Jerk unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:DynamicLimits></code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_dynamic_limits_stage ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_dynamic_limits_stage ist in erlaubt im Betriebsmodus "StageOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_dynamic_limits_stage, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:DynamicLimits></code> Für slsc_cfg_set_dynamic_limits_stage gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Abfragen der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_dynamic_limits_stage Überschreitungen lösen automatisch die in DynamicViolationReaction oder slsc_cfg_set_dynamic_violation_reaction festgelegte Reaktion aus. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_dynamic_limits_stage , slsc_cfg_set_dynamic_violation_reaction , slsc_cfg_set_stage_dynamic_monitoring_level

Name der Funktion	slsc_cfg_set_dynamic_violation_reaction
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Reaktion bei einer Dynamik-Grenzwertüberschreitung
Funktions-Signatur	<code>uint32_t slsc_cfg_set_dynamic_violation_reaction(size_t Handle, slsc_DynamicViolationReaction DynamicViolationReaction);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	DynamicViolationReaction Siehe enum slsc_DynamicViolationReaction .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_dynamic_violation_reaction ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_dynamic_violation_reaction ist in erlaubt im Betriebsmodus "ScannerOnly", "StageOnly", "ScannerAndStage". In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_dynamic_violation_reaction, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:GeneralConfig> → <cfg:DynamicViolationReaction></code> Für slsc_cfg_set_dynamic_violation_reaction gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file // Nur Warnungen slsc_cfg_set_dynamic_violation_reaction(Handle, slsc_DynamicViolationReaction_WarningOnly);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_dynamic_violation_reaction

Name der Funktion	<code>slsc_cfg_set_field_limits_scan_device</code>
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Arbeitsfeldgrenzen des vorgesehenen Scan-Device-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_set_field_limits_scan_device(size_t Handle, const double* FieldLimitsMin, const double* FieldLimitsMax);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	FieldLimitsMin Zeiger auf ein Array der Dimension 2. Entspricht den Attributen Min von <code><cfg:XDirection Min="..." /></code> und <code><cfg:YDirection Min="..." /></code> unter <code><cfg:Configuration></code> → <code><cfg:ScanDeviceConfig></code> → <code><cfg:FieldLimits></code> .
	FieldLimitsMax Zeiger auf ein Array der Dimension 2. Entspricht den Attributen Max von <code><cfg:XDirection Max="..." /></code> und <code><cfg:YDirection Max="..." /></code> unter <code><cfg:Configuration></code> → <code><cfg:ScanDeviceConfig></code> → <code><cfg:FieldLimits></code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_cfg_set_field_limits_scan_device</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten <code>slsc_list_begin*</code>. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. <code>slsc_cfg_set_field_limits_scan_device</code> ist in erlaubt im Betriebsmodus "ScannerOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von <code>slsc_cfg_set_field_limits_scan_device</code>, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration></code> → <code><cfg:ScanDeviceConfig></code> → <code><cfg:FieldLimits></code> Für <code>slsc_cfg_set_field_limits_scan_device</code> gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (<code>slsc_list_*</code>) Keine entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) Zum Abfragen der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> <code>slsc_cfg_get_field_limits_scan_device</code> Überschreitungen lösen automatisch die in <code>DynamicViolationReaction</code> oder <code>slsc_cfg_set_dynamic_violation_reaction</code> festgelegte Reaktion aus. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. double FieldLimitsMin[2] = { -10.0, -10.0 }; double FieldLimitsMax[2] = { 10.0, 10.0 }; // Handle: see Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_set_field_limits_scan_device(Handle, FieldLimitsMin, FieldLimitsMax);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_cfg_get_field_limits_scan_device</code>

Name der Funktion	<code>slsc_cfg_set_field_limits_stage</code>
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Arbeitsfeldgrenzen des vorgesehenen Verfahrtsch-Typs
Funktions-Signatur	<code>uint32_t slsc_cfg_set_field_limits_stage(size_t Handle, slsc_Stage Stage, const double* FieldLimitsMin, const double* FieldLimitsMax);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Stage Siehe enum slsc_Stage .
	FieldLimitsMin Zeiger auf ein Array der Dimension 2. Entspricht den Attributen Min von <code><cfg:XDirection Min="" ... /></code> und <code><cfg:YDirection Min="" ... /></code> unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:FieldLimits></code> .
	FieldLimitsMax Zeiger auf ein Array der Dimension 2. Entspricht den Attributen Max von <code><cfg:XDirection Max="" ... /></code> und <code><cfg:YDirection Max="" ... /></code> unter <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:FieldLimits></code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_field_limits_stage ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_field_limits_stage ist in erlaubt im Betriebsmodus "StageOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_field_limits_stage, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:FieldLimits></code> Für slsc_cfg_set_field_limits_stage gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Abfragen der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_field_limits_stage Überschreitungen lösen automatisch die in <code>DynamicViolationReaction</code> oder slsc_cfg_set_dynamic_violation_reaction festgelegte Reaktion aus. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.

Name der Funktion	slsc_cfg_set_field_limits_stage
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. double FieldLimitsMin[2] = { -150.0, -150.0 }; double FieldLimitsMax[2] = { 150.0, 150.0 }; // Handle: see Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_set_field_limits_stage(Handle, slsc_Stage1, FieldLimitsMin, FieldLimitsMax);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_field_limits_stage

Name der Funktion	<code>slsc_cfg_set_jump_speed</code>
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Sprunggeschwindigkeit
Funktions-Signatur	<code>uint32_t slsc_cfg_set_jump_speed(size_t Handle, double JumpSpeed);</code>
Argument(e)	<code>Handle</code> Handle auf eine syncAXIS control-Instanz .
	<code>JumpSpeed</code> Geschwindigkeit. In mm/s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_cfg_set_jump_speed</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. <code>slsc_cfg_set_jump_speed</code> ist in erlaubt im Betriebsmodus "ScannerOnly", "StageOnly", "ScannerAndStage". In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von <code>slsc_cfg_set_jump_speed</code>, mit der die syncAXIS control-Instanz initialisiert wird. <ul style="list-style-type: none"> – <code><cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:JumpSpeed ...></code> Vor dem ersten Aufruf von <code>slsc_cfg_set_jump_speed</code> sind in der syncAXIS control-Instanz die Konfigurations-Parameter-Werte möglicherweise nicht mehr so eingestellt, wie in der syncAXISConfig.xml (siehe slsc_cfg_initialize_from_file) angegeben. Zwischenzeitlich könnte die Sprunggeschwindigkeit auch durch slsc_cfg_set_trajectory_config geändert worden sein. Die entsprechende Job-Funktion (<code>slsc_list_*</code>) von <code>slsc_cfg_set_jump_speed</code> ist slsc_list_set_jump_speed. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_cfg_initialize_from_file, slsc_cfg_set_trajectory_config, slsc_list_set_jump_speed

Name der Funktion	<code>slsc_cfg_set_list_handling_mode</code>
Zweck	Stellt das Handling und das Rückgabeverhalten der Job-Funktionen (<code>slsc_list_*</code>) ein.
Funktions-Signatur	<code>uint32_t slsc_cfg_set_list_handling_mode(size_t Handle, slsc_ListHandlingMode Mode, bool (*Predicate)(uint32_t));</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mode Siehe enum slsc_ListHandlingMode .
	Predicate Funktions-Signatur für die benutzerdefinierte Funktion.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_set_list_handling_mode ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. • In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_list_handling_mode, mit der die syncAXIS control-Instanz initialisiert wird. • slsc_cfg_set_list_handling_mode verändert das Verhalten aller Job-Funktionen (<code>slsc_list_*</code>). • Das Argument <code>Predicate</code> wird nur für <code>Mode = slsc_ListHandlingMode_RepeatWhilePredicate</code> ausgewertet (nur für diesen Modus kann eine Predicate-Funktion angegeben werden). Bei <code>slsc_ListHandlingMode_ReturnAtOnce</code> und <code>slsc_ListHandlingMode_RepeatWhileBufferFull</code> wird keine Predicate-Funktion benötigt. • Vor dem ersten Aufruf von slsc_cfg_set_list_handling_mode sind in der syncAXIS control-Instanz die Konfigurations-Parameter-Werte so eingestellt, wie in der <code>syncAXISConfig.xml</code> (siehe slsc_cfg_initialize_from_file) angegeben. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.

Name der Funktion	slsc_cfg_set_list_handling_mode
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. // Example for "ReturnAtOnce" slsc_cfg_set_list_handling_mode(Handle, slsc_ListHandlingMode::slsc_ListHandlingMode_ReturnAtOnce, nullptr); // Example for "RepeatWhileBufferFull" slsc_cfg_set_list_handling_mode(Handle, slsc_ListHandlingMode::slsc_ListHandlingMode_RepeatWhileBufferFull, nullptr); // Example for "RepeatWhilePredicate" slsc_cfg_set_list_handling_mode(Handle, slsc_ListHandlingMode::slsc_ListHandlingMode_RepeatWhilePredicate, [](uint32_t RetVal) { bool Flag = (0x0010 == RetVal); if (Flag) { std::this_thread::sleep_for(std::chrono::milliseconds(1)); } return Flag; }));</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_cfg_initialize_from_file

Name der Funktion	<code>slsc_cfg_set_list_handling_mode_with_context</code>
Zweck	Wie <code>slsc_cfg_set_list_handling_mode</code> . Aber es kann zusätzlich ein Kontext angegeben werden. Stellt das Handling und das Rückgabeverhalten der Job-Funktionen (<code>slsc_list_*</code>) ein.
Funktions-Signatur	<code>uint32_t slsc_cfg_set_list_handling_mode_with_context (size_t Handle, slsc_ListHandlingMode Mode, bool (*Predicate)(uint32_t, void*), void* Context);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mode Siehe enum slsc_ListHandlingMode .
	Predicate Funktions-Signatur für die benutzerdefinierte Funktion.
	Context Zeiger auf ein benutzerdefiniertes Objekt. Auf dieses Objekt kann in der Funktion <code>Predicate</code> zugegriffen werden.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_cfg_set_list_handling_mode_with_context</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von <code>slsc_cfg_set_list_handling_mode_with_context</code>, mit der die syncAXIS control-Instanz initialisiert wird. <code>slsc_cfg_set_list_handling_mode_with_context</code> verändert (genauso wie slsc_cfg_set_list_handling_mode) das Verhalten aller Job-Funktionen (<code>slsc_list_*</code>). Das Argument <code>Predicate</code> wird nur für <code>Mode = slsc_ListHandlingMode_RepeatWhilePredicate</code> ausgewertet (nur für diesen Modus kann eine Predicate-Funktion angegeben werden). Bei <code>slsc_ListHandlingMode_ReturnAtOnce</code> und <code>slsc_ListHandlingMode_RepeatWhileBufferFull</code> wird keine Predicate-Funktion benötigt. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	slsc_cfg_set_list_handling_mode , slsc_cfg_initialize_from_file

Name der Funktion	<code>slsc_cfg_set_mark_speed</code>
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Markiergeschwindigkeit
Funktions-Signatur	<code>uint32_t slsc_cfg_set_mark_speed(size_t Handle, double MarkSpeed);</code>
Argument(e)	<code>Handle</code> Handle auf eine syncAXIS control-Instanz .
	<code>MarkSpeed</code> Geschwindigkeit. In mm/s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_cfg_set_mark_speed</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. <code>slsc_cfg_set_mark_speed</code> ist in erlaubt im Betriebsmodus "ScannerOnly", "StageOnly", "ScannerAndStage". In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_mark_speed, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:MarkSpeed ...></code> Vor dem ersten Aufruf von slsc_cfg_set_mark_speed sind in der syncAXIS control-Instanz die Konfiguration-Parameter-Werte möglicherweise nicht mehr so eingestellt, wie in der syncAXISConfig.xml (siehe slsc_cfg_initialize_from_file) angegeben. Zwischenzeitlich könnte die Markiergeschwindigkeit auch durch slsc_cfg_set_trajectory_config geändert worden sein. Die entsprechende Job-Funktion (slsc_list_*) von slsc_cfg_set_mark_speed ist slsc_list_set_mark_speed. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_cfg_initialize_from_file , slsc_cfg_set_trajectory_config , slsc_list_set_mark_speed

Name der Funktion	slsc_cfg_set_matrix_and_offset
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Zielpunkt-Koordinaten gemäß einer Transformationsmatrix und einem Offset-Wert
Funktions-Signatur	<code>uint32_t slsc_cfg_set_matrix_and_offset(size_t Handle, const double* Matrix, const double* Offset);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Matrix Zeiger auf ein Array der Dimension 4. Koeffizienten m11...m22 einer (2 × 2)-Transformationsmatrix.
	Offset Zeiger auf ein Array der Dimension 2. x-Wert und y-Wert, um den die Zielpunkte im Arbeitsfeld verschoben werden. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_matrix_and_offset ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. In der syncAXISConfig.xml gibt es <i>keine</i> Default-Werte für die Argumente von slsc_cfg_set_matrix_and_offset, mit der die syncAXIS control-Instanz initialisiert wird. Zielpunkt-Koordinaten von Job-Funktionen (slsc_list_*): siehe Listenpunkt auf Seite 268. slsc_cfg_set_matrix_and_offset berechnet die neuen Zielpunkte wie slsc_list_set_matrix_and_offset, siehe Listenpunkt auf Seite 266. Mit geeigneten Transformationsmatrix-Koeffizienten (Argument Matrix) kann z.B. ein Skalieren, Drehen oder Umklappen von Markiermustern erzielt werden. Siehe auch Abschnitt "Funktionen zum Ändern der Zielpunkt-Koordinaten", Seite 92. Die entsprechende Job-Funktion (slsc_list_*) von slsc_cfg_set_matrix_and_offset ist slsc_list_set_matrix_and_offset. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	slsc_list_set_matrix_and_offset

Name der Funktion	<code>slsc_cfg_set_mode</code>
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Den Betriebsmodus (<code>ScannerOnly</code>, <code>StageOnly</code>, <code>ScannerAndStage</code>)
Funktions-Signatur	<code>uint32_t slsc_cfg_set_mode(size_t Handle, slsc_OperationMode Mode);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mode Siehe enum slsc_OperationMode .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_mode ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei <i>wird dabei reinitialisiert!</i> Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. In der <code>syncAXISConfig.xml</code> gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_mode, mit der die syncAXIS control-Instanz initialisiert wird. slsc_cfg_set_mode wird nicht akzeptiert, wenn gerade ein Job ausgeführt wird. Dann ist beim Rückgabewert Bit #03 gesetzt (<code>NotAllowedInExecuting</code>). Nach slsc_cfg_set_mode werden alle Jobs aus der Job-Queue gelöscht. Zu den dynamischen Grenzen, die in den einzelnen Betriebsmodi verwendet werden, siehe enum slsc_OperationMode. Im Betriebsmodus <code>ScannerOnly</code> werden die auszuführenden Bewegungen des Jobs nur an den Scan-Kopf gesendet. Im Betriebsmodus <code>StageOnly</code> werden die auszuführenden Bewegungen des Jobs nur an den Verfahrtschiff gesendet. Im Betriebsmodus <code>ScannerAndStage</code> werden die auszuführenden Bewegungen des Jobs entsprechend auf Verfahrtschiff und Scan-Kopf verteilt ("Bewegungsaufteilung"). Der Betriebsmodus der syncAXIS control-Instanz wird mit slsc_cfg_get_mode abgefragt. Wird slsc_cfg_set_mode nicht aufgerufen, dann bleibt der Betriebsmodus der syncAXIS control-Instanz so eingestellt wie in der (zur Initialisierung benutzten) <code>syncAXISConfig.xml</code> (siehe slsc_cfg_initialize_from_file) angegeben ist. Ist eine Funktion (gilt ganz allgemein in syncAXIS control) wegen des aktuellen Betriebsmodus nicht erlaubt, dann ist beim Rückgabewert das Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_cfg_set_mode(Handle, ScannerOnly);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0 .
Verweise	slsc_cfg_get_mode

Name der Funktion	<code>slsc_cfg_set_part_displacement</code>
Zweck	Wendet eine <code>Matrix</code> und einen <code>Offset</code> auf die Soll-Trajektorie für das angegebene Scan-Device (Scan-Kopf) an. Siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332.
Funktions-Signatur	<code>uint32_t slsc_cfg_set_part_displacement(size_t Handle, slsc_ScanDevice ScanDevice, const double* Matrix, const double* Offset);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	ScanDevice Siehe enum <code>slsc_ScanDevice</code> .
	Matrix Zeiger auf ein Array der Dimension 4. Koeffizienten $m_{11}...m_{22}$ einer (2×2) -Transformationsmatrix.
	Offset Zeiger auf ein Array der Dimension 2. x-Wert und y-Wert, um den die Zielpunkte im Arbeitsfeld verschoben werden. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_cfg_set_part_displacement</code> ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten <code>slsc_list_begin*</code>. Sie wird bis zum nächsten <code>slsc_cfg_delete</code> oder <code>slsc_cfg_reinitialize_from_file</code> beibehalten. In der <code>syncAXISConfig.xml</code> kann wahlweise für das angegebene Scan-Device eine "Basis"-Transformation eingestellt werden (Tag <code><cfg:BasePartDisplacement></code>). <code>Matrix</code> und <code>Offset</code> von <code>slsc_cfg_set_part_displacement</code> wirken zusätzlich zu dieser Basistransformation, siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332. <code>slsc_cfg_set_part_displacement</code> ist dazu vorgesehen, um kleinere Offsets und Rotationen auszugleichen. Wie viel genau ausgeglichen werden kann, ist durch die Scan-Device-Dynamik beschränkt und hängt stark von Markiergeschwindigkeit, Größe und Form des Markiermusters sowie der Arbeitsfeldsgröße des Scan-Kopfs ab. Die Auswirkung von <code>Matrix</code> und <code>Offset</code> sind im Simulationsergebnis sichtbar. Für <code>slsc_cfg_set_part_displacement</code> gibt es keine entsprechende Job-Funktion (<code>slsc_list_*</code>). Mit <code>slsc_cfg_set_part_displacement</code> können Transformationen, die von Vision-Systemen empfangen werden, angewendet werden. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.0.
Verweise	–

Name der Funktion	<code>slsc_cfg_set_rot_and_offset_2d</code>
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Zielpunkt-Koordinaten um einen Winkel und Offset-Wert
Funktions-Signatur	<code>uint32_t slsc_cfg_set_rot_and_offset_2d(size_t Handle, double Angle, const double* Offset);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Angle Winkel (um den Ursprung 0,0), um den die Zielpunkte im Arbeitsfeld gedreht werden. In rad. Positive Werte: Drehung erfolgt gegen den Uhrzeigersinn. Negative Werte: Drehung erfolgt im Uhrzeigersinn.
	Offset Zeiger auf ein Array der Dimension 2. x-Wert und y-Wert, um den die Zielpunkte im Arbeitsfeld verschoben werden. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_rot_and_offset_2d ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. In der syncAXISConfig.xml gibt es <i>keine</i> Default-Werte für die Argumente von slsc_cfg_set_rot_and_offset_2d, mit der die syncAXIS control-Instanz initialisiert wird. Zielpunkt-Koordinaten von Job-Funktionen (slsc_list_*): siehe Listenpunkt auf Seite 268. slsc_cfg_set_rot_and_offset_2d berechnet die neuen Zielpunkte wie slsc_list_set_rot_and_offset_2d, siehe dort. Die entsprechende Job-Funktion (slsc_list_*) von slsc_cfg_set_rot_and_offset_2d ist slsc_list_set_rot_and_offset_2d. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_list_arc_abs , slsc_list_circle_2d_abs , slsc_list_jump_abs , slsc_list_mark_abs , slsc_list_set_rot_and_offset_2d

Name der Funktion	<code>slsc_cfg_set_scan_device_dynamic_monitoring_level</code>
Zweck	<p>Ändert die Einstellung der angegebenen syncAXIS control-Instanz für:</p> <ul style="list-style-type: none"> Das Kriterium, auf das die Scan-Devices hin überwacht werden sollen (z.B. <code>slsc_DynamicsMonitoringLevel_Velocity</code>)
Funktions-Signatur	<code>uint32_t slsc_cfg_set_scan_device_dynamic_monitoring_level(size_t Handle, slsc_DynamicsMonitoringLevel DynamicMonitoringLevel);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	DynamicMonitoringLevel Siehe enum slsc_DynamicsMonitoringLevel .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_scan_device_dynamic_monitoring_level ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_scan_device_dynamic_monitoring_level ist in erlaubt im Betriebsmodus "ScannerOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_scan_device_dynamic_monitoring_level, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:MonitoringLevel></code> Für slsc_cfg_set_scan_device_dynamic_monitoring_level gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_scan_device_dynamic_monitoring_level Überschreitungen lösen automatisch die in DynamicViolationReaction oder slsc_cfg_set_dynamic_violation_reaction festgelegte Reaktion aus. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file // Position UND Geschwindigkeit slsc_cfg_set_scan_device_dynamic_monitoring_level(Handle, slsc_DynamicsMonitoringLevel_Velocity);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_scan_device_dynamic_monitoring_level , slsc_cfg_set_dynamic_violation_reaction , slsc_cfg_set_dynamic_limits_scan_device

Name der Funktion	slsc_cfg_set_simulation_setting
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Simulationseinstellung
Funktions-Signatur	<code>uint32_t slsc_cfg_set_simulation_setting(size_t Handle, slsc_SimulationSetting SimulationSetting);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	SimulationSetting Siehe enum slsc_SimulationSetting .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_simulation_setting ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei <i>wird dabei reinitialisiert!</i> Die Änderung wird zur Laufzeit wirksam zum Zeitpunkt des Aufrufs. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_simulation_setting ist wirkungslos, wenn die angesprochene syncAXIS control-Instanz bereits die Simulationseinstellung hat. In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_simulation_setting, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> – <code><cfg:Configuration> → <cfg:GeneralConfig> → <cfg:SimulationConfig> → <cfg:SimulationMode></code> slsc_cfg_set_simulation_setting wird nicht akzeptiert, wenn gerade ein Job ausgeführt wird. Dann ist beim Rückgabewert Bit #03 gesetzt (NotAllowedInExecuting). Nach slsc_cfg_set_simulation_setting werden alle Jobs aus der Job-Queue gelöscht. Die Simulationseinstellung der syncAXIS control-Instanz wird mit slsc_cfg_get_simulation_setting abgefragt. Siehe auch Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31. Für slsc_cfg_set_simulation_setting gibt es: <ul style="list-style-type: none"> – Keine entsprechende Job-Funktion (slsc_list_*) – Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file // Hardwaremodus // slsc_cfg_set_simulation_setting(Handle, slsc_SimulationSetting_HardwareMode); // Simulationsmodus slsc_cfg_set_simulation_setting(Handle, slsc_SimulationSetting_SimulationMode);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_simulation_setting

Name der Funktion	<code>slsc_cfg_set_stage_dynamic_monitoring_level</code>
Zweck	<p>Ändert die Einstellung der angegebenen syncAXIS control-Instanz für:</p> <ul style="list-style-type: none"> Das Kriterium, auf das die Verfahrtsche hin überwacht werden sollen (z.B. <code>slsc_DynamicsMonitoringLevel_Velocity</code>)
Funktions-Signatur	<code>uint32_t slsc_cfg_set_stage_dynamic_monitoring_level(size_t Handle, slsc_DynamicsMonitoringLevel DynamicMonitoringLevel);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	DynamicMonitoringLevel Siehe enum slsc_DynamicsMonitoringLevel .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_stage_dynamic_monitoring_level ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. slsc_cfg_set_stage_dynamic_monitoring_level ist in erlaubt im Betriebsmodus "StageOnly" und "ScannerAndStage". Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_stage_dynamic_monitoring_level, mit der die syncAXIS control-Instanz initialisiert wird: <ul style="list-style-type: none"> <code><cfg:Configuration> → <cfg:StageConfig> → <cfg:MonitoringLevel></code> Für slsc_cfg_set_stage_dynamic_monitoring_level gibt es: <ul style="list-style-type: none"> Keine entsprechende Job-Funktion (slsc_list_*) Keine entsprechende Kontroll-Funktion (slsc_ctrl_*) Zum Ändern der Einstellung steht zur Verfügung: <ul style="list-style-type: none"> slsc_cfg_get_stage_dynamic_monitoring_level Überschreitungen lösen automatisch die in DynamicViolationReaction oder slsc_cfg_set_dynamic_violation_reaction festgelegte Reaktion aus. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file // Position UND Geschwindigkeit slsc_cfg_set_stage_dynamic_monitoring_level(Handle, slsc_DynamicsMonitoringLevel_Velocity);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_cfg_get_stage_dynamic_monitoring_level , slsc_cfg_set_dynamic_violation_reaction , slsc_cfg_set_dynamic_limits_stage

Name der Funktion	slsc_cfg_set_trajectory_config
Zweck	Ändert die Einstellung der angegebenen syncAXIS control-Instanz für: <ul style="list-style-type: none"> Die Konfiguration der Trajektorienplanung
Funktions-Signatur	<code>uint32_t slsc_cfg_set_trajectory_config(size_t Handle, const slsc_TrajectoryConfig* TrajConfig);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	TrajConfig Siehe Struktur slsc_TrajectoryConfig .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_cfg_set_trajectory_config ändert die aktuelle Konfiguration der syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung wird zur Laufzeit wirksam ab dem nächsten slsc_list_begin*. Sie wird bis zum nächsten slsc_cfg_delete oder slsc_cfg_reinitialize_from_file beibehalten. In der syncAXISConfig.xml gibt es Default-Wert(e) für die Argumente von slsc_cfg_set_trajectory_config, mit der die syncAXIS control-Instanz initialisiert wird. Um das Trajektorienkonfigurations-Objekts nach slsc_cfg_set_trajectory_config (zur Vermeidung von Speicherlecks) wieder zu löschen, steht slsc_cfg_delete_trajectory_config zur Verfügung. Vor dem ersten Aufruf von slsc_cfg_set_trajectory_config sind in der syncAXIS control-Instanz die Konfigurations-Parameter-Werte so eingestellt, wie in der syncAXISConfig.xml (siehe slsc_cfg_initialize_from_file) angegeben. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_cfg_delete_trajectory_config , slsc_cfg_get_trajectory_config , slsc_cfg_initialize_from_file

Name der Funktion	<code>slsc_ctrl_disable_laser</code>
Zweck	Unterbindet, dass die Lasersteuersignale LASERON, LASER1 und LASER2 (siehe RTC6-Handbuch) an der RTC6-Karte ausgegeben werden.
Funktions-Signatur	<code>uint32_t slsc_ctrl_disable_laser(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Vor <code>slsc_ctrl_disable_laser</code> muss <code>slsc_cfg_initialize_from_file</code> ausgeführt worden sein. <code>slsc_cfg_initialize_from_file</code> schaltet u.a. auch die Lasersteuerung auf der RTC6-Karte scharf (dazu wird intern der RTC-Kontrollbefehl <code>set_laser_control</code> benutzt). • Mit <code>slsc_ctrl_disable_laser</code> werden in der angegebenen syncAXIS control-Instanz die Lasersteuersignale unterdrückt, die zuvor mit <code>slsc_cfg_initialize_from_file</code> oder <code>slsc_ctrl_enable_laser</code> freigegeben wurden. • Das Verhalten von <code>slsc_ctrl_disable_laser</code> entspricht dem des RTC6-Kontrollbefehls <code>disable_laser</code> (siehe RTC6-Handbuch zu unterdrückten Ports, Port-Werten etc). <code>slsc_ctrl_disable_laser</code> hat jedoch einen Rückgabewert. • <code>slsc_ctrl_disable_laser</code> wird (wie alle Kontroll-Funktionen (<code>slsc_ctrl_*</code>)) immer akzeptiert, wenn der Betriebsstatus "grün" ist (siehe <code>slsc_cfg_get_operation_status</code>). • Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0 .
Verweise	slsc_ctrl_enable_laser , slsc_cfg_get_operation_status , slsc_cfg_initialize_from_file

Name der Funktion	<code>slsc_ctrl_enable_laser</code>
Zweck	Gibt die Lasersteuersignale LASERON, LASER1 und LASER2 (siehe RTC6-Handbuch) an der RTC6-Karte frei.
Funktions-Signatur	<code>uint32_t slsc_ctrl_enable_laser(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Vor <code>slsc_ctrl_enable_laser</code> muss <code>slsc_cfg_initialize_from_file</code> ausgeführt worden sein. • <code>slsc_ctrl_enable_laser</code> ist nicht erlaubt, wenn gerade eine Liste abgearbeitet wird. Anderenfalls ist beim Rückgabewert Bit #03 gesetzt (<code>NotAllowedInExecuting</code>). Ansonsten (und dass <code>slsc_ctrl_enable_laser</code> einen Rückgabewert hat) entspricht das Verhalten von <code>slsc_ctrl_enable_laser</code> dem des RTC6-Kontrollbefehls <code>enable_laser</code> (siehe RTC6-Handbuch zu freigegebenen Ports, Port-Werten etc). • ⚠ Vorsicht! Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_cfg_initialize_from_file , slsc_ctrl_disable_laser

Name der Funktion	<code>slsc_ctrl_follow</code>
Zweck	Zum Wieder-Übernehmen des Verfahrtschis nach einem <code>slsc_ctrl_unfollow</code> .
Funktions-Signatur	<code>uint32_t slsc_ctrl_follow(size_t Handle);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • <code>slsc_ctrl_follow</code> beendet auch automatisch den Modus "Manuelle Positionierung" der angegebenen <code>syncAXIS control</code>-Instanz, siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70. • <code>slsc_ctrl_follow</code> ist ähnlich zu <code>slsc_cfg_acquire_stage</code> (veraltet), aber schneller. Außerdem wird die <code>syncAXIS</code>-DLL-interne Job-Planung nicht unterbrochen und bereits berechnete Jobs gehen auch nicht verloren. • Die komplementäre Funktion von <code>slsc_ctrl_follow</code> ist <code>slsc_ctrl_unfollow</code>. • <code>slsc_ctrl_follow</code> ist nur nach einem <code>slsc_ctrl_unfollow</code> erlaubt. • <code>slsc_ctrl_follow</code> ermöglicht den Wechsel von Verfahrtschis im Zusammenspiel mit <code>slsc_cfg_select_stage</code>. Beachten Sie, dass mit <code>syncAXIS-DLL ≥ V1.0.7</code> dabei die aktuell verwendete Korrekturdatei <i>nicht</i> ausgetauscht wird. • Wenn sich mindestens ein Job in der Job-Queue befindet, dann wird geprüft ob die darin geplanten Verfahrtschisposition für den Job-Anfang und die aktuelle Verfahrtschisposition übereinstimmen. Bei Nichtübereinstimmung (Ursache: wegen <code>slsc_list_begin_absolute</code> oder der Verfahrtschis wurde manuell positioniert) ist beim Rückgabewert das Bit #12 gesetzt (<code>InvalidPosition</code>; Der Verfahrtschis muss dann an die korrekte Position verfahren werden, siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70). • Wenn sich <i>kein</i> Job in der Job-Queue befindet, dann wird für die Trajektorienplanung des nächsten Jobs die aktuelle Verfahrtschisposition verwendet. • Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von <code>syncAXIS control</code>-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab <code>syncAXIS-DLL V1.0.7</code> .
Verweise	<code>slsc_ctrl_unfollow</code>

Name der Funktion	<code>slsc_ctrl_get_error</code>
Zweck	Gibt Informationen zu einem aufgetretenen Fehler (Fehlernummer <code>ErrorNr</code>) zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_error(size_t Handle, size_t ErrorNr, uint64_t* ErrorCode, char* ErrorMessage, size_t MsgBufSize);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	ErrorNr Fehlernummer, für die die Informationen abgefragt werden soll. Erlaubter Bereich: 0...[(<code>ErrorCount</code> aus <code>slsc_ctrl_get_error_count</code>) – 1].
	ErrorCode Fehlercode. Parameterrückgabe: Zeiger. Siehe Kapitel 5 "Fehlercodes (Error Codes) bei <code>slsc_ctrl_get_error</code>, Log-Datei und Konsole" , Seite 282.
	ErrorMessage Parameterrückgabe: Zeiger auf ein Char-Array. Das Char-Array muss durch den Benutzer vorher allokiert worden sein. Reicht zusätzlichen Text zum Fehler durch, sofern vorhanden.
	MsgBufSize Zeichenanzahl des für <code>ErrorMessage</code> allokierten Char-Arrays.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Um einen sinnvollen Wert für <code>ErrorNr</code> angeben zu können, muss zunächst die Anzahl der vorliegenden Fehler festgestellt werden. Deswegen muss erst <code>slsc_ctrl_get_error_count</code> und dann <code>slsc_ctrl_get_error</code> aufgerufen werden. Der erste aufgetretene Fehler ist der älteste detektierte Fehler. Er hat die Nummer 0. <code>slsc_ctrl_get_error</code> wirft eine Exception bei: <ul style="list-style-type: none"> (<code>ErrorCount</code> aus <code>slsc_ctrl_get_error_count</code>) = 0 (d.h. es liegt gar kein Fehler vor) <code>ErrorNr</code> > [(<code>ErrorCount</code> aus <code>slsc_ctrl_get_error_count</code>) – 1] Die Textnachricht wird geclippt auf die Größe von <code>MsgBufSize</code> (d.h. wenn die Textnachricht länger ist als die <code>MsgBufSize</code>, dann enthält <code>ErrorMessage</code> nur so viele Zeichen wie in <code>MsgBufSize</code> definiert sind). Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0. Letzte Änderung mit syncAXIS-DLL V1.1.0: Datentyp von <code>ErrorNr</code> .
Verweise	<code>slsc_ctrl_get_error_count</code>

Name der Funktion	<code>slsc_ctrl_get_error_count</code>
Zweck	Gibt die Anzahl der vorliegenden Fehler zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_error_count(size_t Handle, size_t* ErrorCount);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	ErrorCount Parameterrückgabe: Zeiger. Anzahl der vorliegenden Fehler.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Um einen sinnvollen Wert für ErrorNr (bei slsc_ctrl_get_error) angeben zu können, muss zunächst die Anzahl der vorliegenden Fehler festgestellt werden. Deswegen muss erst slsc_ctrl_get_error_count und dann slsc_ctrl_get_error aufgerufen werden. • Der erste aufgetretene Fehler ist der älteste detektierte Fehler. Er hat die Nummer 0. • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_ctrl_get_error

Name der Funktion	<code>slsc_ctrl_get_exec_state</code>
Zweck	Gibt den Status des Execution Layer zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_exec_state(size_t Handle, slsc_ExecState* State);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	State Parameterrückgabe: Zeiger. Siehe enum slsc_ExecState .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Der Ausführungsstatus der RTC6-Karte (siehe enum slsc_ExecState: <code>slsc_ExecState_Idle</code>, <code>slsc_ExecState_ReadyForExecution</code>, <code>slsc_ExecState_Executing</code>, <code>slsc_ExecState_NotInitOrError</code>) gibt <i>keine</i> Auskunft darüber, ob momentan weitere Job-Funktionen (<code>slsc_list_*</code>) abgesetzt werden können (hierfür siehe slsc_ctrl_is_list_input_buffer_full). Diese beiden Eigenschaften sind <i>nicht</i> voneinander abhängig. Beispielsweise kann mit slsc_ctrl_get_exec_state festgestellt werden, ob <ul style="list-style-type: none"> eine Ausführung gestartet werden kann (Dazu muss der Ausführungsstatus <code>slsc_ExecState_ReadyForExecution</code> sein) die RTC6-Karte wieder (z. B. am Ende des Anwenderprogramms) freigegeben werden kann (Dazu darf der Ausführungsstatus nicht <code>slsc_ExecState_Executing</code> sein) der Betriebsmodus der syncAXIS control-Instanz gewechselt werden kann (Dazu darf der Ausführungsstatus nicht <code>slsc_ExecState_Executing</code> sein) Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. slsc_ExecState State; // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_ctrl_get_exec_state(Handle, &State)</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_ctrl_is_list_input_buffer_full

Name der Funktion	<code>slsc_ctrl_get_free_variable</code>
Zweck	Gibt den aktuellen Wert einer freien Variable der RTC6 zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_free_variable(size_t Handle, uint32_t Number, uint32_t* Value);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Number Nummer derjenigen freien Variablen, deren Wert auf der RTC6 gelesen werden soll. Zulässiger Wertebereich: [0...7]. Es werden nur die drei niederwertigsten Bits ausgewertet.
	Value Wert der freien Variable, der aktuell auf der RTC6 gesetzt ist. Parameterrückgabe: Zeiger.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Im Simulationsmodus haben slsc_ctrl_get_free_variable, slsc_ctrl_set_free_variable, sowie slsc_list_set_free_variable keine Auswirkung. Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. slsc_ctrl_get_free_variable ist eine direkte Implementierung des RTC6-Befehls get_free_variable in syncAXIS control. get_free_variable bietet das Argument Value jedoch nicht. Mit den Funktionen für freie Variablen (slsc_ctrl_set_free_variable, slsc_list_set_free_variable und slsc_ctrl_get_free_variable) können z.B. Inkremente (innerhalb von Jobs) festgestellt und gezählt werden. Weitere Informationen zu freien Variablen finden Sie im RTC6-Handbuch, Kapitel 6.9.1 "Freie Variablen", Seite 134. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.2 .
Verweise	slsc_ctrl_set_free_variable , slsc_list_set_free_variable

Name der Funktion	<code>slsc_ctrl_get_job_characteristic</code>
Zweck	Gibt – für eine angegebene Job-ID – den in der Trajektorienplanung berechneten Wert eines Job-Merkmals ("Key", siehe enum slsc_JobCharacteristic) zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_job_characteristic(size_t Handle, size_t JobID, slsc_JobCharacteristic Key, double* Value);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	JobID Job-ID . Wird von slsc_list_begin* zurückgegeben.
	Key Siehe enum slsc_JobCharacteristic .
	Value Parameterrückgabe: Zeiger.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_ctrl_get_job_characteristic setzt voraus, dass der Status der angegebenen Job-ID mindestens "Berechnung: Beendet" (siehe Abbildung 12, Seite 43) ist. Anderenfalls ist beim Rückgabewert Bit #06 gesetzt (UnplausibleOrUnknownParameter). • slsc_ctrl_get_job_characteristic funktioniert auch ohne aktivierte Dateiausgabe (DisableFileOutput). • Mit slsc_ctrl_get_job_characteristic können die letzten 10 berechneten Job-IDs abgefragt werden. • slsc_ctrl_get_job_characteristic ist vorgesehen, um über Algorithmen die Auswirkungen von Parameterpermutationen (im Simulationsmodus) auszuwerten (d.h. die Parameterwert-Optimierung automatisieren). • Nur bei kurzen Job-IDs (nicht genauer quantifizierbar; wegen der im ersten Listenelement genannten Voraussetzung) kann slsc_ctrl_get_job_characteristic auch genutzt werden, um Sicherheitsabfragen in einer GUI zu implementieren, d.h. wenn Trajektorienplanungs-Berechnungsergebnisse (z.B. maximale Ansteuerwerte für den Verfahrtsch) außerhalb der definierten Grenzen liegen (z.B. "...do you really want to execute Job..."). • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.

Name der Funktion	slsc_ctrl_get_job_characteristic
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. // What is StagePosX, StagePosY in the Job size_t Handle; // from above somewhere double Value0 = 0; // init variable for 1st key double Value1 = 0; // init variable for 2nd key size_t JobID = ; // received by list_begin() slsc_ctrl_get_job_characteristic(Handle, JobID, slsc_JobCharacteristic:: slsc_JobCharacteristic_StagePosX, &Value0); slsc_ctrl_get_job_characteristic(Handle, JobID, slsc_JobCharacteristic::slsc_JobCharacteristic_StagePosY, &Value1); // Value0 contains now the max. X position calculated for the Stage // Value1 contains now the max. Y position calculated for the Stage</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	–

Name der Funktion	slsc_ctrl_get_scan_device_position
Zweck	Gibt die Soll-Position oder Ist-Position des angegebenen Scan-Device (Scan-Kopf) zurück.
Funktions-Signatur	<pre>uint32_t slsc_ctrl_get_scan_device_position(size_t Handle, slsc_ScanDevice ScanDevice, slsc_PositionType Type, double* Position);</pre>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	ScanDevice Siehe enum <code>slsc_ScanDevice</code>.
	Type Siehe enum <code>slsc_PositionType</code>.
	Position Parameterrückgabe: Zeiger auf ein Array der Dimension 2.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.0.
Verweise	slsc_ctrl_get_stage_position

Name der Funktion	slsc_ctrl_get_simulation_filename
Zweck	<p>Veraltet.</p> <p>Zum Abfragen von Simulationsdateinamen mit der syncAXIS-DLL V1.2. Ab der syncAXIS-DLL \geq V1.3.0 muss slsc_ctrl_get_syncaxis_simulation_filename verwendet werden!</p>
Funktions-Signatur	<pre>uint32_t slsc_ctrl_get_simulation_filename(size_t Handle, size_t JobID, slsc_ScanDevice ScanDevice, char* SimulationFileName, size_t FileNameBufSize);</pre>
Argument(e)	<p>Handle Handle auf eine syncAXIS control-Instanz.</p>
	<p>JobID Job-ID. Wird von slsc_list_begin* zurückgegeben.</p>
	<p>ScanDevice Siehe enum slsc_ScanDevice. Das angegebene Scan-Device muss in der syncAXISConfig.xml eingetragen sein.</p>
	<p>SimulationFileName Parameterrückgabe: Zeiger auf ein Char-Array. Das Char-Array muss durch den Benutzer vorher allokiert worden sein. Reicht den Simulationsdateinamen durch.</p>
	<p>FileNameBufSize Zeichenanzahl des für SimulationFileName allokierten Char-Arrays. Muss mindestens 49 für einstellige Job-IDs sein.</p>
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_ctrl_get_simulation_filename ist nur mit syncAXIS-DLL V1.2 sinnvoll nutzbar weil noch für jedes einzelne Scan-Device eine eigene Simulationsdatei erzeugt wird. Dateinamenskonvention in V1.2: "Simulation_ID_<Job-ID>_<Scan-Device>_TS_<13 Ziffern>.txt" [TS = Time Stamp; * _<Scan-Device>* gibt es in \leq V1.1 nicht]. Beispiel: Simulation_ID_1_ScanDevice2_TS_1546938743472.txt. Mit syncAXIS-DLL \geq V1.3.0 wird nur noch eine einzige Simulationsdatei erzeugt, das die Daten der einzelnen Scan-Devices enthält.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.2...V1.2.6.
Verweise	slsc_ctrl_get_syncaxis_simulation_filename

Name der Funktion	<code>slsc_ctrl_get_stage_position</code>
Zweck	Gibt die Soll-Position oder Ist-Position des Verfahrtschis zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_stage_position(size_t Handle, slsc_PositionType Type, double* Position);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
	Type Siehe enum <code>slsc_PositionType</code> .
	Position Parameterrückgabe: Zeiger auf ein Array der Dimension 2.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der <code>syncAXIS-DLL</code> Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von <code>syncAXIS control</code>-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab <code>syncAXIS-DLL V1.2.0</code> .
Verweise	<code>slsc_ctrl_get_scan_device_position</code>

Name der Funktion	<code>slsc_ctrl_get_syncaxis_simulation_filename</code>
Zweck	Nur im Simulationsmodus! Gibt für eine angegebene Job-ID den entsprechenden Simulationsdateinamen zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_syncaxis_simulation_filename(size_t Handle, size_t JobID, char* SimulationFileName, size_t FileNameBufSize);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	JobID Job-ID . Wird von slsc_list_begin* zurückgegeben.
	SimulationFileName Parameterrückgabe: Zeiger auf ein Char-Array. Das Char-Array muss durch den Benutzer vorher allokiert worden sein. Reicht den Simulationsdateinamen durch.
	FileNameBufSize Zeichenanzahl des für SimulationFileName allokierten Char-Arrays. Muss mindestens 37 für einstellige Job-IDs sein.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> • slsc_ctrl_get_syncaxis_simulation_filename ersetzt slsc_ctrl_get_simulation_filename. • slsc_ctrl_get_syncaxis_simulation_filename setzt voraus, dass der Simulationsmodus aktiv ist. Anderenfalls ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). • slsc_ctrl_get_syncaxis_simulation_filename setzt voraus, dass der Status der angegebenen Job-ID mindestens "Berechnung: Beendet" (siehe Abbildung 12, Seite 43) ist. Anderenfalls ist beim Rückgabewert Bit #06 gesetzt (UnplausibleOrUnknownParameter). • slsc_ctrl_get_syncaxis_simulation_filename setzt voraus, dass in der syncAXISConfig.xml die Dateiausgabe eingestellt ist: <code><cfg:DisableFileOutput>false</cfg:DisableFileOutput></code>. Andernfalls ist beim Rückgabewert Bit #06 gesetzt (UnplausibleOrUnknownParameter). • In syncAXIS control ≥ V1.3 gilt für Simulationsdateinamen die folgende Konvention: "Simulation_ID_<JobID>_TS_<13 Ziffern>.txt" [TS = Time Stamp]. Beispiel: <code>Simulation_ID_1_TS_1546938743472.txt</code>. • Mit slsc_ctrl_get_syncaxis_simulation_filename können die letzten 10 berechneten Job-IDs abgefragt werden. • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0 .
Verweise	slsc_ctrl_get_simulation_filename

Name der Funktion	slsc_ctrl_get_value
Zweck	Gibt den aktuellen Wert des angegebenen Signals der angegebenen Achse zurück.
Funktions-Signatur	<code>uint32_t slsc_ctrl_get_value(size_t Handle, size_t AxisIndex, slsc_MeasurementSignal Signal, double* Value);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	AxisIndex 0: Scan-Kopf-Achse X 1: Scan-Kopf-Achse Y 2: Verfahrtsch-Achse X 3: Verfahrtsch-Achse Y
	Signal Siehe enum slsc_MeasurementSignal .
	Value Parameterrückgabe: Zeiger.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Verwenden Sie slsc_ctrl_get_value nicht zum Abfragen von Soll-Positionen und Ist-Positionen. Für diesen Zweck steht (ab syncAXIS-DLL V1.2.0) slsc_ctrl_get_scan_device_position sowie slsc_ctrl_get_stage_position zur Verfügung. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0. Letzte Änderung mit syncAXIS-DLL V1.1.0: Datentyp von AxisIndex, Signal.
Verweise	slsc_ctrl_get_scan_device_position, slsc_ctrl_get_stage_position

Name der Funktion	<code>slsc_ctrl_is_list_input_buffer_full</code>
Zweck	Prüft, ob der syncAXIS-DLL-Eingangspuffer voll ist (und deshalb momentan keine weitere Job-Funktion (<code>slsc_list_*</code>) mehr aufnehmen kann).
Funktions-Signatur	<code>uint32_t slsc_ctrl_is_list_input_buffer_full(size_t Handle, bool* Flag);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	Flag Parameterrückgabe: Zeiger. true: Der Eingangspuffer ist voll. false: Der Eingangspuffer ist nicht voll.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Der Eingangspuffer hat eine begrenzte Aufnahmekapazität (nicht direkt quantifizierbar) für Job-Funktionen (<code>slsc_list_*</code>). Vor dem Abschicken einer Job-Funktion (<code>slsc_list_*</code>) kann mit <code>slsc_ctrl_is_list_input_buffer_full</code> geprüft werden, ob der Eingangspuffer aufnahmebereit ist (Flag false). Wird eine Job-Funktion (<code>slsc_list_*</code>) abgesetzt und der Eingangspuffer ist voll, ist bei deren Rückgabewert Bit #04 gesetzt (BUFFER_FULL). Wenn der Eingangspuffer voll ist, dann wird eine abgeschickte Job-Funktion (<code>slsc_list_*</code>) nicht ausgeführt. Dabei wird keine Exception geworfen! Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	–

Name der Funktion	<code>slsc_ctrl_laser_signal_off</code>
Zweck	Nur im Modus "Manuelle Positionierung": Schaltet den Laser unverzüglich aus.
Funktions-Signatur	<code>uint32_t slsc_ctrl_laser_signal_off(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_ctrl_laser_signal_off ist für eine direkte Lasersteuerung in Kombination mit slsc_ctrl_laser_signal_on vorgesehen. • slsc_ctrl_laser_signal_off wird nur im Modus "Manuelle Positionierung" akzeptiert. Sonst ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). • Siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70. • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	slsc_ctrl_laser_signal_on

Name der Funktion	<code>slsc_ctrl_laser_signal_on</code>
Zweck	Nur im Modus "Manuelle Positionierung": Schaltet den Laser unverzüglich an.
Funktions-Signatur	<code>uint32_t slsc_ctrl_laser_signal_on(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_ctrl_laser_signal_on ist für eine direkte Lasersteuerung in Kombination mit slsc_ctrl_laser_signal_off vorgesehen, z. B. bei Justagen. • Der Laser wird auch ausgeschaltet, wenn: <ul style="list-style-type: none"> – die syncAXIS control-Instanz abgebaut wird – der Betriebsmodus gewechselt wird • slsc_ctrl_laser_signal_on wird nur im Modus "Manuelle Positionierung" akzeptiert. Sonst ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). • ⚠ Vorsicht! Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! • Siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70. • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	slsc_ctrl_laser_signal_off

Name der Funktion	<code>slsc_ctrl_move_scanner_abs</code>
Zweck	Nur im Modus "Manuelle Positionierung": Bringt alle Scan-Devices (ausgehend von der aktuellen Position) mit Sprunggeschwindigkeit in die angegebene Position.
Funktions-Signatur	<code>uint32_t slsc_ctrl_move_scanner_abs(size_t Handle, const double* Position);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Position Zeiger auf ein Array der Dimension 2. Zielposition in absoluten (nicht relativen) Koordinaten. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • <code>slsc_ctrl_move_scanner_abs</code> wird nur im Modus "Manuelle Positionierung" akzeptiert. Sonst ist beim Rückgabewert Bit #11 gesetzt (<code>NotAllowedInCurrentMode</code>). • <code>slsc_ctrl_move_stage_abs</code> und <code>slsc_ctrl_move_scanner_abs</code> schalten vor Beginn der Bewegung den Laser ("aktiv") aus. Dieses Sicherheitsfeature stellt sicher, dass der Laser nicht unkontrolliert an sein kann. • Mit <code>slsc_ctrl_move_scanner_abs</code> und <code>slsc_ctrl_move_stage_abs</code> (die ähnlich zum RTC6-Befehl <code>goto_xy</code> sind) ist es möglich, die Galvanometerscanner im Scan-Kopf und den Verfahrtschisch voneinander unabhängig zu bewegen. • <code>slsc_ctrl_move_scanner_abs</code> verfährt alle Scan-Devices an die angegebene Position, auch in Multi-Head-Systemen. • Siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70. • Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	<code>slsc_ctrl_move_stage_abs</code>

Name der Funktion	<code>slsc_ctrl_move_stage_abs</code>
Zweck	Nur im Modus "Manuelle Positionierung": Bringt den Verfahrtsch (ausgehend von der aktuellen Position) mit den über die ACS-API eingestellten Dynamiken (Beschleunigung und Ruck) in die angegebene Position.
Funktions-Signatur	<code>uint32_t slsc_ctrl_move_stage_abs(size_t Handle, const double* Position, double Speed, double Timeout);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Position Zeiger auf ein Array der Dimension 2. Zielposition in absoluten (nicht relativen) Koordinaten. In mm.
	Speed Maximale Verfahrtschgeschwindigkeit. In mm/s.
	Timeout Zeit, nach der diese Funktion spätestens zurückgekehrt sein sollte. In s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_ctrl_move_stage_abs wird nur im Modus "Manuelle Positionierung" akzeptiert. Sonst ist beim Rückgabewert Bit #11 gesetzt (NotAllowedInCurrentMode). • slsc_ctrl_move_stage_abs und slsc_ctrl_move_scanner_abs schalten vor Beginn der Bewegung den Laser ("aktiv") aus. Dieses Sicherheitsfeature stellt sicher, dass der Laser nicht unkontrolliert an sein kann. • ⚠ Vorsicht! Von einem bewegten Verfahrtsch geht eine mechanische Gefahr aus. Es besteht Quetschgefahr für die Finger und Hände. Achten Sie darauf, dass alle umstehenden Personen während der Ausführung ausreichend Abstand zu dem Gerät halten. • Mit slsc_ctrl_move_scanner_abs und slsc_ctrl_move_stage_abs (die ähnlich zum RTC6-Befehl goto_xy sind) ist es möglich, die Galvanometerscanner im Scan-Kopf und den Verfahrtsch voneinander unabhängig zu bewegen. • Sendet einen ACS-Point-to-Point-Motion-Befehl für den ausgewählten Verfahrtsch (siehe slsc_cfg_select_stage) an den ACS Motion-Controller via TCP/IP. • Siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70. • ⚠ Vorsicht! Stellen Sie sicher, dass der angegebene Speed-Wert im zulässigen Bereich ist, siehe auch Kapitel 2.2 "Über die SICHERE Verwendung von syncAXIS control – Allgemeines Vorgehen", Seite 18. • Der Timeout-Wert legt fest, nach welcher Zeit slsc_ctrl_move_stage_abs spätestens zurückgekehrt sein soll (typischer Timeout-Wert: mehrere Sekunden). Nach Ablauf dieser Zeit ist beim Rückgabewert dann Bit #13 gesetzt (Timeout). • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	slsc_ctrl_move_scanner_abs

Name der Funktion	<code>slsc_ctrl_refresh_correction_file</code>
Zweck	Überträgt unverzüglich eine Korrekturdatei an die RTC6-Karte.
Funktions-Signatur	<code>uint32_t slsc_ctrl_refresh_correction_file(size_t Handle, uint32_t CorrectionFileIndex);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	CorrectionFileIndex Index der Korrekturdatei, die an die RTC6-Karte übertragen werden soll (Korrekturdateien sind in der <code>syncAXISConfig.xml</code> angegeben; siehe auch Abschnitt "Korrekturdatei-bezogene Funktionen" , Seite 99). Erlaubte Werte: 0...3.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Voraussetzung für <code>slsc_ctrl_refresh_correction_file</code> (genauso wie <code>slsc_ctrl_select_correction_file</code>) ist ein Ausführungsstatus <code>slsc_ExecState_Idle = 1</code> oder <code>slsc_ExecState_ReadyForExecution = 1</code> (Abfrage mit <code>slsc_ctrl_get_exec_state</code>). <code>slsc_ctrl_select_correction_file</code> wählt eine auf der RTC6-Karte bereits vorhandene Korrekturdatei aus. Im Gegensatz dazu überträgt <code>slsc_ctrl_refresh_correction_file</code> unverzüglich eine Korrekturdatei auf die RTC6-Karte (ohne die syncAXIS control-Instanz abbauen/neu aufbauen zu müssen). Siehe auch Abschnitt "Korrekturdatei-bezogene Funktionen", Seite 99. Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	<code>slsc_ctrl_select_correction_file</code>

Name der Funktion	<code>slsc_ctrl_select_correction_file</code>
Zweck	Zum Angeben einer Korrekturdatei, die unverzüglich verwendet werden soll.
Funktions-Signatur	<code>uint32_t slsc_ctrl_select_correction_file(size_t Handle, uint32_t CorrectionFileIndex);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	CorrectionFileIndex Index der Korrekturdatei, die verwendet werden soll (Korrekturdateien sind in der <code>syncAXISConfig.xml</code> angegeben; siehe auch Abschnitt "Korrekturdatei-bezogene Funktionen" , Seite 99). Erlaubte Werte: 0...3.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Voraussetzung für <code>slsc_ctrl_select_correction_file</code> ist ein Ausführungsstatus <code>slsc_ExecState_Idle = 1</code> oder <code>slsc_ExecState_ReadyForExecution = 1</code> (Abfrage mit <code>slsc_ctrl_get_exec_state</code>). Siehe auch Abschnitt "Korrekturdatei-bezogene Funktionen", Seite 99. See also XML-Tag <code>CorrectionFilePath</code>. Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	–

Name der Funktion	<code>slsc_ctrl_set_free_variable</code>
Zweck	Setzt auf der RTC6 den Wert einer freien Variable.
Funktions-Signatur	<code>uint32_t slsc_ctrl_set_free_variable(size_t Handle, uint32_t Number, uint32_t Value);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Number Nummer derjenigen freien Variablen, deren Wert auf der RTC6 gesetzt werden soll. Zulässiger Wertebereich: [0...7]. Es werden nur die drei niederwertigsten Bits ausgewertet.
	Value Wert der freien Variable, der auf der RTC6 gesetzt werden soll.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Im Simulationsmodus haben slsc_ctrl_get_free_variable, slsc_ctrl_set_free_variable, sowie slsc_list_set_free_variable keine Auswirkung. Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. slsc_ctrl_set_free_variable ist eine direkte Implementierung des RTC6-Befehls set_free_variable in syncAXIS control. Mit den Funktionen für freie Variablen (slsc_ctrl_set_free_variable, slsc_list_set_free_variable und slsc_ctrl_get_free_variable) können z.B. Inkremente (innerhalb von Jobs) festgestellt und gezählt werden. Weitere Informationen zu freien Variablen finden Sie im RTC6-Handbuch, Kapitel 6.9.1 "Freie Variablen", Seite 134. Die entsprechende Job-Funktion (slsc_list_*) von slsc_ctrl_set_free_variable ist slsc_list_set_free_variable. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.2.
Verweise	slsc_ctrl_get_free_variable , slsc_list_set_free_variable

Name der Funktion	<code>slsc_ctrl_set_laser_pulses</code>
Zweck	Definiert die Ausgabeperiode und die Pulslänge der Lasersignale LASER1 und LASER2 für den "Laser active"-Betrieb der RTC6-Karte.
Funktions-Signatur	<code>uint32_t slsc_ctrl_set_laser_pulses(size_t Handle, double HalfPeriod, double PulseLength);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	HalfPeriod Halbe Ausgabeperiode. In s. Zulässiger Wertebereich: [0...67].
	PulseLength Pulslänge der Lasersignale LASER1 und LASER2. In s. Zulässiger Wertebereich: [0...67].
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Im Simulationsmodus haben <code>slsc_ctrl_set_laser_pulses</code> und <code>slsc_list_set_laser_pulses</code> keine Auswirkung. Wenn <code>HalfPeriod</code> und/oder <code>PulseLength</code> den Wert 0 haben, werden keine Lasersignale ausgegeben. Negative Werte werden zurückgewiesen. Beim Rückgabewert ist dann Bit #06 gesetzt (<code>UnplausibleOrUnknownParameter</code>). Bei $PulseLength \geq 2 \times HalfPeriod$ bleibt der Laser dauernd an. Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. <code>slsc_ctrl_set_laser_pulses</code> ähnelt dem RTC6-Befehl <code>set_laser_pulses_ctrl</code>. Allerdings wird die <i>syncAXIS control</i>-Funktion <code>slsc_ctrl_set_laser_pulses</code> nicht immer akzeptiert (z.B. während ein Job ausgeführt wird). <code>slsc_ctrl_set_laser_pulses</code> und <code>slsc_list_set_laser_pulses</code> werden für diejenigen Benutzer bereitgestellt, die (aufgrund ihres eingesetzten Lasers) die "Automatische Lasersteuerung" zum Erreichen äquidistanter Spotabstände nicht nutzen können und stattdessen die Pulsausgabe über <code>HalfPeriod</code> und <code>PulseLength</code> beeinflussen wollen. <code>HalfPeriod</code> und <code>PulseLength</code> ändern, was bei der Initialisierung (mit den gleichnamigen Attributen im <code>syncAXISConfig.xml</code>-Tag <code><cfg:LaserOutput Unit="s" HalfPeriod="..." PulseLength="..." /></code> gesetzt wurde. Ist die "Automatische Lasersteuerung" aktiv mit <code>SpotDistance</code> als "ActiveChannel", siehe Kapitel 2.9.2 "Definition der Channel und ActiveChannel", Seite 48, dann: <ul style="list-style-type: none"> wirkt <code>HalfPeriod</code> nicht wirkt <code>PulseLength</code> (d.h. die Pulslängen von Lasersignal LASER1 und LASER2 werden geändert) Die entsprechende Job-Funktion (<code>slsc_list_*</code>) von <code>slsc_ctrl_set_laser_pulses</code> ist <code>slsc_list_set_laser_pulses</code>. Zur Zulässigkeit von <i>syncAXIS control</i>-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung" ", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.4.
Verweise	<code>slsc_list_set_laser_pulses</code>

Name der Funktion	<code>slsc_ctrl_start_execution</code>
Zweck	Versucht, die Ausführung eines Jobs durch das Execution Layer zu starten.
Funktions-Signatur	<code>uint32_t slsc_ctrl_start_execution(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • Voraussetzung für slsc_ctrl_start_execution: der Ausführungsstatus muss <code>slsc_ExecState_ReadyForExecution</code> sein (Abfrage mit slsc_ctrl_get_exec_state). • slsc_ctrl_start_execution betrifft nur den ältesten Job in der Job-Queue, der noch nicht ausgeführt wurde und mit Status "Transfer: In Bearbeitung (Enough loaded = ja)". Siehe Abbildung 13, Seite 44. • Es wird sofort(!) versucht, die Ausführung zu starten. Allerdings kann die Zeit bis zum tatsächlichen Ausführungsstart nicht quantifiziert werden. • ⚠ Vorsicht! Achten Sie darauf, dass die Lasersicherheit im gesamten System gewährleistet ist! • ⚠ Vorsicht! Von einem bewegten Verfahrtsch geht eine mechanische Gefahr aus. Es besteht Quetschgefahr für die Finger und Hände. Achten Sie darauf, dass alle umstehenden Personen während der Ausführung ausreichend Abstand zu dem Gerät halten. • Betrifft nur <code>MasterSlaveSynchronizer.exe</code>-Benutzer, siehe auch Handbuch "syncAXIS Master-Slave-Synchronizer": Sollten – durch den Aufruf der Option <code>-ConnectExternalStartStop</code> – mehrere RTC6-Karten für einen synchronen Listenstart konfiguriert worden sein, dann wird der durch slsc_ctrl_start_execution ausgelöste Listenausführungsstart auch auf die syncAXIS control-Instanzen auf den anderen RTC6-Karten weitergereicht. • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_ctrl_stop , slsc_ctrl_stop_controlled

Name der Funktion	slsc_ctrl_stop
Zweck	Beendet die Ausführung des aktuellen Jobs unkontrolliert und sofort über einen direkten Zugriff auf die RTC6-Karte ("Not-Halt").
Funktions-Signatur	<code>slsc_ctrl_stop(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_ctrl_stop ist nur als "Not-Halt" gedacht. Hardware-freundlicher ist slsc_ctrl_stop_controlled. • Siehe Abschnitt "Gegenüberstellung von slsc_ctrl_stop_controlled und slsc_ctrl_stop", Seite 97. • Der Betriebsstatus der syncAXIS control-Instanz wechselt sofort auf "rot". Die syncAXIS control-Instanz muss neu initialisiert werden. • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_ctrl_start_execution, slsc_ctrl_stop_controlled

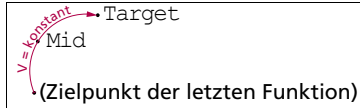
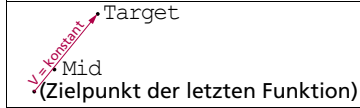
Name der Funktion	slsc_ctrl_stop_controlled
Zweck	Beendet die Ausführung des aktuellen Jobs kontrolliert durch Einfügen einer Ausgleichsbewegung zum Abbremsen.
Funktions-Signatur	<code>slsc_ctrl_stop_controlled(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Nach dem slsc_ctrl_stop_controlled-Aufruf arbeitet die RTC6-Karte bereits geladene RTC6-Mikrovektorbefehle in ihrem Listenspeicher ab. Bis zum tatsächlichem Job-Ausführungsende können daher noch bis zu 30 s vergehen. Anschließend wird die Ausgleichsbewegung zum Abbremsen ausgeführt. slsc_ctrl_get_exec_state gibt in dieser gesamten Zeit "slsc_ExecState_Executing" zurück. Nach der Ausgleichsbewegung wechselt der Betriebsstatus der syncAXIS control-Instanz auf "rot". Die syncAXIS control-Instanz muss neu initialisiert werden. Siehe Abschnitt "Gegenüberstellung von slsc_ctrl_stop_controlled und slsc_ctrl_stop", Seite 97. Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_ctrl_start_execution, slsc_ctrl_stop

Name der Funktion	slsc_ctrl_unfollow
Zweck	Die angegebene syncAXIS control-Instanz gibt vorübergehend den Verfahrtsch frei. Dieser kann nun extern (z.B. durch ein nicht-syncAXIS control-basiertes Anwenderprogramm) gesteuert werden.
Funktions-Signatur	<code>uint32_t slsc_ctrl_unfollow(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_ctrl_unfollow versetzt die angegebene syncAXIS control-Instanz automatisch in den Modus "Manuelle Positionierung", siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70. • slsc_ctrl_unfollow ist ähnlich zu slsc_cfg_release_stage (veraltet), aber schneller. Außerdem wird die syncAXIS-DLL-interne Job-Planung nicht unterbrochen und bereits berechnete Jobs gehen auch nicht verloren. • Die komplementäre Funktion von slsc_ctrl_unfollow ist slsc_ctrl_follow. • slsc_ctrl_unfollow muss einem slsc_ctrl_follow vorausgehen. • Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.0.7.
Verweise	slsc_ctrl_follow

Name der Funktion	<code>slsc_ctrl_write_analog_x</code>
Zweck	Nur im Modus "Manuelle Positionierung"! Schreibt einen Ausgabewert auf den 12-Bit-Analogausgang ANALOG OUT1 oder ANALOG OUT2 aller RTC6-Karten.
Funktions-Signatur	<code>uint32_t slsc_ctrl_write_analog_x(size_t Handle, slsc_AnalogOutput Channel, double Value);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Channel Analogausgang ANALOG OUT1 oder ANALOG OUT2 ("Kanal"). = 1: ANALOG OUT1. = 2: ANALOG OUT2. Zulässiger Wertebereich: [1, 2]. Siehe enum slsc_AnalogOutput .
	Value Ausgabewert am Analogausgang ANALOG OUT1 oder ANALOG OUT2. Value = 0 entspricht einem Ausgabewert von 0 V. Value = 1 entspricht einem Ausgabewert von 10 V.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Das ANALOG OUT1-Signal wird ausgegeben bei <ul style="list-style-type: none"> RTC6 PCI-Express-Karten (wie RTC5-Karten): LASER-Buchse, Pin 08 Das ANALOG OUT2-Signal wird ausgegeben bei <ul style="list-style-type: none"> RTC6 PCI-Express-Karten (wie RTC5-Karten): LASER-Buchse, Pin 15, sowie MARKING ON THE FLY-Stiftleiste, Pin 14 slsc_ctrl_write_analog_x wird nicht akzeptiert, wenn gerade ein Job ausgeführt wird. Dann ist beim Rückgabewert Bit #03 gesetzt (NotAllowedInExecuting). Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. Verwandter RTC6-Befehl: write_da_x. Die entsprechende Job-Funktion (slsc_list_*) von slsc_ctrl_write_analog_x ist slsc_list_write_analog_x. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.0.
Verweise	slsc_list_write_analog_x

Name der Funktion	<code>slsc_ctrl_write_digital_out</code>
Zweck	Nur im Modus "Manuelle Positionierung"! Schreibt einen 16-Bit-Ausgabewert auf den 16-Bit-Digital-Ausgang DIGITAL OUT 0...DIGITAL OUT 15 aller RTC6-Karten.
Funktions-Signatur	<code>uint32_t slsc_ctrl_write_digital_out(size_t Handle, uint16_t Value);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Value 16-Bit-Ausgabewert (DIGITAL OUT 0...DIGITAL OUT 15) am 16-Bit-Digital-Ausgang.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Das DIGITAL OUT 0...DIGITAL OUT 15-Signal wird ausgegeben bei <ul style="list-style-type: none"> RTC6 PCI-Express-Karten (wie RTC5-Karten): EXTENSION 1-Stiftleiste, Pin 01...Pin 31 (nur ungeradzahlige Pins) slsc_ctrl_write_digital_out wird nicht akzeptiert, wenn gerade ein Job ausgeführt wird. Dann ist beim Rückgabewert Bit #03 gesetzt (NotAllowedInExecuting). Zu den Betriebsmodi (siehe enum slsc_OperationMode), in denen diese und andere Kontroll-Funktionen (slsc_ctrl_*) erlaubt sind, siehe Abbildung 37, Seite 98. Verwandter RTC6-Befehl: write_io_port. Die entsprechende Job-Funktion (slsc_list_*) von slsc_ctrl_write_digital_out ist slsc_list_write_digital_out. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.0.
Verweise	slsc_list_write_digital_out

Name der Funktion	<code>slsc_ctrl_write_digital_out_mask</code>
Zweck	Nur im Modus "Manuelle Positionierung"! Schreibt nur diejenigen Bits des <code>Value</code> -Werts auf den 16-Bit-Digital-Ausgang aller RTC6-Karten, die in der benutzerdefinierten Bitmaske (Parameter <code>Mask</code>) festgelegt sind.
Funktions-Signatur	<code>uint32_t slsc_ctrl_write_digital_out_mask(size_t Handle, uint16_t Value, uint16_t Mask);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
	Value 16-Bit-Ausgabewert (DIGITAL OUT 0...DIGITAL OUT 15).
	Mask 16-Bit-Maske (für DIGITAL OUT 0...DIGITAL OUT 15).
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der <code>syncAXIS-DLL</code> Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Das DIGITAL OUT 0...DIGITAL OUT 15-Signal wird ausgegeben bei <ul style="list-style-type: none"> – RTC6 PCI-Express-Karten (wie RTC5-Karten): EXTENSION 1-Stiftleiste, Pin 01...Pin 31 (nur ungeradzahlige Pins) Der Parameter <code>Mask</code> legt fest, welche Bits des 16-Bit-Digital-Ausgangs (siehe <code>slsc_list_write_digital_out</code>) verändert werden, das Argument <code>Value</code> wie sie verändert werden. Die in <code>Mask</code> nicht gesetzten Bits des 16-Bit-Digital-Ausgangs bleiben unverändert. Diese werden also erneut so ausgegeben, wie sie zuletzt waren. Für <code>Mask = 0xFFFF</code> ("setze alle Bits") wirkt <code>slsc_ctrl_write_digital_out_mask</code> wie <code>slsc_ctrl_write_digital_out</code>. <code>slsc_ctrl_write_digital_out_mask</code> wird nicht akzeptiert, wenn gerade ein Job ausgeführt wird. Dann ist beim Rückgabewert Bit #03 gesetzt (<code>NotAllowedInExecuting</code>). Zu den Betriebsmodi (siehe enum <code>slsc_OperationMode</code>), in denen diese und andere Kontroll-Funktionen (<code>slsc_ctrl_*</code>) erlaubt sind, siehe Abbildung 37, Seite 98. Verwandter RTC6-Befehl: <code>write_io_port_mask</code>. Die entsprechende Job-Funktion (<code>slsc_list_*</code>) von <code>slsc_ctrl_write_digital_out_mask</code> ist <code>slsc_list_write_digital_out_mask</code>. Zur Zulässigkeit von <code>syncAXIS control</code>-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab <code>syncAXIS-DLL V1.2.0</code> .
Verweise	<code>slsc_list_write_digital_out_mask</code>

Name der Funktion	slsc_list_arc_abs
Zweck	Definiert einen zu markierenden Kreisbogen (nicht: Ellipsenbogen) mittels absoluter Koordinatenwerte.
Funktions-Signatur	<code>uint32_t slsc_list_arc_abs(size_t Handle, const double* Mid, const double* Target);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mid Zeiger auf ein Array der Dimension 2. Koordinaten eines Punktes auf dem Kreisbogen zwischen dem Zielpunkt der letzten Funktion und Target. In mm.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Der Kreisbogen ist durch 3 Punkte definiert: <ul style="list-style-type: none"> Erster Punkt = Ziel der letzten Funktion. Mid = Ein Punkt (auf dem Kreisbogen) zwischen dem ersten Punkt und Target. Target = Zielpunkt.  Eine gerade Linie wird markiert, wenn die 3 Punkte (fast) auf einer Geraden zu liegen kommen (kollinear sind).  Bei der Ausführung von slsc_list_arc_abs wird der Laser angeschaltet und dann der Kreisbogen mit einer konstanten Geschwindigkeit abgefahren. <ul style="list-style-type: none"> Ist die Markierung kleiner als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit ist die momentan eingestellte Markiergeschwindigkeit. Ist die Markierung größer als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit wird auf eine passende Verfahrtsch-Geschwindigkeit vermindert. Diese ist in der Kennlinie definiert (siehe DynamicReductionFunction). slsc_list_arc_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende slsc_list_para*-Funktion von slsc_list_arc_abs ist slsc_list_para_arc_abs. slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument Target und Mid, siehe auch Seite 268) von slsc_list_arc_abs. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.

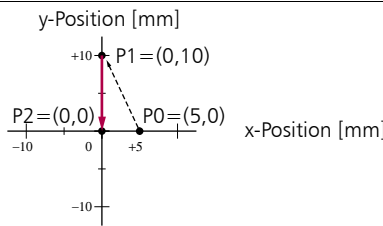
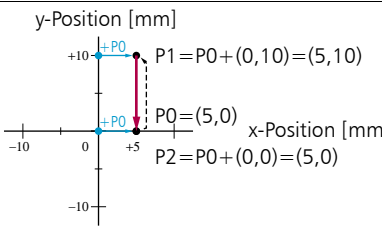
Name der Funktion	slsc_list_arc_abs
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. double Target [2]; // Array of size 2 Target[0] = 1.0; // x value Target[1] = 5.0; // y value double Mid [2]; // Array of size 2 Mid[0] = 2.0; // x value Mid[1] = 4.0; // y value // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_list_arc_abs(Handle, Mid, Target);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_list_para_arc_abs

Name der Funktion	<code>slsc_list_begin</code>
Zweck	Definiert den Beginn eines Jobs . Ist eines von 2 verpflichtenden Strukturelementen eines Jobs .
Funktions-Signatur	<code>uint32_t slsc_list_begin(size_t Handle, size_t* JobID);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	JobID Parameterrückgabe: Zeiger.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Bei der Ausführung von slsc_list_begin erstellt die syncAXIS control-Instanz die Job-ID automatisch. Diese ist fortlaufend und eindeutig. Siehe auch Abbildung 13, Seite 44. Die Job-ID wird bei Event-Callbacks mitgeteilt, siehe Kapitel "Funktionen zum Erfassen von "Callback Events"", Seite 81 slsc_list_begin bewirkt auch eine Aktivierung der durch slsc_cfg_set_jump_speed und slsc_cfg_set_mark_speed gemachten Konfigurations-Änderungen. Der Startpunkt der Markierung ist die aktuelle Verfahrtischposition. Die Auslenkung der Scan-Kopf-Spiegel ist (0,0) (weil slsc_cfg_initialize_from_file wie auch die letzte Job-Funktion (slsc_list_*) die Spiegel auf (0,0) positioniert. slsc_list_begin darf nicht gefolgt werden von einem slsc_list_begin, slsc_list_begin_absolute oder slsc_list_begin_relative. Anderenfalls ist beim Rückgabewert Bit #07 gesetzt (JobStructureNotValid). Die erste Funktion eines Jobs muss slsc_list_begin, slsc_list_begin_absolute oder slsc_list_begin_relative sein. Anderenfalls ist beim Rückgabewert Bit #07 gesetzt (JobStructureNotValid). Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0 .
Verweise	slsc_cfg_initialize_from_file , slsc_cfg_set_jump_speed , slsc_cfg_set_mark_speed , slsc_list_end

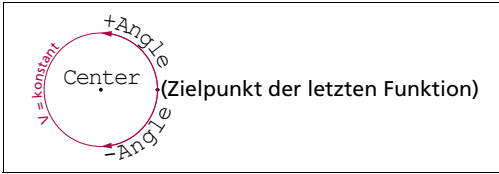
Name der Funktion	slsc_list_begin_absolute						
Zweck	<p>Definiert (alternativ zu slsc_list_begin, slsc_list_begin_relative) den Beginn eines Jobs, um eine (im Modus "Manuelle Positionierung") durch slsc_ctrl_move_stage_abs veranlasste Positionsänderung des Verfahrtschis ausgleichen zu können.</p> <p>Important: slsc_list_begin_absolute kann beim Starten von Jobs zum Abstürzen des Anwenderprogramms führen, wenn Endposition des vorausgehenden Jobs und der bei slsc_list_begin_absolute angegebenen Position nicht zusammenpassen!</p>						
Funktions-Signatur	<code>uint32_t slsc_list_begin_absolute(size_t Handle, size_t* JobID, const double* StartPosition);</code>						
Argument(e)	<table><tr><td>Handle</td><td>Handle auf eine syncAXIS control-Instanz.</td></tr><tr><td>JobID</td><td>Parameterrückgabe: Zeiger.</td></tr><tr><td>StartPosition</td><td>Zeiger auf ein Array der Dimension 2. Koordinaten des Verfahrtschis: Position zu dem Zeitpunkt, an dem dieser Job ausgeführt werden wird. In mm.</td></tr></table>	Handle	Handle auf eine syncAXIS control-Instanz .	JobID	Parameterrückgabe: Zeiger.	StartPosition	Zeiger auf ein Array der Dimension 2. Koordinaten des Verfahrtschis: Position zu dem Zeitpunkt, an dem dieser Job ausgeführt werden wird. In mm.
Handle	Handle auf eine syncAXIS control-Instanz .						
JobID	Parameterrückgabe: Zeiger.						
StartPosition	Zeiger auf ein Array der Dimension 2. Koordinaten des Verfahrtschis: Position zu dem Zeitpunkt, an dem dieser Job ausgeführt werden wird. In mm.						
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.						
Kommentar(e)	<ul style="list-style-type: none">Bei der Ausführung von slsc_list_begin_absolute (genauso wie bei slsc_list_begin, slsc_list_begin_relative) erstellt die syncAXIS control-Instanz die Job-ID automatisch. Diese ist fortlaufend und eindeutig. Siehe auch Abbildung 13, Seite 44.Die Job-ID wird bei Event-Callbacks mitgeteilt, siehe Kapitel "Funktionen zum Erfassen von "Callback Events"", Seite 81.slsc_list_begin_absolute (genauso wie slsc_list_begin, slsc_list_begin_relative) bewirkt auch eine Aktivierung der durch slsc_cfg_set_jump_speed und slsc_cfg_set_mark_speed gemachten Konfigurations-Änderungen.slsc_list_begin_absolute ermöglicht einen Job-Start an einer anderen, als der aktuellen Verfahrtschisposition oder geplanten aktuellen Verfahrtschisposition. Dies ermöglicht ein schnelles Verfahrtschisfreigeben, -verfahren, und -wiederakquirieren sogar während die syncAXIS control-Instanz Jobs noch berechnet und vorbereitet. Ein Zeitgewinn ist möglich, dafür ist die Verwendung komplizierter.Es findet eine Überprüfung statt, ob die tatsächliche Verfahrtschisposition zum Zeitpunkt der Job-Ausführung mit der bei StartPosition angegebenen übereinstimmt. Bei Nichtübereinstimmung ist beim Rückgabewert das Bit #12 gesetzt (InvalidPosition; Der Verfahrtschis muss dann an die korrekte Position verfahren werden, siehe auch Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70).slsc_list_begin_absolute darf nicht gefolgt werden von einem slsc_list_begin, slsc_list_begin_absolute oder slsc_list_begin_relative. Anderenfalls ist beim Rückgabewert Bit #07 gesetzt (JobStructureNotValid).Die erste Funktion eines Jobs muss slsc_list_begin, slsc_list_begin_absolute oder slsc_list_begin_relative sein. Anderenfalls ist beim Rückgabewert Bit #07 gesetzt (JobStructureNotValid).Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.						

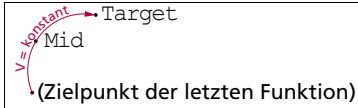
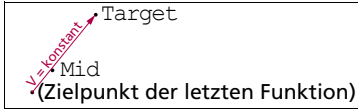
Name der Funktion	slsc_list_begin_absolute
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.0.7.
Verweise	slsc_ctrl_follow , slsc_ctrl_unfollow

Name der Funktion	<code>slsc_list_begin_module</code>
Zweck	Nur im Simulationsmodus erlaubt. Zur "Vorbereitung eines Jobs": Definiert den Anfang eines aufzuzeichnenden Jobs (Modul). Dieser wird wie üblich mit <code>slsc_list_end</code> geschlossen.
Funktions-Signatur	<code>uint32_t slsc_list_begin_module(size_t Handle, size_t* JobID, const double* StartPosition, const char* ModuleFileName);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	JobID Parameterrückgabe: Zeiger.
	StartPosition Zeiger auf ein Array der Dimension 2. Start-Position des aufzuzeichnenden Jobs. In mm.
	ModuleFileName Absoluter Dateipfad der erzeugten Modul-Datei (*.slm). Zeiger auf einen \0-terminierten ANSI-String, 1 Byte pro Zeichen.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_list_begin_module</code> ist nur im Simulationsmodus erlaubt. Sonst ist beim Rückgabewert Bit #09 gesetzt (<code>NotAllowedInCurrentConfiguration</code>). Speziell deshalb wird <code>slsc_cfg_initialize_copy</code> zur Verfügung gestellt. <code>slsc_list_begin_module</code> ist in erlaubt im Betriebsmodus "ScannerOnly", "StageOnly", "ScannerAndStage". Für <code>slsc_list_begin_module</code> gibt es keine entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>). <code>slsc_list_begin_module</code> ist ähnlich zu <code>slsc_list_begin_absolute</code>, siehe daher auch Kommentar(e) dort. Die Startposition für den Job ist zu definieren mit <code>StartPosition</code>. Siehe auch Abschnitt "Funktionen für "Module"", Seite 95. Zu Eigenschaften und Inhalt von Modul-Dateien siehe Abschnitt "Modul-Datei", Seite 66. Die Aufzeichnung der Modul-Datei startet bereits beim Aufruf von <code>slsc_list_begin_module</code>. Ein zusätzlicher Funktions-Aufruf, etwa von <code>slsc_ctrl_start_execution</code>, ist nicht nötig. Die Aufzeichnung der Modul-Datei endet nach der Trajektorienplanung, siehe Abbildung 11, Seite 41. Um den Aufzeichnungsende-Zeitpunkt zu bestimmen, kann <code>slsc_cfg_register_callback_job_end_planned</code> verwendet werden. Dazu finden Sie ein Code-Beispiel in Abbildung 28, Seite 68. Siehe Kapitel 2.11 "Über das Arbeiten mit "Modulen"", Seite 65. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	Siehe Abbildung 28, Seite 68.
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0.
Verweise	<code>slsc_list_playback_module</code> , <code>slsc_cfg_initialize_copy</code>

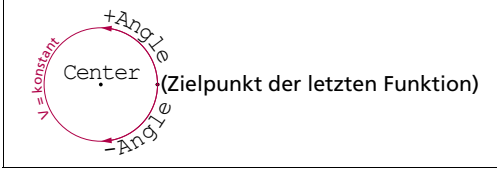
Name der Funktion	<code>slsc_list_begin_relative</code>
Zweck	<p>Definiert (alternativ zu <code>slsc_list_begin</code>) den Beginn eines Jobs. Ist eines von 2 verpflichtenden Strukturelementen eines Jobs.</p> <p>Unterschied zu <code>slsc_list_begin</code>: Bei <code>ScannerAndStage</code> und <code>StageOnly</code> wird die Verfahrtschposition (die bei Beginn der Ausführung dieses Jobs vorläge) berechnet. Diese wird dann als Offset den Koordinaten in diesem Job (und nur für diesen Job!) hinzugefügt.</p>
Funktions-Signatur	<code>uint32_t slsc_list_begin_relative(size_t Handle, size_t* JobID);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	JobID Parameterrückgabe: Zeiger.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Bei der Ausführung von <code>slsc_list_begin_relative</code> (genauso wie bei <code>slsc_list_begin</code>) erstellt die syncAXIS control-Instanz die Job-ID automatisch. Diese ist fortlaufend und eindeutig. Siehe auch Abbildung 13, Seite 44. Die Job-ID wird bei Event-Callbacks mitgeteilt, siehe Kapitel "Funktionen zum Erfassen von "Callback Events"", Seite 81. <code>slsc_list_begin_relative</code> (genauso wie <code>slsc_list_begin</code>) bewirkt auch eine Aktivierung der durch <code>slsc_cfg_set_jump_speed</code> und <code>slsc_cfg_set_mark_speed</code> gemachten Konfigurations-Änderungen. Wie bei <code>slsc_list_begin</code> ist der Startpunkt der Markierung auch die aktuelle Verfahrtschposition. Jedoch wird bei <code>ScannerAndStage</code> und <code>StageOnly</code> diese Verfahrtschposition dann den Koordinaten in diesem Job (bis zum <code>slsc_list_end</code>) hinzugefügt ("Offset"). Die Auslenkung der Scan-Kopf-Spiegel ist (0,0) (weil <code>slsc_cfg_initialize_from_file</code> wie auch die letzte Job-Funktion (<code>slsc_list_*</code>) die Spiegel auf (0,0) positioniert. <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;">  <pre> // Pseudo code slsc_list_begin // does NOT add initial stage pos P0 // as offset to coordinates slsc_list_jump_abs(0,10) slsc_list_mark_abs(0,0) slsc_list_end </pre> </div> <div style="text-align: center;">  <pre> // Pseudo code slsc_list_begin_relative // adds initial stage pos P0 // as offset to coordinates slsc_list_jump_abs(0,10) slsc_list_mark_abs(0,0) slsc_list_end </pre> </div> </div>

Name der Funktion	<code>slsc_list_begin_relative</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • <code>slsc_list_begin_relative</code> darf nicht gefolgt werden von einem <code>slsc_list_begin</code>, <code>slsc_list_begin_absolute</code> oder <code>slsc_list_begin_relative</code>. Anderenfalls ist beim Rückgabewert Bit #07 gesetzt (<code>JobStructureNotValid</code>). • Die erste Funktion eines <code>Jobs</code> muss <code>slsc_list_begin</code>, <code>slsc_list_begin_absolute</code> oder <code>slsc_list_begin_relative</code> sein. Anderenfalls ist beim Rückgabewert Bit #07 gesetzt (<code>JobStructureNotValid</code>). • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	<code>slsc_list_begin</code>

Name der Funktion	<code>slsc_list_circle_2d_abs</code>
Zweck	Definiert einen Kreis (nicht: Ellipse) über den absoluten Koordinatenwert des Kreismittelpunkts. Der Parameter <code>Angle</code> bestimmt die Richtung, mit der die Markierung ausgeführt wird, sowie die Anzahl der Umdrehungen (z.B. $3,25 \times 2\pi$).
Funktions-Signatur	<code>uint32_t slsc_list_circle_2d_abs(size_t Handle, const double* Center, double Angle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Center Zeiger auf ein Array der Dimension 2. Koordinaten (x-Wert und y-Wert) des Kreismittelpunkts. In mm.
	Angle In rad. Positive Werte: Markierung erfolgt gegen den Uhrzeigersinn. Negative Werte: Markierung erfolgt im Uhrzeigersinn.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Der Kreis ist durch 2 Punkte definiert: <ul style="list-style-type: none"> Erster Punkt = Ziel der letzten Markier-Funktion oder Sprung-Funktion. Center = Kreismittelpunkt.  Bei der Ausführung von <code>slsc_list_circle_2d_abs</code> wird der Laser angeschaltet und dann der Kreis mit einer konstanten Geschwindigkeit abgefahren. Sowohl die Ausführungsrichtung der Markierung als auch die Anzahl der Umdrehungen (z.B. $3,25 \times 2\pi$) wird durch das Argument <code>Angle</code> bestimmt. <code>slsc_list_circle_2d_abs</code> kann daher (alternativ zu <code>slsc_list_arc_abs</code>) auch zum Markieren von Kreissegmenten verwendet werden. <code>slsc_list_circle_2d_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende <code>slsc_list_para*</code>-Funktion von <code>slsc_list_circle_2d_abs</code> ist <code>slsc_list_para_circle_2d_abs</code>. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Center</code>, siehe auch Seite 268) von <code>slsc_list_circle_2d_abs</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	<code>slsc_list_arc_abs</code> , <code>slsc_list_para_circle_2d_abs</code>

Name der Funktion	slsc_list_dashed_arc_abs
Zweck	Wie slsc_list_arc_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente <code>NSwitches</code> und <code>LaserSwitches</code> , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Funktions-Signatur	<code>uint32_t slsc_list_dashed_arc_abs(size_t Handle, const double* Mid, const double* Target, size_t NSwitches, const double* LaserSwitches);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Mid Zeiger auf ein Array der Dimension 2. Koordinaten eines Punktes auf dem Kreisbogen zwischen dem Zielpunkt der letzten Funktion und <code>Target</code> . In mm.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	NSwitches NSwitches ist die Größe des LaserSwitches-Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double-Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Der Kreisbogen ist durch 3 Punkte definiert: <ul style="list-style-type: none"> – Erster Punkt = Ziel der letzten Funktion. – Mid = Ein Punkt (auf dem Kreisbogen) zwischen dem ersten Punkt und <code>Target</code>. – Target = Zielpunkt.  Eine gerade Linie wird markiert, wenn die 3 Punkte (fast) auf einer Geraden zu liegen kommen (kollinear sind).  Bei der Ausführung von slsc_list_dashed_arc_abs wird der Laser angeschaltet und dann der Kreisbogen mit einer konstanten Geschwindigkeit abgefahren. <ul style="list-style-type: none"> – Ist die Markierung kleiner als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit ist die momentan eingestellte Markiergeschwindigkeit. – Ist die Markierung größer als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit wird auf eine passende Verfahrtsch-Geschwindigkeit vermindert. Diese ist in der Kennlinie definiert (siehe DynamicReductionFunction). slsc_list_dashed_arc_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45.

Name der Funktion	<code>slsc_list_dashed_arc_abs</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Die entsprechende <code>slsc_list_para*</code>-Funktion von <code>slsc_list_dashed_arc_abs</code> ist <code>slsc_list_para_dashed_arc_abs</code>. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code> und <code>Mid</code>, siehe auch Seite 268) von <code>slsc_list_dashed_arc_abs</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_list_arc_abs , slsc_list_para_dashed_arc_abs

Name der Funktion	<code>slsc_list_dashed_circle_2d_abs</code>
Zweck	Wie <code>slsc_list_circle_2d_abs</code> , aber die korrespondierende <code>[*]dashed[*]</code> -Funktion. Bietet daher zusätzlich die Argumente <code>NSwitches</code> und <code>LaserSwitches</code> , siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91 .
Funktions-Signatur	<code>uint32_t slsc_list_dashed_circle_2d_abs(size_t Handle, const double* Center, double Angle, size_t NSwitches, const double* LaserSwitches);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Center Zeiger auf ein Array der Dimension 2. Koordinaten (x-Wert und y-Wert) des Kreismittelpunkts. In mm.
	Angle In rad. Positive Werte: Markierung erfolgt gegen den Uhrzeigersinn. Negative Werte: Markierung erfolgt im Uhrzeigersinn.
	NSwitches NSwitches ist die Größe des LaserSwitches -Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91 .
	LaserSwitches LaserSwitches ist ein Array von <code>double</code> -Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91 .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Der Kreis ist durch 2 Punkte definiert: <ul style="list-style-type: none"> Erster Punkt = Ziel der letzten Markier-Funktion oder Sprung-Funktion. Center = Kreismittelpunkt.  Bei der Ausführung von <code>slsc_list_dashed_circle_2d_abs</code> wird der Laser angeschaltet und dann der Kreis mit einer konstanten Geschwindigkeit abgefahren. Sowohl die Ausführungsrichtung der Markierung als auch die Anzahl der Umdrehungen (z. B. $3,25 \times 2\pi$) wird durch das Argument <code>Angle</code> bestimmt. <code>slsc_list_circle_2d_abs</code> kann daher (alternativ zu <code>slsc_list_dashed_arc_abs</code>) auch zum Markieren von Kreissegmenten verwendet werden. <code>slsc_list_dashed_circle_2d_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45.

Name der Funktion	<code>slsc_list_dashed_circle_2d_abs</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Die entsprechende <code>slsc_list_para*</code>-Funktion von <code>slsc_list_dashed_circle_2d_abs</code> ist <code>slsc_list_para_dashed_circle_2d_abs</code>. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Center</code>, siehe auch Seite 268) von <code>slsc_list_dashed_circle_2d_abs</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_list_circle_2d_abs</code> , <code>slsc_list_para_dashed_circle_2d_abs</code>

Name der Funktion	slsc_list_dashed_mark_abs
Zweck	Wie slsc_list_mark_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Funktions-Signatur	<code>uint32_t slsc_list_dashed_mark_abs(size_t Handle, const double* Target, size_t NSwitches, const double* LaserSwitches);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	NSwitches NSwitches ist die Größe des LaserSwitches -Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double -Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Das Verhalten von Markiervektoren wird in der Konfiguration der Trajektorienplanung festgelegt, siehe slsc_MarkConfig (z.B. MarkSpeed) und slsc_GeometryConfig (z.B. MaxBlendRadius). slsc_list_dashed_mark_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende slsc_list_para*-Funktion von slsc_list_dashed_mark_abs ist slsc_list_para_dashed_mark_abs. slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument Target, siehe auch Seite 268) von slsc_list_dashed_mark_abs. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung" ", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0 .
Verweise	slsc_list_mark_abs , slsc_list_para_dashed_mark_abs

Name der Funktion	slsc_list_end
Zweck	Definiert das Ende eines Jobs . Ist eines von 2 verpflichtenden Strukturelementen eines Jobs .
Funktions-Signatur	<code>uint32_t slsc_list_end(size_t Handle);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_list_end beendet den Job <ul style="list-style-type: none"> – mit einem Sprung auf die Galvanometerscannerposition 0,0 des Scan-Kopfs – ohne die Verfahrtschposition zu ändern, d.h. der Verfahrtsch verbleibt auf der zuletzt eingestellten Sprungposition bzw. Markierposition. • Um nachfolgend den Verfahrtsch auf eine gewünschte Position zu verfahren: <ul style="list-style-type: none"> – definieren Sie einen weiteren Job – setzen darin mit slsc_cfg_set_mode den Betriebsmodus auf StageOnly und – definieren einen Sprung mit slsc_list_jump_abs auf die Zielposition. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	slsc_cfg_set_mode , slsc_list_jump_abs , slsc_list_begin

Name der Funktion	slsc_list_jump_abs
Zweck	Definiert einen Sprung mittels absoluter Koordinatenwerte.
Funktions-Signatur	<code>uint32_t slsc_list_jump_abs(size_t Handle, const double* Target);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Das Verhalten von Sprüngen wird in der Konfiguration der Trajektorienplanung festgelegt, siehe slsc_MarkConfig (z.B. LaserMinOffTime und JumpSpeed). slsc_list_jump_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende slsc_list_para*-Funktion von slsc_list_jump_abs ist slsc_list_para_jump_abs. slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument Target, siehe auch Seite 268) von slsc_list_jump_abs. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. double Target [2]; // Array of size 2 Target[0] = 2.0; x value Target[1] = 4.0; y value // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_list_jump_abs(Handle, Target);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0 .
Verweise	slsc_list_para_jump_abs

Name der Funktion	<code>slsc_list_jump_abs_min_time</code>
Zweck	Wie <code>slsc_list_jump_abs</code> . Erlaubt aber zusätzlich, eine Minstdauer für den Sprung anzugeben.
Funktions-Signatur	<code>uint32_t slsc_list_jump_abs_min_time(size_t Handle, const double* Target, double MinimalJumpTime);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	MinimalJumpTime Minstdauer für den Sprung. In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Das Verhalten von Sprüngen wird in der Konfiguration der Trajektorienplanung festgelegt, siehe slsc_MarkConfig (z.B. LaserMinOffTime und JumpSpeed). <code>slsc_list_jump_abs_min_time</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende <code>slsc_list_para*</code>-Funktion von <code>slsc_list_jump_abs_min_time</code> ist slsc_list_para_jump_abs_min_time. slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument <code>Target</code>, siehe auch Seite 268) von slsc_list_jump_abs_min_time. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. double Target [2]; // Array of size 2 Target[0] = 2.0; x value Target[1] = 4.0; y value // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_list_jump_abs_min_time(Handle, Target, 0.0001);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0.
Verweise	slsc_list_para_jump_abs_min_time

Name der Funktion	slsc_list_mark_abs
Zweck	Definiert einen Markiervektor mittels absoluter Koordinatenwerte.
Funktions-Signatur	<code>uint32_t slsc_list_mark_abs(size_t Handle, const double* Target);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Das Verhalten von Markiervektoren wird in der Konfiguration der Trajektorienplanung festgelegt, siehe slsc_MarkConfig (z.B. MarkSpeed) und slsc_GeometryConfig (z.B. MaxBlendRadius). slsc_list_mark_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende slsc_list_para*-Funktion von slsc_list_mark_abs ist slsc_list_para_mark_abs. slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument Target, siehe auch Seite 268) von slsc_list_mark_abs. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. double Target [2]; // Array of size 2 Target[0] = 2.0; Target[1] = 4.0; // Handle: siehe Code-Beispiel bei slsc_cfg_initialize_from_file slsc_list_mark_abs(Handle, Target);</pre>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0 .
Verweise	slsc_list_para_mark_abs , slsc_list_wait_with_laser_on

Name der Funktion	<code>slsc_list_multi_para_arc_abs</code>
Zweck	Wie <code>slsc_list_arc_abs</code> . Bietet aber zusätzlich das Argument <code>MultiParaTarget</code> , über das (je "ActiveChannel") eine aus mehreren Abschnitten bestehende Rampe definiert wird.
Funktions-Signatur	<code>uint32_t slsc_list_multi_para_arc_abs(size_t Handle, const double* Mid, const double* Target, const slsc_MultiParaTarget* MultiParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mid Zeiger auf ein Array der Dimension 2. Koordinaten eines Punktes auf dem Kreisbogen zwischen dem Zielpunkt der letzten Funktion und <code>Target</code> . In mm.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	MultiParaTarget Zeiger auf ein Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250 . 1 Rampe pro "ActiveChannel" (es gibt max. 2 "ActiveChannel"), die aus mehreren Abschnitten (ds) besteht, siehe Struktur slsc_MultiParaTarget .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Siehe slsc_list_para_arc_abs. slsc_list_multi_para_arc_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Am Beginn der Rampe wird als Startwert (Faktor Ip) der erreichte Endwert der vorherigen slsc_list_[para/multi_para]*-Funktion (oder anfänglich <code>ParaTargetDefault</code>) verwendet. Wenn die Summe der Abschnitte (ds) <i>kürzer</i> ist als der Kreisbogen dauert, dann wird der Rampenwert, der am Ende des letzten Abschnitts erreicht ist, unverändert bis zum Ende des Markiervektors beibehalten. Wenn die Summe der Abschnitte (ds) <i>länger</i> ist als der Kreisbogen dauert, dann wird die Rampe zu diesem Zeitpunkt abgebrochen. Der bis dahin erreichte Rampenwert wird als Startwert für die nachfolgende Rampe verwendet. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. Siehe auch Abschnitt "Über Rampen", Seite 53. slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument <code>Target</code> und <code>Mid</code>, siehe auch Seite 268) von slsc_list_multi_para_arc_abs. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung" ", Seite 70.
Code-Beispiel	Siehe Abschnitt "Über Rampen" , Seite 53 .
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_list_para_arc_abs

Name der Funktion	<code>slsc_list_multi_para_circle_2d_abs</code>
Zweck	Wie <code>slsc_list_circle_2d_abs</code> . Bietet aber zusätzlich das Argument <code>MultiParaTarget</code> , über das (je "ActiveChannel") eine aus mehreren Abschnitten bestehende Rampe definiert wird.
Funktions-Signatur	<code>uint32_t slsc_list_multi_para_circle_2d_abs(size_t Handle, const double* Center, double Angle, const slsc_MultiParaTarget* MultiParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Center Zeiger auf ein Array der Dimension 2. Koordinaten (x-Wert und y-Wert) des Kreismittelpunkts. In mm.
	Angle In rad. Positive Werte: Markierung erfolgt gegen den Uhrzeigersinn. Negative Werte: Markierung erfolgt im Uhrzeigersinn.
	MultiParaTarget Zeiger auf ein Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250 . 1 Rampe pro "ActiveChannel" (es gibt max. 2 "ActiveChannel"), die aus mehreren Abschnitten (ds) besteht, siehe Struktur slsc_MultiParaTarget .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Siehe <code>slsc_list_para_circle_2d_abs</code>. <code>slsc_list_multi_para_circle_2d_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Am Beginn der Rampe wird als Startwert (Faktor Ip) der erreichte Endwert der vorherigen <code>slsc_list_[para/multi_para]*</code>-Funktion (oder anfänglich <code>ParaTargetDefault</code>) verwendet. Wenn die Summe der Abschnitte (ds) kürzer ist als die Gesamt-Bogenlänge ($\text{Radius} \times \text{Angle}$) dauert, dann wird der Rampenwert, der am Ende des letzten Abschnitts erreicht ist, unverändert bis zum Ende des Markiervektors beibehalten. Wenn die Summe der Abschnitte (ds) länger ist als die Gesamt-Bogenlänge ($\text{Radius} \times \text{Angle}$) dauert, dann wird die Rampe zu diesem Zeitpunkt abgebrochen. Der bis dahin erreichte Rampenwert wird als Startwert für die nachfolgende Rampe verwendet. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. Siehe auch Abschnitt "Über Rampen", Seite 53. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Center</code>, siehe auch Seite 268) von <code>slsc_list_multi_para_circle_2d_abs</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung" ", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	<code>slsc_list_para_circle_2d_abs</code>

Name der Funktion	<code>slsc_list_multi_para_dashed_arc_abs</code>
Zweck	Wie <code>slsc_list_multi_para_arc_abs</code> , aber die korrespondierende <code>[*]dashed[*]</code> -Funktion. Bietet daher zusätzlich die Argumente <code>NSwitches</code> und <code>LaserSwitches</code> , siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91 .
Funktions-Signatur	<code>uint32_t slsc_list_multi_para_dashed_arc_abs(size_t Handle, const double* Mid, const double* Target, size_t NSwitches, const double* LaserSwitches, const slsc_MultiParaTarget* MultiParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mid Zeiger auf ein Array der Dimension 2. Koordinaten eines Punktes auf dem Kreisbogen zwischen dem Zielpunkt der letzten Funktion und <code>Target</code> . In mm.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	NSwitches NSwitches ist die Größe des LaserSwitches-Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double-Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
	MultiParaTarget Zeiger auf ein Array der Dimension 2 oder 1, siehe Kommentar(e), Seite 250 . 1 Rampe pro "ActiveChannel" (es gibt max. 2 "ActiveChannel"), die aus mehreren Abschnitten (<code>ds</code>) besteht, siehe Struktur <code>slsc_MultiParaTarget</code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Siehe <code>slsc_list_para_arc_abs</code>. <code>slsc_list_multi_para_dashed_arc_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Am Beginn der Rampe wird als Startwert (Faktor Ip) der erreichte Endwert der vorherigen <code>slsc_list_[para/multi_para]*</code>-Funktion (oder anfänglich <code>ParaTargetDefault</code>) verwendet. Wenn die Summe der Abschnitte (<code>ds</code>) kürzer ist als der Kreisbogen dauert, dann wird der Rampenwert, der am Ende des letzten Abschnitts erreicht ist, unverändert bis zum Ende des Markiervektors beibehalten. Wenn die Summe der Abschnitte (<code>ds</code>) länger ist als der Kreisbogen dauert, dann wird die Rampe zu diesem Zeitpunkt abgebrochen. Der bis dahin erreichte Rampenwert wird als Startwert für die nachfolgende Rampe verwendet.

Name der Funktion	slsc_list_multi_para_dashed_arc_abs
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. • Siehe auch Abschnitt "Über Rampen", Seite 53. • slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument <code>Target</code> und <code>Mid</code>, siehe auch Seite 268) von slsc_list_multi_para_dashed_arc_abs. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_list_para_arc_abs

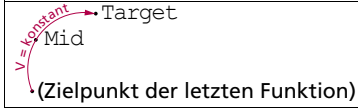
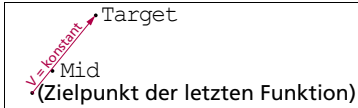
Name der Funktion	<code>slsc_list_multi_para_dashed_circle_2d_abs</code>
Zweck	Wie <code>slsc_list_multi_para_circle_2d_abs</code> , aber die korrespondierende <code>[*]dashed[*]</code> -Funktion. Bietet daher zusätzlich die Argumente <code>NSwitches</code> und <code>LaserSwitches</code> , siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
Funktions-Signatur	<code>uint32_t slsc_list_multi_para_dashed_circle_2d_abs(size_t Handle, const double* Center, double Angle, size_t NSwitches, const double* LaserSwitches, const slsc_MultiParaTarget* MultiParaTarget);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
	Center Zeiger auf ein Array der Dimension 2. Koordinaten (x-Wert und y-Wert) des Kreismittelpunkts. In mm.
	Angle In rad. Positive Werte: Markierung erfolgt gegen den Uhrzeigersinn. Negative Werte: Markierung erfolgt im Uhrzeigersinn.
	NSwitches NSwitches ist die Größe des LaserSwitches-Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double-Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen", Seite 91.
	MultiParaTarget Zeiger auf ein Array der Dimension 2 oder 1, siehe Kommentar(e), Seite 250. 1 Rampe pro "ActiveChannel" (es gibt max. 2 "ActiveChannel"), die aus mehreren Abschnitten (<code>ds</code>) besteht, siehe Struktur <code>slsc_MultiParaTarget</code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Siehe <code>slsc_list_para_circle_2d_abs</code>. <code>slsc_list_multi_para_dashed_circle_2d_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Am Beginn der Rampe wird als Startwert (Faktor <code>lp</code>) der erreichte Endwert der vorherigen <code>slsc_list_[para/multi_para]*</code>-Funktion (oder anfänglich <code>ParaTargetDefault</code>) verwendet. Wenn die Summe der Abschnitte (<code>ds</code>) kürzer ist als die Gesamt-Bogenlänge ($\text{Radius} \times \text{Angle}$) dauert, dann wird der Rampenwert, der am Ende des letzten Abschnitts erreicht ist, unverändert bis zum Ende des Markiervektors beibehalten. Wenn die Summe der Abschnitte (<code>ds</code>) länger ist als die Gesamt-Bogenlänge ($\text{Radius} \times \text{Angle}$) dauert, dann wird die Rampe zu diesem Zeitpunkt abgebrochen. Der bis dahin erreichte Rampenwert wird als Startwert für die nachfolgende Rampe verwendet.

Name der Funktion	<code>slsc_list_multi_para_dashed_circle_2d_abs</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (<code>slsc_list_[para/multi_para]*</code>-Funktionen)", Seite 92. • Siehe auch Abschnitt "Über Rampen", Seite 53. • slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument <code>Center</code>, siehe auch Seite 268) von slsc_list_multi_para_dashed_circle_2d_abs. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_list_multi_para_circle_2d_abs

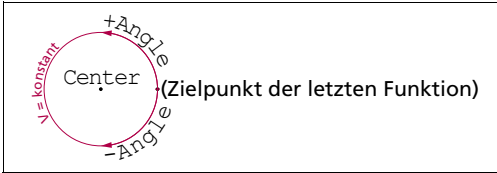
Name der Funktion	slsc_list_multi_para_dashed_mark_abs
Zweck	Wie slsc_list_multi_para_mark_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Funktions-Signatur	<code>uint32_t slsc_list_multi_para_dashed_mark_abs(size_t Handle, const double* Target, size_t NSwitches, const double* LaserSwitches, const slsc_MultiParaTarget* MultiParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	NSwitches NSwitches ist die Größe des LaserSwitches-Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double-Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	MultiParaTarget Zeiger auf ein Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. 1 Rampe pro "ActiveChannel" (es gibt max. 2 "ActiveChannel"), die aus mehreren Abschnitten (ds) besteht, siehe Struktur slsc_MultiParaTarget .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Siehe slsc_list_para_mark_abs. slsc_list_multi_para_dashed_mark_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Am Beginn der Rampe wird als Startwert (Faktor Ip) der erreichte Endwert der vorherigen slsc_list_[para/multi_para]*-Funktion (oder anfänglich ParaTargetDefault) verwendet. Wenn die Summe der Abschnitte (ds) kürzer ist als die Markiervektorlänge dauert, dann wird der Rampenwert, der am Ende des letzten Abschnitts erreicht ist, unverändert bis zum Ende des Markiervektors beibehalten. Wenn die Summe der Abschnitte (ds) länger ist als die Markiervektorlänge dauert, dann wird die Rampe zu diesem Zeitpunkt abgebrochen. Der bis dahin erreichte Rampenwert wird als Startwert für die nachfolgende Rampe verwendet.

Name der Funktion	<code>slsc_list_multi_para_dashed_mark_abs</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (<code>slsc_list_[para/multi_para]*</code>-Funktionen)", Seite 92. • Siehe auch Abschnitt "Über Rampen", Seite 53. • <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code>, siehe auch Seite 268) von <code>slsc_list_multi_para_dashed_mark_abs</code>. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_list_multi_para_mark_abs

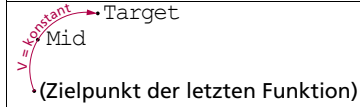
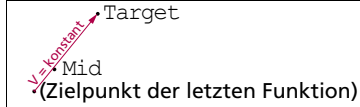
Name der Funktion	<code>slsc_list_multi_para_mark_abs</code>
Zweck	Wie <code>slsc_list_mark_abs</code> . Bietet aber zusätzlich das Argument <code>MultiParaTarget</code> , über das (je "ActiveChannel") eine aus mehreren Abschnitten bestehende Rampe definiert wird.
Funktions-Signatur	<code>uint32_t slsc_list_multi_para_mark_abs(size_t Handle, const double* Target, const slsc_MultiParaTarget* MultiParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	MultiParaTarget Zeiger auf ein Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250 . 1 Rampe pro "ActiveChannel" (es gibt max. 2 "ActiveChannel"), die aus mehreren Abschnitten (ds) besteht, siehe Struktur slsc_MultiParaTarget .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Siehe <code>slsc_list_para_mark_abs</code>. <code>slsc_list_multi_para_mark_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Am Beginn der Rampe wird als Startwert (Faktor Ip) der erreichte Endwert der vorherigen <code>slsc_list_[para/multi_para]*</code>-Funktion (oder anfänglich <code>ParaTargetDefault</code>) verwendet. Wenn die Summe der Abschnitte (ds) <i>kürzer</i> ist als die Markiervektorenlänge dauert, dann wird der Rampenwert, der am Ende des letzten Abschnitts erreicht ist, unverändert bis zum Ende des Markiervektors beibehalten. Wenn die Summe der Abschnitte (ds) <i>länger</i> ist als die Markiervektorenlänge dauert, dann wird die Rampe zu diesem Zeitpunkt abgebrochen. Der bis dahin erreichte Rampenwert wird als Startwert für die nachfolgende Rampe verwendet. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. Siehe auch Abschnitt "Über Rampen", Seite 53. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code>, siehe auch Seite 268) von <code>slsc_list_multi_para_mark_abs</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	<code>slsc_list_para_mark_abs</code>

Name der Funktion	slsc_list_para_arc_abs
Zweck	Wie slsc_list_arc_abs . Bietet aber zusätzlich das Argument <code>ParaTarget</code> , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
Funktions-Signatur	<code>uint32_t slsc_list_para_arc_abs(size_t Handle, const double* Mid, const double* Target, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Mid Zeiger auf ein Array der Dimension 2. Koordinaten eines Punktes auf dem Kreisbogen zwischen dem Zielpunkt der letzten Funktion und <code>Target</code> . In mm.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. <code>ParaTarget</code> bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel-Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur", Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei slsc_list_jump_abs – Der Kreisbogen ist durch 3 Punkte definiert: <ul style="list-style-type: none"> – Erster Punkt = Ziel der letzten Funktion. – Mid = Ein Punkt (auf dem Kreisbogen) zwischen dem ersten Punkt und <code>Target</code>. – Target = Zielpunkt.  Wie bei slsc_list_jump_abs – Eine gerade Linie wird markiert, wenn die 3 Punkte (fast) auf einer Geraden zu liegen kommen (kollinear sind).  Wie bei slsc_list_jump_abs – Bei der Ausführung von slsc_list_para_arc_abs wird der Laser angeschaltet und dann der Kreisbogen mit einer konstanten Geschwindigkeit abgefahren. <ul style="list-style-type: none"> – Ist die Markierung kleiner als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit ist die momentan eingestellte Markiergeschwindigkeit. – Ist die Markierung größer als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit wird auf eine passende Verfahrtisch-Geschwindigkeit vermindert. Diese ist in der Kennlinie definiert (siehe DynamicReductionFunction). slsc_list_para_arc_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45.

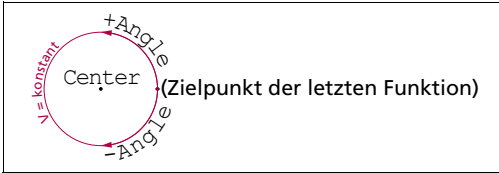
Name der Funktion	<code>slsc_list_para_arc_abs</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Jede <code>slsc_list_[para/multi_para]*</code>-Funktion wirkt wie ihre entsprechende <code>slsc_list*</code>-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. • Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. • <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code> und <code>Mid</code>, siehe auch Seite 268) von <code>slsc_list_para_arc_abs</code>. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_list_arc_abs , slsc_list_para_circle_2d_abs , slsc_list_para_disable , slsc_list_para_enable , slsc_list_para_jump_abs , slsc_list_para_mark_abs

Name der Funktion	slsc_list_para_circle_2d_abs
Zweck	Wie slsc_list_circle_2d_abs . Bietet aber zusätzlich das Argument ParaTarget , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
Funktions-Signatur	<code>uint32_t slsc_list_para_circle_2d_abs(size_t Handle, const double* Center, double Angle, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Center Zeiger auf ein Array der Dimension 2. Koordinaten (x-Wert und y-Wert) des Kreismittelpunkts. In mm.
	Angle In rad. Positive Werte: Markierung erfolgt gegen den Uhrzeigersinn. Negative Werte: Markierung erfolgt im Uhrzeigersinn.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. ParaTarget bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel-Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur" , Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei slsc_list_circle_2d_abs - der Kreis ist durch 2 Punkte definiert: <ul style="list-style-type: none"> Erster Punkt = Ziel der letzten Markier-Funktion oder Sprung-Funktion. Center = Kreismittelpunkt.  Wie bei slsc_list_circle_2d_abs - bei der Ausführung von slsc_list_para_circle_2d_abs wird der Laser angeschaltet und dann der Kreis mit einer konstanten Geschwindigkeit abgefahren. Sowohl die Ausführungsrichtung der Markierung als auch die Anzahl der Umdrehungen (z.B. $3,25 \times 2\pi$) wird durch das Argument Angle bestimmt. slsc_list_para_circle_2d_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Jede slsc_list_[para/multi_para]*-Funktion wirkt wie ihre entsprechende slsc_list*-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. Vor slsc_list_para_circle_2d_abs muss slsc_list_para_enable aufgerufen worden sein. Anderenfalls wirkt slsc_list_para_circle_2d_abs wie slsc_list_circle_2d_abs. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92.

Name der Funktion	slsc_list_para_circle_2d_abs
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument <code>Center</code>, siehe auch Seite 268) von slsc_list_para_circle_2d_abs. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_list_circle_2d_abs , slsc_list_para_arc_abs , slsc_list_para_disable , slsc_list_para_enable , slsc_list_para_jump_abs , slsc_list_para_mark_abs

Name der Funktion	slsc_list_para_dashed_arc_abs
Zweck	Wie slsc_list_para_arc_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Funktions-Signatur	<code>uint32_t slsc_list_para_dashed_arc_abs(size_t Handle, const double* Mid, const double* Target, size_t NSwitches, const double* LaserSwitches, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Mid Zeiger auf ein Array der Dimension 2. Koordinaten eines Punktes auf dem Kreisbogen zwischen dem Zielpunkt der letzten Funktion und Target . In mm.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	NSwitches NSwitches ist die Größe des LaserSwitches -Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double -Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. ParaTarget bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel -Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur" , Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei slsc_list_jump_abs – Der Kreisbogen ist durch 3 Punkte definiert: <ul style="list-style-type: none"> – Erster Punkt = Ziel der letzten Funktion. – Mid = Ein Punkt (auf dem Kreisbogen) zwischen dem ersten Punkt und Target. – Target = Zielpunkt.  Wie bei slsc_list_jump_abs – Eine gerade Linie wird markiert, wenn die 3 Punkte (fast) auf einer Geraden zu liegen kommen (kollinear sind).  Wie bei slsc_list_jump_abs – Bei der Ausführung von slsc_list_para_dashed_arc_abs wird der Laser angeschaltet und dann der Kreisbogen mit einer konstanten Geschwindigkeit abgefahren. <ul style="list-style-type: none"> – Ist die Markierung kleiner als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit ist die momentan eingestellte Markiergeschwindigkeit. – Ist die Markierung größer als etwa das halbe Scan-Kopf-Arbeitsfeld: die verwendete Geschwindigkeit wird auf eine passende Verfahrtsch-Geschwindigkeit vermindert. Diese ist in der Kennlinie definiert (siehe DynamicReductionFunction).

Name der Funktion	slsc_list_para_dashed_arc_abs
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • slsc_list_para_dashed_arc_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. • Jede slsc_list_[para/multi_para]*-Funktion wirkt wie ihre entsprechende slsc_list*-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. • Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. • slsc_cfg_set_rot_and_offset_2d und slsc_list_set_rot_and_offset_2d verändern die Zielpunkt-Koordinaten (Argument Target und Mid, siehe auch Seite 268) von slsc_list_para_dashed_arc_abs. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_list_para_arc_abs

Name der Funktion	slsc_list_para_dashed_circle_2d_abs
Zweck	Wie slsc_list_para_circle_2d_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Funktions-Signatur	<code>uint32_t slsc_list_para_dashed_circle_2d_abs(size_t Handle, const double* Center, double Angle, size_t NSwitches, const double* LaserSwitches, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Center Zeiger auf ein Array der Dimension 2. Koordinaten (x-Wert und y-Wert) des Kreismittelpunkts. In mm.
	Angle In rad. Positive Werte: Markierung erfolgt gegen den Uhrzeigersinn. Negative Werte: Markierung erfolgt im Uhrzeigersinn.
	NSwitches NSwitches ist die Größe des LaserSwitches-Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double-Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. ParaTarget bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel -Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur" , Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei slsc_list_circle_2d_abs - der Kreis ist durch 2 Punkte definiert: <ul style="list-style-type: none"> Erster Punkt = Ziel der letzten Markier-Funktion oder Sprung-Funktion. Center = Kreismittelpunkt.  Wie bei slsc_list_circle_2d_abs - bei der Ausführung von slsc_list_para_dashed_circle_2d_abs wird der Laser angeschaltet und dann der Kreis mit einer konstanten Geschwindigkeit abgefahren. Sowohl die Ausführungsrichtung der Markierung als auch die Anzahl der Umdrehungen (z.B. $3,25 \times 2\pi$) wird durch das Argument Angle bestimmt. slsc_list_para_dashed_circle_2d_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Jede slsc_list_[para/multi_para]*-Funktion wirkt wie ihre entsprechende slsc_list*-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.

Name der Funktion	<code>slsc_list_para_dashed_circle_2d_abs</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Vor <code>slsc_list_para_dashed_circle_2d_abs</code> muss <code>slsc_list_para_enable</code> aufgerufen worden sein. Anderenfalls wirkt <code>slsc_list_para_dashed_circle_2d_abs</code> wie <code>slsc_list_circle_2d_abs</code>. • Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (<code>slsc_list_[para/multi_para]*</code>-Funktionen)", Seite 92. • <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Center</code>, siehe auch Seite 268) von <code>slsc_list_para_dashed_circle_2d_abs</code>. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	<code>slsc_list_para_circle_2d_abs</code>

Name der Funktion	slsc_list_para_dashed_mark_abs
Zweck	Wie slsc_list_para_dashed_mark_abs , aber die korrespondierende [*]dashed[*] -Funktion. Bietet daher zusätzlich die Argumente NSwitches und LaserSwitches , siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
Funktions-Signatur	<code>uint32_t slsc_list_para_dashed_mark_abs(size_t Handle, const double* Target, size_t NSwitches, const double* LaserSwitches, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	NSwitches NSwitches ist die Größe des LaserSwitches -Array. Gibt vor, wie oft entlang des Markiermuster-Abschnitts der Laser ein/ausgeschaltet werden soll. Mindestwert: 1. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	LaserSwitches LaserSwitches ist ein Array von double -Werten. Das Array gibt an, bei welchen Bogenlänge-Werten (in mm) zwischen "Laser standby"-Betrieb und "Laser active"-Betrieb gewechselt wird. Siehe Abschnitt "[*]dashed[*]-Funktionen" , Seite 91.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. ParaTarget bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel -Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur" , Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei slsc_list_mark_abs – Das Verhalten von Markiervektoren wird in der Konfiguration der Trajektorienplanung festgelegt, siehe slsc_MarkConfig (z.B. MarkSpeed) und slsc_GeometryConfig (z.B. MaxBlendRadius). slsc_list_para_dashed_mark_abs zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Jede slsc_list_[para/multi_para]*-Funktion wirkt wie ihre entsprechende slsc_list*-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. Vor slsc_list_para_dashed_mark_abs muss slsc_list_para_enable aufgerufen worden sein. Anderenfalls wirkt slsc_list_para_dashed_mark_abs wie slsc_list_mark_abs.

Name der Funktion	<code>slsc_list_para_dashed_mark_abs</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. • <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code>, siehe auch Seite 268) von <code>slsc_list_para_dashed_mark_abs</code>. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung" ", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_list_para_dashed_mark_abs

Name der Funktion	<code>slsc_list_para_disable</code>
Zweck	Schaltet die Verarbeitung der Argumente <code>ParaTarget</code> (der <code>slsc_list_para</code> -Funktionen) und <code>MultiParaTarget</code> (der <code>slsc_list_multi_para</code> -Funktionen) aus.
Funktions-Signatur	<code>uint32_t slsc_list_para_disable(size_t Handle);</code>
Argument(e)	<code>Handle</code> Handle auf eine syncAXIS control -Instanz.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Nach dem Aufruf von <code>slsc_list_para_disable</code> wirken alle <code>slsc_list_[para/multi_para]</code>-Funktionen wie deren entsprechenden <code>slsc_list</code>-Funktionen. Alle nachfolgenden <code>ParaTarget</code>-Werte und <code>MultiParaTarget</code>-Werte werden auf 1 (Faktor Ip = 1) gesetzt (was für den/die Output-Wert/e der ActiveChannel "keine Änderung" bedeutet). Die Verarbeitung der Argumente <code>ParaTarget</code> (der <code>slsc_list_para</code>-Funktionen) und <code>MultiParaTarget</code> (der <code>slsc_list_multi_para</code>-Funktionen) wird mit <code>slsc_list_para_enable</code> wieder eingeschaltet. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (<code>slsc_list_[para/multi_para]</code>-Funktionen)", Seite 92. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	<code>slsc_list_para_enable</code>

Name der Funktion	slsc_list_para_enable
Zweck	Schaltet die Verarbeitung der Argumente <code>ParaTarget</code> (der slsc_list_para *-Funktionen) und <code>MultiParaTarget</code> (der slsc_list_multi_para *-Funktionen) ein.
Funktions-Signatur	<code>uint32_t slsc_list_para_enable(size_t Handle, const double* ParaTargetDefault);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	<code>ParaTargetDefault</code> Array der Dimension 2 oder 1, siehe Kommentar(e) . Start-Wert/e der ersten Rampe .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Die Dimensionierung des Arrays für den Rampen-Wert sollte der Anzahl der als aktiv definierten Kanäle entsprechen. In der <code>syncAXISConfig.xml</code> gibt es unter <code><cfg:Configuration></code> → <code><cfg:LaserConfig></code> → <code><cfg:AutomaticLaserControl></code> → <code><cfg:ActiveChannel></code> <ul style="list-style-type: none"> 2 × den Tag <code><cfg:Channel></code> = es sind 2 Kanäle als aktiv definiert: Zeiger auf ein Array der Dimension 2. 1 × den Tag <code><cfg:Channel></code> = es ist 1 Kanal als aktiv definiert: Zeiger auf ein Array der Dimension 1. 0 × den Tag <code><cfg:Channel></code> = es ist kein "ActiveChannel" definiert. <code>ParaTargetDefault</code> wird nicht ausgewertet! Wenn slsc_list_para_enable nicht aufgerufen wird, dann wirken alle slsc_list_[para/multi_para]*-Funktionen wie deren entsprechenden slsc_list_*-Funktionen. Gleiches gilt, wenn die Verarbeitung der Argumente <code>ParaTarget</code> (der slsc_list_para*-Funktionen) und <code>MultiParaTarget</code> (der slsc_list_multi_para*-Funktionen) mit slsc_list_para_disable ausgeschaltet wurde. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	slsc_list_para_disable

Name der Funktion	<code>slsc_list_para_jump_abs</code>
Zweck	Wie <code>slsc_list_jump_abs</code> . Bietet aber zusätzlich das Argument <code>ParaTarget</code> , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
Funktions-Signatur	<code>uint32_t slsc_list_para_jump_abs(size_t Handle, const double* Target, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. <code>ParaTarget</code> bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel-Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur" , Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei <code>slsc_list_jump_abs</code> – Das Verhalten von Sprüngen wird in der Konfiguration der Trajektorienplanung festgelegt, siehe <code>slsc_MarkConfig</code> (z.B. <code>LaserMinOffTime</code> und <code>JumpSpeed</code>). <code>slsc_list_para_jump_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Jede <code>slsc_list_[para/multi_para]*</code>-Funktion wirkt wie ihre entsprechende <code>slsc_list*</code>-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. Vor <code>slsc_list_para_jump_abs</code> muss <code>slsc_list_para_enable</code> aufgerufen worden sein. Anderenfalls wirkt <code>slsc_list_para_jump_abs</code> wie <code>slsc_list_jump_abs</code>. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code>, siehe auch Seite 268) von <code>slsc_list_para_jump_abs</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	<code>slsc_list_jump_abs</code> , <code>slsc_list_para_arc_abs</code> , <code>slsc_list_para_circle_2d_abs</code> , <code>slsc_list_para_disable</code> , <code>slsc_list_para_enable</code> , <code>slsc_list_para_mark_abs</code>

Name der Funktion	<code>slsc_list_para_jump_abs_min_time</code>
Zweck	Wie <code>slsc_list_jump_abs_min_time</code> . Bietet aber zusätzlich das Argument <code>ParaTarget</code> , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
Funktions-Signatur	<code>uint32_t slsc_list_para_jump_abs_min_time(size_t Handle, const double* Target, double MinimalJumpTime, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	MinimalJumpTime Minstdauer für den Sprung. In s. Zulässig sind nur positive Werte.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. <code>ParaTarget</code> bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel-Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur" , Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei <code>slsc_list_jump_abs_min_time</code> – Das Verhalten von Sprüngen wird in der Konfiguration der Trajektorienplanung festgelegt, siehe <code>slsc_MarkConfig</code> (z.B. <code>LaserMinOffTime</code> und <code>JumpSpeed</code>). <code>slsc_list_para_jump_abs_min_time</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Jede <code>slsc_list_[para/multi_para]*</code>-Funktion wirkt wie ihre entsprechende <code>slsc_list*</code>-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. Vor <code>slsc_list_para_jump_abs_min_time</code> muss <code>slsc_list_para_enable</code> aufgerufen worden sein. Anderenfalls wirkt <code>slsc_list_para_jump_abs_min_time</code> wie <code>slsc_list_jump_abs_min_time</code>. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (slsc_list_[para/multi_para]*-Funktionen)", Seite 92. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code>, siehe auch Seite 268) von <code>slsc_list_para_jump_abs_min_time</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0.
Verweise	<code>slsc_list_jump_abs_min_time</code>

Name der Funktion	<code>slsc_list_para_mark_abs</code>
Zweck	Wie <code>slsc_list_mark_abs</code> . Bietet aber zusätzlich das Argument <code>ParaTarget</code> , über das eine Rampe definiert (der/die Wert/e eines/zwei "ActiveChannel" wird im Arbeitsfeld linear variiert) wird.
Funktions-Signatur	<code>uint32_t slsc_list_para_mark_abs(size_t Handle, const double* Target, const double* ParaTarget);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	Target Zeiger auf ein Array der Dimension 2. Koordinaten des Zielpunkts. In mm.
	ParaTarget Array der Dimension 2 oder 1, siehe Kommentar(e) , Seite 250. <code>ParaTarget</code> bezieht sich auf das Ende der nachfolgenden Rampe . Wird als Faktor Ip zur Berechnung der ActiveChannel-Werte verwendet, siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur", Seite 51.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Wie bei <code>slsc_list_mark_abs</code> – Das Verhalten von Markiervektoren wird in der Konfiguration der Trajektorienplanung festgelegt, siehe <code>slsc_MarkConfig</code> (z.B. <code>MarkSpeed</code>) und <code>slsc_GeometryConfig</code> (z.B. <code>MaxBlendRadius</code>). <code>slsc_list_para_mark_abs</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Jede <code>slsc_list_[para/multi_para]*</code>-Funktion wirkt wie ihre entsprechende <code>slsc_list*</code>-Funktion, wenn kein "ActiveChannel" eingetragen ist, siehe Abschnitt "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. Vor <code>slsc_list_para_mark_abs</code> muss <code>slsc_list_para_enable</code> aufgerufen worden sein. Anderenfalls wirkt <code>slsc_list_para_mark_abs</code> wie <code>slsc_list_mark_abs</code>. Siehe auch Abschnitt "Funktionen zum Definieren von Rampen (<code>slsc_list_[para/multi_para]*</code>-Funktionen)", Seite 92. <code>slsc_cfg_set_rot_and_offset_2d</code> und <code>slsc_list_set_rot_and_offset_2d</code> verändern die Zielpunkt-Koordinaten (Argument <code>Target</code>, siehe auch Seite 268) von <code>slsc_list_para_mark_abs</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	<code>slsc_list_mark_abs</code> , <code>slsc_list_para_arc_abs</code> , <code>slsc_list_para_circle_2d_abs</code> , <code>slsc_list_para_disable</code> , <code>slsc_list_para_enable</code> , <code>slsc_list_para_jump_abs</code>

Name der Funktion	<code>slsc_list_para_playback_module</code>
Zweck	Wie <code>slsc_list_playback_module</code> , aber es werden Parameter-Werte zu Rampen angewendet, wenn ein <code>slsc_list_para_enable</code> vorausgeht.
Funktions-Signatur	<code>uint32_t slsc_list_para_playback_module(size_t Handle, const char* ModuleFileName);</code>
Argument(e)	Handle Wie <code>slsc_list_playback_module</code> .
	ModuleFileName Wie <code>slsc_list_playback_module</code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • <code>slsc_list_para_playback_module</code> verhält sich wie <code>slsc_list_playback_module</code>, wenn vorher nicht <code>slsc_list_para_enable</code> aufgerufen wurde. • Bezogen auf die Parameter-Werte zu Rampen verhält sich <code>slsc_list_para_playback_module</code> überwiegend wie andere <code>slsc_list_para_*/slsc_list_multi_para_*</code>-Funktionen. Die wesentliche Ausnahme ist, dass diese Werte zur Abspielzeit nicht neu definiert werden können. • Daher starten beim Abspielen des Moduls diese Werte mit den aufgezeichneten Werten und <i>nicht</i> an den Endwerten der vorausgehenden Funktion oder mit dem in <code>slsc_list_para_enable</code> angegebenen <code>ParaTargetDefault</code>-Wert. • Achten Sie bei der Verwendung von <code>slsc_list_para_playback_module</code> darauf, dass die ActiveChannel-Konfiguration in der <code>syncAXISConfig.xml</code> (siehe Kapitel 2.9.2 "Definition der Channel und ActiveChannel", Seite 48) bezüglich Aufzeichnungszeit und Abspielzeit übereinstimmt. Siehe Kommentar auf Seite 256. • Siehe Kapitel 2.11 "Über das Arbeiten mit "Modulen"", Seite 65.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0.
Verweise	<code>slsc_list_playback_module</code>

Name der Funktion	<code>slsc_list_playback_module</code>
Zweck	Integriert ("spielt ab") ein Modul in den aktuellen Job . Parameter-Werte zu Rampen werden nicht angewendet.
Funktions-Signatur	<code>uint32_t slsc_list_playback_module(size_t Handle, const char* ModuleFileName);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	ModuleFileName Absoluter Dateipfad der einzulesenden Modul -Datei (*.slm). Zeiger auf einen \0-terminierten ANSI-String, 1 Byte pro Zeichen.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_list_playback_module ist im <i>Simulationsmodus</i> und (anders als slsc_list_begin_module auch) im Hardwaremodus erlaubt. • slsc_list_playback_module ist in erlaubt im Betriebsmodus "ScannerOnly", "StageOnly", "ScannerAndStage". • Für slsc_list_playback_module gibt es keine entsprechende Konfigurations-Funktion (slsc_cfg_*). • Sollen die in der Modul-Datei aufgezeichneten Parameter-Werte zu Rampen angewendet werden, ist slsc_list_para_playback_module zu verwenden. • Siehe auch Abschnitt "Funktionen für "Module"", Seite 95. • In den folgenden Fällen wird slsc_list_playback_module abgelehnt und ein Bit beim Rückgabewert gesetzt: <ul style="list-style-type: none"> – Die angegebene Modul-Datei kann nicht geöffnet werden Bit #06 (UnplausibleOrUnknownParameter) – Versionsnummer in der Modul-Datei ist inkompatibel Bit #06 (UnplausibleOrUnknownParameter) – Der aufzuzeichnende Job wurde nicht vollständig in die Modul-Datei geschrieben Bit #06 (UnplausibleOrUnknownParameter) – Der Betriebsmodus ist inkompatibel Bit #09 (NotAllowedInCurrentConfiguration) <ul style="list-style-type: none"> • Ein ScannerOnly (ScannerAndStage) aufgezeichnetes Modul darf <i>nicht</i> im StageOnly Betriebsmodus abgespielt werden. • Ein StageOnly aufgezeichnetes Modul darf nur im StageOnly Betriebsmodus abgespielt werden. – JumpSpeed oder MarkSpeed im Modul sind größer als das Geschwindigkeitslimit zur Abspielzeit (Bit #09 (NotAllowedInCurrentConfiguration)). – Beschleunigungslimit oder Rucklimit im Moduls ist sind größer als das entsprechende Limit zur Abspielzeit (Bit #09 (NotAllowedInCurrentConfiguration))

Name der Funktion	slsc_list_playback_module
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> In den folgenden Fällen wird lediglich eine neue [WARN]-Log-Dateizeile (siehe [WARN]-Log-Dateizeilen) erzeugt: <ul style="list-style-type: none"> Die niedrigste Sprungzeit im Modul ist kleiner als die LaserPreTriggerTime zur Abspielzeit. Einige Laserschaltzeitpunkte könnten nicht wie erwartet gesetzt werden. Die Delay-Werte für Scan-Device und Verfahrtsch im Modul sind anders als die zur Abspielzeit. Bei kleineren Delay-Werte zur Abspielzeit könnte es sein, dass einige "SIGNAL"-Funktionen ignoriert werden, für die negative TimeDelay-Werte angegeben sind. Das Modul wurde im Betriebsmodus ScannerOnly (ScannerAndStage) aufgezeichnet und wird im Betriebsmodus ScannerAndStage (ScannerOnly) abgespielt. Nur für slsc_list_para_playback_module relevant: bei Abspielzeit unterschiedliche ActiveChannel-Konfiguration im Modul als zur Aufzeichnungszeit Zu Übergängen ("Ränder" des Moduls) in die Modul-Trajektorie: <ul style="list-style-type: none"> Kann der Anschluss nicht direkt ausgeführt werden, erfolgt immer eine Sky Writing-ähnliche Bewegung (kein Blending). Beginnt (endet) ein Modul mit einem Sprung, so wird dieser nicht mit zur Abspielzeit anschließenden Sprüngen zusammengefasst, sondern beide Sprünge werden getrennt ausgeführt. Siehe Kapitel 2.11 "Über das Arbeiten mit "Modulen"", Seite 65. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	Siehe Abbildung 29 , Seite 69.
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0 .
Verweise	slsc_list_begin_module , slsc_list_para_playback_module

Name der Funktion	<code>slsc_list_set_approx_blend_limit</code>
Zweck	Ändert den <code>ApproxBlendLimit</code> -Wert, der in der Konfiguration der Trajektorienplanung festgelegt ist (s.u.). Diese Änderung gilt für alle nachfolgenden Job-Funktionen (<code>slsc_list_*</code>), aber nur bis zum Ende des Jobs .
Funktions-Signatur	<code>uint32_t slsc_list_set_approx_blend_limit(size_t Handle, double ApproxBlendLimit);</code>
Argument(e)	<code>Handle</code> Handle auf eine syncAXIS control-Instanz .
	<code>ApproxBlendLimit</code> ApproxBlendLimit-Wert . Siehe auch slsc_GeometryConfig .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> slsc_list_set_approx_blend_limit ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle, aber nur bis zum Ende des gerade laufenden Jobs. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	–

Name der Funktion	<code>slsc_list_set_calculation_dynamics_jump_scan_device</code>
Zweck	<p>Ändert:</p> <ul style="list-style-type: none"> Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Sprünge, aber nicht Markierungen <p>Diese Änderung gilt für alle nachfolgenden Job-Funktionen (<code>slsc_list_*</code>), aber nur bis zum Ende des Jobs.</p>
Funktions-Signatur	<code>uint32_t slsc_list_set_calculation_dynamics_jump_scan_device(size_t Handle, double JumpAngularAcc, double JumpAngularJerk);</code>
Argument(e)	Handle Handle auf eine syncAXIS control -Instanz.
	JumpAngularAcc Wie <code>JumpAngularAcc</code> .
	JumpAngularJerk Wie <code>JumpAngularJerk</code> .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> ⚠ Vorsicht! syncAXIS control verwendet den Wert von <code>Acceleration = JumpAngularAcc = JumpAngularAcc</code> zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. ⚠ Vorsicht! syncAXIS control verwendet den Wert von <code>Jerk = JumpAngularJerk = JumpAngularJerk</code> zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. <code>slsc_list_set_calculation_dynamics_jump_scan_device</code> ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle nur bis zum Ende des gerade laufenden Jobs. Die entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>) von <code>slsc_list_set_calculation_dynamics_jump_scan_device</code> ist <code>slsc_cfg_set_calculation_dynamics_jump_scan_device</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0 .
Verweise	<code>slsc_cfg_get_calculation_dynamics_jump_scan_device</code> , <code>slsc_cfg_set_calculation_dynamics_jump_scan_device</code>

Name der Funktion	slsc_list_set_calculation_dynamics_mark_scan_device
Zweck	<p>Ändert:</p> <ul style="list-style-type: none"> Den Höchstwert von Beschleunigung und Ruck des vorgesehenen Scan-Device-Typs. Die Werte werden nur in Trajektorienplanungs-Berechnungen der Scan-Device-Bewegung verwendet – allerdings nur für Markierungen, aber nicht Sprünge <p>Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.</p>
Funktions-Signatur	uint32_t slsc_list_set_calculation_dynamics_mark_scan_device(size_t Handle, double MarkAngularAcc, double MarkAngularJerk);
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	MarkAngularAcc Wie MarkAngularAcc.
	MarkAngularJerk Wie MarkAngularJerk.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> ⚠ Vorsicht! syncAXIS control verwendet den Wert von Acceleration = MarkAngularAcc = MarkAngularAcc zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. ⚠ Vorsicht! syncAXIS control verwendet den Wert von Jerk = MarkAngularJerk = MarkAngularJerk zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. slsc_list_set_calculation_dynamics_mark_scan_device ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle nur bis zum Ende des gerade laufenden Jobs. Die entsprechende Konfigurations-Funktion (slsc_cfg_*) von slsc_list_set_calculation_dynamics_mark_scan_device ist slsc_cfg_set_calculation_dynamics_mark_scan_device. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.6.0.
Verweise	slsc_cfg_get_calculation_dynamics_mark_scan_device, slsc_cfg_set_calculation_dynamics_mark_scan_device

Name der Funktion	<code>slsc_list_set_contour_dependent_speed_control_2d</code>						
Zweck	<p>„Konturabhängige Geschwindigkeitsberechnung“ an-/ausschalten. Außerdem kann geändert werden, wie (die syncAXIS control-Instanz intern) die Geschwindigkeiten entlang von Kurven bestimmt („links“ oder „rechts“ der Kurvenlinien-Mitte; Abstand zu dieser). Bei aktivierter „Automatische Lasersteuerung“ werden diese Ergebnisse dazu verwendet, um entsprechend dazu z.B. die Laser-Spot-Abstände äquidistant zu setzen.</p> <p>Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.</p>						
Funktions-Signatur	<code>uint32_t slsc_list_set_contour_dependent_speed_control_2d(size_t Handle, int32_t Direction, double SpotRadius);</code>						
Argument(e)	<table border="1"> <tr> <td>Handle</td><td>Handle auf eine syncAXIS control-Instanz.</td></tr> <tr> <td>Direction</td><td> <p>0: „Konturabhängige Geschwindigkeitsberechnung“ = Aus. Die Geschwindigkeiten werden auf der Kurvenlinien-Mitte bestimmt. Ist auch der Default-Zustand nach der Initialisierung der syncAXIS control-Instanz mit slsc_cfg_initialize_from_file.</p> <p>+1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden rechts der Kurvenlinien-Mitte bestimmt.</p> <p>-1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden links der Kurvenlinien-Mitte bestimmt.</p> </td></tr> <tr> <td>SpotRadius</td><td> <p>Radius des Laserspots in der Arbeitsebene. In mm. Der Wert legt fest, wie weit rechts oder links von der Kurvenlinien-Mitte entfernt die Geschwindigkeitswerte bestimmt werden.</p> </td></tr> </table>	Handle	Handle auf eine syncAXIS control-Instanz .	Direction	<p>0: „Konturabhängige Geschwindigkeitsberechnung“ = Aus. Die Geschwindigkeiten werden auf der Kurvenlinien-Mitte bestimmt. Ist auch der Default-Zustand nach der Initialisierung der syncAXIS control-Instanz mit slsc_cfg_initialize_from_file.</p> <p>+1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden rechts der Kurvenlinien-Mitte bestimmt.</p> <p>-1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden links der Kurvenlinien-Mitte bestimmt.</p>	SpotRadius	<p>Radius des Laserspots in der Arbeitsebene. In mm. Der Wert legt fest, wie weit rechts oder links von der Kurvenlinien-Mitte entfernt die Geschwindigkeitswerte bestimmt werden.</p>
Handle	Handle auf eine syncAXIS control-Instanz .						
Direction	<p>0: „Konturabhängige Geschwindigkeitsberechnung“ = Aus. Die Geschwindigkeiten werden auf der Kurvenlinien-Mitte bestimmt. Ist auch der Default-Zustand nach der Initialisierung der syncAXIS control-Instanz mit slsc_cfg_initialize_from_file.</p> <p>+1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden rechts der Kurvenlinien-Mitte bestimmt.</p> <p>-1: „Konturabhängige Geschwindigkeitsberechnung“ = An. Die Geschwindigkeiten werden links der Kurvenlinien-Mitte bestimmt.</p>						
SpotRadius	<p>Radius des Laserspots in der Arbeitsebene. In mm. Der Wert legt fest, wie weit rechts oder links von der Kurvenlinien-Mitte entfernt die Geschwindigkeitswerte bestimmt werden.</p>						
Rückgabewert	Siehe Kapitel 4 „Standard-Rückgabewerte der syncAXIS-DLL Funktionen“ , Seite 279.						
Kommentar(e)	<ul style="list-style-type: none"> slsc_list_set_contour_dependent_speed_control_2d ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle, aber nur bis zum Ende des gerade laufenden Jobs. slsc_list_set_contour_dependent_speed_control_2d hat keine Wirkung (es wird kein Fehler zurückgegeben), wenn die „Automatische Lasersteuerung“ nicht eingeschaltet ist (z. B. es ist kein ActiveChannel in der syncAXISConfig.xml eingetragen). Siehe auch Kapitel 2.9.5 „Über die „Konturabhängige Geschwindigkeitsberechnung““, Seite 60. Die entsprechende Konfigurations-Funktion (slsc_cfg_*) von slsc_list_set_contour_dependent_speed_control_2d ist slsc_cfg_set_contour_dependent_speed_control_2d. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus „Manuelle Positionierung“ siehe Kapitel 2.12 „Über den Modus „Manuelle Positionierung““, Seite 70. 						
Code-Beispiel	–						
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0 .						
Verweise	slsc_cfg_set_contour_dependent_speed_control_2d , slsc_list_set_laser_on_move						

Name der Funktion	<code>slsc_list_set_free_variable</code>
Zweck	Wie <code>slsc_ctrl_set_free_variable</code> .
Funktions-Signatur	<code>uint32_t slsc_list_set_free_variable(size_t Handle, uint32_t Number, uint32_t Value, double TimeDelay);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Number Nummer derjenigen freien Variablen, deren Wert auf der RTC6 gesetzt werden soll. Zulässiger Wertebereich: [0...7]. Es werden nur die drei niederwertigsten Bits ausgewertet.
	Value Wert der freien Variable, der auf der RTC6 gesetzt werden soll.
	TimeDelay Relativer Zeitpunkt zwischen 2 Job-Funktionen (<code>slsc_list_*</code>), an dem die Änderung angewendet wird (Bezugspunkt: Zeitpunkt bei der <i>Markierungsausführung</i> , an dem der Zielpunkt der 1. Job-Funktion erreicht wird; siehe Abbildung 14, Seite 46). In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Im Simulationsmodus haben <code>slsc_ctrl_get_free_variable</code>, <code>slsc_ctrl_set_free_variable</code>, sowie <code>slsc_list_set_free_variable</code> keine Auswirkung. Bei überhöhten <code>TimeDelay</code>-Werten wird die Änderung spätestens bei <code>slsc_list_end</code> angewendet. Die Änderung <ul style="list-style-type: none"> ist dauerhaft gilt daher für alle nachfolgenden Jobs <code>slsc_list_set_free_variable</code> zählt zu den "SIGNAL"-Funktionen. Siehe auch Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. <code>slsc_list_set_free_variable</code> ist eine direkte Implementierung des RTC6-Befehls <code>set_free_variable_list</code> in <code>syncAXIS control</code>. <code>set_free_variable_list</code> bietet den Konfigurationsparameter <code>TimeDelay</code> jedoch nicht. Mit den Funktionen für freie Variablen (<code>slsc_ctrl_set_free_variable</code>, <code>slsc_list_set_free_variable</code> und <code>slsc_ctrl_get_free_variable</code>) können z.B. Inkremente (innerhalb von Jobs) festgestellt und gezählt werden. Weitere Informationen zu freien Variablen finden Sie im RTC6-Handbuch, Kapitel 6.9.1 "Freie Variablen", Seite 134. Der aktuelle Wert einer freien Variablen kann mit <code>slsc_ctrl_get_free_variable</code> abgefragt werden.

Name der Funktion	<code>slsc_list_set_free_variable</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Die entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) von <code>slsc_list_set_free_variable</code> ist <code>slsc_ctrl_set_free_variable</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.2. Letzte Änderung mit syncAXIS-DLL V1.2.4: <code>TimeDelay</code> -Werte für V1.1 sind in V1.2 syncAXIS-DLL-intern um ein Delay verschoben (z. B. im Betriebsmodus "ScannerOnly" um 0,00125 s).
Verweise	<code>slsc_ctrl_get_free_variable</code> , <code>slsc_ctrl_set_free_variable</code>

Name der Funktion	<code>slsc_list_set_jump_speed</code>
Zweck	Ändert die Sprunggeschwindigkeit. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (<code>slsc_list_*</code>), aber nur bis zum Ende des Jobs.
Funktions-Signatur	<code>uint32_t slsc_list_set_jump_speed(size_t Handle, double JumpSpeed);</code>
Argument(e)	<code>Handle</code> Handle auf eine syncAXIS control-Instanz.
	<code>JumpSpeed</code> Sprunggeschwindigkeit. In mm/s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_list_set_jump_speed</code> ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle (aber anders als beim ähnlichen RTC-Befehl), aber nur bis zum Ende des gerade laufenden Jobs. Die entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>) von <code>slsc_list_set_jump_speed</code> ist <code>slsc_cfg_set_jump_speed</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	<code>slsc_cfg_set_jump_speed</code>

Name der Funktion	slsc_list_set_laser_on_move
Zweck	Verzögert den "Laser active"-Betrieb um genau die Zeitdauer, die gebraucht wird, um die angegebene Pfadlänge (PathLength) auf dem aktuellen Markierabschnitt abzufahren. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.
Funktions-Signatur	uint32_t slsc_list_set_laser_on_move(size_t Handle, double PathLength;
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	PathLength Pfadlänge auf dem Markierabschnitt, nach der der Laser tatsächlich eingeschaltet werden soll. In mm. Zulässige Werte: ≥ 0 .
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_list_set_laser_on_move ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle, aber nur bis zum Ende des gerade laufenden Jobs. • Für einen PathLength-Wert < 0 ist beim Rückgabewert Bit #06 gesetzt (UnplausibleOrUnknownParameter). • Anwendungsfall für slsc_list_set_laser_on_move: siehe Abschnitt "Über slsc_list_set_laser_on_move", Seite 88. • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.
Verweise	slsc_list_set_contour_dependent_speed_control_2d

Name der Funktion	<code>slsc_list_set_laser_pulses</code>
Zweck	Wie <code>slsc_ctrl_set_laser_pulses</code> .
Funktions-Signatur	<code>uint32_t slsc_list_set_laser_pulses(size_t Handle, double HalfPeriod, double PulseLength, double TimeDelay);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
	HalfPeriod Wie <code>slsc_ctrl_set_laser_pulses</code> .
	PulseLength Wie <code>slsc_ctrl_set_laser_pulses</code> .
	TimeDelay Relativer Zeitpunkt zwischen 2 Job-Funktionen (<code>slsc_list_*</code>), an dem die Änderung angewendet wird (Bezugspunkt: Zeitpunkt bei der <i>Markierungsausführung</i> , an dem der Zielpunkt der 1. Job-Funktion erreicht wird; siehe <i>Abbildung 14, Seite 46</i>). In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe <i>Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279</i> .
Kommentar(e)	<ul style="list-style-type: none"> Im Simulationsmodus haben <code>slsc_ctrl_set_laser_pulses</code> und <code>slsc_list_set_laser_pulses</code> keine Auswirkung. <code>slsc_list_set_laser_pulses</code> ändert die Konfiguration der angegebenen <code>syncAXIS control</code>-Instanz. Die <code>syncAXIS control</code>-Instanz wird dabei nicht reinitialisiert. <code>HalfPeriod</code> und <code>PulseLength</code> ändern, was bei der Initialisierung (mit den gleichnamigen Attributen im <code>syncAXISConfig.xml</code>-Tag <code><cfg:LaserOutput Unit="s" HalfPeriod="..." PulseLength="..." /></code>) gesetzt wurde. <code>slsc_ctrl_set_laser_pulses</code> und <code>slsc_list_set_laser_pulses</code> werden für diejenigen Benutzer bereitgestellt, die (aufgrund ihres eingesetzten Lasers) die "Automatische Lasersteuerung" zum Erreichen äquidistanter Spotabstände nicht nutzen können und stattdessen die Pulsausgabe über <code>HalfPeriod</code> und <code>PulseLength</code> beeinflussen wollen. Ist die "Automatische Lasersteuerung" aktiv mit <code>SpotDistance</code> als "ActiveChannel", siehe <i>Kapitel 2.9.2 "Definition der Channel und ActiveChannel", Seite 48</i>, dann: <ul style="list-style-type: none"> wirkt <code>HalfPeriod</code> nicht wirkt <code>PulseLength</code> (d.h. die Pulslängen von Lasersignal LASER1 und LASER2 werden geändert) Bei überhöhten <code>TimeDelay</code>-Werten wird die Änderung spätestens bei <code>slsc_list_end</code> angewendet. Die Änderung <ul style="list-style-type: none"> ist dauerhaft gilt daher für alle nachfolgenden <i>Jobs</i> Verwandter RTC6-Befehl: <code>set_laser_pulses</code>. Dieser bietet den Konfigurationsparameter <code>TimeDelay</code> jedoch nicht. <code>slsc_list_set_laser_pulses</code> zählt zu den "SIGNAL"-Funktionen. Siehe auch <i>Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45</i>.

Name der Funktion	<code>slsc_list_set_laser_pulses</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Die entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) von <code>slsc_list_set_laser_pulses</code> ist <code>slsc_ctrl_set_laser_pulses</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.4.
Verweise	<code>slsc_ctrl_set_laser_pulses</code>

Name der Funktion	<code>slsc_list_set_mark_speed</code>
Zweck	Ändert die Markiergeschwindigkeit. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (<code>slsc_list_*</code>), aber nur bis zum Ende des Jobs.
Funktions-Signatur	<code>uint32_t slsc_list_set_mark_speed(size_t Handle, double MarkSpeed);</code>
Argument(e)	Handle <code>Handle</code> auf eine syncAXIS control-Instanz.
	MarkSpeed <code>MarkSpeed</code> Markiergeschwindigkeit. In mm/s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_list_set_mark_speed</code> ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle (aber anders als beim ähnlichen RTC-Befehl), aber nur bis zum Ende des gerade laufenden Jobs. Die entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>) von <code>slsc_list_set_mark_speed</code> ist <code>slsc_cfg_set_mark_speed</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0.
Verweise	<code>slsc_cfg_set_mark_speed</code>

Name der Funktion	<code>slsc_list_set_matrix_and_offset</code>
Zweck	Ändert Zielpunkt-Koordinaten entsprechend einer Transformationsmatrix und einem Offset-Wert. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (<code>slsc_list_*</code>), aber nur bis zum Ende des Jobs .
Funktions-Signatur	<code>uint32_t slsc_list_set_matrix_and_offset(size_t Handle, const double* Matrix, const double* Offset);</code>
Argument(e)	<p>Handle Handle auf eine syncAXIS control-Instanz.</p> <p>Matrix Zeiger auf ein Array der Dimension 4. Koeffizienten $m_{11}...m_{22}$ einer (2×2)-Transformationsmatrix.</p> <p>Offset Zeiger auf ein Array der Dimension 2. x-Wert und y-Wert, um den die Zielpunkte im Arbeitsfeld verschoben werden. In mm.</p>
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_list_set_matrix_and_offset</code> ändert (wie z.B. <code>slsc_list_set_rot_and_offset_2d</code>) die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügung, aber nur bis zum Ende des gerade laufenden Jobs. Zielpunkt-Koordinaten von Job-Funktionen (<code>slsc_list_*</code>): siehe Listenpunkt auf Seite 268. <code>slsc_list_set_matrix_and_offset</code> berechnet (wie <code>slsc_cfg_set_matrix_and_offset</code>) die neuen Zielpunkte gemäß $(\text{Transformationsmatrix} \times \text{Zielpunkt}) + \text{Offset}$: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ Mit geeigneten Transformationsmatrix-Koeffizienten (Argument <code>Matrix</code>) kann z.B. ein Skalieren, Drehen oder Umklappen von Markiermustern erzielt werden. Siehe auch Abschnitt "Funktionen zum Ändern der Zielpunkt-Koordinaten", Seite 92. Die entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>) von <code>slsc_list_set_matrix_and_offset</code> ist <code>slsc_cfg_set_matrix_and_offset</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.1.0.
Verweise	<code>slsc_cfg_set_matrix_and_offset</code>

Name der Funktion	<code>slsc_list_set_min_mark_speed</code>
Zweck	Ändert die minimale Markiergeschwindigkeit, siehe <code>MinimalMarkSpeed</code> . Diese Änderung gilt für alle nachfolgenden Job-Funktionen (<code>slsc_list_*</code>), aber nur bis zum Ende des Jobs .
Funktions-Signatur	<code>uint32_t slsc_list_set_min_mark_speed(size_t Handle, double MinimalMarkSpeed);</code>
Argument(e)	<code>Handle</code> Handle auf eine syncAXIS control-Instanz .
	<code>MinimalMarkSpeed</code> Minimale Markiergeschwindigkeit. In mm/s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • <code>slsc_list_set_min_mark_speed</code> ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle (aber anders als beim ähnlichen RTC-Befehl), aber nur bis zum Ende des gerade laufenden Jobs. • Für <code>slsc_list_set_min_mark_speed</code> gibt es keine entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>). • Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.6.
Verweise	–

Name der Funktion	slsc_list_set_rot_and_offset_2d	
Zweck	Ändert Zielpunkt-Koordinaten um einen Winkel und Offset-Wert. Diese Änderung gilt für alle nachfolgenden Job-Funktionen (slsc_list_*), aber nur bis zum Ende des Jobs.	
Funktions-Signatur	uint32_t slsc_list_set_rot_and_offset_2d(size_t Handle, double Angle, const double* Offset);	
Argument(e)	Handle	Handle auf eine syncAXIS control-Instanz.
	Angle	Winkel (um den Ursprung 0,0), um den die Zielpunkte im Arbeitsfeld gedreht werden. In rad. Positive Werte: Drehung erfolgt gegen den Uhrzeigersinn. Negative Werte: Drehung erfolgt im Uhrzeigersinn.
	Offset	Zeiger auf ein Array der Dimension 2. x-Wert und y-Wert, um den die Zielpunkte im Arbeitsfeld verschoben werden. In mm.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.	
Kommentar(e)	<ul style="list-style-type: none"> • slsc_list_set_rot_and_offset_2d ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Die Änderung gilt ab der Einfügestelle, aber nur bis zum Ende des gerade laufenden Jobs. • Zielpunkt-Koordinaten von Job-Funktionen (slsc_list_*) sind: <ul style="list-style-type: none"> – Mid und Target von slsc_list_arc_abs – Mid und Target von slsc_list_dashed_arc_abs – Mid und Target von slsc_list_multi_para_arc_abs – Mid und Target von slsc_list_multi_para_dashed_arc_abs – Mid und Target von slsc_list_para_arc_abs – Mid und Target von slsc_list_para_dashed_arc_abs – Center von slsc_list_circle_2d_abs – Center von slsc_list_dashed_circle_2d_abs – Center von slsc_list_multi_para_circle_2d_abs – Center von slsc_list_multi_para_dashed_circle_2d_abs – Center von slsc_list_para_circle_2d_abs – Center von slsc_list_para_dashed_circle_2d_abs – Target von slsc_list_dashed_mark_abs – Target von slsc_list_jump_abs – Target von slsc_list_jump_abs_min_time – Target von slsc_list_mark_abs – Target von slsc_list_multi_para_dashed_mark_abs – Target von slsc_list_multi_para_mark_abs – Target von slsc_list_para_dashed_mark_abs – Target von slsc_list_para_jump_abs – Target von slsc_list_para_jump_abs_min_time – Target von slsc_list_para_mark_abs 	

Name der Funktion	<code>slsc_list_set_rot_and_offset_2d</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> <code>slsc_list_set_rot_and_offset_2d</code> berechnet die neuen Zielpunkte gemäß: (Rotationsmatrix$_{\alpha}$ \times Zielpunkt) + Offset, also konkret: <ul style="list-style-type: none"> – neuer Zielpunkt x-Wert = $((x \times \cos \alpha) - (y \times \sin \alpha)) + \text{x-Offset-Wert}$ – neuer Zielpunkt y-Wert = $((x \times \sin \alpha) + (y \times \cos \alpha)) + \text{y-Offset-Wert}$ – Beispiel: <div data-bbox="715 629 1332 1090" data-label="Figure"> <p> $P1 = \text{Zielpunkt z.B. von } \text{slsc_list_jump_abs}$ Offset ($x=0, y=-2$) $\alpha = \pi/2$ $R_{\alpha} = \text{Rotationsmatrix}_{\alpha} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}$ </p> </div> <ul style="list-style-type: none"> Die entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>) von <code>slsc_list_set_rot_and_offset_2d</code> ist <code>slsc_cfg_set_rot_and_offset_2d</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70. Ein Modul kann mit <code>slsc_list_set_rot_and_offset_2d</code> beliebig im Raum positioniert und gedreht werden.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.
Verweise	<code>slsc_cfg_set_rot_and_offset_2d</code> , <code>slsc_list_arc_abs</code> , <code>slsc_list_circle_2d_abs</code> , <code>slsc_list_jump_abs</code> , <code>slsc_list_mark_abs</code>

Name der Funktion	<code>slsc_list_suppress_spotdistance_control</code>
Zweck	Nur wenn die "Automatische Lasersteuerung" aktiv ist mit <code>SpotDistance</code> als "ActiveChannel": Zusatzfunktion, die <code>slsc_list_wait_with_laser_on</code> vorausgehen muss.
Funktions-Signatur	<code>uint32_t slsc_list_suppress_spotdistance_control(size_t Handle, double TimeDelay);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
	TimeDelay Relativer Zeitpunkt zwischen 2 Job-Funktionen (<code>slsc_list_*</code>), an dem die Änderung angewendet wird (Bezugspunkt: Zeitpunkt bei der <i>Markierungsausführung</i> , an dem der Zielpunkt der 1. Job-Funktion erreicht wird; siehe Abbildung 14, Seite 46). In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Ist die "Automatische Lasersteuerung" aktiv mit <code>SpotDistance</code> als "ActiveChannel", siehe Kapitel 2.9.2 "Definition der Channel und ActiveChannel", Seite 48, dann muss <code>slsc_list_suppress_spotdistance_control</code> vor <code>slsc_list_wait_with_laser_on</code> aufgerufen werden. Weitere Details siehe auch Abschnitt "Sonderfall: SpotDistance als "ActiveChannel"", Seite 87. <code>slsc_list_suppress_spotdistance_control</code> und <code>slsc_list_unsuppress_spotdistance_control</code> setzen voraus, dass <code>SpotDistance</code> als "ActiveChannel" eingestellt ist. Sonst ist beim Rückgabewert Bit #09 gesetzt (<code>NotAllowedInCurrentConfiguration</code>). <code>slsc_list_suppress_spotdistance_control</code> unterdrückt syncAXIS-DLL-intern die Funktionalität zur Erzeugung äquidistanter Spotabstände. Wenn diese Funktionalität bereits unterdrückt ist, dann hat <code>slsc_list_suppress_spotdistance_control</code> keine Wirkung. Bei überhöhten <code>TimeDelay</code>-Werten wird die Änderung spätestens bei <code>slsc_list_end</code> angewendet. Die Änderung <ul style="list-style-type: none"> ist dauerhaft gilt daher für alle nachfolgenden Jobs <code>slsc_list_suppress_spotdistance_control</code> zählt zu den "SIGNAL"-Funktionen. Siehe auch Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die komplementäre Funktion von <code>slsc_list_suppress_spotdistance_control</code> ist <code>slsc_list_unsuppress_spotdistance_control</code>. Für <code>slsc_list_suppress_spotdistance_control</code> und <code>slsc_list_unsuppress_spotdistance_control</code> gibt es weder eine entsprechende Konfigurations-Funktion (<code>slsc_cfg_*</code>) noch Kontroll-Funktion (<code>slsc_ctrl_*</code>).

Name der Funktion	<code>slsc_list_suppress_spotdistance_control</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> <code>slsc_list_suppress_spotdistance_control</code> und <code>slsc_list_unsuppress_spotdistance_control</code> sind in allen Betriebsmodi (siehe enum <code>slsc_OperationMode</code>) erlaubt. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.7.
Verweise	<code>slsc_list_unsuppress_spotdistance_control</code> , <code>slsc_list_wait_with_laser_on</code>

Name der Funktion	<code>slsc_list_unsuppress_spotdistance_control</code>
Zweck	Hebt die Wirkung von <code>slsc_list_suppress_spotdistance_control</code> wieder auf.
Funktions-Signatur	<code>uint32_t slsc_list_unsuppress_spotdistance_control(size_t Handle, double TimeDelay);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz.
	TimeDelay Relativer Zeitpunkt zwischen 2 Job-Funktionen (<code>slsc_list_*</code>), an dem die Änderung angewendet wird (Bezugspunkt: Zeitpunkt bei der Markierungsausführung, an dem der Zielpunkt der 1. Job-Funktion erreicht wird; siehe Abbildung 14, Seite 46). In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> Siehe <code>slsc_list_suppress_spotdistance_control</code>. Siehe Abschnitt "Sonderfall: SpotDistance als "ActiveChannel"", Seite 87. <code>slsc_list_unsuppress_spotdistance_control</code> zählt zu den "SIGNAL"-Funktionen. Siehe auch Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45.
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.7.
Verweise	<code>slsc_list_suppress_spotdistance_control</code>

Name der Funktion	<code>slsc_list_wait_with_laser_off</code>
Zweck	Wie <code>slsc_list_wait_with_laser_on</code> , aber der Laser ist ausgeschaltet.
Funktions-Signatur	<code>uint32_t slsc_list_wait_with_laser_off(size_t Handle, double Time);</code>
Argument(e)	Handle Handle auf eine <code>syncAXIS control</code> -Instanz.
	Time Zeitdauer, in der das Lasersteuersignal LASERON ausgeschaltet ist. Die Spiegel verbleiben dabei in der zuletzt angefahrenen Position. In s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> • <code>slsc_list_wait_with_laser_off</code> wird zurückgewiesen, wenn <code>Time</code> kürzer ist als eine bestimmte Zeit, die sich aus angegebenen Werten in der <code>syncAXISConfig.xml</code> ergibt (unter <code><cfg:Configuration></code> → <code><cfg:TrajectoryConfig></code> → <code><cfg:MarkConfig></code> → <code><cfg:LaserSwitchConfig></code>): <ul style="list-style-type: none"> – Falls <code>LaserPreTriggerTime > 0</code>, gilt: <code>Time < LaserMinOffTime + LaserPreTriggerTime</code> – Falls <code>LaserPreTriggerTime < 0</code>, gilt: <code>Time < LaserMinOffTime</code> Beim Rückgabewert ist dann Bit #06 gesetzt (<code>UnplausibleOrUnknownParameter</code>). • <code>slsc_list_wait_with_laser_off</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. • <code>slsc_list_wait_with_laser_off</code> verhält sich wie ein <code>slsc_list_mark_abs</code> mit Geschwindigkeit 0 und ausgeschaltetem Laser. • Die komplementäre Funktion von <code>slsc_list_wait_with_laser_off</code> ist <code>slsc_list_wait_with_laser_on</code>. • Zur Zulässigkeit von <code>syncAXIS control</code>-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	Verfügbar ab <code>syncAXIS-DLL V1.2.4</code> .
Verweise	<code>slsc_list_wait_with_laser_on</code> , <code>slsc_list_mark_abs</code>

Name der Funktion	<code>slsc_list_wait_with_laser_on</code>
Zweck	Definiert eine Wartezeit, mit der der Laserspot am zuletzt definierten Zielpunkt mit angeschaltetem Laser warten soll.
Funktions-Signatur	<code>uint32_t slsc_list_wait_with_laser_on(size_t Handle, double Time);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Time Zeitdauer, in der das Lasersteuersignal LASERON eingeschaltet ist. Die Spiegel verbleiben dabei in der zuletzt angefahrenen Position. In s.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_list_wait_with_laser_on</code> wird zurückgewiesen, wenn <code>Time < 0</code>. Beim Rückgabewert ist dann Bit #06 gesetzt (<code>UnplausibleOrUnknownParameter</code>). <code>slsc_list_wait_with_laser_on</code> zählt zu den "SPIEGEL"-Funktionen und ist somit relevant für das Setzen von Output-Signalen, siehe Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. <code>slsc_list_wait_with_laser_on</code> verhält sich wie ein slsc_list_mark_abs mit Geschwindigkeit 0. Ab <code>syncAXIS-DLL ≥ V1.2.6</code> gilt: Mit <code>MinimalMarkSpeed = 0</code> ist der Übergang von <code>slsc_list_wait_with_laser_on</code> nach slsc_list_mark_abs jetzt nahtlos, weil dort die Sky Writing-ähnliche Bewegung (bei der der Laser ausgeschaltet wird) entfällt. Damit wird ermöglicht, unmittelbar nach dem Durchdringen des Werkstücks (z. B. Blech) das Schneiden zu beginnen. Ist die "Automatische Lasersteuerung" aktiv mit <code>SpotDistance</code> als "ActiveChannel", siehe Kapitel 2.9.2 "Definition der Channel und ActiveChannel", Seite 48, dann muss slsc_list_suppress_spotdistance_control vor <code>slsc_list_wait_with_laser_on</code> aufgerufen werden. <code>slsc_list_wait_with_laser_on</code> kann z. B. genutzt werden, wenn die Qualität des Laserspots durch externe Sensoren gemessen werden soll. Die komplementäre Funktion von <code>slsc_list_wait_with_laser_on</code> ist slsc_list_wait_with_laser_off. Zur Zulässigkeit von <code>syncAXIS control-Funktionen</code> im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	<pre>// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende // Anpassung und Simulation auf echten XL SCAN-Systemen aus! // Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16. size_t JobID = 0; // Create JobID variable, // initialize with 0. slsc_list_begin(Handle, &JobID); // Job start. double Target1[2] = {0, 0}; // Array of size 2 for target point 1. slsc_list_jump_abs(Handle, Target1); // Jump to target point 1. slsc_list_wait_with_laser_on(Handle, 0.5); // As if it would be a mark vector with // velocity 0 => set laser control signal LASERON // and stay at current location // for half a sec. slsc_list_end(Handle); // Job end.</pre>

Name der Funktion	slsc_list_wait_with_laser_on
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0. Letzte Änderung mit syncAXIS-DLL V1.2.6: geändertes Verhalten.
Verweise	slsc_list_wait_with_laser_off , slsc_list_mark_abs , slsc_list_suppress_spotdistance_control

Name der Funktion	slsc_list_write_analog_x
Zweck	Schreibt einen Ausgabewert auf den 12-Bit-Analogausgang ANALOG OUT1 oder ANALOG OUT2 aller RTC6-Karten.
Funktions-Signatur	<code>uint32_t slsc_list_write_analog_x(size_t Handle, slsc_AnalogOutput Channel, double Value, double TimeDelay);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Channel Analogausgang ANALOG OUT1 oder ANALOG OUT2 ("Kanal"). = 1: ANALOG OUT1. = 2: ANALOG OUT2. Zulässiger Wertebereich: [1, 2]. Siehe enum slsc_AnalogOutput .
	Value Ausgabewert am Analogausgang ANALOG OUT1 oder ANALOG OUT2. Value = 0 entspricht einem Ausgabewert von 0 V. Value = 1 entspricht einem Ausgabewert von 10 V.
	TimeDelay Relativer Zeitpunkt zwischen 2 Job-Funktionen (slsc_list_*), an dem die Änderung angewendet wird (Bezugspunkt: Zeitpunkt bei der <i>Markierungsausführung</i> , an dem der Zielpunkt der 1. Job-Funktion erreicht wird; siehe Abbildung 14, Seite 46). In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> slsc_list_write_analog_x hat bei aktivierter "Automatische Lasersteuerung" <i>keine</i> Auswirkung, wenn der bei Channel angegebene Analogausgang als "ActiveChannel" definiert ist (siehe Kapitel 2.9.2 "Definition der Channel und ActiveChannel", Seite 48) d.h. der Ausgabewert diesem Analogausgang wird durch die automatische Lasersteuerung vorgegeben. Das ANALOG OUT1-Signal wird ausgegeben bei <ul style="list-style-type: none"> – RTC6 PCI-Express-Karten (wie RTC5-Karten): LASER-Buchse, Pin 08 Das ANALOG OUT2-Signal wird ausgegeben bei <ul style="list-style-type: none"> – RTC6 PCI-Express-Karten (wie RTC5-Karten): LASER-Buchse, Pin 15, sowie MARKING ON THE FLY-Stiftleiste, Pin 14 slsc_list_write_analog_x ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Bei überhöhten TimeDelay-Werten wird die Änderung spätestens bei slsc_list_end angewendet.

Name der Funktion	<code>slsc_list_write_analog_x</code>
Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Die Änderung <ul style="list-style-type: none"> ist dauerhaft gilt daher für alle nachfolgenden Jobs Verwandter RTC6-Befehl: <code>write_da_x_list</code>. Dieser bietet den Konfigurationsparameter <code>TimeDelay</code> jedoch nicht. <code>slsc_list_write_analog_x</code> zählt zu den "SIGNAL"-Funktionen. Siehe auch Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) von <code>slsc_list_write_analog_x</code> ist <code>slsc_ctrl_write_analog_x</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.
Code-Beispiel	–
Versionsinfo	<p>Verfügbar ab syncAXIS-DLL V0.11.0.</p> <p>Änderung mit syncAXIS-DLL V1.1.0: Datentyp von <code>Channel</code>.</p> <p>Letzte Änderung mit syncAXIS-DLL V1.2.4: <code>TimeDelay</code>-Werte für V1.1 sind in V1.2 syncAXIS-DLL-intern um ein Delay verschoben (z. B. im Betriebsmodus "ScannerOnly" um 0,00125 s).</p>
Verweise	<code>slsc_ctrl_write_analog_x</code>

Name der Funktion	<code>slsc_list_write_digital_out</code>
Zweck	Schreibt einen 16-Bit-Ausgabewert auf den 16-Bit-Digital-Ausgang DIGITAL OUT 0...DIGITAL OUT 15 aller RTC6-Karten.
Funktions-Signatur	<code>uint32_t slsc_list_write_digital_out(size_t Handle, uint16_t Value, double TimeDelay);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Value 16-Bit-Ausgabewert (DIGITAL OUT 0...DIGITAL OUT 15) am 16-Bit-Digital-Ausgang.
	TimeDelay Relativer Zeitpunkt zwischen 2 Job-Funktionen (slsc_list_*), an dem die Änderung angewendet wird (Bezugspunkt: Zeitpunkt bei der <i>Markierungsausführung</i> , an dem der Zielpunkt der 1. Job-Funktion erreicht wird; siehe Abbildung 14, Seite 46). In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen" , Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Das DIGITAL OUT 0...DIGITAL OUT 15-Signal wird ausgegeben bei <ul style="list-style-type: none"> – RTC6 PCI-Express-Karten (wie RTC5-Karten): EXTENSION 1-Stiftleiste, Pin 01...Pin 31 (nur ungeradzahlige Pins) slsc_list_write_digital_out ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Bei überhöhten TimeDelay-Werten wird die Änderung spätestens bei slsc_list_end angewendet. Die Änderung <ul style="list-style-type: none"> – ist dauerhaft – gilt daher für alle nachfolgenden Jobs Verwandter RTC6-Befehl: write_io_port_list. Dieser bietet den Konfigurationsparameter TimeDelay jedoch nicht. slsc_list_write_digital_out zählt zu den "SIGNAL"-Funktionen. Siehe auch Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende Kontroll-Funktion (slsc_ctrl_*) von slsc_list_write_digital_out ist slsc_ctrl_write_digital_out. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung", Seite 70.
Code-Beispiel	–
Versionsinfo	<p>Verfügbar ab syncAXIS-DLL V0.9.0.</p> <p>Änderung mit syncAXIS-DLL V1.1.0: Datentyp von Value.</p> <p>Letzte Änderung mit syncAXIS-DLL V1.2.4: TimeDelay-Werte für V1.1 sind in V1.2 syncAXIS-DLL-intern um ein Delay verschoben (z. B. im Betriebsmodus "ScannerOnly" um 0,00125 s).</p>
Verweise	slsc_ctrl_write_digital_out, slsc_list_write_digital_out_mask

Name der Funktion	<code>slsc_list_write_digital_out_mask</code>
Zweck	Schreibt nur diejenigen Bits des <code>Value</code> -Werts auf den 16-Bit-Digital-Ausgang aller RTC6, die in der benutzerdefinierten Bitmaske (Parameter <code>Mask</code>) festgelegt sind.
Funktions-Signatur	<code>uint32_t slsc_list_write_digital_out_mask(size_t Handle, uint16_t Value, uint16_t Mask, double TimeDelay);</code>
Argument(e)	Handle Handle auf eine syncAXIS control-Instanz .
	Value 16-Bit-Ausgabewert (DIGITAL OUT 0...DIGITAL OUT 15).
	Mask 16-Bit-Maske (für DIGITAL OUT 0...DIGITAL OUT 15).
	TimeDelay Relativer Zeitpunkt zwischen 2 Job-Funktionen (<code>slsc_list_*</code>), an dem die Änderung angewendet wird (Bezugspunkt: Zeitpunkt bei der <i>Markierungsausführung</i> , an dem der Zielpunkt der 1. Job-Funktion erreicht wird; siehe Abbildung 14, Seite 46). In s. Zulässig sind nur positive Werte.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279 .
Kommentar(e)	<ul style="list-style-type: none"> Das DIGITAL OUT 0...DIGITAL OUT 15-Signal wird ausgegeben bei <ul style="list-style-type: none"> – RTC6 PCI-Express-Karten (wie RTC5-Karten): EXTENSION 1-Stiftleiste, Pin 01...Pin 31 (nur ungeradzahlige Pins) Der Parameter <code>Mask</code> legt fest, <i>welche</i> Bits des 16-Bit-Digital-Ausgangs (siehe <code>slsc_list_write_digital_out</code>) verändert werden, das Argument <code>Value</code> <i>wie</i> sie verändert werden. Die in <code>Mask</code> nicht gesetzten Bits des 16-Bit-Digital-Ausgangs bleiben unverändert. Diese werden also erneut so ausgegeben, wie sie zuletzt waren. Für <code>Mask = 0xFFFF</code> ("setze alle Bits") wirkt <code>slsc_list_write_digital_out_mask</code> wie <code>slsc_list_write_digital_out</code>. <code>slsc_list_write_digital_out_mask</code> ändert die Konfiguration der angegebenen syncAXIS control-Instanz. Die syncAXIS control-Instanz wird dabei nicht reinitialisiert. Bei überhöhten <code>TimeDelay</code>-Werten wird die Änderung spätestens bei <code>slsc_list_end</code> angewendet. Die Änderung <ul style="list-style-type: none"> – ist dauerhaft – gilt daher für alle nachfolgenden Jobs Verwandter RTC6-Befehl: <code>write_io_port_mask_list</code>. Dieser bietet den Konfigurationsparameter <code>TimeDelay</code> jedoch nicht. <code>slsc_list_write_digital_out_mask</code> zählt zu den "SIGNAL"-Funktionen. Siehe auch Kapitel 2.7.2 "Über den Zeitpunkt, wann Output-Signale tatsächlich gesetzt werden", Seite 45. Die entsprechende Kontroll-Funktion (<code>slsc_ctrl_*</code>) von <code>slsc_list_write_digital_out_mask</code> ist <code>slsc_ctrl_write_digital_out_mask</code>. Zur Zulässigkeit von syncAXIS control-Funktionen im Modus "Manuelle Positionierung" siehe Kapitel 2.12 "Über den Modus "Manuelle Positionierung"", Seite 70.

Name der Funktion	<code>slsc_list_write_digital_out_mask</code>
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.9.0. Änderung mit syncAXIS-DLL V1.1.0: Datentyp von <code>Value</code> , <code>Mask</code> . Letzte Änderung mit syncAXIS-DLL V1.2.4: <code>TimeDelay</code> -Werte für V1.1 sind in V1.2 syncAXIS-DLL-intern um ein Delay verschoben (z. B. im Betriebsmodus " <code>ScannerOnly</code> " um 0,00125 s).
Verweise	<code>slsc_ctrl_write_digital_out_mask</code> , <code>slsc_list_write_digital_out</code>

Name der Funktion	<code>slsc_util_reset_pcie</code>
Zweck	Setzt alle gefundenen RTC6 PCI-Express-Karten "hart" zurück.
Funktions-Signatur	<code>uint32_t slsc_util_reset_pcie(const char* PathToProgramFile);</code>
Argument(e)	<code>PathToProgramFile</code> Absoluter Dateipfad oder der relative Dateipfad vom Ausführungsverzeichnis aus zu den RTC6-Dateien. Diese müssen aus dem syncAXIS control-Software-Paket stammen.
Rückgabewert	Siehe Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen", Seite 279.
Kommentar(e)	<ul style="list-style-type: none"> <code>slsc_util_reset_pcie</code> ist eine Utility-Funktion, siehe auch Kapitel 3.1.4 "Utility-Funktionen (<code>slsc_util_*</code>)", Seite 101. Utility-Funktionen (<code>slsc_util_*</code>) dürfen nur außerhalb des syncAXIS control-Betriebs aufgerufen werden. ⚠ Warnung! Verletzungsgefahr durch Laserstrahlung! <code>slsc_util_reset_pcie</code> kann zu Zuständen der RTC6-Karte(n) führen, bei denen unvorhergesehen der Laser emittieren könnte! Stellen Sie vor dem Aufruf von <code>slsc_util_reset_pcie</code> sicher, dass der Laser ausgeschaltet ist! <code>slsc_util_reset_pcie</code> wird zum "harten" Reset bereitgestellt, wenn sich eine der RTC6 PCI-Express-Karten in einem Zustand befindet, der durch einen Aufruf von <code>slsc_cfg_initialize_from_file</code> nicht behoben wird (Verwenden Sie hier nicht <code>iSCANcfg.exe</code>). <code>slsc_util_reset_pcie</code> referenziert keine syncAXIS control-Instanz (= hat kein <code>Handle</code>-Argument). Daher werden keine syncAXIS control-Instanz-Einstellungen beachtet, insbesondere gibt es keinen Simulationsmodus. Achtung! <code>slsc_util_reset_pcie</code> löst im Wesentlichen den RTC6-Befehl <code>load_program_file</code> bei allen gefundenen RTC6 PCI-Express-Karten aus, unter bestimmten Umständen sogar mehrmals. Dabei wird keine Rücksicht auf laufende syncAXIS control-Instanzen genommen und bringt diese zum Absturz!
Code-Beispiel	–
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0.
Verweise	–

4 Standard-Rückgabewerte der syncAXIS-DLL Funktionen

Die meisten⁽¹⁾ syncAXIS-DLL-Funktionen stellen einen Rückgabewert mit einheitlicher Bedeutung bereit ("Standard-Rückgabewert"; nicht zu verwechseln mit Fehlercodes (Error Codes) bei `slsc_ctrl_get_error`, Log-Datei und Konsole, Seite 282).

Dieser Rückgabewert ist ein 32-Bit-Wert ohne Vorzeichen (Schema in hexadezimaler Notation: 0x nn nn nn nn, siehe Spalte **Bitmaske** in folgender Tabelle).

- Der Rückgabewert ist 0, wenn die Funktion erfolgreich ausgeführt werden konnte ("alles OK").
- Der Rückgabewert ist ... 0, wenn die Funktion nicht erfolgreich ausgeführt werden konnte. Gleichzeitig ist in ihm die Fehlerursache codiert, siehe folgende Tabelle.

Anmerkungen: Rückgabewerte ... 0 kommen dadurch zustande, weil bei einem bestimmten Fehler ein bestimmtes Bit gesetzt wird, siehe nachfolgende Tabelle. Beispiel: **Bit #03** ist gesetzt, alle anderen sind nicht gesetzt. Der Wert von **Bit #03** ist 8, also ergibt sich die **Bitmaske** 0x 00 00 00 08 (`NotAllowedInExecuting`).

Mit syncAXIS control ist immer nur ein Fehler-Bit gesetzt, d.h. es sind nie mehrere Bits gesetzt, selbst wenn mehrere Fehler aufgetreten sind.

(1) `slsc_cfg_get_sync_axis_version` beispielsweise nicht.

Bitmaske	Bit	Kurzname	Beschreibung
0x 00 00 00 00	Bit #00 = 0 (LSB)	OK	Die angegebene Funktion wurde erfolgreich ausgeführt.
0x 00 00 00 01	Bit #00 = 1 (LSB)	InErrorState	Die angegebene syncAXIS control-Instanz ist in einem Fehlerzustand. Dieser wurde wahrscheinlich aber nicht durch diese Funktion verursacht, sondern schon vorher!
0x 00 00 00 02	Bit #01 = 1	ErrorOccurred	Ein Fehler ist aufgetreten, der aber nicht näher spezifizierbar ist.
0x 00 00 00 04	Bit #02 = 1	NotAllowedWithoutInitialization	Den angegebenen Handle gibt es nicht. Auch: Die angegebene Funktion ist an dieser Stelle (noch) nicht möglich. Die syncAXIS control-Instanz ist noch nicht initialisiert. Beispiel: <code>slsc_list_begin</code> steht im Quellcode vor <code>slsc_cfg_initialize_from_file</code> .

Bitmaske	Bit	Kurzname	Beschreibung
0x 00 00 00 08	Bit #03 = 1	NotAllowedInExecuting	Die angegebene Funktion (z.B. slsc_cfg_initialize_from_file) ist an dieser Stelle nicht erlaubt, da gerade eine Ausführung läuft.
0x 00 00 00 10	Bit #04 = 1	BUFFER_FULL	Der Eingangspuffer der angegebenen syncAXIS control-Instanz hat momentan keine Kapazität mehr frei. Mögliche Maßnahme im Programmcode: Gewünschte Job-Funktion nach kurzer Warteschleife erneut absetzen. Siehe auch Kapitel 2.7.1 "Über die Puffer der syncAXIS control-Instanzen" , Seite 42.
0x 00 00 00 20	Bit #05 = 1	NotReadyForExecution	Es wurde versucht, eine Job -Ausführung zu starten. Jedoch ist (aus Sicht der syncAXIS control-Instanz) die RTC6-Karte nicht bereit zum Ausführen.
0x 00 00 00 40	Bit #06 = 1	UnplausibleOrUnknownParameter	Die angegebene Funktion hat versucht, einen Parameter zu setzen, der so nicht möglich ist. Beispiel: slsc_list_set_mark_speed(5000) , aber die Höchstgeschwindigkeit des Galvanometerscanners ist tatsächlich nur 500.
0x 00 00 00 80	Bit #07 = 1	JobStructureNotValid	Der Job hat keine gültige Struktur.
0x 00 00 01 00	Bit #08 = 1	Undefined	Reserviert.
0x 00 00 02 00	Bit #09 = 1	NotAllowedInCurrentConfiguration	Die angegebene Funktion bei der aktuellen Konfiguration nicht erlaubt.
0x 00 00 04 00	Bit #10 = 1	Undefined	Reserviert.
0x 00 00 08 00	Bit #11 = 1	NotAllowedInCurrentMode	Die angegebene Funktion ist im aktuellen Betriebsmodus nicht erlaubt.
0x 00 00 10 00	Bit #12 = 1	InvalidPosition	Die angegebene Funktion schlägt wegen einer ungültigen Position fehl. Beispiel: bei slsc_ctrl_start_execution stimmen aktuelle und die für den Job -Beginn berechnete Verfahrtschposition nicht überein.
0x 00 00 20 00	Bit #13 = 1	Timeout	Die angegebene Funktion überschreitet ein vorgegebenes Zeitlimit und schlägt deswegen fehl.
0x 00 00 40 00	Bit #14 = 1	XmlLoadError	XML-Datei nicht gefunden oder ist nicht valide.
0x 00 00 80 00	Bit #15 = 1	NotEnoughMemory	Die Initialisierung der syncAXIS control-Instanz ist fehlgeschlagen, da nicht genug freier Speicher auf dem PC vorhanden war.
0x 00 01 00 00	Bit #16 = 1	Undefined	Reserviert.
0x 00 02 00 00	Bit #17 = 1	Undefined	Reserviert.
0x 00 04 00 00	Bit #18 = 1	HandshakeFailed	Die Initialisierung des Verfahrtschis ist fehlgeschlagen.

Bitmaske	Bit	Kurzname	Beschreibung
0x 00 08 00 00	Bit #19 = 1	Undefined	Reserviert.
0x 00 10 00 00	Bit #20 = 1	UnknownException	Eine unbekannte Ausnahme ist während der Ausführung der angegebenen Funktion aufgetreten.
0x 00 20 00 00	Bit #21 = 1	Undefined	Reserviert.
0x 00 40 00 00	Bit #22 = 1	Undefined	Reserviert.
0x 00 80 00 00	Bit #23 = 1	Undefined	Reserviert.
0x 01 00 00 00	Bit #24 = 1	Undefined	Reserviert.
0x 02 00 00 00	Bit #25 = 1	Undefined	Reserviert.
0x 04 00 00 00	Bit #26 = 1	Undefined	Reserviert.
0x 08 00 00 00	Bit #27 = 1	Undefined	Reserviert.
0x 10 00 00 00	Bit #28 = 1	UnknownDevice	Das angegebene Gerät ist nicht bekannt.
0x 20 00 00 00	Bit #29 = 1	Undefined	Reserviert.
0x 40 00 00 00	Bit #30 = 1	MaxInstancesReached	Eine weitere syncAXIS control-Instanz kann nicht erzeugt werden. Die höchstmögliche Anzahl der syncAXIS control-Instanzen , die gleichzeitig auf einem PC vorhanden sein dürfen, ist auf dem Dongle codiert.
0x 80 00 00 00	Bit #31 = 1 (MSB)	InvalidOrMissingDongle	Der Dongle ist entweder nicht gültig für syncAXIS control oder nicht eingesteckt.

5 Fehlercodes (Error Codes) bei `slsc_ctrl_get_error`, Log-Datei und Konsole

- Siehe auch [Kapitel 4 "Standard-Rückgabewerte der syncAXIS-DLL Funktionen"](#), Seite 279.

Die folgenden Fehlercodes

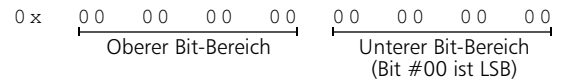
- `0x 00 00 00 02 00 00 00 01`
`EXEC_AUTOSTOP`
- `0x 00 00 00 02 00 00 00 02`
`EXEC_BUFFER_UNDERRUN`
- `0x 00 00 00 06 00 00 00 01`
`INIT_ACS_TCPIP`

beziehen sich auf:

- `slsc_ctrl_get_error`, Argument `ErrorCode`
- Log-Datei (`EnableFileLogging true`), siehe auch [Kapitel 2.8 "Über das Logging in syncAXIS control"](#), Seite 47
- Konsole (`EnableConsoleLogging true`)

Hinweise

- Für die einzelnen Fehler sind die im genannten Fehlerkonstanten vorzudefinieren.
- Zur Interpretation der 64-Bit-Werte:



Die Werte der oberen 32 Bits (Bit #63...32) bestimmen die Bedeutung der unteren 32 Bits (Bit #31...Bit #00 = LSB).

Beispiel: `0x00 00 00 00 nn nn nn nn`

=> Im unteren Bit-Bereich ist ein RTC6-Fehler codiert (wie bei RTC: Mehrere Bits sind gesetzt, wenn mehrere Fehler aufgetreten sind. Hinweis: Wie beim RTC-Kontrollbefehl `get_error` kann hier ein akkumulierter Fehlercode zurückgegeben werden!).

- Mit syncAXIS-DLL werden *keine* Fehlercodes erzeugt, in denen gleichzeitig sowohl RTC-Fehler als auch syncAXIS control-Fehler codiert sind.

Code	0x 00 00 00 02 00 00 00 01
Fehlerkonstante	EXEC_AUTOSTOP
Schlüsselwort-Folge	Autostop detected at master. Headstatus NOK! MarkingInfo-Flags ...
Ursache	<ul style="list-style-type: none"> 1. Ursache im Kontext "Initialisierung der syncAXIS control-Instanz" An der entsprechenden Achse ist kein Drive verbunden oder dieser ist stromlos. => Siehe Behebung zu 1. Ursache im Kontext "Initialisierung der syncAXIS control-Instanz". 2. Ursache im Kontext "laufender Betrieb" und "Not-Halt" Im laufenden Betrieb wurde ein "Not-Halt" durch den Benutzer (Not-Aus-Schalter), Laser, Scan-Device oder Verfahrtsch-Controller ausgelöst. => Siehe Behebung zu 2. Ursache im Kontext "laufender Betrieb" und "Not-Halt". 3. Ursache im Kontext "laufender Betrieb" und ACS-seitiges Nicht-Einhalten der Verfahrtsch-Limits Im laufenden Betrieb hat der ACS Motion-Controller ein Überschreiten der eingestellten Verfahrtsch-Limits festgestellt. => Siehe Behebung zu 3. Ursache im Kontext "laufender Betrieb" und ACS-seitiges Nicht-Einhalten der Verfahrtsch-Limits. 4. Ursache im Kontext "laufender Betrieb" und EXEC_BUFFER_UNDERRUN EXEC_BUFFER_UNDERRUN löst gewöhnlich auch einen EXEC_AUTOSTOP aus. Bei einem EXEC_BUFFER_UNDERRUN wird der Verfahrtsch nicht geregelt abgebremst. Stattdessen wird der Verfahrtsch einfach "hart gestoppt", was mit der Verletzung von Verfahrtsch-Dynamikgrenzen einhergeht. => Siehe Behebung zu 4. Ursache im Kontext "laufender Betrieb" und EXEC_BUFFER_UNDERRUN. 5. Ursache im Kontext "laufender Betrieb" und Scan-Kopf-bezogener Fehler Im laufenden Betrieb sendet der Scan-Kopf kein PWROK-Signal ("Power OK"), siehe excelliSCAN-Handbuch. => Siehe Behebung zu 5. Ursache im Kontext "laufender Betrieb" und Scan-Kopf-bezogener Fehler.

Code	0x 00 00 00 02 00 00 00 01
Behebung	<ul style="list-style-type: none"> • Behebung zu 1. Ursache im Kontext "Initialisierung der syncAXIS control-Instanz" <ol style="list-style-type: none"> (1) ACS-Fehler in ACS SPiiPlus MMI Application Studio auslesen. Dieser lautet in etwa: "Axis <n> disabled: Error 5027 (Motion termination error), Servo Processor Alarm". (2) Nehmen Sie die betroffene Achse mit ACS SPiiPlus MMI Application Studio in Betrieb. (3) Stellen Sie sicher, dass die Achsenkonfiguration in der <code>syncAXISConfig.xml</code> richtig ist, z. B. <pre><cfg:StageAxisX>0</cfg:StageAxisX> <cfg:StageAxisY>1</cfg:StageAxisY> <cfg:SlecEtherCATNodeID>0</cfg:SlecEtherCATNodeID></pre> • Behebung zu 2. Ursache im Kontext "laufender Betrieb" und "Not-Halt" <ol style="list-style-type: none"> (1) ACS-Fehler in ACS SPiiPlus MMI Application Studio auslesen. Dieser lautet in etwa: "Error 5028 (Motion Termination error), Safe Torque Off" (2) Setzen Sie die Wirkung des "Not-Halt" gemäß Ihres Sicherheitskonzepts zurück. (3) Initialisieren Sie die syncAXIS control-Instanz erneut. • Behebung zu 3. Ursache im Kontext "laufender Betrieb" und ACS-seitiges Nicht-Einhalten der Verfahrtsch-Limits <ol style="list-style-type: none"> (1) Die syncAXIS-DLL prüft die Dynamikgrenzen und reagiert wie in <code>DynamicViolationReaction</code> eingestellt. Bei <code>WarningOnly</code> wird eine [WARN]-Log-Dateizeile geschrieben, siehe [WARN]-Log-Dateizeilen. An dieser können Sie sehen, ob die Überschreitung durch syncAXIS-DLL festgestellt wurde. Bei <code>AbortImmediately</code> und <code>StopAndReport</code> wird der Job abgebrochen, ohne dass Sie von syncAXIS-DLL einen Fehler zu sehen bekommen. (2) ACS-Fehler in ACS SPiiPlus MMI Application Studio auslesen. <ul style="list-style-type: none"> • Bei "Error 5015 (Motion termination error), Software Right Limit" wurde die maximal zulässige Position überschritten • Bei "Error 5076 (Motion termination error), Driver Alarm: Drive Saturation" oder "Error 5023 (Motion termination error), Critical Position Error" wurde die Geschwindigkeits-Grenze überschritten • Bei "Error Error 5076 (Motion termination error), Driver Alarm: Drive Saturation" oder "Error 5023 (Motion termination error), Critical Position Error" wurde die Beschleunigungs-Grenze überschritten • Bei "Error 5023 (Motion termination error), Critical Position Error" wurde die Ruck-Grenze überschritten (3) Falls einer der in (2) genannten Fehler die Ursache für den <code>EXEC_AUTOSTOP</code> ist, können Sie möglicherweise (nur im Rahmen des sicheren Bereichs und je nach Prozessanforderung) den maximal erlaubten Positionsfehler im ACS Motion-Controller erhöhen.

Code	0x 00 00 00 02 00 00 00 01
Behebung (Forts.)	<p>(4) Anderenfalls müssen Sie den Job anpassen. Im Simulationsmodus (siehe Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31) müssen Sie zunächst identifizieren, an welcher Stelle des Jobs die Dynamikgrenzen verletzt werden. Wie in Kapitel 2.6 "Über das Optimieren von syncAXIS control-basierten Anwenderprogrammen", Seite 36 beschrieben, müssen Sie durch iteratives Ändern der Werte für Geschwindigkeiten, Dynamiken und <code>FilterBandwidth</code> sicherstellen, dass die Verletzung nicht mehr auftritt.</p> <ul style="list-style-type: none"> • Behebung zu 4. Ursache im Kontext "laufender Betrieb" und EXEC_BUFFER_UNDERRUN Da <code>EXEC_BUFFER_UNDERRUN</code> und <code>EXEC_AUTOSTOP</code> meist assoziiert sind, siehe Abschnitt "Pufferunterläufe vermeiden", Seite 42. • Behebung zu 5. Ursache im Kontext "laufender Betrieb" und Scan-Kopf-bezogener Fehler <ul style="list-style-type: none"> – Prüfen Sie mit <code>iSCANcfg.exe</code> aus dem syncAXIS control-Software-Paket, ob der Scan-Kopf antwortet. Wenn nicht, prüfen Sie die Verkabelung. – Kontaktieren Sie SCANLAB je nach rückgemeldetem Fehler des Scan-Kopf – Falls die Scan-Kopf-Temperatur zu hoch war, definieren Sie den Job mit geringeren Dynamikanforderungen. – Verbessern Sie die Kühlung und thermische Anbindung des Scan-Kopfs.
Kommentar(e)	<ul style="list-style-type: none"> • Einen Hinweis zur Fehlerursache können die mitausgegebenen MarkingInfo-Flags geben. Diese entsprechen dem Rückgabewert des RTC6-Befehls <code>get_marking_info</code>. Beispiel: Wenn Flags für den Scan-Kopf-Anschluss gesetzt sind, an dem der Verfahrtschisch angeschlossen ist, dann deutet das auf ein Problem in der Ansteuerung dieses Verfahrtschischs hin.

Code	0x 00 00 00 02 00 00 00 02
Fehlerkonstante	EXEC_BUFFER_UNDERRUN
Schlüsselwort-Folge	Buffer underrun detected on RTC ...
Ursache	Pufferunterlauf, Seite 14.
Behebung	Siehe Abschnitt "Pufferunterläufe vermeiden", Seite 42.
Kommentar(e)	<ul style="list-style-type: none"> EXEC_BUFFER_UNDERRUN löst gewöhnlich auch einen EXEC_AUTOSTOP aus. Bei einem EXEC_BUFFER_UNDERRUN wird der Verfahrtschicht <i>nicht</i> geregelt abgebremst. Stattdessen wird der Verfahrtschicht einfach "hart gestoppt", was mit der Verletzung von Verfahrtschicht-Dynamikgrenzen einhergeht.

Code	0x 00 00 00 06 00 00 00 01
Fehlerkonstante	INIT_ACS_TCPIP
Schlüsselwort-Folge	Establishing TCP/IP communication to the ACS controller failed ...
Ursache	<ul style="list-style-type: none"> 1. Ursache Der Eintrag in der <code>syncAXISConfig.xml</code> unter <code><cfg:Configuration></code> → <code><cfg:GeneralConfig></code> → <code><cfg:ACSController></code> fehlt oder ist falsch. => Siehe Behebung zu 1. Ursache. 2. Ursache Der ACS Motion-Controller ist noch nicht vollständig hochgefahren. => Siehe Behebung zu 2. Ursache. 3. Ursache Die Verbindung zum ACS Motion-Controller ist fehlerhaft. => Siehe Behebung zu 3. Ursache. 4. Ursache In ACS SPiPlus MMI Application Studio werden "Network Error"-Meldungen geöffnet. => Siehe Behebung zu 4. Ursache. 5. Ursache Probleme mit dem ACS User Mode Driver (UMD). => Siehe Behebung zu 5. Ursache. 6. Ursache Auf dem ACS Motion-Controller ist die XL SCAN-Option nicht freigeschaltet. => Siehe Behebung zu 6. Ursache.
Behebung	<ul style="list-style-type: none"> Behebung zu 1. Ursache (1) Identifizieren Sie die IP-Adresse des ACS Motion-Controllers mit dem ACS SPiPlus MMI Application Studio (z.B. "#SI" in ACS Communication Terminal). (2) Berichtigen Sie den Eintrag in der <code>syncAXISConfig.xml</code> unter <code><cfg:Configuration></code> → <code><cfg:GeneralConfig></code> → <code><cfg:ACSController></code>. Behebung zu 2. Ursache (1) Ein Neustart kann bis zu 5 Minuten dauern. Warten Sie entsprechend. (2) Wenn der Fehler weiterhin besteht: Siehe Behebung zu 3. Ursache.

Code	0x 00 00 00 06 00 00 00 01
Behebung (Forts.)	<ul style="list-style-type: none"> • Behebung zu 3. Ursache <ol style="list-style-type: none"> (1) Prüfen Sie, ob eine Verbindung zum ACS Motion-Controller über ACS SPiiPlus MMI Application Studio möglich ist. (2) Wenn nicht: Finden und beseitigen Sie Fehler, die die Netzwerkverbindung betreffen. Mögliche Ursachen: Defekte Hardware (Netzwerkkarte, Netzwerkkabel, Router, WLAN-Komponenten), falsche Konfiguration, Blockierungen durch eine Firewall. (3) Wenn Sie dann immer noch keine Verbindung herstellen können: Kontaktieren Sie den ACS-Support. • Behebung zu 4. Ursache <ol style="list-style-type: none"> (1) Versuchen Sie, den Fehler mit "#LOG" in ACS Communication Terminal nachzuvollziehen. (2) Finden und beseitigen Sie Fehler, die die Netzwerkverbindung betreffen. Mögliche Ursachen: Defekte Hardware (Netzwerkkarte, Netzwerkkabel, Router, WLAN-Komponenten), falsche Konfiguration, Blockierungen durch eine Firewall. (3) Erhöhen Sie das "Communication Timeout" in ACS SPiiPlus MMI Application Studio. (4) Wenn der "Network Error" weiterhin besteht: Kontaktieren Sie den ACS-Support. • Behebung zu 5. Ursache <ul style="list-style-type: none"> – Starten Sie den ACS User Mode Driver (UMD) neu. • Behebung zu 6. Ursache <ol style="list-style-type: none"> (1) Geben Sie "#SI" im ACS Communication Terminal ein. (2) "XL SCAN units" muss > 0 sein. Wenn "XL SCAN units" = 0 ist: Kontaktieren Sie den ACS-Support fragen Sie nach einem Upgrade.
Kommentar(e)	<ul style="list-style-type: none"> • –

6 Strukturen

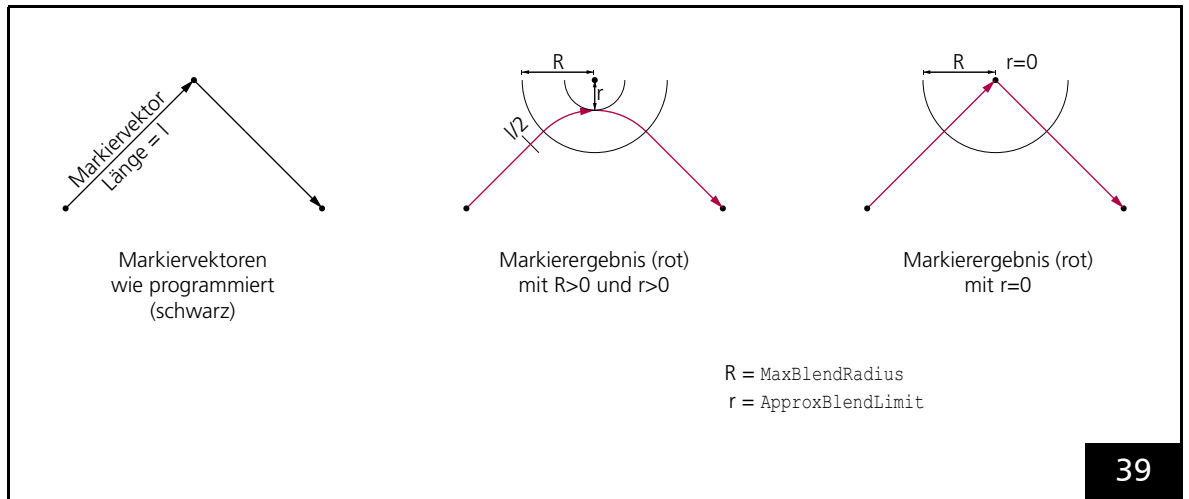
In diesem Kapitel:

- Struktur **slsc_GeometryConfig**
- Struktur **slsc_MarkConfig**
- Struktur **slsc_MultiParaTarget**
- Struktur **slsc_ParaSection**
- Struktur **slsc_TrajectoryConfig**
- Struktur **VersionInfo**

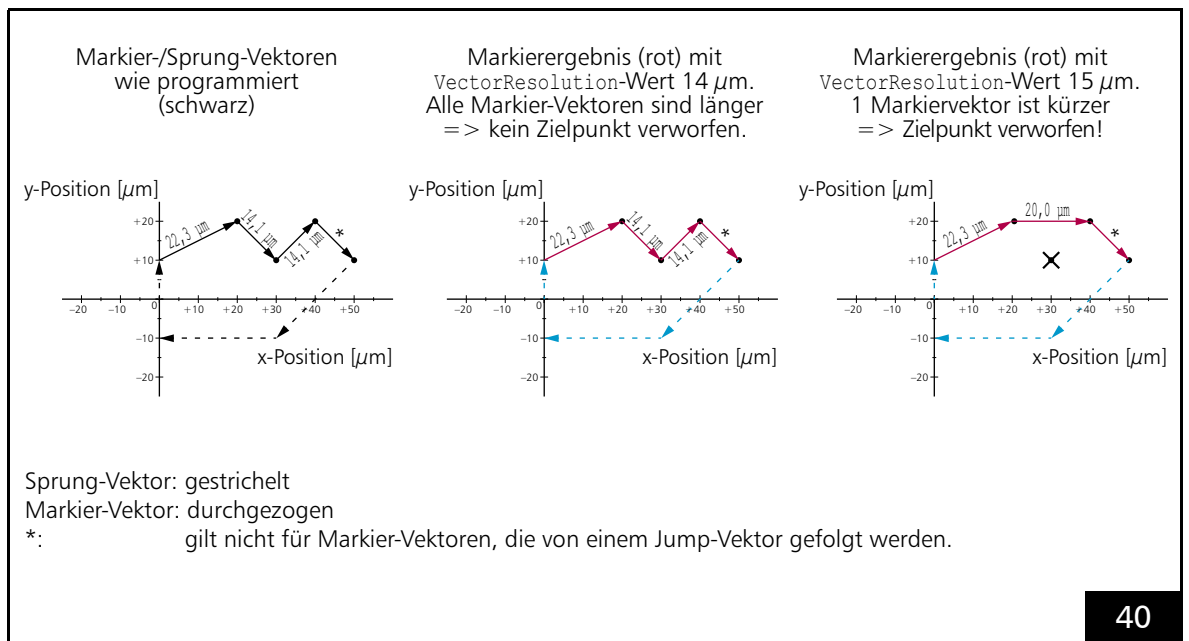
Name der Struktur	slsc_GeometryConfig	
Beschreibung	Diese Struktur definiert (als Teil der Konfiguration der Trajektorienplanung) das Verhalten der Blending-Kurven (\leq V1.4.0: auch von Splines).	
Verwendung	Diese Struktur wird verwendet durch: <ul style="list-style-type: none"> • Struktur slsc_TrajectoryConfig 	
Syntax	<pre>struct slsc_GeometryConfig { double MaxBlendRadius; double ApproxBlendLimit; slsc_BlendModes BlendMode; double VectorResolution; bool AutoCyclicGeometry; double SplineConversionLengthLimit; slsc_SplineModes SplineMode; };</pre>	
Argument(e)	double MaxBlendRadius	<p>Siehe 'R' in Abbildung 39, Seite 292: Radius des Kreises um den Eckpunkt (zwischen 2 Vektoren), in dem die Blending-Kurve liegen soll. Im Markierungsergebnis sind Blending-Kurven auch durch die Vektorlänge begrenzt: $R_{\text{actual}} = \min(R, l/2)$.</p> <p>Wenn die bei MaxBlendRadius und/oder ApproxBlendLimit angegebenen Werte nicht eingehalten werden können, dann führt die syncAXIS control-Instanz einen Sprung (so ähnlich, aber anders als bei RTC-Sky-Writing) aus. Wichtig: Für BlendMode = slsc_BlendModes_MinimalBlending wird der MaxBlendRadius-Wert als Grenze für den Beginn der Blending-Kurve verwendet.</p> <p>Der entsprechende syncAXISConfig.xml-Tag ist MaxBlendRadius.</p>

Name der Struktur	slsc_GeometryConfig		
Argument(e) (Forts.)	double	ApproxBlendLimit	<p>Siehe 'r' in Abbildung 39, Seite 292: Größter tolerierbarer mathematischer Abstand der Blending-Kurve zum Eckpunkt.</p> <p>Wenn die bei <code>MaxBlendRadius</code> und/oder <code>ApproxBlendLimit</code> angegebenen Werte nicht eingehalten werden können, dann führt die syncAXIS control-Instanz einen Sprung (so ähnlich, aber anders als bei RTC-Sky-Writing) aus.</p> <p>Mit slsc_list_set_approx_blend_limit kann der <code>ApproxBlendLimit</code>-Wert für einen bestimmten Job geändert werden. Die Änderung gilt ab der Einfügestelle, aber nur bis zum Ende des gerade laufenden Jobs.</p> <p>Der entsprechende <code>syncAXISConfig.xml</code>-Tag ist <code>ApproxBlendLimit</code>.</p>
	slsc_BlendModes	BlendMode	Der anzuwendende Blend-Modus.
	double	VectorResolution	<p>Siehe Abbildung 40, Seite 292: Falls die Zielpunkte von zwei aufeinanderfolgenden Markier-Funktionen (slsc_list_mark_abs, slsc_list_multi_para_mark_abs, slsc_list_para_mark_abs) einen kleineren Abstand haben als der <code>VectorResolution</code>-Wert, dann werden diese Zielpunkte durch die syncAXIS control-Instanz als identisch betrachtet. Bei Abfolgen von Markiervektor → Markiervektor werden also unter bestimmten Umständen Zielpunkte verworfen.</p> <p>Die vorgesehene Verwendung von <code>VectorResolution</code> ist, Benutzern eine Eingabeungenauigkeit bei der Angabe von Zielpunkten zu ermöglichen (z.B. bei floating points oder wenn Daten automatisiert eingelesen werden). Sinnvolle Werte liegen daher im Mikrometer-Bereich, z.B. 0,02 mm.</p> <p>Der entsprechende <code>syncAXISConfig.xml</code>-Tag ist <code>VectorResolution</code>.</p>

Name der Struktur	slsc_GeometryConfig	
Argument(e) (Forts.)	<code>bool</code> <code>AutoCyclicGeometry</code>	Veraltet. Verwenden Sie nur noch <code>false</code> . Der entsprechende <code>syncAXISConfig.xml</code> -Tag ist <code>AutoCyclicGeometry</code> .
	<code>double</code> <code>SplineConversionLengthLimit</code>	Veraltet. Verwenden/belassen Sie den Default-Wert. Der entsprechende <code>syncAXISConfig.xml</code> -Tag ist <code>SplineConversionLengthLimit</code> .
	<code>slsc_SplineModes</code> <code>SplineMode</code>	Veraltet. Verwenden Sie nur noch <code>slsc_SplineModes_Deactivated</code> . Der entsprechende <code>syncAXISConfig.xml</code> -Tag ist <code>SplineMode</code> .
Kommentar(e)	<ul style="list-style-type: none"> In der <code>syncAXIS-DLL</code> gilt bezüglich der Priorität von Trajektorien-Berechnungen: Blending-Kurve zu Eckpunkt > Sky Writing-ähnliche Bewegung. 	
Versionsinfo	Verfügbar ab <code>syncAXIS-DLL V0.11.0</code> .	



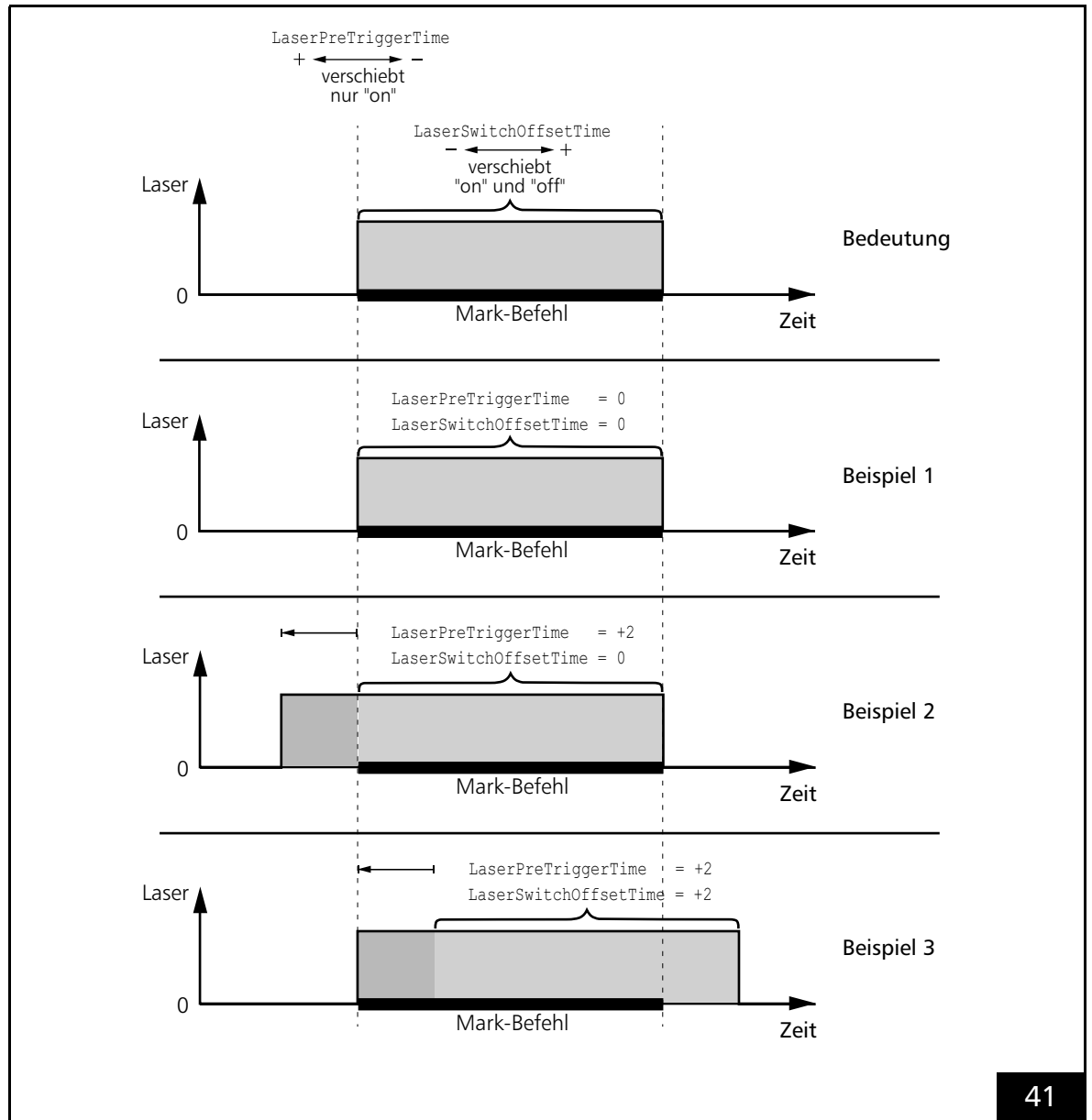
Struktur **slsc_GeometryConfig**: Zur Wirkungsweise der Argumente **MaxBlendRadius** und **ApproxBlendLimit**.

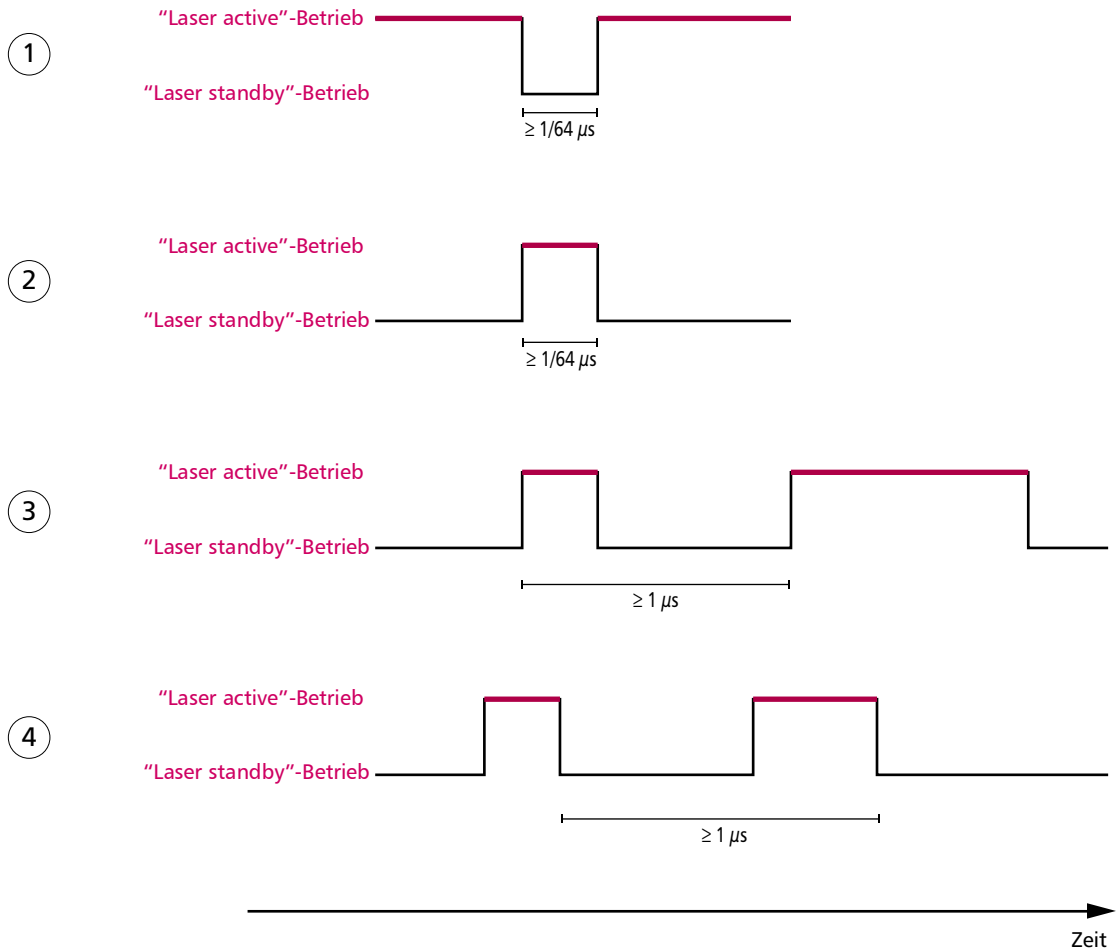


Struktur **slsc_GeometryConfig**: Zur Wirkungsweise des Arguments **VectorResolution** in $\geq V1.2$.

Name der Struktur	slsc_MarkConfig	
Beschreibung	Diese Struktur definiert die Grundeinstellungen der Trajektorienplanung für Markierungen, z. B. Laserschaltzeitpunkte.	
Verwendung	Diese Struktur wird verwendet durch: <ul style="list-style-type: none"> Struktur slsc_TrajectoryConfig 	
Syntax	<pre>struct slsc_MarkConfig { double LaserPreTriggerTime; double LaserSwitchOffsetTime; double LaserMinOffTime; double JumpSpeed; double MarkSpeed; double MinimalMarkSpeed; };</pre>	
Argument(e)	double LaserPreTriggerTime	<p>Siehe Abbildung 41, Seite 295: Zeit, in der das Lasersignal im Voraus getriggert wird, wenn ein Markier-Segment ausgeführt wird. Einheit: s.</p> <p>Der entsprechende <code>syncAXISConfig.xml</code>-Tag ist <code>LaserPreTriggerTime</code>.</p> <p>Verwandter RTC6-Befehl: set_laser_delays (es bestehen Ähnlichkeiten zu LaserOn-Delay und LaserOff-Delay, siehe RTC6-Handbuch, Kapitel 7.2.1 "Laser-Delays", Seite 144).</p>
	double LaserSwitchOffsetTime	<p>Siehe Abbildung 41, Seite 295: Zeit, um die die Ausgabe der Lasersignale verschoben wird. Einheit: s.</p> <p>Der entsprechende <code>syncAXISConfig.xml</code>-Tag ist <code>LaserSwitchOffsetTime</code>.</p> <p>Verwandter RTC6-Befehl: set_laser_delays (es bestehen Ähnlichkeiten zu LaserOn-Delay und LaserOff-Delay, siehe RTC6-Handbuch, Kapitel 7.2.1 "Laser-Delays", Seite 144).</p>
	double LaserMinOffTime	<p>Siehe Abbildung 42, Seite 296 (1): Kürzeste "Laser standby"-Betriebs-Zeitdauer.</p> <p>Einheit: s.</p> <p>Der entsprechende <code>syncAXISConfig.xml</code>-Tag ist <code>LaserMinOffTime</code>.</p>

Name der Struktur	slsc_MarkConfig		
Argument(e) (Forts.)	double	JumpSpeed	Die gewünschte kombinierte Sprunggeschwindigkeit (d.h. die kombinierte Scanner- und Verfahrtschbewegung) während der Job -Ausführung. Ab \geq V1.5.0 gilt: Im Betriebsmodus "StageOnly" wird diese Geschwindigkeit streng als Höchstgeschwindigkeit eingehalten. Im Betriebsmodus "ScannerOnly" und "ScannerAndStage" kann der Wert auch zeitweilig leicht überschritten werden. Der entsprechende <code>syncAXISConfig.xml</code> -Tag ist <code>JumpSpeed</code> .
	double	MarkSpeed	Die höchste erwünschte kombinierte Markiergeschwindigkeit (d.h. die kombinierte Scanner- und Verfahrtschbewegung) während der Job -Ausführung. Bei Blending-Kurven an Eckpunkten kann es sein, dass die <code>syncAXIS control-Instanz</code> eine niedrigere Geschwindigkeit anwenden muss. Dann wird mindestens der Wert von <code>MinimalMarkSpeed</code> verwendet. Der entsprechende <code>syncAXISConfig.xml</code> -Tag ist <code>MarkSpeed</code> .
	double	MinimalMarkSpeed	Die niedrigste erwünschte Spotgeschwindigkeit, die in Ecken der Kontur erreicht werden soll. Bei Blending-Kurven an Eckpunkten darf <code>syncAXIS control-Instanz</code> zu dieser Geschwindigkeit abbremsen, um ein Abrunden zu ermöglichen. Siehe <code>MarkSpeed</code> . Wenn ein genau definiertes Ecken-Blending mit der bekannten Scanner- und Verfahrtschdynamik mit mindestens dieser minimalen Markierungsgeschwindigkeit nicht durchgeführt werden kann, wird eine Sky Writing-ähnliche Bewegung ausgeführt. Der entsprechende <code>syncAXISConfig.xml</code> -Tag ist <code>MinimalMarkSpeed</code> .
Kommentar(e)	• –		
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.		





Legende

1. Die kürzeste "Laser standby"-Betriebs-Zeitdauer (und somit die kürzeste Zeitdauer von Sprung-Segmenten) ist:
 - Der LaserMinOffTime-Wert
 Die syncAXIS-DLL verlängert Sprung-Segmente und Sky Writing-ähnliche Bewegungen entsprechend, wenn sie kürzer als oben angegeben sind. Der kleinste zulässige LaserMinOffTime-Wert ist $1/64 \mu s$.
2. Die kürzeste "Laser active"-Betriebs-Zeitdauer (und somit die kürzeste Zeitdauer von Markier-Segmenten) ist:
 - Immer $1/64 \mu s$
3. "Laser active"-Betrieb-Starts müssen immer mindestens $1 \mu s$ Abstand haben. Anderenfalls gilt:
 - Bei [*]dashed[*]-Funktionen lehnt die syncAXIS-DLL diese ab
 - Bei Sprung-Segmenten verlängert die syncAXIS-DLL deren Dauer entsprechend
4. "Laser standby"-Betrieb-Starts müssen immer mindestens $1 \mu s$ Abstand haben. Anderenfalls gilt:
 - Wie 3

Name der Struktur	slsc_MultiParaTarget		
Beschreibung	Diese Struktur definiert eine Rampe , die aus mehreren Abschnitten (die mit slsc_ParaSection definiert werden) besteht.		
Verwendung	<p>Diese Struktur wird verwendet durch:</p> <ul style="list-style-type: none"> • slsc_list_multi_para_arc_abs • slsc_list_multi_para_circle_2d_abs • slsc_list_multi_para_dashed_arc_abs • slsc_list_multi_para_dashed_circle_2d_abs • slsc_list_multi_para_dashed_mark_abs • slsc_list_multi_para_mark_abs 		
Syntax	<pre>struct slsc_MultiParaTarget { slsc_ParaSection* Targets; size_t NumParaTargets; };</pre>		
Argument(e)	slsc_ParaSection*	Targets	Zeiger auf ein Array mit variabler Größe (diese ist angegeben in NumParaTargets).
	size_t	NumParaTargets	Anzahl der Elemente in Targets.
Kommentar(e)	<ul style="list-style-type: none"> • Siehe auch Abschnitt "Über Rampen", Seite 53. 		
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.		

Name der Struktur	slsc_ParaSection	
Beschreibung	Diese Struktur definiert einen Abschnitt der Rampe (die mit slsc_MultiParaTarget definiert wird).	
Verwendung	Diese Struktur wird verwendet durch: <ul style="list-style-type: none"> Struktur slsc_MultiParaTarget 	
Syntax	<pre>struct slsc_ParaSection { double ds; double MultiParaTarget; };</pre>	
Argument(e)	double ds	Länge des Abschnitts (bezogen auf die Vektorlänge/Kreissegmentlänge). Auch 0 ist zulässig (=sprunghafte Änderung). In mm.
	double MultiParaTarget	Faktor Ip am Ende von ds. Keine Einheit. Zum Faktor Ip siehe Abschnitt "Über die Berechnung der ActiveChannel-Werte entlang einer Kontur" , Seite 51.
Kommentar(e)	<ul style="list-style-type: none"> Diese Struktur eignet sich auch zur Definition von Rampen mit Sägezahn- und Rechteck-Profil, siehe auch Abschnitt "Über Rampen", Seite 53. 	
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.	

Name der Struktur	slsc_TrajectoryConfig		
Beschreibung	Diese Struktur definiert die Konfiguration der Trajektorienplanung .		
Verwendung	Diese Struktur wird verwendet durch: <ul style="list-style-type: none"> • slsc_cfg_delete_trajectory_config • slsc_cfg_get_trajectory_config • slsc_cfg_set_trajectory_config 		
Syntax	<pre>struct slsc_TrajectoryConfig { slsc_MarkConfig MarkConfig; slsc_GeometryConfig GeometryConfig; };</pre>		
Argument(e)	slsc_MarkConfig	MarkConfig	Konfiguriert die Grundeinstellungen für Markierungen.
	slsc_GeometryConfig	GeometryConfig	Konfiguriert das Verhalten der Blending-Kurven (\leq V1.4.0: auch von Splines).
Kommentar(e)	• –		
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.		

Name der Struktur	VersionInfo		
Beschreibung	Diese Struktur definiert die drei Ziffernblöcke der syncAXIS-DLL-Version ("Version n.n.n").		
Verwendung	Diese Struktur wird verwendet durch: <ul style="list-style-type: none"> • slsc_cfg_get_sync_axis_version 		
Syntax	<pre>struct VersionInfo { uint32_t Major; uint32_t Minor; uint32_t Revision; };</pre>		
Argument(e)	uint32_t	Major	Major-Version der syncAXIS-DLL.
	uint32_t	Minor	Minor-Version der syncAXIS-DLL.
	uint32_t	Revision	Revision der syncAXIS-DLL.
Kommentar(e)	• –		
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.1.0.		

7 Aufzählungstypen enum

In diesem Kapitel:

- enum **slsc_AnalogOutput**
- enum **slsc_BlendModes**
- enum **slsc_DynamicsMonitoringLevel**
- enum **slsc_DynamicViolationReaction**
- enum **slsc_ExecState**
- enum **slsc_JobCharacteristic**
- enum **slsc_ListHandlingMode**
- enum **slsc_MeasurementSignal**
- enum **slsc_OperationMode**
- enum **slsc_OperationStatus**
- enum **slsc_PositionType**
- enum **slsc_ScanDevice**
- enum **slsc_SimulationSetting**
- enum **slsc_SplineModes**
- enum **slsc_Stage**

Name des enum	slsc_AnalogOutput	
Beschreibung	Dieses enum legt die Auswahlmöglichkeiten fest für: <ul style="list-style-type: none"> Die Analog-Ausgänge 	
Verwendung	Dieses enum wird verwendet durch: <ul style="list-style-type: none"> slsc_ctrl_write_analog_x slsc_list_write_analog_x 	
Syntax	<pre>enum slsc_AnalogOutput { slsc_AnalogOutput_1 =0, slsc_AnalogOutput_2 =1, };</pre>	
Aufzählungs-konstante(n)	slsc_AnalogOutput_1	Analogausgang 1.
	slsc_AnalogOutput_2	Analogausgang 2.
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.	

Name des enum	slsc_BlendModes	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Den Blend-Modus 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_GeometryConfig 	
Syntax	<pre>enum slsc_BlendModes { slsc_BlendModes_Deactivated = 0, slsc_BlendModes_VariableBlending = 1, slsc_BlendModes_MinimalBlending = 2, slsc_BlendModes_FixedBlending = 3, };</pre>	
Aufzählungskonstante(n)	slsc_BlendModes_Deactivated	<p>Dieser Parameter wirkt auf Abfolgen von:</p> <ul style="list-style-type: none"> Markiervektor → Markiervektor Markiervektor → Kreis Kreis → Markiervektor <p>Die syncAXIS-DLL baut bei jedem dieser Abfolgen ("Eckpunkt"), die das System nicht fahren kann (d. h. wenn die Dynamikwerte dort nicht mehr im zulässigen Bereich wären), eine Sky Writing-ähnliche Bewegung ein.</p> <p>Siehe syncAXISConfig.xml-Tag BlendMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Deactivated</p>
	slsc_BlendModes_VariableBlending	<p>In der aktuellen syncAXIS-DLL wirkt dieser Parameter nur auf Abfolgen von Markiervektor → Markiervektor.</p> <p>Die syncAXIS-DLL versucht, bei jedem Eckpunkt Markiervektor → Markiervektor Blending-Kurven einzubauen. Diese Blending-Kurven werden immer mit <i>größtem</i> Abstand berechnet. D.h. bei MaxBlendRadius (bei Struktur slsc_GeometryConfig) wird immer der Wert $\min(R, l/2)$ beachtet.</p> <p>slsc_BlendModes_VariableBlending hat eine geringere Berechnungszeit als slsc_BlendModes_MinimalBlending.</p> <p>Siehe syncAXISConfig.xml-Tag BlendMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>VariableBlending</p>

Name des enum	slsc_BlendModes	
Aufzählungskonstante(n) (Forts.)	slsc_BlendModes_MinimalBlending	<p>In der aktuellen syncAXIS-DLL wirkt dieser Parameter nur auf Abfolgen von Markiervektor → Markiervektor.</p> <p>Die syncAXIS-DLL versucht, bei Eckpunkten von Markiervektor → Markiervektor jeweils eine Blending-Kurve einzubauen. Diese Blending-Kurven werden immer mit <i>geringstmöglich noch ausführbaren</i> Abstand berechnet.</p> <p>slsc_BlendModes_MinimalBlending hat eine höhere Berechnungszeit als slsc_BlendModes_VariableBlending.</p> <p>Siehe syncAXISConfig.xml-Tag BlendMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>MinimalBlending</p>
	slsc_BlendModes_FixedBlending	<p>Veraltet.</p> <p>In der aktuellen syncAXIS-DLL wirkt dieser Parameter nur auf Abfolgen von:</p> <p>Markiervektor → Markiervektor</p> <p>Die syncAXIS-DLL wandelt die aufeinanderfolgende Linien in einen zusammengesetzten Spline (engl.: composite spline) um. Beachten Sie, dass dies ein völlig anderes Verhalten wie bei slsc_BlendModes_VariableBlending und slsc_BlendModes_MinimalBlending ist.</p> <p>Siehe syncAXISConfig.xml-Tag BlendMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>FixedBlending</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.4.2.	

Name des enum	slsc_DynamicsMonitoringLevel	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Die zu überwachenden Dynamik-Parameter 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_cfg_get_scan_device_dynamic_monitoring_level slsc_cfg_get_stage_dynamic_monitoring_level slsc_cfg_set_scan_device_dynamic_monitoring_level slsc_cfg_set_stage_dynamic_monitoring_level 	
Syntax	<pre>enum slsc_DynamicsMonitoringLevel { slsc_DynamicsMonitoringLevel_Deactivated =0, slsc_DynamicsMonitoringLevel_Position =1, slsc_DynamicsMonitoringLevel_Velocity =2, slsc_DynamicsMonitoringLevel_Acceleration =3, slsc_DynamicsMonitoringLevel_Jerk =4, };</pre>	
Aufzählungskonstante(n)	slsc_DynamicsMonitoringLevel_Deactivated	<p>Es soll keine Überwachung stattfinden. Siehe syncAXISConfig.xml-Tag MonitoringLevel. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Deactivated</p>
	slsc_DynamicsMonitoringLevel_Position	<p>Es sollen Überschreitungen der X- und Y-Positionswerte (Arbeitsfeldgrenzen) überwacht werden. Siehe syncAXISConfig.xml-Tag MonitoringLevel. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Position</p>
	slsc_DynamicsMonitoringLevel_Velocity	<p>Wie slsc_DynamicsMonitoringLevel_Position und zusätzlich: Es sollen Überschreitungen der Geschwindigkeit (Dynamikgrenzen) überwacht werden. Siehe syncAXISConfig.xml-Tag MonitoringLevel. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Velocity</p>
	slsc_DynamicsMonitoringLevel_Acceleration	<p>Wie slsc_DynamicsMonitoringLevel_Velocity und zusätzlich: Es sollen Überschreitungen der Beschleunigung (Dynamikgrenzen) überwacht werden. Siehe syncAXISConfig.xml-Tag MonitoringLevel. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Acceleration</p>
	slsc_DynamicsMonitoringLevel_Jerk	<p>Wie slsc_DynamicsMonitoringLevel_Acceleration und zusätzlich: Es sollen Überschreitungen der Rucke (Dynamikgrenzen) überwacht werden. Siehe syncAXISConfig.xml-Tag MonitoringLevel. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Jerk</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.	

Name des enum	slsc_DynamicViolationReaction	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Die Reaktion bei einer Dynamik-Grenzwertüberschreitung 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_cfg_get_dynamic_violation_reaction slsc_cfg_set_dynamic_violation_reaction 	
Syntax	<pre>enum slsc_DynamicViolationReaction { slsc_DynamicViolationReaction_WarningOnly =0, slsc_DynamicViolationReaction_AbortImmediately =1, slsc_DynamicViolationReaction_StopAndReport =2, };</pre>	
Aufzählungskonstante(n)	slsc_DynamicViolationReaction_WarningOnly	<p>Es sollen nur [WARN]-Log-Dateizeilen erzeugt werden. Siehe syncAXISConfig.xml-Tag DynamicViolationReaction. Vorgeschriebene Notation, wenn dort verwendet: WarningOnly</p>
	slsc_DynamicViolationReaction_AbortImmediately	<p>Es soll sofort abgebrochen werden. Insbesondere im Betriebsmodus "StageOnly" und "ScannerAndStage" führt das zum Abbruch des Prozesses ("Not-Halt"). Die ACS-Steuerung (wenn entsprechend konfiguriert) wechselt in einen Fehlerzustand. Siehe syncAXISConfig.xml-Tag DynamicViolationReaction. Vorgeschriebene Notation, wenn dort verwendet: AbortImmediately</p>
	slsc_DynamicViolationReaction_StopAndReport	<p>Es wird vor dem Abbruch zunächst versucht, eine Abbremsbewegung auszuführen, um anzuhalten (Diese Abbremsbewegung ist erkennbar, wenn die entsprechende Simulationsdatei im syncAXIS Viewer dargestellt wird). Dann wechselt die syncAXIS control-Instanz in einen Fehlerzustand. Benutzer müssen deswegen die syncAXIS control-Instanz neu initialisieren. Die ACS-Steuerung wechselt aber gewöhnlich nicht in einen Fehlerzustand. Siehe syncAXISConfig.xml-Tag DynamicViolationReaction. Vorgeschriebene Notation, wenn dort verwendet: StopAndReport</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.	

Name des enum	slsc_ExecState
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Den Status des Execution Layers
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_ctrl_get_exec_state
Syntax	<pre>enum slsc_ExecState { slsc_ExecState_Idle =0, slsc_ExecState_ReadyForExecution =1, slsc_ExecState_Executing =2, slsc_ExecState_NotInitOrError =3, };</pre>
Aufzählungs-konstante(n)	slsc_ExecState_Idle Die RTC6-Karte ist untätig.
	slsc_ExecState_ReadyForExecution Die RTC6-Karte ist bereit zur Ausführung von Jobs .
	slsc_ExecState_Executing Die RTC6-Karte führt gerade einen Job aus.
	slsc_ExecState_NotInitOrError Die RTC6-Karte ist nicht initialisiert oder es liegt ein Fehler vor.
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.

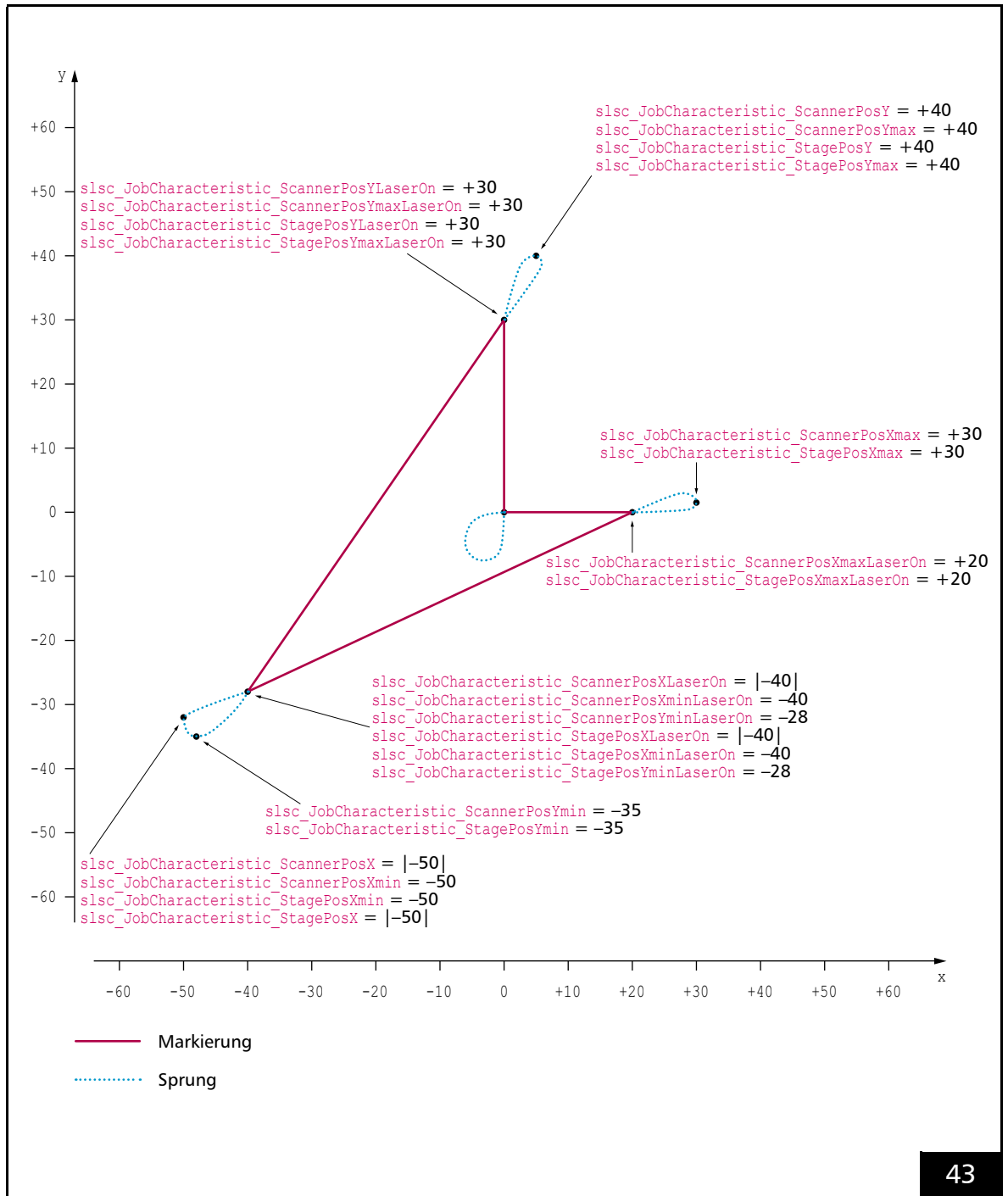
Name des enum	slsc_JobCharacteristic
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Das Job-Merkmal ("Key") <p>Es bezieht sich auf die bei slsc_ctrl_get_job_characteristic angegebene Job-ID (ist also "Job-bezogen").</p> <p>Da die Job-Merkmals-Werte in der Trajektorienplanung berechnet werden, muss die angegebene Job-ID den Status "Berechnung: Beendet" (siehe Abbildung 12, Seite 43), erreicht haben.</p> <p>Siehe auch Abbildung 43, Seite 311.</p>
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_ctrl_get_job_characteristic
Syntax	<pre>enum slsc_JobCharacteristic { slsc_JobCharacteristic_ScannerPosX = 0, slsc_JobCharacteristic_ScannerPosY = 1, slsc_JobCharacteristic_StagePosX = 2, slsc_JobCharacteristic_StagePosY = 3, slsc_JobCharacteristic_ScannerVelX = 4, slsc_JobCharacteristic_ScannerVelY = 5, slsc_JobCharacteristic_StageVelX = 6, slsc_JobCharacteristic_StageVelY = 7, slsc_JobCharacteristic_ScannerAccX = 8, slsc_JobCharacteristic_ScannerAccY = 9, slsc_JobCharacteristic_StageAccX = 10, slsc_JobCharacteristic_StageAccY = 11, slsc_JobCharacteristic_StageJerkX = 12, slsc_JobCharacteristic_StageJerkY = 13, slsc_JobCharacteristic_MotionMicroSteps = 14, slsc_JobCharacteristic_ScannerPosXLaserOn = 30, slsc_JobCharacteristic_ScannerPosYLaserOn = 31, slsc_JobCharacteristic_StagePosXLaserOn = 32, slsc_JobCharacteristic_StagePosYLaserOn = 33, slsc_JobCharacteristic_MinimalMarkSpeed = 50, slsc_JobCharacteristic_MaximalMarkSpeed = 51, slsc_JobCharacteristic_InsertedSkywritings = 200, slsc_JobCharacteristic_ScannerPosXmax = 251, slsc_JobCharacteristic_ScannerPosXmin = 252, slsc_JobCharacteristic_ScannerPosYmax = 253, slsc_JobCharacteristic_ScannerPosYmin = 254, slsc_JobCharacteristic_ScannerPosXmaxLaserOn = 255, slsc_JobCharacteristic_ScannerPosXminLaserOn = 256, slsc_JobCharacteristic_ScannerPosYmaxLaserOn = 257, slsc_JobCharacteristic_ScannerPosYminLaserOn = 258, slsc_JobCharacteristic_StagePosXmax = 259, slsc_JobCharacteristic_StagePosXmin = 260, slsc_JobCharacteristic_StagePosYmax = 261, slsc_JobCharacteristic_StagePosYmin = 262, slsc_JobCharacteristic_StagePosXmaxLaserOn = 263, slsc_JobCharacteristic_StagePosXminLaserOn = 264, slsc_JobCharacteristic_StagePosYmaxLaserOn = 265, slsc_JobCharacteristic_StagePosYminLaserOn = 266, };</pre>

Name des enum	slsc_JobCharacteristic
Aufzählungskonstante(n)	<p>slsc_JobCharacteristic_ScannerPosX</p> <p>Größte Entfernung des Scan-Kopf zum Nullpunkt (ohne Offset, siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332). In x-Richtung. In mm.</p> <p>Hinweise</p> <ul style="list-style-type: none"> Definierte Offsets (siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332) sind in den Werten enthalten und werden nicht kompensiert. In der Trajektorienplanung berechneter Wert. Der tatsächlich ausgeführte Wert kann abweichend sein (z. B. wegen der RTC6-Korrekturdatei).
	<p>slsc_JobCharacteristic_ScannerPosY</p> <p>Größte Entfernung des Scan-Kopf zum Nullpunkt (ohne Offset, siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332). In y-Richtung. In mm.</p> <p>Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.</p>
	<p>slsc_JobCharacteristic_StagePosX</p> <p>Größte Entfernung des Verfahrtisches zum Nullpunkt (ohne Offset, siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332). In x-Richtung. In mm.</p> <p>Hinweise</p> <ul style="list-style-type: none"> Definierte Offsets (siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332) sind in den Werten enthalten und werden nicht kompensiert. In der Trajektorienplanung berechneter Wert. Der tatsächlich ausgeführte Wert kann abweichend sein (z. B. wegen nachgelagerter Vorgänge z. B. in der ACS-Ansteuerung wie Fehlerkorrekturen).
	<p>slsc_JobCharacteristic_StagePosY</p> <p>Größte Entfernung des Verfahrtisches zum Nullpunkt (ohne Offset, siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332). In y-Richtung. In mm.</p> <p>Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.</p>
	<p>slsc_JobCharacteristic_ScannerVelX</p> <p>Betrag der höchsten Scan-Kopf-Geschwindigkeit. In x-Richtung. In mm/s.</p> <p>Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.</p>
	<p>slsc_JobCharacteristic_ScannerVelY</p> <p>Betrag der höchsten Scan-Kopf-Geschwindigkeit. In y-Richtung. In mm/s.</p> <p>Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.</p>

Name des enum	slsc_JobCharacteristic	
Aufzählungskonstante(n) (Forts.)	slsc_JobCharacteristic_StageVelX	Betrag der höchsten Verfahrtisch-Geschwindigkeit. In x-Richtung. In mm/s. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StageVelY	Betrag der höchsten Verfahrtisch-Geschwindigkeit. In y-Richtung. In mm/s. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_ScannerAccX	Betrag der höchsten Scan-Kopf-Beschleunigung. In x-Richtung. In mm/s ² . Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.
	slsc_JobCharacteristic_ScannerAccY	Betrag der höchsten Scan-Kopf-Beschleunigung. In y-Richtung. In mm/s ² . Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.
	slsc_JobCharacteristic_StageAccX	Betrag der höchsten Verfahrtisch-Beschleunigung. In x-Richtung. In mm/s ² . Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StageAccY	Betrag der höchsten Verfahrtisch-Beschleunigung. In y-Richtung. In mm/s ² . Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StageJerkX	Betrag des höchsten Verfahrtisch-Rucks. In x-Richtung. In mm/s ³ . Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StageJerkY	Betrag des höchsten Verfahrtisch-Rucks. In y-Richtung. In mm/s ³ . Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_MotionMicroSteps	Anzahl der Mikrovektoren, aus dem der Job besteht. In der Trajektorienplanung berechneter Wert. slsc_JobCharacteristic_MotionMicroSteps entspricht somit der Minstdauer des Jobs (1 Mikrovektor = 10 µs). Die tatsächliche Ausführungszeit kann aber länger sein (aufgrund nachgelagerter Vorgänge im System, daher ist hier keine quantitative Angabe möglich). slsc_JobCharacteristic_MotionMicroSteps ist insbesondere nützlich, um die Auswirkung von Parameterpermutationen (im Rahmen von Optimierungsschritten) auszuwerten.
	slsc_JobCharacteristic_ScannerPosXLaserOn	Betrag der höchsten Scan-Kopf-Position. In x-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.
	slsc_JobCharacteristic_ScannerPosYLaserOn	Betrag der höchsten Scan-Kopf-Position. In y-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.

Name des enum	slsc_JobCharacteristic	
Aufzählungskonstante(n) (Forts.)	slsc_JobCharacteristic_StagePosXLaserOn	Betrag der höchsten Verfahrtisch-Position. In x-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosYLaserOn	Betrag der höchsten Verfahrtisch-Position. In y-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosY.
	slsc_JobCharacteristic_MinimalMarkSpeed	Die niedrigste Markiergeschwindigkeit während des Jobs . Es werden nur diejenigen Anteile berücksichtigt, in denen der Laser angeschaltet ist (Laserspotgeschwindigkeit).
	slsc_JobCharacteristic_MaximalMarkSpeed	Die höchste Markiergeschwindigkeit während des Jobs . Es werden nur diejenigen Anteile berücksichtigt, in denen der Laser angeschaltet ist (Laserspotgeschwindigkeit).
	slsc_JobCharacteristic_InsertedSkywritings	Betrifft nur die folgenden Abfolgen im Job : <ul style="list-style-type: none"> • Markiervektor → Markiervektor • Markiervektor → Kreissegment • Kreissegment → Markiervektor • Kreissegment → Kreissegment Anzahl der Abfolgen, bei den eine Sky Writing-ähnliche Bewegung eingefügt wurde (weil: ein direktes Überfahren die Dynamikgrenzen verletzen würde und es kann auch kein Blending eingebaut werden).
	slsc_JobCharacteristic_ScannerPosXmax	Höchste Scan-Kopf-Position. In x-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.
	slsc_JobCharacteristic_ScannerPosXmin	Geringste Scan-Kopf-Position. In x-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.
	slsc_JobCharacteristic_ScannerPosYmax	Höchste Scan-Kopf-Position. In y-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosY.
	slsc_JobCharacteristic_ScannerPosYmin	Geringste Scan-Kopf-Position. In y-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosY.
	slsc_JobCharacteristic_ScannerPosXmaxLaserOn	Höchste Scan-Kopf-Position. In x-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.
	slsc_JobCharacteristic_ScannerPosXminLaserOn	Geringste Scan-Kopf-Position. In x-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosX.
	slsc_JobCharacteristic_ScannerPosYmaxLaserOn	Höchste Scan-Kopf-Position. In y-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_ScannerPosY.

Name des enum	slsc_JobCharacteristic	
Aufzählungskonstante(n) (Forts.)	slsc_JobCharacteristic_ScannerPosYminLaserOn	Geringste Scan-Kopf-Position. In y-Richtung. Bei angeschaltetem Laser. In mm. Hinweise
	slsc_JobCharacteristic_StagePosXmax	Höchste Verfahrtisch-Position. In x-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosXmin	Geringste Verfahrtisch-Position. In x-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosYmax	Höchste Verfahrtisch-Position. In y-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosYmin	Geringste Verfahrtisch-Position. In y-Richtung. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosXmaxLaserOn	Höchste Verfahrtisch-Position. In x-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosXminLaserOn	Geringste Verfahrtisch-Position. In x-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosYmaxLaserOn	Höchste Verfahrtisch-Position. In y-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
	slsc_JobCharacteristic_StagePosYminLaserOn	Geringste Verfahrtisch-Position. In y-Richtung. Bei angeschaltetem Laser. In mm. Es gelten die Hinweise bei slsc_JobCharacteristic_StagePosX.
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.	



enum **slsc_JobCharacteristic**. Beispielhafte Werte-Rückgaben.

Name des enum	slsc_ListHandlingMode	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Das Rückgabeverhalten der Job-Funktionen (slsc_list_*) 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_cfg_set_list_handling_mode slsc_cfg_set_list_handling_mode_with_context 	
Syntax	<pre>enum slsc_ListHandlingMode { slsc_ListHandlingMode_ReturnAtOnce =0, slsc_ListHandlingMode_RepeatWhileBufferFull =1, slsc_ListHandlingMode_RepeatWhilePredicate =2, };</pre>	
Aufzählungs-konstante(n)	slsc_ListHandlingMode_ReturnAtOnce	<p>Kein Listen-Handling.</p> <p>Siehe syncAXISConfig.xml-Tag InitialListHandlingMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>ReturnAtOnce</p>
	slsc_ListHandlingMode_RepeatWhileBufferFull	<p>Es wird versucht, die Job-Funktion solange auszuführen, bis beim Rückgabewert das Bit #04 nicht mehr gesetzt ist (nicht mehr BUFFER_FULL). Fehler werden trotzdem zurückgegeben.</p> <p>Siehe syncAXISConfig.xml-Tag InitialListHandlingMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>RepeatWhileBufferFull</p>
	slsc_ListHandlingMode_RepeatWhilePredicate	<p>Es wird versucht, die Job-Funktion solange auszuführen, bis der Rückgabewert der Predicate-Funktion nicht mehr true ist.</p> <p>Siehe syncAXISConfig.xml-Tag InitialListHandlingMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>RepeatWhilePredicate</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.	

Name des enum	slsc_MeasurementSignal	
Beschreibung	Dieses enum legt die Auswahlmöglichkeiten fest für: <ul style="list-style-type: none"> Die Messsignale 	
Verwendung	Dieses enum wird verwendet durch: <ul style="list-style-type: none"> slsc_ctrl_get_value 	
Syntax	<pre>enum slsc_MeasurementSignal { slsc_MeasurementSignal_Status =0, slsc_MeasurementSignal_Sample =1, slsc_MeasurementSignal_AnalogOutput_1 =2, slsc_MeasurementSignal_AnalogOutput_2 =3, slsc_MeasurementSignal_Errors =4, };</pre>	
Aufzählungs-konstante(n)	slsc_MeasurementSignal_Status	Veraltet. Ab syncAXIS-DLL V1.2.0 steht enum slsc_PositionType zur Verfügung. [Bedeutung in ≤ V1.1: Reserviert.]
	slsc_MeasurementSignal_Sample	Veraltet. Ab syncAXIS-DLL V1.2.0 steht enum slsc_PositionType zur Verfügung. [Bedeutung in ≤ V1.1: (Durch die RTC6 kommandierter) Positionswert der angegebenen Achse. Bei Scan-Kopf: Auslenkung im Bildfeld. In mm. Bei Verfahrtsch: Positionswert. In mm.]
	slsc_MeasurementSignal_AnalogOutput_1	Wert am Analogausgang 1.
	slsc_MeasurementSignal_AnalogOutput_2	Wert am Analogausgang 2.
	slsc_MeasurementSignal_Errors	Reserviert.
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.	

Name des enum	slsc_OperationMode	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Den Betriebsmodus der syncAXIS control-Instanz 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_cfg_get_mode slsc_cfg_set_mode 	
Syntax	<pre>enum slsc_OperationMode { slsc_OperationMode_ScannerOnly =0, slsc_OperationMode_StageOnly =1, slsc_OperationMode_ScannerAndStage =2, };</pre>	
Aufzählungskonstante(n)	slsc_OperationMode_ScannerOnly	<p>Nur Scan-Kopf, kein Verfahrtsch.</p> <p>Die Geschwindigkeitsgrenzen sind durch JumpSpeed und MarkSpeed vorgegeben. In diesem Betriebsmodus wird die Dynamik allerdings durch die Scanner-Dynamikgrenzen eingeschränkt.</p> <p>Siehe syncAXISConfig.xml-Tag InitialOperationMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>ScannerOnly</p>
	slsc_OperationMode_StageOnly	<p>Nur Verfahrtsch, kein Scan-Kopf.</p> <p>Die Geschwindigkeitsgrenzen sind durch JumpSpeed und MarkSpeed vorgegeben. In diesem Betriebsmodus wird die Dynamik allerdings durch die reduzierten Verfahrtsch-Dynamikgrenzen eingeschränkt. Die Geschwindigkeitsgrenze ist insbesondere durch die maximale Verfahrtsch-Geschwindigkeit limitiert.</p> <p>Siehe syncAXISConfig.xml-Tag InitialOperationMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>StageOnly</p>
	slsc_OperationMode_ScannerAndStage	<p>Beides, Scan-Kopf und Verfahrtsch.</p> <p>Die Geschwindigkeitsgrenzen sind durch JumpSpeed und MarkSpeed vorgegeben. In diesem Betriebsmodus wird die Dynamik allerdings durch die Scanner-Dynamikgrenzen beschränkt und durch weitere Heuristiken beeinflusst.</p> <p>Siehe syncAXISConfig.xml-Tag InitialOperationMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>ScannerAndStage</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.	

Name des enum	slsc_OperationStatus	
Beschreibung	Dieses enum legt die Auswahlmöglichkeiten fest für: <ul style="list-style-type: none"> Den Betriebsstatus der syncAXIS control-Instanz 	
Verwendung	Dieses enum wird verwendet durch: <ul style="list-style-type: none"> slsc_cfg_get_operation_status 	
Syntax	<pre>enum slsc_OperationStatus { slsc_OperationStatus_Green =0, slsc_OperationStatus_Yellow =1, slsc_OperationStatus_Red =2, };</pre>	
Aufzählungs-konstante(n)	slsc_OperationStatus_Green	Betriebsstatus "grün": Die syncAXIS control-Instanz läuft und es sind keine Fehler aufgetreten.
	slsc_OperationStatus_Yellow	Betriebsstatus "gelb": Die syncAXIS control-Instanz läuft noch nicht, weil die Initialisierung noch nicht abgeschlossen ist.
	slsc_OperationStatus_Red	Betriebsstatus "rot": Die syncAXIS control-Instanz läuft nicht und es ist mindestens ein Fehler aufgetreten.
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.	

Name des enum	slsc_PositionType	
Beschreibung	Dieses enum legt die Auswahlmöglichkeiten fest für: <ul style="list-style-type: none"> Den Positionstyp 	
Verwendung	Dieses enum wird verwendet durch: <ul style="list-style-type: none"> slsc_ctrl_get_scan_device_position slsc_ctrl_get_stage_position 	
Syntax	<pre>enum slsc_PositionType { slsc_PositionType_Set = 0, slsc_PositionType_Actual = 1, };</pre>	
Aufzählungs-konstante(n)	slsc_PositionType_Set	Soll-Position.
	slsc_PositionType_Actual	Ist-Position.
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.4.	

Name des enum	slsc_ScanDevice	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Dasjenige Scan-Device, auf die eine bestimmte Einstellung (z.B. eine Transformation) angewendet werden soll 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_cfg_set_part_displacement 	
Syntax	<pre>enum slsc_ScanDevice { slsc_ScanDevice_None = 0, slsc_ScanDevice1 = 1, slsc_ScanDevice2 = 2, slsc_ScanDevice3 = 3, slsc_ScanDevice4 = 4, };</pre>	
Aufzählungs-konstante(n)	slsc_ScanDevice_None	<p>Kein Scan-Device.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, ScanDevice. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>None</p>
	slsc_ScanDevice1	<p>Scan-Device 1.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, ScanDevice. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>ScanDevice1</p>
	slsc_ScanDevice2	<p>Scan-Device 2.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, ScanDevice. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>ScanDevice2</p>
	slsc_ScanDevice3	<p>Scan-Device 3.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, ScanDevice. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>ScanDevice3</p>
	slsc_ScanDevice4	<p>Scan-Device 4.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, ScanDevice. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>ScanDevice4</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.3.0.	

Name des enum	slsc_SimulationSetting	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Die Simulationseinstellung der syncAXIS control-Instanz 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_cfg_get_simulation_setting slsc_cfg_set_simulation_setting 	
Syntax	<pre>enum slsc_SimulationSetting { slsc_SimulationSetting_SimulationMode =0, slsc_SimulationSetting_HardwareMode =1, };</pre>	
Aufzählungs-konstante(n)	slsc_SimulationSetting_SimulationMode	<p>Die syncAXIS control-Instanz versucht nicht, eine Verbindung zu irgendeiner Hardware herzustellen und simuliert lediglich Jobs auf dem PC.</p> <p>Siehe syncAXISConfig.xml-Tag SimulationMode true.</p>
	slsc_SimulationSetting_HardwareMode	<p>Die syncAXIS control-Instanz stellt eine Verbindung zur gesamten Hardware her und führt Jobs physisch aus.</p> <p>Siehe syncAXISConfig.xml-Tag SimulationMode false.</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.5.0.	

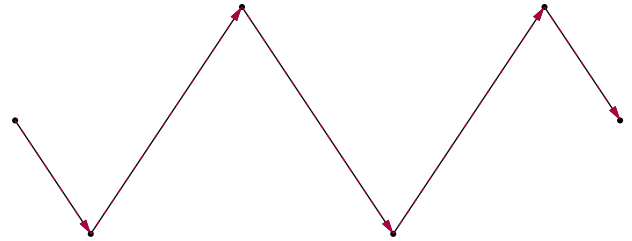
Name des enum	slsc_SplineModes	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Den Spline-Modus (\leq V1.4.0) 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> Struktur slsc_GeometryConfig 	
Syntax	<pre>enum slsc_SplineModes { slsc_SplineModes_Deactivated = 0, slsc_SplineModes_Interpolating = 1, slsc_SplineModes_Approximating = 2, };</pre>	
Aufzählungs-konstante(n)	slsc_SplineModes_Deactivated	<p>Empfohlene Einstellung. Keine Splines, siehe Abbildung 44, Seite 319.</p> <p>Siehe syncAXISConfig.xml-Tag SplineMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Deactivated</p>
	slsc_SplineModes_Interpolating	<p>Veraltet. Es wurden interpolierende Splines verwendet, siehe Abbildung 44, Seite 319.</p> <p>Siehe syncAXISConfig.xml-Tag SplineMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Interpolating</p>
	slsc_SplineModes_Approximating	<p>Veraltet. Es wurden approximierende Splines verwendet, siehe Abbildung 44, Seite 319.</p> <p>Siehe syncAXISConfig.xml-Tag SplineMode. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Approximating</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V0.11.0.	

Die 3 syncAXIS control Spline-Modi. Nur für Abfolgen von > 2 Markier-Vektoren.
 Markiervektoren wie programmiert: schwarz.
 Markiierungsergebnis: rot.

≥ V1.5.0: Empfohlen.

`slsc_SplineModes_Deactivated`

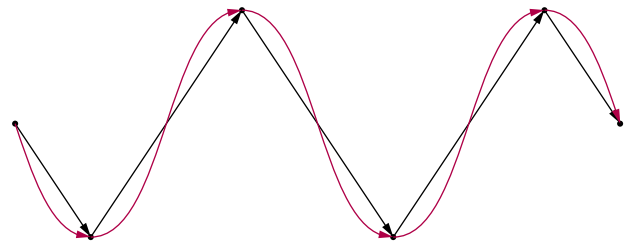
Als Trajektorien werden keine Splines berechnet.
 Das Markierungsergebnis weist keine Kurven auf.



≥ V1.5.0: Veraltet.

`slsc_SplineModes_Interpolating`

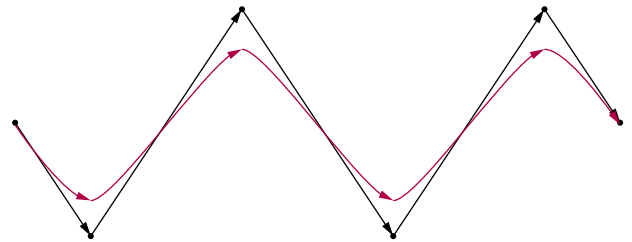
Als Trajektorien werden Splines berechnet.
 Das Markierungsergebnis weist Kurven auf.
 Die Kurve verläuft durch ihre Stützpunkte.



≥ V1.5.0: Veraltet.

`slsc_SplineModes_Approximating`

Als Trajektorien werden Splines berechnet.
 Das Markierungsergebnis weist Kurven auf.
 Die Kurve verläuft nicht durch ihre Stützpunkte
 (Kontrollpunkte liegen neben der Kurve).



44

enum **slsc_SplineModes**: Zur Wirkungsweise der Argumente `slsc_SplineModes_Deactivated`,
`slsc_SplineModes_Interpolating` und `slsc_SplineModes_Approximating`.

Name des enum	slsc_Stage	
Beschreibung	<p>Dieses enum legt die Auswahlmöglichkeiten fest für:</p> <ul style="list-style-type: none"> Denjenigen Verfahrtisch, der verwendet werden soll 	
Verwendung	<p>Dieses enum wird verwendet durch:</p> <ul style="list-style-type: none"> slsc_cfg_select_stage 	
Syntax	<pre>enum slsc_Stage { slsc_Stage_None = 0, slsc_Stage1 = 1, slsc_Stage2 = 2, slsc_Stage3 = 3, slsc_Stage4 = 4, };</pre>	
Aufzählungs-konstante(n)	slsc_Stage_None	<p>Kein Verfahrtisch.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, Stage. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>None</p>
	slsc_Stage1	<p>Verfahrtisch 1.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, Stage. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Stage1</p>
	slsc_Stage2	<p>Verfahrtisch 2.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, Stage. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Stage2</p>
	slsc_Stage3	<p>Verfahrtisch 3.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, Stage. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Stage3</p>
	slsc_Stage4	<p>Verfahrtisch 4.</p> <p>Siehe syncAXISConfig.xml-Tag HeadA, HeadB, Stage. Vorgeschriebene Notation, wenn dort verwendet:</p> <p>Stage4</p>
Versionsinfo	Verfügbar ab syncAXIS-DLL V1.2.4.	

8 Anhang A: syncAXIS control V1.2.4 und höher mit XL SCAN-Multi-Head-Systemen nutzen

Hinweise

- Der Begriff **Multi-Head** bezieht sich auf ein XL SCAN-System, bei dem
 - 1 **syncAXIS control-Instanz**
 - mehr als einen excelliSCAN-Scan-Kopf⁽¹⁾ (in diesem Anhang bezieht sich "Multi" auf $n=4$, siehe **Abbildung 45, Seite 322**) und
 - 1 Verfahrtschicht ansteuert.
 Erforderlich dazu ist ein **Dongle** für syncAXIS control, der ausdrücklich für die entsprechende Anzahl von Scan-Köpfe konfiguriert ist⁽²⁾.

8.1 Über diesen Anhang

Dieser Anhang wendet sich ausschließlich an Systemintegratoren, die bereits wissen, wie man ein XL SCAN-Einzel-Kopf-System implementiert und zusammen mit syncAXIS control verwendet.

Es gilt zusammen mit dem:

- Handbuch "Installation der SCANLAB XL SCAN-Komponenten und Erstinbetriebnahme des XL SCAN-Systems"
- Hauptteil dieses Handbuchs "syncAXIS-DLL – Schnittstelle zur Anwendungsprogrammierung"

Achtung!

Lesen Sie das Dokument "syncAXIS control Lizenzvertrag" sorgfältig durch, bevor Sie syncAXIS control installieren und verwenden. Diese Vereinbarung definiert Themen wie Nutzungsbedingungen, Garantieinformationen und Haftungsausschlüsse. Wenn Sie dazu Fragen haben, wenden Sie sich an SCANLAB.



Vorsicht!

Lesen und befolgen Sie alle Sicherheitshinweise in diesem Handbuch!

SCANLAB übernimmt keine Haftung für Schäden oder Folgeschäden aufgrund Nichtbeachtung dieses Handbuchs, insbesondere der hierin genannten Sicherheitshinweise.



Vorsicht!

Lesen und befolgen Sie alle Sicherheitshinweise in diesen Handbüchern:

- Handbuch "Installation der SCANLAB XL SCAN-Komponenten und Erstinbetriebnahme des XL SCAN-Systems"
- Handbuch "syncAXIS-DLL – Schnittstelle zur Anwendungsprogrammierung"

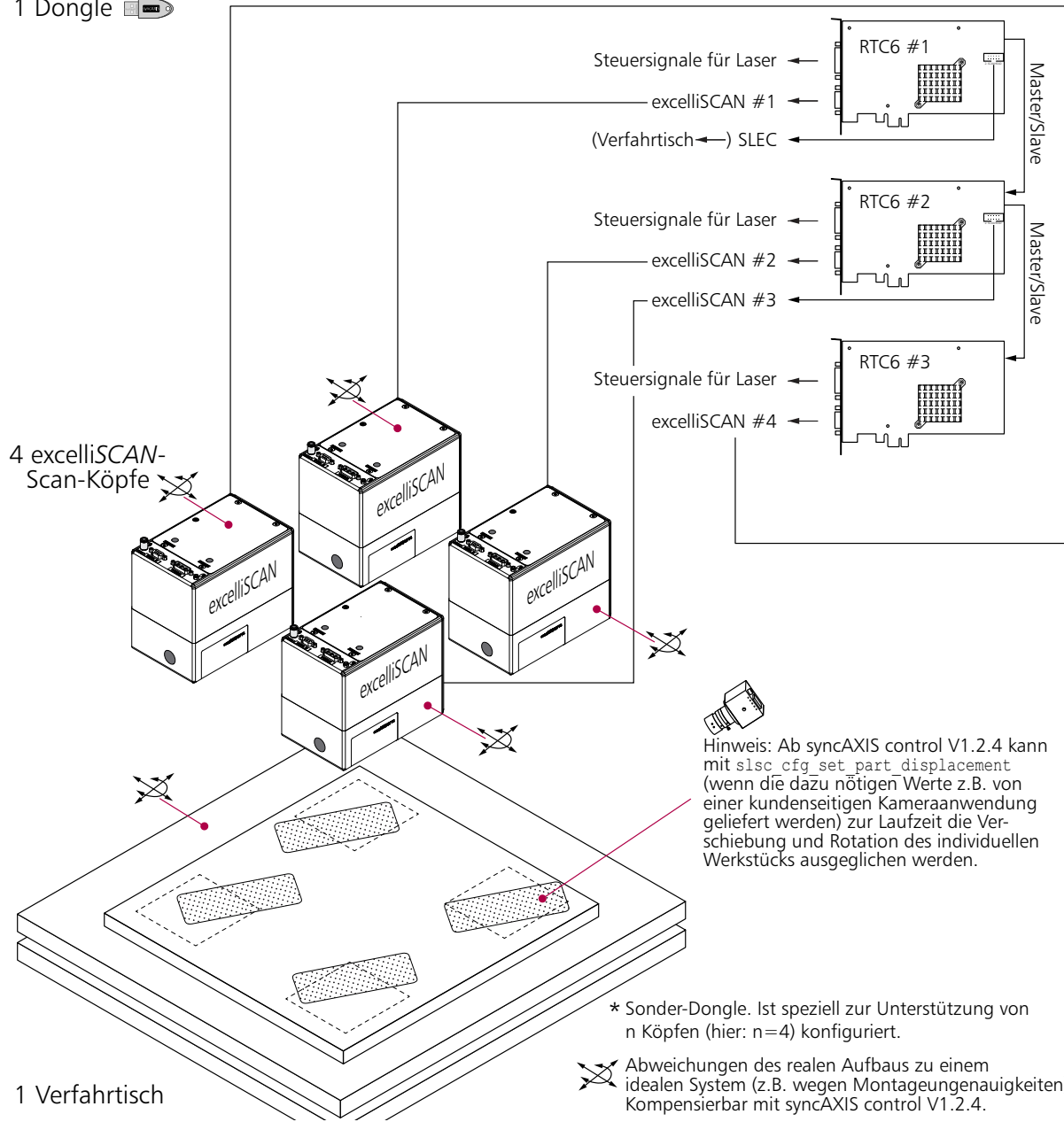
SCANLAB übernimmt keine Haftung für Schäden oder Folgeschäden aufgrund Nichtbeachtung dieses Handbuchs, insbesondere der dort genannten Sicherheitshinweise.

(1) Alle Scan-Köpfe sollen das gleiche Muster markieren.

(2) D.h. ein gewöhnlicher Standard-Dongle ist *nicht* ausreichend.

1 syncAXIS control ($\geq V1.2.4$) -Instanz
1 Dongle *

(über PCIe Bus
an alle RTC6)



Beispiel für einen 4-Kopf-Aufbau (schematische Darstellung): 1 syncAXIS control-Instanz, 4 excelliSCAN-Scan-Köpfe, 1 Verfahrtsch.

In der `syncAXISConfig.xml` muss die Anordnung der Komponenten entsprechend konfiguriert werden, siehe auch **Abbildung 47, Seite 326**.

Der Hauptteil des Handbuch "Installation der SCANLAB XL SCAN-Komponenten und Erstinbetriebnahme des XL SCAN-Systems" bezieht sich auf ein XL SCAN-System mit nur 1 excelliSCAN Scan-Kopf (und nur 1 RTC6-Karte) und 1 Verfahrtisch. Es gilt analog für Multi-Head-Systeme, aber mit den folgenden Unterschieden:

- Es müssen 3 weitere excelliSCAN Scan-Köpfe eingebaut und betriebsbereit gemacht werden.
- Es müssen 2 weitere RTC6-Karten eingebaut werden⁽¹⁾. Anmerkung: bei einer der RTC6-Karten ist der Anschluss für den zweiten Scan-Kopf bereits belegt, weil er zur Verbindung mit dem SLEC benutzt wird.
- Es müssen die RTC6-Karten Master/Slave verbunden werden, siehe [RTC6-Handbuch](#). Dazu sind 2 Master/Slave-Verbindungskabel erforderlich.
- Es müssen 3 weitere SL2-100 Datenkabel bereitgestellt werden und damit die Verkabelung zwischen den weiteren excelliSCANs und den weiteren RTC6-Karten vorgenommen werden.
- Anmerkung: Das "Installation_Project" aus früher gelieferten Software-Paketen kann (ohne Änderungen) sowohl für Einzel-Kopf- als auch Multi-Head-Systeme verwendet werden. Lediglich die [syncAXISConfig.xml](#), wie sie bisher (unter syncAXIS control \leq V1.1) verwendet wurde, muss zur Verwendung mit syncAXIS control \geq V1.2.4 angepasst werden, siehe [Kapitel 8.2 "Verwendung von syncAXIS control V1.2.4 und höher"](#), Seite 324.
- Es muss für jeden weiteren excelliSCAN Scan-Kopf eine optimierte Korrekturdatei (*.ct5) erstellt werden⁽²⁾.
Anmerkung: in Multi-Head-Systemen mit syncAXIS control \geq V1.2.4 müssen diese *nicht* mehr gewährleisten, dass die Achsen von Scan-Kopf und Verfahrtisch (orthogonal) zur Deckung kommen. Dies kann mittels entsprechender Matrizen in der [syncAXISConfig.xml](#) erreicht werden, siehe [Abbildung 48, Seite 328](#) und [Abbildung 49, Seite 329](#).

- Es muss syncAXIS control \geq V1.2.4 verwendet werden, siehe dazu [Kapitel 8.2 "Verwendung von syncAXIS control V1.2.4 und höher"](#), Seite 324.

Achtung!

Es gibt mit syncAXIS control \geq V1.2.4 erweiterte Möglichkeiten, um auch Systeme komfortabel konfigurieren zu können, in den die System-Komponenten anders relativ zueinander angeordnet sind als in [Abbildung 45, Seite 322](#), dargestellt. Siehe auch [Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher"](#), Seite 332.

(1) 1 weiteres SSHC-Slotblech nötig.

(2) Für Präzisionsergebnisse sind die Korrekturdateien, wie sie werksseitig geliefert werden, nicht geeignet.

8.2 Verwendung von syncAXIS control V1.2.4 und höher

8.2.1 Voraussetzungen für diesen Anhang

- Die excelliSCAN Scan-Köpfe sind installiert und betriebsbereit
- Es gibt eine optimierte Korrekturdatei (*.ct5) für jeden excelliSCAN Scan-Kopf.

8.2.2 syncAXISConfig.xml für syncAXIS control V1.2.4 und höher anpassen

Beachten Sie insbesondere die Sicherheitshinweise in **Handbuch "Installation der SCANLAB XL SCAN-Komponenten und Erstinbetriebnahme des XL SCAN-Systems"**, Kapitel 3 **"Die syncAXISConfig.xml prüfen und anpassen"**! Das Kapitel gilt analog, allerdings muss zur Ansteuerung des hier beschriebenen Multi-Head-XL SCAN-Systems (siehe **Seite 14**) zwingend syncAXIS control \geq V1.2.4 benutzt werden. Daraus ergeben sich mehrere Abweichungen und zusätzliche Schritte:

- syncAXIS control \geq V1.2.4 benutzt keine syncAXISSysConfig.xml mehr.
- Mit syncAXIS control \geq V1.2.4 werden XML-Konfigurationsdateien (syncAXISConfig.xml) nunmehr gegen ein XML-Schema \geq syncAXIS_1_2.xsd validiert⁽¹⁾. Das bedeutet, dass jede syncAXISConfig.xml für syncAXIS control \leq V1.1.n nicht mehr mit syncAXIS control \geq V1.2.4 verwendbar ist. Um diese weiter nutzen zu können, muss der Benutzer diese vorher anpassen. Das Anpassen umfasst die beiden, im Folgenden beschriebenen Schritte (Reihenfolge einhalten!):
 - (1) Technische Anpassung
 - (2) Inhaltliche Anpassung

(1) Dieses unterscheidet sich auch dadurch ganz erheblich von Vorgängerversionen, weil es viel mehr Konfigurationsmöglichkeiten gibt.

Schritt 1 von 2: syncAXISConfig.xml technisch anpassen

Die Struktur der syncAXISConfig.xml ist so zu ändern, dass sie gegen das XML-Schema syncAXIS_1_2.xsd valide ist.

Für diesen Zweck erhalten Sie von SCANLAB im Softwarepaket eine ausführbare Datei *.exe (Diese erzeugt eine syncAXISConfig.xml mit der gültigen Struktur und übernimmt die schon vorhandenen Werte).

Sollten Sie bei der Anpassung Hilfe benötigen, wenden Sie sich an SCANLAB.

Als weitere Hilfestellung finden Sie im Softwarepaket auch ein Konfigurationsbeispiel für ein 2-Kopf-System (unter configuration > syncAXISConfig_MultiHead.Template.xml).

Schritt 2 von 2: syncAXISConfig.xml inhaltlich anpassen

Es sind (wegen der Erweiterungen des XML-Schema syncAXIS_1_2.xsd um Multi-Head-Systeme unterstützen zu können) mehr Konfigurationseinstellungen möglich und nötig. Diese sind inhaltliche Änderungen und müssen deswegen durch Benutzer händisch eingepflegt werden, so z. B. die einzelnen Multi-Head-Komponenten und deren Eigenschaften (z. B. an welchem RTC6-Scan-Kopf-Anschluss ist welches Scan-Device angeschlossen).

Hinweise

- Bei der Initialisierung der syncAXIS control-Instanz müssen
 - alle in der syncAXISConfig.xml im Abschnitt `<cfg:RTCCfg>`, siehe **2**, angegebenen RTC6-Karten verfügbar sein
 - alle unter `<cfg:RTCCfg>` angegebene Scan-Devices auch unter `<cfg:ScanDeviceCfg>` definiert sein
 - der unter `<cfg:RTCCfg>` angegebene Verfahrtsch unter `<cfg:StageCfg>` definiert sein
 - alle Scan-Devices verfügbar sein
 - der Verfahrtsch verfügbar sein
 Alle zusätzlichen (nicht unter `<cfg:RTCCfg>` aufgeführten) Einträge für Geräte unter `<cfg:ScanDeviceCfg>` und `<cfg:StageCfg>` haben keine Auswirkung.

(1) Abschnitt `<cfg:GeneralConfig>`, siehe **Abbildung 46**,
Seite 326:

Der Abschnitt wurde für syncAXIS control
≥ V1.2.4 umstrukturiert.

Gelöscht: `<cfg:SysConfigFilePath>`.

Neu: `<cfg:SimulationConfig>`.

Geändert: `<cfg:LogConfig>` umbenannt (vorher
`<cfg:LogConfiguration>`).

(2) Abschnitt `<cfg:RTCCfg>`, siehe **Abbildung 47**,
Seite 326:

Der Abschnitt wurde für syncAXIS control
≥ V1.2.4 umstrukturiert.

Neu: Es kann mehr als 1 RTC6-Karte eingetragen
werden. Für jede RTC6-Karte muss der Namen
der Scan-Device an Scan-Kopf-Ausgang A und B
angegeben werden.

Versoben: `<cfg:CorrectionFilePath ...>` (jetzt im
Abschnitt `<cfg:ScanDeviceConfig>`).

Hinweis: Welche der RTC6-Karten Master oder
Slave ist, muss nicht eingetragen werden (d. h. die
'erste' RTC6-Karte in `syncAXISConfig.xml` muss nicht
unbedingt der Master sein, sondern kann auch
eine der Slave-Karten sein). Die Master-Karte ist
durch die physikalische Verkabelung bestimmt.

```
<cfg:GeneralConfig>
  <cfg:ACSController>127.0.0.1</cfg:ACSController>
  <cfg:InitialOperationMode>ScannerAndStage</cfg:InitialOperationMode>
  <cfg:InitialListHandlingMode>RepeatWhileBufferFull</cfg:InitialListHandlingMode>
  <cfg:BaseDirectoryPath>${BASE_PATH}</cfg:BaseDirectoryPath>
  <cfg:SimulationConfig>
    <cfg:SimulationMode>true</cfg:SimulationMode>
    <cfg:SimOutputFileDirectory>[BaseDirectoryPath]/Simulate/</cfg:SimOutputFileDirectory>
    <cfg:BinaryOutput>false</cfg:BinaryOutput>
    <cfg:DisableFileOutput>false</cfg:DisableFileOutput>
  </cfg:SimulationConfig>
  <cfg:LogConfig>
    <cfg:LogfilePath>[BaseDirectoryPath]/Log</cfg:LogfilePath>
    <cfg:LogLevel>Warn</cfg:LogLevel>
    <cfg:EnableConsoleLogging>true</cfg:EnableConsoleLogging>
    <cfg:EnableFileLogging>false</cfg:EnableFileLogging>
    <cfg:MaxLogfileSize>26214400</cfg:MaxLogfileSize>
    <cfg:MaxBackupFileCount>0</cfg:MaxBackupFileCount>
  </cfg:LogConfig>
</cfg:GeneralConfig>
```

Wahlweise.

46

syncAXIS control V1.2.4: [syncAXISConfig.xml](#), Beispiel-Abschnitt für allgemeine Einstellungen <cfg:GeneralConfig>. Siehe [Seite 325](#).

```
<cfg:RTCCConfig>
  <cfg:BoardIdentificationMethod>BySerialNumber</cfg:BoardIdentificationMethod>
  <cfg:ProgramFileDirectory>[BaseDirectoryPath]/RTC6</cfg:ProgramFileDirectory>
  <cfg:Boards>
    <cfg:RTC6>
      <cfg:SerialNumber>123457</cfg:SerialNumber>
      <cfg:HeadA>ScanDevice1</cfg:HeadA>
      <cfg:HeadB>Stage1</cfg:HeadB>
    </cfg:RTC6>
    <cfg:RTC6>
      <cfg:SerialNumber>123456</cfg:SerialNumber>
      <cfg:HeadA>ScanDevice2</cfg:HeadA>
      <cfg:HeadB>ScanDevice3</cfg:HeadB>
    </cfg:RTC6>
    <cfg:RTC6>
      <cfg:SerialNumber>123455</cfg:SerialNumber>
      <cfg:HeadA>ScanDevice4</cfg:HeadA>
      <cfg:HeadB>None</cfg:HeadB>
    </cfg:RTC6>
  </cfg:Boards>
</cfg:RTCCConfig>
```

47

syncAXIS control V1.2.4: [syncAXISConfig.xml](#), Beispiel-Abschnitt für die RTC6-Karten <cfg:RTCCConfig>. Siehe [Seite 325](#).

(3) Abschnitt `<cfg:ScanDeviceConfig>`, siehe

Abbildung 48, Seite 328:

Dieser ist ein neuer Abschnitt in syncAXIS control \geq V1.2.4 (Scan-Köpfe werden hier generisch als "Scan-Devices" bezeichnet).

- Es können jetzt mehrere Scan-Devices definiert werden
- Pro Scan-Device sind Name, Matrix, Offset und Korrekturdateien anzugeben
- Außerdem kann für Werkstücke unter jeder dieser Scan-Device eine Matrix und Offset angegeben werden.

Stellen Sie sicher, dass die von Ihnen angegebenen Matrizen invertierbar sind!

(4) Abschnitt `<cfg:StageConfig>`, siehe **Abbildung 49, Seite 329:**

Dieser Abschnitt wurde für syncAXIS control \geq V1.2.4 umbenannt (vorher: `<cfg:StageDynamics>`).

Stellen Sie sicher, dass die von Ihnen angegebenen Matrizen invertierbar sind!

(5) Abschnitt `<cfg:LaserConfig>`

Benutzer müssen hier keine Änderungen für syncAXIS control \geq V1.2.4 vornehmen.

Hinweis: Die Lasersignale werden an beiden RTC6-Karten identisch ausgegeben.

(6) Abschnitt `<cfg:TrajectoryConfig>`

Der Unter-Abschnitt `<cfg:HeuristicConfig>` muss aus diesem Abschnitt `<cfg:TrajectoryConfig>` gelöscht und in den Abschnitt `<cfg:MotionDecompositionConfig>` eingefügt werden, siehe **7**.

(7) Abschnitt `<cfg:MotionDecompositionConfig>`, siehe **Abbildung 50, Seite 330.**

Dieser Abschnitt ist für syncAXIS control \geq V1.2.4 neu und muss von Benutzern entsprechend konfiguriert werden.

Hinweise: Der Unter-Abschnitt

`<cfg:HeuristicConfig>` stammt aus dem Abschnitt

`<cfg:TrajectoryConfig>`, siehe **6**. Der Tag

`<cfg:FilterBandwidth>` entspricht dem Tag

`<cfg:Bandwidth>` in syncAXIS SysConfig.xml für syncAXIS control \leq V1.1.

(8) Abschnitt `<cfg:IOConfig>`

Benutzer müssen hier keine Änderungen für syncAXIS control \geq V1.2.4 vornehmen.

```

<cfg:ScanDeviceConfig>
  <cfg:FieldLimits>
    <cfg:XAxis Unit="mm" Max="150" Min="-150" />
    <cfg:YAxis Unit="mm" Max="150" Min="-150" />
  </cfg:FieldLimits>
  <cfg:DynamicLimits>
    <cfg:Velocity Unit="rad/s">90</cfg:Velocity>
    <cfg:Acceleration Unit="rad/s^2">1.1314e5</cfg:Acceleration>
    <cfg:Jerk Unit="rad/s^3">4e9</cfg:Jerk>
  </cfg:DynamicLimits>
  <cfg:DefaultCorrectionFile>0</cfg:DefaultCorrectionFile>
  <cfg:MaxGalvoAngle Unit="rad">10.3</cfg:MaxGalvoAngle>
  <cfg:FocalLength Unit="mm">160</cfg:FocalLength>
  <cfg:Delay Unit="s">0.00125</cfg:Delay>
  <cfg:ScanDeviceList>
    <cfg:ScanDevice Name="ScanDevice1">
      <cfg:Alignment>
        <cfg:Matrix>
          <cfg:T11>1</cfg:T11>
          <cfg:T12>0</cfg:T12>
          <cfg:T21>0</cfg:T21>
          <cfg:T22>1</cfg:T22>
        </cfg:Matrix>
        <cfg:Offset X="0" Y="0" />
      </cfg:Alignment>
      <cfg:BasePartDisplacement>
        <cfg:Matrix>
          <cfg:T11>1</cfg:T11>
          <cfg:T12>0</cfg:T12>
          <cfg:T21>0</cfg:T21>
          <cfg:T22>1</cfg:T22>
        </cfg:Matrix>
        <cfg:Offset X="0" Y="0" />
      </cfg:BasePartDisplacement>
      <cfg:CorrectionFileList>
        <cfg:CorrectionFilePath CalibrationFactor="-1">D2_584_imprvd_4_SD1_1.ct5</cfg:CorrectionFilePath>
        <cfg:CorrectionFilePath CalibrationFactor="-1">D2_584_factory.ct5</cfg:CorrectionFilePath>
      </cfg:CorrectionFileList>
    </cfg:ScanDevice>
    <cfg:ScanDevice Name="ScanDevice2">
      ...etc. ...
    </cfg:ScanDevice>
  </cfg:ScanDeviceList>
</cfg:ScanDeviceConfig>

```

Wahlweise. Muss i.d.R. nicht geändert werden.

Wahlweise. Muss i.d.R. nicht geändert werden.

Wahlweise. Muss i.d.R. nicht geändert werden.

Name des 1. Scan-Kopfs

Koeffizienten m11...m22 einer (2 × 2)-Transformationsmatrix für den 1. Scan-Kopf

Offset für den 1. Scan-Kopf

Koeffizienten m11...m22 einer (2 × 2)-Transformationsmatrix für Werkstücke, die mit 1. Scan-Kopf bearbeitet werden sollen

Offset für diese Werkstücke

Korrekturdateien (bis zu 4 Tags)

Hinweis: Zur Laufzeit ist eine zusätzliche Transformation über **slsc_cfg_set_part_displacement** möglich.

```

<cfg:StageConfig>
  <cfg:Delay Unit="s">0.0014951</cfg:Delay>
  <cfg:FieldLimits>
    <cfg:XDirection Unit="mm" Max="150" Min="-150" />
    <cfg:YDirection Unit="mm" Max="150" Min="-150" />
  </cfg:FieldLimits>
  <cfg:DynamicLimits>
    <cfg:Velocity Unit="mm/s">1000</cfg:Velocity>
    <cfg:Acceleration Unit="mm/s^2">10000</cfg:Acceleration>
    <cfg:Jerk Unit="mm/s^3">100000</cfg:Jerk>
  </cfg:DynamicLimits>
  <cfg:CalculationDynamics>
    <cfg:Velocity Unit="mm/s">500</cfg:Velocity>
    <cfg:Acceleration Unit="mm/s^2">5000</cfg:Acceleration>
    <cfg:Jerk Unit="mm/s^3">50000</cfg:Jerk>
  </cfg:CalculationDynamics>
  <cfg:StageList>
    <cfg:Stage Name="Stage1">
      <cfg:FieldLimits>
        <cfg:XDirection Unit="mm" Max="150" Min="-150" />
        <cfg:YDirection Unit="mm" Max="150" Min="-150" />
      </cfg:FieldLimits>
      <cfg:DynamicLimits>
        <cfg:Velocity Unit="mm/s">1000</cfg:Velocity>
        <cfg:Acceleration Unit="mm/s^2">10000</cfg:Acceleration>
        <cfg:Jerk Unit="mm/s^3">100000</cfg:Jerk>
      </cfg:DynamicLimits>
      <cfg:CalculationDynamics>
        <cfg:Velocity Unit="mm/s">500</cfg:Velocity>
        <cfg:Acceleration Unit="mm/s^2">5000</cfg:Acceleration>
        <cfg:Jerk Unit="mm/s^3">50000</cfg:Jerk>
      </cfg:CalculationDynamics>
      <cfg:StageAxisX>0</cfg:StageAxisX>
      <cfg:StageAxisY>1</cfg:StageAxisY>
      <cfg:SlecEtherCATNodeID>0</cfg:SlecEtherCATNodeID>
      <cfg:Alignment>
        <cfg:Matrix>
          <cfg:T11>1</cfg:T11>
          <cfg:T12>0</cfg:T12>
          <cfg:T21>0</cfg:T21>
          <cfg:T22>1</cfg:T22>
        </cfg:Matrix>
        <cfg:Offset X="0" Y="0" />
      </cfg:Alignment>
    </cfg:Stage>
    ...etc. ...
  </cfg:StageList>
</cfg:StageConfig>

```

Verpflichtend. Werte werden aus der ursprünglichen xml übernommen.

Verpflichtend. Werte werden aus der ursprünglichen xml übernommen.

Verpflichtend. Diese Werte müssen selber berechnet und händisch eingetragen werden: CalculationDynamics (neu) = ReducedStageDynamicFactor (syncAXISysConfig.xml) × DynamicLimits (syncAXISConfig.xml).

Name des Verfahrtsch

Wahlweise. Überschreibt die globale Tischkonfiguration (d.h. Werte von oben).

Wahlweise. Überschreibt die globale Tischkonfiguration (d.h. Werte von oben).

Wahlweise. Überschreibt die globale Tischkonfiguration (d.h. Werte von oben).

Wahlweise. Für Tischaufbauten mit ACS X-Achsenindex ≠ 0.

Wahlweise. Für Tischaufbauten mit ACS Y-Achsenindex ≠ 1.

Wahlweise. Für Tischaufbauten mit SLEC-ID ≠ 0

Koeffizienten m11...m22 einer (2 × 2)-Transformationsmatrix für den Verfahrtsch

Offset für den Verfahrtsch

```
<cfg:MotionDecompositionConfig>
  <cfg:FilterBandwidth>2</cfg:FilterBandwidth>
  <cfg:HeuristicConfig>
    <cfg:DynamicReductionFunction Units="mm and mm/s">
      <cfg:DataPoint Length="0.0" Velocity="2000" />
      <cfg:DataPoint Length="27.0" Velocity="2000" />
      <cfg:DataPoint Length="27.01" Velocity="700" />
      <cfg:DataPoint Length="54.0" Velocity="700" />
    </cfg:DynamicReductionFunction>
  </cfg:HeuristicConfig>
</cfg:MotionDecompositionConfig>
```

☐ Verpflichtend. Wert w. aus [syncAXISSysConfig.xml](#) übernommen.

50

syncAXIS control ≥ V1.2.4: [syncAXISConfig.xml](#), Beispiel-Abschnitt für Filterbandbreite und **Heuristik**
 <cfg:MotionDecompositionConfig>. Siehe [Seite 327](#).

8.2.3 Weitere Anmerkungen zur Verwendung von syncAXIS control V1.2.4 und höher

- Für syncAXIS control \leq V1.2.4 definierte "Jobs" sind kompatibel zu syncAXIS control \geq V1.2.4 und müssen nicht geändert werden.
- Mit syncAXIS control V1.2.4 kann Software für Multi-Head-Systeme ("Mehrkopfanlagen") entwickelt werden. Die Aufteilung der Bewegungen auf die Scan-Köpfe und den Verfahrtschisch erledigt die syncAXIS-DLL. Code für Multi-Head-Systeme ist in der gleichen Art und Weise zu erstellen wie Code für Einkopf-Systeme (d. h. Softwareentwickler müssen keine grundsätzlichen Unterschiede berücksichtigen).
- syncAXIS control V1.2.4 bietet erweiterte Möglichkeiten, um Scan-Kopf und Verfahrtschisch gegeneinander (auch nach erfolgter Erstellung der optimierten *.ct5-Datei) auszurichten. So besteht u. a. die Möglichkeit (um z. B. Fehler beim Hardware-Aufbau kompensieren zu können) relative Verdrehungen zueinander auszugleichen. Es können sogar Achsinvertierungen vorgenommen werden. Dazu gibt es in der `syncAXISConfig.xml` im
 - Abschnitt `<cfg:ScanDeviceConfig>`, siehe [Abbildung 48, Seite 328](#), die Alignment-Tags Matrix und Offset (für die einzelnen Scan-Devices)⁽¹⁾. Achten Sie darauf, dass diese Matrix invertierbar ist!
 - Abschnitt `<cfg:StageConfig>`, siehe [Abbildung 49, Seite 329](#), die Alignment-Tags Matrix und Offset (für die einzelnen Verfahrtschische)⁽²⁾. Achten Sie darauf, dass diese Matrix invertierbar ist!
- Mit syncAXIS control V1.2.4 gibt es außerdem Möglichkeiten, die Soll-Trajektorie jedes einzelnen Werkstücks – für jedes Scan-Device unabhängig – anzupassen (in `syncAXISConfig.xml` unter `<cfg:Configuration>` → `<cfg:ScanDeviceConfig>` → `<cfg:ScanDeviceList>` → `<cfg:ScanDevice ...>` → `<cfg:BasePartDisplacement>`, siehe [Abbildung 48, Seite 328](#)).
Dadurch ergibt sich ein Basis-Koordinatensystem für die einzelnen Scan-Devices (z. B. um die unterschiedliche Lage verschiedener Spannsysteme auszugleichen).
Siehe auch [Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher"](#), Seite 332.
- Mit `slsc_cfg_set_part_displacement` besteht die Möglichkeit, die Soll-Trajektorie auch zur Laufzeit des Anwenderprogramms individuell für jedes einzelne Werkstück (individuell für jede Scan-Device) über die API zu verändern (muss vor dem Start der Job-Berechnung, d. h. dem Aufruf von `slsc_list_begin*`, erfolgen).
Diese Transformation wirkt zusätzlich zu der in `<cfg:BasePartDisplacement>` eingetragenen Koordinatentransformation.
Dazu müssen lediglich die gewünschten Matrix- und Offset-Werte (z. B. erfasst mittels Bilderkennung) an diese Funktion übergeben werden. Siehe auch [Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher"](#), Seite 332.

(1) Die Umsetzung erfolgt als RTC6-Befehl `set_matrix`.

(2) Die Ansteuerwerte für den Verfahrtschisch werden in der syncAXIS-DLL berechnet.

8.3 Über Transformationen in syncAXIS control V1.2.4 und höher

In syncAXIS control \geq V1.2.4 gibt es verschiedene Möglichkeiten, das Markiermuster abzuändern und die Ansteuerung zu beeinflussen.

Je nach Anwendung (Beispiel: eine Kamerabildauswertung erfasst die tatsächliche Position des Werkstücks) und tatsächlich vorliegendem Aufbau können bestimmte Transformationen nötig werden.

Es gibt in syncAXIS control folgende Transformationsarten, siehe auch [Abbildung 51, Seite 333](#) und [Abbildung 52, Seite 334](#):

- Transformation zum Verändern der Zielpunkt-Koordinaten des eingehenden Musters (schon ab V1.0)
 - Siehe [Abbildung 52, Seite 334](#), (1)
 - Via API mit `slsc_cfg_set_matrix_and_offset`
 - Via API mit `slsc_list_set_matrix_and_offset`
 - Via API mit `slsc_cfg_set_rot_and_offset_2d`
 - Via API mit `slsc_list_set_rot_and_offset_2d`

Die Auswirkungen sind im Simulationsergebnis sichtbar.
- Transformation zum Verändern der Soll-Trajektorie eines Markiermusters für individuelle Scan-Devices (ab V1.2.4)
 - Siehe [Abbildung 51, Seite 333](#), (2)
 - Siehe [Abbildung 52, Seite 334](#), (2)
 - In `syncAXISConfig.xml`, Tag `<cfg:BasePartDisplacement>`, siehe auch [Seite 331](#)
 - Via API mit `slsc_cfg_set_part_displacement`

Die Auswirkungen sind im Simulationsergebnis sichtbar.
- Transformation zum Verändern der Ansteuerwerte für den Verfahrtsch (ab V1.2.4)
 - Siehe [Abbildung 51, Seite 333](#), (3)
 - Siehe [Abbildung 52, Seite 334](#), (3)
 - In `syncAXISConfig.xml`, Tag für `Stage1 -> <Alignment>`

Hiermit besteht die Möglichkeit:

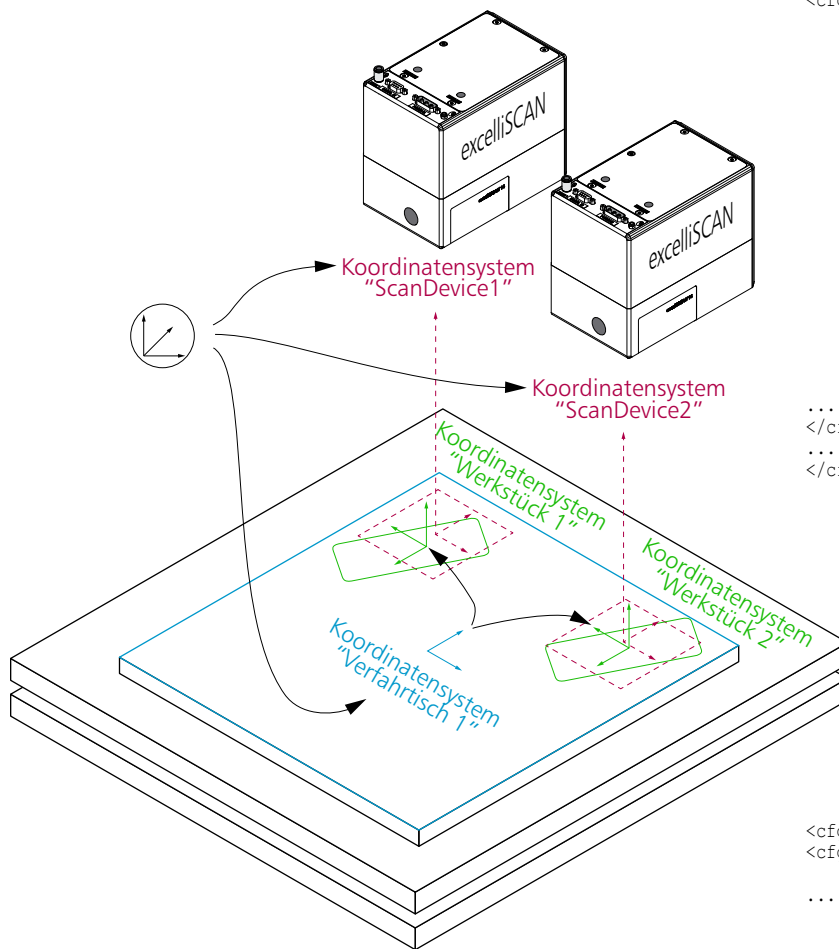
 - die Achsen zu invertieren
 - (in Anlagen mit mehreren-Verfahrtsch) die Verfahrtsche zueinander auszurichten

Die Auswirkungen sind im Simulationsergebnis *nicht* sichtbar.
- Transformation zum Verändern der Ansteuerwerte für das Scan-Device (ab V1.2.4)
 - Siehe [Abbildung 51, Seite 333](#), (4)
 - Siehe [Abbildung 52, Seite 334](#), (4)
 - In `syncAXISConfig.xml`, Tag für `ScanDevice1 -> <Alignment>`

Hiermit besteht die Möglichkeit:

 - das Scan-Device-Koordinatensystem an dem des Verfahrtschs auszurichten (um Montagefehler auszugleichen)
 - die Achsen zu invertieren

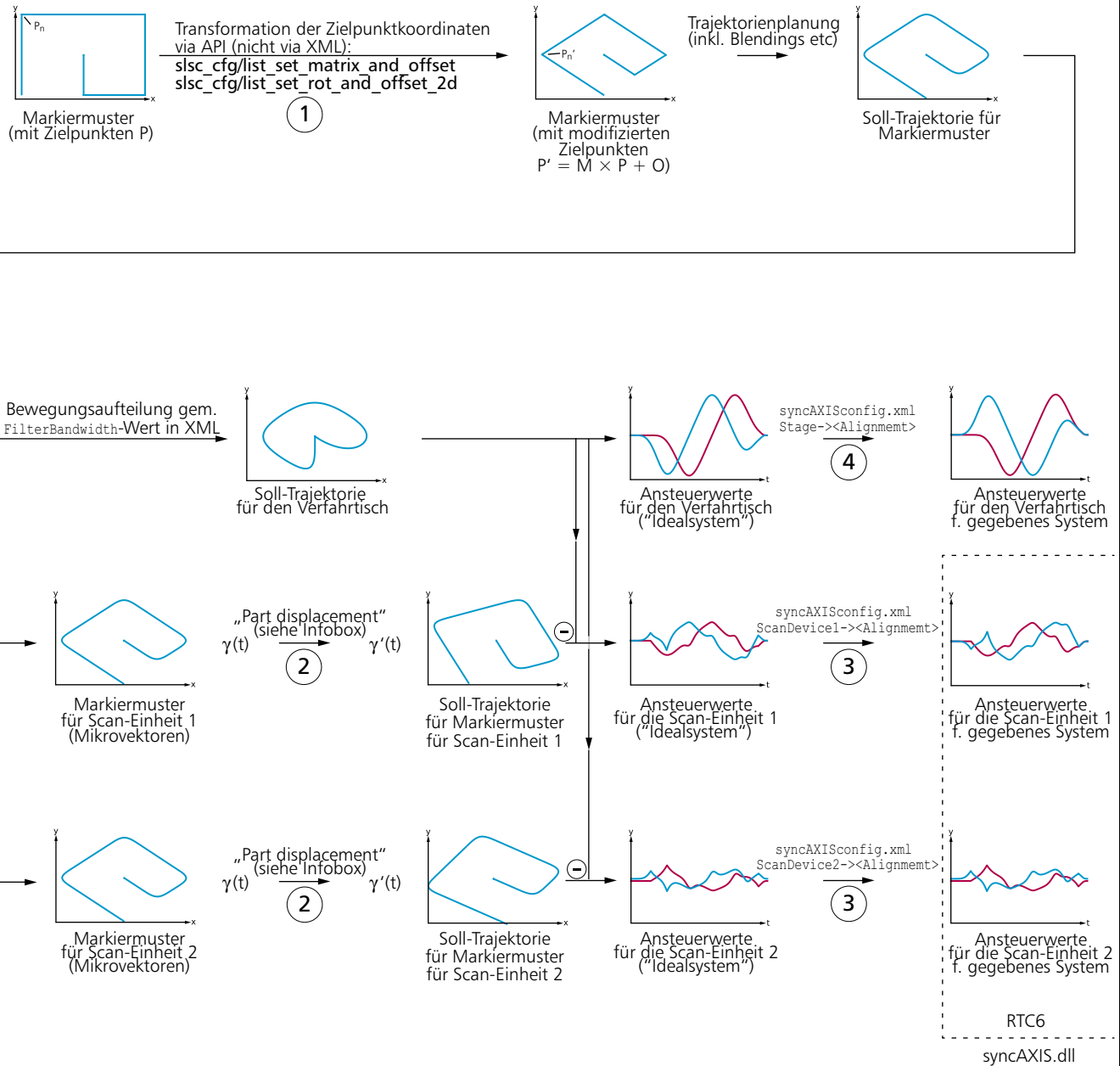
Die Auswirkungen sind im Simulationsergebnis *nicht* sichtbar.



In syncAXISConfig.xml:

```
<cfg:ScanDeviceList>
  <cfg:ScanDevice Name="ScanDevice2">
    <cfg:Alignment>
      <cfg:Matrix>
        <cfg:T11>1</cfg:T11>
        <cfg:T12>0</cfg:T12>
        <cfg:T21>0</cfg:T21>
        <cfg:T22>1</cfg:T22>
      </cfg:Matrix>
      <cfg:Offset X="0" Y="0" />
    </cfg:Alignment>
    <cfg:BasePartDisplacement>
      <cfg:Matrix>
        <cfg:T11>1</cfg:T11>
        <cfg:T12>0</cfg:T12>
        <cfg:T21>0</cfg:T21>
        <cfg:T22>1</cfg:T22>
      </cfg:Matrix>
      <cfg:Offset X="0" Y="0" />
    </cfg:BasePartDisplacement>
  </cfg:ScanDevice>
  ...
</cfg:ScanDeviceList>
```

```
<cfg:StageList>
  <cfg:Stage Name="Stage1">
    <cfg:FieldLimits>...</cfg:FieldLimits>
    ...
    <cfg:Alignment>
      <cfg:Matrix>
        <cfg:T11>1</cfg:T11>
        <cfg:T12>0</cfg:T12>
        <cfg:T21>0</cfg:T21>
        <cfg:T22>1</cfg:T22>
      </cfg:Matrix>
      <cfg:Offset X="0" Y="0" />
    </cfg:Alignment>
  </cfg:Stage>
  ...
</cfg:Stages>
```



Infobox „Part displacement“

$$\gamma'(t) = \begin{matrix} \text{Matrix} \\ \text{Matrix}_{\text{XML}} \times \text{Matrix}_{\text{API}} \end{matrix} \times \gamma + \begin{matrix} \text{+ Offset} \\ \text{Offset}_{\text{XML}} + \text{Offset}_{\text{API}} \end{matrix};$$

wobei $\text{Matrix}_{\text{XML}}$ und $\text{Offset}_{\text{XML}}$ aus dem Tag `<BasePartDisplacement>` in `syncAXISconfig.xml` und $\text{Matrix}_{\text{API}}$ und $\text{Offset}_{\text{API}}$ aus `slsc_cfg_set_part_displacement` stammen.
Beispiel: gibt es keinen Tag `<BasePartDisplacement>`, dann ist $\gamma'(t) = \text{Matrix}_{\text{API}} \times \gamma(t) + \text{Offset}_{\text{API}}$.

9 Anhang B: Anwendungshinweis – Das Handling von Listen mit syncAXIS control

Achtung!

Die Code-Abschnitte dieses Anhangs zeigen nur den notwendigen Mindestsatz an Funktionen, der für ein tatsächlich lauffähiges syncAXIS control-Anwenderprogramm notwendig wäre. Diese Code-Abschnitte dienen lediglich der Veranschaulichung bestimmter Konzepte zur Implementierung von syncAXIS control-basierten Anwenderprogrammen. So beginnt in jedem Beispiel die **Job**-Ausführung entweder, wenn der Puffer voll ist oder wenn der Ausführungsstatus "`slsc_ExecState_ReadyForExecution`" lautet. Im Gegensatz dazu muss der **Job**-Ausführungsbeginn bei echten syncAXIS control-basierten Anwenderprogrammen von mehreren Bedingungen und Sicherheitsvorschriften abhängen. Achten Sie auf die Einhaltung aller relevanter Sicherheitsvorschriften und programmieren Sie Ihren Code entsprechend. Außerdem berücksichtigen die Code-Abschnitte dieses Dokuments keine Fehlerbehandlung. Achten Sie darauf, dass Sie immer den Rückgabewert jeder syncAXIS control-Funktion überwachen und entsprechend reagieren. Sie müssten weiterhin auch bestimmte Parameter der Code-Abschnitte anpassen, um den Anforderungen des tatsächlichen XL SCAN-Systems und des **Jobs** gerecht zu werden.

In XL SCAN-Anlagen werden RTC6-Karten zur Steuerung von Scan-Devices und Laser verwendet. RTC6-Karten speichern Informationen (z. B. Positionen) in ihren Listenspeichern⁽¹⁾ zwischen. Später werden sie in Echtzeit (= alle 10 μ s) abgearbeitet. Der RTC6-Listenspeicher fasst bis zu 8 Millionen RTC6-Listenbefehle.

Bei der Programmierung nur mit der `RTC6DLL.dll` (d. h. ohne syncAXIS control-Software), müssen sich Benutzer selber um das Handling von Listen kümmern. Dafür sind zwei Ansätze häufig:

- Überwachen des Eingabe-Zeigers und des Ausführungs-Zeiger
- Nutzung des "2-Listen-Konzepts" (eine **Liste** füllen, während die andere gerade ausgeführt wird; im stetigen Wechsel).

Bei vielen RTC6-Anwenderprogrammen kann sogar ganz auf ein Listen-Handling verzichtet werden: bei den viel längeren RTC6-Vektorbefehlen reicht der RTC6-Listenspeicher für weit mehr als 40 s Ausführungszeit.

Die syncAXIS-DLL verwendet einen eigenen Funktionssatz für die synchrone Steuerung von Scan-Devices, Verfahrtsch und Laser. Diese syncAXIS control-Funktionen ähneln den RTC6-Mikrovektorbefehlen (siehe **RTC6-Handbuch**), aber jede belegt 2 RTC6-Listenspeicherplätze für 10 μ s-Ausführungszeit.

Wenn der RTC6-Listenspeicher einmalig und vollständig ohne Neuladen verwendet wird, kann (mit seinem Platz für 8 Millionen RTC6-Listenbefehlen) eine maximale Markier-Ausführungszeit von 40 s erreicht werden, siehe auch **Abbildung 11, Seite 41**.

(1) "Listenpuffer".

Die meisten Benutzer wollen auch **Jobs** markieren, die länger als 40 s dauern. Deswegen kann syncAXIS control ein kontinuierliches Beladen von Listen durchführen ("automatisches Listenachladen"). Windows-PCs sind nicht fähig, in Echtzeit zu arbeiten. Dennoch wird, um Pufferunterläufe zu vermeiden, die **Job**-Berechnung und **Job**-Übertragung der **syncAXIS control**-Instanz dergestalt durchgeführt, dass funktionale Blöcke geladen werden und die Planung schneller als die eigentliche **Job**-Ausführung ist.

Deswegen ist es erst nach Abschluss der Verarbeitung des ersten funktionalen Blocks möglich, die **Job**-Ausführung zu starten. Da syncAXIS control mehrere **Jobs** nacheinander berechnen und gleichzeitig **Jobs** an die RTC6-Karte übertragen kann, zielt das "automatische Listenachladen" sowohl auf die Verwendung von mehrerer kleinerer **Jobs** als auch auf einzelne, größere **Jobs**.

Dieser Anhang zeigt beispielhafte Implementierungen in C++ für:

- kleine **Jobs**, ohne Verwendung eines Listen-Handling-Modus, siehe **Abbildung 53, Seite 336**
- große **Jobs**, unter Verwendung der 3 verschiedenen Listen-Handling-Modi
 - "Listen-Handling-Modus "ReturnAtOnce"", Seite 337
 - "Listen-Handling-Modus "RepeatWhileBufferFull"", Seite 339
 - "Listen-Handling-Modus "RepeatWhilePredicate"", Seite 343

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

size_t SLHandle;
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

size_t JobID;
slsc_list_begin(SLHandle, &JobID);
// Hier slsc_list_-Funktionen einfügen.
slsc_list_end(SLHandle);

if (!startSingleJob(SLHandle))
{
    // !Hier eine korrekte Fehlerbehandlung einfügen!
}

slsc_cfg_delete(SLHandle);
```

53

Vereinfachte Codestruktur zum Ausführen kleiner **Jobs** (die den RTC6-Listenspeicher nicht überschreiten).

9.1 Listen-Handling-Modus “ReturnAtOnce”

“ReturnAtOnce” ist derjenige Listen-Handling-Modus, der am ähnlichsten zur RTC6-Programmierung mit der `RTC6DLL.dll` ist (d.h. ohne `syncAXIS-DLL`). Der `syncAXIS-DLL`-interne Puffer ist auf die Größe des RTC6-Listenspeichers beschränkt und kann nicht überschritten werden.

Sobald RTC6-Mikrovektorbefehl 4.000.001 in den RTC6-Listenspeicher geschrieben werden soll, gibt die entsprechende `syncAXIS control`-Funktion einen Rückgabewert $\neq 0$ zurück. Dieser zeigt an, dass der RTC6-Listenspeicher voll ist. Dann liegt es am Benutzer, zunächst Platz im RTC6-Listenspeicher freizugeben, bevor er weitere Job-Funktionen laden kann. Das kann durch Starten einer Job-Ausführung (oder durch Löschen der `syncAXIS control-Instanz`) erreicht werden.

Im Listen-Handling-Modus “ReturnAtOnce” ist es nicht möglich, noch nicht ausgeführte RTC6-Listenbefehle im RTC6-Listenspeicher zu überschreiben.

Das ist der wesentliche Unterschied zur RTC6-Programmierung mit der `RTC6DLL.dll`. Bei dieser wird das Schreiben einfach am Listenspeicheranfang fortgesetzt, wenn der Eingabe-Zeiger den Listenspeicher überschreitet (wodurch noch nicht ausgeführte RTC6-Listenbefehle überschrieben werden können).

Der Hauptvorteil gegenüber Listen-Handling-Modus “RepeatWhileBufferFull” und “RepeatWhilePredicate” besteht darin, dass der Programmablauf nicht an derjenigen `syncAXIS control`-Funktion stehen bleibt, die gerade den RTC6-Listenspeicher überlastet. Dadurch steht es Benutzern frei, Situationen mit vollen RTC6-Listenspeichern nach Belieben zu handhaben.

Ein möglicher Ansatz für die Nutzung des Listen-Handling-Modus “ReturnAtOnce”, siehe [Abbildung 54, Seite 338](#), ist das Laden des Jobs, bis eine `syncAXIS control`-Funktion das entsprechende Fehlerbit zurückgibt. Dann wird die Job-Ausführung gestartet, um etwas RTC6-Listenspeicher freizugeben. Anschließend wird der restliche Job (einschließlich derjenigen `syncAXIS control`-Job-Funktion, die den Rückgabewert $\neq 0$ zurückgegeben hat) stückweise geladen, solange keine Pufferüberläufe angezeigt werden. Sollte dies geschehen, kann der Benutzer einfach warten, bis wieder Listenspeicher frei wurde und das Laden fortsetzen.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

size_t SLHandle;
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");
slsc_cfg_set_list_handling_mode(SLHandle, slsc_ListHandlingMode::slsc_ListHandlingMode_ReturnAtOnce, nullptr);
size_t JobID;

slsc_list_begin(SLHandle, &JobID);

// Timeout anpassen. Wenn der Puffer nach einem bestimmten Timeout nicht freigegeben wird,
// läuft wahrscheinlich kein Job und Sie sollten die Ausführung starten.
size_t Retry_TimeOut = 5;
size_t BufferDelay = 10;

// Für jeden slsc_list_*-Funktionsaufruf (z.B. durch Erstellen von Wrapperfunktionen).
{
    std::array<double, 2> Target{X, Y};
    size_t Retry_Index = 0;
    while ((slsc_list_jump_abs(SLHandle, Target.data()) == 0x0010) && (Retry_TimeOut > Retry_Index))
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(BufferDelay));
        Retry_Index++;
    }
    if (Retry_TimeOut > Retry_Index)
    {
        // Der letzte Puffer ist nun voll. Die Ausführung sollte gestartet werden.
        slsc_ctrl_start_execution(SLHandle);
    }
}

slsc_list_end(SLHandle);

if (!startSingleJob(SLHandle))
{
    // !Hier eine korrekte Fehlerbehandlung einfügen!
}

slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Ausführen größerer **Jobs** (die den RTC6-Listenspeicher überschreiten) und Listen-Handling-Modus `slsc_ListHandlingMode_ReturnAtOnce`.

9.2 Listen-Handling-Modus **“RepeatWhileBufferFull”**

SCANLAB empfiehlt Benutzern, die neu in syncAXIS control sind, den Listen-Handling-Modus **“RepeatWhileBufferFull”** zu verwenden (deshalb wird er auch im **“Installation_Project”** verwendet). Er ist die bequemste Art, mit syncAXIS control große **Jobs** handzuhaben.

Im Listen-Handling-Modus **“RepeatWhileBufferFull”** liest die syncAXIS-DLL die aufgerufenen Job-Funktionen und puffert ihre Weitergabe. Sollte der RTC6-Listenspeicher irgendwann voll sein, dann wartet die syncAXIS-DLL bei der jeweiligen Job-Funktion solange, bis wieder Platz freigegeben ist (z. B. nach einem Start der **Job**-Ausführung).

Beachten Sie allerdings, dass die **syncAXIS control-Instanz** *nicht* in einen Fehlerzustand wechselt, wenn der RTC6-Listenspeicher voll ist. Stattdessen wartet sie stillschweigend darauf, dass dieser vom Benutzer freigegeben wird.

Um ein kontinuierliches **Job**-Nachladen zu erreichen, sind 2 parallele Threads der einfachste Weg:

- (1) einen zum Befüllen des RTC6-Listenspeicher
- (2) einen zum **Job**-Ausführungsstart

Mit **slsc_ctrl_get_exec_state** wird festgestellt, ob der **Job** zum Starten bereit ist.

Im Folgenden ist die Anwendung des Listen-Handling-Modus **“RepeatWhileBufferFull”** zusammen mit parallelen Threads beispielhaft gezeigt:

- Asynchroner Thread 1: Zum Befüllen des RTC6-Listenspeichers, siehe **Abbildung 55, Seite 340**
- Asynchroner Thread 2: Zum Start der **Job**-Ausführung, siehe **Abbildung 56, Seite 341**
- Automatischer **Job**-Start (sobald zur Ausführung bereit) durch Callbacks, siehe **Abbildung 57, Seite 342**

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

size_t SLHandle;
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

slsc_cfg_set_list_handling_mode(SLHandle, slsc_ListHandlingMode::slsc_ListHandlingMode_RepeatWhileBufferFull,
nullptr);

auto ListFilling = std::async(std::launch::async, [SLHandle]()
{
    size_t JobID;
    slsc_list_begin(SLHandle, &JobID);
    // Hier slsc_list_-Funktionen einfügen.
    slsc_list_end(SLHandle);
    return 0;
});

if (!startSingleJob(SLHandle))
{
    // !Hier eine korrekte Fehlerbehandlung einfügen!
}
slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Ausführen größerer **Jobs** (die den RTC6-Listenspeicher überschreiten) und Listen-Handling-Modus **RepeatWhileBufferFull**. Asynchroner Thread 1: Zum Befüllen des RTC6-Listenspeichers.


```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

size_t SLHandle;
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

slsc_cfg_set_list_handling_mode(SLHandle, slsc_ListHandlingMode::slsc_ListHandlingMode_RepeatWhileBufferFull,
nullptr);

slsc_ExecState State = slsc_ExecState_Idle;
auto ListFilling = std::async(std::launch::async, [&]()
{
    while (State != slsc_ExecState_ReadyForExecution)
    {
        slsc_ctrl_get_exec_state(SLHandle, &State);
        std::this_thread::sleep_for(std::chrono::milliseconds(1));
    }
    slsc_ctrl_start_execution(SLHandle);
    return 0;
});

size_t JobID;
slsc_list_begin(SLHandle, &JobID);

// Hier slsc_list_-Funktionen einfügen.

slsc_list_end(SLHandle);
ListFilling.wait();
while (State != slsc_ExecState_Idle)
{
    slsc_ctrl_get_exec_state(SLHandle, &State);
    std::this_thread::sleep_for(std::chrono::milliseconds(1));
}

slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Ausführen größerer **Jobs** (die den RTC6-Listenspeicher überschreiten) und Listen-Handling-Modus **RepeatWhileBufferFull**. Asynchroner Thread 2: Zum Start der **Job**-Ausführung.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

struct Contents
{
    size_t SLHandle;
    uint32_t* Context;
}

size_t SLHandle;
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

slsc_cfg_set_list_handling_mode(SLHandle, slsc_ListHandlingMode::slsc_ListHandlingMode_RepeatWhileBufferFull,
nullptr);

Contents Content = {SLHandle, &Context };

slsc_JobCallback AutoStart = [](size_t JobID, void* Context)
{
    slsc_ExecState State = slsc_ExecState_Idle;

    Contents*Content = static_cast<Contents*>(Context);
    size_t SLHandle = Content->SLHandle;

    while (State != slsc_ExecState_ReadyForExecution)
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(1));
        slsc_ctrl_get_exec_state(SLHandle, &State);
    }
    std::cout << "Start Execution" << std::endl;
    slsc_ctrl_start_execution(SLHandle);

    return;
};

slsc_ExecTimeCallback PrintExecutionFinished = [](size_t JobID, uint64_t Progress, double ExecTime, void* Context)
{
    std::cout << "Execution finished after " << ExecTime << " sec!" << std::endl;
    return;
};

slsc_cfg_register_callback_job_loaded_enough(SLHandle, AutoStart, &Content);
slsc_cfg_register_callback_job_finished_executing(SLHandle, PrintExecutionFinished, &Content);

size_t JobID;
slsc_list_begin(SLHandle, &JobID);
// Hier slsc_list_-Funktionen einfügen.
slsc_list_end(SLHandle);

// Auf Job-Ende warten.
slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Ausführen größerer **Jobs** (die den RTC6-Listenspeicher überschreiten) und Listen-Handling-Modus **RepeatWhileBufferFull**. Automatischer **Job**-Start (sobald zur Ausführung bereit) unter Verwendung von **Callback-Funktionen**.

9.3 Listen-Handling-Modus **"RepeatWhilePredicate"**

Im Listen-Handling-Modus **"RepeatWhilePredicate"** haben Benutzer mehr Freiheiten, aber auch mehr Verantwortung, beim Beladen des RTC6-Listenspeichers. Für diesen Listen-Handling-Modus kann ein Benutzer sein eigenes Prädikat programmieren, das das Verhalten bei vollem RTC6-Listenspeicher definiert.

Im folgendem Beispiel, siehe [Abbildung 58, Seite 343](#), ist das Prädikat so programmiert, dass es den Listen-Handling-Modus **"RepeatWhileBufferFull"** nachahmt. Mit dieser Implementierung kann ein Benutzer z. B. auch die Wartezeit zwischen den einzelnen Ladevorgängen ändern.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

size_t SLHandle;
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

slsc_cfg_set_list_handling_mode(SLHandle, slsc_ListHandlingMode::slsc_ListHandlingMode_RepeatWhilePredicate,
[] (uint32_t RetVal)
{
    bool Flag = (0x0010 & RetVal == 0x0010);
    size_t BufferDelay = 10;
    if (Flag)
    {
        // Anpassen der sleep time zwischen den einzelnen Ladevorgängen
        std::this_thread::sleep_for(std::chrono::milliseconds(BufferDelay));
    }
    return Flag;
});

auto ListFilling = std::async(std::launch::async, [SLHandle]()
{
    size_t JobID;
    slsc_list_begin(SLHandle, &JobID);
    // Hier slsc_list_-Funktionen einfügen.
    slsc_list_end(SLHandle);
    return 0;
});

if (!startSingleJob(SLHandle))
{
    // !Hier eine korrekte Fehlerbehandlung einfügen!
}
slsc_cfg_delete(SLHandle);
```

58

Vereinfachte Codestruktur zum Ausführen größerer **Jobs** (die den RTC6-Listenspeicher überschreiten) und Listen-Handling-Modus **"RepeatWhilePredicate"**.

10 Anhang C: Anwendungshinweis – Texte markieren mittels Modulen







Ein Beispiel zur Anwendung von **Modulen** ist das Markieren von Texten. Mit der vorgestellten Methode besteht kein Risiko für einen **Pufferunterlauf**, obwohl fein definierte Vektorpfade entstehen.

Vorab müssen Sie festlegen, welche Einzel-Zeichen in welcher Schriftart und Größe Sie benötigen werden.

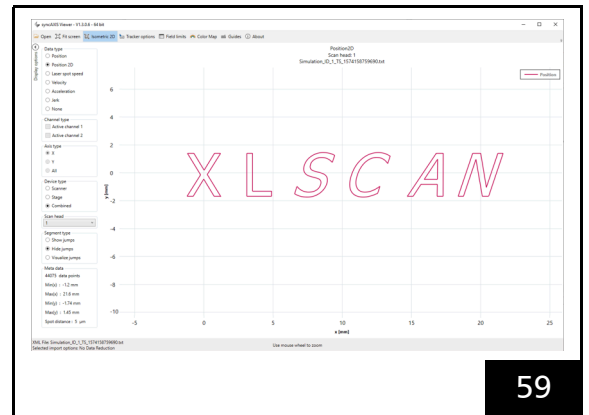
Jedes dieser Zeichen zeichnen Sie als jeweils eigene **Modul-Datei** auf. Der Inhalt jedes **Moduls** entspricht demnach einer Glyph⁽¹⁾ (deren Geometrie ist fest und kann nicht geändert werden).

Da meist mehrere Fonts und Größen verwendet werden sollen, bietet es sich an, diese Informationen in den Dateinamen mit aufzunehmen, z. B.

Name

-  MyFont_12_pt_A.slm
-  MyFont_12_pt_C.slm
-  MyFont_12_pt_L.slm
-  MyFont_12_pt_N.slm
-  MyFont_12_pt_S.slm
-  MyFont_12_pt_X.slm

Zukünftig können Sie die erstellten **Module** beliebig aufrufen, um von Ihnen gewünschte Zeichenketten (erst zu simulieren, siehe **Abbildung 61, Seite 346**, und dann) zu markieren. Im Code-Beispiel **Abbildung 60, Seite 345** wird die Folge "XLSCAN" verwendet.



Simulationsergebnis des Codes aus **Abbildung 61, Seite 346** (angezeigt mit syncAXIS Viewer V1.3).

(1) Die konkrete grafische Darstellung eines Schriftzeichens.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

// Wie in Abbildung 28, Seite 68 wird eine syncAXIS control-Instanz initialisiert, um Module aufzuzeichnen.
// Der Inhalt jedes Moduls ist eine Glyphe, also ein einzelner Buchstabe/Ziffer aus einer bestimmten Schriftart
// und nur in einem bestimmten Schriftgrad.
// Das Vorgehen muss für jede Glyphe in jeder Schriftart und Größe, die später verfügbar sein soll,
// wiederholt werden. Künftig können Sie mit diesen Modulen beliebige Zeichenketten markieren.

size_t JobID = 0;
size_t SLHandle = 0;

std::vector<std::string> Letters = { "X", "L", "S", "C", "A", "N" };
std::array<double, 2> StartPosition{ 0, 0 };

slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

for (std::string Letter : Letters)
{
    std::string Path = ("ModuleAlphabet\\" + Letter + ".slm");
    ModuleRecordingFinished = false;

    slsc_list_begin_module(SLHandle, &JobID, StartPosition.data(), Path.data());

    // Hier die Vektoren der entsprechenden Glyphe einfügen.

    slsc_list_end(SLHandle);

    // Auf den Abschluss jeder Modulaufzeichnung warten.
    if (!startSingleJob(SLHandle))
    {
        // !Hier eine korrekte Fehlerbehandlung einfügen!
    }
}

slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Aufzeichnen mehrerer Module. Hier ist der Modul-Inhalt jeweils eine einzige Glyphe (X, L, S, C, A, N).

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

// Wie in Abbildung 29, Seite 69 wird eine syncAXIS control-Instanz initialisiert um Module abzuspielen.
// In diesem Beispiel werden die Module mit den Glyphen zum Markieren des Texts "XLSCAN" verwendet.
// Zwischen jedem Buchstaben wird ein Versatz (fester Abstand) eingefügt.

size_t SLHandle = 0;
size_t JobID = 0;

std::vector<std::string> Letters = { "X", "L", "S", "C", "A", "N" };
std::array<double, 2> Position{ 0, 0 };
double Spacing = 3.;

slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

slsc_list_begin(SLHandle, &JobID);

for (std::string Letter : Letters)
{
    std::string Path = ("ModuleAlphabet\\" + Letter + ".slm");

    slsc_list_set_rot_and_offset_2d(SLHandle, 0, Position.data());
    Position[0] += Spacing;

    slsc_list_playback_module(SLHandle, Path.data());
}

slsc_list_end(SLHandle);

if (!startSingleJob(SLHandle))
{
    // !Hier eine korrekte Fehlerbehandlung einfügen!
}

slsc_cfg_delete(SLHandle);
```

Vereinfachte Codestruktur zum Abspielen mehrerer Module in Folge um den Text "XLSCAN" zu erzeugen.

11 Anhang D: Anwendungshinweis – Pufferunterlauf vermeiden mittels Modulen

Auf den folgenden Seiten finden Sie kommentierte Code-Abschnitte, die zeigen, wie man **Module** verwenden kann, um einen **Pufferunterlauf** zu vermeiden:

- Teil 1 von 4 Erfolgreiche **Job**-Ausführung erkennen in **Abbildung 62, Seite 347**
- Teil 2 von 4 **Pufferunterlauf** erkennen in **Abbildung 63, Seite 348**
- Teil 3 von 4 **Modul** aufzeichnen in **Abbildung 64, Seite 349**
- Teil 4 von 4 **Modul** abspielen in **Abbildung 65, Seite 349**

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Ausführung eines einzelnen Jobs auslösen, sobald zulässig.
// Dann feststellen, ob die Ausführung erfolgreich war oder nicht.
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
bool startSingleJob(size_t SLHandle)
{
    slsc_ExecState State = slsc_ExecState_Idle;
    while (State != slsc_ExecState_ReadyForExecution)
    {
        uint32_t RetVal = slsc_ctrl_get_exec_state(SLHandle, &State);
        if (RetVal != 0 || State == slsc_ExecState_NotInitOrError)
        {
            return false;
        }
    }
    uint32_t RetVal = slsc_ctrl_start_execution(SLHandle);
    if (RetVal != 0)
    {
        return false;
    }
    while (State != slsc_ExecState_Idle)
    {
        uint32_t RetVal = slsc_ctrl_get_exec_state(SLHandle, &State);
        if (RetVal != 0 || State == slsc_ExecState_NotInitOrError)
        {
            return false;
        }
    }
    return true;
}
```

Vereinfachte Codestruktur, Teil 1 von 4 – Erfolgreiche **Job**-Ausführung erkennen.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

////////////////////////////////////
// Die folgenden Zeilen demonstrieren die Erkennung eines Pufferunterlaufs.
// Mit kurzen Vektoren bei hohen Geschwindigkeiten kann ein Pufferunterlauf
// auftreten, der die Verarbeitung unterbricht und dadurch das Werkstück unbrauchbar macht.
// Um dies zu vermeiden, ist ein möglicher Ansatz, alle Jobs mit ausgeschaltetem
// Laser (z.B. slsc_ctrl_disable_laser) vor der eigentlichen Ausführung zu testen.
////////////////////////////////////
size_t SLHandle = 0;
slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

auto ListFilling = std::async(std::launch::async, [&SLHandle]()
{
    size_t JobID = 0;

    slsc_list_begin(SLHandle, &JobID);
    // Hier die entsprechenden Vektor-Funktionen einfügen.
    slsc_list_end(SLHandle);

    return JobID;
});

bool SuccessfulExecution = startSingleJob(SLHandle);
ListFilling.wait();
bool BufferUnderrunOccurred = false;
// Falls ein Fehler wie z.B. ein Pufferunterlauf vorgekommen ist, könnte die Bearbeitung unvollständig sein.
if (!SuccessfulExecution)
{
    size_t ErrorCount = 0;
    slsc_ctrl_get_error_count(SLHandle, &ErrorCount);

    for (int i = 0; i < ErrorCount; i++)
    {
        uint64_t ErrorCode = 0;
        constexpr static size_t ErrorTextSize = 1000;
        std::array<char, ErrorTextSize> ErrorText;
        slsc_ctrl_get_error(SLHandle, i, &ErrorCode, ErrorText.data(), ErrorText.size());
        BufferUnderrunOccurred |= (ErrorCode == 0x00000000300000001);
    }
}
```

Vereinfachte Codestruktur, Teil 2 von 4 – Pufferunterlauf erkennen.


```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

////////////////////////////////////
// Wenn ein Job wahrscheinlich einen Pufferunterlauf erzeugen wird, empfiehlt sich die Verwendung von
// Modulen um diesen zu vermeiden. Mit slsc_cfg_initialize_copy (≥ V1.3.0) ist es möglich, eine
// syncAXIS control-Instanz im Simulationsmodus in der gleichen Konfiguration wie eine bereits aktive zu
// erzeugen.
// In diesem Beispiel ist eine syncAXIS control-Instanz mit dem Handle SLHandle immer noch aktiv.
////////////////////////////////////
size_t SLHandle = 1; // vom vorherigen Ablauf
size_t JobID = 0;
size_t ModuleHandle = 0;
slsc_cfg_initialize_copy(&ModuleHandle, SLHandle);

std::array<double, 2> Position{ 0, 0 };

// Ersten Job aufzeichnen
slsc_list_begin_module(ModuleHandle, &JobID, Position.data(), "Module.slm");
// Hier die entsprechenden Vektor-Funktionen einfügen.
slsc_list_end(ModuleHandle);

bool ModuleWritingSuccessful = startSingleJob(ModuleHandle);
```

64

Vereinfachte Codestruktur, Teil 3 von 4 – Modul aufzeichnen.

```
// C++-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.

////////////////////////////////////
// Nachdem ein Modul aufgezeichnet wurde, kann es einfach ausgeführt werden, indem eine
// neue syncAXIS control-Instanz initialisiert und das Modul abgespielt wird.
////////////////////////////////////
size_t SLHandle = 0;
size_t JobID = 0;

slsc_cfg_initialize_from_file(&SLHandle, "syncAXISConfig.xml");

// Die Trajektorienkonfiguration ist im Modul gespeichert
// Das Modul abspielen.
slsc_list_begin(SLHandle, &JobID);
slsc_list_playback_module(SLHandle, "Module.slm");
slsc_list_end(SLHandle);

bool SuccessfulExecution = startSingleJob(SLHandle);

if (!SuccessfulExecution)
{
    // !Hier eine korrekte Fehlerbehandlung einfügen!
}

slsc_cfg_delete(SLHandle);
```

65

Vereinfachte Codestruktur, Teil 4 von 4 – Modul abspielen.

12 Anhang E: Anwendungshinweis – Benutzung der syncAXIS-DLL unter C#

Dieses Kapitel erläutert die wesentlichen Unterschiede der syncAXIS-DLL-Funktionen unter C# (im Vergleich zu C).

In diesem Kapitel:

- Unterschiede in den syncAXIS-DLL-Funktionssignaturen, Seite 350
- Unterschiede bei der Verwendung von Callback-Funktionen, Seite 351
- Code-Beispiel 1 (C#), Seite 352
- Code-Beispiel 2 (C#), Seite 357

12.1 Unterschiede in den syncAXIS-DLL-Funktionssignaturen

- Notation der Datentypen, Seite 350
- Zeiger-Ersetzungen für C#, Seite 350

12.1.1 Notation der Datentypen

C, C++	C#
char*	string
uint32_t	uint
uint64_t	ulong

12.1.2 Zeiger-Ersetzungen für C#

C, Beispiel		C#, Beispiel
Zeiger	↔	out
uint32_t slsc_cfg_get_mode(size_t Handle, slsc_OperationMode* Mode);		uint slsc_cfg_get_mode(uint Handle, out slsc_OperationMode Mode);
Zeiger auf Konstante	↔	Array
uint32_t slsc_list_jump_abs(size_t Handle, const double* Target);		uint slsc_list_jump_abs(uint Handle, double[] Target);
Zeiger auf Konstante	↔	Property
uint32_t slsc_cfg_set_trajectory_ config(size_t Handle, const slsc_TrajectoryConfig* TrajConfig);		uint slsc_cfg_set_trajectory_ config(uint Handle, slsc_TrajectoryConfig TrajConfig);
Zeiger auf Zeiger	↔	Property von Property
uint32_t slsc_cfg_get_trajectory_ config(size_t Handle, slsc_TrajectoryConfig** TrajConfig);		slsc_TrajectoryConfig trajectoryConfig = syncAXIS.slsc_cfg_get_tr ajjectory_config(Handle); slsc_BlendModes blendMode = trajectoryConfig.Geometr yConfig.BlendMode;

12.2 Unterschiede bei der Verwendung von Callback- Funktionen

Callback-Funktionen funktionieren in C und C# grundsätzlich gleich. Beim Programmieren in C# müssen Sie allerdings Besonderheiten beachten:

- Callbacks werden in C# durch Vererbung der Callback-Klassen implementiert, siehe [Code-Beispiel 1 \(C#\), Seite 352](#)
- Grundsätzlich wird eine angegebene **Callback-Funktion** bei einem **Callback-Event** des jeweiligen Typs durch Override der `Run()` Methode implementiert
- Objekte, auf die beim **Callback-Event** (also beim Aufruf der `Run()` Methode) zugegriffen werden soll, sind als Member-Variablen der abgeleiteten Callback Klasse anzulegen. Diese können z. B. über einen Konstruktor initialisiert werden.
- Die benötigten Callback-“Rückgabewerte” könnten Sie in der abgeleiteten Klasse realisieren durch (alternativ):
 - Benutzerdefinierte Methoden
 - Properties

12.3 Code-Beispiel 1 (C#)

// C#-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf [Seite 16](#).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Threading;

namespace MyDemo
{
    class DemoCode
    {
        static void Main(string[] args)
        {
            string xmlFilePath = Directory.GetCurrentDirectory() + "\\..\\..\\..\\syncAXISConfig.xml";
            uint handle = 0;
            Console.WriteLine(xmlFilePath);

            uint retVal = syncAXIS.slsc_cfg_initialize_from_file(out handle, xmlFilePath);

            // Example - public static uint slsc_cfg_set_list_handling_mode_with_context(uint Handle,
            //                               slsc_ListHandlingMode mode, ListHandlingCallback internal_CALLBACK);
            slsc_ListHandlingMode mode = slsc_ListHandlingMode.slsc_ListHandlingMode_RepeatWhilePredicate;
            if (mode == slsc_ListHandlingMode.slsc_ListHandlingMode_RepeatWhilePredicate)
            {
                MyListHandlingPredicate listHandlingPredicate = new MyListHandlingPredicate(retVal);
                retVal |= syncAXIS.slsc_cfg_set_list_handling_mode_with_context(handle, mode, listHandlingPredicate);
            }
            else
            {
                // Default constructor
                ListHandlingCallback listHandlingCallback = new ListHandlingCallback();
                retVal |= syncAXIS.slsc_cfg_set_list_handling_mode_with_context(handle, mode, listHandlingCallback);
            }

            // Example - public static slsc_TrajectoryConfig slsc_cfg_get_trajectory_config(uint Handle);
            // No deletion needed.
            slsc_TrajectoryConfig trajectoryConfig = syncAXIS.slsc_cfg_get_trajectory_config(handle);
            Console.WriteLine("Original GeometryConfig Blend Mode = " + trajectoryConfig.GeometryConfig.BlendMode);

            // Example - public static uint slsc_cfg_set_trajectory_config(uint Handle,
            //                               slsc_TrajectoryConfig TrajConfig);
            trajectoryConfig.GeometryConfig.BlendMode = slsc_BlendModes.slsc_BlendModes_FixedBlending;
            syncAXIS.slsc_cfg_set_trajectory_config(handle, trajectoryConfig);
            Console.WriteLine("New GeometryConfig Blend Mode = " + trajectoryConfig.GeometryConfig.BlendMode);
        }
    }
}
```



```
// Example - public static uint slsc_cfg_register_callback_job_start_planned(uint Handle,
//                               JobCallback internal_CALLBACK);
Job_start_planned planningStarted = new Job_start_planned(handle);
retVal |= syncAXIS.slsc_cfg_register_callback_job_start_planned(handle, planningStarted);

// Example - public static uint slsc_cfg_register_callback_job_loaded_enough(uint Handle,
//                               JobCallback internal_CALLBACK);
AutoStart starter = new AutoStart(handle);
retVal |= syncAXIS.slsc_cfg_register_callback_job_loaded_enough(handle, starter);

// Example - public static uint slsc_cfg_register_callback_job_finished_executing(uint Handle,
//                               execTimeCallback internal_CALLBACK);
WaitForFinished waiter = new WaitForFinished();
retVal |= syncAXIS.slsc_cfg_register_callback_job_finished_executing(handle, waiter);

uint jobID = 1;

retVal |= syncAXIS.slsc_list_begin(handle, out jobID);

// Simple Job
double[] lowerMid = new double[] { 0.0, 0.0 };
double[] lowerRightCorner = new double[] { 10.0, 0.0 };
double[] upperRightCorner = new double[] { 10.0, 10.0 };
double[] upperLeftCorner = new double[] { -10.0, 10.0 };
double[] lowerLeftCorner = new double[] { -10.0, 0.0 };
retVal |= syncAXIS.slsc_list_jump_abs(handle, lowerMid);
retVal |= syncAXIS.slsc_list_mark_abs(handle, lowerRightCorner);
retVal |= syncAXIS.slsc_list_mark_abs(handle, upperRightCorner);
retVal |= syncAXIS.slsc_list_mark_abs(handle, upperLeftCorner);
retVal |= syncAXIS.slsc_list_mark_abs(handle, lowerLeftCorner);
retVal |= syncAXIS.slsc_list_mark_abs(handle, lowerMid);

retVal |= syncAXIS.slsc_list_end(handle);

while (waiter.GetFinishedState() == false)
{
    Thread.Sleep(100);
}

Console.WriteLine("Return Value = " + retVal);

WriteError(handle);

syncAXIS.slsc_cfg_delete(handle);

Console.ReadKey();
}
```

```
// Example - ListHandlingCallback internal_CALLBACK
public class MyListHandlingPredicate : ListHandlingCallback
{
    uint returnVal;
    bool flag;
    uint bufferFull = 0x0010;

    public MyListHandlingPredicate(uint returnValIn)
    {
        returnVal = returnValIn;
    }
    public override bool Run(uint returnValIn)
    {
        uint indicator = bufferFull & returnVal;

        flag = (indicator == bufferFull);

        if (flag)
        {
            Console.WriteLine("Buffer is full. List loading waits for free buffer.");
            return flag;
        }
        else
        {
            return flag;
        }
    }
}

// Example - JobCallback internal_CALLBACK callback_job_start_planned
public class Job_start_planned : JobCallback
{
    uint handle;

    public Job_start_planned(uint handleIn)
    {
        handle = handleIn;
    }

    public override void Run(uint jobID)
    {
        Console.WriteLine("Planning started.");
    }
}
```

```
// Example - JobCallback internal_CALLBACK callback_job_loaded_enough
public class AutoStart : JobCallback
{
    uint handle;

    public AutoStart(uint handleIn)
    {
        handle = handleIn;
    }

    public override void Run(uint jobID)
    {
        slsc_ExecState rtc6State;
        uint retVal;

        retVal = syncAXIS.slsc_ctrl_get_exec_state(handle, out rtc6State);
        if (retVal != 0)
        {
            Console.WriteLine("An Error occurred after slsc_ctrl_get_exec_state, return value = " + retVal);
        }
        while (rtc6State != slsc_ExecState.slsc_ExecState_ReadyForExecution)
        {
            retVal = syncAXIS.slsc_ctrl_get_exec_state(handle, out rtc6State);
            if (retVal != 0)
            {
                Console.WriteLine("An Error occurred after slsc_ctrl_get_exec_state, return value = " + retVal);
            }
            Thread.Sleep(10);
            Console.WriteLine("Waiting 10 ms for execution ready to run...");
        }

        retVal = syncAXIS.slsc_ctrl_start_execution(handle);
        if (retVal != 0)
        {
            Console.WriteLine("An Error occurred after slsc_ctrl_start_execution. Return value = " + retVal);
        }
        else
        {
            Console.WriteLine("Execution started.");
        }
    }
}
```

```
// Example - JobCallback internal_CALLBACK callback_job_finished_executing
public class WaitForFinished : ExecTimeCallback
{
    bool isFinished = false;

    public bool GetFinishedState()
    {
        return isFinished;
    }

    public override void Run(uint jobID, ulong progress, double execTime)
    {
        isFinished = true;
        Console.WriteLine("Execution finished. Execution time: " + execTime + " sec.");
    }
}

private static void WriteError(uint handle)
{
    uint count = 0;
    syncAXIS.slsc_ctrl_get_error_count(handle, out count);
    if (count > 0)
    {
        Console.WriteLine(count + " errors detected.");
        for (uint i = 0; i < count; i++)
        {
            ulong ErrorNr = 0;
            string ErrorMessage = "";
            syncAXIS.slsc_ctrl_get_error(handle, i, out ErrorNr, out ErrorMessage);
            Console.WriteLine("ErrorMessage: " + ErrorMessage);
        }
    }
    else
    {
        Console.WriteLine("No error occurred!");
    }
    return;
}
}
```


12.4 Code-Beispiel 2 (C#)

```
// C#-Code-Abschnitt nur zu Schulungszwecken. Führen Sie diesen Code niemals ohne vorhergehende
// Anpassung und Simulation auf echten XL SCAN-Systemen aus!
// Beachten Sie die Sicherheitshinweise und den Haftungsausschluss auf Seite 16.
// Beispiel MultiPara über slsc_list_multi_para_arc_abs
// = mehrteilige (komplexere) Rampe
// Es sind keine Blending-Kurven (blending curves) eingestellt.
// Pseudo-Code (nicht vollständig)

uint jobID = 0;
syncAXIS.slsc_list_begin(handle, out jobID);
double[] TargetPosition_0 = new double[] { 0.0, 0.0 };
double[] TargetPosition_1 = new double[] { 1.0, 1.0 };
double[] TargetPosition_2 = new double[] { 1.025, 1.025 };
double[] TargetPosition_3 = new double[] { 2.0, 1.0 };
double[] TargetPosition_4 = new double[] { 3.0, 0.0 };

syncAXIS.slsc_list_jump_abs(handle, TargetPosition_0);
syncAXIS.slsc_list_mark_abs(handle, TargetPosition_1);
double[] TargetParaDefault = new double[] { 1.0 };
syncAXIS.slsc_list_para_enable(handle, TargetParaDefault);

// Hier wird erstellt: ArrayList vom Typ slsc_ParaChannel.
var Channell = new slsc_ParaChannel();
Channell.Add(new slsc_ParaSection { ds = 0.25, ParaTarget = 0.5 });
Channell.Add(new slsc_ParaSection { ds = 0.6186678697087737, ParaTarget = 0.5 });
Channell.Add(new slsc_ParaSection { ds = 0.25, ParaTarget = 1.0 });

// Hier wird erstellt: ArrayList vom Typ slsc_MultiParaChannels (ActiveChannel == 1).
var Paras = new slsc_MultiParaChannels();
Paras.Add(Channell);
syncAXIS.slsc_list_multi_para_arc_abs(handle, TargetPosition_2, TargetPosition_3, Paras);

syncAXIS.slsc_list_mark_abs(handle, TargetPosition_4);
syncAXIS.slsc_list_jump_abs(handle, TargetPosition_0);

syncAXIS.slsc_list_end(handle);
```

13 Anhang F: Referenz der syncAXISConfig.xml-Tags

13.1 xml-Struktur-Übersicht

- **Legende**
 - (*) Dieser Tag ist optional
 - [] Dieser Tag darf mehr als 1× vorkommen
 - {} 2 sich ausschließende Bedeutungen: Dieser Tag erlaubt entweder genau 1 Child-Tag ("choice") oder beliebig viele ("sequence")
 - Container, STANDARD, STANDARD***, NON-ST'D, NON-ST'D*** ist das **Modul-Abspiel-Verhalten**, **Seite 66** des Tags

Configuration	Container	ScanDeviceConfig	Container
GeneralConfig	Container	DynamicLimits	Container
ACSController(*)	STANDARD	Velocity	NON-ST'D***
InitialOperationMode	STANDARD***	Acceleration	NON-ST'D***
InitialListHandlingMode	STANDARD	Jerk	NON-ST'D***
DynamicViolationReaction	STANDARD	CalculationDynamics	Container
LogConfig	Container	MarkDynamics	Container
LogfilePath	STANDARD	Acceleration	NON-ST'D
LogLevel	STANDARD	Jerk	NON-ST'D
EnableConsoleLogging	STANDARD	JumpDynamics	Container
EnableFileLogging	STANDARD	Acceleration	NON-ST'D
MaxLogfileSize(*)	STANDARD	Jerk	NON-ST'D
MaxBackupFileCount(*)	STANDARD	FieldLimits	Container
BaseDirectoryPath(*)	STANDARD	XDirection	STANDARD
SimulationConfig	Container	YDirection	STANDARD
SimulationMode	STANDARD	ZDirection(*)	STANDARD
SimOutputFileDirectory(*)	STANDARD	MonitoringLevel	STANDARD
BinaryOutput(*)	STANDARD	FocalLength(*)	STANDARD
DisableFileOutput(*)	STANDARD	Delay(*)	STANDARD***
RTCCConfig	Container	ScanDeviceList	Container
BoardIdentificationMethod	STANDARD	ScanDevice[]	STANDARD
ProgramFileDirectory	STANDARD	CorrectionFileList	Container
Boards	Container	CorrectionFilePath[]	STANDARD
RTC6[]	Container	Alignment	Container
SerialNumber(*)	STANDARD	Matrix	Container
HeadA	STANDARD	T11	STANDARD
HeadB	STANDARD	T12	STANDARD
EthSearch{ }(*)	Container	T21	STANDARD
Broadcast	Container	T22	STANDARD
IP	STANDARD	Offset	STANDARD
NetMask	STANDARD	BasePartDisplacement	Container
IPScan	Container	Matrix	Container
StartIp	STANDARD	T11	STANDARD
EndIp	STANDARD	T12	STANDARD
IPList	Container	T21	STANDARD
IPAddress[]	STANDARD	T22	STANDARD
EthMaxTimeout(*)	STANDARD	Offset	STANDARD
		DefaultCorrectionFile	STANDARD

LaserConfig	Container	TrajectoryConfig	Container
LaserMode ^(*)	STANDARD	MarkConfig	Container
LaserPortCfg ^(*)	Container	JumpSpeed	NON-ST'D
LaserOn	STANDARD	MarkSpeed	NON-ST'D
Laser1	STANDARD	MinimalMarkSpeed	NON-ST'D
Laser2	STANDARD	LaserSwitchConfig ^(*)	Container
LaserOutput ^(*)	STANDARD	LaserPreTriggerTime	STANDARD***
LaserStandby ^(*)	STANDARD	LaserSwitchOffsetTime	STANDARD
QSwitchDelay ^(*)	STANDARD	LaserMinOffTime ^(*)	NON-ST'D
FPulseKillerLength ^(*)	STANDARD	GeometryConfig	Container
LaserControlFlags ^(*)	Container	MaxBlendRadius	NON-ST'D
LaserDisable ^(*)	STANDARD	ApproxBlendLimit	NON-ST'D
PulseSwitchSetting ^(*)	STANDARD	BlendMode	NON-ST'D
LaserSignalPhaseShift ^(*)	STANDARD	AutoCyclicGeometry	NON-ST'D
LaserOnSignalActiveLow ^(*)	STANDARD	SplineConversionLengthLimit	NON-ST'D
Laser1Laser2SignalActiveLow ^(*)	STANDARD	SplineMode	NON-ST'D
LaserPulsesAtRisingEdge ^(*)	STANDARD	VectorResolution	STANDARD
OutputSynchronizationOn ^(*)	STANDARD	StageConfig	Container
AutomaticLaserControl	Container	DelayShift ^(*)	STANDARD***
ActiveChannel	Container	CTIME ^(*)	STANDARD***
Channel[] ^(*)	STANDARD***	MonitoringLevel	STANDARD
AnalogOut1 ^(*)	STANDARD	StageList ^(*)	Container
Shift	STANDARD	Stage[]	STANDARD
RadiusFactor	STANDARD	FieldLimits	Container
DataPoint[] ^(*)	STANDARD	XDirection	STANDARD
VelocityFactor	STANDARD	YDirection	STANDARD
DataPoint[] ^(*)	STANDARD	ZDirection ^(*)	STANDARD
AnalogOut2 ^(*)	STANDARD	DynamicLimits	Container
Shift	STANDARD	Velocity	STANDARD
RadiusFactor	STANDARD	Acceleration	STANDARD
DataPoint[] ^(*)	STANDARD	Jerk	STANDARD
VelocityFactor	STANDARD	CalculationDynamics	Container
DataPoint[] ^(*)	STANDARD	Velocity	NON-ST'D***
PulseLength ^(*)	STANDARD	Acceleration	NON-ST'D***
Shift	STANDARD	Jerk	NON-ST'D***
RadiusFactor	STANDARD	Alignment	Container
DataPoint[] ^(*)	STANDARD	Matrix	Container
VelocityFactor	STANDARD	T11	STANDARD
DataPoint[] ^(*)	STANDARD	T12	STANDARD
HalfPeriod ^(*)	STANDARD	T21	STANDARD
Shift	STANDARD	T22	STANDARD
RadiusFactor	STANDARD	Offset	STANDARD
DataPoint[] ^(*)	STANDARD	StageAxisX ^(*)	STANDARD
VelocityFactor	STANDARD	StageAxisY ^(*)	STANDARD
DataPoint[] ^(*)	STANDARD	SlecEtherCATNodeID ^(*)	STANDARD
SpotDistance ^(*)	STANDARD		
Shift	STANDARD		
RadiusFactor	STANDARD		
DataPoint[] ^(*)	STANDARD		
VelocityFactor	STANDARD		
DataPoint[] ^(*)	STANDARD		

IOConfig ^(*)	Container
DefaultOutputs ^(*)	Container
LaserPinOut ^(*)	STANDARD
AnalogOut1 ^(*)	STANDARD
AnalogOut2 ^(*)	STANDARD
LaserInitSequence{ } ^(*)	Container
Delay	STANDARD
SetLaserPinOut	STANDARD
SetAnalogOut1	STANDARD
SetAnalogOut2	STANDARD
SetExt1DigitalOut	STANDARD
LaserShutdownSequence{ } ^(*)	Container
Delay	STANDARD
SetLaserPinOut	STANDARD
SetAnalogOut1	STANDARD
SetAnalogOut2	STANDARD
SetExt1DigitalOut	STANDARD
MotionDecompositionConfig	Container
FilterBandwidth	STANDARD
HeuristicConfig ^(*)	Container
DynamicReductionFunctions ^(*)	Container
DynamicReductionFunction[] ^(*)	Container
DataPoint[]	NON-ST'D
HeuristicForJumpsOnly ^(*)	NON-ST'D
SystemConfig* INTERN - NICHT VERWENDEN	

13.2 xml-Tags

XML-Tag	Configuration
XML-Signatur (mit Defaults)	Configuration '*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:Configuration xmlns:cfg="syncAXIS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="cfg syncAXIS_1_8.xsd" Version="1.7"> <!-- erlaubte/mögliche Child-Tags siehe Configuration in der XML-Struktur-Übersicht --> </cfg:Configuration></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> Configuration ist der oberste Container-Tag in syncAXISConfig.xml.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	GeneralConfig
XML-Signatur (mit Defaults)	GeneralConfig '*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:GeneralConfig> <!-- erlaubte/mögliche Child-Tags siehe GeneralConfig in der XML-Struktur-Übersicht --> </cfg:GeneralConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> GeneralConfig ist nur ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	ACSController
XML-Signatur (mit Defaults)	<p>ACSController* = 127.0.0.1</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>ACSController-Wert: String (Format 'N.N.N.N'). <code><cfg:Configuration></code> → <code><cfg:GeneralConfig></code> → <code><cfg:ACSController></code></p>
XML-Pfad(e)	<code><cfg:Configuration></code> → <code><cfg:GeneralConfig></code> → <code><cfg:ACSController></code>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<code><cfg:ACSController>127.0.0.1</cfg:ACSController></code>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Setzt die IP-Adresse des ACS Motion-Controllers im EtherCAT-Netzwerk. • Stellen Sie vor dem ersten Start sicher, dass der ACSController-Wert eingetragen ist. • Auch für den Simulationsmodus muss der ACSController-Wert ein gültiger Eintrag sein. • Siehe auch <code>0x 00 00 00 06 00 00 00 01 INIT_ACS_TCPIP</code>.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	InitialOperationMode
XML-Signatur (mit Defaults)	InitialOperationMode = <code>ScannerAndStage</code>
	<p><code>'**'</code>=wahlweise; kein <code>'*'</code>=Pflicht.</p> <p>InitialOperationMode-Wert: String.</p>
XML-Pfad(e)	<code><cfg:Configuration></code> → <code><cfg:GeneralConfig></code> → <code><cfg:InitialOperationMode></code>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht" , Seite 358.
XML-Abschnitt-Beispiel	<code><cfg:InitialOperationMode>ScannerAndStage</cfg:InitialOperationMode></code>
Über API setzbar?	<code>slsc_cfg_set_mode</code>
Modul-Abspiel-Verhalten	Der Parameter-Wert der abspielenden <code>syncAXIS control</code> -Instanz wird angewendet. Ausnahmen: Siehe Tabelle Seite 66. Siehe auch Abschnitt "Modul-Abspiel-Verhalten" , Seite 66.
Kommentar(e)	<ul style="list-style-type: none"> Setzt den anfänglichen Betriebsmodus der <code>syncAXIS control</code>-Instanz, siehe auch enum <code>slsc_OperationMode</code>. Erlaubte Einträge: <ul style="list-style-type: none"> <code>ScannerOnly</code> <code>StageOnly</code> <code>ScannerAndStage</code> Der Betriebsmodus ist eine grundlegende Einstellung für die Trajektorienplanung und definiert die Art der Bewegung.
Versionsinfo	<code>syncAXIS_1_8.xsd</code>

XML-Tag	InitialListHandlingMode
XML-Signatur (mit Defaults)	InitialListHandlingMode = ReturnAtOnce
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>InitialListHandlingMode-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:InitialListHandlingMode>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht" , Seite 358.
XML-Abschnitt-Beispiel	<cfg:InitialListHandlingMode>RepeatWhileBufferFull</cfg:InitialListHandlingMode>
Über API setzbar?	slsc_cfg_set_list_handling_mode slsc_cfg_set_list_handling_mode_with_context
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Setzt den anfänglichen Listen-Handling-Modus der syncAXIS control-Instanz, siehe auch enum slsc_ListHandlingMode. Diese Einstellung gibt an, wie die Pufferung der Job-Funktionen (slsc_list_*) behandelt wird. Erlaubte Einträge: <ul style="list-style-type: none"> ReturnAtOnce RepeatWhileBufferFull Achtung: "RepeatWhilePredicate" ist nur über die API setzbar! Im Listen-Handling-Modus "RepeatWhileBufferFull" oder "RepeatWhilePredicate" versucht die syncAXIS control-Instanz ständig, Positionen auf die RTC6-Karte zu laden und wartet andauernd darauf, dass der Puffer freigegeben wird, sobald er voll ist (was durch Starten der Job-Ausführung erreicht werden kann). <i>Achten Sie daher darauf, dass Sie die Liste asynchron zum Job-Start-Thread befüllen!</i> Im Listen-Handling-Modus "ReturnAtOnce" wird der Code 0x10 mit derjenigen Funktion zurückgegeben, die den Puffer überlastet.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DynamicViolationReaction
XML-Signatur (mit Defaults)	DynamicViolationReaction = WarningOnly
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>DynamicViolationReaction-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:DynamicViolationReaction>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:DynamicViolationReaction>AbortImmediately</cfg:DynamicViolationReaction>
Über API setzbar?	slsc_cfg_set_dynamic_violation_reaction
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert, welche Reaktion automatisch erfolgt, sobald ein bestimmtes Überwachungskriterium (Dynamikgrenzen, Arbeitsfeldgrenzen) verletzt wird. Gilt sowohl für den Hardwaremodus als auch den Simulationsmodus. Hinweis: Die Überwachungskriterien für DynamicViolationReaction werden für Scan-Devices und Verfahrtsche getrennt eingestellt unter: <ul style="list-style-type: none"> <cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:MonitoringLevel> <cfg:Configuration> → <cfg:StageConfig> → <cfg:MonitoringLevel> Erlaubte Einträge: <ul style="list-style-type: none"> WarningOnly AbortImmediately StopAndReport syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als: <ul style="list-style-type: none"> Scan-Device-Dynamikgrenzen die DynamicLimits-Werte, Seite 388 Scan-Device-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 395 Scan-Device-Überwachungskriterium den MonitoringLevel-Wert, Seite 397 Verfahrtsch-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 454 Verfahrtsch-Dynamikgrenzen die DynamicLimits-Werte, Seite 456 Verfahrtsch-Überwachungskriterium den MonitoringLevel-Wert, Seite 452 Reaktion auf Überschreitungen den DynamicViolationReaction-Wert, Seite 365
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LogConfig
XML-Signatur (mit Defaults)	LogConfig
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:LogConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:LogConfig> <!-- erlaubte/mögliche Child-Tags siehe LogConfig in der XML-Struktur-Übersicht --> </cfg:LogConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> GeneralConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LogfilePath
XML-Signatur (mit Defaults)	LogfilePath = Error.log
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>LogfilePath-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:LogConfig> → <cfg:LogfilePath>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LogfilePath>[BaseDirectoryPath]/Log</cfg:LogfilePath>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Pfad der Log-Datei, die von der syncAXIS control-Instanz erstellt wird, wenn die Datei-protokollierung angeschaltet ist (siehe EnableFilelogging). • Zu [BaseDirectoryPath] siehe BaseDirectoryPath. • Wenn die Datei bereits vorhanden ist, werden die Log-Informationen der Datei hinzugefügt.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LogLevel
XML-Signatur (mit Defaults)	LogLevel = Warn
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>LogLevel-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:LogConfig> → <cfg:LogLevel>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LogLevel>Warn</cfg:LogLevel>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Der LogLevel-Wert legt die Protokollierungsstufe ("Log-Level") fest, siehe Kapitel 2.8 "Über das Logging in syncAXIS control", Seite 47. Erlaubte Einträge: <ul style="list-style-type: none"> Error Erzeugt werden: [ERROR]-Log-Dateizeilen Warn Erzeugt werden: [ERROR]-Log-Dateizeilen [WARN]-Log-Dateizeilen Info Erzeugt werden: [ERROR]-Log-Dateizeilen [WARN]-Log-Dateizeilen [INFO]-Log-Dateizeilen [ERROR]-Log-Dateizeilen zeigen an, dass das System in einem Fehlerzustand ist. Log-Dateizeilen-Beispiele: <pre> 19-10-04 18:16:10:669 [ERROR] 18:16:10.669762 ErrorCode: 0x0000000500000001 ErrorMessage: "Init failed!" 19-09-19 16:14:21:930 [WARN] Scanner dynamic breached or violated position limits: 0. Regarding Job 1 in Segment 0 19-10-08 18:16:30:508 [INFO] RTC6-SLEC establish handshake, activating stage </pre>
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	EnableConsoleLogging
XML-Signatur (mit Defaults)	EnableConsoleLogging = false
	'*' = wahlweise; kein '*' = Pflicht. EnableConsoleLogging-Wert: Boolean.
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:LogConfig> → <cfg:EnableConsoleLogging>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:EnableConsoleLogging>true</cfg:EnableConsoleLogging>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Legt fest, ob die Konsolenprotokollierung an-/ausgeschaltet wird. • Siehe Kapitel 5 "Fehlercodes (Error Codes) bei slsc_ctrl_get_error, Log-Datei und Konsole", Seite 282.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	EnableFilelogging
XML-Signatur (mit Defaults)	EnableFilelogging = true
	'*' = wahlweise; kein '*' = Pflicht. EnableFilelogging-Wert: Boolean.
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:LogConfig> → <cfg:EnableFilelogging>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:EnableFilelogging>true</cfg:EnableFilelogging>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Legt fest, ob die Dateiprotokollierung an-/ausgeschaltet wird. • Siehe Kapitel 5 "Fehlercodes (Error Codes) bei slsc_ctrl_get_error, Log-Datei und Konsole", Seite 282.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MaxLogfileSize
XML-Signatur (mit Defaults)	MaxLogfileSize* = 5242880
	'*' = wahlweise; kein '*' = Pflicht. MaxLogfileSize-Wert: Positive Ganzzahl.
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:LogConfig> → <cfg:MaxLogfileSize>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:MaxLogfileSize>26214400</cfg:MaxLogfileSize>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Legt die maximale Größe der einzelnen Protokolldateien fest.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MaxBackupFileCount
XML-Signatur (mit Defaults)	MaxBackupFileCount* = 10
	'*' = wahlweise; kein '*' = Pflicht. MaxBackupFileCount-Wert: Nicht-negative Ganzzahl.
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:LogConfig> → <cfg:MaxBackupFileCount>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:MaxBackupFileCount>0</cfg:MaxBackupFileCount>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Legt fest, wie viele Dateien gesichert werden sollen (durch Hinzufügen einer ganzen Zahl zum Dateinamen), bevor die aktuelle Datei kontinuierlich überschrieben wird.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	BaseDirectoryPath
XML-Signatur (mit Defaults)	<p>BaseDirectoryPath* = ""</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>"" bedeutet leerer String.</p> <p>BaseDirectoryPath-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:BaseDirectoryPath>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:BaseDirectoryPath>\${BASE_PATH}</cfg:BaseDirectoryPath>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Der BaseDirectoryPath-Wert funktioniert als Makro, das bei bestimmten Tags (SimOutputFileDirectory, LogfilePath, ProgramFileDirectory) als '[BaseDirectoryPath]' eingefügt werden kann. Zweck ist, Ordnerpfadangaben kürzer darstellen und leichter ändern zu können.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SimulationConfig
XML-Signatur (mit Defaults)	SimulationConfig
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:SimulationConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:SimulationConfig> <!-- erlaubte/mögliche Child-Tags siehe SimulationConfig in der XML-Struktur-Übersicht --> </cfg:SimulationConfig> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> GeneralConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SimulationMode
XML-Signatur (mit Defaults)	SimulationMode = true
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>SimulationMode-Wert: Boolean.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:SimulationConfig> → <cfg:SimulationMode>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:SimulationMode>true</cfg:SimulationMode>
Über API setzbar?	slsc_cfg_set_simulation_setting
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Der SimulationMode-Wert bestimmt, ob der Simulationsmodus oder der Hardwaremodus angeschaltet wird. Siehe auch Kapitel 2.4 "Über die Initialisierung von syncAXIS control-basierten Anwenderprogrammen", Seite 26 und Kapitel 2.5 "Über den syncAXIS control Simulationsmodus", Seite 31. Wenn true: Die syncAXIS control-Instanz wird – außer dem Dongle – mit keiner Hardware kommunizieren. Die Trajektorienplanungsergebnisse werden in Simulationsdateien geschrieben, wenn die Simulationsdatei-Erstellung mit DisableFileOutput eingeschaltet ist.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SimOutputFileDirectory
XML-Signatur (mit Defaults)	<p>SimOutputFileDirectory* = ""</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>"" bedeutet leerer String.</p> <p>SimOutputFileDirectory-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:SimulationConfig> → <cfg:SimOutputFileDirectory>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:SimOutputFileDirectory>[BaseDirectoryPath]/Simulate/</cfg:SimOutputFileDirectory>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Zu [BaseDirectoryPath] siehe BaseDirectoryPath. • Der SimOutputFileDirectory-Wert definiert den Ort, an dem die Simulationsdatei (siehe DisableFileOutput) gespeichert wird. • Zur Simulationsdatei-Benennung siehe Seite 31.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	BinaryOutput
XML-Signatur (mit Defaults)	BinaryOutput* = false
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>BinaryOutput-Wert: Boolean.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:SimulationConfig> → <cfg:BinaryOutput>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:BinaryOutput>true</cfg:BinaryOutput></pre> <p>=> Die Simulationsdateien (siehe DisableFileOutput) werden im Binärformat erzeugt.</p>
	<pre><cfg:BinaryOutput>>false</cfg:BinaryOutput></pre> <p>=> Die Simulationsdateien (siehe DisableFileOutput) werden im ASCII-Format erzeugt.</p>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert, ob Simulationsdateien im ASCII- oder Binärformat erzeugt werden. Das ASCII-Format ist gut geeignet für kleine (Beispiel-)Jobs. Die resultierenden Simulationsdateien können mithilfe eines Texteditors auch noch von einem Menschen gut analysiert werden. Das Binärformat ist besser geeignet für große Jobs. Im Vergleich zum ASCII-Format werden die resultierenden Simulationsdateien schneller geschrieben, sind kleiner, der syncAXIS Viewer kann sie schneller laden und unterliegen außerdem keiner Größenbeschränkung.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DisableFileOutput
XML-Signatur (mit Defaults)	DisableFileOutput* = false
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>DisableFileOutput-Wert: Boolean.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:GeneralConfig> → <cfg:SimulationConfig> → <cfg:DisableFileOutput>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<p><cfg:DisableFileOutput>false</cfg:DisableFileOutput></p> <p>=> Es wird eine Simulationsdatei (siehe DisableFileOutput) erzeugt. Zur Simulationsdatei-Benennung siehe Seite 31.</p> <p><cfg:DisableFileOutput>true</cfg:DisableFileOutput></p> <p>=> Es wird <i>keine</i> Simulationsdatei (siehe DisableFileOutput) erzeugt.</p>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert, ob eine Simulationsdatei erzeugt wird oder nicht. Der Simulationsvorgang ist mit <code>true</code> deutlich schneller als mit <code>false</code>. Die Einstellung <code>true</code> wird empfohlen, wenn (nicht die gesamte Trajektorie, sondern nur) die Job-Merkmale ("<i>Key</i>", siehe enum slsc_JobCharacteristic) von Interesse sind: Die max. Positionswerte und max. Dynamikwerte können mit slsc_ctrl_get_job_characteristic abgefragt werden.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	RTCCConfig
XML-Signatur (mit Defaults)	RTCCConfig
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCCConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:RTCCConfig> <!-- erlaubte/mögliche Child-Tags siehe RTCCConfig in der XML-Struktur-Übersicht --> </cfg:RTCCConfig> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> RTCCConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	BoardIdentificationMethod
XML-Signatur (mit Defaults)	BoardIdentificationMethod = UseFirstFound
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>BoardIdentificationMethod-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:BoardIdentificationMethod>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:BoardIdentificationMethod>BySerialNumber</cfg:BoardIdentificationMethod>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Erlaubte Einträge: <ul style="list-style-type: none"> UseFirstFound BySerialNumber Legt fest, welche RTC6 für die syncAXIS control-Instanz verwendet werden soll. Gibt es nur eine RTC6, wird bei "UseFirstFound" die erste RTC6 benutzt, die die RTC6DLL.dll findet. Mit "BySerialNumber" kann eine bestimmte RTC6 über ihre Seriennummer ausgewählt werden, siehe SerialNumber.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	ProgramFileDirectory
XML-Signatur (mit Defaults)	ProgramFileDirectory* = "" '*'=wahlweise; kein '*'=Pflicht. "" bedeutet leerer String. ProgramFileDirectory-Wert: String.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:ProgramFileDirectory>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:ProgramFileDirectory>[BaseDirectoryPath]/../RTC6</cfg:ProgramFileDirectory>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Zu [BaseDirectoryPath] siehe BaseDirectoryPath. • Setzt den Pfad zu den RTC6-Dateien RTC6RBF.rbf, RTC6DAT.dat und RTC6OUT.out. • Diese werden auf die RTC6 geladen: <ul style="list-style-type: none"> – beim Initialisieren der syncAXIS control-Instanz nach jedem "RTC6 Power Cycle" (d.h. bei RTC6 PCI-Express-Karten der PC-Start) – bei einem Versions-Mismatch (zwischen aktuell benutzter RTC6DLL.dll und den RTC6-Dateien, die auf der RTC6 laufen), z.B. wenn zuvor iSCANcfg6.exe mit anderen RTC6-Dateien gestartet wurde – Wenn die MasterSlaveSynchronizer.exe gestartet wird (aus den als Aufrufparameter angegebenen syncAXISConfig.xml) • Beachten Sie, dass ein Laden der RTC6-Dateien: <ul style="list-style-type: none"> – die RTC6-Karte zurücksetzt – die Clock neu startet – eine Synchronisation (zwischen allen RTC6-Karten und auch dem ACS EtherCAT Netzwerk) aufhebt, die zuvor mit MasterSlaveSynchronizer.exe erreicht wurde
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	Boards
XML-Signatur (mit Defaults)	Boards **=wahlweise; kein **=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:Boards>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:Boards> <!-- erlaubte/mögliche Child-Tags siehe Boards in der XML-Struktur-Übersicht --> </cfg:Boards></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> Boards ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Mit den Child-Tags von Boards werden die zu verwendenden RTC6-Karten konfiguriert.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	RTC6
XML-Signatur (mit Defaults)	RTC6[] **=wahlweise; kein **=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:Boards> → <cfg:RTC6>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:RTC6> <!-- erlaubte/mögliche Child-Tags siehe RTC6 in der XML-Struktur-Übersicht --> </cfg:RTC6></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> RTC6 ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). syncAXIS_1_8.xsd erlaubt bis zu 4 RTC6 Tags.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SerialNumber
XML-Signatur (mit Defaults)	SerialNumber* = 0
	'*' = wahlweise; kein '*' = Pflicht. SerialNumber-Wert: Nicht-negative Ganzzahl.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:Boards> → <cfg:RTC6> → <cfg:SerialNumber>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:SerialNumber>123457</cfg:SerialNumber>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert die Seriennummer derjenigen RTC6-Karte, an der die unter HeadA und HeadB angegebene Hardware angeschlossen ist. Der SerialNumber-Wert wird nur dann ausgewertet, wenn "BySerialNumber" bei BoardIdentificationMethod angegeben ist.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	HeadA
XML-Signatur (mit Defaults)	HeadA
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:Boards> → <cfg:RTC6> → <cfg:HeadA>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:HeadA>ScanDevice1</cfg:HeadA>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Scan-System oder Verfahrtsch, das/der mit dem Anschluss für den ersten Scan-Kopf (SCANHEAD) verbunden ist. Erlaubte Einträge: Siehe enum slsc_ScanDevice und enum slsc_Stage.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	HeadB
XML-Signatur (mit Defaults)	HeadB
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:Boards> → <cfg:RTC6> → <cfg:HeadB>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:HeadB>Stage1</cfg:HeadB> oder <cfg:HeadB>None</cfg:HeadB>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Scan-System oder Verfahrtsch, das/der mit dem Anschluss für den zweiten Scan-Kopf (2. SCANHEAD) verbunden ist. Erlaubte Einträge: Siehe enum slsc_ScanDevice und enum slsc_Stage.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	EthSearch
XML-Signatur (mit Defaults)	EthSearch{ }*
	'*' = wahlweise; kein '*' = Pflicht. '{}' = hier: es ist nur <i>einer</i> der Child-Tags erlaubt ("choice").
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:EthSearch>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:EthSearch> <!-- erlaubte/mögliche Child-Tags siehe EthSearch in der XML-Struktur-Übersicht --> </cfg:EthSearch></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> EthSearch ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Mit dem Child-Tag ('{}' = hier: es ist nur <i>einer</i> der Child-Tags erlaubt ("choice")) von EthSearch wird (werden) die RTC6 Ethernet-Karte(n) konfiguriert.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	Broadcast
XML-Signatur (mit Defaults)	Broadcast
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:EthSearch> → <cfg:Broadcast>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:EthSearch> <cfg:Broadcast> <cfg:IP>169.254.1.0</cfg:IP> <cfg:NetMask>255.255.0.0</cfg:NetMask> </cfg:Broadcast> </cfg:EthSearch> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> Broadcast ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Child-Tags von Broadcast definieren das Subnetz, in dem eine Broadcast-Suche durchgeführt werden soll. Weitere Informationen zu RTC6 Ethernet-Karten finden Sie im RTC6-Handbuch. <p>IP</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): IP = 169.254.1.0 Gibt die Netzadresse des Subnetzes für die Broadcast-Suche an. Format: IP-Adresse in Dezimalpunktschreibweise. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>NetMask</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): NetMask = 255.255.0.0 Gibt die Netzmaske des Subnetzes für die Broadcast-Suche an. Format: IP-Adresse in Dezimalpunktschreibweise. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	IPScan
XML-Signatur (mit Defaults)	IPScan ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:EthSearch> → <cfg:IPScan>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:EthSearch> <cfg:IPScan> <cfg:StartIp>169.254.1.0</cfg:StartIp> <cfg:EndIp>169.254.1.100</cfg:EndIp> </cfg:IPScan> </cfg:EthSearch> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> IPScan ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Child-Tags von IPScan definieren den IP-Adressen-Bereich, in dem ein IP-Scan durchgeführt werden soll. Es werden Netzwerkpakete an diese IP-Adressen gesendet. Weitere Informationen zu RTC6 Ethernet-Karten finden Sie im RTC6-Handbuch. <p>StartIp</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): StartIp = 169.254.1.0 Gibt die untere IP-Bereichsgrenze für den IP-Scan an. Format: IP-Adresse in Dezimalpunktschreibweise. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>EndIp</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): EndIp = 169.254.1.100 Gibt die untere IP-Bereichsgrenze für den IP-Scan an. Format: IP-Adresse in Dezimalpunktschreibweise. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	IPList
XML-Signatur (mit Defaults)	IPList ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:EthSearch> → <cfg:IPList>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:EthSearch> <cfg:IPList> <cfg:IPAddress>192.168.0.1</cfg:IPAddress> <cfg:IPAddress>192.168.0.2</cfg:IPAddress> <cfg:IPAddress>192.168.0.3</cfg:IPAddress> </cfg:IPList> </cfg:EthSearch> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> IPList ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Child-Tags von IPList definieren diejenigen IP-Adressen, bei denen ein IP-Scan durchgeführt werden soll. Es werden Netzwerkpakete an diese IP-Adressen gesendet. Weitere Informationen zu RTC6 Ethernet-Karten finden Sie im RTC6-Handbuch. <p>IPAddress</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): IPAddress[] = 169.254.1.0 Gibt die IP-Adresse einer RTC6 Ethernet-Karte an. syncAXIS_1_8.xsd erlaubt bis zu 255 IPAddress Tags. Format: IP-Adresse in Dezimalpunktschreibweise. Über API setzbar? Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	EthMaxTimeout
XML-Signatur (mit Defaults)	EthMaxTimeout* = 2.0 (Unit*)
	'**'=wahlweise; kein '**'=Pflicht. Unit-Attributwert: Double.
XML-Pfad(e)	<cfg:Configuration> → <cfg:RTCConfig> → <cfg:EthSearch> → <cfg:EthMaxTimeout>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:EthSearch> <cfg:EthMaxTimeout>2.0</cfg:EthMaxTimeout> </cfg:EthSearch></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Legt das maximale Ethernet-Timeout fest. • Der Default-Wert 2.0 bedeutet 2 s. • Eine Änderung des EthMaxTimeout-Werts erfolgt als Aufruf des RTC6-Befehls eth_set_com_timeouts_auto(MaxTimeout) mit MaxTimeout = (EthMaxTimeout in ms). • Weitere Informationen zu RTC6 Ethernet-Karten finden Sie im RTC6-Handbuch.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	ScanDeviceConfig
XML-Signatur (mit Defaults)	ScanDeviceConfig
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:ScanDeviceConfig> <!-- erlaubte/mögliche Child-Tags siehe ScanDeviceConfig in der XML-Struktur-Übersicht --> </cfg:ScanDeviceConfig> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> ScanDeviceConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DynamicLimits
XML-Signatur (mit Defaults)	DynamicLimits ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:DynamicLimits>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:DynamicLimits> <cfg:Velocity Unit="rad/s">90</cfg:Velocity> <cfg:Acceleration Unit="rad/s^2">1.1314e5</cfg:Acceleration> <cfg:Jerk Unit="rad/s^3">4e9</cfg:Jerk> </cfg:DynamicLimits> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> DynamicLimits ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die DynamicLimits-Child-Tags Velocity, Acceleration, Jerk dienen zum Angeben der maximalen dynamischen Fähigkeiten des vorgesehenen Scan-Device-Typs (z.B. excelliSCAN 14 mit Sondertuning, excelliSCAN 20 mit Standardtuning). Sonderfall: syncAXIS Viewer verwendet die Werte bei Velocity, Acceleration und Jerk zur Darstellung von Dynamik-Grenzwertüberschreitungen. syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als: <ul style="list-style-type: none"> Scan-Device-Dynamikgrenzen die DynamicLimits-Werte, Seite 388 Scan-Device-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 395 Scan-Device-Überwachungskriterium den MonitoringLevel-Wert, Seite 397 Verfahrtisch-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 454 Verfahrtisch-Dynamikgrenzen die DynamicLimits-Werte, Seite 456 Verfahrtisch-Überwachungskriterium den MonitoringLevel-Wert, Seite 452 Reaktion auf Überschreitungen den DynamicViolationReaction-Wert, Seite 365 <p>Velocity</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): Velocity = 90.0 (Unit*) Über API setzbar?: slsc_cfg_set_dynamic_limits_scan_device Die max. Geschwindigkeit, zu der der Scan-Device-Typ fähig ist Der Default-Wert 90 [rad/s] ist der eines excelliSCAN 14 mit Standardtuning. Der Wert für einen excelliSCAN 20 mit Standardtuning ist 35 [rad/s]. Modul-Abspiel-Verhalten: Das Modul wird abgelehnt, wenn die abspielende syncAXIS control-Instanz im Betriebsmodus ScannerOnly oder ScannerAndStage sowie der Velocity-Wert kleiner als der MarkSpeed-Wert oder JumpSpeed-Wert im Modul ist. In diesem Fall müssen Sie ein neues Modul mit entsprechend anderem Parameter-Wert aufzeichnen. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66.

<p>Kommentar(e) (Forts.)</p>	<p>Acceleration</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Acceleration = 1.1314e5</code> (Unit*) • Über API setzbar?: <code>slsc_cfg_set_dynamic_limits_scan_device</code> • Die max. Beschleunigung, zu der der Scan-Device-Typ fähig ist. • Der Default-Wert <code>1.1314e5 [rad/s²]</code> ist der eines excelliSCAN 14 mit Standardtuning. Der Wert für einen excelliSCAN 20 mit Standardtuning ist <code>56000 [rad/s²]</code>. • Modul-Abspiel-Verhalten: Das Modul wird abgelehnt, wenn die abspielende syncAXIS control-Instanz im Betriebsmodus <code>ScannerOnly</code> oder <code>ScannerAndStage</code> sowie der Acceleration-Wert kleiner als im Modul ist. In diesem Fall müssen Sie ein neues Modul mit entsprechend anderem Parameter-Wert aufzeichnen. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66. <p>Jerk</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Jerk = 4e9</code> (Unit*) • Über API setzbar?: <code>slsc_cfg_set_dynamic_limits_scan_device</code> • Der max. Ruck, zu der der Scan-Device-Typ fähig ist. • Der Default-Wert <code>4e9 [rad/s³]</code> ist der eines excelliSCAN 14 mit Standardtuning. Der Wert für einen excelliSCAN 20 mit Standardtuning ist <code>10e9 [rad/s³]</code>. • Modul-Abspiel-Verhalten: Das Modul wird abgelehnt, wenn die abspielende syncAXIS control-Instanz im Betriebsmodus <code>ScannerOnly</code> oder <code>ScannerAndStage</code> sowie der Jerk-Wert kleiner als im Modul ist. In diesem Fall müssen Sie ein neues Modul mit entsprechend anderem Parameter-Wert aufzeichnen. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66.
<p>Versionsinfo</p>	<p><code>syncAXIS_1_8.xsd</code></p>

XML-Tag	CalculationDynamics
XML-Signatur (mit Defaults)	CalculationDynamics ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:CalculationDynamics> <!-- erlaubte/mögliche Child-Tags siehe CalculationDynamics in der XML-Struktur-Übersicht --> </cfg:CalculationDynamics> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> CalculationDynamics ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Mit den Child-Tags von CalculationDynamics wird konfiguriert, mit welchen Beschleunigungs&Ruck-Höchstwerten die Trajektorienplanung für die Scan-Devices gerechnet wird. Es handelt sich also um "Planungs-Obergrenzen" und nicht um Plan-Beschleunigungen oder Plan-Rucke. Die Werte werden in eigenen Child-Tags angegeben: <ul style="list-style-type: none"> MarkDynamics für Markierungen JumpDynamics für Sprünge Beachten Sie, dass die entsprechenden Geschwindigkeits-Höchstwerte konfiguriert werden unter: <ul style="list-style-type: none"> <cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:JumpSpeed ...> <cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:MarkSpeed ...>
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MarkDynamics
XML-Signatur (mit Defaults)	MarkDynamics ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:MarkDynamics>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:MarkDynamics> <cfg:Acceleration>1.1314e5</cfg:Acceleration> <cfg:Jerk>4e9</cfg:Jerk> </cfg:MarkDynamics></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> MarkDynamics ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Child-Tags von MarkDynamics betreffen die Werte für Markierungen, siehe Seite 390. Für die Werte für Sprünge gibt es JumpDynamics. <p>Acceleration</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): Acceleration = 1.1314e5 (Unit*) Gibt Beschleunigungs-Höchstwert der Scan-Devices an, mit dem die Trajektorienplanung für die Scan-Devices gerechnet wird. Erlaubte Einträge: <ul style="list-style-type: none"> ≥ 0.00001 Einheit: rad/s². Format: double. ⚠ Vorsicht! syncAXIS control verwendet den Wert von Acceleration = MarkAngularAcc = MarkAngularAcc zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. Über API setzbar?: <pre>slsc_cfg_set_calculation_dynamics_mark_scan_device(MarkAngularAcc) slsc_list_set_calculation_dynamics_mark_scan_device(MarkAngularAcc)</pre> Modul-Abspiel-Verhalten: Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.

Kommentar(e) (Forts.)	<p>Jerk</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Jerk = 4e9 (Unit*)</code> • Gibt Ruck-Höchstwert der Scan-Devices an, mit dem die Trajektorienplanung für die Scan-Devices gerechnet wird. • Erlaubte Einträge: <ul style="list-style-type: none"> – ≥ 0.00001 Einheit: rad/s^3. Format: <code>double</code>. • ⚠ Vorsicht! <code>syncAXIS</code> control verwendet den Wert von <code>Jerk = MarkAngularJerk = MarkAngularJerk</code> zur Planung von Trajektorien für die Betriebsmodi "<code>ScannerOnly</code>" und "<code>ScannerAndStage</code>". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. • Über API setzbar?: <code>slsc_cfg_set_calculation_dynamics_mark_scan_device(MarkAngularJerk)</code> <code>slsc_list_set_calculation_dynamics_mark_scan_device(MarkAngularJerk)</code> • Modul-Abspiel-Verhalten: Kein Standard-Verhalten: Der Parameter-Wert der abspielenden <code>syncAXIS</code> control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Versionsinfo	<code>syncAXIS_1_8.xsd</code>

XML-Tag	JumpDynamics
XML-Signatur (mit Defaults)	JumpDynamics ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:CalculationDynamics> → <cfg:JumpDynamics>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:JumpDynamics> <cfg:Acceleration>1.1314e5</cfg:Acceleration> <cfg:Jerk>4e9</cfg:Jerk> </cfg:JumpDynamics></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> JumpDynamics ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Child-Tags von JumpDynamics betreffen die Werte für Sprünge, siehe Seite 390. Für die Werte für Markierungen gibt es MarkDynamics. <p>Acceleration</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): Acceleration = 1.1314e5 (Unit*) Gibt Beschleunigungs-Höchstwert der Scan-Devices an, mit dem die Trajektorienplanung für die Scan-Devices gerechnet wird. Erlaubte Einträge: <ul style="list-style-type: none"> ≥ 0.00001 Einheit: rad/s². Format: double. ⚠ Vorsicht! syncAXIS control verwendet den Wert von Acceleration = JumpAngularAcc = JumpAngularAcc zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. Über API setzbar?: <pre>slsc_cfg_set_calculation_dynamics_jump_scan_device(JumpAngularAcc) slsc_list_set_calculation_dynamics_jump_scan_device(JumpAngularAcc)</pre> Modul-Abspiel-Verhalten: Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.

Kommentar(e) (Forts.)	<p>Jerk</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Jerk = 4e9 (Unit*)</code> • Gibt Ruck-Höchstwert der Scan-Devices an, mit dem die Trajektorienplanung für die Scan-Devices gerechnet wird. • Erlaubte Einträge: <ul style="list-style-type: none"> – ≥ 0.00001 Einheit: rad/s^3. Format: <code>double</code>. • ⚠ Vorsicht! <code>syncAXIS</code> control verwendet den Wert von <code>Jerk = JumpAngularJerk = JumpAngularJerk</code> zur Planung von Trajektorien für die Betriebsmodi "<code>ScannerOnly</code>" und "<code>ScannerAndStage</code>". Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. • Über API setzbar?: <code>slsc_cfg_set_calculation_dynamics_jump_scan_device(JumpAngularJerk)</code> <code>slsc_list_set_calculation_dynamics_jump_scan_device(JumpAngularJerk)</code> • Modul-Abspiel-Verhalten: Kein Standard-Verhalten: Der Parameter-Wert der abspielenden <code>syncAXIS</code> control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Versionsinfo	<code>syncAXIS_1_8.xsd</code>

XML-Tag	FieldLimits
XML-Signatur (mit Defaults)	FieldLimits ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:FieldLimits>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:FieldLimits> <cfg:XDirection Unit="mm" Max="27" Min="-27" /> <cfg:YDirection Unit="mm" Max="27" Min="-27" /> </cfg:FieldLimits></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> FieldLimits ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die FieldLimits-Child-Tags XDirection und YDirection dienen zum Angeben der Arbeitsfeldgrenzen des vorgesehenen Scan-Device-Typs. syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als: <ul style="list-style-type: none"> Scan-Device-Dynamikgrenzen die DynamicLimits-Werte, Seite 388 Scan-Device-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 395 Scan-Device-Überwachungskriterium den MonitoringLevel-Wert, Seite 397 Verfahrtisch-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 454 Verfahrtisch-Dynamikgrenzen die DynamicLimits-Werte, Seite 456 Verfahrtisch-Überwachungskriterium den MonitoringLevel-Wert, Seite 452 Reaktion auf Überschreitungen den DynamicViolationReaction-Wert, Seite 365 syncAXIS control verwendet die Werte bei XDirection und YDirection <i>nicht</i> zur Planung von Trajektorien! syncAXIS Viewer verwendet die Werte bei XDirection und YDirection zur Darstellung des Scan-Device-Arbeitsfelds und um Grenzwertüberschreitungen zu markieren. Die XDirection- und YDirection-Werte müssen nicht dem nutzbaren Arbeitsfeld entsprechen, das durch die Korrekturdatei (*.ct5) definiert ist. Es können auch niedrigere Feldgrenzen eingestellt werden, wenn ein kleinerer Teil des Arbeitsfelds überwacht werden soll. Höhere Feldgrenzen sollten nicht eingestellt werden.

Kommentar(e) (Forts.)	<p>XDirection</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): XDirection (Unit*, Max, Min) • Max, Min: Double. • Über API setzbar?: slsc_cfg_set_field_limits_scan_device • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>YDirection</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): YDirection (Unit*, Max, Min) • Max, Min: Double. • Über API setzbar?: slsc_cfg_set_field_limits_scan_device • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>ZDirection</p> <ul style="list-style-type: none"> • <i>Dieser Parameter ist aktuell reserviert!</i> • XML-Signatur (mit Defaults): ZDirection* (Unit*, Max, Min) • Max, Min: Double. • Über API setzbar?: slsc_cfg_set_field_limits_scan_device • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MonitoringLevel
XML-Signatur (mit Defaults)	MonitoringLevel = Position
	'*' = wahlweise; kein '*' = Pflicht. MonitoringLevel-Wert: String.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:MonitoringLevel>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:MonitoringLevel> Acceleration </cfg:MonitoringLevel>
Über API setzbar?	slsc_cfg_set_scan_device_dynamic_monitoring_level
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert ein Kriterium, auf das die Scan-Devices hin überwacht werden sollen. Das Kriterium gilt <i>nicht</i> für die Verfahrtische – dafür gibt es den eigenen Tag MonitoringLevel, siehe Seite 452. Überschreitungen lösen automatisch die in DynamicViolationReaction festgelegte Reaktion aus. Erlaubte Einträge: <ul style="list-style-type: none"> Deactivated Position Velocity Acceleration Jerk syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als: <ul style="list-style-type: none"> Scan-Device-Dynamikgrenzen die DynamicLimits-Werte, Seite 388 Scan-Device-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 395 Scan-Device-Überwachungskriterium den MonitoringLevel-Wert, Seite 397 Verfahrtisch-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 454 Verfahrtisch-Dynamikgrenzen die DynamicLimits-Werte, Seite 456 Verfahrtisch-Überwachungskriterium den MonitoringLevel-Wert, Seite 452 Reaktion auf Überschreitungen den DynamicViolationReaction-Wert, Seite 365
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	FocalLength
XML-Signatur (mit Defaults)	FocalLength* = 100 (Unit*)
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:FocalLength ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:FocalLength Unit="mm">100</cfg:FocalLength>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Die Brennweite des Scan-Systems. Ist abhängig von der verwendeten Optik. ⚠ Vorsicht! syncAXIS control verwendet diese Werte zur Planung von Trajektorien für die Betriebsmodi "ScannerOnly" und "ScannerAndStage". Stellen Sie sicher, dass die einge-tragenen Werte korrekt sind.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	Delay
XML-Signatur (mit Defaults)	<p>Delay* = 0.00125 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:Delay ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:Delay Unit="s">0.00125</cfg:Delay>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	<p>Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Ist das (aus <cfg:Configuration> → <cfg:StageConfig> → <cfg:CTIME ...> und <cfg:Configuration> → <cfg:StageConfig> → <cfg:DelayShift ...>) berechnete Verfahrtsch-Delay anders als im Modul, dann werden [WARN]-Log-Dateizeilen generiert. Im Modul enthaltene "SIGNAL"-Funktionen mit einem <i>negativen</i> Delay werden nicht ausgeführt, wenn der Betrag des Delays größer ist als entweder größtes Scan-Device-Delay (<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:Delay ...>) oder berechnetes Verfahrtsch-Delay. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66.</p>
Kommentar(e)	<ul style="list-style-type: none"> • Verzögerung des Scan-Systems. • Für syncAXIS control verwendete excelliSCAN mit Standardtuning haben eine typische Verzögerung von 1,25 ms. • Achtung! Die Einträge in einer Simulationsdatei sind (u.a.) um dieses Delay verschoben. Dies wird im syncAXIS Viewer berücksichtigt. Für eine "einfachere" Simulation kann der Scan-Device-Delay-Wert (hier bei Delay) und den Verfahrtsch-Delay-Wert (bei CTIME) jeweils auf 0 gesetzt werden.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	ScanDeviceList
XML-Signatur (mit Defaults)	ScanDeviceList
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:ScanDeviceList>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:ScanDeviceList> <!-- erlaubte/mögliche Child-Tags siehe ScanDeviceList in der XML-Struktur-Übersicht --> </cfg:ScanDeviceList></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> ScanDeviceList ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Mit den Child-Tags von ScanDeviceList werden die zu verwendenden Scan-Devices konfiguriert.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	ScanDevice
XML-Signatur (mit Defaults)	ScanDevice[] (Name) '*'=wahlweise; kein '*'=Pflicht. Name-Attributwert: String.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:ScanDeviceList> → <cfg:ScanDevice ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:ScanDevice Name="ScanDevice1"> <!-- erlaubte/mögliche Child-Tags siehe ScanDevice in der XML-Struktur-Übersicht --> </cfg:ScanDevice> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • ScanDevice ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 1 Attribut: Name. • Mit syncAXIS_1_8.xsd erlaubte ScanDevice-Tags: <ul style="list-style-type: none"> – bis zu 4 • Erlaubter Name-Attributwert, siehe enum slsc_ScanDevice: "ScanDevice1", "ScanDevice2", "ScanDevice3", "ScanDevice4".
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	CorrectionFileList
XML-Signatur (mit Defaults)	CorrectionFileList ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:ScanDeviceList> → <cfg:ScanDevice ...> → <cfg:CorrectionFileList>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:CorrectionFileList> <!-- erlaubte/mögliche Child-Tags siehe CorrectionFileList in der XML-Struktur-Übersicht --> </cfg:CorrectionFileList></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> CorrectionFileList ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Mit den Child-Tags von CorrectionFileList werden die zu verwendenden Korrekturdateien konfiguriert.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	CorrectionFilePath
XML-Signatur (mit Defaults)	<p>CorrectionFilePath[] = "" (CalibrationFactor* = -1)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>CalibrationFactor-Wert: Double.</p> <p>-1: Der Kalibrierungsfaktor wird aus der Korrekturdatei gelesen.</p> <p>Ordnerpfad: String.</p>
XML-Pfad(e)	<p><cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:ScanDeviceList> → <cfg:ScanDevice ...> → <cfg:CorrectionFileList> → <cfg:CorrectionFilePath ...></p>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:CorrectionFilePath CalibrationFactor="-1">C:\folderpath </cfg:CorrectionFilePath></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • syncAXIS_1_8.xsd erlaubt bis zu 4 CorrectionFilePath Tags. • CorrectionFilePath erzeugt einen Eintrag in der syncAXIS-DLL-internen Korrekturdateiliste. Die ersten 2 Korrekturdateien werden in den RTC6-Speicher geschrieben und können danach schnell mit slsc_ctrl_select_correction_file ausgewählt werden. • Wenn der CalibrationFactor-Wert leer oder ungültig ist, wird eine 1to1-Korrekturdatei geladen. • Weitere Informationen zu Korrekturdateien finden Sie im RTC6-Handbuch.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	Alignment
XML-Signatur (mit Defaults)	Alignment ''='wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:ScanDeviceList> → <cfg:ScanDevice ...> → <cfg:Alignment>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:Alignment> <cfg:Matrix> <cfg:T11>1</cfg:T11> <cfg:T12>0</cfg:T12> <cfg:T21>0</cfg:T21> <cfg:T22>1</cfg:T22> </cfg:Matrix> <cfg:Offset X="0" Y="0" /> </cfg:Alignment> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> Alignment ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Alignment-Child-Tags Matrix und Offset stehen zur Verfügung, um in Multi-Head-Anlagen Montagefehler ausgleichen zu können. Die an ein betreffendes Scan-Device gesendeten Positionen sind dann so transformiert, dass das Scan-Device-Koordinatensystem und das Referenzsystem (z.B. das Verfahrtisch-Koordinatensystem) zueinander passen. Das Alignment wird <i>nicht</i> auf die Trajektorienplanung angewendet. Das Alignment erfolgt erst auf der RTC6. Eine Simulationsdatei enthält die Alignment-Transformation nicht (Simulationsdateien zeigen die Markierung ohne Berücksichtigung einer etwaigen Fehlkalibrierung). Die Matrix muss invertierbar sein, muss aber nicht normiert sein. Weitergehende Informationen siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332. <p>Matrix</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): Matrix Container-Tag (nur). Kein(e) Wert(e), keine Attribut(e). $\text{Matrix} = \begin{bmatrix} T11 & T12 \\ T21 & T22 \end{bmatrix}$ <ul style="list-style-type: none"> Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Kein Verhalten, da Container-Tag

Kommentar(e) (Forts.)	<p>T11</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T11 = 1 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T12</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T12 = 0 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T21</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T21 = 0 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T22</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T22 = 1 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>Offset</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): Offset (X = 0, Y = 0, Unit*) • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	BasePartDisplacement
XML-Signatur (mit Defaults)	BasePartDisplacement ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:ScanDeviceList> → <cfg:ScanDevice ...> → <cfg:BasePartDisplacement>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:BasePartDisplacement> <cfg:Matrix> <cfg:T11>1</cfg:T11> <cfg:T12>0</cfg:T12> <cfg:T21>0</cfg:T21> <cfg:T22>1</cfg:T22> </cfg:Matrix> <cfg:Offset X="0" Y="0" /> </cfg:BasePartDisplacement> </pre>
Über API setzbar?	slsc_cfg_set_part_displacement
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> • BasePartDisplacement ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). • Weitergehende Informationen siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332. • Es ist möglich, eine Fehlausrichtung jedes Werkstücks unter diesem speziellen ScanDevice auszugleichen. • Der Verfahrtisch führt nur einen (durch die Bewegungsaufteilung definierten) bestimmten Anteil an der Gesamt-Kontur aus. Deswegen muss ein solcher Ausgleich komplett durch das Scan-Device erfolgen, was nur kleinere Korrekturen möglich macht. • Die BasePartDisplacement muss eine vorab bekannte Fehlausrichtung eines jeden Werkstücks an einem bestimmten Scan-System sein. • slsc_cfg_set_part_displacement erweitert die Basistransformation (= alle BasePartDisplacement Child-Tags), indem sie die Matrizen multipliziert und die Offsets dieser beiden Transformationen addiert. <p>Matrix</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): Matrix • Kein(e) Wert(e), keine Attribut(e). $\text{Matrix} = \begin{bmatrix} T11 & T12 \\ T21 & T22 \end{bmatrix}$ <ul style="list-style-type: none"> • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Kein Verhalten, da Container-Tag

Kommentar(e) (Forts.)	<p>T11</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T11 = 1 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T12</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T12 = 0 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T21</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T21 = 0 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T22</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T22 = 1 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>Offset</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): Offset (X = 0, Y = 0, Unit*) • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DefaultCorrectionFile
XML-Signatur (mit Defaults)	DefaultCorrectionFile* = 0
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:DefaultCorrectionFile>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:DefaultCorrectionFile>0</cfg:DefaultCorrectionFile>
Über API setzbar?	slsc_ctrl_select_correction_file
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Bestimmt, welche Korrekturdatei aus der syncAXIS-DLL-internen Korrekturdateiliste in dieser syncAXIS control-Instanz benutzt wird. • Erlaubte Einträge: 0...3. Format: Nicht-negative Ganzzahl. • Die aktuell benutzte Korrekturdatei kann mit slsc_ctrl_select_correction_file gewechselt werden. • slsc_ctrl_refresh_correction_file lädt die aktuell benutzte Korrekturdatei wieder vom angegebenen Pfad, allerdings ohne ihren Korrekturdatei-Index zu ändern.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserConfig
XML-Signatur (mit Defaults)	LaserConfig
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:LaserConfig> <!-- erlaubte/mögliche Child-Tags siehe LaserConfig in der XML-Struktur-Übersicht --> </cfg:LaserConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> LaserConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserMode
XML-Signatur (mit Defaults)	LaserMode* = 0
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserMode>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserMode>5</cfg:LaserMode>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert den RTC6-Lasermodus der syncAXIS control-Instanz. Erlaubte Einträge: 0, 4...6. Format: Nicht-negative Ganzzahl. Weitere Informationen zu den Laser-Modi finden Sie im RTC6-Handbuch.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserPortCfg
XML-Signatur (mit Defaults)	LaserPortCfg*
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserPortCfg>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:LaserPortCfg> <cfg:LaserOn>LaserOnSignal</cfg:LaserOn> <cfg:Laser1>Laser1Signal</cfg:Laser1> <cfg:Laser2>Laser2Signal</cfg:Laser2> </cfg:LaserPortCfg> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> LaserPortCfg ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Child-Tags von LaserPortCfg definieren die Ausgabe-Ports der verschiedenen Lasersteuersignale. Standardmäßig wird jedes Lasersteuersignal auf den Port seines Namens gemappt. In manchen Fällen müssen sie möglicherweise anders abgebildet werden. Weitere Informationen zu den Lasersteuersignalen und Ausgabe-Ports finden Sie im RTC6-Handbuch. <p>LaserOn</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): LaserOn = LaserOnSignal Gibt das Lasersignal für den Ausgangskanal LASERON an. Erlaubte Einträge: <ul style="list-style-type: none"> LaserOnSignal Laser1Signal Laser2Signal FirstPulseKillerSignal Format: String. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.

Kommentar(e) (Forts.)	<p>Laser1</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): Laser1 = Laser1Signal • Gibt das Lasersignal für den Ausgangskanal LASER1 an. • Erlaubte Einträge: <ul style="list-style-type: none"> – LaserOnSignal – Laser1Signal – Laser2Signal – FirstPulseKillerSignal <p>Format: String.</p> <ul style="list-style-type: none"> • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>Laser2</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): Laser2 = Laser2Signal • Gibt das Lasersignal für den Ausgangskanal LASER2 an. • Erlaubte Einträge: <ul style="list-style-type: none"> – LaserOnSignal – Laser1Signal – Laser2Signal – FirstPulseKillerSignal <p>Format: String.</p> <ul style="list-style-type: none"> • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserOutput
XML-Signatur (mit Defaults)	<p>LaserOutput* (Unit*, HalfPeriod, PulseLength)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>HalfPeriod-Attributwert: Double.</p> <p>PulseLength-Attributwert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserOutput ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358 .
XML-Abschnitt-Beispiel	<cfg:LaserOutput Unit="s" HalfPeriod="5e-6" PulseLength="1e-6" />
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> LaserOutput definiert (abhängig vom LaserMode) die Impulsfolge der Lasersteuerungsausgabe während des (Lasersteuersignals) LASERON. Der HalfPeriod-Attributwert ist die Hälfte der Zeit zwischen den steigenden Flanken von zwei folgenden Lasersteuerimpulsen. Der PulseLength-Attributwert ist deren Länge. Die Attributwerte werden ignoriert, wenn der jeweilige Kanal (PulseLength, HalfPeriod, SpotDistance) als "ActiveChannel" für die "Automatische Lasersteuerung" eingetragen ist (siehe ActiveChannel). Im Modus "Manuelle Positionierung" und nach slsc_list_suppress_spotdistance_control werden diese Werte wieder ausgegeben. Weitere Informationen zur Lasersteuerung finden Sie im RTC6-Handbuch.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserStandby
XML-Signatur (mit Defaults)	<p>LaserStandby* (Unit*, HalfPeriod, PulseLength)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>HalfPeriod-Attributwert: Double.</p> <p>PulseLength-Attributwert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserStandby ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserStandby Unit="s" HalfPeriod="0.00" PulseLength="0.0" />
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> LaserStandby definiert (abhängig vom LaserMode) die Impulsfolge der Lasersteuerungsausgabe während der Laser aus ist. Der HalfPeriod-Attributwert ist die Hälfte der Zeit zwischen den steigenden Flanken von zwei folgenden Lasersteuerimpulsen. Der PulseLength-Attributwert ist deren Länge. Weitere Informationen zur Lasersteuerung finden Sie im RTC6-Handbuch.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	QSwitchDelay
XML-Signatur (mit Defaults)	<p>QSwitchDelay* = 0 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>QSwitchDelay-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:QSwitchDelay ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:QSwitchDelay Unit="s">0.0</cfg:QSwitchDelay>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert das Delay des Q-Switch-Signals. Weitere Informationen zum Q-Switch-Delay finden Sie im RTC6-Handbuch.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	FPulseKillerLength
XML-Signatur (mit Defaults)	FPulseKillerLength* = 0 (Unit*) ''=wahlweise; kein ''=Pflicht. Unit-Attributwert: Double. FPulseKillerLength-Wert: Double.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:FPulseKillerLength ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:FPulseKillerLength Unit="s">0.0</cfg:FPulseKillerLength>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert die Länge des FirstPulseKiller-Signals. Weitere Informationen zum FirstPulseKiller-Signal finden Sie im RTC6-Handbuch.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserControlFlags
XML-Signatur (mit Defaults)	LaserControlFlags* ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:LaserControlFlags> <!-- erlaubte/mögliche Child-Tags siehe LaserControlFlags in der XML-Struktur-Übersicht --> </cfg:LaserControlFlags></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> LaserControlFlags ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserDisable
XML-Signatur (mit Defaults)	LaserDisable* = false
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>LaserDisable-Wert: Boolean.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags> → <cfg:LaserDisable>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserDisable>false</cfg:LaserDisable>
Über API setzbar?	<p>Auf <code>false</code> setzen: slsc_ctrl_enable_laser.</p> <p>Auf <code>true</code> setzen: slsc_ctrl_disable_laser.</p>
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Legt fest, ob die Lasersteuersignale aktiviert oder deaktiviert sind. • Weitere Informationen zu Freigabe und Unterdrückung der Lasersteuersignale finden Sie im RTC6-Handbuch (set_laser_control, Bit #2).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	PulseSwitchSetting
XML-Signatur (mit Defaults)	PulseSwitchSetting* = false
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>PulseSwitchSetting-Wert: Boolean.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags> → <cfg:PulseSwitchSetting>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:PulseSwitchSetting>false</cfg:PulseSwitchSetting>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Legt fest, ob LASER1-Signal und LASER2-Signal nach der LASERON-Phase (d.h. an der fallenden Flanke des Lasersteuersignals LASERON) abgeschnitten werden. • Weitere Informationen zum Pulse Switch Setting finden Sie im RTC6-Handbuch (set_laser_control, Bit #0).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserSignalPhaseShift
XML-Signatur (mit Defaults)	<p>LaserSignalPhaseShift* = false</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>LaserSignalPhaseShift-Wert: Boolean.</p>
XML-Pfad(e)	<p><cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags> → <cfg:LaserSignalPhaseShift></p>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserSignalPhaseShift>false</cfg:LaserSignalPhaseShift>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • Legt fest, ob die Signale LASER1 und LASER2 phasenverschoben sind (d. h. beide Signale werden im CO2-Modus geschaltet (LaserMode = 0) und das Signal LASER1 wird in YAG-Modi (LaserMode = 1,2,3 und 5) um 180° nach hinten verschoben). • Weitere Informationen zur Phasenverschiebung der Lasersteuersignale finden Sie im RTC6-Handbuch (set_laser_control, Bit #1).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserOnSignalActiveLow
XML-Signatur (mit Defaults)	LaserOnSignalActiveLow* = false
	<p>''=wahlweise; kein ''=Pflicht.</p> <p>LaserOnSignalActiveLow-Wert: Boolean.</p>
XML-Pfad(e)	<code><cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags> →</code> <code><cfg:LaserOnSignalActiveLow></code>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<code><cfg:LaserOnSignalActiveLow>false</cfg:LaserOnSignalActiveLow></code>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • true: LASERON, LASER1 und LASER2 sind Low-aktiv. false: LASERON, LASER1 und LASER2 sind High-aktiv. • Weitere Informationen zum Signal-Pegel des Lasersteuersignals LASERON finden Sie im RTC6-Handbuch (set_laser_control, Bit #3).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	Laser1Laser2SignalActiveLow
XML-Signatur (mit Defaults)	Laser1Laser2SignalActiveLow* = false
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Laser1Laser2SignalActiveLow-Wert: Boolean.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags> → <cfg:Laser1Laser2SignalActiveLow>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:Laser1Laser2SignalActiveLow>false</cfg:Laser1Laser2SignalActiveLow>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> true: LASERON, LASER1 und LASER2 sind Low-aktiv. false: LASERON, LASER1 und LASER2 sind High-aktiv. Weitere Informationen zum LASER1/LASER2-Signal-Pegel finden Sie im RTC6-Handbuch (set_laser_control, Bit #4).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserPulsesAtRisingEdge
XML-Signatur (mit Defaults)	LaserPulsesAtRisingEdge* = false
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>LaserPulsesAtRisingEdge-Wert: Boolean.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags> → <cfg:LaserPulsesAtRisingEdge>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserPulsesAtRisingEdge>false</cfg:LaserPulsesAtRisingEdge>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> LaserPulsesAtRisingEdge ist aktuell reserviert.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	OutputSynchronizationOn
XML-Signatur (mit Defaults)	OutputSynchronizationOn* = false
	'*' = wahlweise; kein '*' = Pflicht. OutputSynchronizationOn-Wert: Boolean.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:LaserControlFlags> → <cfg:OutputSynchronizationOn>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:OutputSynchronizationOn>false</cfg:OutputSynchronizationOn>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> LaserPulsesAtRisingEdge ist aktuell reserviert.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	AutomaticLaserControl
XML-Signatur (mit Defaults)	AutomaticLaserControl
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:AutomaticLaserControl> <!-- erlaubte/mögliche Child-Tags siehe AutomaticLaserControl in der XML-Struktur-Übersicht --> </cfg:AutomaticLaserControl></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> AutomaticLaserControl ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Siehe auch Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	ActiveChannel
XML-Signatur (mit Defaults)	ActiveChannel ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:ActiveChannel>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht" , Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:ActiveChannel> <cfg:Channel>AnalogOut1</cfg:Channel> <cfg:Channel>SpotDistance</cfg:Channel> </cfg:ActiveChannel> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> • ActiveChannel ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). • Für die "Automatische Lasersteuerung" können zusätzlich auch Shift, VelocityFactor und RadiusFactor angegeben werden (diese können jeweils als disabled oder enabled eingestellt werden). • Beachten Sie, dass Rampen nur auf "ActiveChannel" angewendet werden. • Siehe auch Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. Channel <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): Channel[*] • syncAXIS_1_8.xsd erlaubt bis zu 2 Channel Tags. • Definiert, welcher Steuerparameter für die "Automatische Lasersteuerung" verwendet wird. • Erlaubte Einträge: <ul style="list-style-type: none"> – AnalogOut1 – AnalogOut2 – PulseLength – HalfPeriod – SpotDistance Format: String. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Ist er anders als der im Modul, dann werden [WARN]-Log-Dateizeilen generiert. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	AnalogOut1
XML-Signatur (mit Defaults)	<p>AnalogOut1* (DefaultOutput, Format*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>DefaultOutput-Attributwert: Double.</p> <p>Format-Attributwert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut1 ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:AnalogOut1 DefaultOutput="0.5" Format="Factor"> <!-- erlaubte/mögliche Child-Tags siehe AnalogOut1 in der XML-Struktur-Übersicht --> </cfg:AnalogOut1></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> AnalogOut1 ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 2 Attribute: DefaultOutput, Format. Hinweis: Wenn "AnalogOut1" nicht als (einer der beiden) "ActiveChannel" (siehe ActiveChannel) eingestellt ist, dann wird AnalogOut1 (Child-Tag von DefaultOutputs) ausgegeben. Zum Definieren, wie AnalogOut1 basierend auf Radius und Geschwindigkeit angepasst werden soll: Dazu können mit den Child-Tags unterhalb von AnalogOut1 die Kennlinien für RadiusFactor und VelocityFactor definiert werden. DefaultOutput-Attribut: Default-Ausgabe, die mit Faktoren multipliziert wird, siehe Abbildung 17, Seite 52. Format-Attribut: Faktor (Anteil der maximalen Analogspannung von 10 V). Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.
Versionsinfo	syncAXIS_1_8.xsd

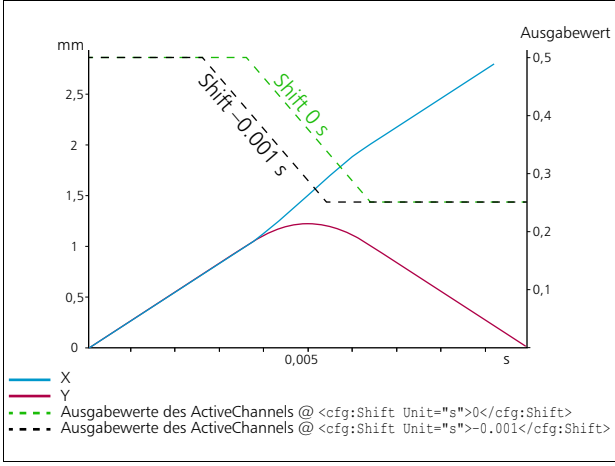
XML-Tag	AnalogOut2
XML-Signatur (mit Defaults)	<p>AnalogOut2* (DefaultOutput, Format*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>DefaultOutput-Attributwert: Double.</p> <p>Format-Attributwert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut2 ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:AnalogOut2 DefaultOutput="0.5" Format="Factor"> <!-- erlaubte/mögliche Child-Tags siehe AnalogOut2 in der XML-Struktur-Übersicht --> </cfg:AnalogOut2></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> AnalogOut2 ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 2 Attribute: DefaultOutput, Format. Hinweis: Wenn "AnalogOut2" nicht als (einer der beiden) "ActiveChannel" (siehe ActiveChannel) eingestellt ist, dann wird AnalogOut2 (Child-Tag von DefaultOutputs) ausgegeben. Zum Definieren, wie AnalogOut2 basierend auf Radius und Geschwindigkeit angepasst werden soll: Dazu können mit den Child-Tags unterhalb von AnalogOut2 die Kennlinien für RadiusFactor und VelocityFactor definiert werden. DefaultOutput-Attribut: Default-Ausgabe, die mit Faktoren multipliziert wird, siehe Abbildung 17, Seite 52. Format-Attribut: Faktor (Anteil der maximalen Analogspannung von 10 V). Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	PulseLength
XML-Signatur (mit Defaults)	PulseLength* (DefaultOutput, Unit*) '*'=wahlweise; kein '*'=Pflicht. DefaultOutput-Attributwert: Double. Unit-Attributwert: Double.
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:PulseLength ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:PulseLength DefaultOutput="1e-5" Unit="s"> <!-- erlaubte/mögliche Child-Tags siehe PulseLength in der XML-Struktur-Übersicht --> </cfg:PulseLength> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> PulseLength ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 2 Attribute: DefaultOutput, Unit. Hinweis: Wenn "PulseLength" nicht als (einer der beiden) "ActiveChannel" (siehe ActiveChannel) eingestellt ist, dann wird der PulseLength-Attributwert von LaserOutput ausgegeben. Zum Definieren, wie die Pulslänge basierend auf Radius und Geschwindigkeit angepasst werden soll: Dazu können mit den Child-Tags unterhalb von PulseLength die Kennlinien für RadiusFactor und VelocityFactor definiert werden. DefaultOutput-Attribut: Default-Ausgabe, die mit Faktoren multipliziert wird, siehe Abbildung 17, Seite 52. Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	HalfPeriod
XML-Signatur (mit Defaults)	<p>HalfPeriod* (DefaultOutput, Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>DefaultOutput-Attributwert: Double.</p> <p>Unit-Attributwert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:HalfPeriod ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:HalfPeriod DefaultOutput="1e-5" Unit="s"> <!-- erlaubte/mögliche Child-Tags siehe HalfPeriod in der XML-Struktur-Übersicht --> </cfg:HalfPeriod></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • HalfPeriod ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 2 Attribute: DefaultOutput, Unit. • Hinweis: Wenn "HalfPeriod" nicht als (einer der beiden) "ActiveChannel" (siehe ActiveChannel) eingestellt ist, dann wird der HalfPeriod-Attributwert von LaserOutput ausgegeben. • Zum Definieren, wie die HalfPeriod basierend auf Radius und Geschwindigkeit angepasst werden soll: Dazu können mit den Child-Tags unterhalb von HalfPeriod die Kennlinien für RadiusFactor und VelocityFactor definiert werden. • Hinweis: Eine Skalierung der HalfPeriod basierend auf der Markiergeschwindigkeit kann zu Lücken im Markierergebnis führen, wenn diese Geschwindigkeit nahe bei 0 liegt. Es wird empfohlen, SpotDistance statt HalfPeriod zu verwenden! • DefaultOutput-Attribut: Default-Ausgabe, die mit Faktoren multipliziert wird, siehe Abbildung 17, Seite 52. • Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SpotDistance
XML-Signatur (mit Defaults)	<p>SpotDistance* (DefaultOutput, Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>DefaultOutput-Attributwert: Double.</p> <p>Unit-Attributwert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:SpotDistance ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:SpotDistance DefaultOutput="0.005" Unit="mm"> <!-- erlaubte/mögliche Child-Tags siehe SpotDistance in der XML-Struktur-Übersicht --> </cfg:SpotDistance></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> SpotDistance ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 2 Attribute: DefaultOutput, Unit. Mit den Child-Tags unterhalb von SpotDistance werden die Kennlinien für RadiusFactor und VelocityFactor definiert. Der DefaultOutput-Attributwert ist der Abstand zwischen den einzelnen Laserspots, wenn kein RadiusFactor oder VelocityFactor angewendet wird. Der DefaultOutput-Wert wird intern auf 1/40 RTC6-Positionsbit, d.h. $(1/40) / K_{xy}$ [mm] gerundet. Das bedeutet, dass sich für lange Linien ein kleiner Rundungsfehler bei der Positionierung akkumulieren kann. Der Fehler beträgt höchstens $1/40 \times (\text{Vektorlänge} \times \text{Arbeitsfeldgröße}) / (2^{20} \times \text{SpotDistance})$. Bei einer Arbeitsfeld-Kantenlänge von 50 mm, einem Pulsabstand von 50 µm und einer Linienlänge von 10 mm ergäbe sich somit ein Rundungsfehler max. 0,24 µm. Ist der DefaultOutput-Wert ein Vielfaches von 1/40 RTC6-Positionsbit, wird es zu keinem Rundungsfehler kommen. syncAXIS control benutzt das "Spot Distance Control"-Feature der RTC6. Dieses ermöglicht eine präzise Positionierung der Laserpulse mit 64 MHz-Auflösung in Abhängigkeit von der Scannerposition (früher war RTC-Karten-seitig nur HalfPeriod möglich). RadiusFactor gilt nicht für die Spotdistanz selbst, sondern für die Spotgeschwindigkeit und somit für die Spotdistanz, die durch die Umkehrung des Faktors (1/Factor) skaliert wird.

Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • <code>SpotDistance</code> ist ein Sonderfall, bei dem – obwohl der <code>VelocityFactor</code> deaktiviert sein könnte – die Geschwindigkeitsinformation benutzt wird, um die Triggerzeitpunkte zum Erreichen von konstanten Spotabständen zu berechnen. <code>VelocityFactor</code> kann aber immer noch zur Feinabstimmung verwendet werden. • Weitere Informationen zum <code>SpotDistance</code>-Channel siehe Kapitel 2.9.5 "Über die "Konturabhängige Geschwindigkeitsberechnung"", Seite 60. • Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	Shift
XML-Signatur (mit Defaults)	Shift* = 0.0 (Unit*) '*'=wahlweise; kein '*'=Pflicht. Unit-Attributwert: Double.
XML-Pfad(e)	(1) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut1> → <cfg:Shift> (2) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut2> → <cfg:Shift> (3) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:PulseLength> → <cfg:Shift> (4) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:HalfPeriod> → <cfg:Shift> (5) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:SpotDistance> → <cfg:Shift>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:Shift Unit="s">-1e-5</cfg:Shift>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Mit Shift können die Ausgabewerte des "ActiveChannel" (siehe auch Abbildung 17, Seite 52, Output-Wert) zeitlich verschoben werden. Shift-Wert mit negativem Vorzeichen: Die Ausgabe erfolgt früher als wie bei 0. Die nachfolgende Abbildung ist Abbildung 21, Seite 56 plus zusätzlich eingezeichneter "Shift -0.001 s"-Kurve (gestrichelt, schwarz).  <p> — X — Y - - - - - Ausgabewerte des ActiveChannels @ <cfg:Shift Unit="s">0</cfg:Shift> - - - - - Ausgabewerte des ActiveChannels @ <cfg:Shift Unit="s">-0.001</cfg:Shift> </p>

Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • <code>Shift</code> wird bereitgestellt, um Verzögerungszeiten der Lasersteuerung ausgleichen zu können. • Für jeden Kanal kann individuell ein zeitlicher Shift angegeben werden, d.h. <code>Shift</code> ist als Child-Tag erlaubt bei <code>AnalogOut1</code>, <code>AnalogOut2</code>, <code>PulseLength</code>, <code>HalfPeriod</code>, <code>SpotDistance</code> (siehe XML-Pfade 1...5, oben). Das wird möglich gemacht, weil z.B. Analogsignale eine längere Einschwingzeit als der Laser-Puls-Trigger benötigen.
Versionsinfo	<code>syncAXIS_1_8.xsd</code>

XML-Tag	RadiusFactor
XML-Signatur (mit Defaults)	<p>RadiusFactor (Enabled, RadiusUnit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>RadiusUnit-Attributwert: String.</p> <p>Enabled-Attributwert: Boolean.</p>
XML-Pfad(e)	<p>(1) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut1> → <cfg:RadiusFactor></p> <p>(2) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut2> → <cfg:RadiusFactor></p> <p>(3) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:PulseLength> → <cfg:RadiusFactor></p> <p>(4) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:HalfPeriod> → <cfg:RadiusFactor></p> <p>(5) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:SpotDistance> → <cfg:RadiusFactor></p>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:RadiusFactor RadiusUnit="mm" Enabled="true"> <cfg:DataPoint Radius="0" Factor="0.0" /> <cfg:DataPoint Radius="15" Factor="1.0" /> <cfg:DataPoint Radius="27" Factor="2.0" /> </cfg:RadiusFactor></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> RadiusFactor ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 2 Attribute: RadiusUnit, Enabled. RadiusFactor ist ein Skalierungsfaktor. Abhängig vom Auslenkungsradius des Laserstrahls wird er mit dem Ausgangswert des jeweiligen Kanals multipliziert. Er kann verwendet werden, um die Energieverteilung entlang der Spotverzerrung durch den Auftreffwinkel auf das Werkstück zu kompensieren. <div data-bbox="715 1547 1334 2009" data-label="Image"> </div>

Kommentar(e) (Forts.)	<ul style="list-style-type: none"> Für diese Art der automatischen Lasersteuerungseinstellung kann eine Kennlinie aus einzelnen Stützpunkten (=Child-Tags <code>DataPoint</code>, s.u.) definiert werden. Zwischen den Stützpunkten werden die Werte linear interpoliert, außerhalb der Stützpunkte konstant extrapoliert. Die Kennlinien für den <code>RadiusFactor</code> können für jeden Kanal (siehe XML-Pfad(e), oben) einzeln definiert werden, auch wenn sie <code>Enabled="false"</code> sind. Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. <p><code>DataPoint</code></p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): <code>DataPoint[]*</code> (Radius, Factor) <code>syncAXIS_1_8.xsd</code> erlaubt eine unbegrenzte Anzahl von <code>DataPoint</code>-Tags. Ein Kennlinien-Stützpunkt. Format: Double. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden <code>syncAXIS control</code>-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	<code>syncAXIS_1_8.xsd</code>

XML-Tag	VelocityFactor
XML-Signatur (mit Defaults)	VelocityFactor (Enabled, VelocityUnit*) '*'=wahlweise; kein '*'=Pflicht. VelocityUnit-Attributwert: String. Enabled-Attributwert: Boolean.
XML-Pfad(e)	(1) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut1> → <cfg:VelocityFactor> (2) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:AnalogOut2> → <cfg:VelocityFactor> (3) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:PulseLength> → <cfg:VelocityFactor> (4) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:HalfPeriod> → <cfg:VelocityFactor> (5) <cfg:Configuration> → <cfg:LaserConfig> → <cfg:AutomaticLaserControl> → <cfg:SpotDistance> → <cfg:VelocityFactor>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:VelocityFactor VelocityUnit="mm" Enabled="true"> <cfg:DataPoint Velocity="0" Factor="0.0"/> <cfg:DataPoint Velocity="400" Factor="1.0"/> <cfg:DataPoint Velocity="4000" Factor="2.0"/> </cfg:VelocityFactor> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> VelocityFactor ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 2 Attribute: VelocityUnit, Enabled. VelocityFactor ist ein Skalierungsfaktor. Abhängig von der Spotgeschwindigkeit des Laserstrahls wird er mit dem Ausgangswert des jeweiligen Kanals multipliziert. Er kann verwendet werden, um größeren Energieeintrag aufgrund geringerer Markiergeschwindigkeit auszugleichen, z.B. in Ecken (ohne Sky Writing-ähnliche Bewegungen). Für diese Art der automatischen Lasersteuerungseinstellung kann eine Kennlinie aus einzelnen Stützpunkten (=Child-Tags DataPoint, s.u.) definiert werden. Zwischen den Stützpunkten werden die Werte linear interpoliert, außerhalb der Stützpunkte konstant extrapoliert. Die Kennlinien für den VelocityFactor können für jeden Kanal (siehe XML-Pfad(e), oben) einzeln definiert werden, auch wenn sie Enabled="false" sind.

Kommentar(e) (Fort.)	<ul style="list-style-type: none"> SpotDistance als Kanal ist ein Sonderfall: die inverse Kennlinie wird auf den tatsächlichen Spotabstand angewendet. Die Kennlinie beeinflusst die Geschwindigkeitsinformationen, die zur Feinabstimmung verwendet werden. Beachten Sie auch, dass bei Verwendung von SpotDistance als "ActiveChannel" die Geschwindigkeitsinformationen unabhängig vom VelocityFactor verwendet werden, um die Triggerzeitpunkte zum Erreichen von konstanten Spotabständen zu berechnen. Siehe Kapitel 2.9 "Über das automatische Steuern des Lasers durch syncAXIS control ("Automatische Lasersteuerung")", Seite 48. <p>DataPoint</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): DataPoint[*] (Velocity, Factor) Ein Kennlinien-Stützpunkt. Format: Double. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	TrajectoryConfig
XML-Signatur (mit Defaults)	TrajectoryConfig '*'=wahlweise; kein '*'=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:TrajectoryConfig> <!-- erlaubte/mögliche Child-Tags siehe TrajectoryConfig in der XML-Struktur-Übersicht --> </cfg:TrajectoryConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> TrajectoryConfig ist nur ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MarkConfig
XML-Signatur (mit Defaults)	MarkConfig (VelocityUnit*) ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:MarkConfig> <!-- erlaubte/mögliche Child-Tags siehe MarkConfig in der XML-Struktur-Übersicht --> </cfg:MarkConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> MarkConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	JumpSpeed
XML-Signatur (mit Defaults)	<p>JumpSpeed = 400 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>JumpSpeed-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:JumpSpeed ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:JumpSpeed Unit="mm/s">400</cfg:JumpSpeed>
Über API setzbar?	<p>slsc_cfg_set_jump_speed</p> <p>slsc_list_set_jump_speed</p> <p>slsc_cfg_set_trajectory_config</p>
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe JumpSpeed.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MarkSpeed
XML-Signatur (mit Defaults)	<p>MarkSpeed = 400 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>MarkSpeed-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:MarkSpeed ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:MarkSpeed Unit="mm/s">400</cfg:MarkSpeed>
Über API setzbar?	slsc_cfg_set_mark_speed slsc_list_set_mark_speed slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe MarkSpeed.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MinimalMarkSpeed
XML-Signatur (mit Defaults)	MinimalMarkSpeed = 0 (Unit*)
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>MinimalMarkSpeed-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:MinimalMarkSpeed ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:MinimalMarkSpeed Unit="mm/s">200</cfg:MinimalMarkSpeed>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe MinimalMarkSpeed.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserSwitchConfig
XML-Signatur (mit Defaults)	<p>LaserSwitchConfig (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:LaserSwitchConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:LaserSwitchConfig> <!-- erlaubte/mögliche Child-Tags siehe LaserSwitchConfig in der XML-Struktur-Übersicht --> </cfg:LaserSwitchConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> LaserSwitchConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserPreTriggerTime
XML-Signatur (mit Defaults)	<p>LaserPreTriggerTime = 0 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>LaserPreTriggerTime-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:LaserSwitchConfig> → <cfg:LaserPreTriggerTime ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht" , Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserPreTriggerTime Unit="s">1e-6</cfg:LaserPreTriggerTime>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Falls der Parameter-Wert der abspielenden syncAXIS control-Instanz größer ist als die kürzeste Sprung-Segment-Dauer im Modul , dann werden [WARN]-Log-Dateizeilen generiert. Bei den Markier-Segmenten , die solchen Sprung-Segmenten nachfolgen, wird das Lasersignal nicht im Voraus getriggert. Siehe auch Abschnitt "Modul-Abspiel-Verhalten" , Seite 66.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe LaserPreTriggerTime.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserSwitchOffsetTime
XML-Signatur (mit Defaults)	<p>LaserSwitchOffsetTime = -20e-6 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>LaserSwitchOffsetTime-Wert: Double.</p>
XML-Pfad(e)	<p><cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:LaserSwitchConfig> → <cfg:LaserSwitchOffsetTime ...></p>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserSwitchOffsetTime Unit="s">-20e-6</cfg:LaserSwitchOffsetTime>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe LaserSwitchOffsetTime.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserMinOffTime
XML-Signatur (mit Defaults)	<p>LaserMinOffTime = 1e-6 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>LaserMinOffTime-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:MarkConfig> → <cfg:LaserSwitchConfig> → <cfg:LaserMinOffTime ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:LaserMinOffTime Unit="s">1.5625e-8</cfg:LaserMinOffTime>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe LaserMinOffTime. Der kleinste zulässige Wert ist: 1/64 μs
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	GeometryConfig
XML-Signatur (mit Defaults)	<p>GeometryConfig</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:GeometryConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:GeometryConfig> <!-- erlaubte/mögliche Child-Tags siehe GeometryConfig in der XML-Struktur-Übersicht --> </cfg:GeometryConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> GeometryConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MaxBlendRadius
XML-Signatur (mit Defaults)	<p>MaxBlendRadius = 1.0 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>MaxBlendRadius-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:GeometryConfig> → <cfg:MaxBlendRadius ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:MaxBlendRadius Unit="mm">1.0</cfg:MaxBlendRadius>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe auch MaxBlendRadius. Definiert den Abstand von den Ecken, ab dem die Blending-Kurven beginnen können. Abhängig von den Blend-Einstellungen kann dieser Abstand kleiner sein. Siehe auch Abbildung 39, Seite 292.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	ApproxBlendLimit
XML-Signatur (mit Defaults)	<p>ApproxBlendLimit = 0.5 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>ApproxBlendLimit-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:GeometryConfig> → <cfg:ApproxBlendLimit ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:ApproxBlendLimit Unit="mm">0.5</cfg:ApproxBlendLimit>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe auch ApproxBlendLimit. Definiert den Abstand zu den Ecken, den die Blending-Kurven mindestens erreichen müssen. Abhängig von den Blend-Einstellungen kann dieser Abstand kleiner sein. Siehe auch Abbildung 39, Seite 292.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	BlendMode
XML-Signatur (mit Defaults)	BlendMode = MinimalBlending ''=wahlweise; kein ''=Pflicht. BlendMode-Wert: String.
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:GeometryConfig> → <cfg:BlendMode>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:BlendMode>VariableBlending</cfg:BlendMode>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Definiert den Blend-Modus der syncAXIS control-Instanz, siehe auch enum slsc_BlendModes. Erlaubte Einträge: <ul style="list-style-type: none"> Deactivated Kein Blending. Es wird eine Sky Writing-ähnliche Bewegung hinzugefügt. VariableBlending Die Blending-Kurven werden so berechnet, dass sie so schnell wie möglich sind und sie die MaxBlendRadius- und ApproxBlendLimit-Beschränkung einhalten. MinimalBlending Die Blending-Kurven sind unter Beibehaltung der MinimalMarkSpeed so nah wie möglich an der definierten Kontur. FixedBlending Veraltet. Wenn ein Blending nicht mit den gewünschten Einschränkungen durchgeführt werden kann, wird (wie bei "Deactivated") eine Sky Writing-ähnliche Bewegung hinzugefügt. Um deren Anzahl zu ermitteln steht die Aufzählungskonstante slsc_JobCharacteristic_InsertedSkywritings zur Verfügung. Siehe auch Abbildung 39, Seite 292.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	AutoCyclicGeometry (veraltet)
XML-Signatur (mit Defaults)	AutoCyclicGeometry = false ''=wahlweise; kein ''=Pflicht. AutoCyclicGeometry-Wert: Boolean.
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:GeometryConfig> → <cfg:AutoCyclicGeometry>
XML-Struktur- Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt- Beispiel	<cfg:AutoCyclicGeometry>false</cfg:AutoCyclicGeometry>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel- Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Veraltet. Empfohlene Einstellung: false. Bedeutung: siehe auch AutoCyclicGeometry. AutoCyclicGeometry war eine Einstellung für Splines. Geschlossene Konturen konnten mit einheitlichen Blend-Einstellungen markiert werden, auch für die Start-/End-Ecke, die normalerweise nicht betroffen gewesen wären (Übergang von Sprung-zu-Mark und Mark-zu-Sprung).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SplineConversionLengthLimit (veraltet)
XML-Signatur (mit Defaults)	<p>SplineConversionLengthLimit = 0.5 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>SplineConversionLengthLimit-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:GeometryConfig> → <cfg:SplineConversionLengthLimit ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:SplineConversionLengthLimit Unit="mm">0.5</cfg:SplineConversionLengthLimit>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> Veraltet. Empfohlene Einstellung: Belassen Sie den Default-Wert. Bedeutung: siehe auch SplineConversionLengthLimit. SplineConversionLengthLimit definierte die maximale Länge für Vektoren, die in Splines umgewandelt werden sollen. Vektoren, die länger als das SplineConversionLengthLimit-Wert sind, wurden nicht zum Spline hinzugefügt und Blending-Kurven oder Sky Writing-ähnliche Bewegungen wurden dazwischen ausgeführt.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SplineMode (veraltet)
XML-Signatur (mit Defaults)	SplineMode = Deactivated
	<p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>SplineConversionLengthLimit-Wert: String.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:GeometryConfig> → <cfg:SplineMode>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:SplineMode>Deactivated</cfg:SplineMode>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> • Veraltet. Empfohlene Einstellung: Deactivated. Definierte den Spline-Modus der syncAXIS control-Instanz, siehe auch enum slsc_SplineModes. • Erlaubte Einträge: <ul style="list-style-type: none"> – Deactivated Empfohlen. Keine Splines. – Interpolating Es wurden Polynombahnen zwischen den Markierungspunkten interpoliert. – Approximating Es wurden Polynombahnen zwischen den Markierungspunkten approximiert. • "Approximating" reduzierte die dynamische Belastung, während "Interpolating" genauer war. • Siehe auch Abbildung 44, Seite 319.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	VectorResolution
XML-Signatur (mit Defaults)	<p>VectorResolution = 0.02 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>VectorResolution-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:TrajectoryConfig> → <cfg:VectorResolution ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:VectorResolution Unit="mm">0.02</cfg:VectorResolution>
Über API setzbar?	slsc_cfg_set_trajectory_config
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Bedeutung: siehe auch VectorResolution. Definiert die "Systemauflösung" – Vektoren, die kürzer als diese Auflösung sind, werden mit dem vorherigen Vektor kombiniert, siehe Abbildung 40, Seite 292.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	StageConfig
XML-Signatur (mit Defaults)	StageConfig
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:StageConfig> <!-- erlaubte/mögliche Child-Tags siehe StageConfig in der XML-Struktur-Übersicht --> </cfg:StageConfig> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> StageConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DelayShift
XML-Signatur (mit Defaults)	DelayShift* = 0.0 (Unit*) '*'=wahlweise; kein '*'=Pflicht. Unit-Attributwert: Double. DelayShift-Wert: Double.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:DelayShift ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:DelayShift Unit="s">0.0</cfg:DelayShift>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	<p>Wie <cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:Delay ...>: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Ist das (aus <cfg:Configuration> → <cfg:StageConfig> → <cfg:CTIME ...> und <cfg:Configuration> → <cfg:StageConfig> → <cfg:DelayShift ...>) berechnete Verfahrtsch-Delay anders als im Modul, dann werden [WARN]-Log-Dateizeilen generiert. Im Modul enthaltene "SIGNAL"-Funktionen mit einem <i>negativen</i> Delay werden nicht ausgeführt, wenn der Betrag des Delays größer ist als entweder größtes Scan-Device-Delay (<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:Delay ...>) oder berechnetes Verfahrtsch-Delay. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66.</p>
Kommentar(e)	<ul style="list-style-type: none"> Mit dem DelayShift-Wert kann die zeitliche Synchronisation von Scan-Device und Verfahrtsch optimiert werden. Der Default-Wert von 0 sollte für alle XL SCAN-Systeme korrekt sein. Die notwendigen Daten werden automatisch ausgelesen und angewendet, siehe CTIME. Sollte sich zeigen, dass eine Nachbesserung der zeitlichen Synchronisation notwendig ist, dann wenden Sie sich an SCANLAB.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	CTIME
XML-Signatur (mit Defaults)	<p>CTIME* = -1 (Unit*)</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>Unit-Attributwert: Double.</p> <p>CTIME-Wert: Double.</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:CTIME ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:CTIME Unit="ms">-1.0</cfg:CTIME>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	<p>Wie <cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:Delay ...>: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Ist das (aus <cfg:Configuration> → <cfg:StageConfig> → <cfg:CTIME ...> und <cfg:Configuration> → <cfg:StageConfig> → <cfg:DelayShift ...>) berechnete Verfahrtsch-Delay anders als im Modul, dann werden [WARN]-Log-Dateizeilen generiert. Im Modul enthaltene "SIGNAL"-Funktionen mit einem negativen Delay werden nicht ausgeführt, wenn der Betrag des Delays größer ist als entweder größtes Scan-Device-Delay (<cfg:Configuration> → <cfg:ScanDeviceConfig> → <cfg:Delay ...>) oder berechnetes Verfahrtsch-Delay. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66.</p>
Kommentar(e)	<ul style="list-style-type: none"> In syncAXIS control ≤ V1.2.6 wurde der Verfahrtsch-Delay-Wert basierend auf der Zyklusdauer des ACS-Protokolls (ACS-Cycle Time, "CTIME") unter <cfg:Configuration> → <cfg:StageConfig> → <cfg:Delay ...> eingetragen. In syncAXIS control ≥ V1.3.0 ist der <cfg:Delay ...>-Tag nicht mehr vorhanden. Stattdessen wird mit CTIME = -1 die CTIME vom ACS Motion-Controller automatisch ausgelesen und ein entsprechender Verfahrtsch-Delay-Wert automatisch gesetzt. Ist hier direkt ein CTIME-Wert eingetragen, dann wird dieser verwendet (kein Auslesen vom ACS Motion-Controller). Ein entsprechender Verfahrtsch-Delay-Wert wird automatisch gesetzt. Wenn ein Verfahrtsch-Delay-Wert = 0 gewünscht ist, stellen Sie <cfg:CTIME Unit="ms">0</cfg:CTIME> ein.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MonitoringLevel
XML-Signatur (mit Defaults)	MonitoringLevel = Jerk
	'*' = wahlweise; kein '*' = Pflicht. MonitoringLevel-Wert: String.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:MonitoringLevel>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:MonitoringLevel> Acceleration </cfg:MonitoringLevel>
Über API setzbar?	slsc_cfg_set_stage_dynamic_monitoring_level
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Definiert ein Kriterium, auf das die Verfahrtsche hin überwacht werden sollen. Das Kriterium gilt <i>nicht</i> für die Scan-Devices – dafür gibt es den eigenen Tag MonitoringLevel, siehe Seite 397. Die Beschreibung dort gilt analog. Überschreitungen lösen automatisch die in DynamicViolationReaction festgelegte Reaktion aus. Erlaubte Einträge: <ul style="list-style-type: none"> Deactivated Position Velocity Acceleration Jerk syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als: <ul style="list-style-type: none"> Scan-Device-Dynamikgrenzen die DynamicLimits-Werte, Seite 388 Scan-Device-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 395 Scan-Device-Überwachungskriterium den MonitoringLevel-Wert, Seite 397 Verfahrtsch-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 454 Verfahrtsch-Dynamikgrenzen die DynamicLimits-Werte, Seite 456 Verfahrtsch-Überwachungskriterium den MonitoringLevel-Wert, Seite 452 Reaktion auf Überschreitungen den DynamicViolationReaction-Wert, Seite 365
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	StageList
XML-Signatur (mit Defaults)	StageList*
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:StageList> <!-- erlaubte/mögliche Child-Tags siehe StageList in der XML-Struktur-Übersicht --> </cfg:StageList></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> StageList ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	Stage
XML-Signatur (mit Defaults)	Stage[] (Name)
	'*' = wahlweise; kein '*' = Pflicht. Name-Attributwert: String.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:Stage Name="Stage1"> <!-- erlaubte/mögliche Child-Tags siehe Stage in der XML-Struktur-Übersicht --> </cfg:Stage></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Stage ist <i>auch</i> ein Container-Tag. Kein(e) Wert(e), 1 Attribut: Name. syncAXIS_1_8.xsd erlaubt bis zu 4 Stage Tags. Erlaubte Einträge bei Name, siehe enum slsc_Stage: "Stage1", "Stage2", "Stage3", "Stage4".
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	FieldLimits
XML-Signatur (mit Defaults)	FieldLimits ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:FieldLimits>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:FieldLimits> <cfg:XDirection Unit="mm" Max="150" Min="-150" /> <cfg:YDirection Unit="mm" Max="150" Min="-150" /> </cfg:FieldLimits></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> FieldLimits ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die FieldLimits-Child-Tags XDirection und YDirection dienen zum Angeben der Arbeitsfeldgrenzen des vorgesehenen Verfahrtisch-Typs. syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als: <ul style="list-style-type: none"> Scan-Device-Dynamikgrenzen die DynamicLimits-Werte, Seite 388 Scan-Device-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 395 Scan-Device-Überwachungskriterium den MonitoringLevel-Wert, Seite 397 Verfahrtisch-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 454 Verfahrtisch-Dynamikgrenzen die DynamicLimits-Werte, Seite 456 Verfahrtisch-Überwachungskriterium den MonitoringLevel-Wert, Seite 452 Reaktion auf Überschreitungen den DynamicViolationReaction-Wert, Seite 365 syncAXIS control verwendet die Werte bei XDirection und YDirection <i>nicht</i> zur Planung von Trajektorien! syncAXIS Viewer verwendet die Werte bei XDirection und YDirection zur Darstellung des Verfahrtisch-Arbeitsfelds und um Grenzwertüberschreitungen zu markieren. Die Werte bei XDirection und YDirection müssen nicht dem nutzbaren Arbeitsfeld entsprechen. Es können auch niedrigere Feldgrenzen eingestellt werden, wenn ein kleinerer Teil des Arbeitsfelds überwacht werden soll. Höhere Feldgrenzen sollten nicht eingestellt werden.

Kommentar(e) (Forts.)	<p>XDirection</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): XDirection (Unit*, Max, Min) • Format: Double. • Über API setzbar?: slsc_cfg_set_field_limits_stage • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>YDirection</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): YDirection (Unit*, Max, Min) • Format: Double. • Über API setzbar?: slsc_cfg_set_field_limits_stage • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>ZDirection</p> <ul style="list-style-type: none"> • <i>Dieser Parameter ist aktuell reserviert!</i> • XML-Signatur (mit Defaults): ZDirection* (Unit*, Max, Min) • Max, Min: Double. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DynamicLimits
XML-Signatur (mit Defaults)	DynamicLimits ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:DynamicLimits>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:DynamicLimits> <cfg:Velocity Unit="mm/s">1000</cfg:Velocity> <cfg:Acceleration Unit="mm/s^2">10000</cfg:Acceleration> <cfg:Jerk Unit="mm/s^3">100000</cfg:Jerk> </cfg:DynamicLimits></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> DynamicLimits ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die DynamicLimits-Child-Tags Velocity, Acceleration und Jerk dienen zum Angeben der Dynamikgrenzen des vorgesehenen Verfahrtisch-Typs. syncAXIS control verwendet zur Prüfung von Arbeitsfeld und Dynamik als: <ul style="list-style-type: none"> Scan-Device-Dynamikgrenzen die DynamicLimits-Werte, Seite 388 Scan-Device-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 395 Scan-Device-Überwachungskriterium den MonitoringLevel-Wert, Seite 397 Verfahrtisch-Arbeitsfeldgrenzen die FieldLimits-Werte, Seite 454 Verfahrtisch-Dynamikgrenzen die DynamicLimits-Werte, Seite 456 Verfahrtisch-Überwachungskriterium den MonitoringLevel-Wert, Seite 452 Reaktion auf Überschreitungen den DynamicViolationReaction-Wert, Seite 365 syncAXIS control verwendet die Werte bei Velocity, Acceleration und Jerk <i>nicht</i> zur Planung von Trajektorien! Dafür wird CalculationDynamics benutzt. syncAXIS Viewer verwendet die Werte bei Velocity, Acceleration und Jerk zur Darstellung von Dynamik-Grenzwertüberschreitungen. Die Werte bei Velocity, Acceleration und Jerk müssen nicht den tatsächlichen Dynamikgrenzen entsprechen. Es können auch niedrigere Dynamikgrenzen eingestellt werden, wenn die Dynamikgrenzen strenger überwacht werden sollen.

Kommentar(e) (Forts.)	<p>Velocity</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Velocity = 1000.0 (Unit*)</code> • Format: Double. • Über API setzbar?: <code>slsc_cfg_set_dynamic_limits_stage</code> • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden <code>syncAXIS control</code>-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>Acceleration</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Acceleration = 10000.0 (Unit*)</code> • Format: Double. • Über API setzbar?: <code>slsc_cfg_set_dynamic_limits_stage</code> • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden <code>syncAXIS control</code>-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>Jerk</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Jerk = 100000.0 (Unit*)</code> • Format: Double. • Über API setzbar?: <code>slsc_cfg_set_dynamic_limits_stage</code> • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden <code>syncAXIS control</code>-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	<code>syncAXIS_1_8.xsd</code>

XML-Tag	CalculationDynamics
XML-Signatur (mit Defaults)	CalculationDynamics ''=wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage> → <cfg:CalculationDynamics>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:CalculationDynamics> <cfg:Velocity Unit="mm/s">1000</cfg:Velocity> <cfg:Acceleration Unit="mm/s^2">10000</cfg:Acceleration> <cfg:Jerk Unit="mm/s^3">100000</cfg:Jerk> </cfg:CalculationDynamics></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> CalculationDynamics ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die CalculationDynamics-Child-Tags Velocity, Acceleration und Jerk dienen zum Angeben der dynamischen Fähigkeiten ("Dynamikgrenzen") des vorgesehenen Verfahrtisch-Typs, die verwendet werden um, die Verfahrtisch-Bewegung zu berechnen. ⚠ Vorsicht! syncAXIS control verwendet die Werte bei Velocity, Acceleration und Jerk zur Planung von Trajektorien für den Betriebsmodus "StageOnly" sowie für die Endbewegung an Job-Enden. Stellen Sie sicher, dass die eingetragenen Werte korrekt sind. Im "ScannerAndStage"-Modus werden diese Werte <ul style="list-style-type: none"> <i>nicht</i> für Markierbewegungen berücksichtigt (weil die Bewegungsaufteilung auf Basis des FilterBandwidth-Wert erfolgt) an Job-Enden (für die Endbewegung) berücksichtigt Für den Betriebsmodus "StageOnly" gilt: Sollte (via syncAXISConfig.xml oder Funktionsaufruf) eine höhere Mark-Speed oder Jump-Speed vorgegeben sein, dann werden diese automatisch auf den Velocity-Wert reduziert. In diesem Fall wird eine [ERROR]-Log-Dateizeile erzeugt, siehe [ERROR]-Log-Dateizeilen.

Kommentar(e) (Forts.)	<p>Velocity</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Velocity = 1000.0 (Unit*)</code> • Format: Double • Über API setzbar?: <code>slsc_cfg_set_calculation_dynamics_stage</code> • <code>Velocity</code> ist die Geschwindigkeit des Verfahrtschis ("Dynamikgrenze") • Modul-Abspiel-Verhalten: Das Modul wird abgelehnt, wenn die abspielende <code>syncAXIS control</code>-Instanz im Betriebsmodus <code>StageOnly</code> und der <code>Velocity</code>-Wert kleiner als der <code>MarkSpeed</code>-Wert oder <code>JumpSpeed</code>-Wert im Modul ist. In diesem Fall müssen Sie ein neues Modul mit entsprechend anderem Parameter-Wert aufzeichnen. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66. <p>Acceleration</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Acceleration = 10000.0 (Unit*)</code> • Format: Double • Über API setzbar?: <code>slsc_cfg_set_calculation_dynamics_stage</code> • <code>Acceleration</code> ist die Beschleunigung des Verfahrtschis ("Dynamikgrenze") • Modul-Abspiel-Verhalten: Das Modul wird abgelehnt, wenn die abspielende <code>syncAXIS control</code>-Instanz im Betriebsmodus <code>StageOnly</code> und der <code>Acceleration</code>-Wert kleiner als im Modul ist. In diesem Fall müssen Sie ein neues Modul mit entsprechend anderem Parameter-Wert aufzeichnen. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66. <p>Jerk</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): <code>Jerk = 100000.0 (Unit*)</code> • Format: Double • Über API setzbar?: <code>slsc_cfg_set_calculation_dynamics_stage</code> • <code>Jerk</code> ist der Ruck des Verfahrtschis ("Dynamikgrenze") • Modul-Abspiel-Verhalten: Das Modul wird abgelehnt, wenn die abspielende <code>syncAXIS control</code>-Instanz im Betriebsmodus <code>StageOnly</code> und der <code>Jerk</code>-Wert kleiner als im Modul ist. In diesem Fall müssen Sie ein neues Modul mit entsprechend anderem Parameter-Wert aufzeichnen. Siehe auch Abschnitt "Modul-Abspiel-Verhalten", Seite 66.
Versionsinfo	<code>syncAXIS_1_8.xsd</code>

XML-Tag	Alignment
XML-Signatur (mit Defaults)	Alignment ''='wahlweise; kein ''=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage ...> → <cfg:Alignment>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:Alignment> <cfg:Matrix> <cfg:T11>1</cfg:T11> <cfg:T12>0</cfg:T12> <cfg:T21>0</cfg:T21> <cfg:T22>1</cfg:T22> </cfg:Matrix> <cfg:Offset X="0" Y="0" /> </cfg:Alignment> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> Alignment ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die Alignment-Child-Tags Matrix und Offset stehen zur Verfügung, um in Multi-Head-Anlagen Montagefehler ausgleichen zu können. Die an einen betreffenden Verfahrtisch gesendeten Positionen sind dann so transformiert, dass das Verfahrtisch-Koordinatensystem und das Referenzsystem (z.B. das Scan-Kopf-Koordinatensystem) zueinander passen. Das Alignment wird auf die Trajektorienplanung angewendet! (Das Alignment erfolgt nicht erst auf der RTC6!). Eine Simulationsdatei enthält die Alignment-Transformation nicht (Simulationsdateien zeigen die Markierung ohne Berücksichtigung einer etwaigen Fehlkalibrierung). Die Matrix muss invertierbar sein, muss aber nicht normiert sein. Weitergehende Informationen siehe Kapitel 8.3 "Über Transformationen in syncAXIS control V1.2.4 und höher", Seite 332. <p>Matrix</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): Matrix Kein(e) Wert(e), keine Attribut(e). $\text{Matrix} = \begin{bmatrix} T11 & T12 \\ T21 & T22 \end{bmatrix}$ <ul style="list-style-type: none"> Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.

<p>Kommentar(e) (Forts.)</p>	<p>T11</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T11 = 1 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T12</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T12 = 0 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T21</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T21 = 0 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>T22</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): T22 = 1 • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>Offset</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): Offset (X = 0, Y = 0, Unit*) • Format: Double. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
<p>Versionsinfo</p>	<p>syncAXIS_1_8.xsd</p>

XML-Tag	StageAxisX
XML-Signatur (mit Defaults)	StageAxisX* = 0
	'*' = wahlweise; kein '*' = Pflicht. StageAxisX-Wert: Nicht-negative Ganzzahl.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage ...> → <cfg:StageAxisX>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:StageAxisX>0</cfg:StageAxisX>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Für Verfahrtschaufbauten mit ACS-X-Achsenindex ≠ 0.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	StageAxisY
XML-Signatur (mit Defaults)	StageAxisY* = 1
	'*' = wahlweise; kein '*' = Pflicht. StageAxisY-Wert: Nicht-negative Ganzzahl.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage ...> → <cfg:StageAxisY>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:StageAxisY>1</cfg:StageAxisY>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Für Verfahrtschaufbauten mit ACS-X-Achsenindex ≠ 1. Siehe auch Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	SlecEtherCATNodeID
XML-Signatur (mit Defaults)	SlecEtherCATNodeID* = 0
	'*' = wahlweise; kein '*' = Pflicht. SlecEtherCATNodeID-Wert: Nicht-negative Ganzzahl.
XML-Pfad(e)	<cfg:Configuration> → <cfg:StageConfig> → <cfg:StageList> → <cfg:Stage ...> → <cfg:SlecEtherCATNodeID>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:SlecEtherCATNodeID>1</cfg:SlecEtherCATNodeID>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> Position des SLEC im EtherCAT-Netzwerk. Für Verfahrtschaufbauten mit SLEC-ID ≠ 0.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	IOConfig
XML-Signatur (mit Defaults)	IOConfig*
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:IOConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:IOConfig> <!-- erlaubte/mögliche Child-Tags siehe IOConfig in der XML-Struktur-Übersicht --> </cfg:IOConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> IOConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DefaultOutputs
XML-Signatur (mit Defaults)	DefaultOutputs*
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:IOConfig> → <cfg:DefaultOutputs>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:DefaultOutputs> <cfg:LaserPinOut Format="Bitpattern" Value="1" /> <cfg:AnalogOut1 Format="Factor" Value="0.5" /> <cfg:AnalogOut2 Format="Factor" Value="0.5" /> </cfg:DefaultOutputs> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> • DefaultOutputs ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). • Nach dem Start der RTC6 sind die analogen Ports low (0 V) und die digitalen Ports befinden sich im (hochohmigen) tristate-Zustand. • Vorausgesetzt, der betreffende Port ist nicht als "ActiveChannel" (siehe Channel) definiert: die angegebenen Werte werden ab der ersten Job-Ausführung solange ausgegeben, bis sie über eine API-Funktion geändert werden oder die LaserShutdownSequence ausgeführt wird. <p>LaserPinOut</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): LaserPinOut* (Format*, Value) • Format: String. • Erlaubte Einträge: 0, 1, 2, 3. • Falls keine "Automatische Lasersteuerung" verwendet wird, wird dieser Wert während der Job-Ausführung am jeweiligen Port ausgegeben. • Über API setzbar?: slsc_list_write_analog_x, slsc_list_write_digital_out • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.

Kommentar(e) (Forts.)	<p>AnalogOut1</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): AnalogOut1* (Format*, Value) • Format: Double. • Erlaubte Einträge: 0...1. • Der Faktor für den Analogausgang bezieht sich auf den Maximalwert von 10 V. Falls keine "Automatische Lasersteuerung" verwendet wird, wird dieser Wert während der Job-Ausführung am jeweiligen Port ausgegeben. • Über API setzbar?: slsc_list_write_analog_x, slsc_list_write_digital_out • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>AnalogOut2</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): AnalogOut2* (Format*, Value) • Format: Double. • Erlaubte Einträge: 0...1. • Der Faktor für den Analogausgang bezieht sich auf den Maximalwert von 10 V. Falls keine "Automatische Lasersteuerung" verwendet wird, wird dieser Wert während der Job-Ausführung am jeweiligen Port ausgegeben. • Über API setzbar?: slsc_list_write_analog_x, slsc_list_write_digital_out • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserInitSequence
XML-Signatur (mit Defaults)	<p>LaserInitSequence{ }*</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>'{' = hier: es sind beliebig viele der Child-Tags erlaubt ("sequence").</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:IOConfig> → <cfg:LaserInitSequence>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:LaserInitSequence> <cfg:Delay>0</cfg:Delay> <cfg:SetLaserPinOut Format="Bitpattern" Value="1" /> <cfg:SetAnalogOut1 Format="Factor" Value="0.5" /> <cfg:SetAnalogOut2 Format="Factor" Value="1.0" /> <cfg:SetExt1DigitalOut Value="1" /> </cfg:LaserInitSequence> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> LaserInitSequence ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die einzelnen Child-Tags von LaserInitSequence werden sequenziell abgearbeitet: <ul style="list-style-type: none"> Bei syncAXIS control-Instanz-Initialisierung Bei Verfahrtisch-Akquisition aufgrund slsc_cfg_acquire_stage (veraltet) Das kann zum Initialisieren von Laser (z. B. Global Enable des Lasers setzen) und Peripheriekomponenten genutzt werden. syncAXIS_1_8.xsd erlaubt das mehrfache Vorkommen der einzelnen LaserInitSequence-Child-Tags. Nach dem Start der RTC6 sind die analogen Ports low (0 V) und die digitalen Ports befinden sich im (hochohmigen) tristate-Zustand. Die zuletzt gesetzten Werte werden solange ausgegeben, bis: <ul style="list-style-type: none"> Die erste Job-Ausführung <i>beginnt</i> Ein API-Funktionsaufruf diese überschreibt <p>Delay</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): Delay (Unit*) Format: Nicht-negativer double. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.

Kommentar(e) (Forts.)	<p>SetLaserPinOut</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetLaserPinOut (Format*, Value) • Format: String. • Erlaubte Einträge: 0, 1, 2, 3. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>SetAnalogOut1</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetAnalogOut1 (Format*, Value) • Format: Double. • Erlaubte Einträge: 0...1. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>SetAnalogOut2</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetAnalogOut1 (Format*, Value) • Format: Double. • Erlaubte Einträge: 0...1. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>SetExt1DigitalOut</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetExt1DigitalOut (Format*, Value, Mask*) • Format: Double. • Erlaubte Einträge: 0...65535 (dezimal) oder 0x0000...0xFFFF (hex). • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	LaserShutdownSequence
XML-Signatur (mit Defaults)	<p>LaserShutdownSequence{*}</p> <p>'*' = wahlweise; kein '*' = Pflicht.</p> <p>'{' = hier: es sind beliebig viele der Child-Tags erlaubt ("sequence").</p>
XML-Pfad(e)	<cfg:Configuration> → <cfg:IOConfig> → <cfg:LaserShutdownSequence>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:LaserShutdownSequence> <cfg:Delay>0</cfg:Delay> <cfg:SetLaserPinOut Format="Bitpattern" Value="0" /> <cfg:SetAnalogOut2 Format="Factor" Value="0.0" /> <cfg:SetAnalogOut1 Format="Factor" Value="0.0" /> <cfg:SetExtlDigitalOut Value="0" /> </cfg:LaserShutdownSequence> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> LaserShutdownSequence ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e). Die einzelnen Child-Tags von LaserShutdownSequence werden sequenziell abgearbeitet: <ul style="list-style-type: none"> Bei syncAXIS control-Instanz-Abbau Bei Verfahrstisch-Freigabe aufgrund slsc_cfg_release_stage (veraltet) Das kann zum Herunterfahren des Lasers (z. B. Global Enable des Lasers ausschalten) und Peripheriekomponenten genutzt werden. syncAXIS_1_8.xsd erlaubt das mehrfache Vorkommen der einzelnen LaserInitSequence-Child-Tags. Nach dem Start der RTC6 sind die analogen Ports low (0 V) und die digitalen Ports befinden sich im (hochohmigen) tristate-Zustand. <p>Delay</p> <ul style="list-style-type: none"> XML-Signatur (mit Defaults): Delay (Unit*) Format: Nicht-negativer double. Über API setzbar?: Nicht möglich. Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.

Kommentar(e) (Forts.)	<p>SetLaserPinOut</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetLaserPinOut (Format*, Value) • Format: String. • Erlaubte Einträge: 0, 1, 2, 3. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>SetAnalogOut1</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetAnalogOut1 (Format*, Value) • Format: Double. • Erlaubte Einträge: 0...1. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>SetAnalogOut2</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetAnalogOut1 (Format*, Value) • Format: Double. • Erlaubte Einträge: 0...1. • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten. <p>SetExt1DigitalOut</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): SetExt1DigitalOut (Format*, Value, Mask*) • Format: Double. • Erlaubte Einträge: 0...65535 (dezimal) oder 0x0000...0xFFFF (hex). • Über API setzbar?: Nicht möglich. • Modul-Abspiel-Verhalten: Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	MotionDecompositionConfig
XML-Signatur (mit Defaults)	MotionDecompositionConfig
	'*' = wahlweise; kein '*' = Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:MotionDecompositionConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre> <cfg:MotionDecompositionConfig> <!-- erlaubte/mögliche Child-Tags siehe MotionDecompositionConfig in der XML-Struktur-Übersicht --> </cfg:MotionDecompositionConfig> </pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> MotionDecompositionConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	FilterBandwidth
XML-Signatur (mit Defaults)	FilterBandwidth = 2 (Unit*) ''=wahlweise; kein ''=Pflicht. FilterBandwidth-Wert: Double.
XML-Pfad(e)	<cfg:Configuration> → <cfg:MotionDecompositionConfig> → <cfg:FilterBandwidth>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:FilterBandwidth>2</cfg:FilterBandwidth>
Über API setzbar?	slsc_cfg_set_bandwidth
Modul-Abspiel-Verhalten	Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird angewendet. Sie müssen also nicht jedes Mal ein neues Modul aufzeichnen, wenn Sie den Parameter-Wert variieren möchten.
Kommentar(e)	<ul style="list-style-type: none"> • slsc_cfg_initialize_from_file schlägt fehl, wenn der FilterBandwidth-Wert kleiner als 0,23 ist. Beim Rückgabewert ist dann Bit #14 gesetzt (XmlLoadError). • Der FilterBandwidth-Wert ist derjenige Parameter, der die Aufteilung von Scan-Device und Verfahrtisch-Bewegung bestimmt. Je größer der FilterBandwidth-Wert, desto mehr dynamische Last wird dem Verfahrtisch zugewiesen. Je kleiner der FilterBandwidth-Wert, desto mehr Last wird dem Scan-Device zugewiesen. Abhängig von den Prozessbeschränkungen wünschen sich Anwender möglicherweise <ul style="list-style-type: none"> – eine minimale Belastung des Verfahrtisches, um Ungenauigkeiten durch Vibrationen des Verfahrtisches zu reduzieren, – ein minimales Arbeitsfeld des Scanners, um Ungenauigkeiten zu reduzieren, die durch Kalibrierungsungenauigkeiten des Scanners bedingt sind, oder – eine minimale Prozesszeit durch eine maximale Ausnutzung der Scanner und dynamischen Fähigkeiten des Verfahrtisches. Der optimale FilterBandwidth-Wert hängt stark von der Kontur ab. Er kann durch Überprüfung der dynamischen Belastung (die die Trajektorienplanung berechnet hat) mit Hilfe des Simulationsmodus bestimmt werden, siehe Kapitel 2.6 "Über das Optimieren von syncAXIS control-basierten Anwenderprogrammen", Seite 36. • Die Dynamik- und Positionierfähigkeiten des Scanners und des Verfahrtisches werden bei der Bewegungsaufteilung nicht berücksichtigt. Auch aus diesem Grund müssen Sie jeden Job immer erst simulieren, bevor Sie ihn erstmalig auf der Hardware ausführen. Nur so können Sie sicherstellen, dass keine Grenzen überschritten werden. Siehe Kapitel 2.2 "Über die SICHERE Verwendung von syncAXIS control – Allgemeines Vorgehen", Seite 18.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	HeuristicConfig
XML-Signatur (mit Defaults)	HeuristicConfig* *'='wahlweise; kein '*'=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:MotionDecompositionConfig> → <cfg:HeuristicConfig>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:HeuristicConfig> <!-- erlaubte/mögliche Child-Tags siehe HeuristicConfig in der XML-Struktur-Übersicht --> </cfg:HeuristicConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> HeuristicConfig ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DynamicReductionFunctions
XML-Signatur (mit Defaults)	DynamicReductionFunctions* *'='wahlweise; kein '*'=Pflicht.
XML-Pfad(e)	<cfg:Configuration> → <cfg:MotionDecompositionConfig> → <cfg:HeuristicConfig> → <cfg:DynamicReductionFunctions>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:HeuristicConfig> <!-- erlaubte/mögliche Child-Tags siehe DynamicReductionFunctions in der XML-Struktur-Übersicht --> </cfg:HeuristicConfig></pre>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> DynamicReductionFunctions ist <i>nur</i> ein Container-Tag. Kein(e) Wert(e), keine Attribut(e).
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	DynamicReductionFunction
XML-Signatur (mit Defaults)	DynamicReductionFunction[*] (Units*) '*'=wahlweise; kein '*'=Pflicht. DynamicReductionFunction-Attributwert: 'mm and mm/s'.
XML-Pfad(e)	<cfg:Configuration> → <cfg:MotionDecompositionConfig> → <cfg:HeuristicConfig> → <cfg:DynamicReductionFunctions> → <cfg:DynamicReductionFunction ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<pre><cfg:DynamicReductionFunction Units="mm and mm/s"> <cfg:DataPoint Length="0.0" Velocity="2000" /> <cfg:DataPoint Length="27.0" Velocity="2000" /> <cfg:DataPoint Length="27.01" Velocity="700" /> <cfg:DataPoint Length="54.0" Velocity="700" /> </cfg:DynamicReductionFunction></pre>
Über API setzbar?	slsc_cfg_select_heuristic
Modul-Abspiel-Verhalten	Kein Verhalten, da Container-Tag
Kommentar(e)	<ul style="list-style-type: none"> • Zweck des DynamicReductionFunction-Tags ist, Kennlinien für Geschwindigkeitsvermindierungen definieren zu können, siehe Listenelement "Anwendungsfall" unten. • Die Heuristik kann nicht angewendet werden, wenn gar keine Kennlinie definiert ist (= kein DynamicReductionFunction-Tag vorhanden). • syncAXIS_1_8.xsd erlaubt eine unbegrenzte Anzahl von DynamicReductionFunction-Tags. Es können also beliebig viele Kennlinien definiert werden. • Die syncAXIS control-Instanz wird mit den Werten des ersten (obersten) DynamicReductionFunction-Tag initialisiert (entspricht slsc_cfg_select_heuristic(HeuristicIndex = 0)). Die erste (= oberste) Kennlinie wird also als Voreinstellung verwendet. • Siehe auch Kapitel 2.10 "Über Heuristik und Kennlinien für Geschwindigkeits-Vermindierungen", Seite 64. • Anwendungsfall: Bei langen Vektoren (die länger als das halbe Scanner-Arbeitsfeld sind), die mit hohen Geschwindigkeiten ausgeführt werden, kann der Verfahrtschritt einer höheren dynamischen Belastung ausgesetzt sein, als für ihn möglich. Um die kombinierte Geschwindigkeit in solchen Fällen zu verringern, kann eine Kennlinie angelegt werden: dazu werden die einzelnen Kennlinien-Stützpunkte (mit Längen- und Geschwindigkeitsinformationen) mittels DataPoint-Tags definiert. Weiterhin können Sie dedizierte Kennlinien (z.B. für niederfrequente Markier-Muster und kleine hochfrequente Markier-Muster) in je einem eigenen DynamicReductionFunction-Tag definieren. Die gewünschte Kennlinie wird mit slsc_cfg_select_heuristic(HeuristicIndex = [0...(DynamicReductionFunction – 1)]) eingestellt. Beachten Sie, dass die Ausführungszeit in seltenen Fällen (z. B. bei Vektoren, die gerade etwas länger sind als das halbe Scan-Kopf-Arbeitsfeld und somit keine Grenzwerte überschreiten) mit Kennlinie länger sein kann als ohne Kennlinie. • Nicht-Anwendungsfall: Viele kurze Vektoren, die aufeinanderfolgen.

Kommentar(e) (Forts.)	<ul style="list-style-type: none"> • <code>HeuristicForJumpsOnly = true</code> stellt ein, dass die Kennlinie nur auf Sprung-Segmente und nicht auf Markier-Segmente angewendet wird. <p>DataPoint</p> <ul style="list-style-type: none"> • XML-Signatur (mit Defaults): DataPoint[] (Length, Velocity) • syncAXIS_1_8.xsd erlaubt eine unbegrenzte Anzahl von DataPoint-Tags. • Ein Kennlinien-Stützpunkt. • Format: Double. • Über API setzbar?: Nicht möglich. • Achten Sie darauf, dass die DataPoint-Tags aufeinanderfolgen wie im XML-Abschnitt-Beispiel, Seite 473 gezeigt: <ul style="list-style-type: none"> – Aufsteigende Length-Werte – Absteigend Velocity-Werte • Modul-Abspiel-Verhalten: Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Versionsinfo	syncAXIS_1_8.xsd

XML-Tag	HeuristicForJumpsOnly
XML-Signatur (mit Defaults)	HeuristicForJumpsOnly* = false ''=wahlweise; kein ''=Pflicht. HeuristicForJumpsOnly-Wert: Boolean.
XML-Pfad(e)	<cfg:Configuration> → <cfg:MotionDecompositionConfig> → <cfg:HeuristicConfig> → <cfg:HeuristicForJumpsOnly ...>
XML-Struktur-Übersicht	Siehe Kapitel 13.1 "xml-Struktur-Übersicht", Seite 358.
XML-Abschnitt-Beispiel	<cfg:HeuristicForJumpsOnly>true</cfg:HeuristicForJumpsOnly>
Über API setzbar?	Nicht möglich.
Modul-Abspiel-Verhalten	Kein Standard-Verhalten: Der Parameter-Wert der abspielenden syncAXIS control-Instanz wird nicht angewendet. Wenn Sie den Parameter-Wert variieren möchten, dann müssen Sie dafür jedes Mal ein neues Modul aufzeichnen.
Kommentar(e)	<ul style="list-style-type: none"> HeuristicForJumpsOnly bezieht sich auf die unter DynamicReductionFunction angelegte Kennlinien für Geschwindigkeitsverminderungen: <ul style="list-style-type: none"> – true Die Kennlinie wird nur auf Sprung-Segmente, aber nicht auf Markier-Segmente angewendet. Diese Einstellung vermeidet Einbrände/schlechte Markierungsergebnisse, wenn Nicht-"pulse on demand"-Laser eingesetzt werden. – false Die Kennlinie wird sowohl auf Sprung-Segmente als auch auf Markier-Segmente angewendet. Die Dynamik- und Positionierfähigkeiten des Scan-Device und des Verfahrtischs werden bei der Bewegungsaufteilung nicht berücksichtigt. Auch aus diesem Grund müssen Sie jeden Job immer erst simulieren, bevor Sie ihn erstmalig auf der Hardware ausführen. Nur so können Sie sicherstellen, dass keine Grenzen überschritten werden. Siehe Kapitel 2.2 "Über die SICHERE Verwendung von syncAXIS control – Allgemeines Vorgehen", Seite 18.
Versionsinfo	syncAXIS_1_8.xsd

14 Änderungsindex

Nachfolgend genannt sind Änderungen an diesem Handbuch aufgrund technischer Weiterentwicklung des Produkts sowie wesentliche redaktionelle Änderungen.

Änderungen nach Dokument-Revision **1.9.19 de-DE** von Dokument-Revision 0.9.18 de-DE

Wo	Was
Global	Dokument Revision <ul style="list-style-type: none"> • 1.9.19 de-DE gilt für syncAXIS control-Software-Paket <ul style="list-style-type: none"> • V1.7.0
slsc_cfg_get_jump_time , Seite 128	Software-Änderung. Neue Funktion.
Kapitel 12 "Anhang E: Anwendungshinweis – Benutzung der syncAXIS-DLL unter C#", Seite 350	Redaktionelle Erweiterung.
Änderungsindex, Seite 476	

Änderungen nach Dokument-Revision 1.9.20 de-DE von Dokument-Revision 1.9.19 de-DE

Wo	Was
Global	Dokument Revision <ul style="list-style-type: none"> 1.9.20 de-DE gilt für syncAXIS control-Software-Paket <ul style="list-style-type: none"> V1.8.0
Kapitel 5 "Fehlercodes (Error Codes) bei slsc_ctrl_get_error, Log-Datei und Konsole", Seite 282	Software-Änderung. Fehlercode gelöscht: 0x 00 00 00 00 nn nn nn nn RTC6 ERROR.
Kapitel 12.4 "Code-Beispiel 2 (C#)", Seite 357	Redaktionelle Erweiterung.
EthMaxTimeout, Seite 386	Software-Änderung. Neuer Tag.
Änderungsindex, Seite 476	



Notizen