

Referenzhandbuch

VCS-API

Funktionsbibliothek

VisualControl
Application
Programming
Interface

REV 30.11.2005

LPKF
Motion & Control GmbH
Mittelbergstraße 17
98527 Suhl • Deutschland
Tel. +49-(0) 36 81-89 24-0
Fax +49-(0) 36 81-89 24-44
E-Mail: info@lpkf-mc.de
Internet: <http://www.lpkf-mc.de/>



Bestimmungsgemäße Verwendung

Dieses Handbuch dient zur Unterstützung bei der Programmierung von Applikationen mit der VisualControl-Funktionsbibliothek von LPKF.

Haftungsausschluss

Wir garantieren die Fehlerfreiheit des Produktes im Sinne unserer Werbung, der von uns herausgegebenen Produktinformationen und dieses Handbuchs. Darüber hinausgehende Produkteigenschaften werden nicht zugesagt. Wir übernehmen keine Verantwortung für die Wirtschaftlichkeit oder fehlerfreie Funktion bei Einsatz für einen anderen Zweck als dem im Abschnitt „Bestimmungsgemäße Verwendung“ definierten, speziell wenn das Produkt ohne Rücksprache mit uns in andere Softwarepakete und Systeme integriert wird.

Schadenersatzansprüche sind generell ausgeschlossen, ausgenommen bei Nachweis von Vorsatz, grober Fahrlässigkeit des Herstellers, oder Fehlen zugesicherter Eigenschaften. Wird das Produkt in Umgebungen eingesetzt, für die es nicht geeignet ist oder die nicht dem üblichen Stand der Technik entsprechen, so sind wir für die Folgen nicht verantwortlich.

Ferner lehnen wir die Verantwortung für Schäden an Anlagensystemen im Umfeld des Produktes ab, die auf eine Fehlfunktion des Produktes oder Fehler in der Bedienungsanleitung zurückzuführen sind.

Wir behalten uns das Recht auf Änderungen ohne spezielle Mitteilung vor. In keinem Fall übernehmen wir die Verantwortung für irgendwelche Neben- oder Folgeschäden oder entgangene Gewinne, die aus auf dieses Handbuch bezogenen Tätigkeiten entstehen, speziell wenn auf die Möglichkeit solcher Schäden hingewiesen wurde und sie bekannt sein müssten.

Wir sind nicht verantwortlich für die Verletzung von Patent- und anderen Rechten Dritter außerhalb der Bundesrepublik Deutschland.

Urheberrecht

© Copyright *LPKF Motion & Control GmbH*. Alle Rechte vorbehalten.

Diese Publikation oder Teile daraus dürfen ohne ausdrückliche Genehmigung von *LPKF Motion & Control GmbH* nicht kopiert oder Dritten zugänglich gemacht werden. Handbücher dienen nur zum persönlichen Gebrauch.

Warenzeichen

Die Marke LPKF ist Eigentum der *LPKF Laser & Electronics AG* bzw. der mit ihr verbundenen Unternehmen. Alle in der Publikation verwendeten Warenzeichen oder eingetragenen Warenzeichen werden als Eigentum ihrer Besitzer anerkannt.

Wichtige Instruktionen

Wichtige Informationen und Instruktionen für die Sicherheit von Personal und Geräten werden wie folgt angezeigt:



Vorsicht:

Wichtige Hinweise, die bei Nichtbeachtung zu Geräteschäden führen können.



Hinweis:

Spezielle Anmerkungen.

Unterstützung

Telefon

+49-(0)3681-8924-0

Fax

+49-(0)3681-8924-44

E-Mail

info@lpcf-mc.de

Internet

<http://www.lpcf-mc.de>

Firmensitz

LPKF Motion & Control GmbH • Mittelbergstraße 17 • 98527 Suhl • Deutschland

Inhaltsverzeichnis

Das VisualControl - Application Programming Interface

Vorbereitungen	13
Funktionsweise des API	15
Grundlegendes zum Handbuch	17
Kurzreferenz	18

Funktionsübersicht

1 Initialisierung

1.1 Übersicht

1.2 Initialisierungsfunktionen

VCS_Begin	22
VCS_End	22
VCS_GetAPIVersionNo	23
VCS_SelectMsgLanguage	24
VCS_SelectThreadPriority	25
VCS_SetCtrlType	26
VCS_DefineParamsCOM	27
VCS_DefineBaudrateCOM	29
VCS_DefineInterfaceCOM	30
VCS_OpenChannelCOM	31
VCS_CloseChannel	32
VCS_UserLogin	33
VCS_DebugChannel	34
VCS_DebugMode	35
VCS_AutomaticCfgMode	36
VCS_Define_CAA	37

1.3 Beispiel Initialisierungsfunktionen

2 Timeoutsteuerung

2.1 Übersicht

2.2 Timeoutfunktionen

VCS_GetTimeout	41
VCS_SetTimeout	41
VCS_GetMoveTimeout	42
VCS_SetMoveTimeout	42

2.3 Beispiel Timeoutfunktionen

3 Bewegungsfunktionen

3.1 Übersicht

3.2 Allgemeine Bewegungsfunktionen

3.2.1 Kreis / Kreisbogen	45
_move_arc_abs	45
_move_arc_rel	46
3.2.2 Vektoren	47
_move_line_abs	47
_move_line_rel	48
3.2.3 Referenzfahrt	49
_move_axis_home	49
_move_home	49
3.2.4 Bewegungsbereich ausmessen	50
_measure_axis_motion_range	50
_measure_motion_range	51
3.2.5 Positionsabfrage	52
_get_moving_axes_reference	52
_get_axis_reference_position	53
_get_reference_position	54
3.2.6 Achsenselektierung	55
_set_moving_axes	55
3.2.7 Beispiel allgemeiner Bewegungsfunktionen	56

4 Parameterfunktionen

4.1 Übersicht

4.2 Allgemeine Parameterfunktionen

4.2.1 Geschwindigkeit	59
_get_axis_velocity	59
_set_axis_velocity	59
_get_velocity	60
_set_velocity	60
_get_trajectory_velocity	61
_set_trajectory_velocity	61
4.2.2 Beschleunigung	62
_get_axis_acceleration	62
_set_axis_acceleration	62
_get_acceleration	63
_set_acceleration	63
4.2.3 Bewegungsbereich	64
_get_axis_limits	64
_set_axis_limits	65
_get_limits	66
_set_limits	67
4.2.4 Stabilisierungszeit	67
4.2.5 Wartezeit	68
_wait_time	68

4.2.6	Auflösung	68
	_get_axis_resolution.	68
	_get_resolution	69
4.2.7	Beispiel allgemeiner Parameterfunktionen	69
4.3	Steuerungsspezifische Parameterfunktionen	
4.3.1	Parameterfunktionen	71
	get_parameter	71
	set_parameter	72
4.3.2	Beispiel steuerungsspezifischer Parameterfunktionen	73
5	Steuerfunktionen	
5.1	Übersicht	
5.2	Allgemeine Steuerungsfunktionen	
	_direct_go	75
	_direct_halt	75
	_direct_clear_buffer	76
5.3	Steuerungsspezifische Steuerungsfunktionen	
5.3.1	Zusatzfunktionen	78
	burn_parameter_set	78
	execute	79
6	I/O-Steuerung	
6.1	Übersicht	
6.2	Allgemeine Funktionen zur I/O-Steuerung	
6.2.1	Port lesen	81
	_read_io_port	81
6.2.2	Port schreiben.	82
	_write_io_port	82
6.2.3	Beispiel zu allgemeinen Funktionen zur I/O-Steuerung	83
7	Fehlerbehandlung	
7.1	Übersicht	

7.2 Allgemeine Fehlerbehandlungsfunktionen	
VCS_GetError	86
VCS_GetErrorEx	87
VCS_GetErrorList	88
VCS_GetErrorListEx	89
VCS_GetWarningList	90
VCS_GetWarningListEx	91
VCS_GetErrorText	92
VCS_GetErrorTextEx	93
VCS_ReadMessageQueue	95
VCS_DefineErrorCallback	97
VCS_DefineErrorCallbackEx	98
VCS_DefineEventCallbackEx	99
VCS_DefineResultCallbackEx	100
VCS_ClearBuffer	101
VCS_ClearError	102
7.2.1 Beispiel allgemeine Fehlerbehandlungsfunktionen	103

8 Kommandoflusssteuerung

8.1 Übersicht	
8.2 Funktionen zur Kommandoflusssteuerung	
VCS_ConfigBufferDepth	108
VCS_SetSyncMode	109
VCS_SetBlockID	111
VCS_GetBlockID	113
VCS_GetBufferContent	115
8.2.1 Beispiel Funktionen zur Kommandoflusssteuerung	116

9 Scriptinterpreter

9.1 Übersicht	
9.2 Scriptinterpreterfunktionen des API	
VCS_KillScript	119
VCS_ExecScript	120
VCS_ExecScriptFile	121
VCS_ExecScriptEx	122
VCS_ExecScriptFileEx	125
VCS_ExecScriptCallbackEx	128
VCS_ExecScriptFileCallbackEx	131
VCS_PutScript	134
VCS_WaitForCommand	135
VCS_WaitForCommandEx	137
VCS_GetStatusCommand	140
VCS_GetStatusCommandEx	142

9.3 Sprachbeschreibung LPKF MotionScript

Leerzeichen und Kommentare	145
Bezeichner (IDENT)	145
Zahlen (NUMBER)	146
Matrizen	146
Variablen	146
Globale Variablen	147
Strings (STRING)	147
Objekte (dotident)	147
Aliases	148
Gültigkeitsbereiche von Variablen, Aliases, Objekten	148
Indizierung	148
Anweisungen	149
Arithmetische Operatoren (+, -, *, /)	149
Relationale Operatoren (==, !=, <, <=, >, >=)	150
Vorrang (Priorität) von Operatoren	150
Klammern	150
Alternative (if - elseif - else)	151
Wiederholung (while)	151
Vorzeitiger Abbruch einer Schleife (break)	152
Funktions- und Prozedurdefinition	152
Funktions- und Prozeduraufruf	153
Vorzeitiger Abbruch einer Funktion (return)	153
Parameterübergabe an Funktionen	154
Fehlerzustände Erkennen und Abfangen (try-catch-Notation)	155

9.4 Prädefinierte interne Funktionen

9.4.1 Gruppe allgemeine Funktionen	157
who	157
rows	157
columns	157
all	158
any	158
identcmp	158
9.4.2 Gruppe Programmflusssteuerung	159
wait	159
time	159
timediff	159
exit	160
bufferdepth	160
log_value	160
application_callback	160
syncmode	161
resync	161
user_login	161
error	162

9.4.3 Gruppe Mathematische Funktionen	163
sin	163
cos	163
tan	163
exp	163
log	163
sqrt	164
asin	164
acos	164
atan	164
atan2	164
transpose	165
floor	165
abs	165
round	165
9.4.4 Gruppe Bit Logik	166
not	166
and	166
or	166
xor	166
9.5 Prädefinierte ALIASES	
axis_x	167
axis_y	167
axis_z	167
axis_z1	167
axis_z2	167
axis_z3	167
io_0	168
io_1	168
io_2	168
io_3	168
io_4	168
io_5	168
pi	169
PI	169
9.6 Prädefinierte globale Variablen	
_config_id	170
_machine_type	170
_last_error	170
_n_axes	170
_axis_pid	171
_n_mover	171
_mover_n_axes	171
_mover_first_axis	171
_mover_pid	172
_active_mover	172
_n_io	172
_io_pid	172
_axis_resolution	172
_axis_reference	173
_axis_lower_limit	173
_axis_upper_limit	174
_moving_axes	174
_moving_axis	174
_moving_axis_resolution	175
9.7 Funktion und Anwendung des LPKF MotionScript an Hand eines	

Beispiels

10 weitere Funktionen - Hilfsfunktionen

10.1 Übersicht

10.2 weitere Funktionen / Hilfsfunktionen

VCS_GetConfigData	179
VCS_SaveParameters	184
VCS_RestoreParameters	185
VCS_DownloadAvailable	186
VCS_DownloadFiles	187
VCS_GetStickMode	189
VCS_SetStickMode	190
VCS_CheckMachineInfo	191
VCS_CheckMachine	193

10.3 Beispiel weitere Funktionen - Hilfsfunktionen

11 Fehlercodes

11.1 Übersicht

11.2 EVCS-Fehlercodes des API

11.3 EVCS-Fehlercodes der Steuerung SMCU II

Das VisualControl - Application Programming Interface

Das Referenzhandbuch beschreibt den Aufbau der VisualControl-API-Funktionen und deren Verwendung zur Erstellung von Bedienprogrammen für die LPKF-Steuerungen SMCU, MotionSystem und SMCU II und zukünftiger Steuerungen. Die Funktionsbibliothek ist als 32-Bit-Version aufgebaut und kann von verschiedenen Programmiersprachen aufgerufen und verwendet werden.

Das VisualControl-API ist ein sehr leistungsfähiges Programmierwerkzeug, das den Anwender bei der Erstellung seiner eigenen Programme unterstützt und von der Realisierung der komplexen Kommunikationsprotokolle zwischen PC und Steuerung entlastet. Alle erforderlichen Kommunikationsprotokolle sind bereits in dem API implementiert. Der Anwender öffnet in seinem Programm lediglich einen logischen Kanal und das API organisiert das gesamte Kommunikationsprotokoll mit der zugeordneten physikalischen Schnittstelle automatisch im Hintergrund.

Alle Funktionen sind in einer bestimmten Art und Weise benannt. Sie beginnen alle mit VCS_, um sie als Funktionen des VisualControl-Systems zu kennzeichnen.

Für das grafische Programmsystem LabView von National Instruments Corp. wird eine Funktionsbibliothek auf Basis des VisualControl-API mitgeliefert. Mit dieser Erweiterung lassen sich unter LabView sehr leicht anspruchsvolle Programmanwendungen für Steuerungsaufgaben erstellen.

Systemvoraussetzung

- ◆ 32bit-Betriebssysteme: Microsoft Windows 9x, Windows NT 4.0 und Windows 2000 und XP; Neuere Varianten der Betriebssysteme sollen ebenfalls unterstützt werden, sobald sie verfügbar, stabil und verbreitet sind.
- ◆ Entwicklungsumgebung: Microsoft Visual C++ ab Version 4.0 oder VisualBasic ab Version 4.0
- ◆ Windows 3.1x oder ältere Versionen, sowie die Betriebssysteme OS/2 und UNIX und deren Derivate werden derzeit nicht unterstützt.

Für die Benutzung der LabView-Funktionsbibliothek ist LabView ab Version 6.0 erforderlich.

Vorbereitungen

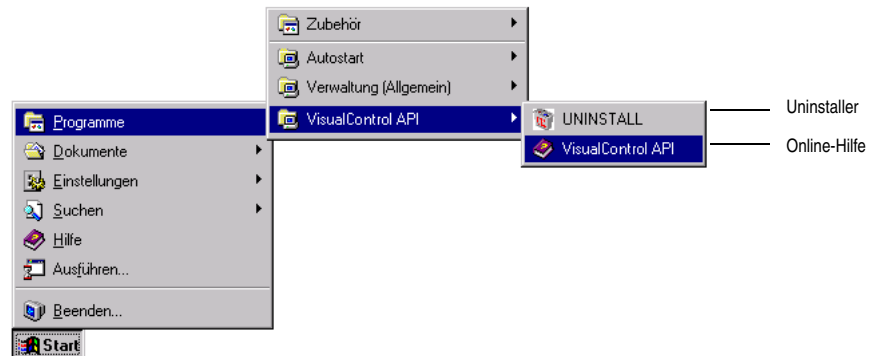
Das API installieren

Installieren Sie das API wie folgt:

- (1) Datei *setup.exe* auf der Installations-CD ausführen: Die VisualControl-API-Installation startet.
- (2) Zielverzeichnis eingeben (Voreinstellung: *C:\Programme\VC_API*) und die Schaltfläche **Weiter** anklicken
- (3) Kontrollkästchen für zu installierende Komponenten ankreuzen: API (notwendig)

Das Zielverzeichnis mit den Dateien des API wird angelegt. Die zu installierenden Dateien werden in das Zielverzeichnis kopiert. Wenn die Installation beendet ist, erscheint auf dem Bildschirm die Meldung „Die Installation ist vollständig!“.

In der Startleiste wird der Eintrag „VisualControl API“ im Ordner „Programme“ angelegt. Hier befinden sich die Online-Hilfe und der Uninstaller für das VisualControl-API.



(4) Betriebssystem neu starten, um die Änderungen der Registrierungsdateien wirksam zu machen.

Folgende Dateien des VCS-API werden installiert:

- ◆ *Unwise.exe* [Uninstaller]
- ◆ *VC_API.dll* [VCS-API-DLL]
- ◆ *VC_APIRES.dll* [VCS-API-Ressourcen-DLL]
- ◆ *VC_API.chm* [Hilfedatei]
- ◆ *VC_API.lib* [VCS-Bibliothek]
- ◆ *VCS.h* [Deklarationsdatei]
- ◆ *vc_api.ini* [Defaultkonfigurationsdatei]
- ◆ *vc_api.mcf* [Metakonfigurationsdatei]
- ◆ *CAA_lib.dll* [caa-Funktionsbibliothek]
- ◆ *readme.txt* [Textdatei mit den letzten Ergänzungsinformationen zum API]

Online-Hilfe	Die Online-Hilfe enthält die Syntaxbeschreibung der API-Funktionen. Um die Online Hilfe zu starten, doppelklicken Sie auf die Datei <i>VC_API.chm</i> , welche sich im Verzeichnis des API befindet oder wählen sie aus dem Startverzeichnis den Eintrag für die Online-Hilfe unter Start > Programme > VC API > VisualControl API (siehe Abb. oben).
Handbuch	Die Funktionen des API sind in Gruppen gegliedert. Jede Funktionsgruppe wird in einem Abschnitt des Handbuchs beschrieben. In weiteren Abschnitten findet sich ein API-Programmbeispiel in der Programmiersprache C, eine Liste der Rückgabewerte der API-Funktionen sowie die Anleitung zur Installation und Benutzung der Erweiterung für die LabView-Bibliothek.
Programmbeispiel	Beispiele zu den einzelnen Funktionen entnehmen Sie bitte den im Text explizit aufgeführten Programmbeispielen.

Funktionsweise des API

Die Bibliothek kann mehrere Steuerungen verwalten. Maximal ist es möglich, 16 Steuerungen über die Bibliothek gleichzeitig zu betreiben. Die Bibliothek ist multithreadingfähig und kann daher die aufgerufenen Funktionen zeitlich parallel abarbeiten.

Für jede Steuerung muss sich ein Programm:

- ◆ bei der Bibliothek anmelden (**VCS_Begin**)

Durch die Anmeldung erhält das aufrufende Programm ein Handle, über das die Bibliothek die entsprechende Steuerung adressiert und verwaltet.

- ◆ einen (Kommunikations-)Kanal zur gewünschten Steuerung öffnen (**VCS_OpenChannelCOM**)

Durch das Öffnen des Kanals zur Steuerung erhält das aufrufende Programm die Möglichkeit mit der durch das Handle (**VCS_Begin**) adressierten Steuerung über API-Methoden anzusprechen und zu steuern.

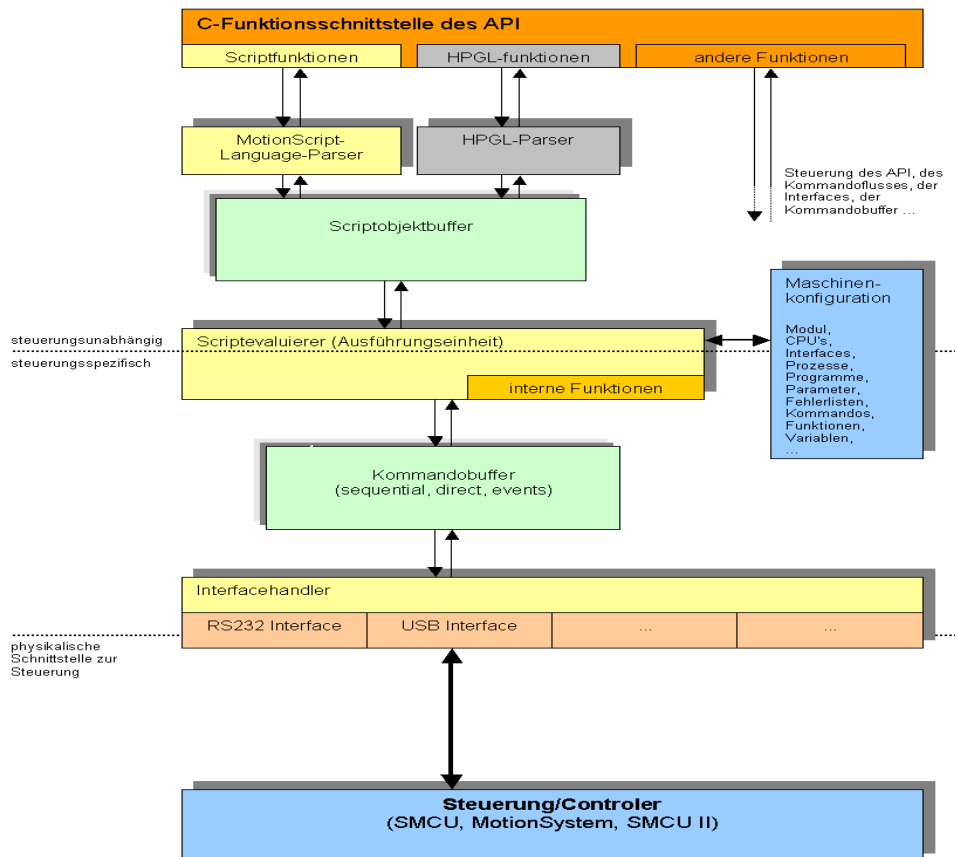


Bild: interne Struktur des API

Ein Programm kann über das API mit einer oder auch mit mehreren Steuerungen arbeiten. Die Bibliothek stellt die Methoden zur Verfügung, um die Steuerungen mit einfachen Mitteln anzusprechen. Der korrekte Ablauf liegt in der Verantwortung der Programme.

Folgende Arbeitsweisen von Programmen können durch das API realisiert werden:

- ◆ Programm arbeitet mit einer Steuerung (Normalfall)
- ◆ mehrere Programme arbeiten mit jeweils verschiedenen Steuerungen
- ◆ ein Programm arbeitet mit mehreren Steuerungen
- ◆ mehrere Programme arbeiten mit jeweils einer oder mehreren Steuerungen, wobei für jede Steuerung gilt, nicht zeitgleich von verschiedenen Programmen angesteuert werden zu können

Aufrufreihenfolge für
ein Programm an
einer Steuerung

Die Aufrufreihenfolge für den einfachen Fall eines Programms an einer Steuerung sieht folgendermaßen aus:

```
-----  
VCS_Begin()  
    VCS_SetCtrlType() (optional)  
    VCS_DefineInterfaceCOM() (optional)  
    VCS_DefineBaudrateCOM() (optional)  
    VCS_DefineParamsCOM() (optional)  
    VCS_OpenChannelCOM()  
  
    // Kanal ist geöffnet,  
    // eigentliche Funktionen der Applikation hier  
    ...  
    VCS_CloseChannel()  
VCS_End()  
-----
```

Grundlegendes zum Handbuch

Aufbau des Handbuchs

Kapitel	Inhalt / Verwendung des Kapitels
	Grundlagen, Installation, Schichtenmodell und Grundstruktur von Programmen, Kurzreferenz
	Funktionsübersicht
1	Initialisierung von auf dem VisualControl-API basierenden Anwendungen
2	Timeoutsteuerung
3	Bewegungsfunktionen
4	Parameterfunktionen
5	Steuerfunktionen
6	Funktionen zur I/O-Steuerung
7	Fehlerbehandlung
8	Kommandoflusssteuerung
9	Scriptinterpreter und -funktionen
10	weitere Funktionen - Hilfsfunktionen
11	Fehlercodes
	Programmierung von VisualControl-Anwendungen mit der grafischen Programmiersprache LabView
	Index zum Inhalt des Handbuchs
	Index zu den Funktionen

Textkonventionen

Formatierung/Symbol	Bedeutung
[]	Optionale Worte und Parameter, die in einem Ausdruck nicht benötigt werden, aber verwendet werden können, um diesen eindeutiger zu machen oder um Optionen zu ergänzen.
	Austauschbare Ausdrücke
()	Zwingend benötigte Klammern, die den Ausdruck einschließen müssen.
<i>kursiv</i>	Dateien, Pfade, Menü- und Fensterbezeichnungen
< <i>kursiv</i> >	Parameter oder Ausdrücke, die einen bestimmten Wert darstellen oder ergeben.
=>	Verweis auf Begriffe
>	Verschachtelte Menüs, z.B. <i>Insert > Submenü...</i>
GROSSBUCHSTABEN	Hervorhebung
gesperrt	Kennzeichnung von Zeichen, die per Tastatur eingegeben oder übertragen werden: Code-Abschnitte, Programmierbeispiele und Meldungen
Console	Syntax von Funktionen, z.B. <code>short VCS_Begin</code>
"	Übergabewert oder Rückgabewert, z.B. „0“



Hinweis:

Zur Syntax der C++ / VBasic Programme siehe die Handbücher zu diesen Programmiersprachen.

Kurzreferenz

API-Funktion	Kurzname der Funktion	Erklärung	Seite
Initialisierungsfunktionen			
VCS_Begin	Begin	meldet Anwendung an dem API an	22
VCS_End	End	meldet Anwendung an dem API ab	22
VCS_GetAPIVersionNo	GetAPIVersionsNo	Abfrage der API Version	23
VCS_SelectMsgLanguage	SelectMsgLanguage	Spracheinstellung des API	24
VCS_SelectThreadPriority	SelectThreadPriority	Threadprioritäteneinstellung des API	25
VCS_SetCtrlType	SetCtrlType	wählt den Steuerungstyp am Kommunikationskanal und konfiguriert die Maschinenparameter	26
VCS_AutomaticCfgMode	AutomaticCfgMode	schaltet den automatischen Konfigurationsmode ein/aus	36
VCS_DefineParamsCOM	DefineParamsCOM	wählt serielle Schnittstellenparameter	27
VCS_DefineBaudrateCOM	DefineBaudrateCOM	wählt Baudrate der seriellen Schnittstelle	29
VCS_DefineInterfaceCOM	DefineInterfaceCOM	wählt den Interfacetypen und die Seriennummer der Steuerung	30
VCS_UserLogin	UserLogin	wählt den Autorisierungslevel des Anwenders	33
VCS_OpenChannelCOM	OpenChannelCOM	öffnet logischen Kommunikationskanal	31
VCS_CloseChannel	CloseChannel	schließt logischen Kommunikationskanal	32
VCS_DebugChannel	DebugChannel	schaltet DebugLogmodus ein/aus	34
VCS_DebugMode	DebugMode	schaltet Debugmodus ein/aus	35
VCS_Define_CAA	Define_CAA	schaltet die CAA-Feldkorrektur ein/aus	37
Timeout-Funktionen			
VCS_GetTimeout	GetTimeout	fragt Timeout für Steuerkommandos ab	41
VCS_SetTimeout	SetTimeout	setzt Timeout für Steuerkommandos	41
VCS_GetMoveTimeout	GetMoveTimeout	fragt Timeout für Bewegungskommandos ab	42
VCS_SetMoveTimeout	SetMoveTimeout	setzt Timeout für Bewegungskommandos	42
allgemeine Bewegungsfunktionen			
_set_moving_axes		selektiert Achsen und Achsreihenfolge eines Moverprozesses für nachfolgende Bewegungsfunktionen	55
_move_arc_abs		fährt Kreisbogen um absolute Koordinate	45
_move_arc_rel		fährt Kreisbogen um relative Koordinate	46
_move_line_abs		führt einen absoluten Bewegungsvektor aus	47
_move_line_rel		führt einen relativen Bewegungsvektor aus	48
_move_axis_home		fährt eine Achse in ihre Referenzposition	49
_move_home		fährt alle Achsen in ihre jeweiligen Referenzpositionen	49
_measure_axis_motion_range		ermittelt den Bewegungsbereich einer Achse	50
_measure_motion_range		ermittelt den Bewegungsbereich aller Achsen	51
_get_axis_reference_position		liefert die aktuelle Position einer Achse	53
_get_moving_axes_reference		liefert die aktuellen Positionen aller selektierten Achsen	52
_get_reference_position		liefert die aktuellen Positionen aller Achsen	54
allgemeine Parameterfunktionen			
_set_axis_velocity		setze Geschwindigkeit einer Achse	59
_set_velocity		setze Geschwindigkeit aller Achsen	60
_get_axis_velocity		lese Geschwindigkeit einer Achse	59
_get_velocity		lese Geschwindigkeit aller Achsen	60
_set_trajectory_velocity		setze Bahngeschwindigkeit	61
_get_trajectory_velocity		lese Bahngeschwindigkeit	61
_set_axis_acceleration		setze Beschleunigung einer Achse	62

API-Funktion	Kurzname der Funktion	Erklärung	Seite
<code>_set_acceleration</code>		setze Beschleunigungen aller Achsen	63
<code>_get_axis_acceleration</code>		lese Beschleunigung einer Achse	62
<code>_get_acceleration</code>		lese Beschleunigungen aller Achsen	63
<code>_set_axis_limits</code>		setzt Bewegungsbereichsgrenzen einer Achse	65
<code>_set_limits</code>		setzt Bewegungsbereichsgrenzen aller Achsen	67
<code>_get_axis_limits</code>		lese Bewegungsbereichsgrenzen einer Achse	64
<code>_get_limits</code>		lese Bewegungsbereichsgrenzen aller Achsen	66
<code>_wait_time</code>		Wartezeit	68
<code>_get_axis_resolution</code>		lese Achsauflösung einer Achse	68
<code>_get_resolution</code>		liest Achsauflösung aller Achsen	69
steuerungsspezifische Parameterfunktionen			
<code>get_parameter</code>	<code>get_parameter</code>	liest Parameter von der Steuerung	71
<code>set_parameter</code>	<code>set_parameter</code>	setzt Parameter der Steuerung	72
allgemeine Steuerungsfunktionen			
<code>_direct_go</code>	<code>direct go</code>	startet die sequentielle Kommandoabarbeitung	75
<code>_direct_halt</code>	<code>direct halt</code>	hält die sequentielle Kommandoabarbeitung an	75
<code>_direct_clear_buffer</code>	<code>direct clear buffer</code>	löscht den sequentiellen Kommandobuffer	76
steuerungsspezifische Steuerungsfunktionen			
<code>burn_parameter_set</code>	Burn Parameter Set	Speichern des aktuellen Parametersatzes in den nicht-flüchtigen Speicher der Steuerung	78
<code>execute</code>	Execute	Ausführen eines steuerungsinternen Programmes (Makros)	79
allgemeine Funktionen zur I/O-Steuerung			
<code>_read_io_port</code>	<code>read io port</code>	lese I/O-Port	81
<code>_write_io_port</code>	<code>write io port</code>	schreibe I/O-Port	82
allgemeine Fehlerbehandlungsfunktionen			
<code>VCS_GetError</code>	<code>GetError</code>	erfragt den aktuellen gehaltenen Fehlercode	86
<code>VCS_GetErrorEx</code>	<code>GetErrorEx</code>	erfragt den aktuellen gehaltenen Fehlercode aus der Fehlercodequeue und seiner ProzessID	87
<code>VCS_GetErrorList</code>	<code>GetErrorList</code>	erfragt den aktuellen gehaltenen Fehlercode aus der Fehlercodequeue	88
<code>VCS_GetErrorListEx</code>	<code>GetErrorListEx</code>	erfragt den aktuellen gehaltenen Fehlercode und seiner ProzessID aus der Fehlercodequeue	89
<code>VCS_GetWarningList</code>	<code>GetWarningList</code>	erfragt den gehaltenen Warnungscode aus der Warnungscodequeue	90
<code>VCS_GetWarningListEx</code>	<code>GetWarningListEx</code>	erfragt den aktuellen gehaltenen Warnungscode und seiner ProzessID aus der Warnungscodequeue	91
<code>VCS_GetErrorText</code>	<code>GetErrorText</code>	liefert den Text zu einem Fehler- oder Warnungscode	92
<code>VCS_GetErrorTextEx</code>	<code>GetErrorTextEx</code>	liefert den Text zu einem Fehler- oder Warnungscode samt dessen ProzessID	93
<code>VCS_ReadMessageQueue</code>	<code>ReadMessageQueue</code>	liefert EventMessages des API	95
<code>VCS_DefineErrorCallback</code>	<code>DefineErrorCallback</code>	definiert Callbackfunktion für den Fehlerfall	97
<code>VCS_DefineErrorCallbackEx</code>	<code>DefineErrorCallbackEx</code>	definiert Callbackfunktion für den Fehlerfall	98
<code>VCS_DefineEventCallbackEx</code>	<code>DefineEventCallbackEx</code>	definiert Callbackfunktion für Events	99
<code>VCS_DefineResultCallbackEx</code>	<code>DefineResultCallbackEx</code>	definiert Callbackfunktion für Application Callbacks	100
<code>VCS_ClearBuffer</code>	<code>ClearBuffer</code>	löscht die Buffer des API und/oder der Steuerung	101
<code>VCS_ClearError</code>	<code>ClearError</code>	setzt den Fehlerzustand der Steuerung und des API zurück	102
Funktionen zur Kommandoflusssteuerung			

API-Funktion	Kurzname der Funktion	Erklärung	Seite
VCS_ConfigBufferDepth	ConfigBufferDepth	erlaubt maximale Anzahl Kommandos im Steuerungsbuffer (asynchroner Modus des API)	108
VCS_SetSyncMode	SetSyncMode	schaltet das API in den asynchronen/synchronen Arbeitsmode bzw. dient zum Resynchronisieren	109
VCS_SetBlockID	SetBlockID	setzt eine BlockID	111
VCS_GetBlockID	GetBlockID	liefert aktuelle BlockID	113
VCS_GetBufferContent	GetBufferContent	liefert aktuellen Füllstatus des Script- und des Kommandobuffers des API	115
Scriptinterpreterfunktionen des API			
VCS_KillScript	KillScript	Abbruch der momentanen Scriptabarbeitung	119
VCS_ExecScript	ExecScript	Ausführen eines Scriptstrings	120
VCS_ExecScriptFile	ExecScriptFile	Ausführen einer Scriptdatei	121
VCS_ExecScriptEx	ExecScriptEx	Ausführen eines Scriptstrings (mit Rückgabewerten)	122
VCS_ExecScriptFileEx	ExecScriptFileEx	Ausführen einer Scriptdatei (mit Rückgabewerten)	125
VCS_ExecScriptCallbackEx	ExecScriptCallbackEx	Ausführen eines Scriptstrings (mit Rückgabewerten)	128
VCS_ExecScriptFileCallbackEx	ExecScriptFileCallbackEx	Ausführen einer Scriptdatei (mit Rückgabewerten)	131
VCS_PutScript	PutScript	Ausführen einer Scriptanweisung (asynchron)	134
VCS_WaitForCommand	WaitForCommand	Wartet auf das Abarbeitungsende einer mit PutScript gesendeten Anweisung	135
VCS_GetStatusCommand	GetStatusCommand	Erfragt den Status einer mit PutScript gesendeten Anweisung	140
VCS_WaitForCommandEx	WaitForCommandEx	Wartet auf das Abarbeitungsende einer mit PutScript gesendeten Anweisung	137
VCS_GetStatusCommandEx	GetStatusCommandEx	Erfragt den Status einer mit PutScript gesendeten Anweisung	142
weitere Funktionen - Hilfsfunktionen des API			
VCS_GetConfigData	GetConfigData	fragt Konfigurationsdaten der Steuerung ab	179
VCS_SaveParameters	SaveParameters	Export von Steuerungsparametern in eine Datei	184
VCS_RestoreParameters	RestoreParameters	Import von Steuerungsparametern aus einer Datei	185
VCS_DownloadAvailable	DownloadAvailable	Abfrage, ob bestimmte Datei(en) in Steuerung existiert und heruntergeladen werden kann	186
VCS_DownloadFiles	DownloadFiles	Herunterladen bestimmter Datei(en) aus der Steuerung auf den PC	187
VCS_GetStickMode	GetStickMode	Abfrage des Status des API-Trackballmodes	189
VCS_SetStickMode	SetStickMode	Setzen des API-Trackballmodes	190
VCS_CheckMachineInfo	CheckMachineInfo	Abfrage auf vorhandene Testroutinen	191
VCS_CheckMachine	CheckMachine	Ausführen vorhandener Testroutinen	193