DOCUMENTATION ISG-kernel

Manual
Configuration list of realtime parameters

Short Description:
RTCF

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de

Documentation version: 1.07
08/11/2024

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.

Table of contents

List of figures

1 Overview of real-time parameters

Attention

This document is not relevant for TwinCAT users.

The real-time parameters required in TwinCAT systems are parameterised in the System Manager.

The overview of real-time parameters is sorted into a 4-column table.

- Column 1 contains the unambiguous identifier of the real-time parameter called the “ID” which consists of the prefix “P-RTCF” and a unique 5-digit number,
  e.g. P-RTCF-00002.
- Column 2 represents the data structure which defines the parameter,
  e.g. thread[i].
  The structure is a categorisation aid and is described in the following section. If an entry is missing in ‘structure’, this is not an error. The parameter in column 3 is then only valid on its own.
- Column 3 contains the “parameter” with its exact name,
  e.g. priority
  The important thing is that “structure”+”parameter” always belong together and must be configured in exactly the same way in the real-time parameters list,
  e.g. thread[i].priority.
- Column 4 contains the “functionality” in a summarised term/short description,
  e.g. priority/thread call sequence.

ID	Structure	Parameter	Functionality/short description
P-RTCF-00001 [▶ 13]		interrupt_source	Interrupt source
P-RTCF-00002 [▶ 13]		cycle_time	NC cycle time
P-RTCF-00003 [▶ 14]		time_slice	Timer interrupt
P-RTCF-00004 [▶ 15]		windows_time	Guaranteed Windows computing time
P-RTCF-00005 [▶ 16]		external_object_name	Name of the external operating system semaphore
P-RTCF-00006 [▶ 24]	external_thread[i].	cycle	Cycle time of an external thread
P-RTCF-00007 [▶ 24]	external_thread[i].	semaphore_name	Name of the semaphore of an external thread
P-RTCF-00008 [▶ 17]	thread[i].	name	Name of the thread
P-RTCF-00009 [▶ 17]	thread[i].	cycle	Cycle time of the thread
P-RTCF-00010 [▶ 18]	thread[i].	priority	Defines the thread call sequence in the scheduler
P-RTCF-00011 [▶ 18]	thread[i].	error_on_overflow	Error message when the related thread overflows
P-RTCF-00012 [▶ 19]	thread[i].function[j].	name	Name of the function within a thread
P-RTCF-00013 [▶ 20]	thread[i].function[j].	calls_per_cycle	Number of calls for a specific function in a thread
P-RTCF-00014 [▶ 20]	thread[i].function[j].	trace_bit	Flag bit of a thread function for diagnostic purposes
P-RTCF-00015 [▶ 21]	thread[i].	cpu	Assigning the thread to a CPU kernel
P-RTCF-00017 [▶ 21]	thread[i].	context	Context information of a thread
P-RTCF-00018 [▶ 22]		schedule	Schedule

2 General description

2.1 Links 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.

2.2 Using rtconf.lis

The RT configuration list is used whenever the Windows operating system is run on a joint platform with a thread-based real-time operating system (such as VxWorks or RTX).

Scheduling with the following features can be parameterised using the parameters **of rtconf.lis**:

- Scalable scheduling for efficient use of CPU performance.
- Fixed specified CPU sharing between Windows and real-time operating system
- Call frequency of each thread per cycle
- Reduction of a thread's call frequency (modulo cycle time).
- Integration of external threads (e.g. of a PLC runtime system)

The following figure shows scheduling with an external interrupt and an internal timer interrupt for control of the individual threads' computing time.

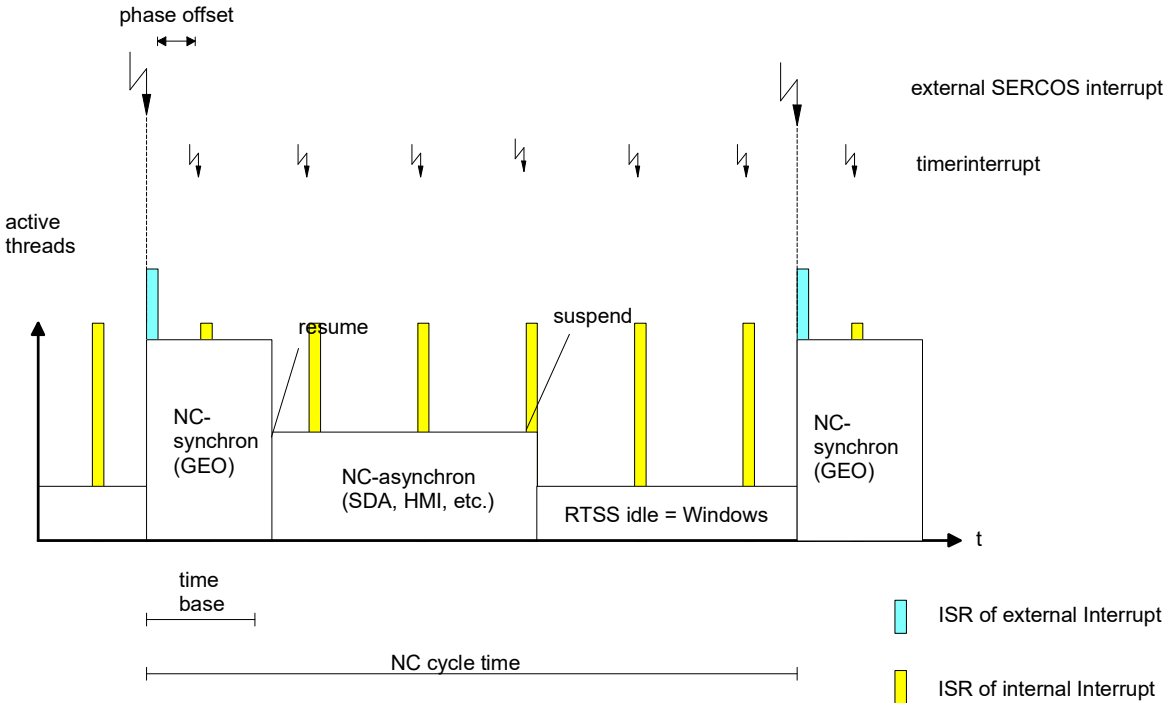

Fig. 1: Scheduling the CNC thread and an external thread

After the external interrupt, all threads participating in scheduling are set to the ready for execution state. Then, the order of execution of the threads depends solely on their priority. This is also parameterised in the RT configuration list. As a rule, the synchronous NC execution part (GEO) has the highest priority and is therefore executed first. This is followed by the asynchronous thread, consisting for example of the function calls for control data processing (SDA), HMI and system sequential control.

The functions of the asynchronous thread are called in a loop the same number of times as parameterised in the RT configuration list. There is, however, an upper limit for the execution time of the asynchronous thread. This is monitored by higher resolution timer interrupts. In other words, the asynchronous thread is stopped as required so that Windows can receive the parameterised computing time (also in rtconf.lis).

2.3 Structure and breakdown of the RT configuration data

The parameters of the **R**eal-**T**ime-**Conf**iguration **Lis**t (**rtconf.lis**) are listed hierarchically. They consist of:

Breakdown criteria	**Contents**
General parameters	Basic parameters for overall scheduling
Thread-specific parameters	Details of parameterisation per thread
Parameters per function within a thread	Details of parameterisation of the function calls within a thread
External threads	Parameters for external threads

Within the software, the parameters of **rtconf.lis** are defined in the following C structures:

Structure	**Definition in the file:**	**String in the ASCII file:**	**Contents**
RTCONF_DATA	rt_conf.inc	-	Parameters for scheduling
CNC_THREAD_ PROPERTIES	rt_conf.inc	thread[i]	Thread-specific parameters
CNC_FUNCTION_ PROPERTIES	rt_conf.inc	thread[i].function[j]	Function-specific parameters per thread
CNC_EXTERNAL_ THREAD_PROPERTIES	rt_conf.inc	external_thread[i]	Specific parameters for external thread

2.4 Syntax and interpretation of ASCII list file

An interpreter copies the entries in the ASCII list file into identical internal structures which are then checked for plausibility. To ensure reliable controller start-up every time, any defective entries found by the plausibility check are replaced by default values.

Unknown entries are not taken over. These irregularities are displayed by warning messages. We advise you to investigate the cause for these warning messages and remove defective entries from the ASCII list file.

Notice

The following agreement applies to BOOLEAN data:

Value	Meaning
0	Definition of FALSE
1	Definition of TRUE

Notice

The following agreement applies to STRING data:

If a character string containing characters with a special meaning in ASCII lists (e.g. comment characters, spaces [▶ 11]) is assigned to a STRING type list parameter, this character string must be defined in inverted commas "..." (available as of V3.1.3081.0, V3.1.3108.0).

```
example[0].name "STRING_WITH_COMMENT( # /*)_CHARACTERS"
```

Trailing spaces are discarded on import. The entry..

```
example[0].name "STRING_WITH_POST_SPACES "
```

..has the same meaning as

```
example[0].name "STRING_WITH_POST_SPACES"
```

If the character string only contains characters without any special meaning, no inverted commas are required.

```
example[0].name STRING_WITH_STANDARD_CHARACTERS!
```

2.5 Comments in the ASCII list file

Comments can be in an entire line or can be added at the end of a line.

With a comment spanning an entire line, the comment character "'#" must be placed at the start of the line and followed by a blank.

If a comment is to be inserted at the end of a line, only a blank is required before the comment. Blank lines are also possible.

Example

Comments in the ASCII list file

```
#
*****************************************************************************
# Data
#
*****************************************************************************
#
# List comments after numerical values

dummy[1] 1 Comment
dummy[2] 1 # Comment
dummy[3] 1 ( Comment
dummy[4] 1 /* Comment
...
...
```

However, if a character string was assigned to the list parameter as a value in the line, any following comment must be opened by a bracket '('. The comment characters space, # and /* are not permitted.

If a '(' itself is part of the character string, the character string must be defined in inverted commas "..." (available as of V3.1.3081.0, V3.1.3108.0).

```
# List comments after strings

beispiel[0].bezeichnung STRING_1 (comment requires a '(' bracket!)

beispiel[1].bezeichnung" STRING_(2)" (comment requires a '(' bracket!)
```

3 Description of the basic parameters

The basic parameters for overall scheduling are described in this section. The parameters are also shown in the following figure:

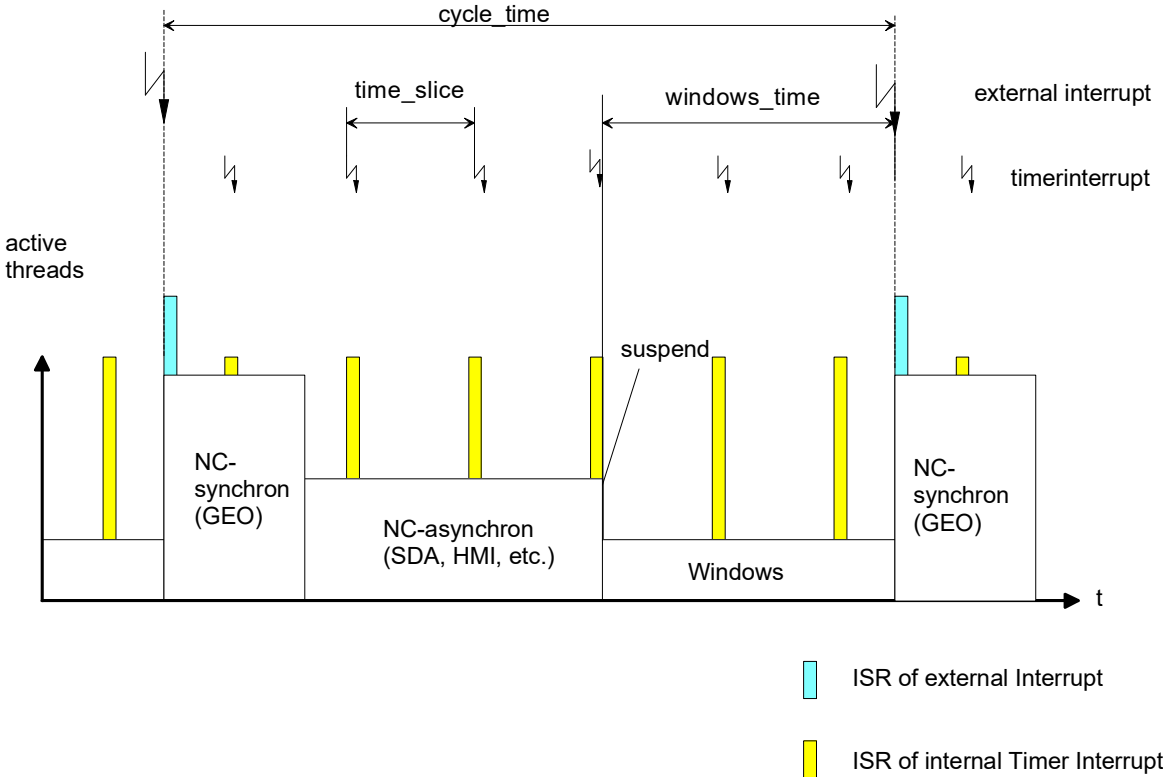

Fig. 2: Basic parameters of scheduling

3.1 Interrupt source (P-RTCF-00001)

P-RTCF-00001	Source of the interrupt that triggers the start of a new NC cycle.
Description	A distinction is made between an external interrupt (e.g. from SERCOS hardware) and an internal interrupt (timer interrupt of the real-time operating system). If an external interrupt is available, execution of the NC-synchronous thread is coupled directly to it. The operating system's higher-resolution timer interrupt is then merely needed to possibly stop the asynchronous thread. The NC cycle time and timer interrupt do not run synchronously. This also means that suspension of the asynchronous thread can vary by the amount of P-RTCF-00003 [▶ 14] (time_slice). If no external interrupt is available, only a high-resolution timer interrupt of the operating system is used to start the NC-synchronous thread (with every nth timer interrupt). The NC cycle time and timer interrupt then run synchronously. When an external operating system semaphore is used, the semaphore's name is defined by P-RTCF-00005 [▶ 16] (external_object_name).
Parameter	interrupt_source
Data type	UNS16
Data range	1: Hardware ISR 2: Real-time operating system timer 3: Real-time operating system semaphore
Dimension	----
Default value	0
Remarks	

3.2 NC cycle time (P-RTCF-00002)

P-RTCF-00002	NC cycle time
Description	The NC cycle time must always be an integer multiple of the cycle time of the real-time operating system timer interrupt P-RTCF-00003 [▶ 14] (time_slice). If this is not the case, it is rounded up to the next higher integer multiple.
Parameter	cycle_time
Data type	UNS32
Data range	0 < cycle_time < MAX(UNS32)
Dimension	µs
Default value	-
Remarks	

3.3 Timer-Interrupt (P-RTCF-00003)

P-RTCF-00003	**Timer-interrupt time distance of the real-time operating system**
Description	The timer-interrupt time distance of the real-time operating system should be clearly less than the NC cycle time (e.g. by a factor of 3 to 20) However, an error is generated only if the ratio is less than 1. The internal time can be set with VxWorks and RTX operating systems.
Parameter	time_slice
Data type	UNS32
Data range	0 < time_slice < MAX(UNS32)
Dimension	µs
Default value	-
Remarks	The following applies to RTX: in the Initialisation menu (RTX Properties Register Setting) of RTX, the HAL timer period must match the value of *time_slice*, otherwise an error message is output.

3.4 Guaranteed Windows computing time (P-RTCF-00004)

P-RTCF-00004	**Guaranteed Windows computing time within one NC cycle**
Description	The windows_time must be an integer multiple of the time_slice. If this is not the case, it is rounded down to the next lower integer multiple. If the predecessor thread is prematurely complete, the remaining time period up to the end of the cycle time is made available to Windows as computing time. If the guaranteed computing time of Windows is truncated by the predecessor thread, it is stopped by the timer ISR. Furthermore, the selected windows_time must not be so large, otherwise the threads that can be stopped by Windows (all threads except the GEO task) will not receive sufficient computing time (or none at all). If windows_time is selected smaller then P-RTCF-00003 [▶ 14] (time_slice), an error message is output.
Parameter	windows_time
Data type	UNS16
Data range	0 < windows_time < MAX(UNS16)
Dimension	µs
Default value	-
Remarks	In the case of *windows_time* = 0, suspension of the predecessor process is dispensed with completely. As a result Windows no longer receives any guaranteed computing time. Such a parameter definition only makes sense if steps are taken to ensure that all threads regularly become "ready" before expiry of the NC cycle time. This presumes that all threads behave cooperatively. Suitable setting of the *windows_time* in relation to *time_slice* (P-RTCF-00003 [▶ 14]): **windows_time = time_slice = 500µs** Shorter time periods (e.g. 100 - 200µs) are less favourable because a context change between Windows and the real-time operating system is relatively time-intensive. Choosing a *windows_time* that is too short (e.g. 100µs) causes instability in real-time operation (RTX problem)

3.5 Name of the external operating system semaphore (P-RTCF-00005)

P-RTCF-00005	Name of the external operating system semaphore
Description	The parameter defines the name of the semaphore for P-RTCF-00001 [▶ 13] (interrupt source).
Parameter	external_object_name
Data type	STRING
Data range	Maximum name length dependent on operating system
Dimension	----
Default value	*
Remarks	* Note: The default value of variables is a blank string. This parameter is used only if *interrupt_source = 3*.

4 Thread-specific parameters (thread[i].*)

Structure name	Index
thread[i]	i = 0 ... 6 (maximum number of threads 7, application-specific)

4.1 Name of thread (P-RTCF-00008)

P-RTCF-00008	Name of the thread in the scheduler
Description	The parameter defines the name of the thread in the scheduler.
Parameter	thread[i].name
Data type	STRING
Data range	Maximum of 127 characters
Dimension	----
Default value	*
Remarks	* Note: The default value of variables is a blank string. In principle the name is freely selectable, e.g. GEO, BACKGROUND, MMI_DRIVER, COM

4.2 Cycle time of thread (P-RTCF-00009)

P-RTCF-00009	Cycle time of the thread
Description	This parameter defines whether the scheduler calls the thread in every interval or only in every nth interval. This is why the chosen value of the parameter must be an integer multiple of the cycle time. If this is not the case, the value is rounded up to the next integer multiple of the NC cycle time.
Parameter	thread[i].cycle
Data type	UNS32
Data range	n * cycle_time (P-RTCF-00002 [▶ 13])
Dimension	----
Default value	0
Remarks	

4.3 Call sequence of the thread (P-RTCF-00010)

P-RTCF-00010	**Defines the thread call sequence in the scheduler**
Description	This parameter defines the call sequence of the thread in the scheduler.
Parameter	thread[i].priority
Data type	SGN32
Data range	Dependent on the operating system: In the case of RTX: 0 … 127 (127 = highest) In the case of VxWorks: 255 ... 0 (0 = highest)
Dimension	----
Default value	0
Remarks	It must be mentioned once again here that the thread priorities define the sequence of execution of the individual threads. **The following applies to RTX:** no priorities > 124 may be parameterised in the case of RTX because the scheduler itself (the ISR threads) and the shutdown handler run in this priority range. Typical priority selection: • GEO 115 • BACKGROUND 105 • MMI_DRIVER 95 • COM 85

4.4 Error message on thread overflow (P-RTCF-00011)

P-RTCF-00011	**Error message when the related thread overflows**
Description	The parameter defines whether an error message is output when the related thread overflows.
Parameter	thread[i].error_on_overflow
Data type	BOOLEAN
Data range	0: No error message output 1: Error message output on thread overflow
Dimension	----
Default value	0
Remarks	

4.5 Functions in a thread (thread[i].function[j].*)

Every thread can consist of functions which are parameterised by the variables explained here.

Structure name	index
thread[i].function[j]	j = 0 ... 6 (maximum number of thread functions 7, application-specific)

4.5.1 Name of a thread function (P-RTCF-00012)

P-RTCF-00012	Name of the function within a thread
Description	The parameter defines the name of a function within the thread.
Parameter	thread[i].function[j].name
Data type	STRING
Data range	Maximum of 127 characters The names to be used are already permanently compiled in the source. The following parameter assignments are allowed: • task_rnd: Rotating thread with FB SDA • task_int: Interrupt thread with FB GEO • task_com: Communication thread with FB COM • task_mmi_driver: System sequential control
Dimension	----
Default value	*
Remarks	* Note: The default value of variables is a blank string. Parameterisation example: *thread[0].name* *GEO1* **thread[0].context_info** **1** *thread[0].cycle* *2000* *thread[0].priority* *31 # HIGHEST* *thread[0].error_on_overflow* *1* *thread[0].function[0].name* *task_int* *thread[0].function[0].calls_per_cycle* *1*

4.5.2 Number of calls of a thread function (P-RTCF-00013)

P-RTCF-00013	**Number of calls for a specific function in a thread.**
Description	The parameter defines the number of calls for a specific function in a thread.
Parameter	thread[i].function[j].calls_per_cycle
Data type	UNS32
Data range	0 < P-RTFC-00013 < MAX(UNS32)
Dimension	----
Default value	1 *
Remarks	* In general *calls_per_cycle* = 1 is set. In specific applications (e.g. in the HSC area), it may be practical to call a rotating part multiple times (BACKGROUND). The parameter must then be adapted to the specific needs. Parameterisation example: *thread[0].name GEO1* **thread[0].context_info 1** *thread[0].cycle 2000* *thread[0].priority 31 # HIGHEST* *thread[0].error_on_overflow 1* *thread[0].function[0].name task_int* *thread[0].function[0].calls_per_cycle 1*

4.5.3 Flag bit of a thread function (P-RTCF-00014)

P-RTCF-00014	**Flag bit of a thread function for diagnostic purposes**
Description	For diagnosis with an oscilloscope, the start and end of a thread function can be displayed by setting a bit on the parallel interface. The timing of the scheduler can then be recorded.
Parameter	thread[i].function[j].trace_bit
Data type	UNS16
Data range	0 ... 7 0: No flag bit set
Dimension	----
Default value	0
Remarks	Parameterisation example: *thread[0].name GEO1* thread[0].context_info 1 *thread[0].cycle 2000* *thread[0].priority 31 # HIGHEST* *thread[0].error_on_overflow 1* *thread[0].function[0].name task_int* *thread[0].function[0].calls_per_cycle 1* **thread[0].function[0].trace_bit 1**

4.5.4 Assigning the thread to a CPU kernel (P-RTCF-00015)

P-RTCF-00015	Assigning the thread to a CPU kernel
Description	The parameter can assign the thread to a specific CPU kernel. The thread is then executed only on this kernel. If the thread is assigned to a kernel that does not exist, error ID 1000180 is output. The operating system then assumes the task of assigning the thread to one or several processor kernels. If the assignment cannot be executed for one reason or another, an error message with ID 1000181 is output.
Parameter	thread[i].cpu
Data type	UNS16
Data range	1 ... 7
Dimension	----
Default value	1
Remarks	Parameterisation example: *thread[0].cpu 1*

4.6 Context information of a thread (P-RTCF-00017)

P-RTCF-00017	Context information of a thread
Description	When this parameter is specified, the channel can be assigned to a thread (CPU). In order to assign the GEO, SDA or COM task of a channel to the same context, the values of the channel parameter entered in P-CHAN-00410, P-CHAN-00411 and P-CHAN-00409 must be set accordingly.
Parameter	thread[i].context_info
Data type	UNS32
Data range	0 <= thread[i].context_info
Dimension	----
Default value	0
Remarks	Parameterisation example *thread[0].name GEO1* **thread[0].context_info 1** *thread[0].cycle 2000* *thread[0].priority 31 # HIGHEST* *thread[0].error_on_overflow 1* *thread[0].function[0].name task_int* *thread[0].function[0].calls_per_cycle 1* To assign the GEO TASK (P-CHAN-00410) based on the above parameters: **schedule.context.geo 1**

4.7 Schedule (P-RTCF-00018)

P-RTCF-00018	Schedule
Description	Defines the order in which the actual axis values are read in. Shows the output in which axis command values and interpolation are processed, see CNC scheduler.
Parameter	schedule
Data type	SGN32
Data range	STANDARD COMLETE SWITCHED
Dimension	----
Default value	STANDARD
Remarks	Parameterisation example: *schedule STANDARD*

5 External threads

5.1 General description and overview

Named semaphores are set up to synchronise the NC with one or more external thread(s). In each cycle, these semaphores are issued once, as a result of which the external threads waiting for them are woken up and they can do their work. External threads are integrated into overall scheduling depending on their priority.

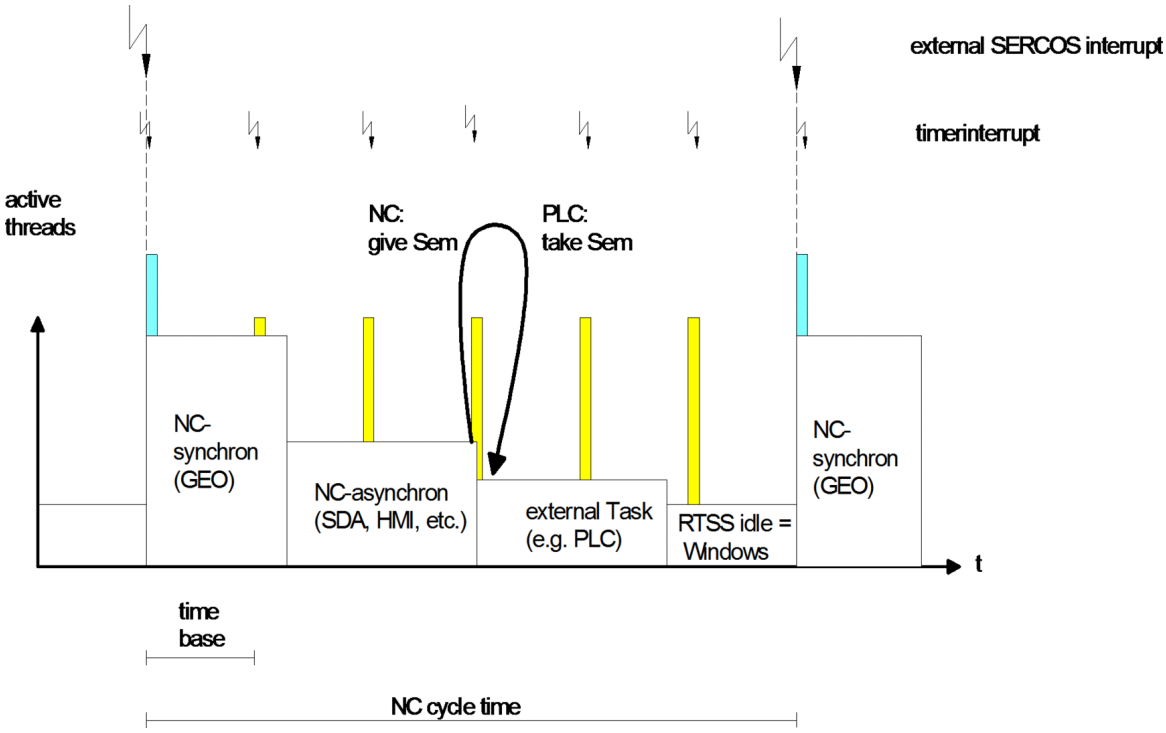

Fig. 3: Synchronisation with an external thread

As threads cannot be stopped across processes, external threads must behave cooperatively. Depending on the external thread's priority, its runtime is to the detriment of the asynchronous NC thread or to the detriment of the guaranteed Windows time.

5.2 Specific parameters of external threads (external_thread[i].*)

Structure name	Index
external_thread[i]	i = 0 ... 3 (maximum number of external threads 4, application-specific)

5.2.1 Cycle time of an external thread (P-RTCF-00006)

P-RTCF-00006	Cycle time of an external thread
Description	This parameter defines whether the scheduler calls the external thread in every interval or only in every nth interval. This is why the chosen value of the parameter must be an integer multiple of the cycle time. If this is not the case, the value is rounded up to the next integer multiple of the NC cycle time.
Parameter	external_thread[i].cycle
Data type	UNS32
Data range	n * cycle_time (P-RTCF-00002 [▶ 13])
Dimension	----
Default value	0
Remarks	Parameterisation example: *external_thread[0].cycle 2000*

5.2.2 Name of the semaphore of an external thread (P-RTCF-00007)

P-RTCF-00007	Name of the semaphore of an external thread
Description	The parameter defines the name of the semaphore of an external thread
Parameter	external_thread[i].semaphore_name
Data type	STRING
Data range	Maximum of 127 characters
Dimension	----
Default value	*
Remarks	* Note: The default value of variables is a blank string. Parameterisation example: *external_thread[0].semaphore_name external_sps*

5.2.3 Example of a Rtconf.lis for a SERCOS application

```
# ***************************************
#
# ***************************************
# RTX [0; 127] 127 highest
# VXWORKS [0; 255] 0 highest
# INTIME [0; 255] 0 highest
#
interrupt_source 2 # 1: internal timer, 2: IRQ, 3: Semaphore
cycle_time 2000 # Cycle time in micro s
time_slice 500
windows_time 500 # 0 turned off
#
thread[0].name GEO
thread[0].cycle 2000
thread[0].priority 115 # HIGHEST
thread[0].error_on_overflow 1
thread[0].function[0].name task_int
thread[0].function[0].calls_per_cycle 1
thread[0].function[0].trace_bit 1
#
thread[1].name BACKGROUND_
thread[1].cycle 2000
thread[1].priority 105 # NORMAL
thread[1].error_on_overflow 0
thread[1].function[0].name task_rnd
thread[1].function[0].calls_per_cycle 1
thread[1].function[0].trace_bit 2
#
thread[2].name MMI_DRIVER
thread[2].cycle 6000
thread[2].priority 95 # LOWEST
thread[2].error_on_overflow 0
thread[2].function[0].name task_mmi_driver
thread[2].function[0].calls_per_cycle 1
thread[2].function[0].trace_bit 0
#
thread[3].name COM
thread[3].cycle 4000
thread[3].priority 90
thread[3].error_on_overflow 0
thread[3].function[0].name task_com
thread[3].function[0].calls_per_cycle 1
thread[3].function[0].trace_bit 4
#
external_thread[0].semaphore_name external_sps
external_thread[0].cycle 2000
#
End
```

Alternatively, the communication can also be called together with the rotating part. The rtconf.lis then looks like this:

```
# ***************************************
#
# ***************************************
# RTX [0; 127] 127 highest
# VXWORKS [0; 255] 0 highest
# INTIME [0; 255] 0 highest
#
interrupt_source 2 # 1 == internal timer, 2 == external
cycle_time 2000 # Cycle time in micro s
time_slice 500
windows_time 500 # 0 turned off
#
thread[0].name GEO
thread[0].cycle 2000
thread[0].priority 115 # HIGHEST
thread[0].error_on_overflow 1
thread[0].function[0].name task_int
thread[0].function[0].calls_per_cycle 1
thread[0].function[0].trace_bit 1
#
thread[1].name BACKGROUND_
thread[1].cycle 2000
thread[1].priority 105 # NORMAL
thread[1].error_on_overflow 0
thread[1].function[0].name task_rnd
thread[1].function[0].calls_per_cycle 1
thread[1].function[0].trace_bit 2
thread[1].function[1].name task_com
thread[1].function[1].calls_per_cycle 1
thread[1].function[1].trace_bit 4
#
thread[2].name MMI_DRIVER
thread[2].cycle 6000
thread[2].priority 95 # LOWEST
thread[2].error_on_overflow 0
thread[2].function[0].name task_mmi_driver
thread[2].function[0].calls_per_cycle 1
thread[2].function[0].trace_bit 0
# task_com repositioned to thread[1]:
# thread[3].name COM
# thread[3].cycle 4000
# thread[3].priority 90
# thread[3].error_on_overflow 0
# thread[3].function[0].name task_com
# thread[3].function[0].calls_per_cycle 1
# thread[3].function[0].trace_bit 4
#
external_thread[0].semaphore_name external_sps
external_thread[0].cycle 2000
#
End
```

6 Appendix

6.1 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

Index

P

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de