rexroth
A Bosch Company

Application Manual

G-Code Runtime App

G-Code Interpreter for ctrlX OS 03VRS

R911425246 Edition 01

Disclaimer

The data specified above only serve to describe the product. As our products are constantly being further developed, no statements concerning a certain condition or suitability for a certain application can be derived from our information. The information given does not release the user from the obligation of own judgment and verification. It must be remembered that our products are subject to a natural process of wear and aging.

Table of contents

1 About this documentation

Editions of this documentation

Edition	Release date	Note
01	2024-10	First edition version GCO-V-0302

2 Important directions on use

2.1 Intended use

2.1.1 Introduction

Rexroth products are developed and manufactured to the state-of-the-art.

The products are tested prior to delivery to ensure operational safety and reliability.

⚠ WARNING

Personal injury and damage to property due to incorrect use of products!

The products may only be used as intended.

Failure to use the products as intended may cause situations resulting in property damage and personal injury.

NOTICE

Damages resulting from unintended use

Rexroth As the manufacturer does not assume any warranty, liability or compensatory claims for damages resulting from unintended use of the products. The user alone shall bear the risks of an unintended use of the products.

Before using Rexroth products, make sure that all the prerequisites for an intended use of the products are met:

- Personnel that in any way, shape or form uses Rexroth products must first read and understand the relevant safety instructions and be familiar with their intended use
- Leave hardware products in their original state, i.e., do not make any structural modifications. It its not permitted to decompile software products or alter source codes
- Do not install damaged or defective products or commission them
- It has to be ensured that the products have been installed as described in the relevant documentation

2.1.2 Areas of use and application

Products of the ctrlX series are suitable for Motion/Logic applications.

NOTICE	Products of the ctrlX series may only be used with the accessories, mounting parts, and other components specified in this documentation. Components that are not expressly mentioned must neither be attached nor connected. The same applies to cables and lines. Only to be operated with the hardware component configurations and combinations expressly specified and with the software and firmware specified in the corresponding documentations and functional descriptions.

Products of the ctrlX series are suitable for single-axis as well as for multi-axis drive and control tasks. Device types with different equipment and interfaces are available for using the system in specific applications.

Typical areas of application:

- Building automation
- IoT and Security Gateway or Device
- Handling & Robotic

Controls of the ctrlX CORE series may only be operated under the mounting and installation conditions, in the position of normal use and under the ambient conditions (temperature, degree of protection, humidity, EMC, etc.) specified in the related documentations.

2.2 Unintended use

"Unintended use" refers to using the ctrlX products outside of the above-mentioned areas of application or under operating conditions and technical data other than described and specified in the documentation.

ctrlX products must not be used if they are exposed to following conditions:

- Operating conditions that do not meet the specified ambient conditions. Operation under water, under extreme temperature fluctuations or under extreme maximum temperatures is prohibited
- Applications that have not been expressly authorized by Rexroth

3 Safety instructions

The Safety instructions contained in the available application documentation feature specific signal words (DANGER, WARNING, CAUTION or NOTICE) and, where required, a safety alert symbol (in accordance with ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and identifies the hazard severity.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words DANGER, WARNING and CAUTION, is used to alert the reader to personal injury hazards.

The Safety instructions in this documentation are designed as follows:

⚠ DANGER	In case of non-compliance with this safety instruction, death or serious injury **will** occur.
⚠ WARNING	In case of non-compliance with this safety instruction, death or serious injury **could** occur.
⚠ CAUTION	In case of non-compliance with this safety instruction, minor or moderate injury could occur.
NOTICE	In case of non-compliance with this safety instruction, property damage could occur.

4 Introduction and overview

The G-Code Runtime app is used to compile G-code (DIN 66025-1/ISO 6983-1) into Motion commands (Motion app).

5 Prerequisites

License

The following licenses are required to use the G-Code Runtime app:

License	Type code	Part number
G-Code Runtime App	SWL-XC*-GCO-GCORUN-TIME*-BANN	R911420238
Motion app (for 4 axes)	SWL-XC*-MOT-STDMO-TION****-NNNN	R911400503
Motion in-app option: Cartesian Kinematics	SWL-XC*-MOT-CARTE-SIAN****-NNNN	R911400509

Installation

ctrlX OS side navigation *“Settings → Apps”*

For more information on the installation, please refer to the following links:

ctrlX OS Runtime, Application Manual: ➦ Licenses – Overview

ctrlX OS Runtime, Application Manual: ➦ App basics

ctrlX OS Runtime, Application Manual: ➦ Licensing notes

➦ Motion App - Motion Runtime Environment for ctrlX OS, Application Manual

6 Basic principles of NC programming

6.1 Introduction and overview

An NC control receives all information required to machine a workpiece on a machine tool via an NC program (parts program).

The structure of such an NC program is variable so that nearly any kind of workpiece can be machined with a wide range of technologies. With ctrlX Motion CNC, the technologies "additive manufacturing", "dispensing" and "bending" are currently supported. The parts program contains not only information on motions describing the path of the tool relative to the workpiece, but also information on technologies.

Motion information is divided into individual elementary contour elements (straight lines, circles and spline cornering).

The control can then perform the motions for each of these geometrically simple contour elements in one machining step provided that all machining steps are specified in the NC program in the correct order and with all required boundary conditions. Among other things, the required boundary conditions consist of technology functions (velocities, speeds, etc.) and auxiliary machine functions (e.g. for coolants and axis clamping).

- Basic guidelines to structure an NC program are described in DIN 66025.
- The content of DIN 66025 "Program Structure for Numerically Controlled Machine Tools" (part 1 and 2) corresponds to the international standard: ISO/DIS 6983 and ISO/DP 6983 "Numerical control of machines".
- The G-code Interpreter in the G-Code Runtime app (app.gcode) compiles the NC program into Motion commands. The motion is controlled by the ➦ Motion app (app.motion).

6.2 Executing NC program

To execute a script, the installation of theG-Code Runtime app has to be completed successfully.

For the detailed information on the G-Code Interpreter, refer to the Data Layer path:

script/state/languages/gcode

Example

```
{
    "language": "gcode",
    "version": "4.2.0",
    "executable": "/var/snap/rexroth-automationcore/956/script/runner/rexroth-gcode/
script.gcode",

    "fileEnding": [
        "npg"
    ],
    "license": "SWL-XCx-GCO-GCORUNTIMEx-BANN",
    "licenseVersion": "1.0"
}
```

After the G-Code app has been installed, ensure that the license for the G-Code app is activated before you create a G-code instance.

6.2.1 Saving the NC program

ctrlX OS manages NC programs in the file system of the device.

For more information on the file system and access rights, as well as information on how to create subprograms, see the ➦ ctrlX OS Runtime, Application Manual.

All scripts are part of the configuration and should be stored in "Solution" to be processed further in ctrlX OS.

The default path of the NC script is:

$SNAP_COMMON/solutions/activeConfiguration/scripts/gcode

6.2.2 Starting NC program

Example of starting a G-code script instance using the command in the Data Layer

1. Create a script instance with Payload in the *script/instances* path.

   ℹ The name of the G-code instance has to match the name of the kinematics to be attached.

   ```
   {
            "name": “Kinematics_01”
            "language":  “gcode”
     }
   ```

   ℹ As of the G-Code app V02.06, the G-Code script instance is created automatically after the Motion app switches to the “running” status.

2. Commands to run the script in the Data Layer:
   - *script/instances/[name]/cmd/abort*
   - *script/instances/[name]/cmd/file*
   - *script/instances/[name]/cmd/reset*
   - *script/instances/[name]/cmd/string*

   Beispiel

   ```
   {
        "name":"activeConfiguration/script/gcode/main_script.npg"
        "param”:[]

   }
   ```

6.2.3 Status of the NC program execution

To retrieve status and diagnostic information, go to the following Data Layer paths:

- *script/instances//[name]/diag*
- *script/instances//[name]/state*

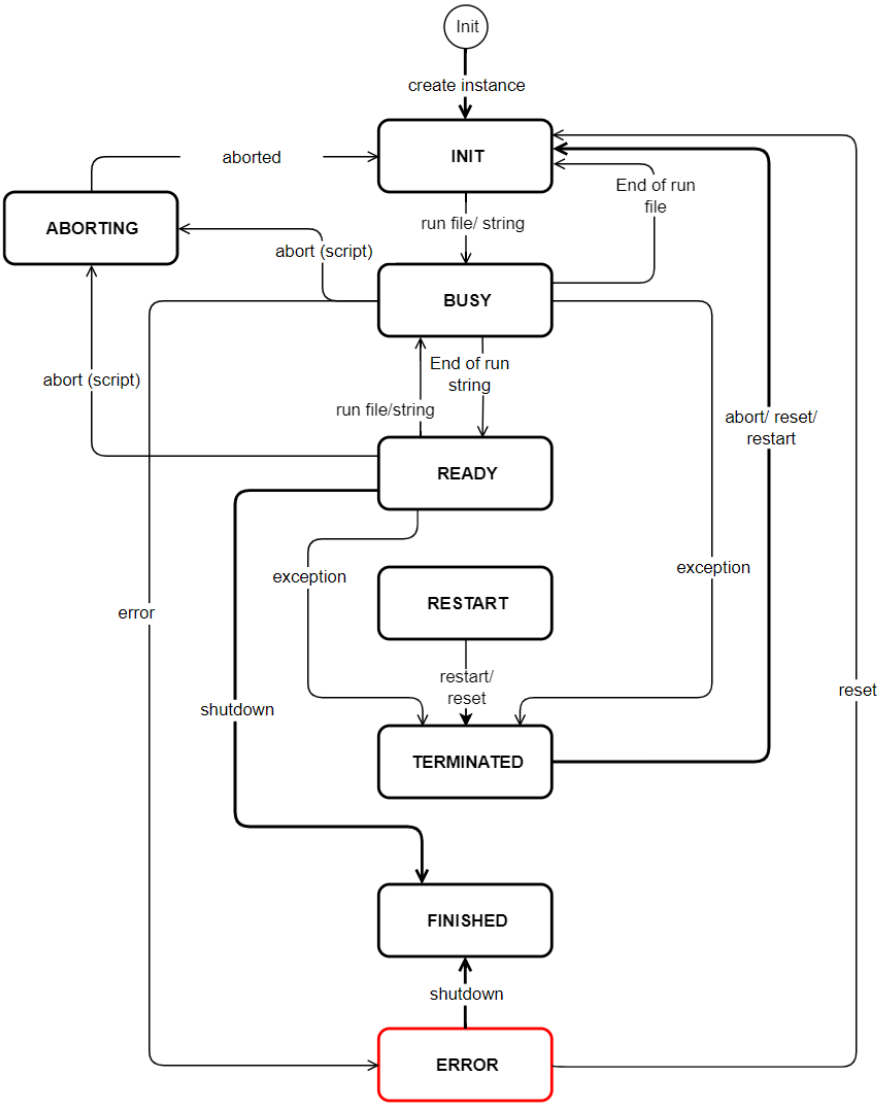

Fig. 1: State machine of the G-code instance

For more detailed information on the Script Interpreter:

- ctrlX OS Runtime, Application Manual, chapter ➦"Script parser/ interpreter(Python)"

6.3 Basic components of the NC program

6.3.1 Program block

An NC program consists of at least one program block.

The following applies to a program block:

- Only up to one contour element (e.g. straight line, arc) can be programmed for each program block.
- Blank lines before or after a program block are permitted to improve the organization/readability of the program code.
- A program block must not contain more than 512 characters.
- A program block is completed by the ASCII character <LINEFEED>.
- A program block consists of at least one program word (e.g. NC function, function parameter).

6.3.2 Commands

Commands are program words that directly or indirectly affect the tool path, the program sequence, the status or the response of the control.

For the available NC functions and the syntax rules required, refer to the chapter ➦NC functions with high-level language syntax.

Path functions are important among the instructions:

Path functions describe how a position is to be approached (e.g. straight line, circle, approaching motion in feed or rapid traverse).

Examples:
G0, G1, G2

Path functions are often programmed in the same block as position, path or radius specifications. In such cases, path functions always also initiate traversing motions.

Analogously, position or path specifications programmed in a program block without path function always trigger traversing motions since there is always an active path function.

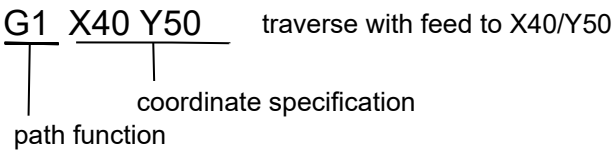

Fig. 2: Command structure

6.3.3 Additional conditions

Additional conditions are program words used to specify technologies or machining boundary conditions for the machine.

Important program words used as additional conditions include:

- F<number>
  Affects the feed of slave axes

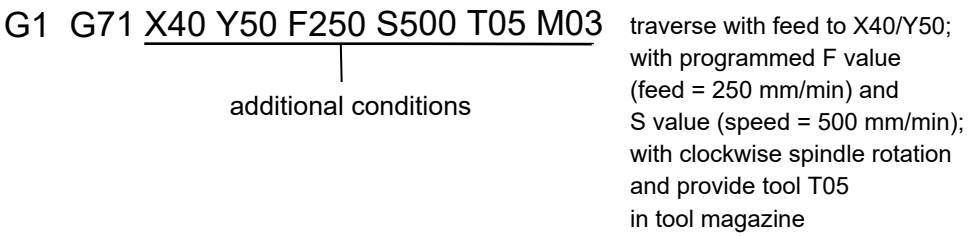

Fig. 3: Additional conditions

6.4 Program words

6.4.1 Introduction and overview

A program word is an NC function or a parameter with a value or a parameter list.

A parameter is addressed in the parts program using its syntax (address).

Each program word always consists of one or two subwords which can be combined as follows:

Table 1: Combinations of subwords

Program word	Partial character string 1	Subword 2	Example
NC function	Function syntax	-	G0 G54.1 WAIT
NC function with high-level language syntax	Function syntax	Parameter list	SetSignal(1) Contour(1)
Address word	Parameter syntax	Value	X-22.3 F4000

Description

G- and M-codes can occur with or without subword 2:

- For G- and M-codes with fixed internal functions, the numerical value is part of the syntax, e.g. G0, G17, G54, M0, M3, M19.
- For customized functions (subroutine calls and auxiliary functions), the numerical value is not part the syntax, but represents the value for the function.
  These functions are configured in the Data Layer of the Motion app.

6.4.2 Program words of NC functions

The following applies to program words of the NC functions:

- The syntax of NC functions can consist of G- and M-codes as well as of general language elements.
  Examples:
  "G0", "G41", "G141", "G52.0", "M30", "SetSignal(...)
  For all available NC functions with the respective syntax rules, refer to the chapters ➥NC functions with syntax according to DIN 66025 and ➥NC functions with high-level language syntax.
- NC functions can contain additional parameters that can be used to affect the operation of the NC function.
  There are two cases:
  - Address word:
    Parameters programmed as individual program word in the NC block, e.g. "G2 X10 I1.3 J2.5 G94 F1000". These parameters are called address word and are mainly parameters defined in DIN 66025.
  - Local parameters:
    Parameters programmed within a parameter list with specific syntax elements enclosed in parentheses, e.g. "SetSignal(signalId= 10)", "ResetSignal(10)".
- The parameters of NC functions can be optional.
  If these parameters are not programmed, default settings that are hard-coded or saved in the Data Layer of the Motion app are used in general.

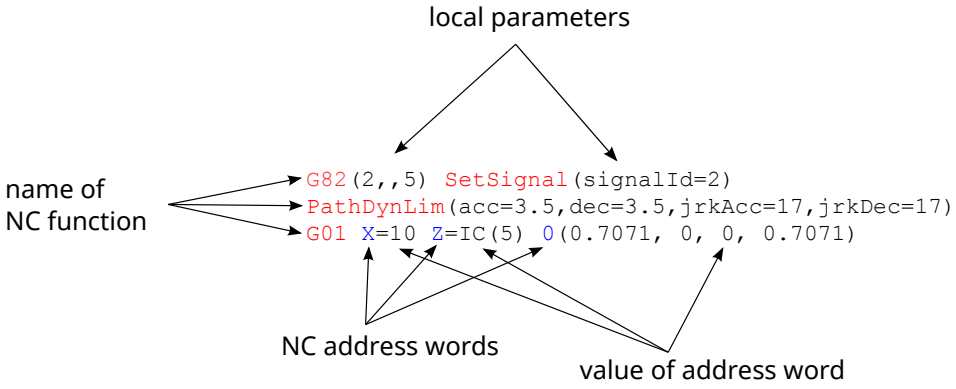

Fig. 4: Program words of NC functions

6.4.3 Program words as parameters (address words)

The following applies to program words as parameters:

- An address always starts with a letter and can consist of multiple characters.
- Program words with an address and numbers are used for programming, e.g.:
  - Axis and coordinate names (e.g. X..., Y..., Z..., B...)
  - Radii (R...) and interpolation parameters (I..., J..., K...)
  - Feed/time values (F...)
  - Spindle speeds and cutting velocities (S..., Si=...)
  - Auxiliary functions (M..., T...)
- Leading zeros do not have to be programmed.
- Non-integer values are written with a decimal point; following zeros can be omitted (e.g. "X100.500" corresponds to "X100.5")
- If a positive sign (or no sign) is programmed, the following value is interpreted as positive value. A negative sign declares a negative value.

Example:

Program word consisting of address letters and a number (in the example: coordinate value of the x-axis)

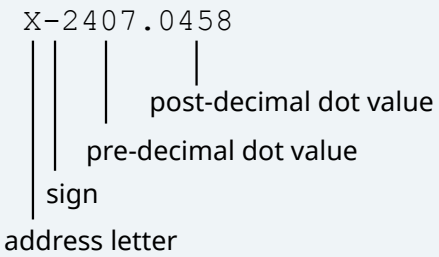

Fig. 5: Program word as parameter

6.4.4 Using separators between two subwords

Each subword can consist of one or more characters (character string).

Each programmable character can be divided into the following groups:

- Letters

"A" - "Z", "a" - "z"

- Numbers (including decimal point):
  "0" - "9", "."
- Characters used as separators:
  " ", "=", "+", "-", "(", ")"
- Further special characters

A separator has to be programmed between two adjacent subwords if the preceding subword ends with a letter or with a number and if the following subword begins with a letter or a number.

For example, this is the case

- if an additional syntax follows an NC function without a value and a parameter list:
  `N10 G54 ; suitable separator:" "`
- if a parameter syntax ends with a number and a numerical value is to be assigned to the parameter:
  `N20 X2=2 ; suitable separator:"="`

6.5 Effect of program words

Program words can act "modally" or "non-modally".

6.5.1 Modal

"Modal" means that a program word is applied to every subsequent program block until:

- The same program word is programmed with a different value
- Another program word is programmed that cancels its effect
- The function of the program word is specifically switched off

Example

N20 G0 X0 Y0	Linear interpolation in rapid traverse to the position X0/Y0 G0 acts modally
N30 Z100	Linear interpolation in rapid traverse to the position Z100
N40 G1 X10 Y10	Linear interpolation with feed F1000 to the position X10/Y10 G1 cancels the effect of G0
N50 X20 F100	Linear interpolation on X20 with F100 F100 acts modally

6.5.2 Non-modal

"Non-modal" means that a program word is only applied to the program block in which it was programmed.

Example

N10 G1 F1000	The active linear interpolation G1 F1000 acts modally
N20 G75.1 X100 Y100	G75.1 Travel to first touch probe. G75.1 acts non-modally
N20 Z100	Linear interpolation at F1000 to Z100 G1 remains modally active

6.6 Special elements for programming

6.6.1 Blank lines in program code

Blank lines can be used to structure the program. They thus increase the readability of the program. The control skips blank lines.

6.6.2 Comments in a parts program

The control skips commands when executing the program.

Use comments to document the program code or insert explanations.

Comments can be used for complete program lines or for single parts.

Comments in the text:

- "//":
  The text after "//" is a comment.
- ";":
  The text after ";" is a comment.

Examples

```
N10 X10 ; This is a comment.
N10 X10 // This is a comment.
```

6.7 Subroutines

Subroutines (SR) are programs that are called using a subroutine call. When the execution of a subroutine is complete, the calling program continues to run at the position at which the subroutine was called.

The main program (MP) is the program from which the execution jumps to the first subroutine level (SR level).

There is no formal distinction between main programs and subroutines.

The following applies:

- Subroutines can contain standard NC blocks and CPL blocks.
- Each subprogram can be called by other programs as subroutine. However, a program is not able to call itself as a subroutine **(a recursive call is not possible**).
- The calling program cannot transfer parameters to a subroutine.
- The maximum nesting depth is 24, i.e. the control can keep up to 24 subroutine levels open at the same time.
- The names of the subroutines are case-sensitive.

The maximum permissible subroutine nesting is shown schematically for example. N1 is the first block of the program. The next subroutine is called in block N9. Reaching the end of file completes the respective subroutine.

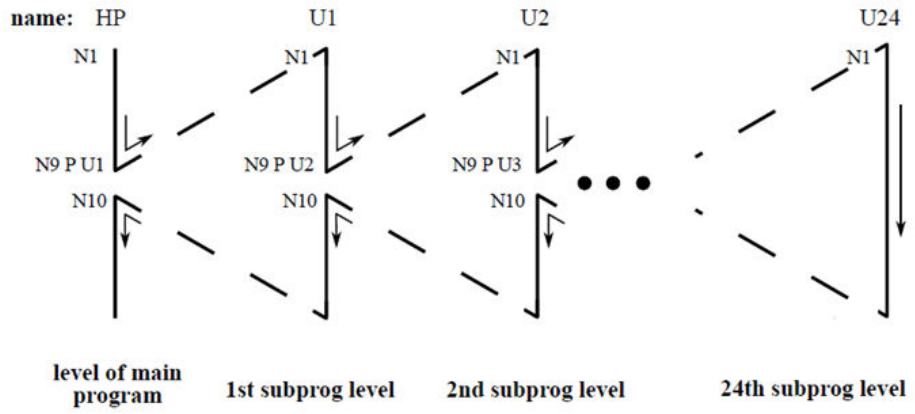

Fig. 6: Nesting subroutine

Effects of subroutines:

- The subroutine call is locally effective (not modally).

6.7.1 Calling subroutine

Syntax

- Call with IP address and subroutine names from a standard NC block (optionally with relative **path specification**).
- Call subroutine without IP address from a standard NC block

Example

```
P Subroutine
Subroutine2
P folder/Sub.npg
folder/sub.npg
```

6.7.2 Search path of the subroutine

The folder containing the main subroutine is scanned with the highest priority.

The search path can also be added under the Data Layer path *g-code/cfg-instances/[Instance] /cfg/subroutine/path*. The variable [Instance] has to match the G-code script instance (as well as the kinematic name).

6.8 Programming high-level expressions in the NC block

6.8.1 Introduction and overview

Both standard NC programming and high-level syntax programming can be used within one part program.

High-level syntax programming is described in chapter ➦ Advanced programming with high-level syntax.

In the following cases, expressions programmed with high-level syntax can also be used in NC blocks:

- High-level expression as value of NC address words
- High-level expression as arguments of an NC function
- High-level expression (with string type) as NC elements

6.8.2 High-level expression as value of NC address words

Syntax:

```
X [expression]
X = [expression]
X = IC (expression)
M[expression]
```

- If a high-level expression is programmed in a standard NC block together with an NC address word, the parts of the expression should be enclosed in square brackets ("[" and "]").
- If the value of the M-code is an expression:
  - The parts of the expression should be enclosed in square brackets ("[" and "]").
  - Spaces are not permitted between the characters "M" and "[".
- If a high-level expression is programmed as value of the NC address word attribute (IC/AC...), the expression can be programmed directly in brackets.
- The value of the expression is determined at runtime when the current block is executed.
- The type of expression has to be a number.

Example

Example	Description
N10 YPos = DL.plc.app.Application.sym.PLC_PRG.TargetPos N20 G1 Y[YPos]	In the NC block N10, "YPos" is read from a variable defined in the PLC variable (via the Data Layer interface) In the NC block N20, the y-axis is to be traversed to the value of the variable "YPos".
N10 X[XPOS*2 + 30] N20 X = [XPOS + 30] F[VEL] N30 X= IC(XPOS)	"XPOS" and "VEL" are variables, the value of the F-word and the axis address word are determined at runtime after the return value of the CPL expression in brackets was determined.
N10 NUM = 20 N20 M[NUM]	The M-code M20 is executed in the NC block N20.

6.8.3 High-level expressions as arguments of an NC function

High-level expressions can be programmed directly as value of arguments of NC functions.

Example

Example	Description
N10 SetSignal(ID = SigID) N20 SetSignal(SigID)	In the NC block N10 and N20, the parameter ID is defined in the NC function SetSignal using the SigID variable. The value type of the SigID variable has to be an INTEGER and the ID range is also checked.
N10 PathDynLim(ACC = 2, DEC = VarDec) N20 PathDynLim(2, VarDec)	The parameters ACC are defined with the constant value 2 in the NC blocks N10 and N20. The DEC parameters are defined using the VarDec variable.

6.8.4 High-level expression (with string type) as NC element

If program CPL expressions are programmed within a standard NC block (e.g. symbolic variables), the CPL parts have to be enclosed in square brackets ("[" and "]").

The value of the CPL expression in brackets has to be interpretable as valid NC element (e.g. address word, NC function, subroutine call).

- Multiple NC elements can be combined in a character string enclosed in square brackets, e.g. "G90G1".
- If an empty character string is specified in brackets, there is no action and no error is reported.

Example

Example	Description
N10 ProgMode = "G91" N20 [ProgMode] X5 Y2	In the NC block N20, the program mode is defined by the value of the ProgMode variable.

N10 STR = "M0"
IF DebugMode == TRUE THEN
N20 [STR]
ENDIF
	In the NC block N20, the expression [STR] is interpreted as M-code M0.

N10 SubroutinName = "MySubroutine" N20 [SubroutinName]	The subprogram is called in the NC block N20.
N10 NcfName = "SetSignal(10)" N20 [NcfName]	The NC function SetSignal(10) is called in the NC block N20.
N10 Path = "G91X2Y2" N20 [Path]	In the NC block N20, the X-axis and Y-axis move to the target position in G91 mode.

7 NC functions with syntax acc. to DIN 66025 (incl. extensions)

7.1 Introduction and overview

This chapter introduces the NC functions with the syntax according to DIN 66025.

The released functions are listed in ➥ Chapter 10.1.1 NC functions with syntax acc. to DIN 66025 on page 42.

7.2 G-codes

7.2.1 Straight line interpolation in rapid traverse "G0"

The NC function G0 approaches a programmed position with interpolation on a straight line in rapid traverse.

At least one axis traverses at a maximum velocity or acceleration. The velocity of the axes is limited by the maximum dynamic limits of the configured kinematics and controlled so that all axes reach the target point at the same time.

The function G0 is modal and deletes the other modal functions, e.g. G1.

Syntax

Function name	G0, G00, G000
Modal group	Geo

Example

```
N10 G0 X0 Y0 Z0 ; Move to initial position
N20 X10 Y10 Z0   ; Move to target position with max. configured dynamic limits
```

Configuration

When using G0, the dynamic limit of the kinematics is configured in the following Data Layer path:

motion/kin//cfg/lim/**

Respective Motion command

`moveABS`

7.2.2 Straight line interpolation in the feed "G1"

The NC function G1 approaches a programmed position by interpolating on a straight line with effective feed.

The coordination of the motion ensures that all axes involved reach the programmed endpoint at the same time. The programmed feed value (F) acts as path feed. When traversing multiple axes, the feed of each axis should be less than F. An error is reported if `F<number>` is not programmed explicitly.

The function G1 is modal and deletes the other modal functions e.g.: G0.

Properties

The programmed feed is applied until overwritten by a new feed value.

The programmed path velocity can be limited by the Data Layer.

The dynamic limit (acceleration/deceleration/jerk) can be changed by the PathDynLim function.

Syntax

Program name	G1, G01, G001
Modal group	Geo

Example

```
N10 G0 X0 Y0 Z0          ; Move to initial position
N20 G1 X10 Y10 Z10 F10   ; Move to target linearly with a feed of F10
```

Configuration

When PathDynLim is switched off, the Motion uses the default dynamic limit of the kinematics which can be configured in the following Data Layer path:

motion/kin//cfg/lim/**

Respective Motion command

`moveABS`

7.2.3 Circular interpolation in clockwise/anticlockwise rotation "G2"/"G3"

The NC function G2/G3 moves to a programmed position in the active machining plane interpolating on a circular path in clockwise/anticlockwise rotation at the current feed (programmable via the F-address). Additional axes beyond the working plane can be included into the programming with linearly interpolated motions. The axes traversing on the arc are defined by the selected working plane (G17, G18, G19, G20). The programmed feed value (F) acts as path feed. When traversing multiple axes, the feed of each axis should be less than F. An error is reported if no value F<*Number*> is explicitly programmed.

The function is modal and deletes the functions G0, G2, G3, G5, G6, G12, G13, G33.

Syntax

Function name	G2, G02, G002
	G3, G03, G003
Modal group	Geo

The circular path can be programmed using radius programming or center point programming.

- Radius programming:

Meaning	Value
<EP>	Coordinates of the endpoint
F	Feed value
R	• Radius of the circular path • The radius has to be at least half the distance between the starting point and the endpoint. • A positive radius value means that it is traversed along the smaller circular arc (Arc <= 180 degrees). A negative radius value means that it is traversed along the larger circular arc (Arc > 180 degrees). • *Radius = distance (end point, start point)/2* results in a semicircle, whereby any sign (positive/negative) can be used.

- Center point programming

Meaning		Value
<EP>		Coordinates of the endpoint If the starting and endpoints are identical within the circular plane, a full circle is generated automatically.
F		Feed value
<IP>	I(X-coordinate)	Interpolation parameters
	J(Y-coordinate)	
	K(Z-coordinate)	

Meaning		Value
		• I, J and K define the distance between the center point and the starting point along the respective axis • An inaccurate center point can be automatically corrected as per the tolerance set. • If the interpolation parameters are not suitable for the selected position, errors are generated by the motion.

Example

```
N10 G0 X0 Y0 Z0       ; Move to initial position
N20 G17               ; Activate the XY plane
N30 G2 X20 Y0 Z10 F10 ; Clockwise circular interpolation with radius parameter in XY
plane, linear interpolation along Z axis
N50 G3 X40 Y0 I10 J0  ; Counterclockwise circular interpolation with center point parameter
```

Configuration

When PathDynLim is switched off, the Motion uses the default dynamic limit of the kinematics which can be configured in the following Data Layer path:

motion/kin//cfg/lim/**

Respective Motion command

moveCircle2D

7.2.4 Path slope "G8", "G9"

The NC function G8 enables and G9 disables the path slope. Without path slope, it is accelerated to the programmed velocity at the beginning of an NC block and decelerated to v=0 at the end of the NC block. Although this reduces the contour deviation at block transition, it requires a longer machining time. If the path slope is active, the control intends to create a velocity that is as constant as possible and that is as high as the programmed feed. This reduces the machining time and smoothens the contour at the corners.

Syntax

Function name	G8, G9
Modal group	Path slope

Example

```
N10 G8                ; Activate the path slope
CONTOUR(3)
N20 G90 X0 Y0 Z0 F400  ; The path slope will take effect on the following motions
N30 X100 Y0 Z0
N40 X100 Y100 Z0
CONTOUR()
N30 G9                ; Deactivate the path slope
```

Respective Motion command

`KinContMotion`

7.2.5 Plane switching "G16"/"G17-19"

The NC function G16 disables the current active plane. Functions G17-G19 enable the selected planes. The active plane is defined by a main and a secondary coordinate, an infeed axis is perpendicular to the plane. After selecting the active plane, the interpolation parameters I, J, K are evaluated for the circular motion.

	Main coordinate	Secondary coordinate	Infeed coordinate	Active plane
G17	X	Y	Z	XY
G18	Z	X	Y	ZX
G19	Y	Z	X	YZ

The functions G16, G17, G18, G19 and G20 form a modal group and therefore cancel each other.

Syntax

Function name	G16, G016
	G17, G017
	G18, G018
	G19, G019
Modal group	Vor_Geo

Example

```
N10 G17             ; Activate the XY plane
N20 G2 X38 Y20 R15  ; Clockwise circular interpolation with radius parameter in XY plane
N30 G16             ; Deactivate XY plane
```

Respective Motion command

kinActPlane

7.2.6 Tool length correction "G47", "G48"

The NC function G47/G48 enables/disables the tool length correction.

- The brackets and the content of G47 are optional.
- If G47 is programmed without brackets and parameters, the function calls a PCS set group with the name "G47".
  Before the tool length correction is enabled, PCS sets with the name "G47" should be available.

G47 and G48 act modally and deselect each other.

Syntax

Function name	G47{(SET=<setName>)}		
	G48		
Modal group	Tool length correction		
Enabling	G47{(SET=<setName>)}		
	G47		
Disabling	G48		
Parameters	Keyword	Position index	Meaning
	SET	1	Name of the selected PCS set

Example

```
N10 G1 G47               ; Activate PCS tool “G47”
N20 G90 X10 Y10 Z10 F100 ; Move to X10 Y10 Z10 with tool length correction on
N30 G48                  ; Deactivate Tool Length Correction
N30 G47(SET=”set1”)           ; Activate PCS tool set "set1"
```

Configuration

The tool length correction values of the kinematics are configured under the following Data Layer path:

*motion/cfg/coord-systems/pcs/[set]/**

Add "PCS sets group" with the name "G47" manually with the payload of PCS sets (group) names.

Respective Motion command

`KinPCSTool`

7.2.7 Selecting zero offset table "G53", "G54-G59"

The NC functions G54 to G59 select and enable the zero offset tables (e.g. Zero Offset G54) to move the machine coordinate system in space. The NC function G53 disables all zero offset tables. Enabling the zero offset is based on the Motion command option *opt-pcs* (command options for kinematic product coordinate system). The ZO table should be available and stored under "PCS sets" before it is enabled.

G53 to G59 act modally and deselect each other.

Syntax

Function name	G53-G59
Modal group	Select zero offset table

Example

```
N10 G1 G54                ; Activate the PCS with ZO table of G54
N20 G90 X10 Y10 Z10 F100  ; Move to X10 Y10 Z10 based on PCS
N30 G53                   ; All ZO table off
```

Configuration

The offset distances for the machine coordinates in the kinematics can be configured in the following Data Layer path:

*motion/cfg/coord-systems/pcs/[set]/**

Add "PCS sets group" with the name "G54".... "G59" manually with the payload of PCS set names.

Respective Motion command

`KinPCSP`

7.2.8 Absolut dimension programming "G90", Relative dimension programming "G91"

The NC functions G90 and G91 specify whether the control should interpret the dimensions for the axes and coordinates as absolute or relative (incremental) values. The absolute motion G90 refers to the current zero point in the program coordinate system (PCS).

G90 and G91 act modally and deselect each other.

Syntax

Function name	G90/G91
Modal group	Programming mode

Example

```
N10 G1 G90 F100     ; Absolute dimension programming ON
N20 X100 Y100       ; Move to coordinates X100, Y100
N30 G91 X100 Y100   ; Relative dimension programming ON, move to X200, Y200
```

Respective Motion command

`moveABS/moveREL`

7.2.9 Feed programming "G94"

The NC function G94 interprets the F-words as feed for a programmed contour. The F-word specifies the velocity within the dynamic limits of a given Motion command.

G94 is the "feed mode" active by default.

Syntax

Function name	G94
Modal group	Feed mode

Example

```
N10 G94 G90 F10     ; Set Feed mode with feed velocity 10
N20 X100 Y100       ; Move to coordinates X100, Y100
```

Configuration

The unit of the velocity of G94, which is set to "mm/min" by default, can be configured in the following Data Layer path:

motion/kin//cfg/units/velocity*

Respective Motion command

`moveABS`

7.2.10 Product coordinate system "G153", "G154"

The NC function G153/G154 enables/disables the selected PCS sets.

PCS sets can be used to move the machine coordinate system in space.

- Enabling from PCS offsets is based on the Motion command option *opt-pcs* (command options for kinematic product coordinate system).
- The offset sets for the machine coordinates in the kinematics are saved in the "motion/cfg/coord-systems/pcs/[set]/*" directory.
- Add the PCS set group with the entered set name manually with the payload of the PCS set (group) names.
- Before enabling the PCS set, the expected offset distance should be editable.
- When G154 is programmed with a specific PCS set, the G-Code Interpreter generates a permanent kinematic Motion command option:
  opt-pcs("set")
  If the selected PCS set does not exist, an error is generated by motion.core.
- The new G154 command with the name of the selected PCS set deletes the previous active PCS set.

Syntax

Function name	G153 G154
Enabling	G154(SET = <setName>) G154
Disabling	G153

Parameters	Keyword	Position index	Meaning
	SET	1	Name of the selected PCS set

Example

```
G154("SetA")                ; Aktivieren des PCS mit der Offset-Tabelle "SetA"
N10 G1 G90 X10 Y10 Z10 F100 ; Anfahren der Zielposition im ausgewählten PCS
G153                        ; Deaktivieren aller PCS-Tabellen
G154("SetB")                    ; Aktivieren des PCS mit der Offset-Tabelle "SetB"
```

Respective Motion command

motion/kin/[name]/cmd/opt-pcs

8 NC functions with high-level language syntax

8.1 Introduction and overview

This chapter introduces the NC functions with high-level language syntax and some special Motion commands.

The released functions are listed in ➦ Chapter 10.1.2 NC functions with high-level language syntax on page 43.

8.2 Global signals

8.2.1 "SetSignal"

The "SetSignal" function triggers a global signal at runtime. Thus, the level of the programmed events is set to 1 (irrespective of whether the value was 0 or 1). Multiple "SetSignal" functions can be programmed in one block. The signal is triggered by the programmed sequence.

Syntax

Function name	SetSignal SETSIGNAL
Short form	SSG
Parameters	ID = <number> * The index of the signal ID is triggered ** The value range for signal ID is between 0 - 99 for the entire system

Example

```
N10 SetSignal(1)                              ; Signal 1 is triggered
N20 SSG(2)                                    ; Signal 2 is triggered
N30 SetSignal(ID =3)                          ; Signal 3 is triggered
N40 SetSignal(4) Delete SetSig(5) SSG(6)  ; Signals 4,5,6 are triggered
```

Respective Motion command

motion/kin//cmd/set-signal*

8.2.2 "ResetSignal"

The function "ResetSignal" resets a signal at runtime. Multiple "ResetSignal" functions can be programmed in one block. The signal is triggered by the programmed sequence.

Syntax

Function name	ResetSignal
	RESETSIGNAL
Short form	RSG
Parameters	`ID = <number>`
	* The index of the signal ID is reset
	** The value range for signal ID is between 0 - 99 for the entire system

Example

```
N10 ResetSignal(1)     ; Reset signal 1
N20 RSG(10)            ; Reset signal 10
```

Respective Motion command

motion/kin//cmd/reset-signal*

8.2.3 "WaitForSignal"

Use the "WaitForSignal" function to create a Motion command sequence that waits at runtime until a signal is triggered.

Multiple "WaitForSignal" functions can be programmed in one block. The output point in time for the signal is specified by the programmed sequence.

Syntax

Program name	WaitForSignal
	WAITFORSIGNAL
Short form	WSG
Parameters	`ID = <number>`
	* Waiting for a signal
	** The value range for signal ID is between 0 - 99 for the entire system

Example

```
N10 WaitForSignal(1)   ; Waiting for signal 1
N20 WSG(10)            ; Waiting for signal 10
```

Respective Motion command

motion/kin//cmd/wait-for-signal*

8.3 M-code

M-codes are used to synchronize signals between the G-code interpreter and the PLC application.

Features

- A total of 100 M-codes are supported by G-code for all kinematics (M0-M99)
- M0-M99 are subject to acknowledgement
- All M-codes are available in one NC program
- All M-codes can be programmed in one NC block
- Only global M-codes are supported (no kinematics-specific ones)

Example

The behavior of the M-codes is identical to that of the MTX.

Case	NC block	Description
Single M-code in one block	M20	M0-M99 are subject to acknowledgement.
Same M-codes in one block	M17 M17 M38 M17	Identical M-codes are automatically merged and only executed once.
M-codes in different blocks	N100 M20 N110 M21	
M-code in one block	N100 M20 M21	Signals that correspond to the M-codes are triggered by the programming sequence, the NC block is completed until all signals are reset.
M-codes and position commands in the same block	N10 M210 M169 X10 M184	All signals are triggered before the motion, the NC block is completed until the target position is reached and all signals are reset.

Reading/writing M-codes in the PLC project

Use of the following function blocks in the CXA_Motion library:

- ML_GetSignal: Determines the signal status
- ML_SetSignal: Sets a signal (to TRUE)
- ML_ResetSignal: Resets a signal (to FALSE)

Use of the following function blocks in the CXA_PLCOpen library:

- MB_GetSignal: Determines the signal status
- MB_SetSignal: Sets a signal (to TRUE)
- MB_ResetSignal: Resets a signal (to FALSE)

Data Layer path to monitor signals:

- *motion/state/functions/somo/signals*

Respective Motion commands

- *motion/kin/*/cmd/set-signal*
- *motion/kin/*/cmd/wait-for-signal*

8.4 "WAIT"

Use the "WAIT" function to stop preparing previous motion commands and to stop executing the G-code script until all previous motion commands have been executed.

Syntax

Function name	WAIT
Parameters	Without parameters

Example

```
N10 WAIT          ; Stop executing this script.
```

Respective Motion command

motion/kin//cmd/pre-wait*

8.5 "PolyTrans"

Use this command function to enable blending permanently, inserting a rounding via spline function between successive motion commands until blending is explicitly disabled again.

Syntax

Program name	PolyTrans POLYTRANS
Abbreviation	PTR
Enabling	`PolyTrans(D1=<number>, D2 =<number>)` `PolyTrans(EPS=<number>)`
Disabling	`PolyTrans()`
Parameters	* The parameters `D1` and `D2` cannot be used simultaneously with the parameter `EPS`. ** An empty parameter means switching off.

Beispiel

```
N10 PolyTrans(D1=1, D2=1)  ; Switch on polytrans permanently, and define
                           ; the corner with parameter D1 and D2.
N20 X10 Y10 Z10
N30 X30 Y10 Z10
…
N100 PolyTrans()           ; Switch off polytrans permanently.
```

Respective Motion command

motion/kin//cmd/pt-poly-trans*

8.6 "PathDynLim"

This command function reduces the upper limits for the geometry NC function G1, G2, G3, G5, G6, G12, G13, G33 in the part program.

- Path acceleration
- Path deceleration
- Path-jerk acceleration
- Path-jerk deceleration

If PythDynLim is not enabled, the kinematics moves using the configured dynamic limit values in the path *motion/kin/*/cfg/lim*.

If PathDynLim is enabled, the kinematics moves using the programmed dynamic limit values.

If PathDynLim is switched off, the kinematics moves again with the configured dynamic limit values.

Syntax

Program name	PathDynLim PATHDYNLIM
Abbreviation	PDL

Enabling	`PathDynLim(ACC=<value>, DEC=<value>, JRKACC=<value>, JRKDEC=<value>)`
	`PathDynLim(ACC=<value>, DEC=<value>)`
	`PathDynLim(<value>, <value>, <value>, <value>)`
	`PathDynLim( , , , <value>)`

Disabling	`PathDynLim()`

Parameters

Keyword	Position index	Meaning
ACC	1	Double, new path acceleration. Has to be > 0. Optional. If not programmed, the path acceleration is not changed.
DEC	2	Double, new path deceleration. Has to be > 0. Optional. If not programmed, the path deceleration is not changed.
JRKACC	3	Double, new path-jerk acceleration. Has to be > 0. Optional. If not programmed, the jerk acceleration is not changed.
JRKDEC	4	Double, new path-jerk deceleration. Has to be > 0. Optional. If not programmed, the jerk acceleration is not changed.

Example

```
N10 G1 X10 Y10 Z10          ; Move to target position with defalut dynamic limits
N20 PathDynLim(acc=1,dec=2) ; Move to target position with defalut dynamic limits
N30 G1 X30 Y10 Z10          ; Move to target linear in with new dynamic limits
N20 PathDynLim()            ; Reset dynamic limits to default values
```

Respective Motion command

Configuration:

The dynamic limit values for the default path are configured in the following Data Layer path:

- *motion/kin/*/cfg/lim/acc*
- *motion/kin/*/cfg/lim/dec*
- *motion/kin/*/cfg/lim/jrkAcc*
- *motion/kin/*/cfg/lim/jrkDec*

The units of the dynamic limit values can be read using the following Data Layer path:

- *motion/kin/*/cfg/lim*

The units of the dynamic limit values can be changed using the following Data Layer path:

- Jerk: *motion/kin/*/cfg/units/jerk*
- Acceleration: *motion/kin/*/cfg/units/acceleration*

8.7 "AxsDynLim"

The function permanently reduces the maximum dynamic limits of a kinematic axis for all subsequent motion commands.

Syntax

Program name	AxsDynLim AXSDYNLIM
Abbreviation	ADL

Enabling	`AxsDynLim(AXS=<name>{ {, ACC=<value>} {, DEC=<value>} {, JRKACC=<value>} {, JRKDEC=<value>} })`
	
	`AxsDynLim(<name>, <value>, <value>, <value>, <value>, <value>)`

Disabling	`AxsDynLim(<name>)`
	`AxsDynLim(<name>,  ,  ,  ,  ,)`
	`AxsDynLim(<name>,<0>,<0>,<0>,<0>,<0>)`
	`AxsDynLim(AXS=<name>,<0>,<0>,<0>,<0>,<0>)`

Parameters

Keyword	Position index	Meaning
AXS	1	String, axis name
VEL	2	Double, new velocity limit value. Optional. The last programmed acceleration limit value is used if this parameter is not programmed.
ACC	3	Double, new acceleration limit value. Optional. The last programmed deceleration limit value is used if this parameter is not programmed.
DEC	4	Double, new deceleration limit value. Optional. The last programmed acceleration limit value is used if this parameter is not programmed.
JRKACC	5	Double, new jerk acceleration limit value. Optional. The last programmed acceleration limit value is used if this parameter is not programmed.
JRKDEC	6	Double, new jerk deceleration limit value. Optional. The last programmed acceleration limit value is used if this parameter is not programmed.

Example

```
N10 G1 G90 X10 Y10 Z10 F100 ; Move to target position with default dynamic limits
N30 AxsDynLim(AXS="X", VEL=10, ACC=1,DEC=1,JRKACC=5,JRKDEC=5) ; Further reduce the dynamic
limits of "X" axis
N40 X20 Y20 Z20             ; Move to target linear position with new axis dynamic limits
N50 AxsDynLim(AXS="X")      ; Reset the dynamic limits of "X" back to the default
configuration
```

Respective Motion command

Configuration:

The dynamic limit values of the axes can be read with the following Data Layer path:

- *motion/axs/*/cfg/lim*

The units of the dynamic limit values can be changed using the following Data Layer path:

- Jerk: *motion/axs/*/cfg/units/jerk*
- Acceleration: *motion/axs/*/cfg/units/acceleration*

The following Data Layer path has the same effect as the AxsDynLim function:

- *motion/kin/*/cmd/opt-axs-dyn-lim*

8.8 "Blend", "BlendLocal"

Blending means that two or more commands can be executed simultaneously. The superimposition of commands is also referred to as “BlockSlope”.

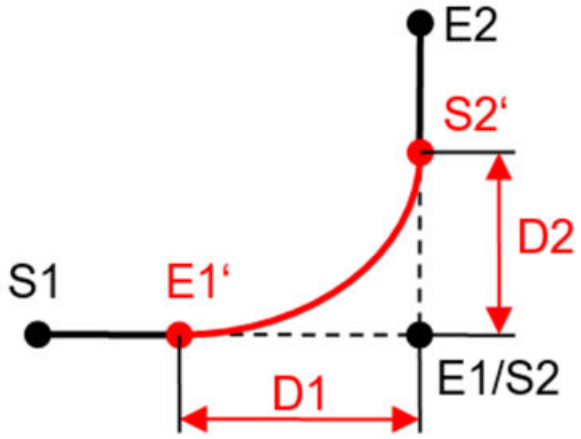

Fig. 7: Blending between commands

8.8.1 Blending between "Blend" commands

Description

The modal NC function "Blend" follows the Motion commands permanently.

Syntax

Program name	Blend	
Abbreviation	BLEND	
	BLD	
Parameters	`Blend(D1=<number>, D2 = <number>)`	
Enabling	`Blend(D1=<number>, D2 = <number>)`	Switch on "Blend" permanently, the corner is defined with the parameters D1 and D2
	`Blend(<number>, <number>)`	Switch on "Blend" permanently, the corner is defined with the parameters D1 and D2 (position parameters)
Disabling	`Blend()`	Switch off "Blend" permanently

Parameters

Keyword parameter:

- `Blend(D1 = <number>, D2 = <number>)`

Parameter group:

Keyword	Meaning
D1	double; Distance to the end position of the first Motion command (corner E1/S2) until blending can start.
D2	double; Distance after the starting position of the second Motion command (corner E1/S2) in which the Motion has to be retraced to the path (end of blending).

`Blend()` means that blending is permanently switched off.

8.8.2 Local (non-modal) blending "BlendLocal"

Description

The non-modal NC function "BlendLocal" follows the Motion commands (once) locally.

Syntax

Program name	BlendLocal	
Abbreviation	BLENDLOCAL	
	BLENDL	
	BLDL	
Parameters	`BlendLocal(D1=<number>, D2 = <number>)`	
Enabling	`BlendLocal(D1=<number>, D2 = <number>)`	Switch on "Blend" locally, the corner is defined with the parameters D1 and D2
	`BlendLocal(<number>, <number>)`	Switch on "Blend" locally, the corner is defined with the parameters D1 and D2 (position parameters)

Parameters

Keyword parameter:

- `BlendLocal(D1 = <number>, D2 = <number>)`

Parameter group:

Keyword	Meaning
D1	double; Distance to the end position of the first Motion command (corner E1/S2) until blending can start.
D2	double; Distance after the starting position of the second Motion command (corner E1/S2) in which the Motion has to be retraced to the path (end of blending).

`Blend()` means that blending is permanently switched off.

8.8.3 Example of a sequence

NC block	Remarks	Blending		Description
N10 Blend(D1=10, D2=10)	Blend_1 (permanently switched on)			Blending is permanently switched on Defining the corner with the parameters D1, D2

NC block	Remarks	Blending		Description
N20 X10 Y10 Z10		Blend_1		
N30 Blend-Local(5, 5)	Blend_2 (once)			Blending is switched on locally
N40 X30			Blend_ 2	
N50 Y20		Blend_1		
N60 X60				
N70 Blend()				Blending is permanently switched off
N80 combina-tion(2,2)	Blend_3 (permanently switched on)			
N90 Z10		Blend_3		
N100 combina-tion(4,4)	Blend_4 (permanently switched on)			
N110 X10			Blend_ 4	
N120 X30				
N140 Blend()				

Respective Motion command

`KinBlend, KinBlendP`

9 Advanced programming with high-level syntax

9.1 Introduction and overview

The use of high-level programming syntax increases the flexibility of a program and results in a better programming structure.

The advanced programming includes:

- Variables and computations
- Flow control: Decision and loops
- Jump instructions

9.2 Type and values

ctrlX G-code is a dynamically typed language.

Dynamically typed variables means:

- There are no type definitions in the language; as they are defined at runtime.

Basic types in ctrlX G-code:

Type	Description	Example
NIL	Represents the absence of a useful value	NIL
BOOLEAN	Has two values: TRUE or FALSE	TRUE
INT	64-bit integer (positive or negative)	123
FLOAT	Double-precision (64-bit) floating point numbers	-123.456

Type	Description	Example
STRING	Represents invariable byte sequences	"Hello, ctrlX"

9.3 Variables (global)

Variables can store values.

Naming of variables (regex expression: [a-zA-Z_][a-zA-Z_0-9]+)

- Variable name can contain letters and numbers as well as "-"
- Variable name must not start with a number
- The variable name must not contain any alternation between numbers and letters, e.g. name 'H3V3' is invalid
- System-reserved words cannot be programmed as variables (e.g. IF/WHILE/STEP, NC function names...)
- A variable name consisting of the letter "N" followed only by numbers is not a valid variable name. This character string pattern N<Number> always stands for a line number.

Only global variables are available in the current version.

- Life cycle of global variables:
  - Definition of global variables:
    Global variables do not have to be declared explicitly with a keyword.
    Global variables are created the first time they are used.
  - Destruction of global variables:
    After a script file has been ended or paused, all global variables are destroyed.
    After the script has been executed, the global variables are not destroyed and retain their value.
- Scope of global variables:
  - When a global variable contains a value, it can be called in all script sections (main program and subroutine).

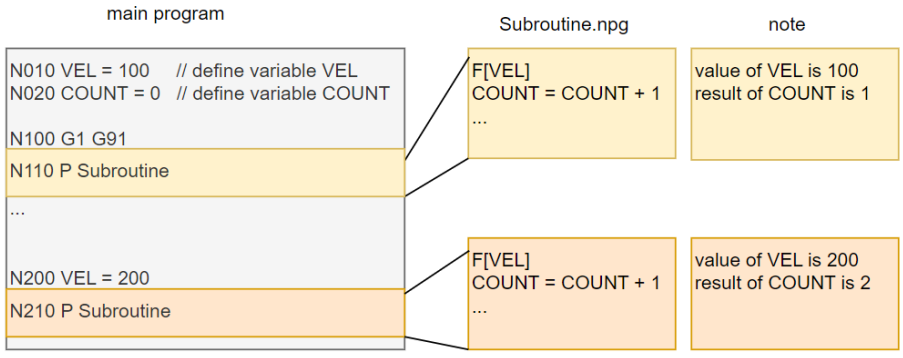

Fig. 8: Example of global variables

Data Layer nodes can be used to store the variables with a "permanent" life cycle. The PLC or the Key Value Database app can be useful when creating own Data Layer nodes.

9.4 Value assignment

Global variables can be linked to values.

The "=" sign (equal sign) is used.

Example

Var_1 = 100	Assignment of an Int value to the global variable "Var_1"
Var_1 = 2029.876	Assignment of an Float value to the global variable "Var_1"
str = "Hello, ctrlX"	Assignment of a string value to the global variable "str"
N100 Var_2 = Var_1 + 3	Use the value of the global variable "Var_1" and assign the expression to the global variable "Var_2"

9.5 Label programming and jump instructions

9.5.1 Introduction and overview

Jump commands can be used to jump to a label at a specific position.

ctrlX G-Code supports the "Jump to label" function, which is explicitly programmed in the current version.

Jumping to a specific line number (Nxx) is currently not possible.

9.5.2 Labels in scripts

A label is a jump label for a jump instruction.

Definition of a label:

Name: <LF>

Programming labels in G-code:

- The label always has to be programmed directly at the beginning of the block.
- The label can be programmed after a line number (Nxx).
- A label has to be programmed in a separate line.
- The name of the label can consist of characters, letters, underscores and numbers. However, it must not begin with numbers.
- A colon must be programmed after the label name at the target of the jump.
- A label can only be defined once in a program area.

9.5.3 Jump command GOTO

Syntax:

```
GOTO [Name]
```

- The GOTO command jumps to a defined label in the current program area or in the outer area.
- The GOTO jump command cannot be used to jump to a control structure (e.g. main part of an IF/WHILE instruction).

Example

```
a = 0
b=10
IF b > 0 THEN
  GOTO label1
ENDIF
label1:
G90
```

Error case:

```
a = 0
b=10
GOTO label1
IF b > 0 THEN
  label1:
ENDIF
G90
```

Error description:

"label1" is defined within the IF instruction. Jumping "label1" from beyond the definition area is not possible.

9.6 Decision and branching instructions

9.6.1 Introduction and overview

Decision and branching instructions are used to execute individual program blocks and program sections or complete subroutines depending on certain events.

ctrlX G-Code provides the following options:

- IF instruction IF-THEN-ELSE-ENDIF

9.6.2 IF instruction

This function is a simple conditional branching instruction. IF a certain condition is met, THEN a routine is executed, ELSE the other routine is executed.

Syntax:

```
IF <expression> THEN <LF>
  block
{ELSE <LF>
block}
ENDIF <LF>
```

- The condition for the IF command has to be an expression.
- It always has to be terminated with the ENDIF keyword.
- ELSE block is optional.
- THEN block and ELSE block should contain CPL or NC lines, an empty record causes a syntax error
- The reserved word IF has to start with a new line or after a line number Nxx.
- The reserved word THEN/ELSE/ENDIF has to end with <LF>.

Example

```
a = DL.plc.foo.bar
IF a > 1 THEN
  F2000
ELSE
  vel = 1000 ; vel is a global variable
  F100
ENDIF
X10
```

9.7 Repeat instructions

9.7.1 Introduction and overview

If one or more program blocks are to be processed repeatedly depending on certain conditions, program it using repeat instructions.

- WHILE-DO-ENDWHILE (short form: ENDW)
- FOR-STEP-TO-ENDFOR

9.7.2 WHILE loop

Syntax:

```
WHILE <expression> DO <LF>
 block
ENDW <LF>

WHILE <expression> DO <LF>
 block
ENDWHILE <LF>
```

Example

```
a = 0
WHILE a < 5 DO
  X10
  a = a + 1
ENDW
```

The GOTO command cannot be used to jump to a WHILE-DO-END command.

9.7.3 FOR loop

If the abort condition for the repeat instruction is to be a direct consequence of the routine processing, a tracking counter would be required, for example.

This counter requires no specific programming for the FOR loop. A counting variable (integer or floating point number) is determined whose starting and end counter has to be specified. If the counting step width deviates from 1, the step width (STEP) can be specified separately.

Syntax:

```
FOR <CountingVar>=<InitialValue> [STEP <StepWidth>] TO
<EndValue>
  block
ENDFOR
```

- Counting variable does not have to be declared
- Default step width is 1 if the STEP keyword is not programmed
- Initial value can be variable
- Step width can be variable
- End value can be variable

Example

```
N10 FOR INDEX = 1 STEP 2 TO 10
N20  Subroutine_draw
N30 ENDFOR
```

After the end of the loop, the counting variable is provided with a value greater than the end value (max. step width).

In this example, the value change of the counting variable INDEX is: 1, 3, 5, 7, 9, 11 and the loop is exited after a value greater than 10. FOR loops with a variable step width can also be programmed.

9.7.4 BREAK instruction

The BREAK instruction interrupts the execution of a WHILE loop or a FOR loop and jumps to the next instruction after the loop.

Syntax:

```
BREAK<LF>
BREAK from FOR loop
FOR <CountingVar>=<InitialValue> [STEP <Step Width>] TO
<EndValue>
  IF <expression> THEN
     BREAK
   ENDIF
ENDFOR
```

- BREAK ends the inner enclosing loop.
- BREAK cannot be programmed outside the WHILE/FOR loop.

Example

```
WHILE TRUE DO
  IF COUNT > 10  THEN
     BREAK
  ENDIF
ENDW
```

9.8 Operators

9.8.1 Mathematical operators

Arithmetic functions that affect variables, constants or higher-level expressions can still be called.

As a rule, "multiplication and division take precedence over addition and subtraction", i.e. multiplication and division are carried out first, followed by addition and subtraction.

For further information, refer to ➥ Chapter 9.8.4 Priority on page 40.

G-code supports the following arithmetic operators:

- **+**: Addition
- **-**: Subtraction

- *****: Multiplication
- **/**: Float division
- **%**: Modulo
- **-**: single minus

The result of the arithmetic operators works as follows:

- If both operands are integers, the operation is performed on integers and the result is an integer (in addition to the division of floating point numbers).
- If one of the operands is a floating point number, the operation is carried out according to the rules of floating point arithmetic and the result is a floating point number.

Example

Expression	Result	Result data type
1+1-3	-1	INT
1.0 + 1	2.0	FLOAT
5/2	2.5	FLOAT
5/1	5.0	FLOAT

9.8.2 Relative operators

G-code supports the following relative operators:

- **==**: Equality
- **!=**: Inequality
- **<**: smaller than
- **>**: greater than
- **<=**: smaller than or equal to
- **>=**: greater than or equal to

These operators always result in **TRUE** or **FALSE**.

Example

Expression	Result	Result data type
1<2	TRUE	BOOLEAN
1 > =2	FALSE	BOOLEAN
a != (b-1)		BOOLEAN

9.8.3 Logical operators

The logical operators in G-code are:

- AND
- OR
- NOT

All logical operators consider **NIL** and **FALSE** as well as **0** as false and everything else as true.

Result of logical operators:

- The **NOT** negation operator always returns **FASLE** or **TRUE**.
- The **AND** conjunction operator returns its first argument if the value is logically false (FALSE, NIL, 0). Otherwise, **AND** returns its second argument.
- The **OR** disjunction operator returns its first argument if this value is logically true (in contrast to FALSE, NIL, 0). Otherwise, **OR** returns its second argument.

Operator	Logical expression	Description	Example
AND	a AND b	If "a" is false, "a" or "b" is returned	(10 AND 20) results in 20 (FALSE AND 10) results in FALSE
OR	a OR b	If "a" is true, "a" or "b" is returned	(10 OR 10) results in 10 (FALSE OR 20) results in 20
NOT	NOT a	If "a" is true, FALSE is returned If "a" is false, TRUE is returned	NOT (10 AND 20) results in FALSE

Example

Expression	Result	Note
10 AND 20	10	
NIL AND 10	NIL	
FALSE AND error()	FALSE	error() is the call of an error function
FALSE AND NIL	FALSE	
FALSE OR NIL	NIL	
10 OR 20	10	
10 OR error()	10	error() is the call of an error function
NIL OR "str"	"str"	

9.8.4 Priority

The priority of the operators in the ctrlX G-code follows the table from higher to lower priority:

	Same priority					
high	NOT	-(unary)				
↓	*	/	%			
↓	+	-				
↓	<	>	<=	>=	!=	==
low	AND					
	OR					

Using brackets in an expression, the order of operations can be specified explicitly.

Example

Expression	Priority with brackets
b+3 < 3+1*2	(b+3) < (3+1*2)
2 AND 3 AND 3+1*2 OR b AND d/2	((2 AND 3) AND 3+1*2) OR (b AND (d/2))
1 < 2 AND 3<=4	(1 < 2) AND (3<=4)
2 == 2 AND 3>=4 OR 5*2 -1	((2 == 2) AND (3>=4)) OR (5*2 -1)
NOT 2 AND NIL OR 4 AND 4/1 AND 3	((NOT 2) AND NIL) OR (4 AND 4/1 AND 3)

9.8.5 Accessing Data Layer nodes

ctrlX G-code can read or write Data Layer nodes to interact with other systems (e.g. PLC variables).

A Data Layer node is represented by a character string (e.g. *plc/app/Application/sym/Machine/Status*).

In the ctrlx G-code script, the node can be displayed as "DL.plc.app.Application.sym.Machine.Status".

Syntax of the Data Layer node

- Prefixed by the keyword "DL."
- Followed by one or more characters beginning with "." and containing a combination of lowercase letters, uppercase letters, numbers and underscores.
- Character "/" in the Data Layer path has to be replaced by "."

Example

- Reading the node value and assigning it to a variable:
  - varFoo = DL.plc.app.Application.sym.Machine.Status
- Writing the node value:
  - DL.plc.app.Application.sym.Machine.Status = 10 + 1

Converting the data type between G-code and Data Layer system

The Data Layer node has more data types than G-code. The conversion of the data type (with range check) is automatically completed when the Data Layer node is read or written.

The following table shows the assignment between Data Layer and G-code:

Data type Data Layer	Data type Data Layer (detailed)	Value in G-code	Example of G-code values
Simple data types	BOOL8	BOOLEAN	TRUE FALSE
	INT8 UINT8 INT16 UINT16 INT32 UINT32 INT64 UINT64	INT: 64-bit integer	123
	FLOAT32 FLOAT64	FLOAT: Double-precision (64-bit) floating point numbers	123.456
	STRING	STRING	"rexroth"
Array types	-	-	-
Complex data types (FlatBuffer)	-	-	-

"-" means unsupported conversion. An error is reported.

Restrictions

- ctrlX G-code only accepts Data Layer paths encoded in ASCII, i.e. if a character string in UTF-8 format is used in a Data Layer node, a syntax error is reported.
- If the axis name is written in German or Chinese characters for example, it is not supported.
- Data Layer nodes with array and FlatBuffer types can neither be read nor written.

10 Appendix

10.1 Overview on NC functions

10.1.1 NC functions with syntax acc. to DIN 66025

NC function name			Group	Description
Name	Short form	Alias		
➦ G0	-	G00 G000	Interpolation	Straight line interpolation (rapid traverse)
➦ G1	-	G01 G001	Interpolation	Straight line interpolation (feed)
➦ G2	-	G02 G002	Geometry	Motion on a circular path in clockwise rotation
➦ G3	-	G03 G003	Geometry	Motion on a circular path in counterclockwise rotation
➦ G8	-	-	Path slope	Path slope ON Based on the Motion command: *opt-cont-motion* with payload {"permType": "PermOn"}
➦ G9	-	-	Path slope	Path slope OFF Based on the Motion command: *opt-cont-motion* with payload {"permType": "PermOff"}
➦ G16	-	-	Active plane	Active plane off
➦ G17 (G17-19)	-	-	Active plane	Select active plane XY, ZX or YZ
➦ G47	-	-	Tool length correction	Switching on tool length correction Based on the Motion command: *opt-pcs-tool* with payload {"permType": "PermOn"}
➦ G48	-	-	Tool length correction	Switching off tool length correction Based on the Motion command: *opt-pcs-tool* with payload {"permType": "PermOff"}

NC function name			Group	Description
Name	Short form	Alias		
G53	-	-	Zero offset	All zero offsets OFF
G54	-	-	Zero offset	Motion command option PCS set (group) G54 ON
G55	-	-	Zero offset	Motion command option PCS set (group) G55 ON
G56	-	-	Zero offset	Motion command option PCS set (group) G56 ON
G57	-	-	Zero offset	Motion command option PCS set (group) G57 ON
G58	-	-	Zero offset	Motion command option PCS set (group) G58 ON
G59	-	-	Zero offset	Motion command option PCS set (group) G59 ON
G90	-	-	Programming mode	Absolute dimension programming
G91	-	-	Programming mode	Relative dimension programming
G94	-	-	Feed mode	Programming (mm/min)
G153				Switching off product coordinate system
G154				G154(SET = <setName>) Enabling the product coordinate system with set (group) names

10.1.2 NC functions with high-level language syntax

NC function name			Group	Description
Name	Short form	Alias		

PathDynLim	PDL	PATHDYNLIM	-	Programming the dynamic path limits:
				Switching on:
				`PDL({ACC=<number>,}`
				`{DEC=<number> ,}`
				`{JRKACC=<number>, }`
				`{JRKDEC=<number>})`
				
				Switching off:
				`PDL()`
				Programming of dynamic path limits, either separately for *ACC/DEC/JRKACC/JRKDEC* or combined.

WAIT	-	-	-	Stopping the preparation of the script and the Motion command buffer

NC function name			Group	Description
Name	**Short form**	**Alias**		

PolyTrans	-	POLYTRANS	-	Switching on permanently:
				`PTR`
				Switching off:
				`PolyTrans(D1=<num>, D2=<num>)`
				`PolyTrans(EPS=<num>)`
				Based on the Motion command option:
				`path-rounding-with-polynomia`

SetSignal	SSG	SETSIGNAL	-	Triggering a global signal: `SSG(ID=1)` SetSignal(signalId = <INT>) Signal state: *motion/state/functions/somo/signals/[sigId]*
WaitForSignal	WSG	WAITFORSIGNAL	-	Waiting for a global signal: `WSG(ID=1)`
ResetSignal	RSG	RESETSIGNAL	-	`RSG(ID=1)`
AxsDynLim	ADL	AXSDYNLIM	-	Reducing the dynamic limits of the kinematic axis • Enabling: AxsDynLim(AXS="X", VEL=10, ACC=2, DEC=2, JRKACC=17, JRKDEC=17) • Disabling: AxsDynLim(AXS=”X”)
Blend	BLD	BLEND		Permanent blending between Motion commands • Enabling: BLD(D1 = 1, D2=1) • Disabling: BLD()
BlendLocal	BLDL	BLENDLOCAL		Blending to the next two Motion commands (local) • Enabling: BLDL(D1= 1, D2=1)

11 Related documentation

11.1 Overview

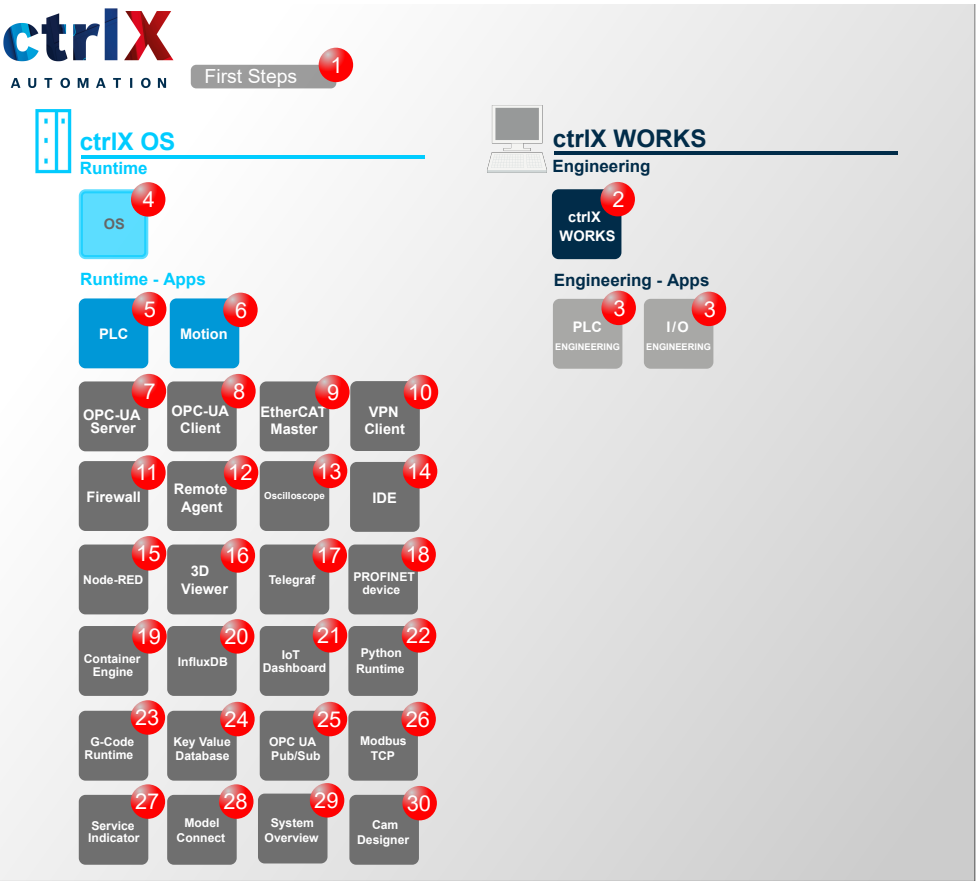

Fig. 9: Overview on further documentations

11.2 ctrlX AUTOMATION

No.	Documentation
1	**ctrlX WORKS First Steps** Quick Start Guide ➦ Web documentation link Ordering information: • DOK-XWORKS-F*STEP*****-QU01-EN-P • R911403760

11.3 ctrlX WORKS

No.	Documentation
2	**ctrlX WORKS - Basic System 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XWORKS-WRK***V03**-APRS-EN-P • R911423376
3	**ctrlX PLC Engineering - PLC Programming System 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XPLC**-PLE***V03**-APRS-EN-P • R911423378
3	**ctrlX PLC Engineering - PLC Libraries 03VRS** Reference Book ➥ Web documentation link Ordering information: • DOK-XPLC**-LIB***V03**-RERS-EN-P • R911423456

11.4 ctrlX OS

No.	Documentation
4	**ctrlX OS - Operating System for ctrlX CORE Control Devices 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-XCR***V03**-APRS-EN-P • R911423382
	ctrlX OS - Data Layer Nodes 03VRS Reference Book ➥ Web documentation link Ordering information: • DOK-XCORE*-DL****V03**-RERS-EN-P • R911423384
	ctrlX OS - Diagnostics 03VRS Reference Book ➥ Web documentation link Ordering information: • DOK-XCORE*-DIAG**V03**-RERS-EN-P • R911423386

11.5 ctrlX OS apps

No.	Documentation
5	**PLC App - PLC Runtime Environment for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-PLC***V03**-APRS-EN-P • R911423401
6	**Motion App - Motion Runtime Environment for ctrlX CORE 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-MOT***V03**-APRS-EN-P • R911423405
7	**OPC UA Server App - OPC UA Server for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-UAS***V03**-APRS-EN-P • R911423392
8	**OPC UA Client App - OPC UA Client for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-UAC***V03**-APRS-EN-P • R911423390
9	**EtherCAT Master App - EtherCAT Master for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-ECM***V03**-APRS-EN-P • R911423394
10	**VPN Client App - Remote Support Software for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-VPN***V03**-APRS-EN-P • R911423388
11	**Firewall App - Security Functions for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-FRW***V03**-APRS-EN-P • R911423397

No.	Documentation
12	**Remote Agent App - ctrlX Device Portal Connection for ctrlX OS Devices 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-RMA***V03**-APRS-EN-P • R911423399
13	**Oscilloscope App - Oscilloscope Function for ctrlX OS Devices 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-OSC***V03**-APRS-EN-P • R911423407
14	**IDE App - Integrated Development Environment 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-IEN***V02**-APRS-EN-P • R911421612
15	**Node RED App - Graphic Programming for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-RED***V03**-APRS-EN-P • R911423403
16	**3D Viewer App - Browser-based 3D Kinematic Simulation for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-3DV***V03**-APRS-EN-P • R911423411
17	**Telegraf App - Server Agent for Collecting Data in the Data Layer 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-TSA***V03**-APRS-EN-P • R911425238
18	**PROFINET Device App - PROFINET Device for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-PND***V03**-APRS-EN-P • R911425232

No.	Documentation
19	**Container Engine App - Using Docker® Images on the ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-DOE***V03**-APRS-EN-P • R911425234
20	**InfluxDB App - Influx Database Connection for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-IDB***V03**-APRS-EN-P • R911425240
21	**IoT Dashboard App - Data Visualization in Dynamic, Interactive Dashboards 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-GDB***V03**-APRS-EN-P • R911425248
22	**Python Runtime App - Python Runtime Environment for ctrlX CORE 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-PYR***V03**-APRS-EN-P • R911425244
23	**G-Code Runtime App - G-Code Interpreter for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-GCO***V03**-APRS-EN-P • R911425246
24	**Key Value Database App - Data Layer Management 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-KVD***V03**-APRS-EN-P • R911425250
25	**OPC UA Pub/Sub App - OPC UA Pub/Sub for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-UAP***V03**-APRS-EN-P • R911423409

No.	Documentation
26	**Modbus TCP App - Modbus TCP Communication over ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-MBT***V03**-APRS-EN-P • R911425236
27	**Service Indicator - App Service Indicator for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-SIN***V03**-APRS-EN-P • R911425242
28	**Model Connect App - Target for Model-based Development and Simulation for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-MOC***V03**-APRS-EN-P • R911425252
29	**System Overview App - System Topology and System Information 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-SOV***V03**-APRS-EN-P • R911425254
30	**Cam Designer - Configuring ctrlX MOTION Cams 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XWORKS-CAM***V03**-APRS-EN-P • R911427217

12 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts provide you with advice and assistance. You can contact us **24/7**.

Service Germany

Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone:	**+49 9352 40 5060**
Fax:	**+49 9352 18 4941**
Email:	➦ service.svc@boschrexroth.de
Internet:	➦ http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide

Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information

To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

13 Index

rexroth
A Bosch Company

Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr a.Main
Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics

R911425246 01