RTA-OS

FR81/Softune Port Guide

Copyright

Document: 10643-PG-2.0.1 EN-04-2015

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

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 -> R4.0.3 and R4.1.1 -> R4.1.3) and OSEK/VDX 2.2.3 standards. The operating system is configured and built on a PC, but runs on your target hardware.

This document describes the RTA-OS FR81/Softune port plug-in that customizes the RTA-OS development tools for the Fujitsu FR81 with the Softune 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 FR81/Softune port and cover:

- how to install the FR81/Softune port plug-in;
- how to configure FR81/Softune-specific attributes;
- how to build an example application to check that the FR81/Softune 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 FR81, 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 **File > Open**.	Menu options appear in **bold, blue** characters.
Click **OK**.	Button labels appear in **bold** 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 blue letters.
OS	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. 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 FR81/Softune 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 (or higher) IBM compatible PC.
- 512Mb RAM.
- 500Mb 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 2000 (Service Pack 3 or later)
- Windows XP (Service Pack 2 or later)
- Windows Vista
- Windows 7

Integration Guidance 2.2:*The tools provided with RTA-OS require Microsoft's .NET Framework* v2.0 *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* `http://www.microsoft.com/net/Download.aspx`*.*
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 FR81/Softune 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 FR81/Softune 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 `FR81Softune_2.0.1`. This contains everything to do with the port plug-in.

Each version of the port installs in its own directory - the trailing `_2.0.1` 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:

FR81Softune.dll - the port plug-in that is used by **rtaosgen** and **rtaoscfg**.

RTA-OS FR81Softune Port Guide.pdf - the documentation for the port (the document you are reading now).

RTA-OS FR81Softune 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`.

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

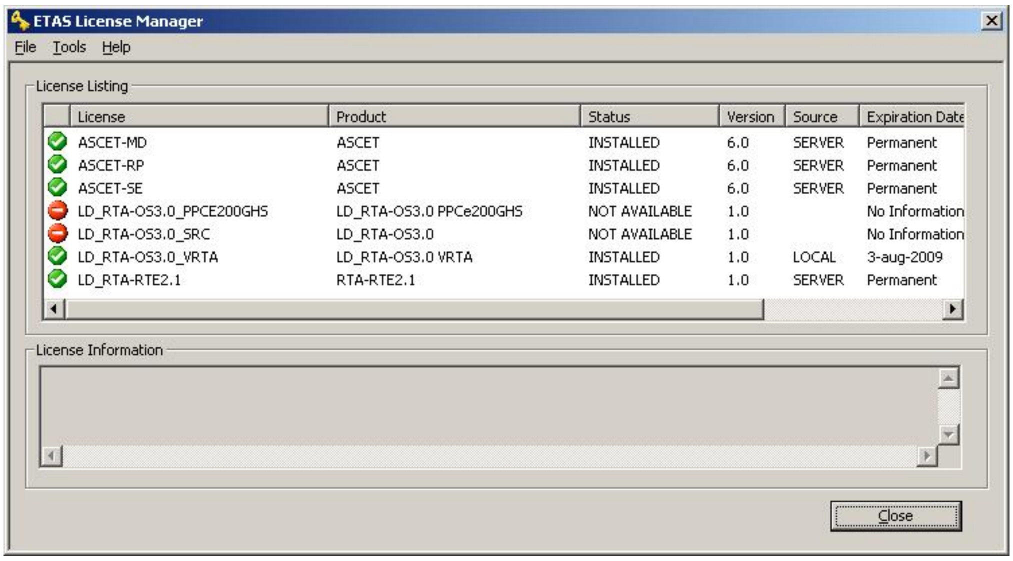
Figure 2.1 shows the ETAS License Manager in operation.

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;



	License	Product	Status	Version	Source	Expiration Date
✓	ASCET-MD	ASCET	INSTALLED	6.0	SERVER	Permanent
✓	ASCET-RP	ASCET	INSTALLED	6.0	SERVER	Permanent
✓	ASCET-SE	ASCET	INSTALLED	6.0	SERVER	Permanent
🔴	LD_RTA-OS3.0_PPCE200GHS	LD_RTA-OS3.0 PPCe200GHS	NOT AVAILABLE	1.0		No Information
🔴	LD_RTA-OS3.0_SRC	LD_RTA-OS3.0	NOT AVAILABLE	1.0		No Information
✓	LD_RTA-OS3.0_VRTA	LD_RTA-OS3.0 VRTA	INSTALLED	1.0	LOCAL	3-aug-2009
✓	LD_RTA-RTE2.1	RTA-RTE2.1	INSTALLED	1.0	SERVER	Permanent

Figure 2.1: The ETAS License manager

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 seven 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.

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.

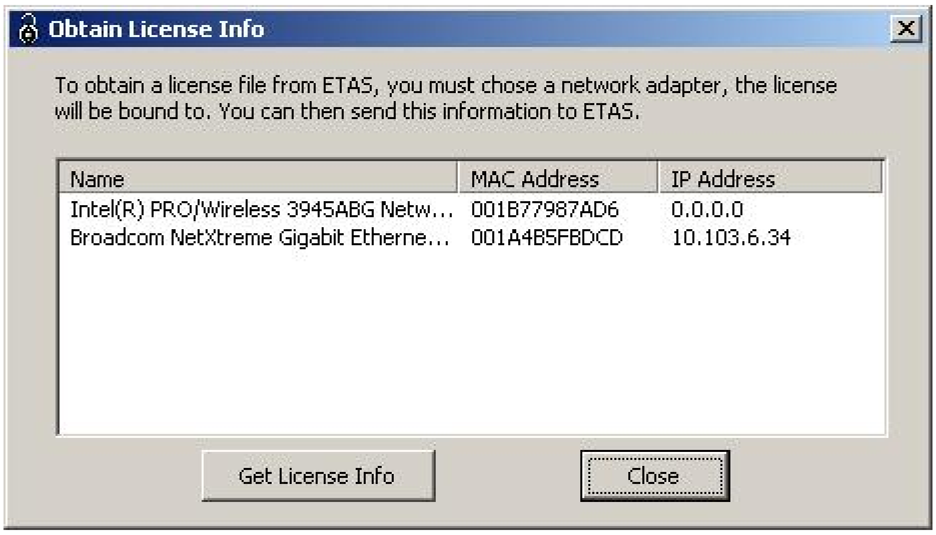
Open the ETAS License Manager and choose **Tools → Obtain License Info** from the menu. For machine-named licenses you can then select the network adaptor which provides the Host ID (MAC address) that you want to use as shown in Figure 2.2. For a user-based license, the ETAS License Manager automatically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



Obtain License Info

To obtain a license file from ETAS, you must chose a network adapter, the license will be bound to. You can then send this information to ETAS.

Name	MAC Address	IP Address
Intel(R) PRO/Wireless 3945ABG Netw...	001B77987AD6	0.0.0.0
Broadcom NetXtreme Gigabit Etherne...	001A4B5FBDCD	10.103.6.34

Figure 2.2: Obtaining License Information

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.

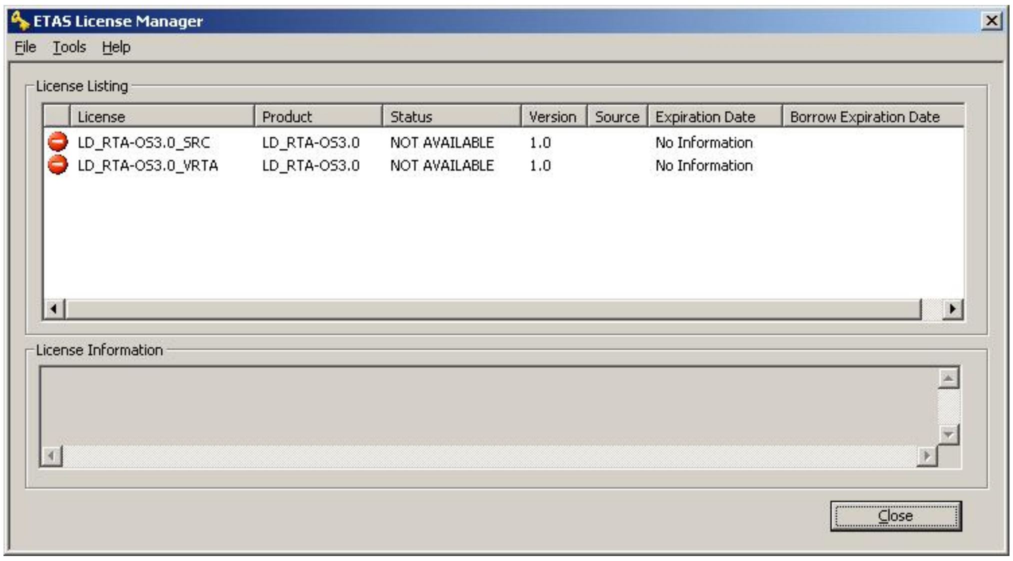
In most organizations there will be a single FLEXnet license server manager that is administered by your IT department. You will need to ask your IT department to install the ETAS vendor daemon and the associated concurrent license.

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 RTA-OS without a valid license, you will be given the opportunity to start the ETAS License Manager and select a license.



	License	Product	Status	Version	Source	Expiration Date	Borrow Expiration Date
⛔	LD_RTA-OS3.0_SRC	LD_RTA-OS3.0	NOT AVAILABLE	1.0		No Information	
⛔	LD_RTA-OS3.0_VRTA	LD_RTA-OS3.0	NOT AVAILABLE	1.0		No Information	

Figure 2.3: Unlicensed RTA-OS Installation

When the ETAS License Manager is launched, it will display the RTA-OS license state as `NOT AVAILABLE` and you will not be able to use any of the tools until a valid license is installed. This is shown in Figure 2.3.

License Key Installation

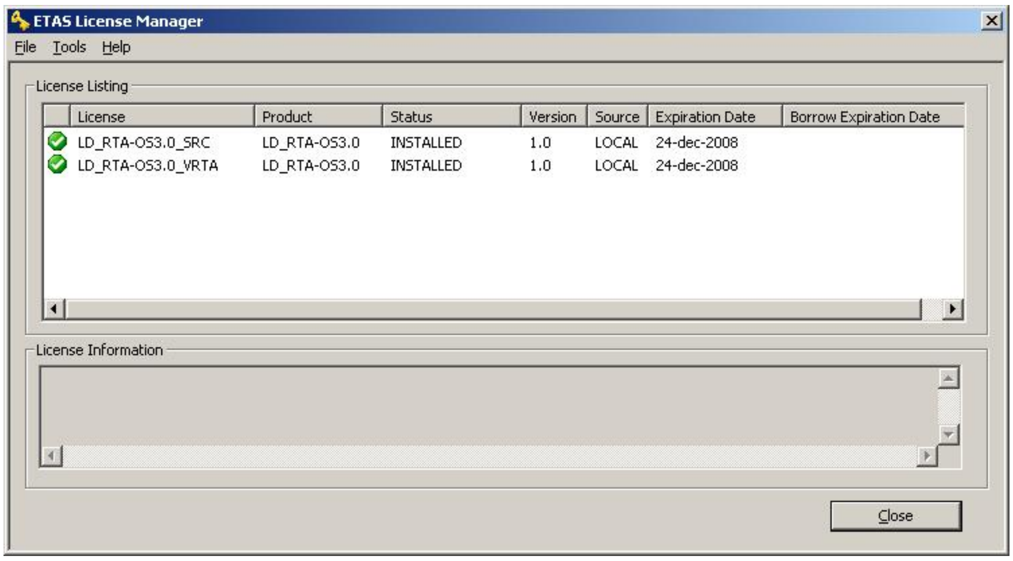
License keys are supplied in an ASCII text file, which will be sent to you on completion of a valid license agreement.

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:



	License	Product	Status	Version	Source	Expiration Date	Borrow Expiration Date
✓	LD_RTA-OS3.0_SRC	LD_RTA-OS3.0	INSTALLED	1.0	LOCAL	24-dec-2008	
✓	LD_RTA-OS3.0_VRTA	LD_RTA-OS3.0	INSTALLED	1.0	LOCAL	24-dec-2008	

Figure 2.4: Licensed features for RTA-OS

```
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.

When a license is installed by the ETAS License Manager it is placed in: `C:\Documents and Settings\All Users\Application Data\ETAS\FlexNet`

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”

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**.

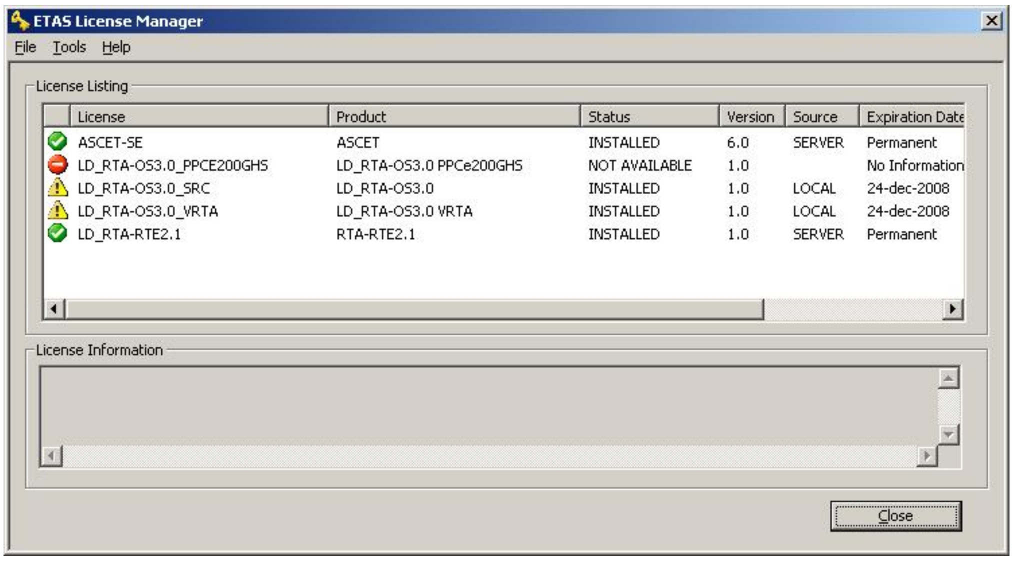
The ETAS License Manager lists all license features and their status. Valid licenses have status `INSTALLED`. Invalid licenses have status `NOT AVAILABLE`.

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_RTA-OS3.0_VRTA` and `LD_RTA-OS3.0_SRC` are due to expire.

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



	License	Product	Status	Version	Source	Expiration Date
✓	ASCET-SE	ASCET	INSTALLED	6.0	SERVER	Permanent
⛔	LD_RTA-OS3.0_PPCE200GHS	LD_RTA-OS3.0 PPCe200GHS	NOT AVAILABLE	1.0		No Information
⚠	LD_RTA-OS3.0_SRC	LD_RTA-OS3.0	INSTALLED	1.0	LOCAL	24-dec-2008
⚠	LD_RTA-OS3.0_VRTA	LD_RTA-OS3.0 VRTA	INSTALLED	1.0	LOCAL	24-dec-2008
✓	LD_RTA-RTE2.1	RTA-RTE2.1	INSTALLED	1.0	SERVER	Permanent

Figure 2.5: Licensed features that are due to expire

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 FR81/Softune port.

4.1 Parameters of Implementation

To be a valid OSEK 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 FR81/Softune 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 FR81/Softune 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 Os_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. The value does not normally change if the OS configuration changes.

4.2.3 Stack overheads for ISR activation

XML name SpIDisp

Description

The 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 the OS may need to use marginally more stack than if it activates a task that is of lower priority. 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 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, assuming the maximum depth of preemption that can occur. 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 occurred. The value may change if significant changes are made to the OS configuration. e.g. STANDARD/EXTENDED, SC1/2/3/4.

4.2.6 ISRs preserve BP

XML name isrs_preserve_bp

Description

By default the BP register does not get preserved by Category 2 interrupts. Set this option TRUE to preserve BP.

Settings

Value	Description
TRUE	save
FALSE	do not save (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.
`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 **rtaosgen** 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 FR81/Softune 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

Initializes the TBR register and ICR registers as per the current configuration before enabling Category 1 ISRs. Os_InitializeVectorTable() should be called before StartOS(). It should be called even if 'Suppress Vector Table Generation' is set to TRUE.

As part of initializing the vector table, it copies the table Os_VectorPriorities, which contains the ICR values required for the vectors configured in the OS, to the location of the ICR registers. If Os_InitializeVectorTable() is not called, this table should be copied to the ICR registers before StartOS().

Example

```
Os_InitializeVectorTable();
```

See Also

StartOS

5.2 Callbacks

5.2.1 Os_Cbk_GetInterruptStack

Callback routine to provide the start address of the stack to use to temporarily store interrupt context, and for Category 1 ISRs to use. Only required when untrusted applications have been configured.

Syntax

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

Return Values

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

Description

When untrusted applications have been configured, RTA-OS uses the user stack as its primary stack in order to use a single-stack model; however, the CPU will automatically store interrupt context to an area pointed to by the system stack pointer.

For this reason, this callback is used during initialization to get an area of memory to use as an interrupt stack. It will be used to temporarily store interrupt context before it is moved onto the user stack; it is also the stack on which Category 1 ISRs will run, so that they do not incur the overhead of having to switch back to the user stack after firing.

This area must be large enough for an interrupt at every level of preemption to occur while each level is at its worst-case system stack usage. Each ISR will initially store two words (8 bytes) of interrupt context on the stack. For Category 2 ISRs, two extra words (8 bytes) of system stack are then used in order to transfer to the user stack. This means that this callback needs to provide (i) 16 bytes for every IPL which contains any Category 2 ISRs, (ii) enough stack space for each IPL containing a Category 1 ISR to be preempted at its worst-case preemption depth, and (iii) enough for any fault handlers to run.

Example

```
FUNC(void *, OS_APPL_CODE) Os_Cbk_GetInterruptStack(void)
{
  /* Note: The size of the interrupt stack provided by the default
   * implementation should be adjusted based on the user's
   * configuration. */
  static uint32 stack[0x400];
  return &stack[0x400];
}
```

Required when

This callback must be present if there are any untrusted applications. If it is not provided by the user, a default implementation will be provided by the OS.

5.3 Macros

5.3.1 CAT1_ISR

Macro that should be used to create Category 1 ISR entry functions, including handlers for CPU/INT exceptions and the NMI. This macro exists to help make your code portable between targets.

Example

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

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 (ESP).

Example

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

6 Toolchain

This chapter contains important details about RTA-OS and the Softune 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

Name	fcc911s.exe
Version	V65L06R03

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.

`-cpu MB91F528YW` Specify the CPU information to use.

`-Xdof` Do not read from a default option file.

`-w 8` Highest warning level.

`-cwno` Fail when warnings occur.

`-B` Allow C++-style "//" comments.

`-K SPEED` Optimize for speed.

`-Xalign` Don’t force functions to be 4 byte aligned.

`-K LONGADDRESS` Use the entire 32 bit address space.

`-K NOUNROLL` Do not unroll loops.

`-K SCHEDULE` Optimize instruction scheduling.

`-K NOLIB` Do not auto-inline standard library functions.

`-K A1` Do not force 4 byte alignment for external and static variables.

`-K NOEOPT` Do not optimize the order/type of arithmetic operations.

`-K LONGLONG` Allow use of type ’long long int’.

`-K NOFPU` Disable floating-point support in the kernel.

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 the 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 Assembler

Name	fcc911s.exe
Version	V65L06R03

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 the 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

6.3 Linker

Name	flnk911s.exe
Version	V60L09

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.

`-cpu MB91F528YW` Specify the CPU information to use.

`-Xdof` Do not read from a default option file.

`-g` Output debugging information.

`-w 2` Highest warning level.

`-cwno` Fail when warnings occur.

`-NCI0302LIB` Ignore warnings about lack of debug information.

`-nd` Do not include the standard library.

`-a` Output a file in absolute format.

`-check_locate` Verify section placement.

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

6.4 Debugger

Name	Softune Workbench Debugger
Version	V60L10 or later

7 Hardware

7.1 Supported Devices

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

Name:	Fujitsu
Device:	FR81

The following variants of the FR81 are supported:

- GenericMB91520
- MB91F52xU
- MB91F52xY

If you require support for a variant of FR81 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
ICRxx	The Interrupt Control Registers must each be set to values that correspond to the interrupt priorities declared in the configuration. This can be done by calling Os_InitializeVectorTable()
PS:S	The S bit (bit 5) in the Program Status register may be set (i.e. the processor may be using the user stack pointer) if and only if there are any untrusted applications in the OS configuration. This is to allow user code to be stack-agnostic, so that it may e.g. call StartOS() from a power-on reset using the system stack, but disregard which stack is being used when the OS is restarted. If all applications are trusted, the S bit must be clear (i.e. the processor must be using the system stack).
PS:UM	The UM bit (bit 30) in the Program Status register must be cleared (i.e. the processor must be in privilege mode).
R15 (SP)	The stack must be allocated and R15 (SP) initialized *before* calling Os_InitializeVectorTable().
TBR	The Table Base Register must be pointing to a valid vector table that matches that declared in the configuration. This can be done by calling Os_InitializeVectorTable().

7.2.2 Modification

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

Register	Notes
ICRxx	User code must not modify the Interrupt Control Registers.
ILM	User code must not modify the interrupt Level Mask Register.
PS	User code must not modify the Program Status register, other than as a result of normal program flow.
R15 (SP), SSP and USP	User code must not modify either the System or User stack pointers, either directly or by modifying R15, other than as a result of normal program flow.
TBR	User code must not modify the Table Base Register.

Note that this port currently does not handle saving any floating-point register context—if this is required, it must be performed using register sets.

7.3 Interrupts

This section explains the implementation of RTA-OS's interrupt model on the FR81.

7.3.1 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 FR81:

IPL	ILM	Description
0	0x1f	User (task) level. No interrupts are masked.
1	0x1e	Category 1 and 2 maskable interrupts.
...	...	Category 1 and 2 maskable interrupts.
15	0x10	Category 1 and 2 maskable interrupts.
16	0x0f	Category 1 NMI handler.
16	N/A	Category 1 CPU trap/INT handlers.

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

7.3.2 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 FR81. A ✓ indicates that the mapping is permitted and a ✗ indicates that it is not permitted:

Address	Category 1	Category 2
0x01-0x0e, CPU trap handlers	✓	✗
0x0f, NMI handler	✓	✗
0x10-0x3f, Peripheral interrupts	✓	✓
0x40-0xff, INT handlers	✓	✗

7.3.3 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	Includes all vectors up to and including the highest configured interrupt in the system, but *not* the reset handler. Should normally be placed so that the first entry (with the lowest vector number) lies at address 0xf_fff8—the example `linker.opt` script contains an "`-sc`" directive which places this at the usual location.

7.3.4 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 */
}
```

Note that the CAT1_ISR macro may also include code to ensure that a Category 1 ISR runs at the correct priority, where its dispatch and base priorities differ (e.g. when two Category 1 ISRs share an internal resource). If you choose not to use the CAT1_ISR macro, you must ensure that you implement this correctly.

7.3.5 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.

7.3.6 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 7.3.4 for further details).

7.4 Memory Model

The following memory models are supported:

Model	Description
Standard	The standard 32-bit memory model is used.

7.5 Processor Modes

RTA-OS can run in the following processor modes:

Mode	Notes
Privilege	All trusted code runs in privilege mode (i.e. PS.UM bit clear).
User	All untrusted code runs in user mode (i.e. PS.UM bit set).

7.6 Stack Handling

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

Note that when the processor is in user mode, the FR81 enforces use of the user stack pointer. As a result of this, if there are any untrusted applications configured, RTA-OS will use the user stack pointer in all tasks and Category 2 ISRs. Note that RTA-OS will automatically generate code to enforce use of the user stack when necessary, so no user code needs to be changed when untrusted applications are added.

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 FR81/Softune 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:

Device	MB91F52xY on SK-MB91F528Y-416BGA
CPU Clock Speed	80.0MHz
Stopwatch Speed	80.0MHz

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:

- `Os_Cfg_Counters` includes data for counters, alarms and schedule tables.
- `Os_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
Resource	8	4
ScheduleTable	16	16
Task	16	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	CODE	CONST	DATA	OS_CODE	OS_CODE_FAST	OS_VECTORS
ActivateTask	130					

Library Module	CODE	CONST	DATA	OS_CODE	OS_CODE_FAST	OS_VECTORS
AdvanceCounter	4					
CallTrustedFunction	28					
CancelAlarm	104					
ChainTask	126					
CheckISRMemoryAccess	46					
CheckObjectAccess	112	24				
CheckObjectOwnership	108	24				
CheckTaskMemoryAccess	46					
ClearEvent	44					
ControlIdle	74		4			
DisableAllInterrupts	50		8			
DispatchTask	148					
ElapsedTime	228					
EnableAllInterrupts	52					
GetActiveApplicationMode	10					
GetAlarm	164					
GetAlarmBase	52					
GetApplicationID	32					
GetCounterValue	58					
GetElapsedCounterValue	88					
GetEvent	44					
GetExecutionTime	44					
GetISRID	10					
GetIsrMaxExecutionTime	44					
GetIsrMaxStackUsage	44					
GetResource	80					
GetScheduleTableStatus	60					
GetStackSize	4					
GetStackUsage	44					
GetStackValue	20					
GetTaskID	14					
GetTaskMaxExecutionTime	44					
GetTaskMaxStackUsage	44					
GetTaskState	40					
GetVersionInfo	36					
Idle	4					
InShutdown	2					
IncrementCounter	14					

Library Module	CODE	CONST	DATA	OS_CODE	OS_CODE_FAST	OS_VECTORS
NextScheduleTable	162					
Os_Cfg	256	648	621			
Os_Cfg_Counters	5114	728				
Os_Exception_Handlers				16		
Os_Validation	28					
Os_Vectors				52		176
Os_Wrapper	74					
Os_jmp				66		
Os_memclr	18					
Os_misc					14	
Os_setup_interrupt_stack				8		
Os_sp					8	
Os_trust					8	
Os_vec_init	48	48				
ProtectionSupport	58					
ReleaseResource	88					
ResetIsrMaxExecutionTime	44					
ResetIsrMaxStackUsage	44					
ResetTaskMaxExecutionTime	44					
ResetTaskMaxStackUsage	44					
ResumeAllInterrupts	52					
ResumeOSInterrupts	52					
Schedule	94					
SetAbsAlarm	120					
SetEvent	44					
SetRelAlarm	174					
SetScheduleTableAsync	74					
ShutdownOS	26					
StackOverrunHook	12					
StartOS	92					
StartScheduleTableAbs	168					
StartScheduleTableRel	138					
StartScheduleTableSynchron	74					
StopScheduleTable	98					
SuspendAllInterrupts	50		8			
SuspendOSInterrupts	66		8			
SyncScheduleTable	76					
SyncScheduleTableRel	76					

Library Module	CODE	CONST	DATA	OS_CODE	OS_CODE_FAST	OS_VECTORS
TerminateTask	4					
ValidateCounter	24					
ValidateISR	18					
ValidateResource	24					
ValidateScheduleTable	24					
ValidateTask	24					
WaitEvent	44					

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:

Time in microseconds = Time in cycles / CPU Clock rate in MHz

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

- at 20MHz = 50/20 = 2.5μs
- at 80MHz = 50/80 = 0.625μs
- at 150MHz = 50/150 = 0.333μs

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 0 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()`).

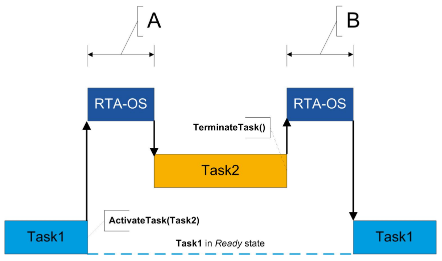
Interrupt latency is the time between an interrupt request being recognized by the target hardware and the execution of the first instruction of the user provided handler function:

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

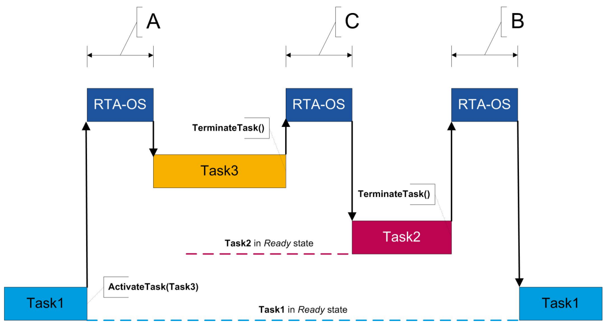
For Category 2 ISRs this is the time required for the hardware to recognize the interrupt plus the time required by RTA-OS to set-up the context in which the ISR runs.

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

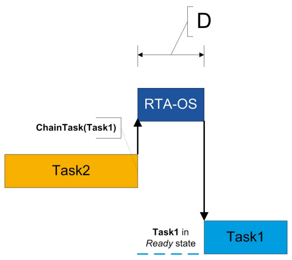
Switch	Key	CPU Cycles	Actual Time
Task activation	A	102	1.28us
Task termination with resume	B	61	763ns
Task termination with switch to new task	C	64	800ns
Chaining a task	D	128	1.6us
Waiting for an event resulting in transition to the WAITING state	E	265	3.31us
Setting an event results in task switch	F	348	4.35us
Non-preemptive task offers a preemption point (co-operative scheduling)	G	91	1.14us
Releasing a resource results in a task switch	H	89	1.11us
Entering a Category 2 ISR	I	52	650ns
Exiting a Category 2 ISR and resuming the interrupted task	J	63	788ns
Exiting a Category 2 ISR and switching to a new task	K	93	1.16us
Entering a Category 1 ISR	L	20	250ns



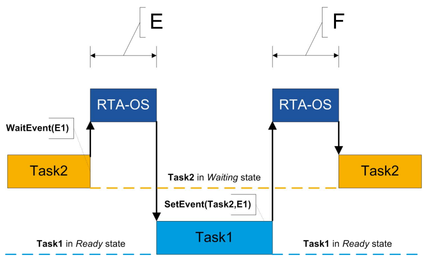
(a) Task activated. Termination resumes preempted task.



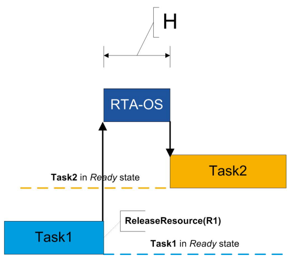
(b) Task activated. Termination switches into new task.



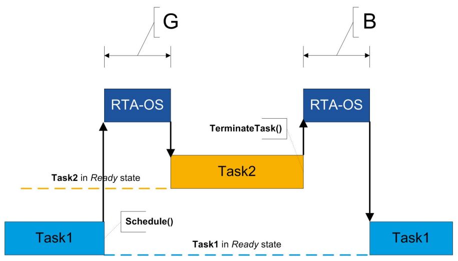
(c) Task chained.



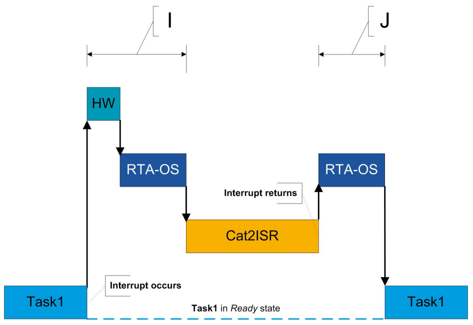
(d) Task waits. Task is resumed when event set.



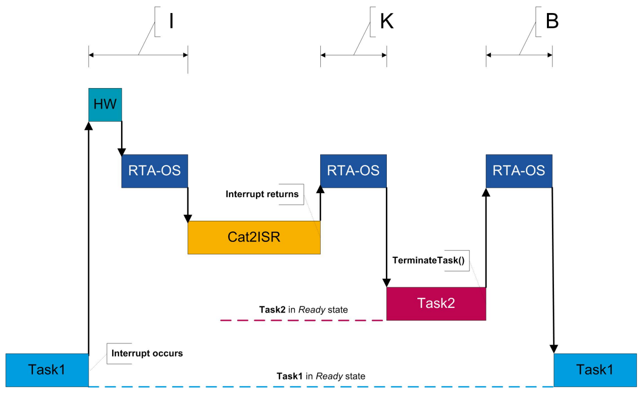
(e) Task switch when resource is released.



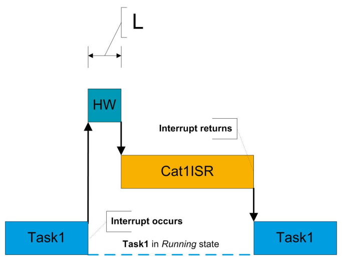
(f) Request for scheduling made by non-preemptive task.



(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 FR81/Softune-specific parts of RTA-OS can be found in the following manuals:

FR81/Softune Release Note. This document provides information about the FR81/Softune 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:

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 14	Phone:	+49 711 3423-0
70469 Stuttgart	Fax:	+49 711 3423-2106
Germany	WWW:	`www.etas.com`

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`
ETAS technical support	`www.etas.com/en/hotlines.php`

Index