RENESAS
BIG IDEAS FOR EVERY SPACE

BASICS OF THE RENESAS SYNERGY™ PLATFORM

Richard Oed

2020.04

CHAPTER 11
EVENT ANALYSIS WITH TRACEX®

CONTENTS

11 EVENT ANALYSIS WITH TRACEX® 03

11.1 An Introduction to TraceX® 03

11.2 Built-in Views and How to Use Them 04

11.3 Installing 04

11.4 Setting Up TraceX® Inside the Development Environments 05

11.4.1 Setting Up TraceX® in e² studio 05

11.4.2 Setting Up TraceX® in IAR Embedded Workbench for Renesas Synergy™ 05

11.5 Configuring the Application for TraceX® 06

11.6 Collecting Trace Data and Starting TraceX® 08

11.6.1 Starting TraceX® in e² studio 08

11.6.2 Starting TraceX® in IAR Embedded Workbench® for Renesas Synergy™ 08

11.7 Viewing and Interpreting the Data 09

Disclaimer 11

11 EVENT ANALYSIS WITH TRACEX®

What you will learn in this chapter:

- What TraceX® is and how it can help you debugging your code.
- How to install and use TraceX® and how to interpret the data collected.

Tracing events at run-time is an important debug capability, as it provides visibility into the sequence of RTOS activities of an application.

In this chapter, we will employ TraceX®, a Windows® workstation-based tool to trace all the events happening when we press the push-button SW4 on the Renesas Synergy Promotion Kit (PK) in the exercise from chapter 10. For this we will modify the program we used there. If you have not done the hands-on of chapter 10, this is not a problem. You can download the complete project from the website for the book (**https://www.renesas.com/synergy-book**). Once downloaded, you only need to import it according to the instructions given in chapter 5.1.3.

11.1 An Introduction to TraceX®

TraceX® is host-based and integrates seamlessly with the Renesas Synergy™ Software Package (SSP). It provides developers with a graphical view of real-time events, enabling them to better understand the behaviour of their system. Visualized are occurrence and sequence of system events like interrupts, context switches or the setting of a semaphore, events which are normally out of the view of standard debugging tools. This way, problems can be easily diagnosed and the system fine-tuned to optimise performance and efficiency.

The trace information itself is stored in a buffer on the target system, with the buffer location and size determined in the Properties of the TraceX® module during configuration. TraceX® implements a circular buffer, which enables the most recent "N" events to be available for inspection, even in the event of system malfunction or other significant events.

Other features and benefits of TraceX® include:

- Full integration with ThreadX® and all other X-Ware™ components, but it runs independently on the host postmortem or at breakpoints
- Detect priority inversions
- Task execution profiles
- Stack usage
- Delta ticks between events
- Performance analysis and bottleneck elimination
- NetX™, and FileX® statistics
- Raw Trace Dumps and Trace Buffers

All these features save a lot of time during the development of an application. There is no need to instrument your program with printf()-statements or with toggling LEDs to visualize a sequence of events in question if you need to isolate a certain problem in your code. This really helps you to save time and energy!

11.2 Built-in Views and How to Use Them

TraceX® provides several means to view and to analyse the data. The buttons on the toolbar allow you to open files with previously recorded and saved events and to navigate through the different captured events based on a setting in a drop down list (event, object, switch or ID), as well as to generate different statistics.

The large visualization window can display events either in *Sequential View Mode*, which is the default and shows events immediately after each other, no matter how much time has elapsed between them, or in *Time View Mode*, where events are shown in a time relative manner. Switching between these view modes is done by selecting the respective tab at the top of the view window. Both views have their own value when analysing the run-time performance of an application, but in most cases using both of them together will get you closer to your solution. The *Sequential View Mode* lets you quickly assess if the events occurred in the right order and the *Time View Mode* if they took place at the right time.

All events are displayed in the context they happened, which can be seen in the left pane of the window in the *System Context*. Hovering with the mouse over a specific context will display more information. On the event line of a specific context, the different events are shown in individual colours with distinctive abbreviations.

More details about the different displays can be found in the documentation of TraceX®, which can be accessed from the TraceX® download page on the Synergy Solutions Gallery. Some of them we will explain in section 11.7. as well.

11.3 Installation

If you used the Platform Installer for one the toolchains and opted to install TraceX® when installing either e² studio or IAR Embedded Workbench for Renesas Synergy™ (IAR EW for Synergy), you can skip the following installation guidelines and move directly to Chapter 11.4. If you used the Standalone Installers, you will need to download TraceX® separately from the Synergy Solutions Gallery, as it is not included in the standard versions of the ISDEs.

On the Solutions Gallery, scroll down to the *Software Tools* section. Click on the TraceX® entry and the page for the tool will show. There, scroll down to *Download* and click on the *Download* button to the right. If you are not logged in, the *My Renesas* login dialog will show. Enter your credentials and click on *Login*. You will be redirected back to the TraceX page. Click again on the *Download* button and accept the license agreement. The download of the archive of the installer named *TraceX_for_Renesas_Synergy.zip* will now start.

While the installer downloads, please take time to review the *installation Instructions*, as the information given there is very important and should be followed closely. It also provides you with information necessary if the automatic installation of the license file fails. This information can later on also be reviewed by opening the file *TraceX-readme.txt*, which is placed in the installation folder of TraceX® after uncompressing the archive.

Once the download is complete, extract the files in the downloaded archive to a new folder of your choice. To start the setup-process right-click on *setup.exe* and select *Run as administrator*.

Figure 11-1: The installer for TraceX

Acknowledge the license agreement for the software once it shows. Follow the instructions of the installer and later on either accept the default path for the installation or change it to a location of your choice. If you do so, please note the new location, as you will need it later to register the tool inside the development environments. At the screen showing the summary, click on *Install* and the installation will start.

Once the installation has finished, the last screen has a check box *Launch the program*. It is selected by default, but we will do that later from inside the ISDE. For the time being, please deselect the check box.

11.4 Setting Up TraceX® Inside the Development Environments

The process of setting up TraceX® is different for both development environments, so please follow the steps relating to your set of development tools.

11.4.1 Setting Up TraceX® in e² studio

As TraceX® is an external program to e² studio, we need to tell the ISDE where the executable can be found, so that we can run it without leaving the development environment.

Inside e² studio, go to *Window → Preferences* and expand first the *C/C++* and then the *Renesas* branch. Select *TraceX* and navigate to or enter the correct location where you installed the TraceX® executable. If you installed it to the default location, this will be *C:\Express_Logic\TraceX_5.2.0\TraceX.exe*. Click on *Apply and Close*.

11.4.2 Setting Up TraceX® in IAR Embedded Workbench for Renesas Synergy™

TraceX® is an external program to the EW for Synergy, so we need to tell the IDE where the executable can be found and which trace-file should be used, so that we can run it without leaving the development environment.

For that, go to *Tools → Configure Tools* inside the IDE and click on *New* once the *Configure Tools* window appear. Enter the following settings: Under *Menu Text*, enter the name of the tool you want to add to the *Tools* menu. In our case, simply enter *TraceX*. Under *Command*, enter or browse to the correct location where you installed the TraceX® executable. If you installed it to the default location, this will be *C:\Express_Logic\TraceX_5.2.0\TraceX.exe*. In the *Argument* line, you will need to enter the name and the location of the file containing the trace samples later on. Enter *$PROJ_DIR$\trace.trx*. This will advise TraceX® to look for a file named *trace.trx* in the directory of your current project. Your window should look like the one in Figure 11-2. Click on *OK* once you made all entries. This will add a *TraceX*-command to the *Tools*-menu.

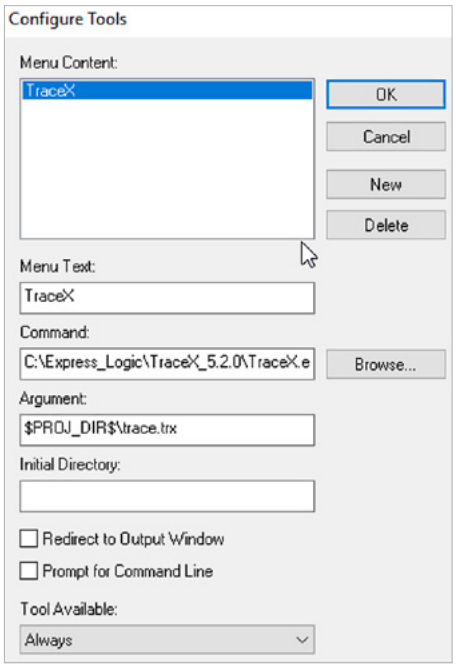

Figure 11-2: In the Configure Tools window, please enter the settings as above

11.5 Configuring the Application for TraceX®

The next step is to configure your application to perform the actual collection of trace events. To enable tracing, we will need to add the *ThreadX®* and USBX™ sources.

For this, make sure your RTOS project from the last chapter is active and switch to the *Synergy Configuration* perspective in e² studio or call the *Synergy Standalone Configurator* inside the IAR EW for Synergy. Switch to the *Threads* tab and select *HAL/Common Modules* in the *Threads* pane. Add the ThreadX® source by clicking on *New → X-Ware → ThreadX → ThreadX Source* in the *HAL/Common Stacks* pane (see Figure 11-3).

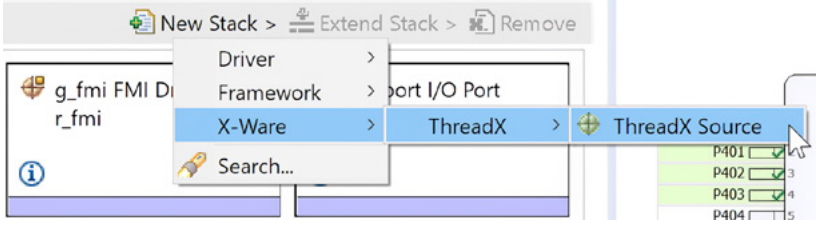

Figure 11-3: The ThreadX® source code needs to be added to the project in order to use TraceX®

Select the newly added *ThreadX* Source, and in the *Properties* view, scroll down to *Event Trace* and change it from *Disabled* to *Enabled*. Note the name of the trace buffer `(g_tx_trace_buffer)` and the buffer size (65536). You will need this information later on.

The module will still indicate an error. Hover with the mouse over it and read the description. It is a warning that multiple symbol linkage errors can occur and refers you to the release notes in case this happens. Remove this error by scrolling to the end in the *Properties* view and change the property *Show linkage warning* from *Enabled* to *Disabled*.

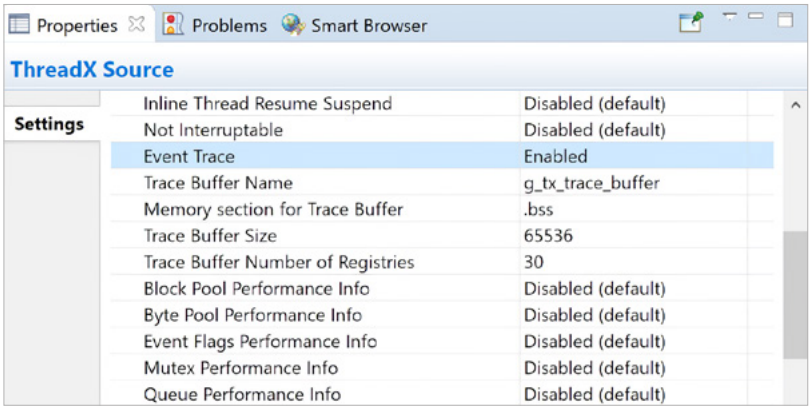

Figure 11-4: The ThreadX® source code needs to be added to the project in order to use TraceX®

Now add the USBX™ source. Select the *Comms Thread* in the *Threads* pane and left-click on the *Add USBX Source* module with the pink bar. Select *New →USBX Source*. This module will also show up in red with the warning of possible multiple symbol linkage errors. Dismiss this error as well by setting the *Show linkage warning* property to *Disabled*.

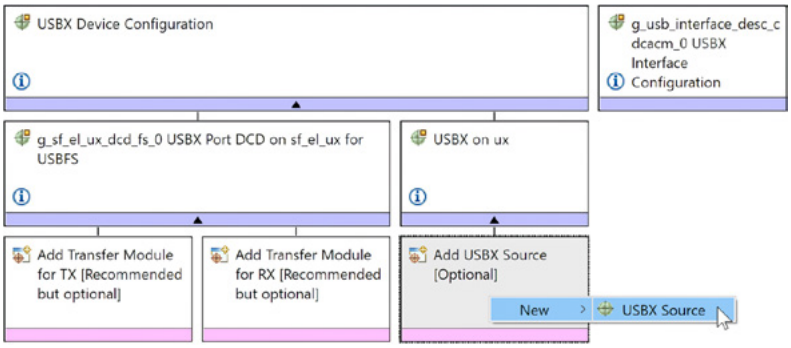

Figure 11-5: For our example, also the source code for USBX™ needs to be added

You will notice that the module named *g_ux_device_class_cdc_acm0* shows an error that "Express Logic source and default pre-built library should not be combined in the stack configuration." This is because if ThreadX source code is added to a project, then the source code for any other Express Logic component should be used as well, not a pre-built library. The reason for this is that a pre-built library is built against the default configuration of ThreadX. So, if the configuration of ThreadX is changed, then all the other Express logic components also need to be built against this new ThreadX configuration.

To resolve this error, click on the *Add USBX Device Class CDC-ACM Source* module. Select *New → USBX Device Class CDC-ACM Source* to add the source code for this module as well. Do not forget to set the *Show linkage warning* property to *Disabled*.

No other changes need to be made, so save the updated SSP configuration by pressing *<ctrl>-<s>* and click on the *Generate Project Content* button to generate the updated versions of the files. Once the generator completes, build the project by clicking on the *build*-button in e² studio or the *make*-button inside IAR EW for Synergy IDE. Please note that a lot of files need to be compiled this time, so the build will take quite a while. On my workstation, the complete process took a little more than two minutes.

11.6 Collecting Trace Data and Starting TraceX®

With the build process complete, it is time to start the debugger, but make sure that your Promotion Kit is still connected with both USB cables (one connected to J19, the other one connected to J5 on the board) to your workstation.

Start the debugger in either one of the development environments. This will download the code to the board. Start the execution of the program and then start your terminal program as in chapter 10. Please make sure that you do not start it before the application is running, as Windows® or the terminal program might not be able to detect the connection properly if started too early.

With the program running, press the push-button SW4 a couple of times to create sufficient system events. All of them will be recorded in the trace buffer, which was created when setting up the frameworks. Pause the program by selecting *Run → Suspend* (or by clicking on the pause symbol in the main menu bar) in e² studio or *Break* in EW for Synergy. Now that TraceX® was able to collect enough data samples, it is time to actually view the events.

11.6.1 Starting TraceX® in e² studio

The next step is to launch TraceX® by selecting *Run → TraceX → Launch TraceX Debugging*. In the dialog box showing, the start address of the trace buffer, as well as its size and the path to TraceX® should be pre-populated. If not, please enter the start address of the trace buffer and the size you noted in section 11.5. Click on *OK*.

11.6.2 Starting TraceX® in IAR Embedded Workbench for Renesas Synergy™

Before you can actually launch TraceX® in EW for Synergy, you need to save the trace buffer to a file in your project directory. For this, open a memory-window in the IDE by selecting *View → Memory → Memory 1* from the menu.

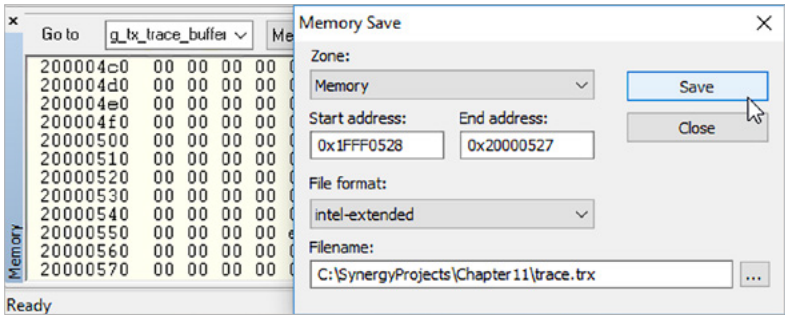

Figure 11-6: It is important to enter the correct path and filename, as TraceX® will not find the data otherwise

In the memory window appearing enter the name (symbolic address) of the trace buffer in the *Goto* field. If you used the settings as suggested in section 11.5, this will be *g_tx_trace_buffer* and hit *Return*. This will display the buffer in the memory window. Next, click on the small arrow to the right of *Memory* drop-down list and select *Memory Save*. In the dialog appearing the start- and end-address of the buffer will be pre-populated. If not, enter the start address as shown in the memory window and add the buffer size for the end address. Check that the file format selected is *intel-extended*. Enter the path to your project directory and append *trace.trx* as filename. Click on Save. This will save the trace buffer to a data file on your hard drive.

Once the data is saved, you can call TraceX® by selecting *Tools → TraceX* from the menu. TraceX® will start and display the trace data.

11.7 Viewing and Interpreting the Data

If TraceX® does not start, but complains about a missing license file, click on *OK*. TraceX® will then exit. Go to the folder where you uncompressed the installation of TraceX® to and copy the file named *tracex.tag* into the installation folder of the program. This will be *c:\Express_Logic\TraceX_5.2.0*, if you installed it into the default location. Go back to your development environment (there is no need to close it for that procedure) and start TraceX® again. It should now start and show *Synergy_Use_Only* as serial number. This procedure is also explained in the file *TraceX-readme.txt* located in the folder of the installer.

Once the TraceX® window shows, you will see the different events recorded in *Sequential View*. To inspect the ones related to our LED and Comms Thread, move the horizontal slider to the right until you see the events similar to the ones in Figure 11-7.

Context changes are represented by the vertical black lines connecting the context lines. The currently selected event is represented by a solid red line. In the example shown in Figure 11-7, this is event 519. Hovering with the mouse over the different events will give you additional information about them.

TraceX® will display the status of a thread. A green horizontal line indicates that a thread is ready, a purple one that the thread is suspended. After start-up, you will only see the ready status. To see all others, you will need to go to *Options → Status Lines* and select *All On*.

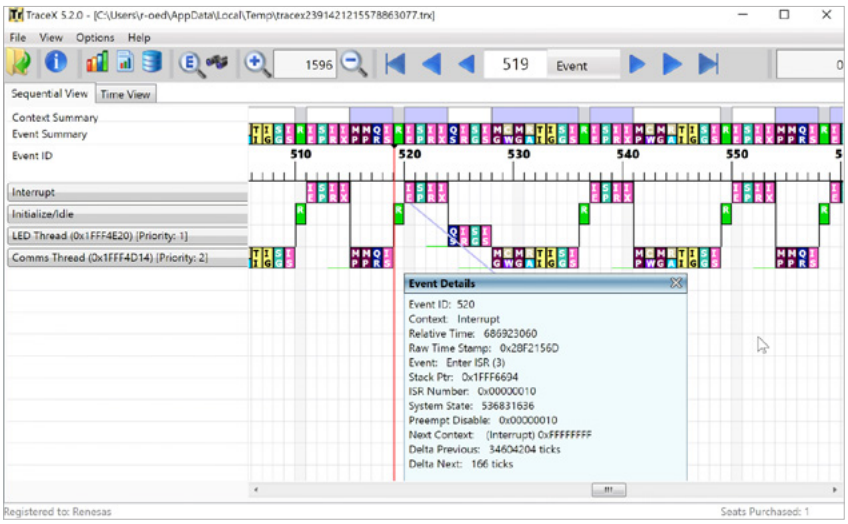

Figure 11-7: The sequence of events as visualized by TraceX®

Each event is represented by a specific colour and abbreviation. In our example, these would be the following (starting at event 519 in Figure 11-7):

- **R:** Running
- **IE:** Interrupt enter
- **SP:** Semaphore put (SW4 semaphore)
- **IR:** Internal thread resume (LED thread)
- **IX:** Interrupt exit
- **QS:** Queue send (CDC queue)
- **IR:** Internal thread resume (Comms thread)
- **SG:** Semaphore get (SW4 semaphore)
- **IS:** Internal thread suspend (LED Thread)
- **MG:** Mutex get (Comms thread)
- **CW:** CDC write (Comms thread)
- **MG:** Mutex get (Comms thread
- **RA:** Device stack transfer all request abort (USB)
- **TI:** Thread identify (Comms thread)
- **IG:** Thread info get (Comms thread)
- **SG:** Semaphore get (ux_transfer_request_semaphore)
- **IS:** Internal thread suspend (Comms thread)

If you want to see detailed information about an event, simply click on one and the *Event Details* will be displayed. A complete list of the events and icons can be found inside TraceX® by going to *View* and selecting one of the legends (for example *ThreadX Legend*) or in the User's Guide, which can be accessed through the round blue icon with the white 'i' on the main menu bar of TraceX®. The Users' Guide will also explain all the other features not covered in this short tutorial, like *Time View* (see Figure 11-8) or the different possibilities for performance analysis and statistics.

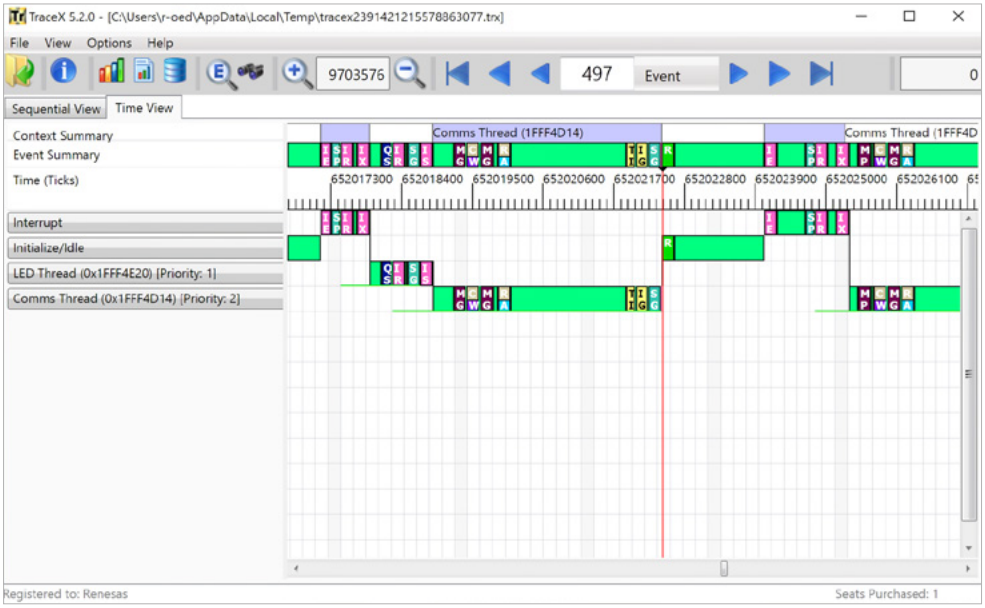

Figure 11-8: When in Time View, TraceX® displays the exact timings between the different events

If you want to collect further TraceX® data, resume the execution of your code in the development environment. Suspend the execution again once enough events have been recorded. Then follow the procedures mentioned in the previous sections to save the trace buffer and to start TraceX®.

All that is left in this exercise, is to terminate the debugger by clicking on the *Terminate* or *Stop Debugging* icon on the main menu bar.

CONGRATULATIONS!

You just finished the last exercise in this book!

Points to take away from this chapter:

- Being able to collect and analyse trace data makes TraceX® a valuable tool in your toolbox.
- Instrumenting a project for TraceX is straightforward and needs only a couple of clicks with the mouse.