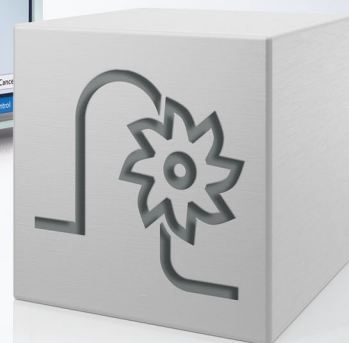
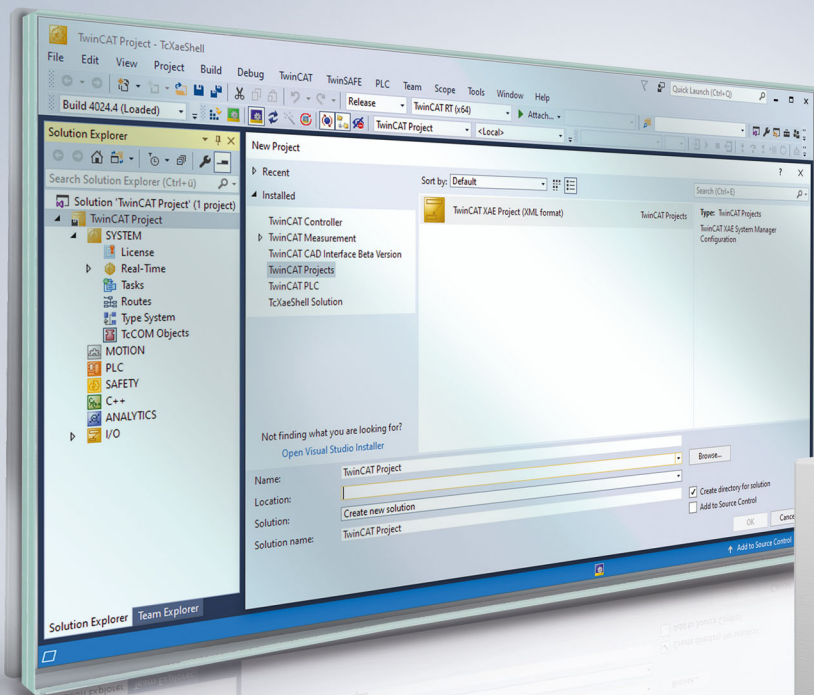


# BECKHOFF New Automation Technology

Functional description | EN

## TF5261 | TwinCAT 3 CNC

Realtime cycles





## Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

### Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

### Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

### Patent Pending

The EtherCAT technology is patent protected, in particular by the following applications and patents:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702

with corresponding applications or registrations in various other countries.



EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

### Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilisation of this document as well as the communication of its contents to others without express authorisation are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

# General and safety instructions

## Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

## Icons in explanatory text

1. Indicates an action.

⇒ Indicates an action statement.

### **DANGER**

#### **Acute danger to life!**

If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.

### **CAUTION**

#### **Personal injury and damage to machines!**


If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.

### **NOTICE**

#### **Restriction or error**

This icon describes restrictions or warns of errors.

#### **Tips and other notes**

 This icon indicates information to assist in general understanding or to provide additional information.


## General example

Example that clarifies the text.

## NC programming example

Programming example (complete NC program or program sequence) of the described function or NC command.

#### **Specific version information**

 Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.

# Table of contents

<b>Notes on the documentation.....</b>	<b>3</b>
<b>General and safety instructions .....</b>	<b>4</b>
<b>1 Overview .....</b>	<b>7</b>
<b>2 Realtime cycle .....</b>	<b>8</b>
2.1 Definition .....	8
2.2 Separating definition and activation .....	9
2.3 Validity.....	10
2.3.1 BLOCK validity .....	10
2.3.2 PROG validity.....	11
2.3.3 GLOBAL validity.....	11
2.4 Action at end .....	12
2.5 Time-critical execution .....	12
2.6 Managing realtime cycles.....	13
2.7 \$IF control structure .....	16
2.7.1 \$IF without frequency.....	16
2.7.2 \$IF with frequency.....	16
2.8 Mathematical expressions.....	20
<b>3 Applicable variables .....</b>	<b>21</b>
3.1 External variables.....	21
3.2 Realtime variables.....	23
3.2.1 Channel-specific realtime variables .....	23
3.2.2 Axis-specific realtime variables .....	24
3.3 Channel variables .....	26
3.3.1 Create / Delete .....	26
3.3.2 Variable types .....	26
3.3.3 Basic properties / visibility / scope .....	26
3.4 Influencing variable synchronisation .....	26
<b>4 Applicable functionality.....</b>	<b>28</b>
4.1 Timer functionality .....	28
4.2 Output of user errors .....	28
4.3 M and H technology functions .....	29
4.4 Spindle programming .....	29
4.4.1 Spindle speed and direction for endless turning .....	30
4.4.2 Positioning spindles .....	31
4.4.3 Spindle stop .....	32
4.4.4 PLC spindles .....	32
4.5 Single axis motions .....	33
4.5.1 Programming syntax .....	33
4.5.2 Position request .....	35
4.5.3 Unit.....	36
<b>5 Diagnosis .....</b>	<b>37</b>
<b>6 Parameter.....</b>	<b>39</b>

---

6.1	Overview of channel and axis parameters .....	39
6.2	Channel parameters .....	40
6.3	Axis parameters .....	42
<b>7</b>	<b>Support and Service .....</b>	<b>43</b>
	<b>Index .....</b>	<b>44</b>

# 1 Overview

## Task

Realtime cycles are cycles which are executed in parallel in the realtime part of the controller. This permits responses to realtime influences.

Possible responses include:

- Spindle control
- Single axis motions
- Output of M functions
- Execution of NC additional functions using # commands

---

**i** This function is an additional option requiring a license.

---

**i** This function is available as of CNC Build V3.1.3105.

---

## Parameterisation

The function must be activated by the channel parameter [P-CHAN-00406 \[► 40\]](#).

Additional user setting options:

- Available memory (P-CHAN-00407) of realtime cycles
- Optional execution time monitoring of realtime cycles for each CNC cycle to prevent realtime timeouts (P-CHAN-00425, [P-CHAN-00426 \[► 41\]](#) and [P-CHAN-00427 \[► 41\]](#))

## Programming

A realtime cycle is defined in the NC program as a complete unit by the NC command **#RT CYCLE**.

### ***Mandatory note on references to other documents***

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

## 2 Realtime cycle

### Scope of a realtime cycle

A realtime cycle is defined in an NC program. It consists of a series of instructions and control structures, e.g. **\$IF conditions**.

After the realtime cycle is loaded and started in the realtime part of the controller, it runs through every CNC cycle **completely**.

### 2.1 Definition

The definition is executed in a block which is enclosed by **#RT CYCLE** and **#RT CYCLE END**.

Syntax:

```
#RT CYCLE [DEF] [[ ID =.. ] SCOPE =.. END_ACTION =..]
```

```
...Instructions
```

```
#RT CYCLE END
```

DEF	Realtime cycle is only defined but not started yet.
ID=..	Unambiguous identifier of the realtime cycle, see Note.
SCOPE	For validity, see <a href="#">Validity [▶ 10]</a> .
	Permissible identifiers are <b>BLOCK</b> , <b>PROG</b> or <b>GLOBAL</b> .
END_ACTION	Behaviour at end of the real-time cycle, see <a href="#">Action at end [▶ 12]</a> .
	Permissible identifiers are <b>MOVE_ABORT</b> or <b>MOVE_CONT</b> .



**The realtime cycle ID must be specified if the keyword DEF is used or SCOPE = GLOBAL.**

If the ID is not specified in these cases, error ID 22003 is output.



## 2.2 Separating definition and activation

If the keyword **DEF** is **not** used, the realtime cycle is defined and **started immediately** when it is loaded to the realtime part of the controller.

### Define and start realtime cycle

```
; Define and start realtime cycle
#RT CYCLE [SCOPE = PROG]
; ...
#RT CYCLE END

; Realtime cycle is already active in this block
N10 G00 X100
```

The realtime cycle is defined by **DEF** but **not started yet**. In this case an ID must be specified. Activation may take later with **#RT CYCLE START**, see [Managing realtime cycles \[► 13\]](#).

### Define realtime cycle only but do not start yet

```
; Define realtime cycle only but do not start
#RT CYCLE DEF [ID = 17 SCOPE = PROG]
; ...
#RT CYCLE END

; Start realtime cycle at any time
#RT CYCLE START [ID = 17]
```

## 2.3 Validity

Every realtime cycle has a defined validity which determines its life. This is programmed by the keyword **SCOPE**.

SCOPE	Validity
BLOCK	Valid in the next NC block.
PROG	Valid for the duration of the active main program.
GLOBAL	Continues action after the end of the main program.

### 2.3.1 BLOCK validity

A realtime cycle with BLOCK validity acts for the duration of the next NC block.

#### BLOCK validity

```
; Define and start realtime cycle
#RT CYCLE [SCOPE = BLOCK]
; ...
#RT CYCLE END

; Realtime cycle only acts on this NC block
G00 X100
```

If the realtime cycle with BLOCK validity is additionally defined by the keyword DEF, it can be used multiple times.

#### BLOCK validity - multiple usage

```
; Define realtime cycle but do not start
#RT CYCLE DEF [ID = 17 SCOPE = BLOCK]
; ...
#RT CYCLE END

; Use realtime cycle for the next block
#RT CYCLE START [ID = 17]
G00 X100
; ...

; Repeat use of the realtime cycle
#RT CYCLE START [ID = 17]
G00 X200
; ...
M30
```

## 2.3.2 PROG validity

A realtime cycle with PROG validity acts for the duration of the active main program. It is deleted at the end of the main program (M30).

### PROG validity

```
; Define and start realtime cycle
#RT CYCLE [SCOPE = PROG]
;...
#RT CYCLE END

; Realtime cycle starts action here
;...

M30
; Realtime cycle is deleted here
```

## 2.3.3 GLOBAL validity

A realtime cycle with **GLOBAL** validity continues its action after the end of the main program (M30). In this case an ID must be specified to permit subsequent management of the realtime cycle.

### NOTICE

**During the reset, cycles with GLOBAL validity are not run through.**

## 2.4 Action at end

A realtime cycle can end in two ways:

1. It loses its validity.
2. It is explicitly deleted by **#RT CYCLE DELETE**.

The keyword **END\_ACTION** controls what is supposed to happen to axis motions which are triggered by the realtime cycle and which are still being executed at the end of the action.

END_ACTION	Meaning
MOVE_ABORT	Abort axis motions (default)
MOVE_CONT	Continue axis motions

## 2.5 Time-critical execution

### NOTICE

**Every active realtime cycle is executed in every CNC interpolator cycle. If several instructions are programmed, execution may take too much time in the CNC cycle. P-CHAN-00425 allows the user to limit the execution time of realtime cycles.**

The parameters P-CHAN-00425, P-CHAN-00426 and P-CHAN-00427 represent a safety mechanism to prevent such realtime timeouts as early as possible.

## 2.6 Managing realtime cycles

The actions of a realtime cycle can be influenced by the following keywords:

**#RT CYCLE [START | HOLD | CONTINUE | ABORT ACTION | DELETE] [ID=..]**

Keyword	Description
<b>START</b>	Realtime cycle runs through now; state of the <b>\$IF</b> condition is re-initialised; motions/actions are executed
<b>HOLD</b>	Realtime cycle no longer runs through; state of the <b>\$IF</b> condition is retained; motions/actions are on hold
<b>CONTINUE</b>	Realtime cycle which is put on hold by <b>HOLD</b> runs through again; state of the <b>\$IF</b> is same as before <b>HOLD</b> ; motions/actions previously on hold are continued
<b>ABORT ACTION</b>	Realtime cycle continues to run through; state of the <b>\$IF</b> condition is retained; motions/actions are aborted
<b>DELETE</b>	Realtime cycle is deleted

## Managing a realtime cycle

```
; Move X axis to 0mm
G00 X0

; Define realtime cycle but do not start
#RT CYCLE DEF [ID = 17 SCOPE = PROG]

; Query ACS position of X axis
$IF ONCE V.RTA.ACS.ACT_POS.X > 200

; Start independent Z axis motion
; ...

$ENDIF

#RT CYCLE END

; Move X axis
G00 X50

; Start realtime cycle
#RT CYCLE START [ID = 17]

; Move X axis, Z motion is started
G00 X250

; Hold Z motion
#RT CYCLE HOLD [ID = 17]

; ...

; Continue Z motion
#RT CYCLE CONTINUE [ID = 17]

; ...

; End main program
; PROG cycle 17 is deleted automatically
M30
```

**Delete global realtime cycle**

```
; Define global realtime cycle
#RT CYCLE [ID = 17 SCOPE = GLOBAL]

; instructions
; ...

#RT CYCLE END

; ...

; Delete global realtime cycle explicitly
#RT CYCLE DELETE [ID = 17]

; End main program
M30
```

## 2.7 \$IF control structure

In realtime cycles a distinction is made between

- \$IF without frequency and
- \$IF with frequency.

\$IF conditions without frequency can be additionally supplemented by **\$ELSEIF** and **\$ELSE**. This is skipped with \$IF conditions with a specified frequency.

### 2.7.1 \$IF without frequency

The instruction block is executed exactly when the condition is true.

#### \$IF with multiple branches

```

; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; Query value of external variable
$IF V.E.VALUE > 0
    ; Increment positive counter
    V.E.COUNT_POS += 1
$ELSEIF V.E.VALUE < 0
    ; Increment negative counter
    V.E.COUNT_NEG += 1
$ELSE
    ; Increment zero counter
    V.E.COUNT_NULL += 1
$ENDIF

#RT CYCLE END
    
```

### 2.7.2 \$IF with frequency

The \$IF control structure has an extended syntax in the realtime cycle. Every **\$IF** can be qualified by several keywords.

**\$IF [ONCE | EDGE | ALWAYS]=..**

; instructions

**\$ENDIF**

Keyword	Meaning
ONCE	If the condition is fulfilled, the instruction section is executed exactly once.
EDGE	For every change in value of the condition from FALSE to TRUE.
ALWAYS	As soon as the condition is fulfilled, the instructions are executed in every run-through.



In the case of an **\$IF** with a specified frequency, multiple branches with **\$ELSEIF** and **\$ELSE** are not available.

#### 2.7.2.1 \$IF ONCE

When the condition is fulfilled the first time, the instruction section is executed once. The instruction section is no longer executed in later run-throughs, even if the condition is fulfilled.



## \$IF ONCE

```
; Define real-time cycle
#RT CYCLE [SCOPE = PROG]

; Query ACS position of X axis
$IF ONCE V.RTA.ACS.ACT_POS.X > 200

    ; Output M function
    M100

$ENDIF

#RT CYCLE END
```

### 2.7.2.2 \$IF EDGE

The instruction block is executed once at every state transition of the condition from **FALSE** to **TRUE** (rising edge).

## \$IF EDGE

```
; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; Query ACS position of X axis
$IF EDGE V.RTA.ACS.ACT_POS.X > 200

    ; Output M function every time X transitions the 200mm limit
    ; in positive direction
    M100

$ENDIF

#RT CYCLE END
```

### 2.7.2.3 \$IF ALWAYS

After the condition was fulfilled once, the instruction block is executed cyclically in every cycle as long as the realtime cycle is active.

From the time when the condition is fulfilled the first time, the instruction part is executed in every run-through. The condition is no longer checked in the following run-throughs; it is assumed to be TRUE as long as the realtime cycle is active.

#### \$IF ALWAYS

```
; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; Query ACS position of X axis
$IF ALWAYS V.RTA.ACS.ACT_POS.X > 200

    ; As soon as X transitions the 200mm limit, the
    ; M function is output in every CNC cycle.
    ; Even if the X position becomes smaller,
    ; M100 continues to be output.
    M100

$ENDIF

#RT CYCLE END
```

### 2.7.2.4 Combination of \$IF conditions with and without frequency



**\$IF** conditions with and without specified frequency can be nested in one another. In this case, ensure that instructions can only be executed when they are actually reached.

For example, a single activation of an **ALWAYS** block does not result in its instructions being executed in all the following CNC cycles. You can use a higher-level **\$IF** condition to prevent this if the result is negative.

If V.E.CONDITION is true in the following NC program and then V.E.VALUE > 100, the ALWAYS block runs through. However, this only applies as long as V.E.CONDITION has the value 1.

#### Nested \$IF conditions

```
; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; external condition
$IF V.E.CONDITION == 1

    ; ALWAYS block
    $IF ALWAYS V.E.VALUE > 100

        ; ALWAYS instructions
        ; ...

    $ENDIF

$ENDIF

#RT CYCLE END
; ...
; End main program
M30
```

## 2.8 Mathematical expressions

Mathematical expressions, functions and operators can be used in the realtime cycle. This applies to the condition of an **\$IF** clause, for variable assignments, etc.

### Examples of mathematical expressions in the realtime cycle

```
; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; Expression in condition
IF SQRT[7 * 5] < 12 * SIN[30]

; ...

$ENDIF

; Assign value of a math. expression to external variable
V.E.VALUE1 = 12 * SIN[V.E.VALUE2] + SQRT[33]

#RT CYCLE END

; End main program
M30
```

## 3 Applicable variables

Synchronous variables must be used for writing within a real-time cycle. The reason for this is that the values of these variables are available at the time of execution. Variables which are not available at the time of declaration are treated as constants and are therefore unassignable.

The following variables can be used:

- V.E variables if they are marked as synchronous
- V.RTG/V.RTA variables
- V.CH variables if they are marked as synchronous

These variable types are discussed in the next subsections.

### **i** Only synchronised variables may be written in a real-time cycle.

If asynchronous variables are written, error ID 22009 is output.

## 3.1 External variables

Read/write accesses to channel-specific and global external variables **V.E.\*** can be made from the realtime cycle.. The variable must be defined as synchronous for a write access.

### Definition of a synchronised V.E. variable

```
...
var[0].name           VALUE1
var[0].type           REAL64
var[0].scope          GLOBAL
var[0].synchronisation TRUE
var[0].access_rights  READ_WRITE
...
```

Only synchronised variables may be written in realtime cycles.

With a read access to a V.E. variable, its value is dependent on its synchronisation.

- A synchronised variable is evaluated in the realtime context of the CNC.
- An asynchronous variable is evaluated at the time of decoding and is used as a constant.

**Accesses to V.E. variables**

```
; Pre-assign variables
V.E.SYNC = 47 ; synchronous variable
V.E.SYNC = 11 ; asynchronous variable

; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; Run through instructions only once
$IF ONCE 1 < 2

    ; Assign new value
    V.E.SYNC = 99

    ; has the value 198
    V.E.VALUE1 = 2 * V.E.SYNC

    ; V.E.ASYNC always has the value 11 in this realtime cycle
    ; right side has the value 22
    V.E.VALUE2 = 2 * V.E.ASYNC

$ENDIF

#RT CYCLE END

; End main program
M30
```

## 3.2 Realtime variables

Variables with the prefix **V.RT\*** are realtime variables. They permit access to the realtime context of the controller. They are synchronised and can both be read and written from the usual context and from the realtime cycle. In this case

- **V.RTG.\*** are channel-specific variables and
- **V.RTA.\*** are axis-specific variables.

### 3.2.1 Channel-specific realtime variables

Variable name	Meaning	Data type	Unit	Permitted access	
				Decoder	Real-time cycle
<b>V.RTG.TIMER[]</b>	Timer value for real-time context, see Section <a href="#">Real-time variables</a> [▶ 23].	UNS32	ms	R	R
<b>V.RTG.CYCLES.DIAG_LEVEL</b>	Diagnosis level for real-time cycles, see Section <a href="#">Diagnosis</a> . [▶ 37]	SGN32	-	R/W	R/W
<b>V.RTG.OVERRIDE.VEL.CYCLE</b>	Velocity override from real-time cycles	UNS16	%	R	R/W
<b>V.RTG.OVERRIDE.VEL.TOTAL</b>	Velocity override combined from all influences	UNS32	%	R	R
<b>V.RTG.MEAS_DELTA</b>	Delta between programmed and actual edge banding	REAL64	mm	R/W	-
<b>V.RTG.LOOP.ENABLED</b> as of V3.1.3105.01	Loop condition for a real-time loop	BOOL	-	R/W	R/W
<b>V.RTG.LOOP.COUNT</b> as of V3.1.3105.01	Number of real-time loops executed	SGN32	-	R/W	R/W

## Reduce path override

```

; Define real-time cycle
#RT CYCLE [SCOPE = PROG]

; Query ACS position of X axis
; Override must be written cyclically, i.e. without ONCE
$IF V.RTA.ACS.ACT_POS.X > 200

    ; Reduce channel override to 75%
    V.RTG.OVERRIDE.VEL.CYCLE = 75

$ENDIF

#RT CYCLE END

; Move X axis to 500mm
; move slower at 200mm
G00 X500

; End main program
M30

```

### 3.2.2 Axis-specific realtime variables

Variable name	Meaning	Data type	Unit	Permitted access	
				Decoder	Real-time cycle
<b>V.RTA.FIXED_STOP.ACTIVE.X</b>	Move to fixed stop active	Boolean	-	L	L
<b>V.RTA.FIXED_STOP.DETECTED.X</b>	Move to fixed stop detected	Boolean	-	L	L
<b>V.RTA.FIXED_STOP.ACS.POS.X</b>	Fixed stop position recorded in axis coordinate system	SGN64	[mm, inch]	L	L

Variable name	Meaning	Data type	Unit	Permitted access	
				Decoder	Real-time cycle
<b>V.RTA.OVERRIDE.VEL.CYCLE</b>	Velocity override from realtime cycles, for independent axis motion.	UNS16	%	R	R/W
<b>V.RTA.OVERRIDE.VEL.TOTAL</b>	Total velocity override of axis	UNS16	%	R	R
<b>V.RTA.ACS.ACT_POS</b>	Current command axis position in axis coordinate system	SGN64	mm	R	R
<b>V.RTA.ACS.CUR_POS</b>	Current actual axis position in axis coordinate system	REAL64	mm	R	R
<b>V.RTA.ACS.CMD_POS</b>	Current target axis position in axis coordinate system	REAL64	mm	R	R
<b>V.RTA.MCS.ACT_POS</b>	Current command axis position in machine coordinate system	SGN32	mm	R	R
<b>V.RTA.MCS.CUR_POS</b>	Current actual axis position in machine coordinate system	SGN32	mm	R	R
<b>V.RTA.MCS.CMD_POS</b>	Current target axis position in machine coordinate system	SGN32	mm	R	R
<b>V.RTA.MCS.DIST_TO_GO</b>	Current axis distance to go in machine coordinate system	SGN32	mm	R	R



<b>V.RTA.PCS.ACT_POS</b>	Current command axis position in program coordinate system	SGN32	mm	R	R
<b>V.RTA.PCS.CUR_POS</b>	Current actual axis position in program coordinate system	SGN32	mm	R	R
<b>V.RTA.PCS.CMD_POS</b>	Current target axis position in program coordinate system	SGN32	mm	R	R
<b>V.RTA.PCS.DIST_TO_GO</b>	Current axis distance to go in program coordinate system	SGN32	mm	R	R
<b>V.RTA.IPO.ACT_POS</b>	Current command axis position in current interpolator coordinate system	SGN32	mm	R	R
<b>V.RTA.IPO.CUR_POS</b>	Current actual axis position in current interpolator coordinate system	SGN32	mm	R	R
<b>V.RTA.IPO.CMD_POS</b>	Current target axis position in current interpolator coordinate system	SGN32	mm	R	R
<b>V.RTA.IPO.DIST_TO_GO</b>	Axis distance to go in current interpolator coordinate system	SGN32	mm	R	R

## 3.3 Channel variables

V.CH. variables are self-defined channel-specific variables.



In order to use V.CH. variables, memory must be reserved using the parameter P-CHAN-00424.

### 3.3.1 Create / Delete

V.CH. variables are created in the V.E. variable lists. They are marked by the prefix “vch”.

Parameter settings include:

- Name
- Type
- Array size (with not arrays)
- Synchronisation

#### Create / Delete

vch[0].name	Var_name
vch[0].type	SGN08
vch[0].array_size	0
vch[0].synchronisation	TRUE

Variables can only be deleted by removing them from the list and re-interpreting them.

### 3.3.2 Variable types

V.CH. variables can assume all V.E. variable types. This also applies to structure definitions. The basic types are:

BOOLEAN / SGN08 / UNS08 / SGN16 / UNS16 / SGN32 / UNS32 / REAL64 / STRING



#### Alignment

V.CH. structure variables have the alignment of V.E. variables. V.CH. structures are therefore mutually assignable to V.E. structures of the same type.

### 3.3.3 Basic properties / visibility / scope

The variables are readable and writeable from all NC program levels. No access to V.CH. Variables is possible from the PLC.

## 3.4 Influencing variable synchronisation

If an asynchronous V.CH./V.E. variable has to be used for a realtime cycle, synchronicity can be forced for the current NC block.

This is done by adding an “s” to the V node of the variable. It is only valid for the current block. Access to the variable remains asynchronous for all other blocks.

```
N10 #RT CYCLE [ID=17 SCOPE = PROG]
N20 $IF ONCE Vs.CH.VarTest1 < 1 ; Asynchronous access is ; forced
here
N30 Vs.CH.VarTest1 = 1 ; Asynchronous access is ;
forced here
N40 $ENDIF
N50 #RT CYCLE END
```

```
N60 M30
```

Synchronisation is also possible outside of a realtime cycle.

```
N10 #RT CYCLE [ID=17 SCOPE = GLOBAL]
N20 $IF ONCE 1 < 2
N30 V.E.LEVEL_1_A[0].REAL64 = V.E.LEVEL_1_A[0]
N40 $ENDIF
N50 #RT CYCLE END

N60 Vs.E.LEVEL_1_A[0].REAL64 ; synchronous access
N70 M30
```

## 4 Applicable functionality

### 4.1 Timer functionality

The **#TIMER** functionality is available for realtime cycles. The measured times are saved to the realtime variables **V.RTG.TIMER[]**. Realtime timers are different from decoder timers.

#### #TIMER functionality

```
; Output variables
$FOR P1 = 0, 4, 1
#MSG SAVE EXCLUSIVE ["V.RTG.TIMER[%d] = %f", P1, V.RTG.TIMER[P1]]
$ENDFOR

; Define realtime cycle
#RT CYCLE [ID = 17 SCOPE = PROG]
  $IF ONCE 1 < 2
    #TIMER START SYN [ID = 0] ; Start Timer 0
    #TIMER START SYN [ID = 2] ; Start Timer 2
  $ENDIF
#RT CYCLE END
#FLUSH WAIT

; please wait
#TIME 1.5

; Define realtime cycle
#RT CYCLE [ID = 18 SCOPE = PROG]
  $IF ONCE 1 < 2
    #TIMER STOP SYN [ID = 0] ; Stop Timer 0
    #TIMER READ SYN [ID = 0] ; Read Timer 0
    #TIMER READ SYN [ID = 2] ; Read Timer 2 without stopping
  $ENDIF
#RT CYCLE END
#FLUSH WAIT

; Output variables
$FOR P1 = 0, 4, 1
  #MSG SAVE EXCLUSIVE ["V.RTG.TIMER[%d] = %f", P1, V.RTG.TIMER[P1]]
$ENDFOR

; End main program
M30
```

#### NOTICE

Only asynchronous **#TIMER** commands are permitted. All **#TIMER** commands in realtime cycles must be marked as synchronous by the keyword **SYN**.

### 4.2 Output of user errors

The NC command **#ERROR** permits user-defined errors messages within a real-time cycle. The syntax is described in [PROG//User-defined error output (**#ERROR**)].



**If an error is output with RC >= 1, the NC channel goes to error state and possible path movements are stopped. If warnings are output (RF = 0), path movements are continued.**

#### #ERROR Command within a real-time cycle

```
; Define RT cycle; output error as soon as X > 99
#RT CYCLE [SCOPE = PROG]
  $IF ONCE V.RTA.ACS.ACT_POS.X > 99
    #ERROR [ID=666 RC=2 PM1=1 PV1=99]
  $ENDIF
#RT CYCLE END
```

## 4.3 M and H technology functions

Technology functions can be output from the realtime cycle. The following rules then apply:

- Only channel-specific technology functions can be output but not axis-specific technology functions.
- Only technology functions defined in the channel parameters can be output. All others result in an error.
- All technology functions are output as MOS (without synchronisation), irrespective of the configured synchronisation type.

Realtime technology functions are output in a newly created section of the HLI and **not** in the section of classic technology functions. On the HLI, they are provided with the new identifier **POS\_RT**.

Realtime technology functions are output on the HLI in the same CNC cycle in which they are commanded. The system does not wait for free resources. An error is output if realtime technology functions cannot be output in the correct cycle, e.g. because the HLI is assigned. Realtime technology functions are created on the HLI in the sequence in which they are commanded by realtime cycles. They are numbered consecutively within a cycle and time-stamped.

The user must ensure the following points to ensure that they function:

- The PLC must have a task in the CNC cycle to read realtime technology functions.
- The PLC must acknowledge a sufficient number of realtime technology functions in every cycle in order to create sufficient space on the HLI for realtime technology functions of the next CNC cycle.

### Output of technology functions

```
; Move X axis to 0mm
G00 X0

; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; Query ACS position of X axis
$IF ONCE V.RTA.ACS.ACT_POS.X > 200

; Output M function
; M100 must be configured in the channel
; will still be output in this cycle
; will be output without synchronisation (MOS)
M100

$ENDIF

#RT CYCLE END

; Move X axis to 300mm
G00 X300

; End main program
M30
```

## 4.4 Spindle programming

Spindle programming is possible in realtime cycles. Programming both main and counter spindles is supported.

### Assigned resources

**I** Spindle jobs from realtime cycles are output over standard CNC mechanisms. As opposed to spindle programming from the decoder context (default response), the system does not wait until there are sufficient resources for commanding. An error is output if there is no buffer space available for realtime commands or if the spindle interface is still occupied with previous jobs.

Programming must take place completely in one block. For example, it is not permitted to first define the direction of rotation with **M3** and then program the speed in a subsequent block. **M3** and speed must be programmed in the same block.

### 4.4.1 Spindle speed and direction for endless turning

Endless spindle turning can take place by simultaneously programming **M3/M4** and

- the **S** word for main spindles
- the **REV** parameter for counter spindles

#### Endless turning

```
; Define realtime cycle
#RT CYCLE [...]

; external trigger condition
$IF ONCE V.E.TRIGGER == 1

; Command main spindle
M03 S1000

; Command counter spindle
S2[M03 REV=1000]

$ENDIF

#RT CYCLE END

; ...

; End main program
M30
```

## 4.4.2 Positioning spindles

To position spindles, the following must be programmed:

- **M3/M4** for direction of movement,
- **M19** for spindle positioning,
- **S/REV** for direction of rotation and
- **S.POS/POS** for target position

### Position spindle

```
; Define realtime cycle
#RT CYCLE [...]

; external trigger condition
$IF ONCE V.E.TRIGGER == 1

; Command main spindle
M03 M19 S1000 S.POS314

; Command counter spindle
S2[M03 M19 REV=1000 POS=314]

$ENDIF

#RT CYCLE END

; ...

; End main program
M30
```

### 4.4.3 Spindle stop

Spindle stop is programmed with **M5**.

#### Spindle stop

```
; Define realtime cycle
#RT CYCLE [...]

; external trigger condition
$IF ONCE V.E.TRIGGER == 1

; Command main spindle
M05

; Command counter spindle
S2[M05]

$ENDIF

#RT CYCLE END

; ...

; End main program
M30
```

### 4.4.4 PLC spindles

A PLC spindle is identified by the channel parameter P-CHAN-00069 **plc\_control**.

#### Configure PLC spindle

```
...
spindel[2].plc_control 1
spindel[2].bezeichnung S3
spindel[2].log_achs_nr 10
...
```

Programming a PLC spindle in the realtime cycle is no different than programming a CNC spindle. The commands are sent directly to the PLC. All technology functions involved are output with the synchronisation type MOS.



## 4.5 Single axis motions

In order to use single axis motion in realtime cycles, the channel axis interface must be enabled by P-AXIS-00457.

### 4.5.1 Programming syntax

Syntax:

```
<axis_name> [ INDP ABORTING | BUFFERED [ OFFSET=..] G90 | G91 G00 | G01 [FEED=..] [POS=..] [DIR=..] | STOP ]
```

<axis_name>	Name of independent axis
INDP	Identifier for an independent axis
ABORTING / BUFFERED	ABORTING interrupts the axis motion of a programmed axis that was previously started. If the keyword is not specified, ABORTING is used by default. <b>Note:</b> BUFFERED not available.
STOP	Axis stop and abortion of current motion job. Not combinable with other keywords
OFFSET	Parameter specifying which axis offsets are to be included in the calculation, see Offset table
G90 / G91	Absolute/relative dimension
G00 / G01	Rapid traverse/linear interpolation
FEED	Axis-specific feed rate in [mm/min, m/min, inch/min] If FEED is programmed without specifying the direction and with a G function G00/G01, the axis is triggered in endless motion at the specified feed rate.
POS	Axis position in [mm, inch]
DIR	Direction; permitted parameters: <ul style="list-style-type: none"> <li>• POS, positive direction</li> <li>• CUR, current direction</li> <li>• NEG, negative direction</li> </ul> Mathematical expressions can also be used to result in one of the values.



**If 'POS' is used, it is mandatory to specify 'G90' or 'G91'.**

If this specification is missing, the error ID 50967 is output.

Offset keyword	Meaning
ALL	All active axis offsets
ZERO	Zero offsets
ADD_ZERO	Additive zero offsets and reference point offsets
PSET	Position presets
CLAMP	Clamping offsets
TOOL	Tool offsets
MEASURE	Measuring offsets
MANUAL	Manual mode offsets

### Positioning with abort condition

```

; Task definition:
; As soon as the X axis moves across Position 100, the Z axis
; moves to Position 900.
; If the Z axis has already moved, this motion
; is aborted
; The motion is executed with all offsets included in the calculation.
N010 #RT CYCLE [ID=2 SCOPE=PROG]
N020 $IF ONCE V.RTA.ACS.ACT_POS.X > 100
N030 Z [INDP ABORTING G01 G90 FEED=500 POS=900 OFFSET=ALL]
N040 $ENDIF
N050 #RT CYCLE END
N060 G90 X100
N070 G90 X200
N080 M30

```

### Endless turning with start condition

```

; Task definition:
; Turn Z endlessly in positive direction if X > 1mm

; Define real-time cycle
N10 #RT CYCLE [SCOPE = PROG]
N20 $IF ONCE V.RTA.ACS.ACT_POS.X > 1
N30 Z[INDP DIR = POS FEED = 1000]
N40 $ENDIF
N50 #RT CYCLE END

; Start motion
N60 G01 X100

; End--
N70 M30

```

## 4.5.2 Position request

A position request takes place after deleting a realtime cycle with **#RT CYCLE DELETE**.

With realtime cycles which have the validity SCOPE = BLOCK, a position request occurs after the validity is left, provided an axis for which an independent axis motion was previously programmed in the realtime cycle is programmed in the NC program.

### Realtime cycle position request

```
N010 G00 X0 Y0 Z0 F500 G70
N020 #FLUSH WAIT

; Activate realtime cycle and move Z
N030 #RT CYCLE [SCOPE = BLOCK END_ACTION = MOVE_ABORT]
N040 $IF ONCE V.RTA.ACS.ACT POS.X > 10
N050     Z[INDP ABORTING G0 G90 POS = 137]
N060 $ENDIF
N070 #RT CYCLE END

; This is the block in which the realtime cycle is executed.
N080 G01 X100 F1000
N090 G01 X-100 F1000
N100 G01 X100 F1000

; Here no position request takes place
; as Z was moved
N120 G01 Z100 F1000
; Here the realtime cycle is already outside its validity
; It is therefore ended and all axis motions which it
; caused are aborted..

N130 G01 Z-100 F1000
N140 G01 Z100 F1000

N150 M30
```

### 4.5.3 Unit

The unit in a realtime cycle is defined at the time of its definition. It is the unit valid at the time of definition.

When the unit changes within the NC program in which realtime cycles are used, switchover takes place at the start of the realtime cycle.

#### Switch over units for realtime cycles

```

N010 G00 X0 Y0 Z0 F500 G71
N020 #FLUSH WAIT

; Activate realtime cycle and move Z,
; here the position is traversed in mm
N030 #RT CYCLE DEF[ID = 15 SCOPE = BLOCK END_ACTION = MOVE_ABORT]
N040 $IF ONCE V.RTA.ACS.ACT_POS.X > 10
N050     Z[INDP ABORTING G0 G90 POS = 137 ]
N060 $ENDIF
N070 #RT CYCLE END

; here a switchover to inch takes place
N080 G70
; here the realtime cycle is started but positions
; in the realtime cycle are in mm
N090 #RT CYCLE START [ID = 15]

; This is the block in which the real.time cycle is executed.
N100 G01 X100 F1000
N110 G01 X-100 F1000
N120 G01 X100 F1000

; Here no position request takes place
; as Z was moved
N130 G01 Z100 F1000
N140 G01 Z-100 F1000
N150 G01 Z100 F1000

N160 M30

```

## 5 Diagnosis

If the realtime cycle function is activated, information about the realtime cycles is also output in order to generate diagnosis data.

### Example of diagnosis data

```

PATH : REALTIME CYCLES DIAGNOSIS DATA CHANNEL NO.: 1
=====

Note:
Some messages may be hidden in the diagnosis.
Please refer to the realtime cycle documentation to find out how to influence the diagnosis.

Timestamp          Level          Message
-----
50024      INFO      Realtime cycle manager was initialised
50024      INFO      Realtime cycle manager: 0 cycles
426098     INFO      Cycle 1 was created with PROG validity
426098     INFO      Cycle 1 was started
...
    
```

Every line is provided with a diagnosis level. The user can influence the output by using the realtime variable **V.RTG.CYCLES.DIAG\_LEVEL**. The higher the variable value, the more information is output. The following diagnosis levels can be set.

Value	Description
0	No diagnosis data.
1	Errors from realtime cycles are output.
2	Warnings that may indicate a problem.
3	Information on managing realtime cycles, default.
4	Information about realtime cycle actions.

All information about levels 0 to 3 are output with the value preset to **V.RTG.CYCLES.DIAG\_LEVEL = 3**.

**Setting diagnosis level**

```
; Set diagnosis very high
; All information is output
V.RTG.CYCLES.DIAG_LEVEL = 99

; Define realtime cycle
#RT CYCLE [SCOPE = PROG]

; Actions
; ...

#RT CYCLE END

; ...

; End main program
M30
```

## 6 Parameter

### 6.1 Overview of channel and axis parameters

ID	Description
P-CHAN-00406	Enable realtime cycle functionality
P-CHAN-00407	Memory for real-time cycles
P-CHAN-00424	Memory size for V.CH. variables
P-CHAN-00425	Max. execution time of realtime cycles per CNC cycle
P-CHAN-00426	Number of elementary instructions for time check
P-CHAN-00427	Max. number of elementary instructions per CNC cycle
P-CHAN-00480	Max. number of actions within a real-time cycle

ID	Description
P-AXIS-00457	Enable PLCopen interface of a channel axis

## 6.2 Channel parameters

P-CHAN-00406	Activating real-time cycles
Description	This parameter enables the real-time cycle function in the NC channel. The controller must be restarted to adopt the change. Example: <code>configuration.rt_cycles.enable 1</code>
Parameter	<code>configuration.rt_cycles.enable</code>
Data type	BOOLEAN
Data range	0/1
Dimension	----
Default value	0
Remarks	This parameter is available as of CNC Build V3.1.3107.10 or higher. Use of the parameter "rt_cycles.enable" <code>rt_cycles.enable 1</code> (as of V3.1.3105) continues to be supported.

P-CHAN-00407	Memory size for real-time cycles
Description	This parameter defines the size of the memory for real-time cycles. The size of the memory is specified in bytes. The controller must be restarted to adopt the change. Then the specified memory is also available for real-time cycles. Example: <code>configuration.rt_cycles.memory 60000</code>
Parameter	<code>configuration.rt_cycles.memory</code>
Data type	UNS32
Data range	0 ... MAX(UNS32) - 1
Dimension	----
Default value	48000
Remarks	<b>Note:</b> The assignment of P-CHAN-00407 is only necessary if the memory set by default is no longer sufficient due to activation of the real-time cycles (P-CHAN-00406 [► 40]).  This parameter is available as of CNC Build V3.1.3107.10 or higher. Use of the parameter "rt_cycles.memory" <code>rt_cycles.memory 60000</code> (as of V3.1.3105) continues to be supported.

P-CHAN-00424	Memory size for V.CH. variables
Description	This parameter defines the size of the memory in bytes for V.CH. variables. Example: <code>configuration.decoder.v_ch_memory 10000</code>
Parameter	<code>configuration.decoder.v_ch_memory</code>
Data type	UNS32
Data range	0 ... MAX(UNS32) - 1
Dimension	Byte
Default value	0
Remarks	The memory contains application data and internal management data. This means that the actually available user memory is always smaller than the set value. This parameter is available as of Version V3.1.3107.10.



	Direct use of the parameter “v_ch_memory“ without any structure (as of V3.1.3104) is still supported but should no longer be used in new applications. v_ch_memory 10000
--	---

<b>P-CHAN-00425 Max. execution time of realtime cycles per CNC cycle</b>	
Description	This parameter defines the maximum execution time of realtime cycles in the NC channel. The parameter is specified in percent (%) and refers to the length of the CNC cycle. Example: If the realtime task of the CNC is clocked at 2ms and parameter P-CHAN-00425 is 75, the realtime cycles may require a total execution time of maximum 1.5ms. If this time is exceeded, error ID 50939 is output.
Parameter	rt_cycles.max_duration
Data type	UNS16
Data range	0 < P-CHAN-00425 < MAX_UN16
Dimension	%
Default value	75
Remarks	The user has no restrictions regarding the number of instructions within a realtime cycle. Realtime timeouts will occur if realtime cycles contain too many instructions and cannot be executed in one CNC cycle,  The parameters P-CHAN-00426 and P-CHAN-00427 represent a safety mechanism to prevent such realtime timeouts as early as possible.

<b>P-CHAN-00426 Number of elementary instructions for time check</b>	
Description	This parameter defines the number of elementary instructions after which another time check is executed.  A check for the execution time of realtime cycles must be made within a CNC cycle to see whether the permitted execution time is already exceeded. This takes place by checking the expired time after a specific number of elementary instructions in a cycle. Parameter P-CHAN-00426 indicates the number of these elementary instructions.
Parameter	rt_cycles.cont_steps
Data type	UNS32
Data range	0 < P-CHAN-00426 < MAX_UN32
Dimension	----
Default value	100
Remarks	The user has no restrictions regarding the number of instructions within a realtime cycle. Realtime timeouts will occur if realtime cycles contain too many instructions and cannot be executed in one CNC cycle,  The parameters P-CHAN-00425 and P-CHAN-00427 represent a safety mechanism to prevent such realtime timeouts as early as possible.

<b>P-CHAN-00427 Max. number of elementary instructions per CNC cycle</b>	
Description	This parameter defines the maximum number of elementary instructions per CNC cycle. P-CHAN-00427 as well as P-CHAN-00425 and P-CHAN-00426 can limit the execution time in realtime cycles in the CNC cycle.  If the number of elementary instructions in the current CNC cycle exceeds the value of this parameter, error ID 50854 is output.
Parameter	rt_cycles.max_steps
Data type	UNS32

Data range	0 < P-CHAN-00427 < MAX_UN32
Dimension	----
Default value	MAX_UN32 - 1
Remarks	<p>The user has no restrictions regarding the number of instructions within a realtime cycle. Realtime timeouts will occur if realtime cycles contain too many instructions and cannot be executed in one CNC cycle,</p> <p>The parameters P-CHAN-00425 and P-CHAN-00426 represent a safety mechanism to prevent such realtime timeouts as early as possible.</p>

<b>P-CHAN-00480</b>	<b>Max. number of actions in a real-time cycle</b>
Description	<p>This parameter defines the maximum number of possible actions within a real-time cycle.</p> <p>Possible actions include single-axis movement, spindle command, etc.</p> <p>If too many actions are commanded within a real-time cycle, error ID 51028 is output.</p>
Parameter	configuration.rt_cycles.buffers
Data type	UNS16
Data range	0 ... MAX(UNS16) - 1
Dimension	----
Default value	5
Remarks	The parameter is available as of V3.1.3107.10

### 6.3 Axis parameters

<b>P-AXIS-00457</b>	<b>Enable PLCopen interface of a channel axis</b>
Description	<p>This parameter enables the PLCopen interface of a channel axis. The axis can then be commanded by the PLC or by the NC program (see PROG//PLCopen programming) with motion commands (absolute and relative positioning, path motions with velocity, and stopping).</p> <p>The following arrays are supported:</p> <ul style="list-style-type: none"> <li>• MC_MoveAbsolute</li> <li>• MC_MoveRelative</li> <li>• MC_MoveVelocity</li> <li>• MC_Halt</li> </ul>
Parameter	kenngr.enable_single_axis
Data type	BOOLEAN
Data range	0/1
Axis types	T, R
Dimension	T: ----   R: ----
Default value	0
Drive types	----
Remarks	Commands are possible for linear axes and rotary axes (modulo axes), see P-AXIS-00015.

## 7 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

### Download finder

Our [download finder](#) contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

### Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for [local support and service](#) on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: [www.beckhoff.com](http://www.beckhoff.com)

You will also find further documentation for Beckhoff components there.

### Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963-157  
e-mail: [support@beckhoff.com](mailto:support@beckhoff.com)

### Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963-460  
e-mail: [service@beckhoff.com](mailto:service@beckhoff.com)

### Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20  
33415 Verl  
Germany

Phone: +49 5246 963-0  
e-mail: [info@beckhoff.com](mailto:info@beckhoff.com)  
web: [www.beckhoff.com](http://www.beckhoff.com)

# Index

## **P**

---

P-AXIS-00457	42
P-CHAN-00406	40
P-CHAN-00407	40
P-CHAN-00424	40
P-CHAN-00425	41
P-CHAN-00426	41
P-CHAN-00427	41
P-CHAN-00480	42



More Information:  
[www.beckhoff.com/TF5261](http://www.beckhoff.com/TF5261)

Beckhoff Automation GmbH & Co. KG  
Hülshorstweg 20  
33415 Verl  
Germany  
Phone: +49 5246 9630  
[info@beckhoff.com](mailto:info@beckhoff.com)  
[www.beckhoff.com](http://www.beckhoff.com)

