Using Tracealyzer with e^2 studio for snapshots on RZ/T2

Application Note PA-035, 2024-01-23

This application note will cover setting up snapshot recording for Tracealyzer for FreeRTOS in e^2 studio. Streaming mode over the on-board J-Link is also possible, but the focus of this Application Note is on how to use e^2 studio.

Requirements

To follow the instructions in this document, you need the following:

- **Target system:** Renesas RZ/T2M or RZ/N2L
    - For example, the RSK+RZT2M board.
- **RTOS:** FreeRTOS (v7.3 or newer)
- **Development tools**
    - e^2 studio
    - Any debug probe supported by e^2 studio; e.g., an on-board J-Link
    - Percepio Tracealyzer with a license matching your RTOS

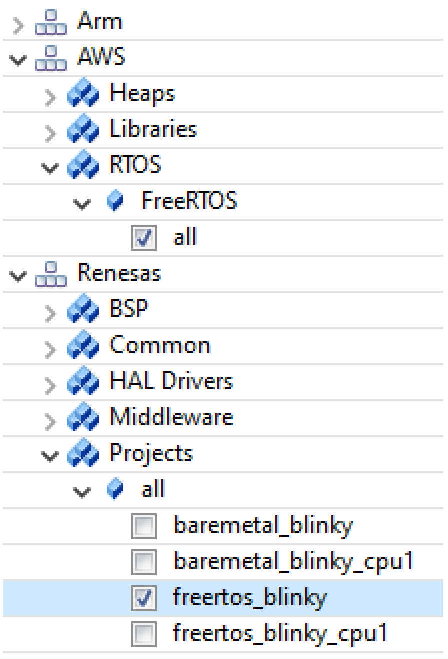
Target-side setup

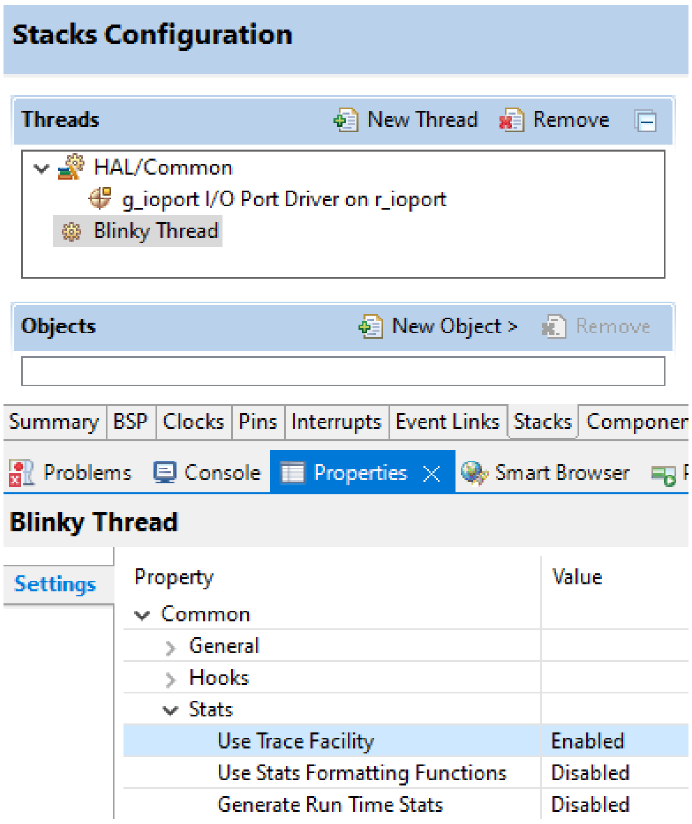
Follow the tutorial chapter in Renesas' *RZ/T2M, RZ/N2L Getting Started with Flexible Software Package* to verify a working setup. Change the configuration or follow the tutorial again but this time include FreeRTOS and use the freertos_blinky project. Verify that it runs as expected.

Integrating the recorder works a bit differently in an FSP project, as some files are automatically generated.

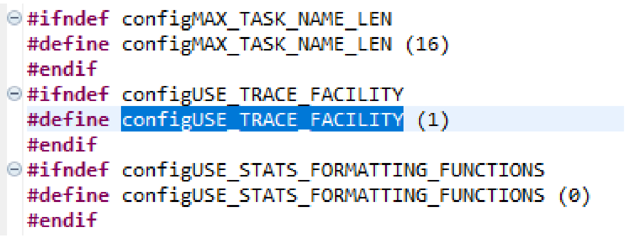
First, we need to enable the trace facility of FreeRTOS:

- In FSP Config, go to the Stacks tab
- In the Threads panel, select the first user thread, in this case "Blinky Thread"
- Switch to, or Open, the Properties window
- Go to Common – Stats – Use Trace Facility and select Enabled
- Save & Generate Project Content





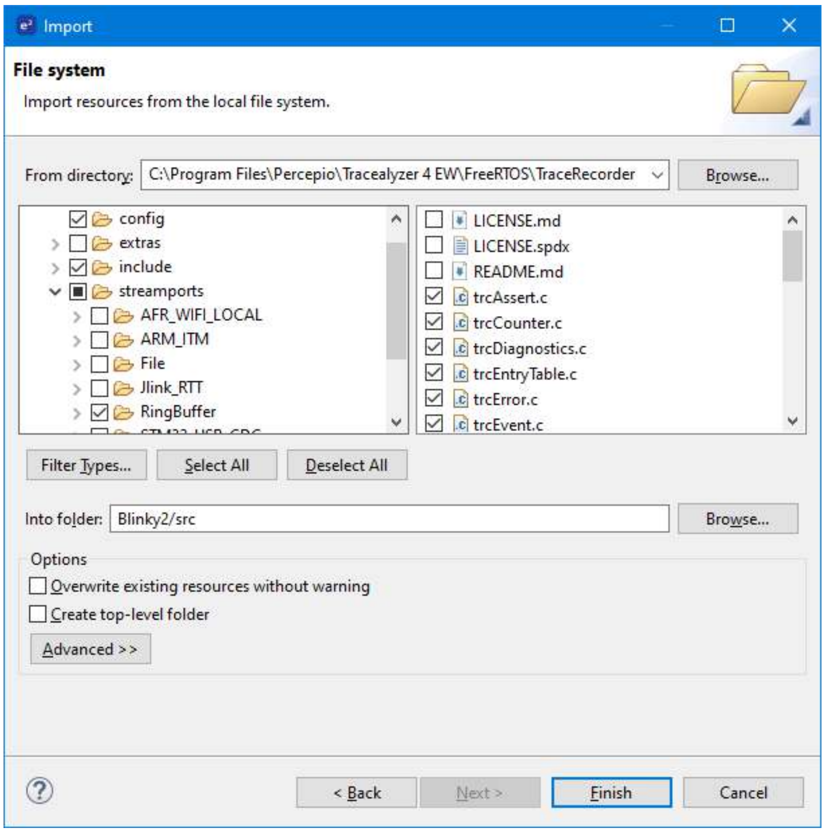
Property	Value
Common	
General	
Hooks	
Stats	
Use Trace Facility	Enabled
Use Stats Formatting Functions	Disabled
Generate Run Time Stats	Disabled



```
#ifndef configMAX_TASK_NAME_LEN
#define configMAX_TASK_NAME_LEN (16)
#endif
#ifndef configUSE_TRACE_FACILITY
#define configUSE_TRACE_FACILITY (1)
#endif
#ifndef configUSE_STATS_FORMATTING_FUNCTIONS
#define configUSE_STATS_FORMATTING_FUNCTIONS (0)
#endif
```

Now, configUSE_TRACE_FACILITY should be set to 1 in rzt_cfg/aws/FreeRTOSConfig.h.

Second, we need to add the necessary files of the trace recorder. Select the ‘src’ folder, click Import – File System and browse to the location of the FreeRTOS trace recorder in your Tracealyzer installation directory. Select all .c files in the root, as well as the directories ‘config’, ‘include’ and ‘RingBuffer’ (under streamports).


Import

File system

Import resources from the local file system.

From directory: C:\Program Files\Percepio\Tracealyzer 4 EW\FreeRTOS\TraceRecorder

- [x] config
- [ ] extras
- [x] include
- [■] streamports
  - [ ] AFR_WIFI_LOCAL
  - [ ] ARM_ITM
  - [ ] File
  - [ ] Jlink_RTT
  - [x] RingBuffer

- [ ] LICENSE.md
- [ ] LICENSE.spdx
- [ ] README.md
- [x] trcAssert.c
- [x] trcCounter.c
- [x] trcDiagnostics.c
- [x] trcEntryTable.c
- [x] trcError.c
- [x] trcEvent.c

Into folder: Blinky2/src

Options
- [ ] Overwrite existing resources without warning
- [ ] Create top-level folder

Now we need to include trcRecorder.h whenever FreeRTOSConfig.h is included, but the latter file is autogenerated. The easiest way to accomplish the inclusion is therefore via the project properties: go to C/C++ Build – Settings – Cross ARM C Compiler – Includes and in the area for "Include files (-include)" add an entry for trcRecorder.h. This will include our header in every C file. You might see some errors being flagged in the code editor and project explorer, but builds should still succeed.

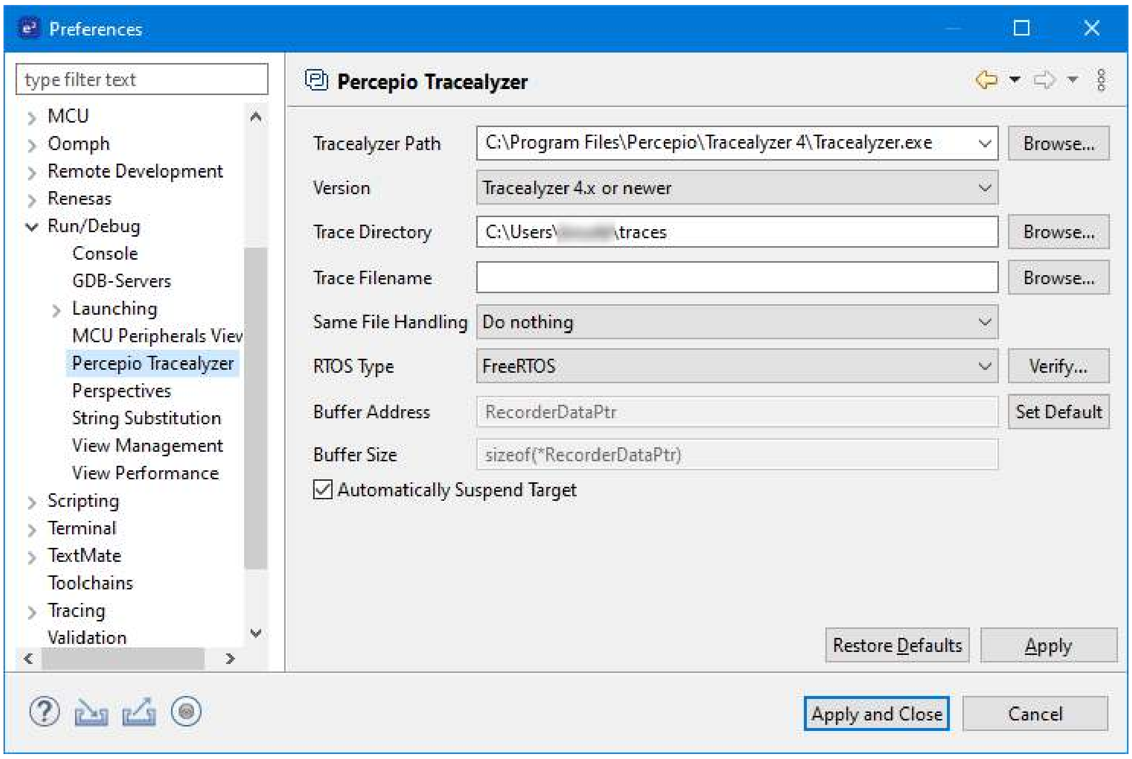
Normally we would enable the recorder in main(), but that is again autogenerated. Instead, add a call to xTraceEnable() in src/hal_entry.c.

```
void R_BSP_WarmStart (bsp_warm_start_event_t event)
{
    if (BSP_WARM_START_POST_C == event)
    {
        /* C runtime environment and system clocks are setup. */

        /* Configure pins. */
        R_IOPORT_Open(&g_ioport_ctrl, &g_bsp_pin_cfg);

        xTraceEnable(TRC_START);
    }
}
```

Next, we need to set a few configuration values. In trcConfig.h, comment out the #error line in the beginning and set TRC_CFG_HARDWARE_PORT to TRC_HARDWARE_PORT_ARMv8AR_A32. In trcKernelPortConfig.h, set TRC_CFG_FREERTOS_VERSION to the appropriate value. The RSK+RZ2TM board has plenty of RAM, so you may want to increase TRC_CFG_STREAM_PORT_BUFFER_SIZE in trcStreamPortConfig.h.

Now your project should (still) build and run.

Since e² studio is Eclipse-based, you can install the Percepio Trace Exporter plug-in via the Eclipse Marketplace (under menu Help), configure it in Preferences, and use it to download snapshots.



Percepio Tracealyzer

Tracealyzer Path	C:\Program Files\Percepio\Tracealyzer 4\Tracealyzer.exe
Version	Tracealyzer 4.x or newer
Trace Directory	C:\Users\[redacted]\traces
Trace Filename	
Same File Handling	Do nothing
RTOS Type	FreeRTOS
Buffer Address	RecorderDataPtr
Buffer Size	sizeof(*RecorderDataPtr)

[✓] Automatically Suspend Target

For questions, please contact support@percepio.com