# DOCUMENTATION ISG-kernel

# Funktionsbeschreibung Realtime cycles

Short Description:
FCT-C32

Documentation version: 1.13
07/11/2024

# Preface

## Legal information

This documentation was produced with utmost care. The products and scope of functions described are under continuous development. We reserve the right to revise and amend the documentation at any time and without prior notice.

No claims may be made for products which have already been delivered if such claims are based on the specifications, figures and descriptions contained in this documentation.

## Personnel qualifications

This description is solely intended for skilled technicians who were trained in control, automation and drive systems and who are familiar with the applicable standards, the relevant documentation and the machining application.

It is absolutely vital to refer to this documentation, the instructions below and the explanations to carry out installation and commissioning work. Skilled technicians are under the obligation to use the documentation duly published for every installation and commissioning operation.

Skilled technicians must ensure that the application or use of the products described fulfil all safety requirements including all applicable laws, regulations, provisions and standards.

## Further information

Links below (DE)

https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

or (EN)

https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

contains further information on messages generated in the NC kernel, online help, PLC libraries, tools, etc. in addition to the current documentation.

## Disclaimer

It is forbidden to make any changes to the software configuration which are not contained in the options described in this documentation.

## Trade marks and patents

The name ISG®, ISG kernel®, ISG virtuos®, ISG dirigent® and the associated logos are registered and licensed trade marks of ISG Industrielle Steuerungstechnik GmbH.

The use of other trade marks or logos contained in this documentation by third parties may result in a violation of the rights of the respective trade mark owners.

## Copyright

# 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

➢ 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.

**Attention**

**Restriction or error**

This icon describes restrictions or warns of errors.

**Notice**

**Tips and other notes**

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

**Example**

**General example**

Example that clarifies the text.

**Programing Example**

**NC programming example**

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

**Release Note**

**Specific version information**

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

# Contents

# 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

**Notice**

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

**Release Note**

**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 [▸ 42])

## 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 Validity [▸ 9]. |
| | Permissible identifiers are **BLOCK**, **PROG** or **GLOBAL**. |
| END_ACTION | Behaviour at end of the real-time cycle, see Action at end [▸ 11]. |
| | Permissible identifiers are **MOVE_ABORT** or **MOVE_CONT**. |

**Notice**

**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.

**Programing Example**

**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 [▸ 12]**.**

**Programing Example**

**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.

**Programing Example**

**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.

**Programing Example**

**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).

**Programing Example**

**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.

**Attention**

**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

**Attention**

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

## Programing Example

**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
```

## Programing Example

**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.

**Programing Example**

**$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. |

**Notice**

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.

**Programing Example**

**$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).

**Programing Example**

**$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.

**Programing Example**

**$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

**Notice**

**$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.

**Programing Example**

**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.

**Programing Example**

**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.

**Notice**

**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.

**Programing Example**

**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 asynchronised variable is evaluated at the time of decoding and is used as a constant.

## Programing Example

### 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.

**Notice**

**A read or write access to a realtime variable causes flushing of the NC channel.**

For example, flushing the NC channel can result in the error ID 20651 if tool radius compensation (G41/G42) is active.

### 3.2.1 Channel-specific realtime variables

| Variable name | Meaning | Data type | Unit | Permitted access | |
|---|---|---|---|---|---|
| | | | | Decoder | Real-time cycle |
| **V.RTG.TIMER[]** | Timer value for real-time context, see Section Real-time variables [▸ 22]. | UNS32 | ms | R | R |
| **V.RTG.CYCLES.DIAG_LEVEL** | Diagnosis level for real-time cycles, see Section Diagnosis. [▸ 37] | SGN32 | - | R/W | R/W |
| **V.RTG.OVERRIDE.VEL.CYCLE** | Velocity override from real-time cycles | UNS16 | % | R | R/W |
| **V.RTG.OVERRIDE.VEL.TOTAL** | Velocity override combined from all influences | UNS32 | % | R | R |
| **V.RTG.MEAS_DELTA** | Delta between programmed and actual edge banding | REAL64 | mm | R/W | - |
| **V.RTG.LOOP.ENABLED as of V3.1.3105.01** | Loop condition for a real-time loop | BOOL | - | R/W | R/W |
| **V.RTG.LOOP.COUNT as of V3.1.3105.01** | Number of real-time loops executed | SGN32 | - | R/W | R/W |

**Programing Example**

**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 |
| V.RTA.FIXED_STOP.ACTIVE.X | Move to fixed stop active | Boolean | - | L | L |
| V.RTA.FIXED_STOP.DETECTED.X | Move to fixed stop detected | Boolean | - | L | L |
| V.RTA.FIXED_STOP.ACS.POS.X | 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 |
| **V.RTA.OVERRIDE.VEL. CYCLE** | Velocity override from real-time cycles, for independent axis motion. | UNS16 | % | R | R/W |
| **V.RTA.OVERRIDE.VEL. TOTAL** | Total velocity override of axis | UNS16 | % | R | R |
| **V.RTA.ACS.ACT_POS** | Current command axis position in axis coordinate system | SGN64 | mm | R | R |
| **V.RTA.ACS.CUR_POS** | Current actual axis position in axis coordinate system | REAL64 | mm | R | R |
| **V.RTA.ACS.CMD_POS** | Current target axis position in axis coordinate system | REAL64 | mm | R | R |
| **V.RTA.MCS.ACT_POS** | Current command axis position in machine coordinate system | SGN32 | mm | R | R |
| **V.RTA.MCS.CUR_POS** | Current actual axis position in machine coordinate system | SGN32 | mm | R | R |
| **V.RTA.MCS.CMD_POS** | Current target axis position in machine coordinate system | SGN32 | mm | R | R |
| **V.RTA.MCS.DIST_TO_GO** | Current axis distance to go in machine coordinate system | SGN32 | mm | R | R |
| **V.RTA.PCS.ACT_POS** | Current command axis position in program coordinate system | SGN32 | mm | R | R |
| **V.RTA.PCS.CUR_POS** | Current actual axis position in program coordinate system | SGN32 | mm | R | R |
| **V.RTA.PCS.CMD_POS** | Current target axis position in program coordinate system | SGN32 | mm | R | R |
| **V.RTA.PCS.DIST_TO_GO** | Current axis distance to go in program coordinate system | SGN32 | mm | R | R |
| **V.RTA.IPO.ACT_POS** | Current command axis position in current interpolator coordinate system | SGN32 | mm | R | R |
| **V.RTA.IPO.CUR_POS** | Current actual axis position in current interpolator coordinate system | SGN32 | mm | R | R |
| **V.RTA.IPO.CMD_POS** | Current target axis position in current interpolator coordinate system | SGN32 | mm | R | R |
| **V.RTA.IPO.DIST_TO_GO** | 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.

**Notice**

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

**Programing Example**

**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

**Notice**

**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.

**Programing Example**

**#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
```

**Attention**

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 reald-time cycle. The syntax is described in [PROG//User-defined error output (#ERROR)].

**Notice**

**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.**

**Programing Example**

**#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.

**Programing Example**

**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.

**Notice**

**Assigned resources**

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

.

**Programing Example**

**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

.

**Programing Example**

**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**.

**Programing Example**

**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**.

**Programing Example**

**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.<br>**Note:** 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]<br>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:<br>• POS, positive direction<br>• CUR, current direction<br>• NEG, negative direction<br>Mathematical expressions can also be used to result in one of the values. |

**Notice**

**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 |

## Programing Example

**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
```

## Programing Example

**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.

**Programing Example**

**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.

**Programing Example**

**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**

**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 in-
fluence 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 real-time 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**.

**Programing Example**

**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.<br>The controller must be restarted to adopt the change.<br>Example:<br>`configuration.rt_cycles.enable 1` |
| Parameter | configuration.rt_cycles.enable |
| 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.<br>Use of the parameter “rt_cycles.enable“<br>`rt_cycles.enable 1`<br>(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.<br>The controller must be restarted to adopt the change. Then the specified memory is also available for real-time cycles.<br>Example:<br>`configuration.rt_cycles.memory 60000` |
| Parameter | configuration.rt_cycles.memory |
| Data type | UNS32 |
| Data range | 0 ... MAX(UNS32) - 1 |
| Dimension | ---- |
| Default value | 48000 |
| Remarks | **Note:**<br>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]).<br><br>This parameter is available as of CNC Build V3.1.3107.10 or higher.<br>Use of the parameter “rt_cycles.memory“<br>`rt_cycles.memory 60000`<br>(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.<br>Example:<br>`configuration.decoder.v_ch_memory 10000` |
| Parameter | configuration.decoder.v_ch_memory |
| 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.<br>This parameter is available as of Version V3.1.3107.10.<br>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.<br>`v_ch_memory 10000` |

| P-CHAN-00425 | Max. execution time of realtime cycles per CNC cycle |
|---|---|
| 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.<br>Example:<br>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_UNS16 |
| Dimension | % |
| Default value | 75 |
| Remarks | The user has no restrictions regarding the number of instructions within a realtime cycle.<br>Realtime timeouts will occur if realtime cycles contain too many instructions and cannot be executed in one CNC cycle,<br><br>The parameters P-CHAN-00426 and P-CHAN-00427 represent a safety mechanism to prevent such realtime timeouts as early as possible. |

| **P-CHAN-00426** | **Number of elementary instructions for time check** |
|---|---|
| Description | This parameter defines the number of elementary instructions after which another time check is executed.<br><br>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_UNS32 |
| Dimension | ---- |
| Default value | 100 |
| Remarks | The user has no restrictions regarding the number of instructions within a realtime cycle.<br>Realtime timeouts will occur if realtime cycles contain too many instructions and cannot be executed in one CNC cycle,<br><br>The parameters P-CHAN-00425 and P-CHAN-00427 represent a safety mechanism to prevent such realtime timeouts as early as possible. |

| **P-CHAN-00427** | **Max. number of elementary instructions per CNC cycle** |
|---|---|
| Description | This parameter defines the maximum number of elementary instructions per CNC cycle.<br>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.<br><br>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_UNS32 |
| Dimension | ---- |
| Default value | MAX_UNS32 - 1 |
| Remarks | The user has no restrictions regarding the number of instructions within a realtime cycle.<br>Realtime timeouts will occur if realtime cycles contain too many instructions and cannot be executed in one CNC cycle,<br><br>The parameters P-CHAN-00425 and P-CHAN-00426 represent a safety mechanism to prevent such realtime timeouts as early as possible. |

| P-CHAN-00480 | **Max. number of actions in a real-time cycle** |
|---|---|
| Description | This parameter defines the maximum number of possible actions within a real-time cycle.<br>Possible actions include single-axis movement, spindle command, etc.<br>If too many actions are commanded within a real-time cycle, error ID 51028 is output. |
| 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

| P-AXIS-00457 | **Enable PLCopen interface of a channel axis** | |
|---|---|---|
| Description | 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).<br>The following arrays are supported:<br>• MC_MoveAbsolute<br>• MC_MoveRelative<br>• MC_MoveVelocity<br>• MC_Halt | |
| 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 Appendix

## 7.1 Suggestions, corrections and the latest documentation

Did you find any errors? Do you have any suggestions or constructive criticism? Then please contact us at documentation@isg-stuttgart.de. The latest documentation is posted in our Online Help (DE/EN):

**QR code link:** https://www.isg-stuttgart.de/documentation-kernel/

**The link above forwards you to:**

https://www.isg-stuttgart.de/fileadmin/kernel/kernel-html/index.html

**Notice**

**Change options for favourite links in your browser;**

Technical changes to the website layout concerning folder paths or a change in the HTML framework and therefore the link structure cannot be excluded.

We recommend you to save the above "QR code link" as your primary favourite link.

**PDFs for download:**

DE:
https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

EN:
https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

**E-Mail:** documentation@isg-stuttgart.de

# Keyword index