

RTI Watchdog Blockset

# Reference

For RTI Watchdog Blockset 2.1.3

Release 2021-A – May 2021

## How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	<a href="mailto:info@dspace.de">info@dspace.de</a>
Web:	<a href="http://www.dspace.com">http://www.dspace.com</a>

## How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: <http://www.dspace.com/go/locations>
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.  
Tel.: +49 5251 1638-941 or e-mail: [support@dspace.de](mailto:support@dspace.de)

You can also use the support request form: <http://www.dspace.com/go/supportrequest>. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

## Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit <http://www.dspace.com/go/patches> for software updates and patches.

## Important Notice

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2013 - 2021 by:  
dSPACE GmbH  
Rathenaustraße 26  
33102 Paderborn  
Germany

This publication and the contents hereof are subject to change without notice.

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

# Contents

About This Reference	5
General Information on the RTI Watchdog Blockset	9
Overview of the RTI Watchdog Blockset.....	9
Components of the RTI Watchdog Blockset	17
Watchdog Handling.....	18
WATCHDOG_SETUP.....	18
Block Description (WATCHDOG_SETUP).....	19
Parameters Page (WATCHDOG_SETUP).....	20
WATCHDOG_RETRIGGER_BLx.....	21
Block Description (WATCHDOG_RETRIGGER_BLx).....	21
Parameters Page (WATCHDOG_RETRIGGER_BLx).....	23
WATCHDOG_INTERRUPT.....	23
Block Description (WATCHDOG_INTERRUPT).....	24
Unit Page (WATCHDOG_INTERRUPT).....	25
WATCHDOG_EXPIRED.....	26
Block Description (WATCHDOG_EXPIRED).....	26
Challenge-Response Monitoring.....	28
CRM_UNIT_SETUP_BL.....	28
Block Description (CRM_UNIT_SETUP_BL).....	29
Unit Page (CRM_UNIT_SETUP_BL).....	31
Parameters Page (CRM_UNIT_SETUP_BL).....	32
Challenge-Response Page (CRM_UNIT_SETUP_BL).....	33
Advanced Page (CRM_UNIT_SETUP_BL).....	34
CRM_RESPONSE_BL.....	35
Block Description (CRM_RESPONSE_BL).....	35
Unit Page (CRM_RESPONSE_BL).....	36
CRM_SUPERVISION_STATUS.....	37
Block Description (CRM_SUPERVISION_STATUS).....	37

CRM_INTERRUPT.....	38
Block Description (CRM_INTERRUPT).....	39
Unit Page (CRM_INTERRUPT_BL).....	40
CRM_INTERRUPT_DECODE.....	40
Block Description (CRM_INTERRUPT_DECODE).....	41
Memory Integrity And Extras.....	43
MEMORY_VBAT_MONITORING.....	43
Block Description (MEMORY_VBAT_MONITORING).....	44
Unit Page (MEMORY_VBAT_MONITORING).....	47
MEMORY_VBAT_INTERRUPT.....	48
Block Description (MEMORY_VBAT_INTERRUPT).....	49
Unit Page (MEMORY_VBAT_INTERRUPT).....	50
CUSTOM_FAILURE_ACTION.....	50
Block Description (CUSTOM_FAILURE_ACTION).....	51
Unit Page (CUSTOM_FAILURE_ACTION).....	52
CUSTOM_FAILURE_INTERRUPT.....	54
Block Description (CUSTOM_FAILURE_INTERRUPT).....	54
Unit Page (CUSTOM_FAILURE_INTERRUPT).....	55

Index	57
-------	----

# About This Reference

## Content





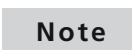



This RTI Reference is a complete description of the Real-Time Interface (RTI) blocks and their settings provided by the RTI Watchdog Blockset.

You can use this blockset to implement safety functions in your Simulink model.

It is supported by MicroAutoBox II.

## Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
	Indicates a hazard that, if not avoided, could result in property damage.
	Indicates important information that you should take into account to avoid malfunctions.
	Indicates tips that can make your work easier.
	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
	Precedes the document title in a link that refers to another document.

## Naming conventions

dSPACE user documentation uses the following naming conventions:

**%name%** Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Examples:

- Where you find terms such as `rti<XXXX>` replace them by the RTI platform support you are using, for example, `rti1007`.
- Where you find terms such as `<model>` or `<submodel>` in this document, replace them by the actual name of your model or submodel. For example, if the name of your Simulink model is `smd_1007_s1.slx` and you are asked to edit the `<model>_usr.c` file, you actually have to edit the `smd_1007_s1_usr.c` file.

**RTI block name conventions** All I/O blocks have default names based on dSPACE's board naming conventions:

- Most RTI block names start with the board name.
- A short description of functionality is added.
- Most RTI block names also have a suffix.

Suffix	Meaning
B	Board number (for PHS-bus-based systems)
M	Module number (for MicroAutoBox II)
C	Channel number
G	Group number
CON	Converter number
BL	Block number
P	Port number
I	Interrupt number

A suffix is followed by the appropriate number. For example, `DS2201IN_B2_C14` represents a digital input block located on a DS2201 board. The suffix indicates board number 2 and channel number 14 of the block. For more general block naming, the numbers are replaced by variables (for example, `DS2201IN_Bx_Cy`).

## Special folders

Some software products use the following special folders:

**Common Program Data folder** A standard folder for application-specific configuration data that is used by all users.

`%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>`

or

%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>

**Documents folder** A standard folder for user-specific documents.

%USERPROFILE%\Documents\dSPACE\<ProductName>\  
<VersionNumber>

**Local Program Data folder** A standard folder for application-specific configuration data that is used by the current, non-roaming user.

%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\  
<ProductName>

---

## Accessing dSPACE Help and PDF Files


After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

**dSPACE Help (local)** You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

**dSPACE Help (Web)** You can access the Web version of dSPACE Help at [www.dspace.com](http://www.dspace.com).

To access the Web version, you must have a *mydSPACE* account.

**PDF files** You can access PDF files via the  icon in dSPACE Help. The PDF opens on the first page.





# General Information on the RTI Watchdog Blockset

---

<b>Introduction</b>	Provides basic information on the RTI Watchdog Blockset.
---------------------	--

## Overview of the RTI Watchdog Blockset

---

<b>Introduction</b>	To provide a short description of the blockset's components and how to access them.
---------------------	---

---

<b>Main features</b>	<p>The blockset provides two mechanisms that allow a multistage run-time monitoring of tasks in your Simulink model and the built real-time application.</p> <p>The first safety mechanism is available in the <b>Watchdog</b> blockset. A watchdog monitors a task that must retrigger the watchdog's timer within the specified timeout limit. Otherwise, the specified failure action is executed, which usually results in terminating the real-time application.</p>
----------------------	---

**Note**

The watchdog feature is not designed to be used for safety-critical applications.

The second safety mechanism is available in the **Challenge-Response Monitoring** blockset. With the challenge-response monitoring feature, the monitored entity has to respond within the specified timeout limit with a value that corresponds to the current challenge value. Otherwise, the specified failure action is executed, which usually results in terminating the real-time application.

The **Memory Integrity and Extras** blockset allows you to monitor the supply voltage of the real-time hardware and the memory integrity of the real-time application. These monitoring facilities are available as predefined features. For any other failure detection, you can specify a custom failure action.

For more information, refer to [Safety Functions \(MicroAutoBox II Features !\[\]\(529949c2c3dadbaa4e538e8c643454bc\_img.jpg\)](#)).

---

## Supported hardware

The RTI Watchdog blockset can be used with:

- MicroAutoBox II (any variant)
  - The Watchdog blockset requires System PLD firmware version 1.4.0 or later.
  - The Challenge-Response Monitoring blockset requires System PLD firmware version 1.5.0 or later.
  - The Memory Integrity and Extras blockset requires System PLD firmware version 1.6.0 or later.

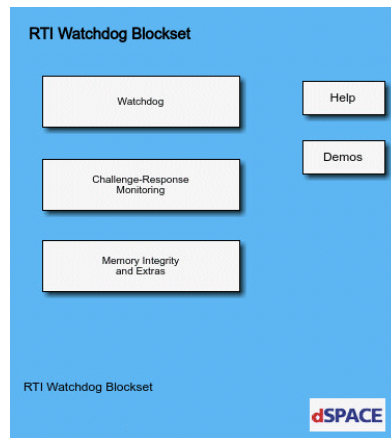
---

## Library access

The library can be opened with one of the following methods:

- Enter `rtiwatchdog` in the MATLAB Command Window.
- Navigate to the dSPACE RTI Watchdog Blockset folder in the Simulink Library Browser to access the RTI blocks of the library separately.
- Click the Watchdog button in the Base Board II blockset of a MicroAutoBox II variant.
- Click Blocksets - WATCHDOG Blockset in the MicroAutoBox II blockset.

When you open the block library, the blockset is displayed.



---

## Library components

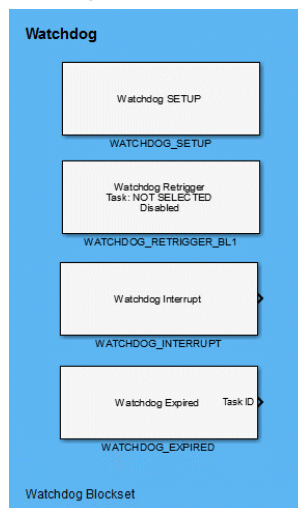
The library provides the following components:

- Watchdog  
Provides blocks for controlling safety functions via standard watchdog handling. Refer to [Main features of watchdog handling](#) on page 11.
- Challenge-Response Monitoring  
Provides blocks for controlling safety functions via the challenge-response monitoring. Refer to [Main features of challenge-response monitoring](#) on page 13.

- **Memory Integrity and Extras**  
Provides blocks for monitoring the supply voltage of your hardware and the memory integrity of your real-time application. Refer to [Main features of memory integrity and extras](#) on page 14.
- **Help**  
Opens dSPACE Help providing the user documentation for this blockset.
- **Demos**  
Provides access to the demos for this blockset.

## Main features of watchdog handling

The Watchdog blockset provides blocks for implementing standard watchdog handling.



- [WATCHDOG\\_SETUP](#) on page 18  
To set up the watchdog handling.
- [WATCHDOG\\_RETRIGGER\\_BLx](#) on page 21  
To retrigger a monitored task.
- [WATCHDOG\\_EXPIRED](#) on page 26  
To get the ID of an expired task.
- [WATCHDOG\\_INTERRUPT](#) on page 23  
To trigger a subsystem if a watchdog generates an interrupt.

**Specifying watchdogs** Initializing the watchdog feature enables a hardware watchdog that monitors the reactivity of your real-time application as a whole. If any task in your real-time application hangs, the background service of your application will not service the hardware watchdog. This leads to a configured reaction of the hardware watchdog, for example, it will reset the hardware system.

Based on the hardware watchdog, you can add software watchdogs to monitor specific tasks in your real-time application. It is guaranteed that you can initialize up to 20 software watchdogs. Whether you can add further software watchdogs depends on the available memory.

For each watchdog you can individually configure its timeout limit. It makes sense to set the timeout limit of the hardware watchdog to the lowest timeout limit of all the tasks registered with the watchdog feature. This guarantees that the requirements of the task with the shortest timeout interval are met. However, any running task might then prevent the background service from executing. If a task has an execution time greater than the timeout limit of the background task, the hardware watchdog resets the system or generates an interrupt.

**Retriggering watchdog timers** The timer of the hardware watchdog is automatically retriggered by the background service of your real-time application. The timers of the software watchdogs have to be retriggered explicitly within the tasks by using the `WATCHDOG_RETRIGGER_BLx` block. If retriggering was not executed within the specified timeout limit, the watchdog reacts by generating a machine check (MCP) interrupt to start a previously specified subsystem or by generating a hardware reset (HRESET) pulse to restart the entire system or by both. For example, you can use the subsystem to store data or to set the outputs to their termination states.

#### CAUTION

##### **Risk of personal injury**

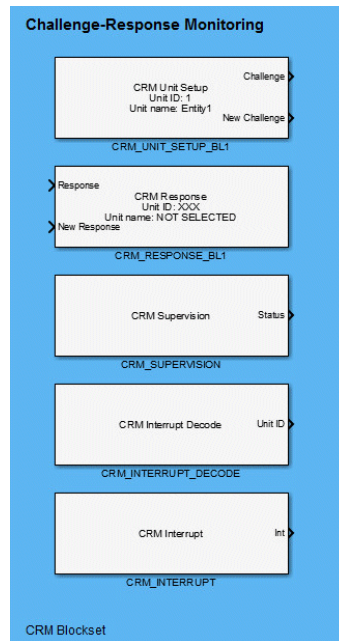
It is not guaranteed that the subsystem is executed. There might be a fatal system crash that triggers the generation of an interrupt but prevents the subsystem from being executed.

**Stopping task monitoring** The `WATCHDOG_SETUP` block is able to start and stop task monitoring as a reaction to simulation state changes, represented by the value of the *simState* variable in your model.

For more information, refer to [Basics on Watchdog Handling \(MicroAutoBox II Features !\[\]\(ec9132f1d27c8919987d92907322654d\_img.jpg\)](#)).

## Main features of challenge-response monitoring

The Challenge-Response Monitoring blockset provides blocks for implementing challenge-response monitoring.



- [CRM\\_UNIT\\_SETUP\\_BL](#) on page 28  
To initialize and configure a challenge-response monitoring unit.
- [CRM\\_SUPERVISION\\_STATUS](#) on page 37  
To provide status information on the supervision of the monitoring hardware.
- [CRM\\_RESPONSE\\_BL](#) on page 35  
To return a response to the corresponding challenge-response monitoring unit.
- [CRM\\_INTERRUPT](#) on page 38  
To trigger a subsystem if a CRM interrupts occurs.
- [CRM\\_INTERRUPT\\_DECODE](#) on page 40  
To determine the monitoring unit that caused the CRM hardware to generate the MCP interrupt.

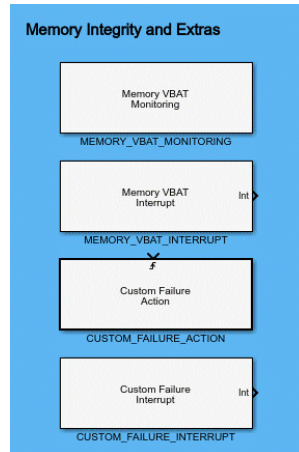
**Specifying challenge-response monitoring units** The hardware component of a monitoring unit is not used by the real-time application. The execution states of the real-time application and its monitoring are decoupled. If you initialize the first monitoring unit, a self-test unit is automatically instantiated and started. The status of the self-test can be requested using the `CRM_SUPERVISION_STATUS` block. A challenge-response monitor unit can control not only tasks or subsystems but any kind of entity in your model. If the monitored entity does not react at all, not in the specified time interval, or with an invalid response value, a failure counter is incremented. If the specified failure count limit is exceeded, the specified failure action is activated. Usually, an interrupt is generated to trigger a subsystem before the real-time application is exited, but you can also specify to directly reset the hardware or only to trigger an output port.

You can use up to 14 monitor units in your model, which you can configure individually.

For more information, refer to [Basics on Challenge-Response Monitoring \(MicroAutoBox II Features !\[\]\(5eb1325dfdc3f1cad8426726c0db51cd\_img.jpg\)\)](#).

## Main features of memory integrity and extras

The Memory Integrity and Extras blockset provides blocks for implementing memory and VBAT monitoring as well as customizable failure actions.



- [MEMORY\\_VBAT\\_MONITORING](#) on page 43  
To configure memory and VBAT monitoring.
- [MEMORY\\_VBAT\\_INTERRUPT](#) on page 48  
To trigger a subsystem if a memory or VBAT interrupt occurs.
- [CUSTOM\\_FAILURE\\_ACTION](#) on page 50  
To configure a custom failure action.
- [CUSTOM\\_FAILURE\\_INTERRUPT](#) on page 54  
To trigger a subsystem if a custom failure interrupt occurs.

**Specifying memory monitoring** The memory monitoring provides different facilities. With the ROM check, the integrity of the read-only data of the real-time application in the memory is checked once when you download the real-time application to the hardware. If the check fails, the real-time application terminates before its initialization phase. With the ROM monitoring, heap monitoring, and stack monitoring, the integrity of these memory sections is periodically checked in the background loop of your model. If you enabled the ROM check, the ROM monitoring, or the heap monitoring, the PPC file of your real-time application is automatically patched with special codes, such as checksum values.

If a memory check fails, the specified failure action is executed. Usually, an interrupt is generated to trigger a subsystem before the real-time application is exited, but you can also specify to directly reset the hardware, to generate an interrupt followed by a hardware reset, or to trigger only an output port.

You can use only one `MEMORY_VBAT_MONITORING` block in your model.

**Specifying VBAT monitoring** With the VBAT monitoring, you can check the voltage level of the power supply of your real-time hardware. If the voltage level falls below the specified voltage threshold, the specified failure action is executed. Usually, an interrupt is generated to trigger a subsystem before the real-time application is exited, but you can also specify to directly reset the hardware, to generate an interrupt followed by a hardware reset, or to trigger only an output port.

For the VBAT monitoring, you use the same RTI block as for the memory monitoring facilities.

**Specifying custom failure action** The failure actions that you know from the VBAT monitoring and the memory monitoring facilities can also be used with a check function implemented by you. For example, you can periodically read the value of a model parameter and trigger a failure action if the current value is out of a specified range.

You can use only one `CUSTOM_FAILURE_ACTION` block in your model.

For more information, refer to [Basics on Memory Integrity and Extras \(MicroAutoBox II Features !\[\]\(8d0f0e0fe25b320c33272c52aec1fbca\_img.jpg\)](#)).

---

#### Demo model

For the Simulink models that show how to use the various features of the RTI Watchdog Blockset, refer to the blockset's Demo library.





# Components of the RTI Watchdog Blockset

---

## Introduction

The RTI Watchdog blockset provides RTI blocks that you use in the Simulink model to implement watchdogs for task-controlling purposes, challenge-response monitor units for cyclically observing the correct execution of a particular entity, or monitor units for observing the memory integrity, the voltage supply, or custom conditions.

---

## Where to go from here

### Information in this section

#### [Watchdog Handling..... 18](#)

The RTI Watchdog blockset provides some RTI blocks in its Watchdog library to implement standard watchdog handling.

#### [Challenge-Response Monitoring.....28](#)

The RTI Watchdog blockset provides RTI blocks in the Challenge-Response Monitoring library to implement challenge-response monitoring.

#### [Memory Integrity And Extras.....43](#)

The RTI Watchdog blockset provides RTI blocks in the Memory Integrity and Extras library to implement monitoring facilities for checking memory integrity, the supply voltage, or custom conditions.

# Watchdog Handling

**Introduction** The RTI Watchdog blockset provides some RTI blocks in its Watchdog library to implement standard watchdog handling.

Where to go from here	Information in this section
	<a href="#">WATCHDOG_SETUP.....</a> 18 To set up the watchdog handling.
	<a href="#">WATCHDOG_RETRIGGER_BLx.....</a> 21 To retrigger a monitored task.
	<a href="#">WATCHDOG_INTERRUPT.....</a> 23 To trigger a subsystem if a watchdog generates an interrupt.
	<a href="#">WATCHDOG_EXPIRED.....</a> 26 To get the ID of an expired task.

## WATCHDOG\_SETUP

**Purpose** To set up the watchdog handling.

Where to go from here	Information in this section
	<a href="#">Block Description (WATCHDOG_SETUP).....</a> 19 Gives you information about the appearance and purpose of the block.
	<a href="#">Parameters Page (WATCHDOG_SETUP).....</a> 20 To configure watchdog handling.

## Block Description (WATCHDOG\_SETUP)

### Block

Gives you information about the appearance and purpose of the block.



### Purpose

To configure watchdog handling for the specified tasks.

### Description

With the **WATCHDOG\_SETUP** block, you can specify the tasks to be monitored in your model. The background service of your model is automatically added to the task list and cannot be removed. It is required for monitoring the specified tasks.

To get an overview of the available model tasks, open the RTI Task Configuration Dialog.

For each task that you added to the task list, you can separately configure the timeout limit and the action to be performed if the specified timeout limit has been reached. The task IDs and task names that you can specify in the task list do not need to be identical to the model task entries in the RTI Task Configuration dialog.

Only one **WATCHDOG\_SETUP** block is allowed in your model.

Your model must contain a **WATCHDOG\_RETRIGGER\_BLx** block for each enabled task, except for the background service.


The task monitoring is only active in the *RUN* simulation state. It is deactivated in the *PAUSE* or *STOP* simulation states.

### Dialog pages

The dialog settings can be specified on the following page:

- Parameters page (refer to [Parameters Page \(WATCHDOG\\_SETUP\)](#) on page 20)

### Related RTLib functions

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- `ds1401_wd_init`
- `ds1401_wd_task_monitoring_init`
- `ds1401_wd_task_monitor_restart`
- `ds1401_wd_task_monitor_stop_arm`
- `ds1401_wd_task_monitor_stop_perform`
- `ds1401_wd_interrupt_hook_fcn_set`

## Related topics

## References

RTI Task Configuration Dialog (RTI and RTI-MP Implementation Reference )  
 WATCHDOG\_RETRIGGER\_BLx..... 21

## Parameters Page (WATCHDOG\_SETUP)

**Purpose** To configure watchdog handling.

## Dialog settings

**Add** Lets you add a task to the task list. You can modify the default entries by editing the Task ID, Task name, Timeout limit and Mode settings.

**Remove** Lets you remove a task from the task list. The background service cannot be removed.

**Task ID** Lets you specify the task ID of an added task in the range 1 ... 65535. The task ID of the background service is always 0 and cannot be modified. The specified number must be unique in the task list. The WATCHDOG\_EXPIRED block uses the task ID to inform you about a task with an expired watchdog timer.

**Task name** Lets you specify the name of an added task with up to 32 characters. Each task must have a unique name. The name of the background service is always *BackgroundService* and cannot be modified.

**Timeout limit** Lets you specify the timeout limit for each task separately in the range 100 µs ... 10 s. The unit to be used is seconds.

**Mode** Lets you specify whether to generate a machine check (MCP) interrupt and/or a hardware reset (HRESET) pulse if the timeout limit of the task-specific watchdog timer is exceeded.

Mode	Meaning
Interrupt	Generates a machine check (MCP) interrupt when the task-specific watchdog timer is exceeded for the first time. For example, user-specific functions can be executed to put the application in a safe state before it stops.
Interrupt and HW-Reset	Generates a machine check (MCP) interrupt when the task-specific watchdog timer is exceeded for the first time. After the timeout limit of the background service, the second timeout generates an HRESET pulse, that means that the system is reset immediately. The system does not wait for finishing the user-specific functions triggered by the formerly MCP interrupt.
HW-Reset	Generates an HRESET pulse when the task-specific watchdog timer is exceeded for the first time. The system is reset immediately without executing any user-specific functions beforehand.

The background service only supports the *Interrupt and HW-RESET* and *HW-RESET* modes.

**Enable monitoring** Lets you enable the monitoring of a task that is displayed in the task list. When you add a task to the task list, monitoring is enabled by default. Monitoring of the background service cannot be disabled.

You can use this setting to temporarily disable a task without modifying the model. You do not have to remove the corresponding WATCHDOG\_RETRIGGER\_BLx block from the model because it automatically adapts to the setting. However, it is not erroneous if there is no WATCHDOG\_RETRIGGER\_BLx block in the model that corresponds to a disabled task.

#### Related topics

#### References

Block Description (WATCHDOG_SETUP).....	19
WATCHDOG_EXPIRED.....	26

## WATCHDOG\_RETRIGGER\_BLx

#### Purpose

To retrigger a monitored task.

#### Where to go from here

#### Information in this section

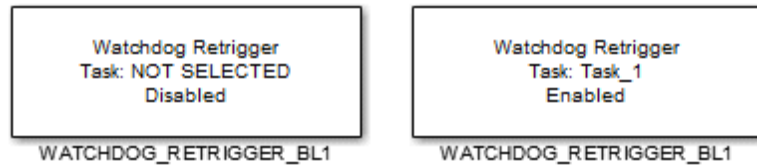
Block Description (WATCHDOG_RETRIGGER_BLx).....	21
Gives you information about the appearance and purpose of the block.	
Parameters Page (WATCHDOG_RETRIGGER_BLx).....	23
To specify the task to be retriggered.	

## Block Description (WATCHDOG\_RETRIGGER\_BLx)

#### Block

Gives you information about the appearance and purpose of the block.

The block display shows the specified task name and whether the task monitoring is enabled or disabled. Modifications in the WATCHDOG\_SETUP block are displayed after updating the diagram.



<b>Purpose</b>	To retrigger a monitored task.
<b>Description</b>	<p>This block is used to reset the task-specific timer. This must be done regularly before the timeout limit specified in the <code>WATCHDOG_SETUP</code> block is reached. Otherwise, the task-specific watchdog triggers an interrupt and/or a hardware reset according to the <code>Mode</code> setting specified in the <code>WATCHDOG_SETUP</code> block.</p> <p>These dependencies cause error messages if there is no <code>WATCHDOG_SETUP</code> block in the model or more than one.</p> <p>For each specified and enabled task in the <code>WATCHDOG_SETUP</code> block, except for the background service, the model must contain a corresponding <code>WATCHDOG_RETRIGGER_BLx</code> block. The block must be placed in the model task that you want to monitor. This can be a task in the root model or in a subsystem.</p> <p>If you want to cover the entire model, you have to monitor every task in the model. To get an overview of the available model tasks, open the RTI Task Configuration Dialog.</p>
<b>Dialog pages</b>	<p>The dialog settings can be specified on the following page:</p> <ul style="list-style-type: none"> <li>Parameters page (refer to <a href="#">Parameters Page (WATCHDOG_RETRIGGER_BLx)</a> on page 23)</li> </ul>
<b>Related RTLib functions</b>	<p>This RTI block is implemented using the following RTLib functions. The <a href="#">MicroAutoBox II RTLib Reference</a> contains descriptions of these functions.</p> <ul style="list-style-type: none"> <li><code>ds1401_wd_task_retrigger_arm</code></li> <li><code>ds1401_wd_task_retrigger_perform</code></li> </ul>
<b>Related topics</b>	<p>References</p> <div> <a href="#">RTI Task Configuration Dialog (RTI and RTI-MP Implementation Reference)</a>  <a href="#">WATCHDOG_SETUP</a>..... 18 </div>

## Parameters Page (WATCHDOG\_RETRIGGER\_BLx)

**Purpose** To specify the task to be retriggered.

**Dialog settings**

**Task name** Lets you select the task to be triggered. The list provides the tasks specified in the WATCHDOG\_SETUP block. The background service is not available, because it is automatically retriggered within the background task. It does not require a block for explicit retriggering.

The task name setting is automatically reset to its initial value *NOT SELECTED*, if an already configured WATCHDOG\_RETRIGGER\_BLx block cannot find the specified task in the task list of the WATCHDOG\_SETUP block. The task has been either removed from the task list or renamed.

### Related topics

### References

[Block Description \(WATCHDOG\\_RETRIGGER\\_BLx\).....](#) 21

[WATCHDOG\\_SETUP.....](#) 18

## WATCHDOG\_INTERRUPT

**Purpose** To trigger a subsystem if a watchdog generates an interrupt.

### Where to go from here

### Information in this section

[Block Description \(WATCHDOG\\_INTERRUPT\).....](#) 24

Gives you information about the appearance and purpose of the block.

[Unit Page \(WATCHDOG\\_INTERRUPT\).....](#) 25

To configure the model termination.

## Block Description (WATCHDOG\_INTERRUPT)

### Block

Gives you information about the appearance and purpose of the block.



### Purpose

To trigger a subsystem if a watchdog generates an interrupt.

### Description

This block can only be used if at least one task in the task list of the `WATCHDOG_SETUP` block is configured for interrupt generation. The *Mode* parameter in the setup dialog must be set to *Interrupt* or *Interrupt and HW-Reset*.

Only one `WATCHDOG_INTERRUPT` block is allowed in a model. All the specified watchdogs use the same interrupt signal. You can therefore specify one subsystem only to execute termination functions. These can optionally include the `MdlTerminate` function that will be executed after the subsystem's functions. If you have specified additional user code in the `<model>_usr.c` file, the `MdlTerminate` function also starts the execution of an included `usr_terminate` function.

The `WATCHDOG_INTERRUPT` block must be directly connected to a Function-call subsystem. Other subsystem elements, for example, a Function Splitter, are not allowed. To get information about the task whose watchdog timer has expired, you must add the `WATCHDOG_EXPIRED` block to the triggered subsystem.

#### Note

- The triggered subsystem can only be used to start a defined termination of the application. It is not possible to restart the application.
- If you have specified the *Interrupt and HW-Reset* mode, the termination functions are only executed during the specified timeout limit of the background service. Afterwards, the hardware reset stops the application immediately.

#### ⚠ CAUTION

##### Risk of personal injury

It is not guaranteed that the subsystem is executed. There might be a fatal system crash that triggers the generation of an interrupt but prevents the subsystem from being executed.



**I/O characteristics**

The following table describes the ports of the block:

Port	Description
Output	
Interrupt	Outputs the interrupt signal to trigger a subsystem. Data type: Function call

**Dialog pages**

The dialog settings can be specified on the following page:

- Unit page (refer to [Unit Page \(WATCHDOG\\_INTERRUPT\)](#) on page 25)

**Related RTLib functions**

None

**Related topics****References**

<a href="#">WATCHDOG_EXPIRED</a> .....	26
<a href="#">WATCHDOG_SETUP</a> .....	18

## Unit Page (WATCHDOG\_INTERRUPT)

**Purpose**

To configure the model termination.

**Dialog settings**

**Execute model termination when a watchdog interrupt occurs** Lets you specify whether to execute the **MdlTerminate** function of your model in addition to the termination function specified in the triggered function-call subsystem. User-specific instructions in the **MdlTerminate** function are also executed.

**Related topics****References**

<a href="#">Block Description (WATCHDOG_INTERRUPT)</a> .....	24
--	----

## WATCHDOG\_EXPIRED

**Purpose** To get the ID of an expired task.

### Block Description (WATCHDOG\_EXPIRED)

**Block** Gives you information about the appearance and purpose of the block.



**Purpose** To get the ID of an expired task.

**Description** This block is used to get the task identifier of a monitored task whose watchdog timer has expired. The block outputs the task ID that you have formerly specified in the `WATCHDOG_SETUP` block. If the block outputs 0 as the task ID, the watchdog timer of the background service has expired.

Only one `WATCHDOG_EXPIRED` block is allowed in a model. It must be placed in the function-call subsystem that is triggered by a `WATCHDOG_INTERRUPT` block. Otherwise the build process stops.

#### I/O characteristics

The following table describes the ports of the block:


Port	Description
Output	
Task ID	Returns the identifier of the expired task. Data type: UINT16 Data width: 1 Range: 0 ... 65535

#### Dialog pages

There are no dialog settings to be specified on the Unit page.

---

**Related RTLib functions**

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_wd\_task\_expired\_get

# Challenge-Response Monitoring

## Introduction

The RTI Watchdog blockset provides RTI blocks in the Challenge-Response Monitoring library to implement challenge-response monitoring.

## Where to go from here

### Information in this section

<a href="#">CRM_UNIT_SETUP_BL</a> .....	28
To initialize and configure a challenge-response monitoring unit.	
<a href="#">CRM_RESPONSE_BL</a> .....	35
To transfer a response value to the specified monitor unit.	
<a href="#">CRM_SUPERVISION_STATUS</a> .....	37
To provide status information on the supervision process.	
<a href="#">CRM_INTERRUPT</a> .....	38
To trigger a subsystem if a CRM interrupt occurs.	
<a href="#">CRM_INTERRUPT_DECODE</a> .....	40
To identify the monitor unit that caused the interrupt generation.	

### Information in other sections

[Basics on Challenge-Response Monitoring \(MicroAutoBox II Features !\[\]\(05be7c7a8995decd503647c99211f7c2\_img.jpg\)\)](#)

[Challenge-Response Monitoring \(MicroAutoBox II RTLib Reference !\[\]\(16cd6e1a39784ecf52b4db09f4865f40\_img.jpg\)\)](#)

# CRM\_UNIT\_SETUP\_BL

## Purpose

To initialize and configure a challenge-response monitoring unit.

## Where to go from here

### Information in this section

<a href="#">Block Description (CRM_UNIT_SETUP_BL)</a> .....	29
Gives you information about the appearance and purpose of the block.	
<a href="#">Unit Page (CRM_UNIT_SETUP_BL)</a> .....	31
To specify settings for the identification of the monitoring unit.	

<a href="#">Parameters Page (CRM_UNIT_SETUP_BL).....</a>	<a href="#">32</a>
To configure the monitoring unit.	
<a href="#">Challenge-Response Page (CRM_UNIT_SETUP_BL).....</a>	<a href="#">33</a>
To specify the challenge-response pairs.	
<a href="#">Advanced Page (CRM_UNIT_SETUP_BL).....</a>	<a href="#">34</a>
To enable optional outputs.	

#### Information in other sections

[Basics on Challenge-Response Monitoring \(MicroAutoBox II Features !\[\]\(bd1a142de767a21e5362c595f844a4ff\_img.jpg\)\)](#)

## Block Description (CRM\_UNIT\_SETUP\_BL)

### Block

Gives you information about the appearance and purpose of the block.



### Purpose

To initialize and configure a challenge-response monitoring unit.

### Description

You can specify up to 14 challenge-response monitoring units in your model. With each CRM\_UNIT\_SETUP\_BL block in your model, you initialize a separate challenge-response monitoring hardware. The monitoring unit which you configure with this block lets you periodically observe the correct execution of a particular entity, such as a timer task or a subsystem. The specified values for the challenge-response pairs are used to evaluate the correctness of the responses, which must be sent within the specified time interval.

For each CRM\_UNIT\_SETUP\_BLx block in your model, there must be a corresponding CRM\_RESPONSE\_BLx block. Within the specified response timeout there might be multiple responses, but only the first response will be evaluated by the monitoring unit.

You can specify how many successive failures are tolerated before the specified failure action is executed. If you specified to generate an interrupt, the model must contain a corresponding CRM\_INTERRUPT block.

The challenge-response monitoring does not react to the `simState` variable. If you stop or pause your real-time application, the challenge-response monitoring will execute the specified failure handling. Your hardware might be reset.

It is recommended not to pause or stop the real-time application when using the challenge-response monitoring.

#### I/O characteristics

The following table describes the ports of the block:

Port	Description
Output	
Challenge	Provides the current challenge value. The value is updated only once every challenge cycle-time and kept for one task sample. Then it is reset to 0. Data type: UInt32 Data width: 1 Range: 1 ... 4,294,967,295 ( $=2^{32}-1$ )
New Challenge	Provides a flag indicating whether the current challenge value was already written to the model. The value is updated only once every challenge cycle time and kept for one task sample. Then it is reset to 0. Data type: Boolean Data width: 1 Range: 0, 1 <ul style="list-style-type: none"> <li>0: The current value was already written.</li> <li>1: A new value is available.</li> </ul>
Current Failure Count	Provides the number of currently detected failures. This failure counter is reset if the monitoring unit received a valid response value within the specified timeout. Only the first response value in the current challenge cycle time is evaluated. Available only if <b>Enable Current Failure Count output</b> is set on the <b>Advanced</b> page. Data type: UInt32 Data width: 1 Range: 0 ... $2^{32}-1$
Total Failure Count	Provides the total number of detected failures. This failure counter is reset only if you reload the real-time application or reset the real-time hardware. Available only if <b>Enable Total Failure Count output</b> is set on the <b>Advanced</b> page. Data type: UInt32 Data width: 1 Range: 0 ... $2^{32}-1$
Failure Limit Exceeded	Provides a flag that indicates whether the specified failure limit is exceeded. Once set, the flag is not reset until you reload the real-time application. With this variable, you can control your own failure action without generating an interrupt or hardware reset and exiting the real-time


Port	Description
	<p>application. You can use this port, for example, for debugging purposes or for monitoring entities that are not safety-relevant.</p> <p>Available only if By output port is set as the failure action on the Parameters page.</p> <p>Data type: Boolean</p> <p>Data width:1</p> <p>Range: 0, 1</p> <ul style="list-style-type: none"> <li>▪ 0: The current failure counter has not exceeded the number of successive failures to be tolerated.</li> <li>▪ 1: The current failure counter exceeded the number of successive failures to be tolerated.</li> </ul>

### Dialog pages

The dialog settings can be specified on the following pages:

- Unit page (refer to [Unit Page \(CRM\\_UNIT\\_SETUP\\_BL\)](#) on page 31)
- Parameters page (refer to [Parameters Page \(CRM\\_UNIT\\_SETUP\\_BL\)](#) on page 32)
- Challenge-Response page (refer to [Challenge-Response Page \(CRM\\_UNIT\\_SETUP\\_BL\)](#) on page 33)
- Advanced page (refer to [Advanced Page \(CRM\\_UNIT\\_SETUP\\_BL\)](#) on page 34)

### Related RTLib functions

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_crm\_init
- ds1401\_crm\_unit\_init
- ds1401\_crm\_unit\_challenge\_get
- ds1401\_crm\_unit\_fail\_cntrs\_get

## Unit Page (CRM\_UNIT\_SETUP\_BL)

### Purpose

To specify settings for the identification of the monitoring unit.

### Dialog settings

**Unit ID** Lets you specify a number in the range 1 ... 14 as the unit identifier. The value must be unique within the model.

**Unit name** Lets you specify a descriptive name for the specified unit with up to 32 characters. The string must be unique within the model.

#### Note

A valid unit name is a character string of letters, digits, and underscores. There are the following naming restrictions for the unit name:

- The first character must be a letter.
- The name must not be a keyword, such as **while** or **if**.

#### Related topics

#### References

Advanced Page (CRM_UNIT_SETUP_BL).....	34
Block Description (CRM_UNIT_SETUP_BL).....	29
Challenge-Response Page (CRM_UNIT_SETUP_BL).....	33
Parameters Page (CRM_UNIT_SETUP_BL).....	32

## Parameters Page (CRM\_UNIT\_SETUP\_BL)

#### Purpose

To configure the monitoring unit.

#### Dialog settings

**Unit ID** Displays the monitoring unit's identifier that you specified on the Unit page.

**Failure action mode** Lets you specify the failure action to be executed if a failure is activated. The activation of a failure depends on the number of successive failures to be ignored that you specify in the **Failure count limit** setting.

Failure Action Mode	Meaning
Interrupt	An activated failure generates a machine check (MCP) interrupt. With a CRM_INTERRUPT block, you can use the generated interrupt to trigger a subsystem for storing data or setting the outputs to a safe state before the real-time application exits.
Interrupt and HW Reset	An activated failure generates a machine check (MCP) interrupt at first that triggers the subsystem and exits the real-time application. Then the challenge-response monitoring hardware detects missing responses because of the terminated real-time application and generates an HRESET pulse. The time between the generation of the interrupt and the generation of the HRESET pulse takes the time of at least one challenge cycle and at most two challenge cycles.



Failure Action Mode	Meaning
HW Reset	An activated failure generates an HRESET pulse to immediately reset the hardware. No termination process is executed. A hardware reset resets the PowerPC processor and the intermodule bus. The host communication is not affected.
By output port	An activated failure generates a signal on the <b>Failure Limit Exceeded</b> output port.

**Challenge cycle time** Lets you specify the periodic time interval in which a new challenge is provided to the monitored entity in the range 2e-3 ... 16.384 s. The accuracy is 1 ms.

**Response timeout** Lets you specify the maximum time in which the monitored entity has to return a response to the monitor unit in the range 1e-3 ... (Challenge cycle time - 0.001) s. The accuracy is 1 ms.

**Failure count limit** Lets you specify the number of successive failures to be ignored before a failure action is activated in the range 0 ... 255. If you specify 0, the first failure immediately activates the action that you specified in the Failure action mode setting.

## Related topics

## References

Advanced Page (CRM_UNIT_SETUP_BL).....	34
Block Description (CRM_UNIT_SETUP_BL).....	29
Challenge-Response Page (CRM_UNIT_SETUP_BL).....	33
Unit Page (CRM_UNIT_SETUP_BL).....	31

# Challenge-Response Page (CRM\_UNIT\_SETUP\_BL)

**Purpose** To specify the challenge-response pairs.

## Dialog settings

**Unit ID** Displays the monitoring unit's identifier that you specified on the Unit page.

**Number of challenge-response pairs** Lets you specify the number of challenge-response pairs in the range 1 ... 16.

**Values** Lets you specify the challenge values and the response values for the enabled challenge-response pairs. You can specify a value in the range 1 ... 4,294,967,295 ( $=2^{32}-1$ ). The specified values must be unique within the same group (challenge or response).

---

**Related topics****References**

Advanced Page (CRM_UNIT_SETUP_BL).....	34
Block Description (CRM_UNIT_SETUP_BL).....	29
Parameters Page (CRM_UNIT_SETUP_BL).....	32
Unit Page (CRM_UNIT_SETUP_BL).....	31

## Advanced Page (CRM\_UNIT\_SETUP\_BL)

---

**Purpose**

To enable optional outputs.

---

**Dialog settings**

**Unit ID** Displays the monitoring unit's identifier that you specified on the Unit page.

**Enable Current Failure Count output** Lets you enable the optional output providing the number of currently detected failures. This failure counter is reset if a valid response value was received within the specified timeout limit or if the real-time application was reloaded.

**Enable Total Failure Count output** Lets you enable the optional output providing the total number of detected failures. This failure counter is reset if you reload the real-time application or reset the hardware.

---

**Related topics****References**

Block Description (CRM_UNIT_SETUP_BL).....	29
Challenge-Response Page (CRM_UNIT_SETUP_BL).....	33
Parameters Page (CRM_UNIT_SETUP_BL).....	32
Unit Page (CRM_UNIT_SETUP_BL).....	31

## CRM\_RESPONSE\_BL

**Purpose** To transfer a response value to the specified monitor unit.

### Where to go from here

### Information in this section

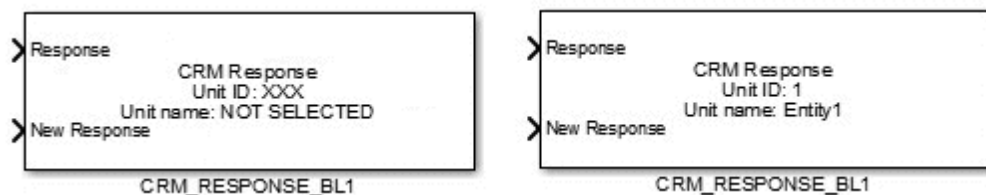
- [Block Description \(CRM\\_RESPONSE\\_BL\)..... 35](#)  
Gives you information about the appearance and purpose of the block.
- [Unit Page \(CRM\\_RESPONSE\\_BL\)..... 36](#)  
To return a response to the specified monitor unit.

### Information in other sections

- [CRM\\_UNIT\\_SETUP\\_BL..... 28](#)  
To initialize and configure a challenge-response monitoring unit.
- [Basics on Challenge-Response Monitoring \(MicroAutoBox II Features !\[\]\(67ff022fd78f943b679992c2874bbfd1\_img.jpg\)\)](#)

## Block Description (CRM\_RESPONSE\_BL)

**Block** Gives you information about the appearance and purpose of the block.



**Purpose** To return a response to the specified monitoring unit.

**Description** This block is used to read a static or dynamically generated value as the response value from the model. The value is transferred to the specified monitoring unit if it is marked as a new one.

**I/O characteristics**

The following table describes the ports of the block:


Port	Description
Input	
Response	Specifies the value that is the response to the current challenge. Data type: UInt32 Data width: 1 Range: 1 ... $2^{32}-1$
New Response	Controls writing the response value to the specified monitor unit. While New Response is set to 0, no new response is available. If the value is new, you can set this value to >0 for one cycle to start the data transfer. Data type: UInt32 Data width: 1 Range: <ul style="list-style-type: none"> <li>0: The transfer of the response value to the monitor unit is disabled, e.g., because the response value was already written.</li> <li>&gt;0: The transfer of the response value to the monitor unit is enabled, e.g., because the response value is new..</li> </ul>

**Dialog pages**

The dialog settings can be specified on the following page:

- Unit page (refer to [Unit Page \(CRM\\_RESPONSE\\_BL\)](#) on page 36)

**Related RTLib functions**

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_crm\_unit\_response\_set

## Unit Page (CRM\_RESPONSE\_BL)

**Purpose**

To return a response to the specified monitor unit.

**Dialog settings**

**Monitor unit** Lets you select the monitoring unit to which the response is to be returned. If no monitoring unit is selected or a previously selected monitoring unit is no longer available in the Simulink model, the **NOT SELECTED** entry is

displayed. If you open the list, all monitoring units that are available in the Simulink model are displayed with their unit names and unit identifiers.

## Related topics

## References

[Block Description \(CRM\\_RESPONSE\\_BL\)..... 35](#)

# CRM\_SUPERVISION\_STATUS

## Purpose

To provide status information on the supervision process.

## Where to go from here

## Information in this section

[Block Description \(CRM\\_SUPERVISION\\_STATUS\)..... 37](#)  
Gives you information about the appearance and purpose of the block.

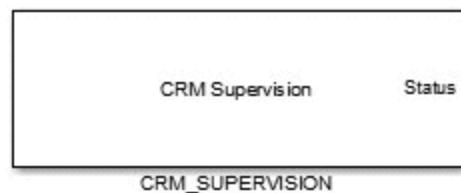
## Information in other sections

[Basics on Challenge-Response Monitoring \(MicroAutoBox II Features !\[\]\(bd3b31712ad9bab5a241210fa6925cdd\_img.jpg\)](#))

## Block Description (CRM\_SUPERVISION\_STATUS)

## Block

Gives you information about the appearance and purpose of the block.



## Purpose

To provide status information on the supervision process.

**Description**

As soon as the challenge-response monitoring is enabled by using a `CRM_UNIT_SETUP_BL` block in your model, a self-test is periodically executed to detect hardware failures. Only one `CRM_SUPERVISION` block is allowed in your model to get the self-test status of the monitoring hardware.

**I/O characteristics**


The following table describes the ports of the block:

Port	Description
Output	
Status	<p>Returns the status information on the supervision process for the monitoring hardware. The supervision process checks whether the monitoring hardware is working correctly.</p> <p>Data type: UInt32</p> <p>Data width: 1</p> <p>Range: 0, 1</p> <ul style="list-style-type: none"> <li>0: The challenge-response monitoring hardware is working correctly.</li> <li>1: The challenge-response monitoring hardware is not working correctly.</li> </ul>

**Dialog pages**

This block provides the Unit page, but there are no settings to be specified.

**Related RTLib functions**

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- `ds1401_crm_supervision_stat_get`

## CRM\_INTERRUPT

**Purpose**

To trigger a subsystem if a CRM interrupt occurs.

**Where to go from here****Information in this section**

<a href="#">Block Description (CRM_INTERRUPT)</a> .....	39
Gives you information about the appearance and purpose of the block.	
<a href="#">Unit Page (CRM_INTERRUPT_BL)</a> .....	40
To configure the model termination.	

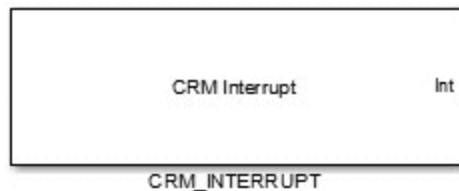
## Information in other sections

[Basics on Challenge-Response Monitoring \(MicroAutoBox II Features !\[\]\(d84e7ea36f695d92cb39ec32c307ac93\_img.jpg\)\)](#)

## Block Description (CRM\_INTERRUPT)

### Block

Gives you information about the appearance and purpose of the block.



### Purpose

To trigger a subsystem if a CRM interrupt occurs.

### Description

In the CRM\_UNIT\_SETUP\_BL block, you can specify the failure action to be executed if the conditions for activating the failure handling are fulfilled. If you specified **Interrupt** or **Interrupt and HW Reset** as the failure action, a machine check (MCP) interrupt is generated. An interrupt triggered by the challenge-response monitoring can be made available in your model by using the CRM\_INTERRUPT block. Instead of triggering a subsystem, you can specify to directly execute the model termination.

Because the block receives the interrupts of any instantiated monitoring unit in your model, you have to note the following points:

- Only one CRM\_INTERRUPT block is allowed in the model.
- You can use the CRM\_INTERRUPT\_DECODE block to get the identifier of the monitoring unit that caused the interrupt generation.

### I/O characteristics

The following table describes the ports of the block:


Port	Description
Output	
Int	Provides the generated CRM interrupt in your model as a trigger output. Data type: Function call

**Dialog pages**

The dialog settings can be specified on the following page:

- Unit page (refer to [Unit Page \(CRM\\_INTERRUPT\\_BL\)](#) on page 40)

**Related RTLib functions**

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_crm\_int\_hook\_fcn\_set

**Related topics****References**

<a href="#">CRM_INTERRUPT_DECODE</a> .....	40
<a href="#">CRM_UNIT_SETUP_BL</a> .....	28

## Unit Page (CRM\_INTERRUPT\_BL)

**Purpose**

To configure the model termination.

**Dialog settings**

**Execute model termination when a CRM interrupt occurs** Lets you specify to directly execute the model termination process before the real-time application exits.

To add user code to the model termination, you have to add the code to the `usr_terminate` function in the `<model>_usr.c` file.

**Related topics****References**

<a href="#">Block Description (CRM_INTERRUPT)</a> .....	39
---	----

## CRM\_INTERRUPT\_DECODE

**Purpose**

To identify the monitor unit that caused the interrupt generation.



**Where to go from here****Information in this section**

[Block Description \(CRM\\_INTERRUPT\\_DECODE\).....41](#)  
 Gives you information about the appearance and purpose of the block.

**Information in other sections**

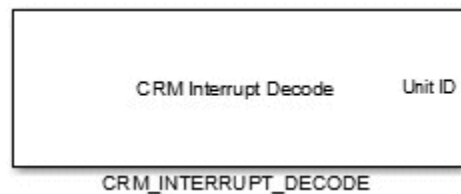
[CRM\\_INTERRUPT.....38](#)  
 To trigger a subsystem if a CRM interrupt occurs.

[Basics on Challenge-Response Monitoring \(MicroAutoBox II Features !\[\]\(d66ff64371a51729ac8c1cdaa685ba6f\_img.jpg\)\)](#)

## Block Description (CRM\_INTERRUPT\_DECODE)

**Block**

Gives you information about the appearance and purpose of the block.

**Purpose**

To identify the monitoring unit that caused the interrupt generation.

**Description**

If you specified to generate an interrupt as the failure action in more than one instantiated monitoring unit in your model, you can use this block to get information on the interrupt-causing monitoring unit. The block must be added to the subsystem, that will be triggered by the CRM interrupt. If you specified to directly execute the model termination, you must use the **ds1401\_crm\_unit\_causing\_int\_get** RTLib function in the **usr\_terminate** code in the **<model>\_usr.c** file.

**I/O characteristics**

The following table describes the ports of the block:

Port	Description
Output	
Unit ID	Provides the identifier of the monitor unit that caused the interrupt generation.

Port	Description
	Data type: UInt32 Data width: 1 Range: 1 ... 14


---

#### Dialog pages

This block provides the Unit page, but there are no settings to be specified.

---

#### Related RTLib functions

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_crm\_unit\_causing\_int\_get

# Memory Integrity And Extras

## Introduction

The RTI Watchdog blockset provides RTI blocks in the **Memory Integrity and Extras** library to implement monitoring facilities for checking memory integrity, the supply voltage, or custom conditions.

## Where to go from here

### Information in this section

<a href="#">MEMORY_VBAT_MONITORING.....</a>	<a href="#">43</a>
To configure memory and VBAT monitoring.	
<a href="#">MEMORY_VBAT_INTERRUPT.....</a>	<a href="#">48</a>
To trigger a subsystem if a memory or VBAT interrupt occurs.	
<a href="#">CUSTOM_FAILURE_ACTION.....</a>	<a href="#">50</a>
To configure a custom failure action.	
<a href="#">CUSTOM_FAILURE_INTERRUPT.....</a>	<a href="#">54</a>
To trigger a subsystem if a custom failure interrupt occurs.	

### Information in other sections

[Basics on Memory Integrity and Extras \(MicroAutoBox II Features !\[\]\(3e2231b1ad3ca8da8658228c00dd08e0\_img.jpg\)\)](#)

[Memory Integrity and Extras \(MicroAutoBox II RTLib Reference !\[\]\(96a82dd1250f57fd139c5f3b80c9d977\_img.jpg\)\)](#)

# MEMORY\_VBAT\_MONITORING

## Purpose

To configure memory and VBAT monitoring.

## Where to go from here

### Information in this section

<a href="#">Block Description (MEMORY_VBAT_MONITORING).....</a>	<a href="#">44</a>
Gives you information about the appearance and purpose of the block.	
<a href="#">Unit Page (MEMORY_VBAT_MONITORING).....</a>	<a href="#">47</a>
To configure memory and VBAT monitoring.	

## Information in other sections

[Basics on Memory Integrity and Extras \(MicroAutoBox II Features !\[\]\(2bdfe261b986065ee0ac76460d6528c9\_img.jpg\)\)](#)

## Block Description (MEMORY\_VBAT\_MONITORING)

**Block** Gives you information about the appearance and purpose of the block.



**Purpose** To configure memory and VBAT monitoring.

**Description** You can use only one MEMORY\_VBAT\_MONITORING block in your model. You must enable at least one of the monitoring facilities or the ROM check to make the block applicable.

The specified failure action is relevant to each enabled monitoring facility, except for the ROM check, which terminates the real-time application before its initialization phase. If a failure is detected during run time of the real-time application, the specified failure action will be triggered immediately. If you specified to report the failure at an output port, the report flag is set in the next sample step. You can reset the status of this flag only by reloading the real-time application.

The monitoring is done by a software implementation that is executed on the application processor. External hardware is used for triggering the interrupt or resetting the hardware.

The monitoring is periodically executed in the background loop of your real-time application. It is executed with a low priority. To guarantee the execution of the monitoring facilities, you have to consider the turnaround time of your application. For more information, refer to [Basics on Memory Integrity and Extras \(MicroAutoBox II Features !\[\]\(dd161862f9164df98f62b726e9846241\_img.jpg\)\)](#).

If you specified to generate an interrupt, the model must contain a corresponding `MEMORY_VBAT_INTERRUPT` block.

#### Note

If you enable the ROM check, the ROM monitoring, or the heap monitoring, the PPC file of your real-time application will be modified after the build process to provide special code used for checking the memory integrity.

### I/O characteristics

The following table describes the ports of the block:

Port	Description
Output	
ROM Failure Detected	<p>Provides a flag that indicates whether a ROM failure was detected. Once set, the flag is not reset until you reload the real-time application. With this flag, you can control your own failure action without generating an interrupt or hardware reset and exiting the real-time application. You can use this port, for example, for debugging purposes.</p> <p>Available only if ROM monitoring is enabled and <b>By output port</b> is set as the failure action on the <b>Unit</b> page.</p> <p>Data type: Boolean Data width: 1 Range: 0, 1</p> <ul style="list-style-type: none"> <li>0: No error.</li> <li>1: ROM data is corrupted. The checksum of the current ROM contents differs from the reference value.</li> </ul>
Heap Failure Detected	<p>Provides a flag that indicates whether a heap failure was detected. Once set, the flag is not reset until you reload the real-time application. With this flag, you can control your own failure action without generating an interrupt or hardware reset and exiting the real-time application. You can use this port, for example, for debugging purposes.</p> <p>Available only if heap monitoring is enabled and <b>By output port</b> is set as the failure action on the <b>Unit</b> page.</p> <p>Data type: Boolean Data width: 1 Range: 0, 1</p> <ul style="list-style-type: none"> <li>0: No error.</li> <li>1: The contents of a heap block is corrupted.</li> </ul>
Stack Failure Detected	<p>Provides a flag that indicates whether a stack failure was detected. Once set, the flag is not reset until you reload the real-time application. With this flag, you can control your own failure action</p>


Port	Description
VBAT Failure Detected	<p>without generating an interrupt or hardware reset and exiting the real-time application. You can use this port, for example, for debugging purposes.</p> <p>Available only if stack monitoring is enabled and <b>By output port</b> is set as the failure action on the <b>Unit</b> page.</p> <p>Data type: Boolean Data width: 1 Range: 0, 1</p> <ul style="list-style-type: none"> <li>0: No error.</li> <li>1: Stack overflow occurred.</li> </ul> <p>Provides a flag that indicates whether a VBAT failure was detected. Once set, the flag is not reset until you reload the real-time application. With this flag, you can control your own failure action without generating an interrupt or hardware reset and exiting the real-time application. You can use this port, for example, for debugging purposes.</p> <p>Available only if VBAT monitoring is enabled and <b>By output port</b> is set as the failure action on the <b>Unit</b> page.</p> <p>Data type: Boolean Data width: 1 Range: 0, 1</p> <ul style="list-style-type: none"> <li>0: No error.</li> <li>1: The supply voltage is below the specified voltage threshold.</li> </ul>

**Dialog pages**

The dialog settings can be specified on the following pages:

- Unit page (refer to [Unit Page \(MEMORY\\_VBAT\\_MONITORING\)](#) on page 47)

**Related RTLib functions**

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_fse\_init
- ds1401\_fse\_heap\_mon\_setup
- ds1401\_fse\_heap\_mon\_stat\_get
- ds1401\_fse\_rom\_mon\_setup
- ds1401\_fse\_rom\_mon\_stat\_get
- ds1401\_fse\_stack\_mon\_setup
- ds1401\_fse\_stack\_mon\_stat\_get
- ds1401\_fse\_vbat\_mon\_setup
- ds1401\_fse\_vbat\_mon\_stat\_get

## Unit Page (MEMORY\_VBAT\_MONITORING)

**Purpose** To configure memory and VBAT monitoring.

### Dialog settings

**Enable ROM monitoring** Lets you specify to periodically verify the integrity of the read-only data of the real-time application. If the ROM data is corrupted, a failure is detected.

**Enable heap monitoring** Lets you specify to periodically verify the integrity of the heap memory allocated by the real-time application. If the contents of a heap block is corrupted, a failure is detected. The reference data is dynamically added to the allocated heap blocks. Heap monitoring therefore increases the required heap memory by 64 bytes per heap block.

**Enable stack monitoring** Lets you specify to periodically verify the integrity of the stack memory allocated by the real-time application. If a stack overflow occurs, a failure is detected.

**Enable VBAT monitoring** Lets you specify to periodically verify the voltage level of the power supply of the real-time hardware. If the voltage level falls below the specified threshold level, a failure is detected.

**Threshold level** Lets you specify the minimum voltage level to be tolerated in the range 4.0 ... 40.0 V in steps of 0.0244 V. This setting is enabled only if you have enabled the VBAT monitoring.

**Failure action mode** Lets you specify the failure action to be executed if a failure is detected. This setting is only enabled if you set at least one of the monitoring facilities.

Failure Action Mode	Meaning
Interrupt	A detected failure generates a machine check (MCP) interrupt. With a <b>MEMORY_VBAT_INTERRUPT</b> block, you can use the generated interrupt to trigger a subsystem for storing data or setting the outputs to a safe state before the real-time application exits.
Interrupt and HW Reset	A detected failure first generates a machine check (MCP) interrupt that triggers the subsystem and exits the real-time application. Then, an HRESET pulse is generated after the time specified by the <b>HW-Reset delay</b> setting.
HW Reset	A detected failure generates an HRESET pulse to immediately reset the hardware. No termination process is executed. A hardware reset resets the PowerPC processor and the intermodule bus. The host communication is not affected.
By output port	A detected failure generates a signal on the corresponding output port. For each enabled monitoring facility, this failure action mode enables separate output ports in the block.

**HW-Reset delay** Lets you specify the time interval between the generation of an interrupt and the generation of the HRESET pulse in the range

0.001 ... 16.383 s in steps of 1 ms. This setting is enabled only if you set **Interrupt and HW Reset** as the failure action mode.

**Enable ROM check** Lets you specify to check the read-only data of the real-time application once before the real-time application starts on the hardware. The ROM check verifies the checksum of the current ROM contents and stops execution if the checksum differs from the reference value, which was written to the PPC file. This check displays an error message and exits the real-time application. A specified failure action mode is not relevant to the ROM check.

## Related topics

## References

[Block Description \(MEMORY\\_VBAT\\_MONITORING\)..... 44](#)

# MEMORY\_VBAT\_INTERRUPT

## Purpose

To trigger a subsystem if a memory or VBAT interrupt occurs.

## Where to go from here

## Information in this section

[Block Description \(MEMORY\\_VBAT\\_INTERRUPT\)..... 49](#)

Gives you information about the appearance and purpose of the block.

[Unit Page \(MEMORY\\_VBAT\\_INTERRUPT\)..... 50](#)

To trigger a subsystem if a memory or VBAT interrupt occurs.

## Information in other sections

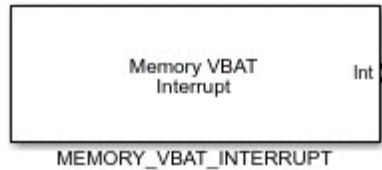
[Basics on Memory Integrity and Extras \(MicroAutoBox II Features !\[\]\(4146d17f71dced09c6ad789cacceaa6d\_img.jpg\) \)](#)



## Block Description (MEMORY\_VBAT\_INTERRUPT)

### Block

Gives you information about the appearance and purpose of the block.



### Purpose

To trigger a subsystem if a memory or VBAT interrupt occurs.

### Description

In the **MEMORY\_VBAT\_MONITORING** block, you can specify the failure action to be executed if an error condition is detected. If you specified **Interrupt** or **Interrupt and HW Reset** as the failure action, a machine check (MCP) interrupt is generated. An interrupt triggered by the memory or VBAT monitoring can be made available in your model by using the **MEMORY\_VBAT\_INTERRUPT** block. Additional to triggering a subsystem, you can specify to directly execute the model termination.

Because there is only one interrupt for all enabled memory and VBAT monitoring facilities in your model, only one **MEMORY\_VBAT\_INTERRUPT** block is allowed in the model.

### I/O characteristics

The following table describes the ports of the block:


Port	Description
Output	
Int	Provides the generated memory or VBAT interrupt in your model as a trigger output. Data type: Function call

### Dialog pages

The dialog settings can be specified on the following pages:

- Unit page (refer to [Unit Page \(MEMORY\\_VBAT\\_INTERRUPT\)](#) on page 50)

### Related RTLib functions

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_fse\_int\_hook\_fcn\_set

## Unit Page (MEMORY\_VBAT\_INTERRUPT)

<b>Purpose</b>	To trigger a subsystem if a memory or VBAT interrupt occurs.
<b>Dialog settings</b>	<p><b>Execute model termination when a memory or VBAT interrupt occurs</b> Lets you specify to directly execute the model termination process before the real-time application exits.</p> <p>To add user code to the model termination, you have to add the code to the <code>usr_terminate</code> function in the <code>&lt;model&gt;_usr.c</code> file.</p>
<b>Related topics</b>	<p><b>References</b></p>

[Block Description \(MEMORY\\_VBAT\\_INTERRUPT\)..... 49](#)

## CUSTOM\_FAILURE\_ACTION

<b>Purpose</b>	To configure a custom failure action.
----------------	---------------------------------------

<b>Where to go from here</b>	<p><b>Information in this section</b></p> <p><a href="#">Block Description (CUSTOM_FAILURE_ACTION)..... 51</a> Gives you information about the appearance and purpose of the block.</p> <p><a href="#">Unit Page (CUSTOM_FAILURE_ACTION)..... 52</a> To configure a custom failure action.</p>
------------------------------	--

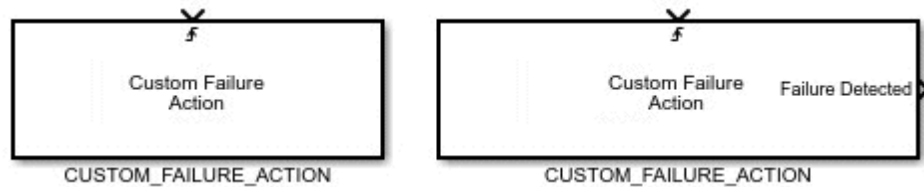
### Information in other sections

[Basics on Memory Integrity and Extras \(MicroAutoBox II Features !\[\]\(3211b5d1d968fc1665909b34f9f16010\_img.jpg\)\)](#)

## Block Description (CUSTOM\_FAILURE\_ACTION)

### Block

Gives you information about the appearance and purpose of the block.



The symbol for the trigger input changes according to the specified trigger type.

### Purpose

To initialize and configure a custom failure action.

### Description

This block allows you to react to a custom check with the same failure action modes that are available for the memory and VBAT monitoring. The failure actions for memory and VBAT monitoring and the failure actions for custom failure monitoring can be separately specified. A failure detected by your custom check is to be transmitted to the block as trigger input.

For example, if you monitor the range of a model parameter and you detect a range violation, you can generate a rising edge on the signal that is connected to the block's trigger input to execute the specified custom failure action.

The monitoring is done by a software implementation that is executed on the application processor. External hardware is used for triggering the interrupt or resetting the hardware.

You can use only one CUSTOM\_FAILURE\_ACTION block in your model.

If you specified to generate an interrupt, the model must contain a corresponding CUSTOM\_FAILURE\_INTERRUPT block.

If you specified to report the failure at the output port, you can reset the status of this flag only by restarting the real-time application.

### I/O characteristics

The following table describes the ports of the block:

Port	Description
Input	
Trigger input	Provides the input port for the trigger signal that can be of the following type: <ul style="list-style-type: none"> <li>▪ Rising edge</li> <li>▪ Falling edge</li> <li>▪ Rising or falling edge</li> <li>▪ Function call</li> </ul>

Port	Description
Output Failure Detected	<p>Provides a flag that indicates whether a custom failure was detected. Once set, the flag is not reset until you reload the real-time application. With this variable, you can control your own failure action without generating an interrupt or hardware reset and exiting the real-time application. You can use this port, for example, for debugging purposes.</p> <p>Available only if By output port is set as the failure action on the Unit page.</p> <p>Data type: Boolean</p> <p>Data width: 1</p> <p>Range: 0, 1</p> <ul style="list-style-type: none"> <li>0: No error.</li> <li>1: The specified custom failure was detected.</li> </ul>

**Dialog pages**

The dialog settings can be specified on the following pages:

- Unit page (refer to [Unit Page \(CUSTOM\\_FAILURE\\_ACTION\)](#) on page 52)

**Related RTLib functions**

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#) contains descriptions of these functions.

- ds1401\_fse\_custom\_mon\_setup
- ds1401\_fse\_cust\_mon\_stat\_get
- ds1401\_fse\_cust\_fail\_action\_set
- ds1401\_fse\_cust\_mon\_action\_arm
- ds1401\_fse\_cust\_mon\_action\_trig


## Unit Page (CUSTOM\_FAILURE\_ACTION)



**Purpose**

To configure a custom failure action.

**Dialog settings**

**Trigger type** Lets you specify the type of the trigger input signal provided by your custom check when a failure is detected.

Trigger type	Meaning	Trigger Symbol
Rising	The block reacts to a rising edge in the trigger signal.	

Trigger type	Meaning	Trigger Symbol
Falling	The block reacts to a falling edge in the trigger signal.	
Either	The block reacts to a rising and a falling edge in the trigger signal.	
Function call	The block reacts to a function call at the trigger input port.	Trigger()

**Failure action mode** Lets you specify the failure action to be executed if a custom failure is detected.

Failure Action Mode	Meaning
Interrupt	A detected failure generates a machine check (MCP) interrupt. With a <code>CUSTOM_FAILURE_INTERRUPT</code> block, you can use the generated interrupt to trigger a subsystem for storing data or setting the outputs to a safe state before the real-time application exits.
Interrupt and HW Reset	A detected failure generates a machine check (MCP) interrupt at first that triggers the subsystem and exits the real-time application. Then, an HRESET pulse is generated after the time specified by the <code>HW-Reset delay</code> setting.
HW Reset	A detected failure generates an HRESET pulse to immediately reset the hardware. No termination process is executed. A hardware reset resets the PowerPC processor and the intermodule bus. The host communication is not affected.
By output port	A detected failure generates a signal on the Failure Detected output port.

**HW-Reset delay** Lets you specify the time interval between the generation of an interrupt and the generation of the HRESET pulse in the range 0.001 ... 16.383 s in steps of 1 ms.

## Related topics

## References

[Block Description \(CUSTOM\\_FAILURE\\_ACTION\).....51](#)

## CUSTOM\_FAILURE\_INTERRUPT

**Purpose** To trigger a subsystem if a custom failure interrupt occurs.

### Where to go from here

### Information in this section

[Block Description \(CUSTOM\\_FAILURE\\_INTERRUPT\)..... 54](#)  
Gives you information about the appearance and purpose of the block.

[Unit Page \(CUSTOM\\_FAILURE\\_INTERRUPT\)..... 55](#)  
To trigger a subsystem if a custom failure interrupt occurs.

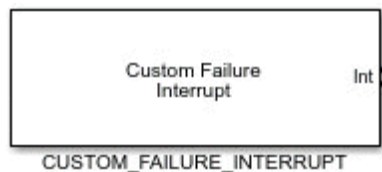
### Information in other sections

[Basics on Memory Integrity and Extras \(MicroAutoBox II Features !\[\]\(e1d6102fe77919492c04879c8450f1f5\_img.jpg\)\)](#)

## Block Description (CUSTOM\_FAILURE\_INTERRUPT)

### Block

Gives you information about the appearance and purpose of the block.



### Purpose

To trigger a subsystem if a custom failure interrupt occurs.

### Description

In the CUSTOM\_FAILURE\_ACTION block, you can specify the failure action to be executed if an error condition is detected. If you specified Interrupt or Interrupt and HW Reset as the failure action, a machine check (MCP) interrupt is generated. An interrupt triggered by the custom monitoring can be made available in your model by using the CUSTOM\_FAILURE\_INTERRUPT block. Additional to triggering a subsystem, you can specify to directly execute the model termination.

Only one CUSTOM\_FAILURE\_INTERRUPT block is allowed in the model.

**I/O characteristics**

The following table describes the ports of the block:


Port	Description
Output	
Int	Provides the generated custom failure interrupt in your model as a trigger output. Data type: Function call

**Dialog pages**

The dialog settings can be specified on the following pages:

- Unit page (refer to [Unit Page \(CUSTOM\\_FAILURE\\_INTERRUPT\)](#) on page 55)

**Related RTLib functions**

This RTI block is implemented using the following RTLib functions. The [MicroAutoBox II RTLib Reference](#)  contains descriptions of these functions.

- ds1401\_fse\_cust\_int\_hook\_fcn\_set

## Unit Page (CUSTOM\_FAILURE\_INTERRUPT)

**Purpose**

To trigger a subsystem if a custom failure interrupt occurs.

**Dialog settings**

**Execute model termination when a custom failure interrupt occurs** Lets you specify to directly execute the model termination process before the real-time application exits.

To add user code to the model termination, you have to add the code to the `usr_terminate` function in the `<model>_usr.c` file.

**Related topics****References**

[Block Description \(CUSTOM\\_FAILURE\\_INTERRUPT\)](#)..... 54





**C**

- challenge-response monitoring
  - overview of the blockset 9
  - specifying monitoring units 13
- Common Program Data folder 6
- CRM\_INTERRUPT 38
  - block description 39
- CRM\_INTERRUPT\_BL
  - Unit page 40
- CRM\_INTERRUPT\_DECODE 40
  - block description 41
- CRM\_RESPONSE\_BL 35
  - block description 35
  - Unit page 36
- CRM\_SUPERVISION\_STATUS 37
  - block description 37
- CRM\_UNIT\_SETUP\_BL 28
  - Advanced page 34
  - block description 29
  - Challenge-Response page 33
  - Parameters page 32
  - Unit page 31
- CUSTOM\_FAILURE\_ACTION 50
  - block description 51
  - Unit page 52
- CUSTOM\_FAILURE\_INTERRUPT 54
  - block description 54
  - Unit page 55

**D**

- Documents folder 7

**L**

- Local Program Data folder 7

**M**

- memory integrity and extras
  - specifying custom failure action 15
  - specifying memory monitoring 14
  - specifying VBAT monitoring 15
- MEMORY\_VBAT\_INTERRUPT 48
  - block description 49
  - Unit page 50
- MEMORY\_VBAT\_MONITORING 43
  - block description 44
  - Unit page 47

**R**

- retriggering watchdog timers 12
- RTI Watchdog Blockset
  - components 10
  - demo model 15
  - library access 10
  - overview 9

**S**

- specifying custom failure action 15

- specifying memory monitoring 14
- specifying monitoring units 13
- specifying VBAT monitoring 15
- specifying watchdogs 11
- stopping watchdogs 12

**W**

- watchdog handling
  - overview of the blockset 9
  - retriggering watchdog timers 12
  - specifying watchdogs 11
  - stopping watchdogs 12
- WATCHDOG\_EXPIRED 26
  - block description 26
- WATCHDOG\_INTERRUPT 23
  - block description 24
  - Unit page 25
- WATCHDOG\_RETRIGGER\_BLx 21
  - block description 21
  - Parameters page 23
- WATCHDOG\_SETUP 18
  - block description 19
  - Parameters page 20

