RTA-TRACE

Getting Started Guide

Contact Details

ETAS Group

www.etasgroup.com

Germany

ETAS GmbH
Borsigstraße 14
70469 Stuttgart

Tel.:+49 (711) 8 96 61-102
Fax:+49 (711) 8 96 61-106

www.etas.de

USA

ETAS Inc.
3021 Miller Road
Ann Arbor, MI 48103

Tel.: +1 (888) ETAS INC
Fax: +1 (734) 997-94 49

www.etasinc.com

Japan

ETAS K.K.
Queen's Tower C-17F,
2-3-5, Minatomirai, Nishi-ku,
Yokohama, Kanagawa
220-6217 Japan

Tel.: +81 (45) 222-0900
Fax: +81 (45) 222-0956

www.etas.co.jp

France

ETAS S.A.S.
1, place des États-Unis
SILIC 307
94588 Rungis Cedex

Tel.: +33 (1) 56 70 00 50
Fax: +33 (1) 56 70 00 51

www.etas.fr

Korea

ETAS Korea Co. Ltd.
3F, Samseung Bldg. 61-1
Yangjae-dong, Seocho-gu
Seoul

Tel.: +82 (2) 57 47-016
Fax: +82 (2) 57 47-120

www.etas.co.kr

Great Britain

ETAS UK Ltd.
Studio 3, Waterside Court
Third Avenue, Centrum 100
Burton-upon-Trent
Staffordshire DE14 2WQ

Tel.: +44 (0) 1283 - 54 65 12
Fax: +44 (0) 1283 - 54 87 67

www.etas-uk.net

Contents

1 About this Manual

This manual serves as an introduction to RTA-TRACE. It covers the installation process for RTA-TRACE, as well as building a simple application with tracing enabled. The manual also covers initial interaction with the RTA-TRACE user interface.

The host software components run under Windows XP, Windows 98SE, Windows NT, and Windows ME.

1.1 Who Should Read this Manual?

The *RTA-TRACE Getting Started Guide* is aimed at the technical reader who should have experience programming embedded systems in C using their chosen RTOS (such as ERCOS^{EK} or RTA-OSEK). The reader should already have installed the RTOS and verified that the tutorial application provided with it builds and runs.

1.2 Conventions

Important: Notes that appear like this contain important information that you need to be aware of. Make sure that you read them carefully and that you follow any instructions that you are given.

Portability: Notes that appear like this describe things that you will need to know if you want to write code that will work on any target processor.

In this guide you'll see that program code, header file names, C type names, C functions and API call names all appear in the `courier` typeface. When the name of an object is made available to the programmer the name also appears in the `courier` typeface, so, for example, a task named Task1 appears as a task handle called `Task1`.

Sessions at the command prompt are shown with output shown in `courier` and user input shown in **`courier bold`**.

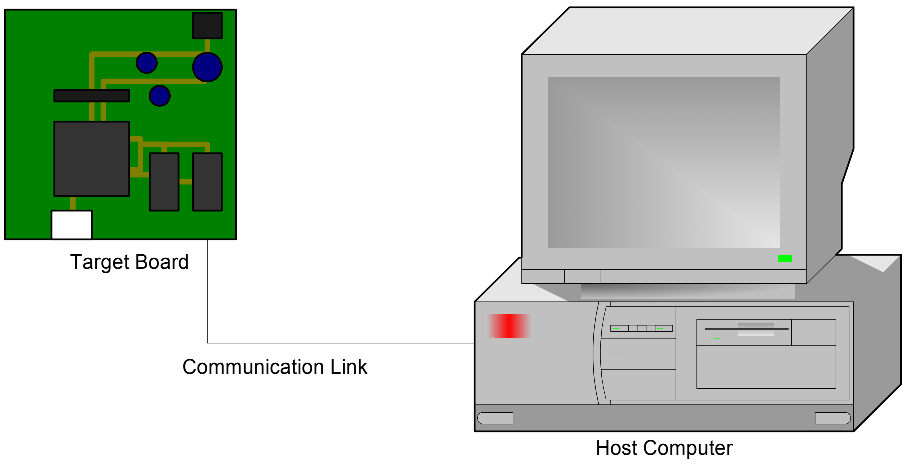
2 What is RTA-TRACE?

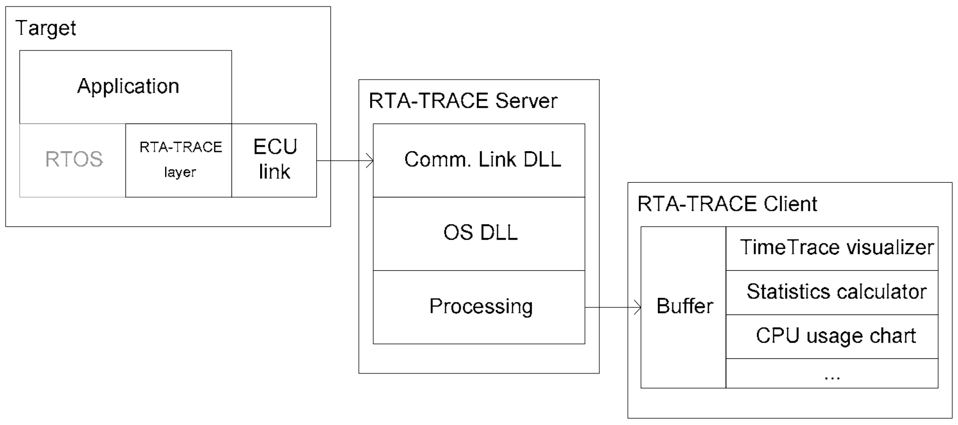
RTA-TRACE records the activity of a running application and displays both instantaneous and summary information at runtime. It gives the engineer a valuable insight into the actual behavior of a system and allows detailed analysis of behavior in pathological cases via runtime visualization, reporting, and measurement tools.

RTA-TRACE can record OS activity (such as task activations, resource-locks, and alarms) as well as user-specified occurrences. Users can define *tracepoints*, *task-tracepoints*, and *intervals* to mark specific sections of application activity. These terms are defined more fully in the *RTA-TRACE User Manual* and online help files.

2.1 RTA-TRACE overview

Tracing an application using RTA-TRACE requires target hardware (running the application to be examined) and a host PC on which to run the RTA-TRACE tools.





The RTA-TRACE tools consist of two elements: the RTA-TRACE Server and the RTA-TRACE Client. The Server is responsible for receiving data from the target software and making it available to the RTA-TRACE Client. The Client reads and stores the data received from the Server (into the *trace buffer*) and hosts one or more *plugins* (such as the TimeTrace visualizer) allowing the trace data to be visualized/analyzed by the user.

The target software comprises the core RTA-TRACE library (and its associated header files) and, optionally, an ECU Link library (to support, for example RS232). RTA-TRACE also supports the use of a debugger to extract trace data from the target where this is the most convenient option. ECU Links are described more fully in the *RTA-TRACE ECU Link Guide*.

2.2 How tracing is enabled

In the simplest case, RTA-TRACE will trace a typical set of events with only a small change to the configuration files and the build environment (makefile or batch file as appropriate). No other changes, including to the C, are necessary.

In this simple case, when tracing is enabled for RTA-OSEK and ERCOS^{EK}, the build process uses an instrumented version of the operating system to record trace data, which can then be dumped using debugger scripts and subsequently loaded into the RTA-TRACE Client for analysis.

Usually, however, other features will be required (such as runtime upload of trace data via RS232 or user-defined trace information). In this case, the user must augment the application with calls to the tracing API, and may also be required to supply some functions for the ECU Link to call (See *RTA-TRACE User Manual* and *ECU Link Guide*).

2.3 ERCOS^{EK} Configuration

For ERCOS^{EK}, tracing configuration is managed through the file `RTAtrace.cfg`, contained within the build directory alongside `run.bat` and `project_settings.mk`. With this file, *classes* of trace event such as task-activations, resource-locks etc. can be switched on or off.

In addition, tracepoints, task-tracepoints, and intervals can be defined here, and some configuration of the RTA-TRACE target runtime can be performed. See *RTA-TRACE Configuration Guide* for a full explanation.

Note: Tracing is generally enabled with an attribute of the OS clause in the application code, not using a directive in this file. For example, ERCOS^{EK} users will use the following fragment:

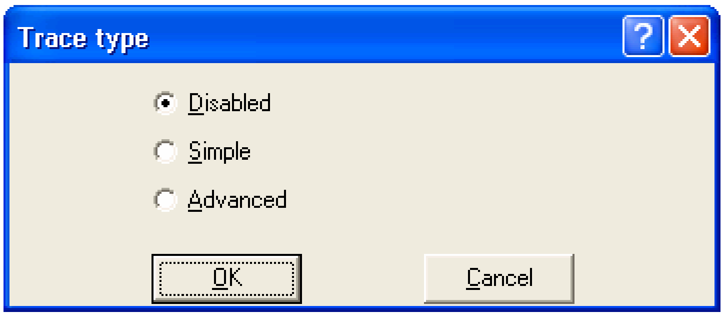
```
OS {

      RT_ENABLE = TRUE;

};
```

2.4 RTA-OSEK Configuration

For RTA-OSEK, tracing configuration is managed through the GUI configuration tool. Tracing configuration pages are accessible through the 'Trace' tab. *Classes* of trace event such as task-activations, resource-locks etc. can be switched on or off.

In addition, *tracepoints*, *task-tracepoints*, and *intervals* can be defined here, and some configuration of the RTA-TRACE target runtime can be performed. See *RTA-TRACE Configuration Guide* for a full explanation.

Tracing is enabled by selecting a Trace Type of Simple or Advanced in the Trace/Configuration section of the GUI.


Trace type

[✓ Disabled]
Simple
Advanced

Advanced tracing provides more detailed tracing than simple tracing, with a corresponding increase in trace-records.

3 Installation Procedure

3.1 Host PC

The RTA-TRACE host software is provided on a CD.

1. Insert the RTA-TRACE tools CD into the drive;
2. If the installer program does not start automatically, locate and run the 'setup.exe' file on the CD.
3. Clicking on 'Install' will start the RTA-TRACE installation wizard. When installation is complete, click the 'Finish' button.

3.2 Target Software

3.2.1 ERCOS^{EK}

For operation with ERCOS^{EK} versions earlier than 4.3, installation of target software and modification of the build process must be performed by LiveDevices engineers. Contact LiveDevices for further information. Versions 4.3 and later already contain tracing support.

Once the tools and target software have been installed, you are ready to build the example application, ‘`TRACE_EG`’, described in section 4.

3.2.2 RTA-OSEK

RTA-TRACE support is built in to RTA-OSEK from version 4, and no further installation steps are necessary.

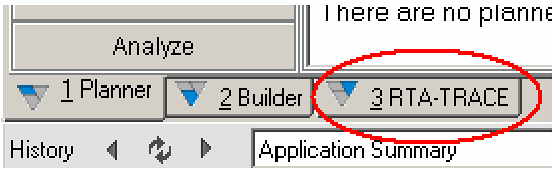
4 Tracing the RTA-OSEK example application

4.1 Walkthrough

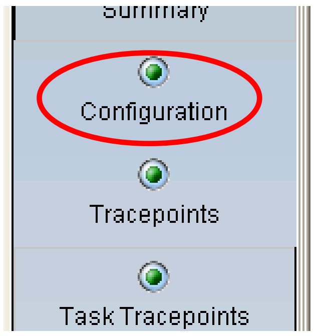
RTA-OSEK is supplied with an example application which can be used to verify the build-environment. Tracing can be enabled for this application in order to subsequently verify that RTA-TRACE for RTA-OSEK is functioning correctly.

The steps required to enable tracing are as follows:

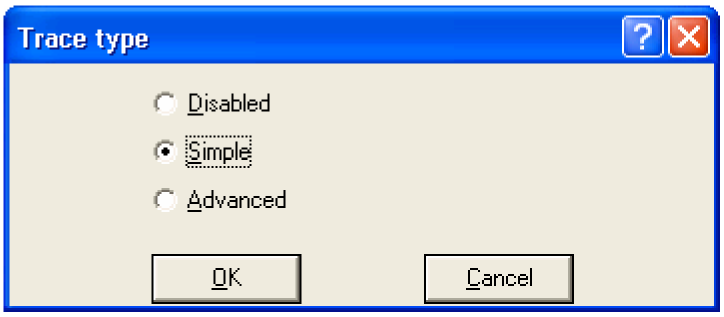
1. Copy the file `RTrs232.c` file from the `<target>/Drivers` directory to the example directory
2. Select the RTA-TRACE tab from the lower panel:



3. Select the configuration pane:

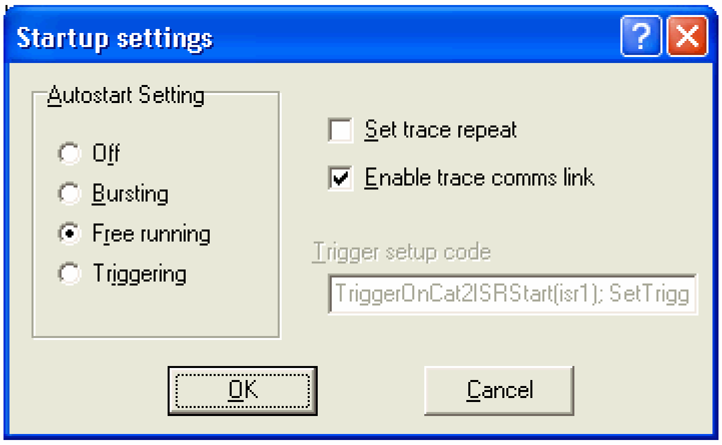


4. then click the Trace Type button and enable Simple tracing:



Disabled [✓ Simple] Advanced

5. Click the Autostart button and select "Free running". Also select "Enable trace comms link":

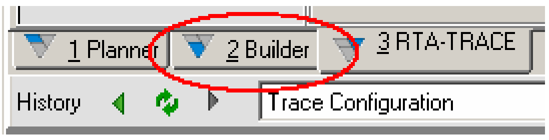

Startup settings

Autostart Setting
- Off
- Bursting
- [✓ Free running]
- Triggering

- Set trace repeat
- [✓ Enable trace comms link]

Trigger setup code: TriggerOnCat2ISRStart(isr1); SetTrigg

6. Go to the Builder tab:



7. Click on the "Custom Build" button, then click "Configure" – The custom Build Options dialog box will be shown.
8. In the 'Build Script' text entry area, add the following beneath 'call rtkbuild.bat':

```
%CC% %COPTS% RTrs232.c
```

9. Edit the linker control line/file to add the `RTserbyt.$(LIBEXT)` library and `RTrs232.$(OBJEXT)` object file.
10. open `target.h`. Ensure that the macro TARGET_IDLE contains code that calls `CheckTraceOutput()` and `UploadTraceData()` when `OSTRACE_ENABLED` is defined. Without this, the trace data will not be uploaded.
11. edit `target.c` to add a stopwatch function with the following prototype:

```
OS_NONREENTRANT(StopwatchTickType)
osTraceStopwatch(void)
```

12. Once these changes have been made, the build file will need to be recreated – click on 'Create 'rtkbuild.bat' ' to do this.
13. The example can now be built by clicking on 'Build Now'. Turn to section 6 for information about using the tracing tools.

5 Building an ERCOS^{EK} Application with Tracing Enabled

RTA-TRACE for ERCOS^{EK} comes with a simple example application called 'TRACE_EG'. The application can be quickly used to verify that RTA-TRACE is correctly set up, as well as giving a simple introduction to the RTA-TRACE tools.

This section describes example application and explains how to build it and examine its behavior across an RS232 link. Some target-specific functions may need to be tailored to your specific target hardware.

The examples described here were written for the Diab toolchain for the Motorola PowerPC 56x but are also applicable to other targets.

5.1 TRACE_EG, the Example Application

The example application is a simple system consisting of four tasks, and two alarms. The tasks are described here, highest priority first.

5.1.1 tskLeader

This task is activated every 250,000 clock ticks by `almLeader`.

The task performs some 'work' (actually a busy-wait loop) for a small amount of time and activates `tskFollower` at its end.,

At the start and the end of the task, a task-tracepoint named `leaderBounds` is logged. The task-tracepoint appears on the tape belonging to `tskLeader`.

5.1.2 tskFollower

`tskFollower` is a very simple task which gets the resource `rscContention`, performs some 'work', and then releases the resource.

The resource `rscContention` is shared with task `tskInterfere` described below.

5.1.3 tskInterfere

This task is activated every 240,000 clock ticks by `almInterfere`; `tskInterfere` therefore 'phases' against `tskLeader` (and subsequently `tskFollower`).

`tskInterfere` is the most complex task in the system, alternating between two execution modes on each run. The execution modes (A and B) perform the following functions:

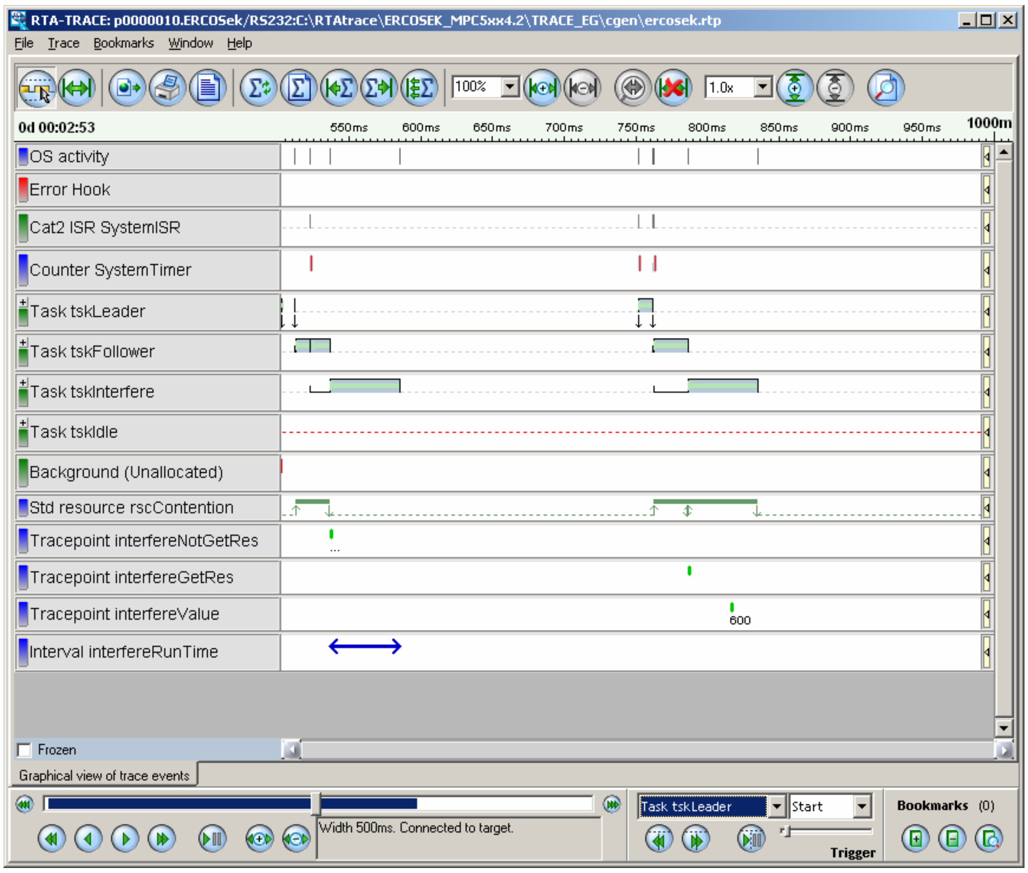
- In mode A, the task logs the tracepoint `interfereGetRes` and gets the resource `rscContention` for the duration of its execution. This occasionally causes `tskFollower` to be deferred until the resource is released.

- In mode B, `tskInterfere` inserts the tracepoint `interfereNotGetRes` with a small block of data for display in the time trace.

In each mode, the task then proceeds to perform some 'work' by entering two loops in sequence, the total duration of which is constant.

- In mode A, the tracepoint `interfereValue` is logged in the break between the two loops, showing a value corresponding to the amount of work carried out so far.
- In mode B, an interval is started in the break which is closed at the end of the task.

The example application can be seen running in the following screenshot of RTA-TRACE running the TimeTrace Visualizer (this is from the ERCOS^{EK} version).



The screenshot shows a typical trace – triggering has been used so that the trace display is fixed relative to the start of `tskLeader`. Note the task "tapes", initially in priority order, and the tracepoint and interval tapes below them. Note that task-tracepoints appear on the tape for the task causing them (e.g. `tskLeader`.)

5.1.4 tskIdle

This task is started with the application mode `modeMain` and starts the two repeating alarms in the system, `almLeader` and `almInterfere`. It then enters a loop polling `UploadTraceData()` to transfer trace data to the host.

This task is also responsible for starting the RTA-TRACE target code. Since the task is already running at the point at which the tracing starts, it is not visible in the TimeTrace Visualizer.

5.2 Building TRACE_EG for ERCOS^{EK}

5.2.1 File Locations

TRACE_EG is installed as a directory tree under your ERCOS^{EK} install directory (e.g. `C:\ETAS\ERSOEK_MPC5xx4.x`) the files of interest are shown below:

…`\TRACE_EG\RTAtrace.cfg`	The RTA-TRACE configuration file.
…`\TRACE_EG\SOURCE\tsk*.c`	The code for the various tasks.
…`\TRACE_EG\SOURCE\target.c`	The target-specific code.
…`\TRACE_EG\SOURCE\tgt_serl.c`	Target-specific serial driver

Open a Command Prompt window and change to the ERCOSEK install directory, e.g.

```
C:\>cd ETAS
C:\ETAS>cd ERCOSEK_MPC5xx4.x
```

5.2.2 Building

1. In the TRACE_EG subdirectory you will find the build scripts.

```
C:\ETAS\ERCOSEK_MPC5xx4.x>cd TRACE_EG
C:\ETAS\ERCOSEK_MPC5xx4.x\TRACE_EG>dir /b
RTAtrace.cfg
project_settings.mk
SOURCE
T32
run.bat
cgen
```

2. Edit the file SOURCE\tgt_serl.c to drive your board's serial hardware. The functions `osTraceInitUART`, `osTraceCommTxByte`, `osTraceCommDisableTxIST`, `osTraceCommEnableTxIST`, and `osTraceCommTxReady` must be amended to allow `RTArs232.a` to drive the board's serial hardware.
3. Build the application using the supplied makefile and batchfile:

```
...\TRACE_EG>run.bat project_settings.mk build
build step "ESCAPE analysing, preprocessing, and
parsing"
analysing include dependencies  trace_eg.esc
preprocessing and parsing       trace_eg.esc
build step "ECCO code generation"

ECCO V4.2 -- (C) ETAS GmbH, Stuttgart
        Ok, doing first analysis pass ...
        analyzing c_bastyp.db ... done in 0.00 sec
...
<ECCO output removed>
...
...\TRACE_EG>
```

4. The build process generates two files which are interesting at this stage; one is the executable file (trace_eg.elf), the other is the RTA-TRACE application description file ercosek.rtp, used by the RTA-TRACE Server.
5. You can now download the generated application into your target board, and start it running.

6 Understanding the RTA-TRACE Client Display

6.1 Start the RTA-TRACE Client

Once you have the target software running, the example application will begin sending trace data.

1. Start the RTA-TRACE Client by choosing 'Start > Programs > RTA-TRACE > RTA-TRACE'.

   The program will appear with File and Help menus and a blank window.

2. Choose 'File > New Connection…'

   You are presented with a dialog requesting the location of the RTA-TRACE Server – this defaults to 'localhost' (i.e. the machine running the RTA-TRACE Client). The address here should be an IP address or a resolvable machine name. 'localhost' is the usual selection here.

3. Next a dialog showing the available OS/ECU-Link combinations which are available on this machine.

4. Choose the combination relevant to your situation (i.e. RTAOSEK-RS232)

   You are presented with a browse dialog.

5. Locate the application description (`.rtp` or `.rta` extension) file that was generated during the build process (this will be probably be in the same directory as the program executable) and click OK.

   The RTA-TRACE Client will connect to the RTA-TRACE Server and start up the *TimeTrace* visualizer by default.

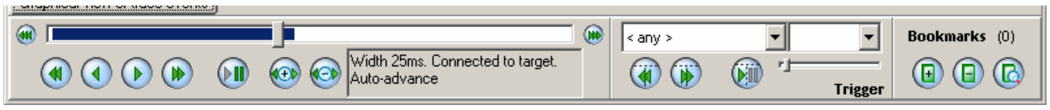
   This shows a sequence of *tapes* (analogous to paper tape), indicating the behavior of the various tasks and other objects being traced.

6. If trace data is not immediately visible, open the connection-configuration dialog (select 'File > Configure Connection…') and set up the communication parameters to match the target hardware.

At the bottom of the Client are the *transport controls*, the *buffer status indicator*, the *timebase controls* and the *trigger controls*. These will now be described.

6.2 Controls

The control area is shown below – descriptions of individual buttons follow.


Width 25ms. Connected to target.
Auto-advance

6.2.1 Transport Controls

These resemble the controls of a tape player or VCR, from left to right:

Back one page through the trace buffer. The size of a page is set using the timebase controls described below.

Back 10% of a page

Forward 10% of a page through the trace buffer

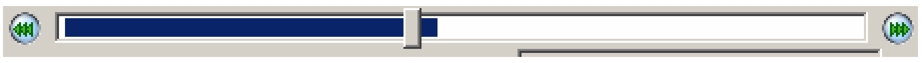
Forward one page through the trace buffer.

Pause/Auto-advance

View earliest data in the trace buffer.

View latest data in the trace buffer.

6.2.2 Buffer Status Indicator



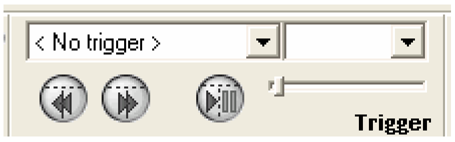
This indicates the relative position of the display in the trace buffer. The grey slider control represents the position of the currently visible trace data. The extent of the buffer is shown by the blue/white area, with the portion of trace buffer filled shown in blue.

6.2.3 Timebase Controls

⊕	Zoom in: This control has the effect of reducing the portion of the trace buffer seen in the visualizer – e.g. from 50ms to 25ms.
⊖	Zoom-out: This control has the effect of increasing the portion of the trace buffer seen in the visualizer – e.g. from 25ms to 50ms.

The timebase control sets the size of the visualizer buffer presented to all visualizer plugins. It is analogous to the timebase control on an oscilloscope. Note that individual plugins may or may not display the whole of this window: for example, the 'magnify' control of the TimeTrace visualizer will zoom in on a portion of the buffer.

6.2.4 Trigger Controls



The trigger control is extremely useful for making sense of the large volumes of trace data available.

The left drop-down list box is the *Trigger Object* box. It contains a list of the objects available for triggering. Next to it is the *Trigger Event* box. The contents of this control vary depending on which Trigger Object is selected (for example, a task has 'Activate', 'Start', and 'Stop').

The slider is the *Pre-trigger Control*. This allows you to adjust the amount of data shown before the trigger point.

The buttons allow you to move between trigger events, as well as toggling the trigger functionality.

6.3 Example trace data

The RTA-TRACE CD contains some example trace data which can be viewed using the RTA-TRACE visualizer. The example trace data can be found on the CD in the directory named `\Example Trace Data\`. The directory contains trace data (in `.rtt` files) as well as descriptions of each example (in a similarly-named `.txt` file). If RTA-TRACE has been correctly installed, the `.rtt` files will open in the RTA-TRACE visualizer when double-clicked.

Further details about use of RTA-TRACE can be found in the online-help files.

Support

For product support, please contact your local ETAS representative.

Office locations and contact details can be found on the ETAS Group website www.etasgroup.com.