

## **RTA-OS Centauri M4/GHS V2.0.0**

Port Guide

Status: Released



## Copyright

---

The data in this document may not be altered or amended without special notification from ETAS GmbH. ETAS GmbH undertakes no further obligation in relation to this document. The software described in it can only be used if the customer is in possession of a general license agreement or single license. Using and copying is only allowed in concurrence with the specifications stipulated in the contract. Under no circumstances may any part of this document be copied, reproduced, transmitted, stored in a retrieval system or translated into another language without the express written permission of ETAS GmbH.

© Copyright 2008-2019 ETAS GmbH, Stuttgart.

The names and designations used in this document are trademarks or brands belonging to the respective owners.

**Document: 10820-PG-2.0.0 EN-07-2019**

## **Safety Notice**

---

This ETAS product fulfills standard quality management requirements. If requirements of specific safety standards (e.g. IEC 61508, ISO 26262) need to be fulfilled, these requirements must be explicitly defined and ordered by the customer. Before use of the product, customer must verify the compliance with specific safety standards.

## Contents

---

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	About You . . . . .	7
1.2	Document Conventions . . . . .	8
1.3	References . . . . .	8
<b>2</b>	<b>Installing the RTA-OS Port Plug-in</b>	<b>9</b>
2.1	Preparing to Install . . . . .	9
2.1.1	Hardware Requirements . . . . .	9
2.1.2	Software Requirements . . . . .	9
2.2	Installation . . . . .	10
2.2.1	Installation Directory . . . . .	10
2.3	Licensing . . . . .	11
2.3.1	Installing the ETAS License Manager . . . . .	11
2.3.2	Licenses . . . . .	12
2.3.3	Installing a Concurrent License Server . . . . .	13
2.3.4	Using the ETAS License Manager . . . . .	14
2.3.5	Troubleshooting Licenses . . . . .	16
<b>3</b>	<b>Verifying your Installation</b>	<b>19</b>
3.1	Checking the Port . . . . .	19
3.2	Running the Sample Applications . . . . .	19
<b>4</b>	<b>Port Characteristics</b>	<b>21</b>
4.1	Parameters of Implementation . . . . .	21
4.2	Configuration Parameters . . . . .	21
4.2.1	Stack used for C-startup . . . . .	21
4.2.2	Stack used when idle . . . . .	22
4.2.3	Stack overheads for ISR activation . . . . .	22
4.2.4	Stack overheads for ECC tasks . . . . .	22
4.2.5	Stack overheads for ISR . . . . .	22
4.2.6	Orti22/Lauterbach . . . . .	23
4.2.7	Orti23/Lauterbach . . . . .	23
4.2.8	Block default interrupt . . . . .	23
4.2.9	Enable stack repositioning . . . . .	24
4.2.10	Enable untrusted stack check . . . . .	24
4.2.11	Handle FPU context . . . . .	24
4.3	Generated Files . . . . .	25

<b>5</b>	<b>Port-Specific API</b>	<b>26</b>
5.1	API Calls	26
5.1.1	Os_InitializeVectorTable	26
5.2	Callbacks	26
5.2.1	Os_Cbk_GetAbortStack	26
5.3	Macros	27
5.3.1	CAT1_ISR	27
5.3.2	Os_Clear_x	27
5.3.3	Os_DisableAllConfiguredInterrupts	28
5.3.4	Os_Disable_x	28
5.3.5	Os_EnableAllConfiguredInterrupts	28
5.3.6	Os_Enable_x	28
5.3.7	Os_IntChannel_x	28
5.3.8	Os_IntID_x	29
5.4	Type Definitions	29
5.4.1	Os_StackSizeType	29
5.4.2	Os_StackValueType	29
<b>6</b>	<b>Toolchain</b>	<b>30</b>
6.1	Compiler Versions	30
6.1.1	Green Hills Software, Compiler v2018.1.4	30
6.2	Options used to generate this guide	30
6.2.1	Compiler	30
6.2.2	Assembler	32
6.2.3	Librarian	33
6.2.4	Linker	33
6.2.5	Debugger	34
<b>7</b>	<b>Hardware</b>	<b>35</b>
7.1	Supported Devices	35
7.2	Register Usage	35
7.2.1	Initialization	35
7.2.2	Modification	36
7.3	Required OS resources	37
7.4	Interrupts	37
7.4.1	Interrupt Priority Levels	37
7.4.2	Allocation of ISRs to Interrupt Vectors	38
7.4.3	Vector Table	38
7.4.4	Writing Category 1 Interrupt Handlers	38
7.4.5	Writing Category 2 Interrupt Handlers	39
7.4.6	Default Interrupt	39
7.5	Memory Model	39
7.6	Processor Modes	39
7.7	Stack Handling	40

<b>8</b>	<b>Performance</b>	<b>41</b>
8.1	Measurement Environment . . . . .	41
8.2	RAM and ROM Usage for OS Objects . . . . .	41
8.3	Stack Usage . . . . .	42
8.4	Library Module Sizes . . . . .	42
8.5	Execution Time . . . . .	45
8.5.1	Context Switching Time . . . . .	45
<b>9</b>	<b>Finding Out More</b>	<b>48</b>
<b>10</b>	<b>Contacting ETAS</b>	<b>49</b>
10.1	Technical Support . . . . .	49
10.2	General Enquiries . . . . .	49
10.2.1	ETAS Global Headquarters . . . . .	49
10.2.2	ETAS Local Sales & Support Offices . . . . .	49

# 1 Introduction

---

RTA-OS is a small and fast real-time operating system that conforms to both the AUTOSAR OS (R3.0.1 -> R3.0.7, R3.1.1 -> R3.1.5, R3.2.1 -> R3.2.2, R4.0.1 -> R4.3.1) and OSEK/VDX 2.2.3 standards (OSEK is now standardized in ISO 17356). The operating system is configured and built on a PC, but runs on your target hardware.

This document describes the RTA-OS Centauri M4/GHS port plug-in that customizes the RTA-OS development tools for the STMicroelectronics Centauri with the Green Hills compiler. It supplements the more general information you can find in the *User Guide* and the *Reference Guide*.

The document has two parts. Chapters 2 to 3 help you understand the Centauri M4/GHS port and cover:

- how to install the Centauri M4/GHS port plug-in;
- how to configure Centauri M4/GHS-specific attributes;
- how to build an example application to check that the Centauri M4/GHS port plug-in works.

Chapters 4 to 8 provide reference information including:

- the number of OS objects supported;
- required and recommended toolchain parameters;
- how RTA-OS interacts with the Centauri, including required register settings, memory models and interrupt handling;
- memory consumption for each OS object;
- memory consumption of each API call;
- execution times for each API call.

For the best experience with RTA-OS it is essential that you read and understand this document.

## 1.1 About You

---

You are a trained embedded systems developer who wants to build real-time applications using a preemptive operating system. You should have knowledge of the C programming language, including the compilation, assembling and linking of C code for embedded applications with your chosen toolchain. Elementary knowledge about your target microcontroller, such as the start address, memory layout, location of peripherals and so on, is essential.

You should also be familiar with common use of the Microsoft Windows operating system, including installing software, selecting menu items, clicking buttons, navigating files and folders.

## 1.2 Document Conventions

---

The following conventions are used in this guide:

- |                                    |  |
|------------------------------------|--|
| Choose <b>File &gt; Open</b> .     | Menu options appear in <b>bold, blue</b> characters.   |
| Click <b>OK</b> .                  | Button labels appear in <b>bold</b> characters   |
| Press <Enter>.                     | Key commands are enclosed in angle brackets.   |
| The "Open file" dialog box appears | GUI element names, for example window titles, fields, etc. are enclosed in double quotes.                          |
| Activate(Task1)                    | Program code, header file names, C type names, C functions and API call names all appear in a monospaced typeface. |
| See Section 1.2.                   | Internal document hyperlinks are shown in <b>blue letters</b> .  |



Functionality in RTA-OS that might not be portable to other implementations of AUTOSAR OS is marked with the RTA-OS icon.



Important instructions that you must follow carefully to ensure RTA-OS works as expected are marked with a caution sign.

## 1.3 References

---

OSEK is a European automotive industry standards effort to produce open systems interfaces for vehicle electronics. OSEK is now standardized in ISO 17356. For details of the OSEK standards, please refer to:

<http://www.osek-vdx.org>

AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers. For details of the AUTOSAR standards, please refer to:

<http://www.autosar.org>



## 2 Installing the RTA-OS Port Plug-in

---

### 2.1 Preparing to Install

---

RTA-OS port plug-ins are supplied as a downloadable electronic installation image which you obtain from the ETAS Web Portal. You will have been provided with access to the download when you bought the port. You may optionally have requested an installation CD which will have been shipped to you. In either case, the electronic image and the installation CD contain identical content.



**Integration Guidance 2.1:** *You must have installed the RTA-OS tools before installing the Centauri M4/GHS port plug-in. If you have not yet done this then please follow the instructions in the Getting Started Guide.*

#### 2.1.1 Hardware Requirements

---

You should make sure that you are using at least the following hardware before installing and using RTA-OS on a host PC:

- 1GHz Pentium Windows-capable PC.
- 2G RAM.
- 20G hard disk space.
- CD-ROM or DVD drive (Optional)
- Ethernet card.

#### 2.1.2 Software Requirements

---

RTA-OS requires that your host PC has one of the following versions of Microsoft Windows installed:

- Windows 7
- Windows 8
- Windows 10



**Integration Guidance 2.2:** *The tools provided with RTA-OS require Microsoft's .NET Framework v2.0 (included as part of .NET Framework v3.5) and v4.0 to be installed. You should ensure that these have been installed before installing RTA-OS. The .NET framework is not supplied with RTA-OS but is freely available from <https://www.microsoft.com/net/download>. To install .NET 3.5 on Windows 10 see <https://docs.microsoft.com/en-us/dotnet/framework/install/dotnet-35-windows-10>.*

The migration of the code from v2.0 to v4.0 will occur over a period of time for performance and maintenance reasons.

## 2.2 Installation

---

Target port plug-ins are installed in the same way as the tools:

### 1. Either

- Double click the executable image; or
- Insert the RTA-OS Centauri M4/GHS CD into your CD-ROM or DVD drive.

If the installation program does not run automatically then you will need to start the installation manually. Navigate to the root directory of your CD/DVD drive and double click `autostart.exe` to start the setup.

### 2. Follow the on-screen instructions to install the Centauri M4/GHS port plug-in.

By default, ports are installed into `C:\ETAS\RTA-OS\Targets`. During the installation process, you will be given the option to change the folder to which RTA-OS ports are installed. You will normally want to ensure that you install the port plug-in in the same location that you have installed the RTA-OS tools. You can install different versions of the tools/targets into different directories and they will not interfere with each other.



**Integration Guidance 2.3:** *Port plug-ins can be installed into any location, but using a non-default directory requires the use of the `--target_include` argument to both `rtaosgen` and `rtaoscfg`. For example:*

```
rtaosgen --target_include:<target_directory>
```

### 2.2.1 Installation Directory

---

The installation will create a sub-directory under `Targets` with the name `Centauri M4/GHS_2.0.0`. This contains everything to do with the port plug-in.

Each version of the port installs in its own directory - the trailing `_2.0.0` is the port's version identifier. You can have multiple different versions of the same port installed at the same time and select a specific version in a project's configuration.

The port directory contains:

**Centauri M4/GHS.dll** - the port plug-in that is used by `rtaosgen` and `rtaoscfg`.

**RTA-OS Centauri M4/GHS Port Guide.pdf** - the documentation for the port (the document you are reading now).

**RTA-OS Centauri M4/GHS Release Note.pdf** - the release note for the port. This document provides information about the port plug-in release, including a list of changes from previous releases and a list of known limitations.

There may be other port-specific documentation supplied which you can also find in the root directory of the port installation. All user documentation is distributed in PDF format which can be read using Adobe Acrobat Reader. Adobe Acrobat Reader is not supplied with RTA-OS but is freely available from <http://www.adobe.com>.

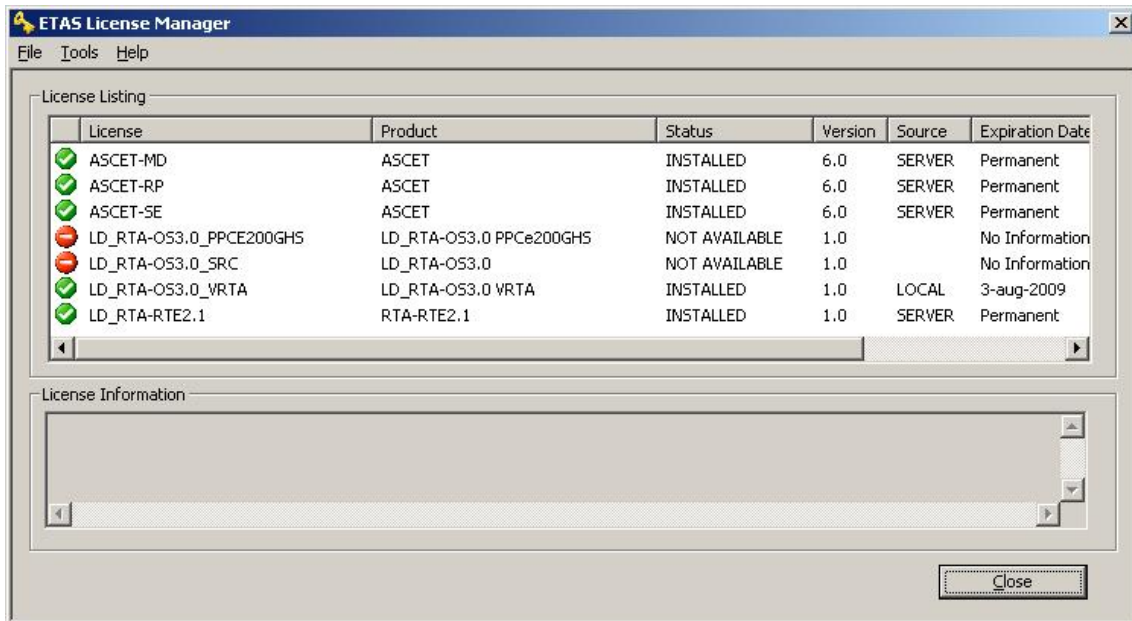


Figure 2.1: The ETAS License manager

## 2.3 Licensing

RTA-OS is protected by FLEXnet licensing technology. You will need a valid license key in order to use RTA-OS.

Licenses for the product are managed using the ETAS License Manager which keeps track of which licenses are installed and where to find them. The information about which features are required for RTA-OS and any port plug-ins is stored as license signature files that are stored in the folder <install\_folder>\bin\Licenses.

The ETAS License Manager can also tell you key information about your licenses including:

- Which ETAS products are installed
- Which license features are required to use each product
- Which licenses are installed
- When licenses expire
- Whether you are using a local or a server-based license

Figure 2.1 shows the ETAS License Manager in operation.

### 2.3.1 Installing the ETAS License Manager



**Integration Guidance 2.4:** *The ETAS License Manager must be installed for RTA-OS to work. It is highly recommended that you install the ETAS License Manager during your installation of RTA-OS.*

The installer for the ETAS License Manager contains two components:

1. the ETAS License Manager itself;
2. a set of re-distributable FLEXnet utilities. The utilities include the software and instructions required to setup and run a FLEXnet license server manager if concurrent licenses are required (see Sections 2.3.2 and 2.3.3 for further details)

During the installation of RTA-OS you will be asked if you want to install the ETAS License Manager. If not, you can install it manually at a later time by running `<install_folder>\LicenseManager\LicensingStandaloneInstallation.exe`.

Once the installation is complete, the ETAS License Manager can be found in `C:\Program Files\Common Files\ETAS\Licensing`.

After it is installed, a link to the ETAS License Manager can be found in the Windows Start menu under **Programs → ETAS → License Management → ETAS License Manager**.

### 2.3.2 Licenses

---

When you install RTA-OS for the first time the ETAS License Manager will allow the software to be used in *grace mode* for 14 days. Once the grace mode period has expired, a license key must be installed. If a license key is not available, please contact your local ETAS sales representative. Contact details can be found in Chapter 10.

You should identify which type of license you need and then provide ETAS with the appropriate information as follows:

**Machine-named licenses** allows RTA-OS to be used by any user logged onto the PC on which RTA-OS and the machine-named license is installed.

A machine-named license can be issued by ETAS when you provide the host ID (Ethernet MAC address) of the host PC

**User-named licenses** allow the named user (or users) to use RTA-OS on any PC in the network domain.

A user-named license can be issued by ETAS when you provide the Windows user-name for your network domain.

**Concurrent licenses** allow any user on any PC up to a specified number of users to use RTA-OS. Concurrent licenses are sometimes called *floating* licenses because the license can *float* between users.

A concurrent license can be issued by ETAS when you provide the following information:

1. The name of the server
2. The Host ID (MAC address) of the server.
3. The TCP/IP port over which your FLEXnet license server will serve licenses. A default installation of the FLEXnet license server uses port 27000.



Figure 2.2: Obtaining License Information

You can use the ETAS License Manager to get the details that you must provide to ETAS when requesting a machine-named or user-named license and (optionally) store this information in a text file.

Open the ETAS License Manager and choose **Tools → Obtain License Info** from the menu. For machine-named licenses you can then select the network adaptor which provides the Host ID (MAC address) that you want to use as shown in Figure 2.2. For a user-based license, the ETAS License Manager automatically identifies the Windows username for the current user.

Selecting “Get License Info” tells you the Host ID and User information and lets you save this as a text file to a location of your choice.

### 2.3.3 Installing a Concurrent License Server

Concurrent licenses are allocated to client PCs by a FLEXnet license server manager working together with a vendor daemon. The vendor daemon for ETAS is called ETAS.exe. A copy of the vendor daemon is placed on disk when you install the ETAS License Manager and can be found in:

C:\Program Files\Common Files\ETAS\Licensing\Utility

To work with an ETAS concurrent license, a license server must be configured which is accessible from the PCs wishing to use a license. The server must be configured with the following software:

- FLEXnet license server manager;
- ETAS vendor daemon (ETAS.exe);

It is also necessary to install your concurrent license on the license server.

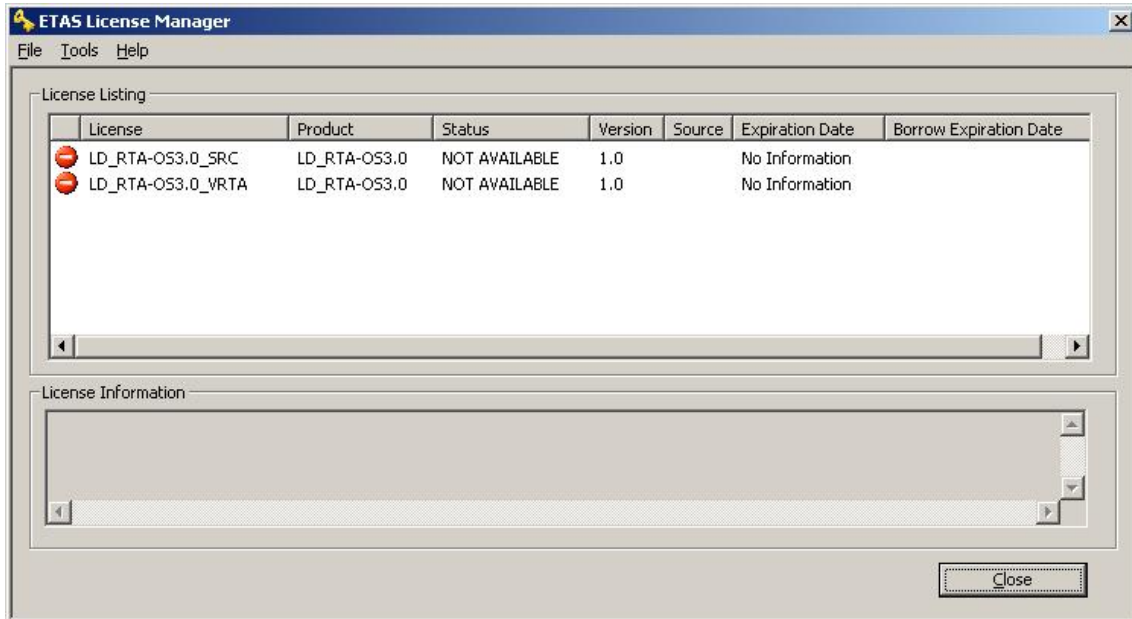


Figure 2.3: Unlicensed RTA-OS Installation

In most organizations there will be a single FLEXnet license server manager that is administered by your IT department. You will need to ask your IT department to install the ETAS vendor daemon and the associated concurrent license.

If you do not already have a FLEXnet license server then you will need to arrange for one to be installed. A copy of the FLEXnet license server, the ETAS vendor daemon and the instructions for installing and using the server (LicensingEndUserGuide.pdf) are placed on disk when you install the ETAS License manager and can be found in:

C:\Program Files\Common Files\ETAS\Licensing\Utility

#### 2.3.4 Using the ETAS License Manager

If you try to run the RTA-OS GUI **rtaoscfg** without a valid license, you will be given the opportunity to start the ETAS License Manager and select a license. (The command-line tool **rtaosgen** will just report the license is not valid.)

When the ETAS License Manager is launched, it will display the RTA-OS license state as NOT AVAILABLE. This is shown in Figure 2.3.

Note that if the ETAS License Manager window is slow to start, **rtaoscfg** may ask a second time whether you want to launch it. You should ignore the request until the ETAS License Manager has opened and you have completed the configuration of the licenses. You should then say yes again, but you can then close the ETAS License Manager and continue working.

## License Key Installation

---

License keys are supplied in an ASCII text file, which will be sent to you on completion of a valid license agreement.

If you have a machine-based or user-based license key then you can simply install the license by opening the ETAS License Manager and selecting **File → Add License File** menu.

If you have a concurrent license key then you will need to create a license stub file that tells the client PC to look for a license on the FLEXnet server as follows:

1. create a copy of the concurrent license file
2. open the copy of the concurrent license file and delete every line *except* the one starting with SERVER
3. add a new line containing USE\_SERVER
4. add a blank line
5. save the file

The file you create should look something like this:

```
SERVER <server name> <MAC address> <TCP/IP Port>¶  
USE_SERVER¶  
¶
```

Once you have create the license stub file you can install the license by opening the ETAS License Manager and selecting **File → Add License File** menu and choosing the license stub file.

## License Key Status

---

When a valid license has been installed, the ETAS License Manager will display the license version, status, expiration date and source as shown in Figure 2.4.

## Borrowing a concurrent license

---

If you use a concurrent license and need to use RTA-OS on a PC that will be disconnected from the network (for example, you take a demonstration to a customer site), then the concurrent license will not be valid once you are disconnected.

To address this problem, the ETAS License Manager allows you to temporarily borrow a license from the license server.

To borrow a license:

1. Right click on the license feature you need to borrow.
2. Select "Borrow License"
3. From the calendar, choose the date that the borrowed license should expire.
4. Click "OK"

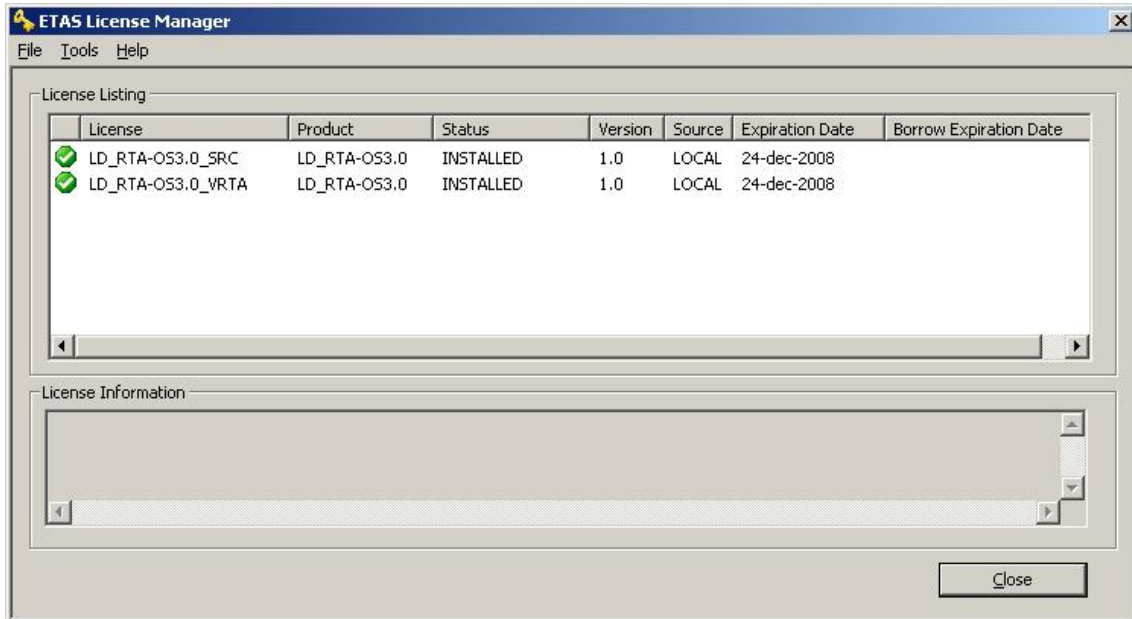


Figure 2.4: Licensed features for RTA-OS

The license will automatically expire when the borrow date elapses. A borrowed license can also be returned before this date. To return a license:

1. Reconnect to the network;
2. Right-click on the license feature you have borrowed;
3. Select "Return License".

### 2.3.5 Troubleshooting Licenses

RTA-OS tools will report an error if you try to use a feature for which a correct license key cannot be found. If you think that you should have a license for a feature but the RTA-OS tools appear not to work, then the following troubleshooting steps should be followed before contacting ETAS:

#### **Can the ETAS License Manager see the license?**

The ETAS License Manager must be able to see a valid license key for each product or product feature you are trying to use.

You can check what the ETAS License Manager can see by starting it from the **Help → License Manager...** menu option in **rtaoscfg** or directly from the Windows Start Menu - **Start → ETAS → License Management → ETAS License Manager**.

The ETAS License Manager lists all license features and their status. Valid licenses have status **INSTALLED**. Invalid licenses have status **NOT AVAILABLE**.



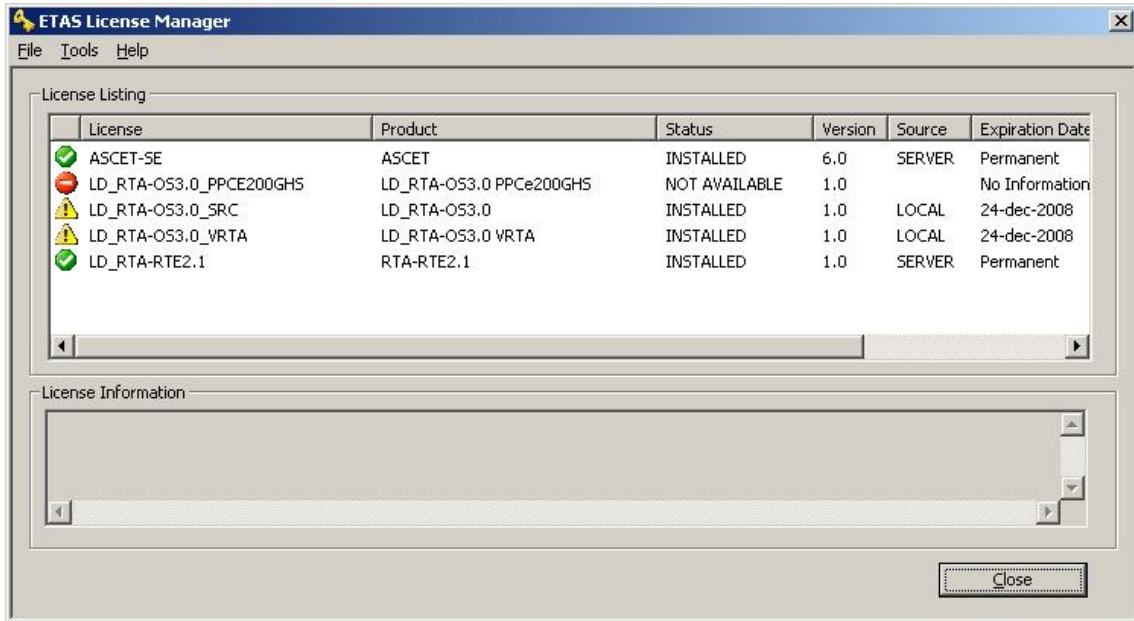


Figure 2.5: Licensed features that are due to expire

**Is the license valid?**

You may have been provided with a time-limited license (for example, for evaluation purposes) and the license may have expired. You can check that the Expiration Date for your licensed features to check that it has not elapsed using the ETAS License Manager.

If a license is due to expire within the next 30 days, the ETAS License Manager will use a warning triangle to indicate that you need to get a new license. Figure 2.5 shows that the license features LD\_RTА-0S3.0\_VRTA and LD\_RTА-0S3.0\_SRC are due to expire.

If your license has elapsed then please contact your local ETAS sales representative to discuss your options.

**Does the Ethernet MAC address match the one specified?**

If you have a machine based license then it is locked to a specific MAC address. You can find out the MAC address of your PC by using the ETAS License Manager (**Tools → Obtain License Info**) or using the Microsoft program **ipconfig /all** at a Windows Command Prompt.

You can check that the MAC address in your license file by opening your license file in a text editor and checking that the HOSTID matches the MAC address identified by the ETAS License Manager or the *Physical Address* reported by **ipconfig /all**.

If the HOSTID in the license file (or files) does not match your MAC address then you do not have a valid license for your PC. You should contact your local ETAS sales representative to discuss your options.

**Is your Ethernet Controller enabled?**

If you use a laptop and RTA-OS stops working when you disconnect from the network then you should check your hardware settings to ensure that your Ethernet controller is not turned off to save power when a network connection is not present. You can do this using Windows Control Panel. Select **System → Hardware → Device Manager** then select your Network Adapter. Right click to open **Properties** and check that the Ethernet controller is not configured for power saving in **Advanced** and/or **Power Management** settings.

**Is the FlexNet License Server visible?**

If your license is served by a FlexNet license server, then the ETAS License Manager will report the license as NOT AVAILABLE if the license server cannot be accessed.

You should contact your IT department to check that the server is working correctly.

**Still not fixed?**

If you have not resolved your issues, after confirming these points above, please contact ETAS technical support. The contact address is provided in Section [10.1](#). You must provide the contents and location of your license file and your Ethernet MAC address.

## 3 Verifying your Installation

---

Now that you have installed both the RTA-OS tools and a port plug-in and have obtained and installed a valid license key you can check that things are working.

### 3.1 Checking the Port

---

The first thing to check is that the RTA-OS tools can see the new port. You can do this in two ways:

1. use the **rtaosgen** tool

You can run the command **rtaosgen --target:?** to get a list of available targets, the versions of each target and the variants supported, for example:

```
RTA-OS Code Generator
Version p.q.r.s, Copyright © ETAS nnnn
Available targets:
  TriCoreHighTec_n.n.n [TC1797...]
  VRTA_n.n.n [MinGW,VS2005,VS2008,VS2010]
```

2. use the **rtaoscfg** tool

The second way to check that the port plug-in can be seen is by starting **rtaoscfg** and selecting **Help → Information...** drop down menu. This will show information about your complete RTA-OS installation and license checks that have been performed.



**Integration Guidance 3.1:** *If the target port plug-ins have been installed to a non-default location, then the `--target_include` argument must be used to specify the target location.*

If the tools can see the port then you can move on to the next stage – checking that you can build an RTA-OS library and use this in a real program that will run on your target hardware.

### 3.2 Running the Sample Applications

---

Each RTA-OS port is supplied with a set of sample applications that allow you to check that things are running correctly. To generate the sample applications:

1. Create a new *working* directory in which to build the sample applications.
2. Open a Windows command prompt in the new directory.
3. Execute the command:

```
rtaosgen --target:<your target> --samples:[Applications]
```

e.g.

```
rtaosgen --target:[MPC5777Mv2]PPCe200HighTec_5.0.8
--samples:[Applications]
```

You can then use the build.bat and run.bat files that get created for each sample application to build and run the sample. For example:

```
cd Samples\Applications\HelloWorld
build.bat
run.bat
```

Remember that your target toolchain must be accessible on the Windows PATH for the build to be able to run successfully.



**Integration Guidance 3.2:** *It is strongly recommended that you build and run at least the Hello World example in order to verify that RTA-OS can use your compiler toolchain to generate an OS kernel and that a simple application can run with that kernel.*

For further advice on building and running the sample applications, please consult your *Getting Started Guide*.

## 4 Port Characteristics

This chapter tells you about the characteristics of RTA-OS for the Centauri M4/GHS port.

### 4.1 Parameters of Implementation

To be a valid OSEK (ISO 17356) or AUTOSAR OS, an implementation must support a minimum number of OS objects. The following table specifies the *minimum* numbers of each object required by the standards and the *maximum* number of each object supported by RTA-OS for the Centauri M4/GHS port.

Parameter	Required	RTA-OS
Tasks	16	1024
Tasks not in SUSPENDED state	16	1024
Priorities	16	1024
Tasks per priority	-	1024
Queued activations per priority	-	4294967296
Events per task	8	32
Software Counters	8	4294967296
Hardware Counters	-	4294967296
Alarms	1	4294967296
Standard Resources	8	4294967296
Linked Resources	-	4294967296
Nested calls to GetResource()	-	4294967296
Internal Resources	2	no limit
Application Modes	1	4294967296
Schedule Tables	2	4294967296
Expiry Points per Schedule Table	-	4294967296
OS Applications	-	4294967295
Trusted functions	-	4294967295
Spinlocks (multicore)	-	4294967295
Register sets	-	4294967296

### 4.2 Configuration Parameters

Port-specific parameters are configured in the **General** → **Target** workspace of **rtaoscfg**, under the “Target-Specific” tab.

The following sections describe the port-specific configuration parameters for the Centauri M4/GHS port, the name of the parameter as it will appear in the XML configuration and the range of permitted values (where appropriate).

#### 4.2.1 Stack used for C-startup

**XML name** SpPreStartOS

**Description**

The amount of stack already in use at the point that StartOS() is called. This value is simply added to the total stack size that the OS needs to support all tasks and interrupts at run-time. Typically you use this to obtain the amount of stack that the linker must allocate. The value does not normally change if the OS configuration changes.

4.2.2 Stack used when idle

---

**XML name** SpStartOS

**Description**

The amount of stack used when the OS is in the idle state (typically inside Os\_Cbk\_Idle()). This is just the difference between the stack used at the point that Os\_StartOS() is called and the stack used when no task or interrupt is running. This can be zero if Os\_Cbk\_Idle() is not used. It must include the stack used by any function called while in the idle state. The value does not normally change if the OS configuration changes.

4.2.3 Stack overheads for ISR activation

---

**XML name** SpIDisp

**Description**

The extra amount of stack needed to activate a task from within an ISR. If a task is activated within a Category 2 ISR, and that task has a higher priority than any currently running task, then for some targets the OS may need to use marginally more stack than if it activates a task that is of lower priority. This value accounts for that. On most targets this value is zero. This value is used in worst-case stack size calculations. The value may change if significant changes are made to the OS configuration. e.g. STANDARD/EXTENDED, SC1/2/3/4.

4.2.4 Stack overheads for ECC tasks

---

**XML name** SpECC

**Description**

The extra amount of stack needed to start an ECC task. ECC tasks need to save slightly more state on the stack when they are started than BCC tasks. This value contains the difference. The value may change if significant changes are made to the OS configuration. e.g. STANDARD/EXTENDED, SC1/2/3/4.

4.2.5 Stack overheads for ISR

---

**XML name** SpPreemption

**Description**

The amount of stack used to service a Category 2 ISR. When a Category 2 ISR interrupts a task, it usually places some data on the stack. If the ISR measures the stack to determine if the preempted task has exceeded its stack budget, then it will overestimate the stack usage unless this value is subtracted from the measured size. The value is also used when calculating the worst-case stack usage of the system. Be careful to set this value accurately. If its value is too high then when the subtraction occurs, 32-bit underflow can occur and cause the OS to think that a budget overrun has been detected. The value may change if significant changes are made to the OS configuration. e.g. STANDARD/EXTENDED, SC1/2/3/4.

4.2.6 Orti22/Lauterbach

---

**XML name** Orti22Lauterbach

**Description**

Select ORTI 2.2 generation for the Lauterbach debugger.

**Settings**

Value	Description
<b>true</b>	Generate ORTI
<b>false</b>	No ORTI (default)

4.2.7 Orti23/Lauterbach

---

**XML name** Orti23Lauterbach

**Description**

Select ORTI 2.3 generation for the Lauterbach debugger.

**Settings**

Value	Description
<b>true</b>	Generate ORTI
<b>false</b>	No ORTI (default)

4.2.8 Block default interrupt

---

**XML name** block\_default\_interrupt

**Description**

Where a default interrupt is specified, it will normally execute if an unexpected interrupt fires. You can block this behavior using this option. The option affects the enabling of unused interrupt sources.

**Settings**

Value	Description
<b>true</b>	Block the default interrupt
<b>false</b>	Allow the default interrupt handler to run if an unexpected interrupt fires (default)

4.2.9 Enable stack repositioning

---

**XML name** AlignUntrustedStacks

**Description**

Use to support realignment of the stack for untrusted code when there are MPU protection region granularity issues. Refer to the documentation for Os\_Cbk\_SetMemoryAccess.

**Settings**

Value	Description
<b>true</b>	Support repositioning
<b>false</b>	Normal behavior (default)

4.2.10 Enable untrusted stack check

---

**XML name** DistrustStacks

**Description**

Extra code can be placed in interrupt handlers to detect when untrusted code has an illegal stack pointer value. Also exception handlers may run on a private stack (Refer to the documentation for Os\_Cbk\_GetAbortStack). This has a small performance overhead, so is made optional.

**Settings**

Value	Description
<b>true</b>	Perform the checks
<b>false</b>	Do not check (default)

4.2.11 Handle FPU context

---

**XML name** handle\_FPU\_context

**Description**

Extra code can be added during context switches to preserve floating point registers used by tasks and ISRs. This option should only be used if the majority of tasks and ISRs in the application contain FPU instructions otherwise the register set mechanism may be more efficient.



### Settings

Value	Description
<b>true</b>	Save FPU context
<b>false</b>	Do not save FPU context (default)

### 4.3 Generated Files

The following table lists the files that are generated by **rtaosgen** for all ports:

Filename	Contents
Os.h	The main include file for the OS.
Os_Cfg.h	Declarations of the objects you have configured. This is included by Os.h.
Os_MemMap.h	AUTOSAR memory mapping configuration used by RTA-OS to merge with the system-wide MemMap.h file in AUTOSAR versions 4.0 and earlier. From AUTOSAR version 4.1, Os_MemMap.h is used by the OS instead of MemMap.h.
RTAOS.<lib>	The RTA-OS library for your application. The extension <lib> depends on your target.
RTAOS.<lib>.sig	A signature file for the library for your application. This is used by <b>rtaosgen</b> to work out which parts of the kernel library need to be rebuilt if the configuration has changed. The extension <lib> depends on your target.
<projectname>.log	A log file that contains a copy of the text that the tool and compiler sent to the screen during the build process.

## 5 Port-Specific API

---

The following sections list the port-specific aspects of the RTA-OS programmers reference for the Centauri M4/GHS port that are provided either as:

- additions to the material that is documented in the *Reference Guide*; or
- overrides for the material that is documented in the *Reference Guide*. When a definition is provided by both the *Reference Guide* and this document, the definition provided in this document takes precedence.

### 5.1 API Calls

---

#### 5.1.1 Os\_InitializeVectorTable

---

Initialize the vector table.

##### Syntax

```
void Os_InitializeVectorTable(void)
```

##### Description

Os\_InitializeVectorTable() initializes the NVIC and other control registers according to the project configuration. If RTA-OS has been configured to generate a vector table, it also sets the VTOR register to the address of the OS-generated vector table that has entries for all the appropriate CPU exceptions and interrupts. It enables Category 1 ISRs but leaves Category 2 ISRs disabled until after calling StartOS().

Os\_InitializeVectorTable() should be called before StartOS(). It should be called even if 'Suppress Vector Table Generation' is set to TRUE.

##### Example

```
Os_InitializeVectorTable();
```

##### See Also

StartOS

### 5.2 Callbacks

---

#### 5.2.1 Os\_Cbk\_GetAbortStack

---

Callback routine to provide the start address of the stack to use for some exception conditions.

##### Syntax

```
FUNC(void *, OS_APPL_CODE) Os_Cbk_GetAbortStack(void)
```

##### Return Values

The call returns values of type **void** \*.

## Description

Untrusted code can misbehave and cause a protection exception. When this happens, AUTOSAR requires that ProtectionHook is called and the task, ISR or OS Application must be terminated.

It is possible that at the time of the fault the stack pointer is invalid. For this reason, if 'Enable untrusted stack check' is configured, RTA-OS will call Os\_Cbk\_GetAbortStack to get the address of a safe area of memory that it should use for the stack while it performs this processing.

If the value zero is returned, the stack does not get adjusted.

The stack only needs to be large enough to perform the ProtectionHook. Maskable interrupts are disabled during this process.

A default implementation of Os\_Cbk\_GetAbortStack is supplied in the RTA-OS library that returns the address of an area of static memory.

## Example

```
FUNC(void *, OS_APPL_CODE) Os_Cbk_GetAbortStack(void) {
    /* 64-bit alignment is needed for EABI. */
    static long long abortstack[18U];
    return &abortstack[18U];
}
```

## Required when

The callback must be present if 'Enable untrusted stack check' is configured and there are untrusted OS Applications.

## 5.3 Macros

---

### 5.3.1 CAT1\_ISR

---

Macro that should be used to create a Category 1 ISR entry function. This macro exists to help make your code portable between targets.

## Example

```
CAT1_ISR(MyISR) { ... }
```

### 5.3.2 Os\_Clear\_x

---

The Os\_Clear\_x() macro can be used to clear the interrupt pending bit for a specified NVIC interrupt. The interrupt can be identified using either the NVIC channel number or the RTA-OS configured vector name. The macro is only available if you #include the file Os\_ConfigInterrupts.h. The macro may not be used by untrusted code.

**Example**

```
Os_Clear_IntChannel_20();
Os_Clear_Millisecond();
```

### 5.3.3 Os\_DisableAllConfiguredInterrupts

---

The `Os_DisableAllConfiguredInterrupts()` can be used to disable all configured NVIC interrupts. The macro is only available if you `#include` the file `Os_ConfigInterrupts.h`. The macro may not be used by untrusted code.

**Example**

```
Os_DisableAllConfiguredInterrupts();
```

### 5.3.4 Os\_Disable\_x

---

The `Os_Disable_x()` macro can be used to disable a specified NVIC interrupt. The interrupt can be identified using either the NVIC channel number or the RTA-OS configured vector name. The macro is only available if you `#include` the file `Os_ConfigInterrupts.h`. The macro may not be used by untrusted code.

**Example**

```
Os_Disable_IntChannel_20();
Os_Disable_Millisecond();
```

### 5.3.5 Os\_EnableAllConfiguredInterrupts

---

The `Os_EnableAllConfiguredInterrupts()` can be used to enable all configured NVIC interrupts. The macro is only available if you `#include` the file `Os_ConfigInterrupts.h`. The macro may not be used by untrusted code.

**Example**

```
Os_EnableAllConfiguredInterrupts();
```

### 5.3.6 Os\_Enable\_x

---

The `Os_Enable_x()` macro can be used to enable a specified NVIC interrupt. The interrupt can be identified using either the NVIC channel number or the RTA-OS configured vector name. The macro is only available if you `#include` the file `Os_ConfigInterrupts.h`. The macro may not be used by untrusted code.

**Example**

```
Os_Enable_IntChannel_20();
Os_Enable_Millisecond();
```

### 5.3.7 Os\_IntChannel\_x

---

The `Os_IntChannel_x` macro can be used to get the interrupt channel number for a named NVIC interrupt. The channel number is the index into the NVIC hardware, so it is 16 less than the associated `IntID`. The `Os_IntChannel_x` macro is only available if you `#include` the file `Os_ConfigInterrupts.h`.

**Example**

```
trigger_nvic_interrupt(Os_IntChannel_Millisecond);
```

5.3.8 Os\_IntID\_x

The `Os_IntID_x` macro can be used to get the CPU INTID value for a named interrupt. It is available for CPU and NVIC interrupts. The `Os_IntID_x` macro is only available if you `#include` the file `Os_ConfigInterrupts.h`.

**Example**

```
trigger_cpu_interrupt(Os_IntID_Millisecond);
```

5.4 Type Definitions

5.4.1 Os\_StackSizeType

An unsigned value representing an amount of stack in bytes.

**Example**

```
Os_StackSizeType stack_size;  
stack_size = Os_GetStackSize(start_position, end_position);
```

5.4.2 Os\_StackValueType

An unsigned value representing the position of the stack pointer.

**Example**

```
Os_StackValueType start_position;  
start_position = Os_GetStackValue();
```

## 6 Toolchain

---

This chapter contains important details about RTA-OS and the Green Hills toolchain. A port of RTA-OS is specific to both the target hardware and a specific version of the compiler toolchain. You must make sure that you build your application with the supported toolchain.

In addition to the version of the toolchain, RTA-OS may use specific tool options (switches). The options are divided into three classes:

**kernel** options are those used by **rtaosgen** to build the RTA-OS kernel.

**mandatory** options must be used to build application code so that it will work with the RTA-OS kernel.

**forbidden** options must not be used to build application code.

Any options that are not explicitly forbidden can be used by application code providing that they do not conflict with the kernel and mandatory options for RTA-OS.

**Integration Guidance 6.1:** *ETAS has developed and tested RTA-OS using the tool versions and options indicated in the following sections. Correct operation of RTA-OS is only covered by the warranty in the terms and conditions of your deployment license agreement when using identical versions and options. If you choose to use a different version of the toolchain or an alternative set of options then it is your responsibility to check that the system works correctly. If you require a statement that RTA-OS works correctly with your chosen tool version and options then please contact ETAS to discuss validation possibilities.*



### 6.1 Compiler Versions

---

This port of RTA-OS has been developed to work with the following compiler(s):

#### 6.1.1 Green Hills Software, Compiler v2018.1.4

---

Ensure that ccarm.exe is on the path.

**Tested on** Green Hills Software, Compiler v2018.1.4 Release Date: Wed Apr 18 01:00:05 2018

**See also** [http://www.ghs.com/products/arm\\_development.html](http://www.ghs.com/products/arm_development.html)

If you require support for a compiler version not listed above, please contact ETAS.

### 6.2 Options used to generate this guide

---

#### 6.2.1 Compiler

---

**Name** cctthumb.exe

**Version** v2018.1.4

## Options

---

### Kernel Options

The following options were used to build the RTA-OS kernel for the configuration that was used to generate the performance figures in this document. If you select different target options, then the values used to build the kernel might change. You can run a Configuration Summary report to check the values used for your configuration.

- c Compile only
- Wundef Warn for undefined preprocessor symbols
- Wimplicit-int Warn for implicit return type
- diag\_error=193,940 Messages 193 and 940 are errors
- diag\_warning=2003 Message 2003 is warning
- no\_wrap\_diagnostics Controls diagnostic messages
- no\_vla No variable length arrays
- Osize Optimize for size
- Omax Max optimizations
- no\_commons Disable commons
- no\_discard\_zero\_initializers Place zero initializers in data sections
- gnu\_asm Allow GNU extended asm
- preprocess\_assembly\_files Preprocess assembly files
- passsource Source in listings (requires appropriate license)
- split\_data\_sections\_by\_alignment Use aligned data sections
- individual\_data\_sections Use individual data sections
- individual\_pragma\_data\_sections Use individual renamed data sections
- individual\_function\_sections Use individual function sections
- individual\_pragma\_function\_sections Use individual renamed function sections
- individual\_attribute\_data\_sections Use individual user defined data sections
- individual\_attribute\_function\_sections Use individual user function data sections
- individual\_section\_name\_extra\_dot Use extra dot in section names
- g Plain debugging

- dwarf2** Dwarf 2 debugging
- cpu=cortexm4f** Select the CPU type
- thumb** Thumb mode
- thumb\_lib** Thumb libraries
- align8** 8 byte data alignment
- X5261** Customer required option
- unknown\_pragma\_errors** Treat unknown pragma as error
- incorrect\_pragma\_errors** Treat incorrect pragma as error

### Mandatory Options for Application Code

The following options were mandatory for application code used with the configuration that was used to generate the performance figures in this document. If you select different target options, then the values required by application code might change. You can run a Configuration Summary report to check the values used for your configuration.

- Options compatible with those used for kernel

### Forbidden Options for Application Code

The following options were forbidden for application code used with the configuration that was used to generate the performance figures in this document. If you select different target options, then the forbidden values might change. You can run a Configuration Summary report to check the values used for your configuration.

- bigendian** Generate big-endian code
- std=x** Other C standard code apart from C99
- Any other options that conflict with kernel options

## 6.2.2 Assembler

---

**Name** cctthumb.exe  
**Version** v2018.1.4

Options

---

### Kernel Options

The following options were used to build the RTA-OS kernel for the configuration that was used to generate the performance figures in this document. If you select different target options, then the values used to build the kernel might change. You can run a Configuration Summary report to check the values used for your configuration.

- The same options as for compilation



### Mandatory Options for Application Code

The following options were mandatory for application code used with the configuration that was used to generate the performance figures in this document. If you select different target options, then the values required by application code might change. You can run a Configuration Summary report to check the values used for your configuration.

- The same options as for compilation

### Forbidden Options for Application Code

The following options were forbidden for application code used with the configuration that was used to generate the performance figures in this document. If you select different target options, then the forbidden values might change. You can run a Configuration Summary report to check the values used for your configuration.

- Any options that conflict with kernel options

#### 6.2.3 Librarian

---

**Name** ax.exe  
**Version** v2018.1.4

#### 6.2.4 Linker

---

**Name** elxr.exe  
**Version** v2018.1.4

Options

---

#### Kernel Options

The following options were used to build the RTA-OS kernel for the configuration that was used to generate the performance figures in this document. If you select different target options, then the values used to build the kernel might change. You can run a Configuration Summary report to check the values used for your configuration.

- globalcheck=normal** Normal checking on types
- map** Emit map file
- mapfile\_type=2** Detailed map file
- Man** Sort alphabetic and numeric
- Ml** Include locals in map file
- Mx** Include cross-reference in map
- Mu** Display unused functions in map

- keepmap** Keep map if link fails
- delete** Remove unreferenced functions
- data\_delete** Remove unreferenced data
- ignore\_debug\_references** Ignore debug relocations

### Mandatory Options for Application Code

The following options were mandatory for application code used with the configuration that was used to generate the performance figures in this document. If you select different target options, then the values required by application code might change. You can run a Configuration Summary report to check the values used for your configuration.

- The same options as for kernel

### Forbidden Options for Application Code

The following options were forbidden for application code used with the configuration that was used to generate the performance figures in this document. If you select different target options, then the forbidden values might change. You can run a Configuration Summary report to check the values used for your configuration.

- Any options that conflict with kernel options

## 6.2.5 Debugger

---

**Name** Lauterbach TRACE32 for ARM64  
**Version** Build 109980 or later

## 7 Hardware

---

### 7.1 Supported Devices

---

This port of RTA-OS has been developed to work with the following target:

**Name:** STMicroelectronics  
**Device:** Centauri

The following variants of the Centauri are supported:

- DME\_Cut1 (Centauri Data Move Engine)
- DSP-H\_Cut1 (Centauri DSP H)
- HSM\_Cut1 (Centauri Hardware Security Module)

If you require support for a variant of Centauri not listed above, please contact ETAS.

### 7.2 Register Usage

---

#### 7.2.1 Initialization

---

RTA-OS requires the following registers to be initialized to the indicated values before `StartOS()` is called.

Register	Setting
CCR.NONBASETHRDENA	This must be one so that the processor can enter Thread mode with exceptions active. Calling <code>Os_InitializeVectorTable()</code> will do this for you.
CCR.STKALIGN	This must be one so that 8-byte stack alignment is forced for exceptions. Calling <code>Os_InitializeVectorTable()</code> will do this for you.
CONTROL.SPSEL	This must be zero so that <code>SP_main</code> is used in thread mode. Calling <code>Os_InitializeVectorTable()</code> will do this for you.
CONTROL.nPRIV	This must be zero so that <code>StartOS()</code> is entered in privileged mode. Calling <code>Os_InitializeVectorTable()</code> will do this for you.
CPACR/FPCCR	These need to be initialized appropriately if floating point features are used.
FAULTMASK	This must be cleared before <code>StartOS()</code> . Calling <code>Os_InitializeVectorTable()</code> will do this for you.
IPSR	This must be cleared before <code>StartOS()</code> . Calling <code>Os_InitializeVectorTable()</code> will do this for you.
PRIMASK	This must be cleared before <code>StartOS()</code> . Calling <code>Os_InitializeVectorTable()</code> will do this for you.
SP (R13)	Must be set to the area of memory to be used for the stack before calling <code>StartOS()</code> .
VTOR	This must be setup before <code>StartOS()</code> . Calling <code>Os_InitializeVectorTable()</code> will do this for you, except when you are generating your own vector table.

### 7.2.2 Modification

The following registers must not be modified by user code after the call to `StartOS()`:

Register	Notes
ACTLR	User code may not change the values in the ACTLR register.
BASEPRI	User code may not make changes to BASEPRI other than via OS APIs.
CCR STKALIGN/NONBASETHRENA	User code may not make changes to these bits.
CONTROL	User code may not change the values in the CONTROL register.
FAULTMASK	User code may not make changes to FAULTMASK other than via OS APIs.
FPCCR	User code may not make direct changes to FPCCR.
NVIC	User code may not make changes to the interrupt controller (NVIC) other than via OS APIs.
PRIMASK	User code may not make changes to PRIMASK other than via OS APIs.
SHCSR	User code may not make changes to SHCSR.
SHPRx	User code may not make changes to SHPRx.
SP (R13)	User code may not change the stack pointer other than as a result of normal program flow.

### 7.3 Required OS resources

RTA-OS needs the following resources for correct operation.

Resource	Description
SVCALL	If untrusted code is present, the OS will use SVCALL to move from unprivileged to privileged mode. It is also used when handling exceptions. If overridden by user-code, its behavior must be replicated.

### 7.4 Interrupts

This section explains the implementation of RTA-OS’s interrupt model on the Centauri.

#### 7.4.1 Interrupt Priority Levels

Interrupts execute at an interrupt priority level (IPL). RTA-OS standardizes IPLs across all targets. IPL 0 indicates task level. IPL 1 and higher indicate an interrupt priority. It is important that you don’t confuse IPLs with task priorities. An IPL of 1 is higher than the highest task priority used in your application.

The IPL is a target-independent description of the interrupt priority on your target hardware. The following table shows how IPLs are mapped onto the hardware interrupt priorities of the Centauri:

IPL	BASEPRI	Description
0	BASEPRI = 0x0	User (task) level
1-15	BASEPRI = 0xF0-0x10	Category 1 and 2 interrupts. Configurable priority traps.
16	-	HardFault. Not affected by DisableAllInterrupts.
17	-	NMI. Not affected by DisableAllInterrupts.

Even though a particular mapping is permitted, all Category 1 ISRs must have equal or higher IPL than all of your Category 2 ISRs.

#### 7.4.2 Allocation of ISRs to Interrupt Vectors

The following restrictions apply for the allocation of Category 1 and Category 2 interrupt service routines (ISRs) to interrupt vectors on the Centauri. A ✓ indicates that the mapping is permitted and a ✗ indicates that it is not permitted:

Address	Category 1	Category 2
NMI (Vector 2)	✓	✗
HardFault (Vector 3)	✓	✗
MemManage (Vector 4)	✓	✗
BusFault (Vector 5)	✓	✗
UsageFault (Vector 6)	✓	✗
SVCALL (Vector 11)	✓	✗
DebugMon (Vector 12)	✓	✗
PendSV (Vector 14)	✓	✗
SysTick (Vector 15)	✓	✓
NVIC interrupts (Vectors 16+)	✓	✓

#### 7.4.3 Vector Table

**rtaosgen** normally generates an interrupt vector table for you automatically. You can configure “Suppress Vector Table Generation” as true to stop RTA-OS from generating the interrupt vector table.

Depending upon your target, you may be responsible for locating the generated vector table at the correct base address. The following table shows the section (or sections) that need to be located and the associated valid base address:

Section	Valid Addresses
.os_vectors	Section containing Os_Vectors. First entry is the address placed into VTOR.
.os_text	Section containing exception handling code

#### 7.4.4 Writing Category 1 Interrupt Handlers

Raw Category 1 interrupt service routines (ISRs) must correctly handle the interrupt context themselves. RTA-OS provides an optional helper macro CAT1\_ISR that can be

used to make code more portable. Depending on the target, this may cause the selection of an appropriate interrupt control directive to indicate to the compiler that a function requires additional code to save and restore the interrupt context.

A Category 1 ISR therefore has the same structure as a Category 2 ISR, as shown below.

```
CAT1_ISR(Category1Handler) {
    /* Handler routine */
}
```

#### 7.4.5 Writing Category 2 Interrupt Handlers

Category 2 ISRs are provided with a C function context by RTA-OS, since the RTA-OS kernel handles the interrupt context itself. The handlers are written using the ISR() macro as shown below:

```
#include <Os.h>
ISR(MyISR) {
    /* Handler routine */
}
```

You must not insert a return from interrupt instruction in such a function. The return is handled automatically by RTA-OS.

#### 7.4.6 Default Interrupt

The 'default interrupt' is intended to be used to catch all unexpected interrupts. All unused interrupts have their interrupt vectors directed to the named routine that you specify. The routine you provide is not handled by RTA-OS and must correctly handle the interrupt context itself. The handler must use the CAT1\_ISR macro in the same way as a Category 1 ISR (see Section 7.4.4 for further details).

### 7.5 Memory Model

The following memory models are supported:

Model	Description
Standard	The standard flat memory model is used, following standard EABI.

### 7.6 Processor Modes

RTA-OS can run in the following processor modes:

Mode	Notes
Handler	Category 1 ISRs run in handler mode.
Trusted	Trusted tasks, category 2 ISRs and functions run in thread-privileged mode.
Untrusted	Untrusted tasks, category 2 ISRs and functions run in thread-unprivileged mode.

## 7.7 Stack Handling

---

RTA-OS uses a single stack for all tasks and ISRs.

SP\_main is used for the stack in both handler and thread mode. The process stack is not used.



## 8 Performance

---

This chapter provides detailed information on the functionality, performance and memory demands of the RTA-OS kernel. RTA-OS is highly scalable. As a result, different figures will be obtained when your application uses different sets of features. The figures presented in this chapter are representative for the Centauri M4/GHS port based on the following configuration:

- There are 32 tasks in the system
- Standard build is used
- Stack monitoring is disabled
- Time monitoring is disabled
- There are no calls to any hooks
- Tasks have unique priorities
- Tasks are not queued (i.e. tasks are BCC1 or ECC1)
- All tasks terminate/wait in their entry function
- Tasks and ISRs do not save any auxiliary registers (for example, floating point registers)
- Resources are shared by tasks only
- The generation of the resource RES\_SCHEDULER is disabled

### 8.1 Measurement Environment

---

The following hardware environment was used to take the measurements in this chapter:

<b>Device</b>	DME_Cut1 on
<b>CPU Clock Speed</b>	60.0MHz
<b>Stopwatch Speed</b>	20.0MHz

### 8.2 RAM and ROM Usage for OS Objects

---

Each OS object requires some ROM and/or RAM. The OS objects are generated by **rtaosgen** and placed in the RTA-OS library. In the main:

- 0s\_Cfg\_Counters includes data for counters, alarms and schedule tables.
- 0s\_Cfg contains the data for most other OS objects.

The following table gives the ROM and/or RAM requirements (in bytes) for each OS object in a simple configuration. Note that object sizes will vary depending on the project configuration and compiler packing issues.

Object	ROM	RAM
Alarm	2	12
Cat 2 ISR	8	0
Counter	20	4
CounterCallback	4	0
ExpiryPoint	3.5	0
OS Overheads (max)	0	69
OS-Application	0	0
PeripheralArea	0	0
Resource	8	4
ScheduleTable	16	16
Task	20	0

### 8.3 Stack Usage

The amount of stack used by each Task/ISR in RTA-OS is equal to the stack used in the Task/ISR body plus the context saved by RTA-OS. The size of the run-time context saved by RTA-OS depends on the Task/ISR type and the exact system configuration. The only reliable way to get the correct value for Task/ISR stack usage is to call the `Os_GetStackUsage()` API function.

Note that because RTA-OS uses a single-stack architecture, the run-time contexts of all tasks reside on the same stack and are recovered when the task terminates. As a result, run-time contexts of mutually exclusive tasks (for example, those that share an internal resource) are effectively overlaid. This means that the worst case stack usage can be significantly less than the sum of the worst cases of each object on the system. The RTA-OS tools automatically calculate the total worst case stack usage for you and present this as part of the configuration report.

### 8.4 Library Module Sizes

The RTA-OS kernel is demand linked. This means that each API call is placed into a separately linkable module. The following table lists the section sizes for each API module (in bytes) for the simple configuration in standard status.

Library Module	.app_text	.bss.a4	.os_bss	.os_const	.os_reset_vector	.os_text	.os_vectors
ActivateTask						92	
AdvanceCounter						4	
CallTrustedFunction						26	
CancelAlarm						60	
ChainTask						88	
CheckISRMemoryAccess						40	

Library Module	.app_text	.bss.a4	.os_bss	.os_const	.os_reset_vector	.os_text	.os_vectors
CheckObjectAccess						78	
CheckObjectOwnership						74	
CheckTaskMemoryAccess						40	
ClearEvent						18	
ControllIdle			8			12	
DisableAllInterrupts		4	4			40	
DispatchTask						148	
ElapsedTime						18	
EnableAllInterrupts						32	
GetActiveApplicationMode						12	
GetAlarm						108	
GetAlarmBase						48	
GetApplicationID						36	
GetCounterValue						24	
GetCurrentApplicationID						36	
GetElapsedCounterValue						42	
GetEvent						18	
GetExecutionTime						18	
GetISRID						12	
GetIsrMaxExecutionTime						18	
GetIsrMaxStackUsage						18	
GetResource						44	
GetScheduleTableStatus						24	
GetStackSize						4	
GetStackUsage						18	
GetStackValue						16	
GetTaskID						16	
GetTaskMaxExecutionTime						18	
GetTaskMaxStackUsage						18	
GetTaskState						40	
GetVersionInfo						26	
Idle	4						
InShutdown	2						
IncrementCounter						14	
InterruptSource				8		52	
ModifyPeripheral						28	
NextScheduleTable						84	
Os_CPUExceptionHandlers						58	
Os_Cat1ExceptionHandlers						40	

Library Module	.app_text	.bss.a4	.os_bss	.os_const	.os_reset_vector	.os_text	.os_vectors
Os_Cat2ExceptionHandlers						64	
Os_Cfg		4	621	776		56	
Os_Cfg_Counters				728		160	
Os_Cfg_KL						44	
Os_GetCurrentIMask						6	
Os_GetCurrentTPL						28	
Os_H2T						48	
Os_ResetVector					18		
Os_Vectors							572
Os_Wrapper				16		108	
Os_setjmp						28	
Os_vec_init						264	
ProtectionSupport						22	
ReadPeripheral						22	
ReleaseResource						52	
ResetIsrMaxExecutionTime						18	
ResetIsrMaxStackUsage						18	
ResetTaskMaxExecutionTime						18	
ResetTaskMaxStackUsage						18	
ResumeAllInterrupts						32	
ResumeOSInterrupts						32	
Schedule						64	
SetAbsAlarm						68	
SetEvent						18	
SetRelAlarm						112	
SetScheduleTableAsync						40	
ShutdownOS						60	
StackOverrunHook	6						
StartOS						92	
StartScheduleTableAbs						80	
StartScheduleTableRel						76	
StartScheduleTableSynchron						40	
StopScheduleTable						52	
SuspendAllInterrupts		4	4			40	
SuspendOSInterrupts		4	4			52	
SyncScheduleTable						40	
SyncScheduleTableRel						40	
TerminateTask						24	
ValidateCounter						40	

Library Module	.app_text	.bss.a4	.os_bss	.os_const	.os_reset_vector	.os_text	.os_vectors
ValidateISR						16	
ValidateResource						32	
ValidateScheduleTable						32	
ValidateTask						32	
WaitEvent						18	
WritePeripheral						20	

## 8.5 Execution Time

The following tables give the execution times in CPU cycles, i.e. in terms of ticks of the processor’s program counter. These figures will normally be independent of the frequency at which you clock the CPU. To convert between CPU cycles and SI time units the following formula can be used:

$$\text{Time in microseconds} = \text{Time in cycles} / \text{CPU Clock rate in MHz}$$

For example, an operation that takes 50 CPU cycles would be:

- at 20MHz =  $50/20 = 2.5\mu s$
- at 80MHz =  $50/80 = 0.625\mu s$
- at 150MHz =  $50/150 = 0.333\mu s$

While every effort is made to measure execution times using a stopwatch running at the same rate as the CPU clock, this is not always possible on the target hardware. If the stopwatch runs slower than the CPU clock, then when RTA-OS reads the stopwatch, there is a possibility that the time read is less than the actual amount of time that has elapsed due to the difference in resolution between the CPU clock and the stopwatch (the *User Guide* provides further details on the issue of uncertainty in execution time measurement).

The figures presented in Section 8.5.1 have an uncertainty of 2 CPU cycle(s).

### 8.5.1 Context Switching Time

Task switching time is the time between the last instruction of the previous task and the first instruction of the next task. The switching time differs depending on the switching contexts (e.g. an `ActivateTask()` versus a `ChainTask()`).

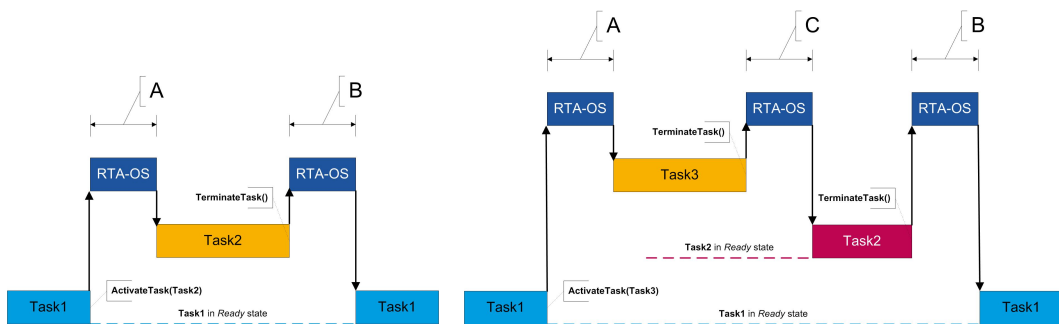
Interrupt latency is the time between an interrupt request being recognized by the target hardware and the execution of the first instruction of the user provided handler function:

**For Category 1 ISRs** this is the time required for the hardware to recognize the interrupt.

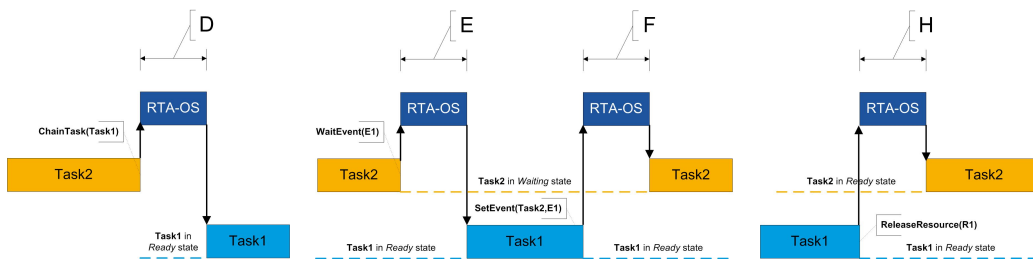
**For Category 2 ISRs** this is the time required for the hardware to recognize the interrupt plus the time required by RTA-OS to set-up the context in which the ISR runs.

Figure 8.1 shows the measured context switch times for RTA-OS.

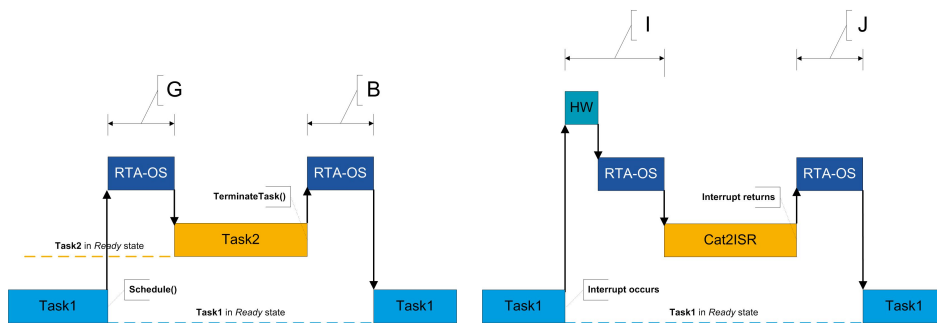
Switch	Key	CPU Cycles	Actual Time
Task activation	A	3537	59us
Task termination with resume	B	1914	31.9us
Task termination with switch to new task	C	2166	36.1us
Chaining a task	D	4080	68us
Waiting for an event resulting in transition to the WAITING state	E	8919	149us
Setting an event results in task switch	F	11919	199us
Non-preemptive task offers a preemption point (co-operative scheduling)	G	3465	57.8us
Releasing a resource results in a task switch	H	3141	52.4us
Entering a Category 2 ISR	I	4335	72.2us
Exiting a Category 2 ISR and resuming the interrupted task	J	3105	51.8us
Exiting a Category 2 ISR and switching to a new task	K	3501	58.4us
Entering a Category 1 ISR	L	1155	19.2us



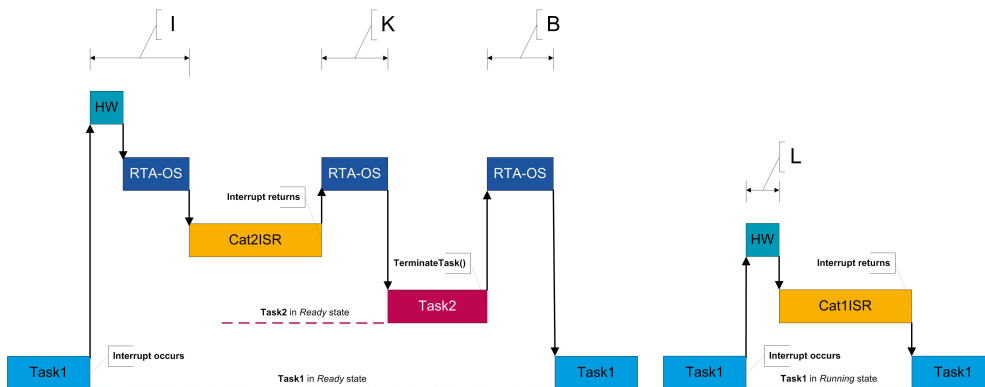
(a) Task activated. Termination resumes preempted task. (b) Task activated. Termination switches into new task.



(c) Task chained. (d) Task waits. Task is resumed when event set. (e) Task switch when resource is released.



(f) Request for scheduling made by non-preemptive task. (g) Category 2 interrupt entry. Interrupted task resumed on exit.



(h) Category 2 interrupt entry. Switch to new task on exit. (i) Category 1 interrupt entry.

Figure 8.1: Context Switching

## 9 Finding Out More

---

Additional information about Centauri M4/GHS-specific parts of RTA-OS can be found in the following manuals:

***Centauri M4/GHS Release Note.*** This document provides information about the Centauri M4/GHS port plug-in release, including a list of changes from previous releases and a list of known limitations.

Information about the port-independent parts of RTA-OS can be found in the following manuals, which can be found in the RTA-OS installation (typically in the Documents folder):

***Getting Started Guide.*** This document explains how to install RTA-OS tools and describes the underlying principles of the operating system

***Reference Guide.*** This guide provides a complete reference to the API, programming conventions and tool operation for RTA-OS.

***User Guide.*** This guide shows you how to use RTA-OS to build real-time applications.



## 10 Contacting ETAS

---

### 10.1 Technical Support

---

Technical support is available to all users with a valid support contract. If you do not have a valid support contract, please contact your regional sales office (see Section 10.2.2).

The best way to get technical support is by email. Any problems or questions about the use of the product should be sent to:

rta.hotline.uk@etas.com

If you prefer to discuss your problem with the technical support team, you call the support hotline on:

+44 (0)1904 562624.

The hotline is available during normal office hours (0900-1730 GMT/BST).

In either case, it is helpful if you can provide technical support with the following information:

- Your support contract number
- Your .xml, .arxml, .rtaos and/or .stc files
- The command line which caused the error
- The version of the ETAS tools you are using
- The version of the compiler tool chain you are using
- The error message you received (if any)
- The file Diagnostic.dmp if it was generated

### 10.2 General Enquiries

---

#### 10.2.1 ETAS Global Headquarters

---

**ETAS GmbH**

Borsigstrasse 24  
70469 Stuttgart  
Germany

Phone:	+49 711 3423-0
Fax:	+49 711 3423-2106
WWW:	<a href="http://www.etas.com">www.etas.com</a>

#### 10.2.2 ETAS Local Sales & Support Offices

---

Contact details for your local sales office and local technical support team (where available) can be found on the ETAS web site:

ETAS subsidiaries      [www.etas.com/en/contact.php](http://www.etas.com/en/contact.php)  
 ETAS technical support   [www.etas.com/en/hotlines.php](http://www.etas.com/en/hotlines.php)

## Index

---

### A

Assembler, [32](#)  
 AUTOSAR OS includes  
   Os.h, [25](#)  
   Os\_Cfg.h, [25](#)  
   Os\_MemMap.h, [25](#)

### C

CAT1\_ISR, [27](#)  
 Compiler, [30](#)  
 Compiler (Green Hills Software, Compiler v2018.1.4), [30](#)  
 Compiler Versions, [30](#)  
 Configuration  
   Port-Specific Parameters, [21](#)

### D

Debugger, [34](#)

### E

ETAS License Manager, [11](#)  
   Installation, [11](#)

### F

Files, [25](#)

### H

Hardware  
   Requirements, [9](#)

### I

Installation, [9](#)  
   Default Directory, [10](#)  
   Verification, [19](#)  
 Interrupts, [37](#)  
   Category 1, [38](#)  
   Category 2, [39](#)  
   Default, [39](#)  
 IPL, [37](#)

### L

Librarian, [33](#)  
 Library  
   Name of, [25](#)  
 License, [11](#)  
   Borrowing, [15](#)  
   Concurrent, [12](#)

Grace Mode, [12](#)  
 Installation, [15](#)  
 Machine-named, [12](#)  
 Status, [15](#)  
 Troubleshooting, [16](#)  
 User-named, [12](#)

Linker, [33](#)

### M

Memory Model, [39](#)

### O

Options, [30](#)  
 Os\_Cbk\_GetAbortStack, [26](#)  
 Os\_Clear\_x, [27](#)  
 Os\_Disable\_x, [28](#)  
 Os\_DisableAllConfiguredInterrupts, [28](#)  
 Os\_Enable\_x, [28](#)  
 Os\_EnableAllConfiguredInterrupts, [28](#)  
 Os\_InitializeVectorTable, [26](#)  
 Os\_IntChannel\_x, [28](#)  
 Os\_IntID\_x, [29](#)  
 Os\_StackSizeType, [29](#)  
 Os\_StackValueType, [29](#)

### P

Parameters of Implementation, [21](#)  
 Performance, [41](#)  
   Context Switching Times, [45](#)  
   Library Module Sizes, [42](#)  
   RAM and ROM, [41](#)  
   Stack Usage, [42](#)  
 Processor Modes, [39](#)  
   Handler, [39](#)  
   Trusted, [39](#)  
   Untrusted, [39](#)

### R

Registers  
   ACTLR, [37](#)  
   BASEPRI, [37](#)  
   CCR STKALIGN/NONBASETHRENA, [37](#)  
   CCR.NONBASETHRDENA, [36](#)  
   CCR.STKALIGN, [36](#)  
   CONTROL, [37](#)  
   CONTROL.nPRIV, [36](#)

- CONTROL.SPSEL, [36](#)
  - CPACR/FPCCR, [36](#)
  - FAULTMASK, [36](#), [37](#)
  - FPCCR, [37](#)
  - Initialization, [35](#)
  - IPSR, [36](#)
  - Non-modifiable, [36](#)
  - NVIC, [37](#)
  - PRIMASK, [36](#), [37](#)
  - SHCSR, [37](#)
  - SHPRx, [37](#)
  - SP (R13), [36](#), [37](#)
  - VTOR, [36](#)
  - Resource
    - SVCall, [37](#)
- S**
- Software
    - Requirements, [9](#)
  - Stack, [40](#)
- T**
- Target, [35](#)
    - Variants, [35](#)
  - Toolchain, [30](#)
- V**
- Variants, [35](#)
  - Vector Table
    - Base Address, [38](#)