# Renesas RA Family

## Tracealyzer® for FreeRTOS debugging

### Introduction

FreeRTOS is an RTOS from Amazon Web Services, which is based on a high-performance embedded kernel.

Percepio Tracealyzer® is the premier solution for visual trace diagnostics for developers of RTOS- or Linux-based embedded software systems.

This application note provides procedures to check FreeRTOS thread and object states (referred to as resources) during the development of applications in e² studio. The procedure for starting Tracealyzer® is also explained.

### Target Device

RA6M3 MCU Group (R7FA6M3AH)

### Operating Environment (using UART)

| | |
|---|---|
| Target Board | EK-RA6M3 |
| IDE | e² studio version 2021-04 and FSP v3.0.0 |
| Trace Tool | Percepio Tracealyzer® v4.4.2 |
| OS | FreeRTOS 10.4.3 |
| Toolchains | GNU Arm Embedded Toolchain: 9-2020-q2-major<br>(GNU ARM Embedded 9.3.1.20200408) |
| Cable | FTDI TTL-232R-3V3 (USB to TTL Serial Cable) |

### Operating Environment (using J-Link RTT)

| | |
|---|---|
| Target Board | EK-RA6M3 |
| IDE | e² studio version 2021-04 and FSP v3.0.0 |
| Trace Tool | Percepio Tracealyzer® v4.4.2 |
| OS | FreeRTOS 10.4.3 |
| Toolchains | GNU Arm Embedded Toolchain: 9-2020-q2-major<br>(GNU ARM Embedded 9.3.1.20200408) |

Note: Please download and install tools from the following URL in advance.

- Quick Start Guide for e² studio for RA download site:
  Quick Start Guide for e² studio for RA
- FSP with e² studio installer download site:
  https://github.com/renesas/fsp/releases
- EK-RA6M3 Example Project Bundle - Sample Code download site:
  EK-RA6M3 Example Project Bundle - Sample Code
- Tracealyzer® for FreeRTOS User Manual site:
  Tracealyzer® for FreeRTOS - User Manual
- Percepio Tracealyzer® download site:
  Download Tracealyzer® - Percepio AB

## Contents

## 1. Install FSP with e² studio

Refer to “2.1 Installing the FSP with e² studio Installer” of “Renesas e² studio 2021-04 or higher User’s Manual: Quick Start Guide”.

## 2. Install Tracealyzer®

Refer to Tracealyzer® for FreeRTOS User Manual.

## 3. Creating a project in e² studio

A project generation wizard is available in e² studio to generate an RA project with a project name and the associated device and board, including drivers.

Launch e² studio and choose a workspace folder in the e² studio Launcher. To create a new RA project, follow these steps:

1. Select **File** menu **> New > Renesas C/C++ Project > Renesas R**A.
2. Select the **Renesas RA: Renesas RA C/C++ Project** template. Click **Next** to continue.

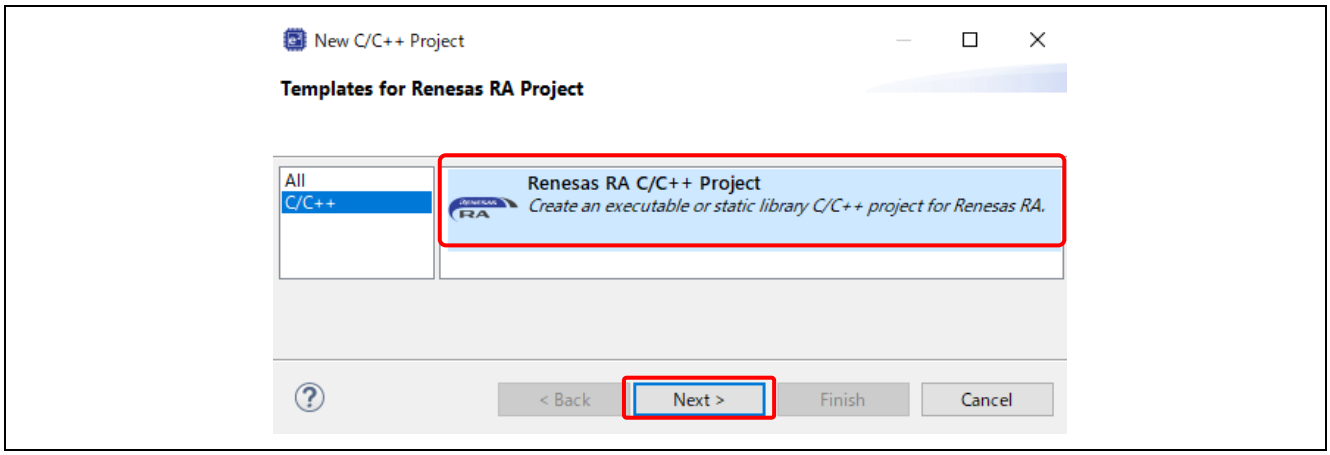



**Figure 1. Template Selection**

3. In the next dialog box, enter a project name and click **Next**.

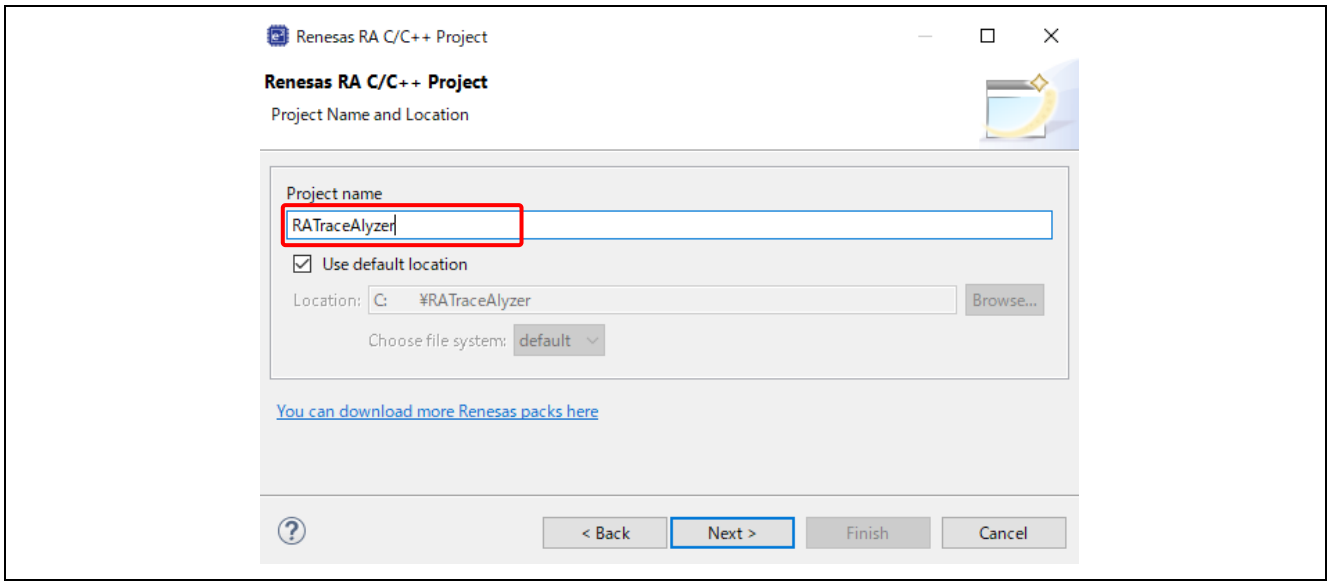



**Figure 2. Project Name and Location**

4. In the device selection dialog, enter device and tool information as follows.
   - FSP version: **3.0.0**
   - Board: **EK-RA6M3**
   - Device: **Auto selected**
   - Language: **C**
   - Toolchain version: Latest GNU Arm Embedded Toolchain approved for use with Renesas RA. (for example, **GCC ARM Embedded 9.3.1.20200408**)
   - Debugger: **J-Link ARM**
   - Click **Next** to continue

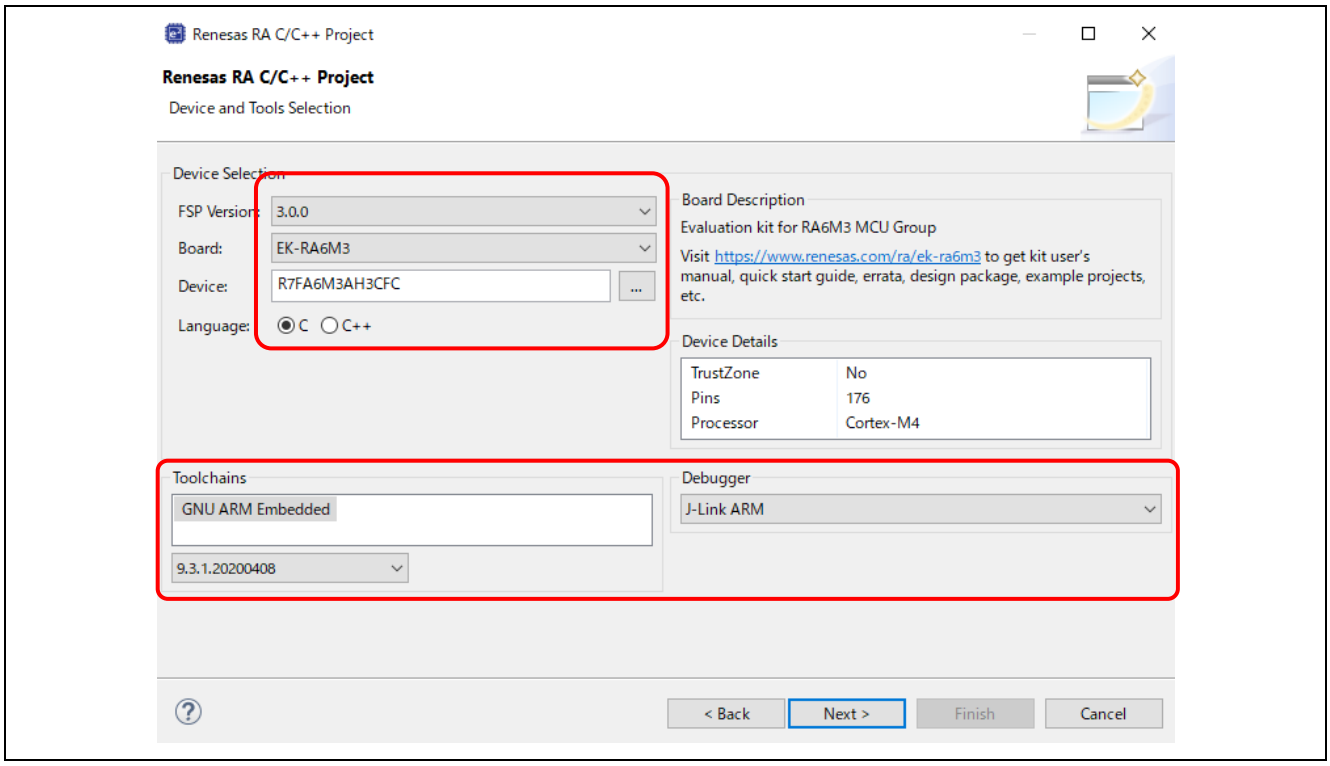



**Figure 3. Create New Project for EK-RA6M3**

5. Build Artifact Selection: **Executable**.
   RTOS Selection: **FreeRTOS**

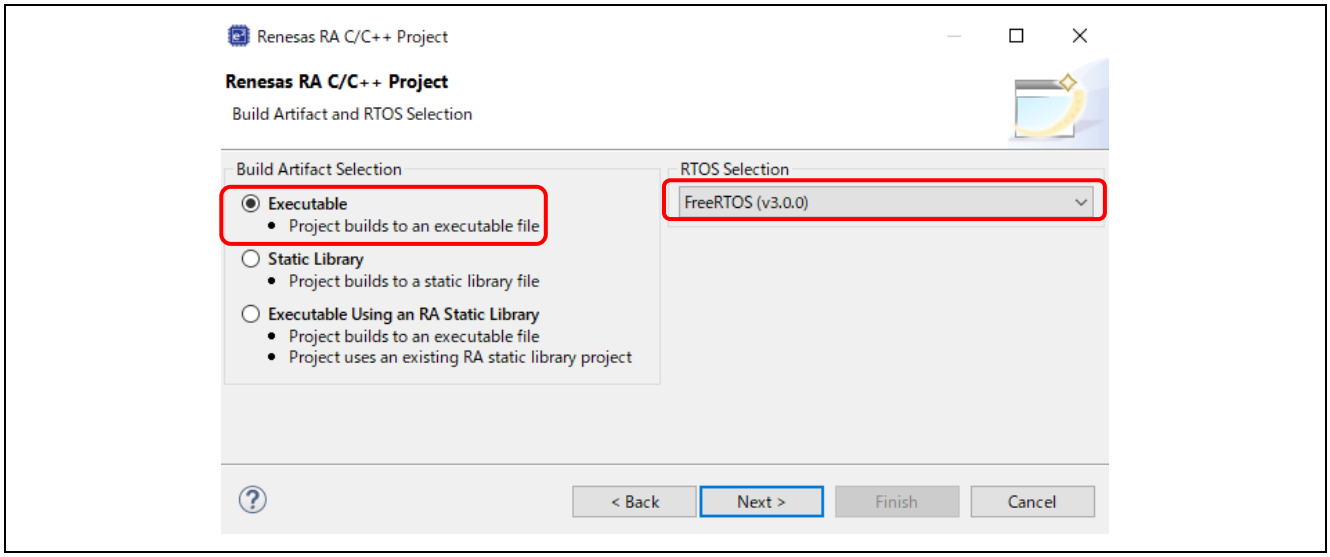



**Figure 4. Build Artifact and RTOS Selection**

6. In the project template dialog, select **FreeRTOS – Binky – Static Allocation** and click **Finish**.

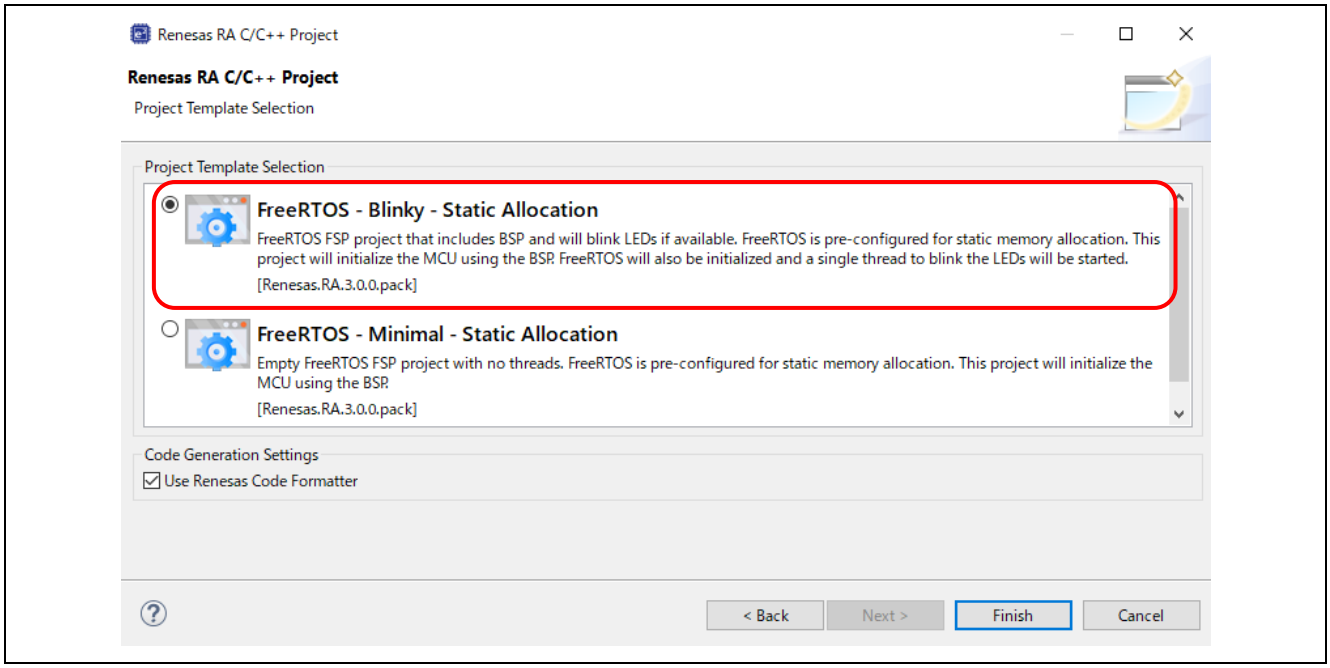



**Figure 5. Project Template Selection**

7. Once complete, e² studio creates a new project with the FSP Configuration perspective open and ready for project configuration.

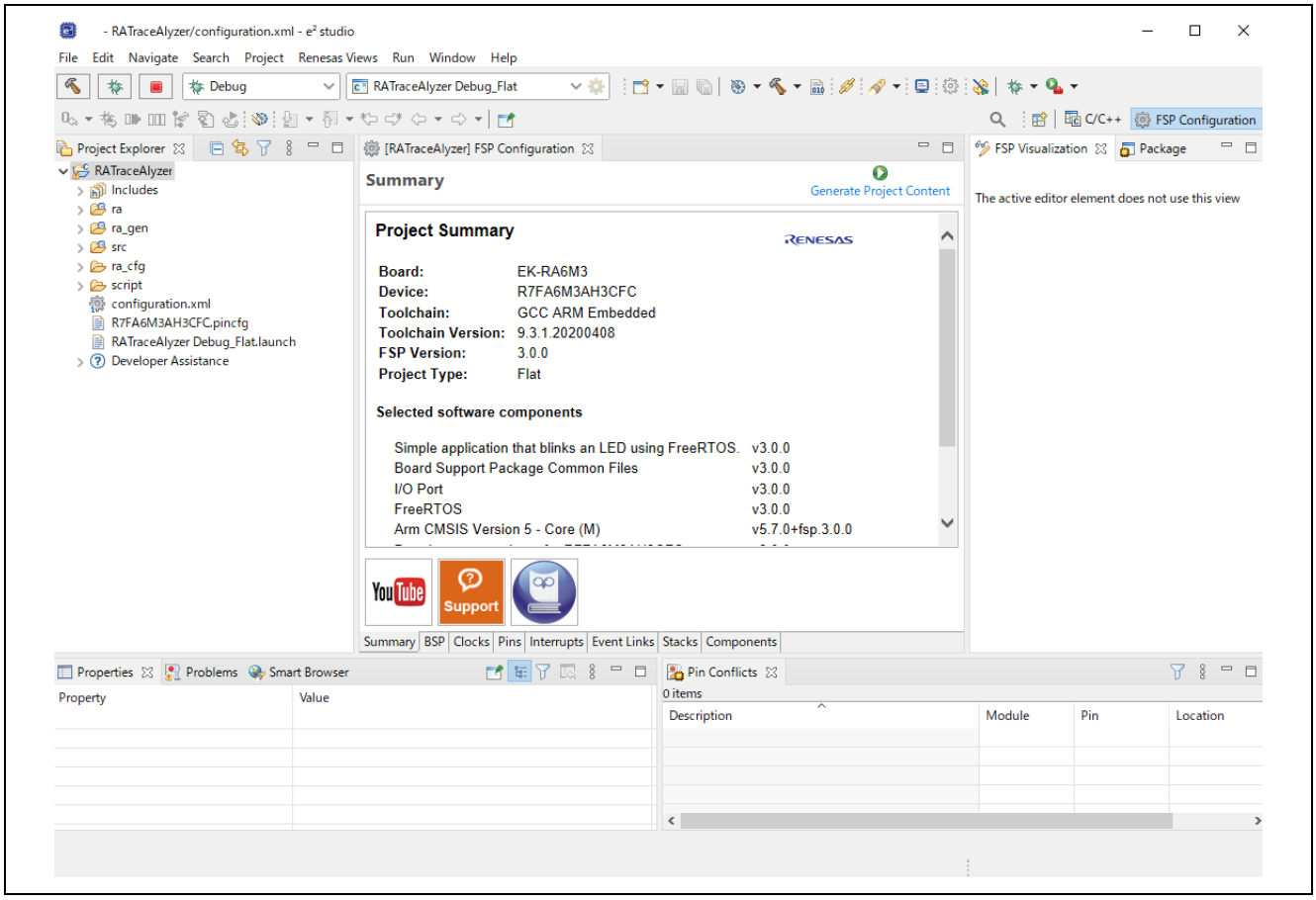



**Figure 6. New Project for EK-RA6M3**

# 4. Debugging via UART with Tracealyzer®

This section describes how to use Tracealyzer® with UART.

## 4.1 Copy and remove Tracealyzer® for FreeRTOS into a project

### 4.1.1 Copy Tracealyzer® for FreeRTOS source under the Tracealyzer® installation folder

Copy the `Program Files\Percecio\Tracealyzer 4\FreeRTOS\TraceRecorder` folder into workspace folder `src` using File Explorer.

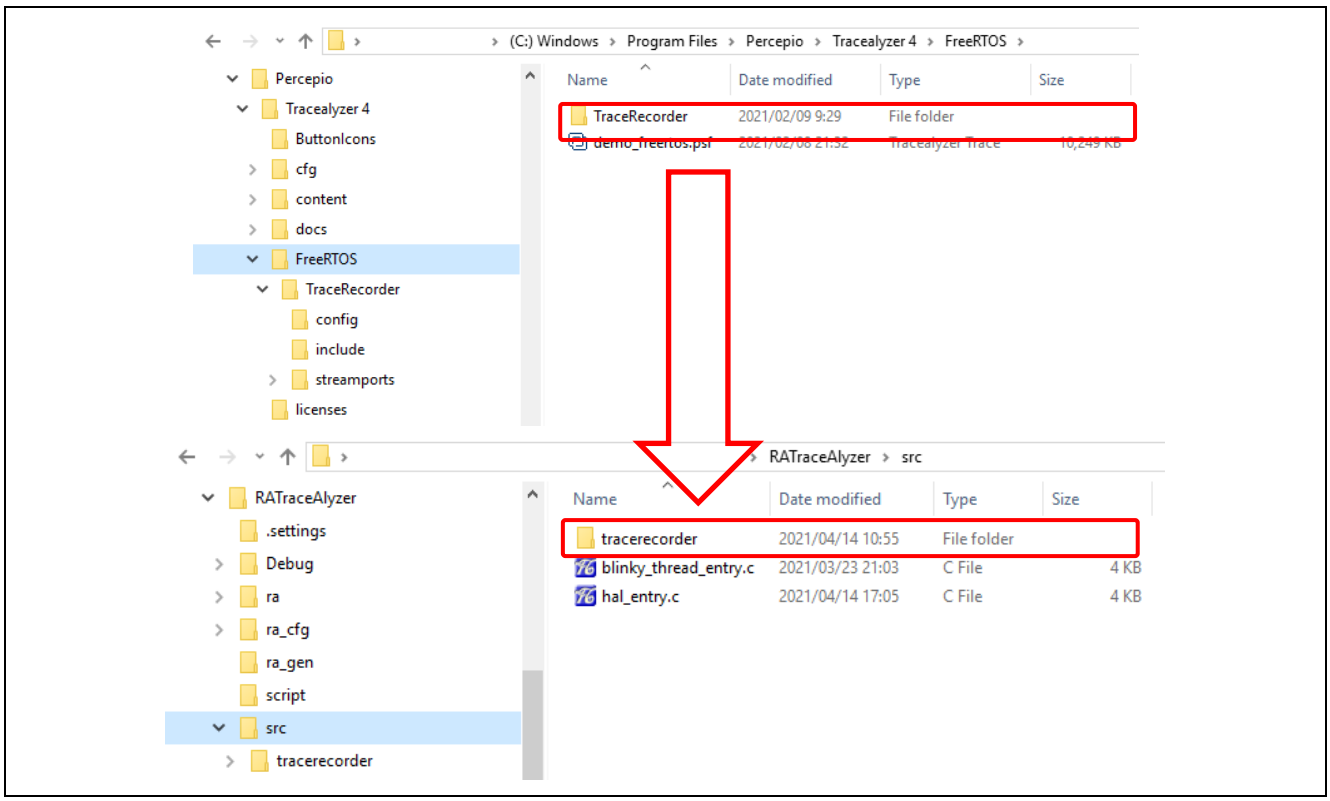



**Figure 7. Copy Folder**

### 4.1.2 Remove unnecessary folders

Remove all sub-folders in workspace folder `src/TraceRecorder/streamports`.

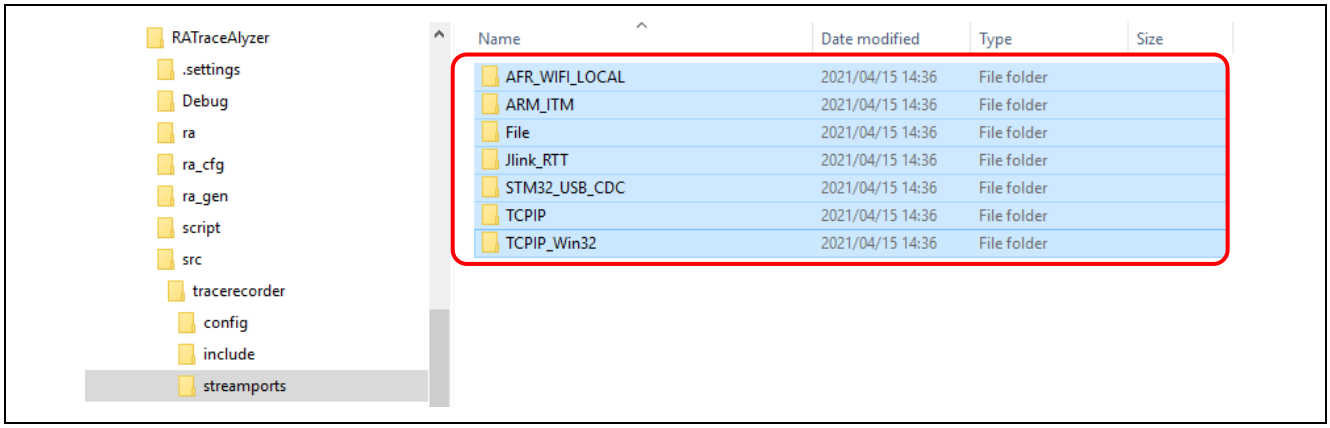



**Figure 8. Remove Folder (UART)**

## 4.2 FSP Configuration

### 4.2.1 UART driver settings

Use P410/P411 (SCI0) for UART communication. EK-RA6M3 is selected SPI0 as default.

| ピン番号 | | | | | | | | Extbus | | タイマ | | | | 通信インタフェース | | | | | | | | | | アナログ | | HMI | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BGA176 | LQFP176 | LGA145 | LQFP144 | LQFP100 | 電源、システム、クロック、デバッグ、CAC | 割り込み | I/Oポート | 外部バス | SDRAM | AGT | GPT | GPT | RTC | USBFS, CAN | SCI0, 2, 4, 6, 8 (30 MHz) | SCI1, 3, 5, 7, 9 (30 MHz) | IIC | SPI, QSPI | SSIE | ETHERC (MII) (25 MHz) | ETHERC (RMII) (50 MHz) | USBHS | SDHI | ADC12 | DAC12, ACMPHS | CTSU | GLCDC, PDC |
| C15 | 40 | D11 | 32 | 21 | - | IRQ4 | P411 | - | - | AGTOA1 | GTOVUP | GTIOC 9A | - | - | TXD0/ MOSI0 /SDA0 | CTS3_ RTS3/ SS3 | - | MOSIA _B | - | ET0_ER XD1 | RMII0_ RXD0_ A | - | SD0 DAT0 _A | - | - | TS07 | PIX01 |
| C14 | 41 | C12 | 33 | 22 | - | IRQ5 | P410 | - | - | AGTOB1 | GTOVLO | GTIOC 9B | - | - | RXD0/ MISO0 /SCL0 | SC | | MISOA _B | - | ET0_ER XD0 | RMII0_ RXD1_ A | - | SD0 DAT1 _A | - | - | TS06 | PIXD0 |

**Figure 9. EK-RA6M3 on Pin Configuration**

Disable SPI0 that is assigned “P410/P411” to use as SCI0 (UART Driver).

- Move to the **Pins** tab.
- Select **Peripherals > Connectivity:SPI > SPI0**.
- Operation Mode: **Disabled**.

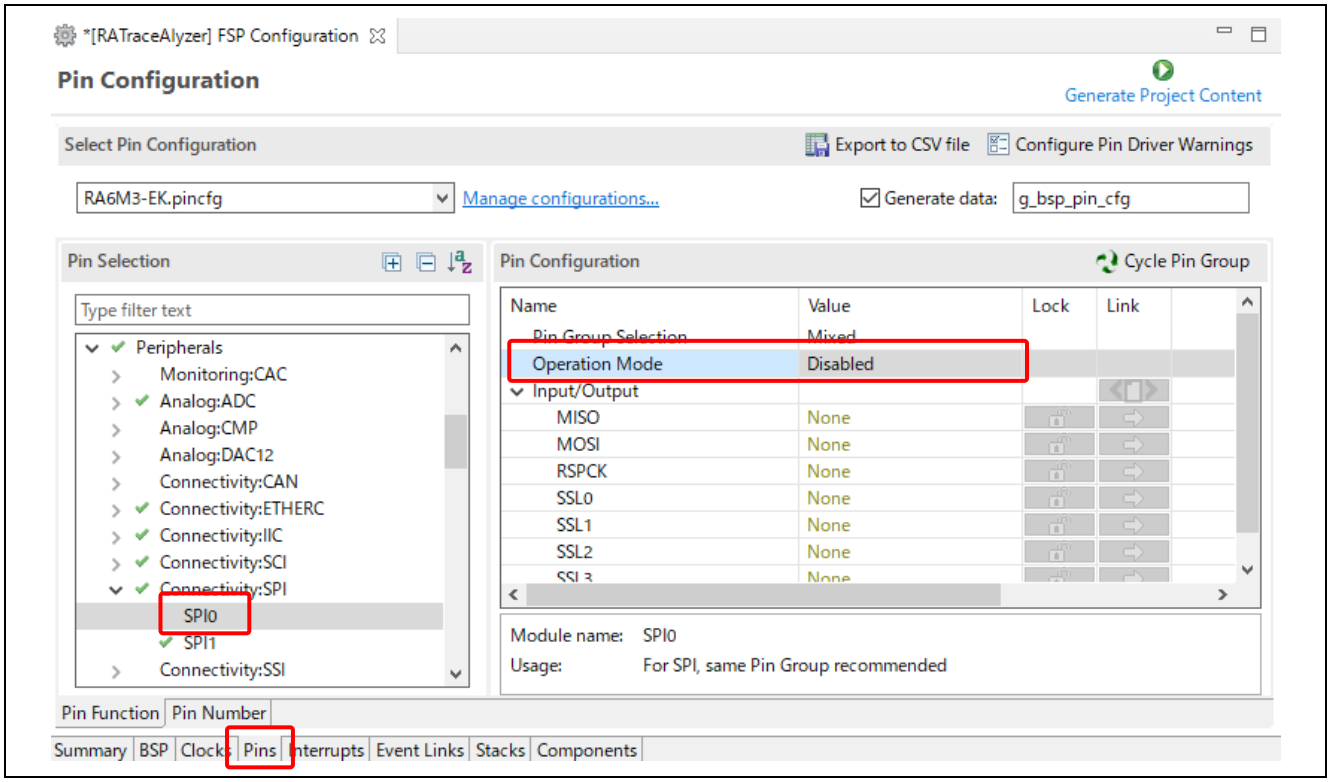



**Figure 10. Disable SPI0 on Pin Configuration**

Move to the **Stacks** tab. Add the UART Driver by clicking **New Stack > Driver > Connectivity > UART Driver on r_sci_uart**.

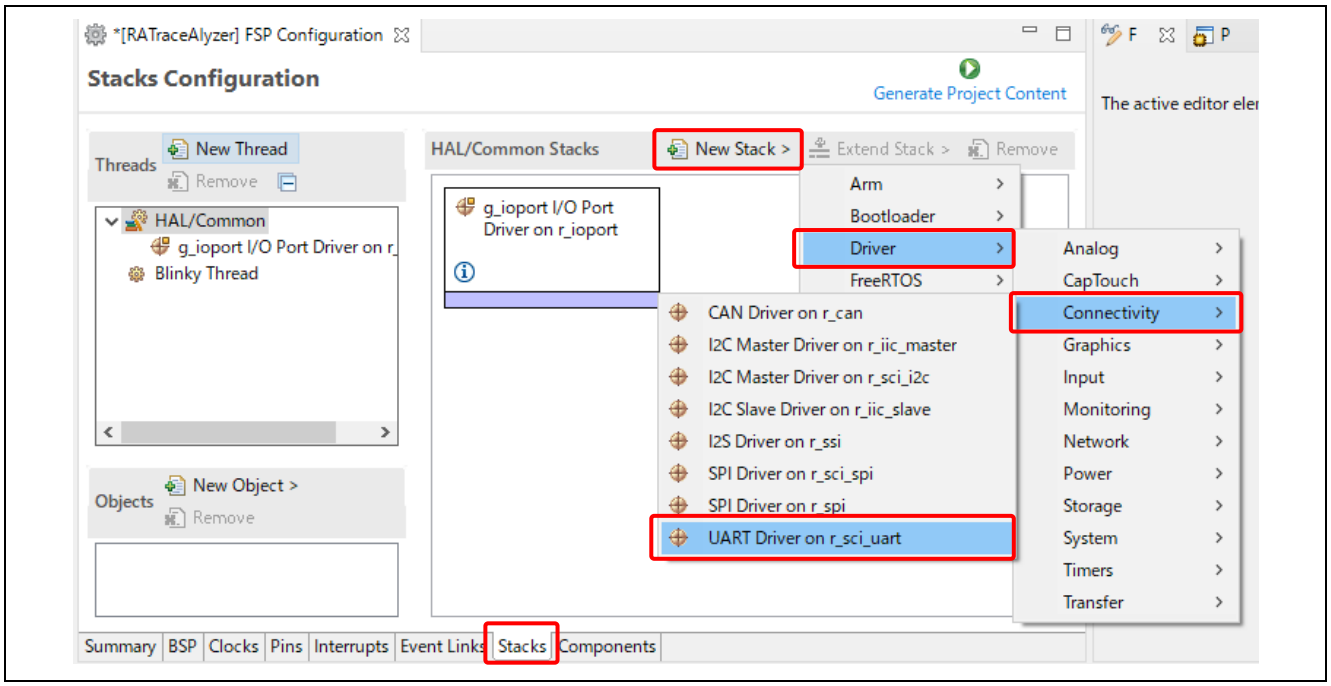



**Figure 11. Add UART Driver on Stacks Configuration**

Change the **Properties** of the UART driver as follows.

- Baud Rate: **921600**
- Callback: **sci_callback_tracealyzer**
- TXD_MOSI: **P411**
- RXD_MISO: **P410**

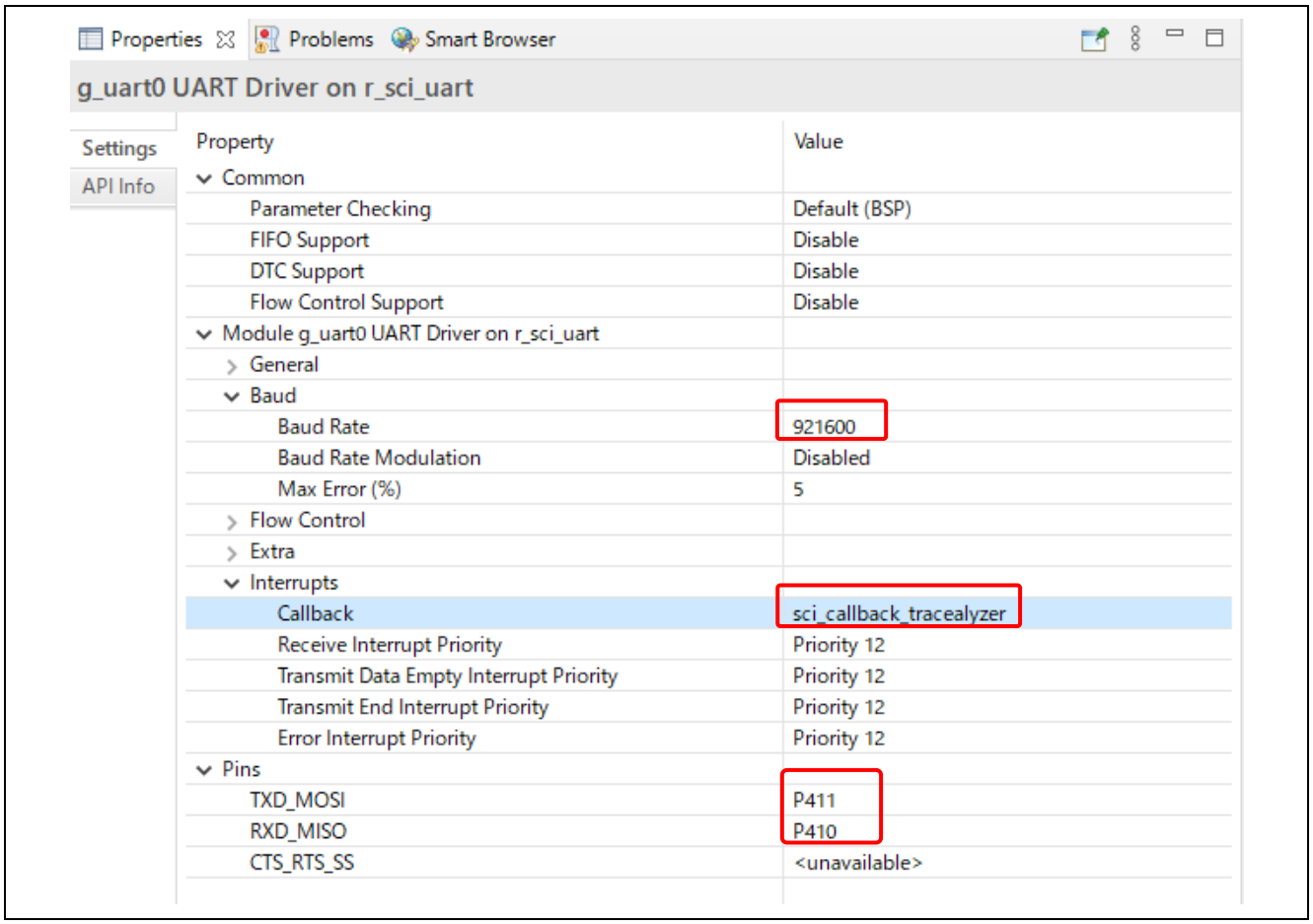



**Figure 12. UART Properties**

### 4.2.2 Blinky Thread settings

Add the Heap to the Blinky Thread as follows.

- Move to the **Stacks** tab.
- Select **Blinky Thread**.
- Click **New Stack > FreeRTOS > Memory Management > Heap 1**.

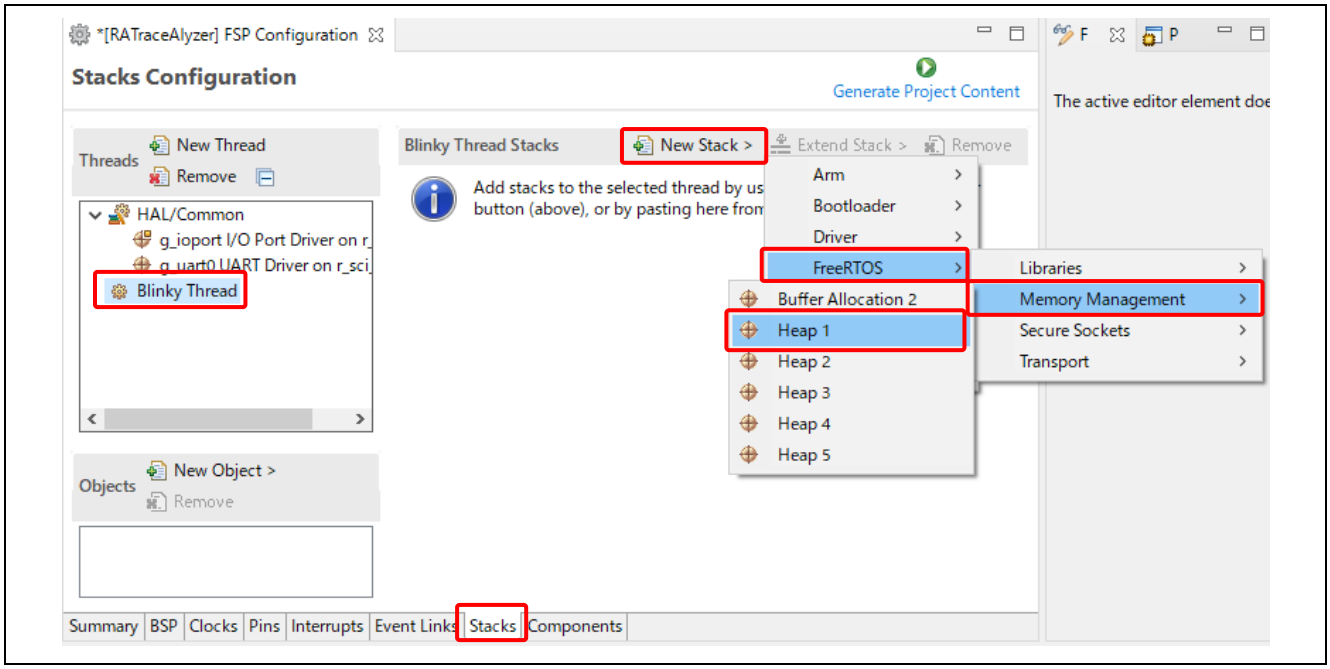



**Figure 13. Add Heap on Stacks Configuration**

Change the **Properties > General** on **Blinky Thread** as follows.

- Minimal Stack Size: **512**
- Use Mutexes: **Enabled**
- Use Recursive Mutexes: **Enabled**
- Use Queue Sets: **Enabled**
- Enable Backward Compatibility: **Enabled**

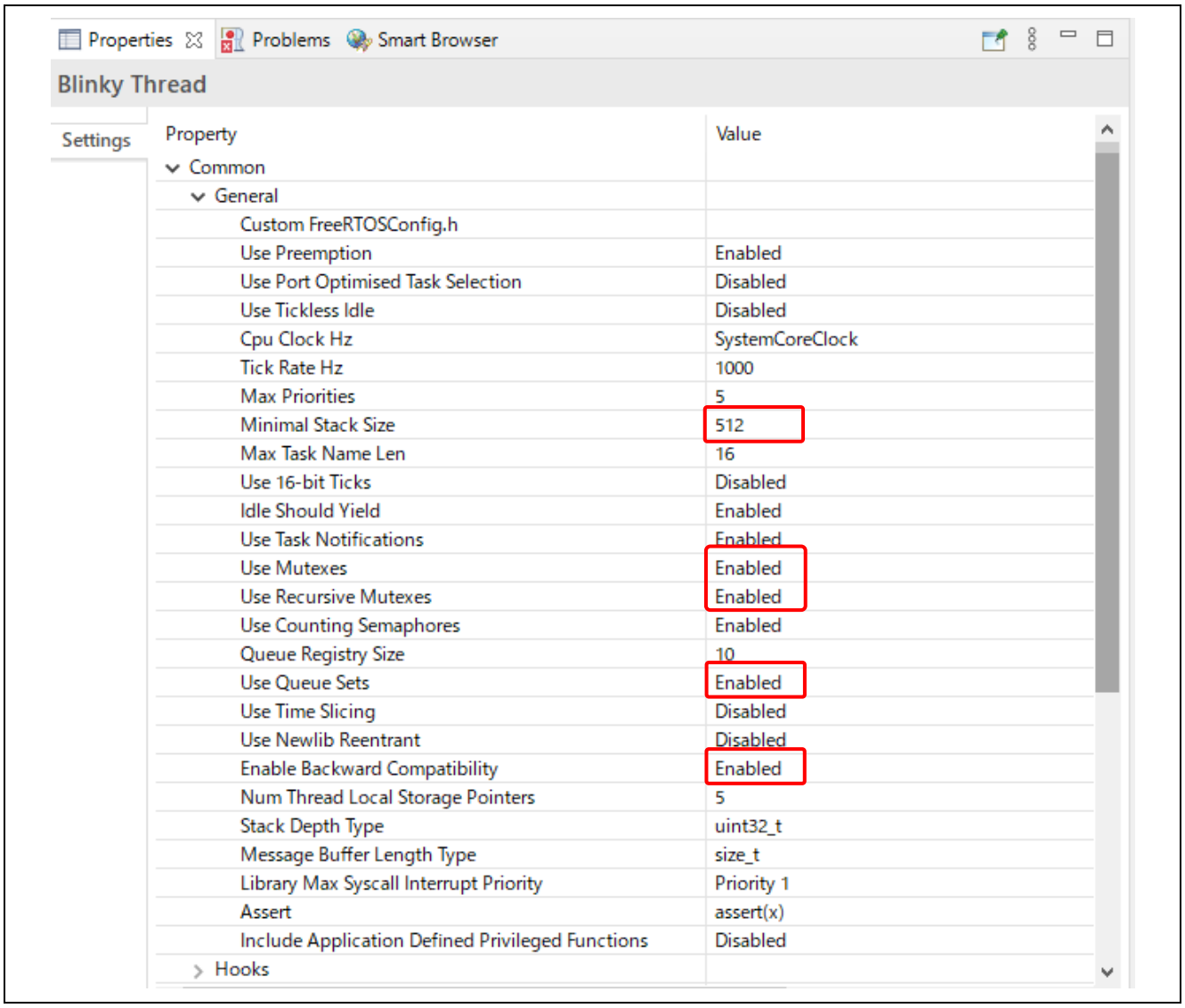



**Figure 14. Blinky Thread Properties 1**

Change the **Properties > Hooks**, **Stats**, **Memory Allocation**, **Timers** on **Blinky Thread** as follows.

- Use Idle Hook: **Disabled**
- Use Malloc Failed Hook: **Enabled**
- Use Trace Facility: **Enabled**
- Use Stats Formatting Functions: **Enabled**
- Support Dynamic Allocation: **Enabled**
- Total Heap Size: **262,144** (256 * 1,024)
- Timer Task Static Depth: **3,072** (1024 * 3)

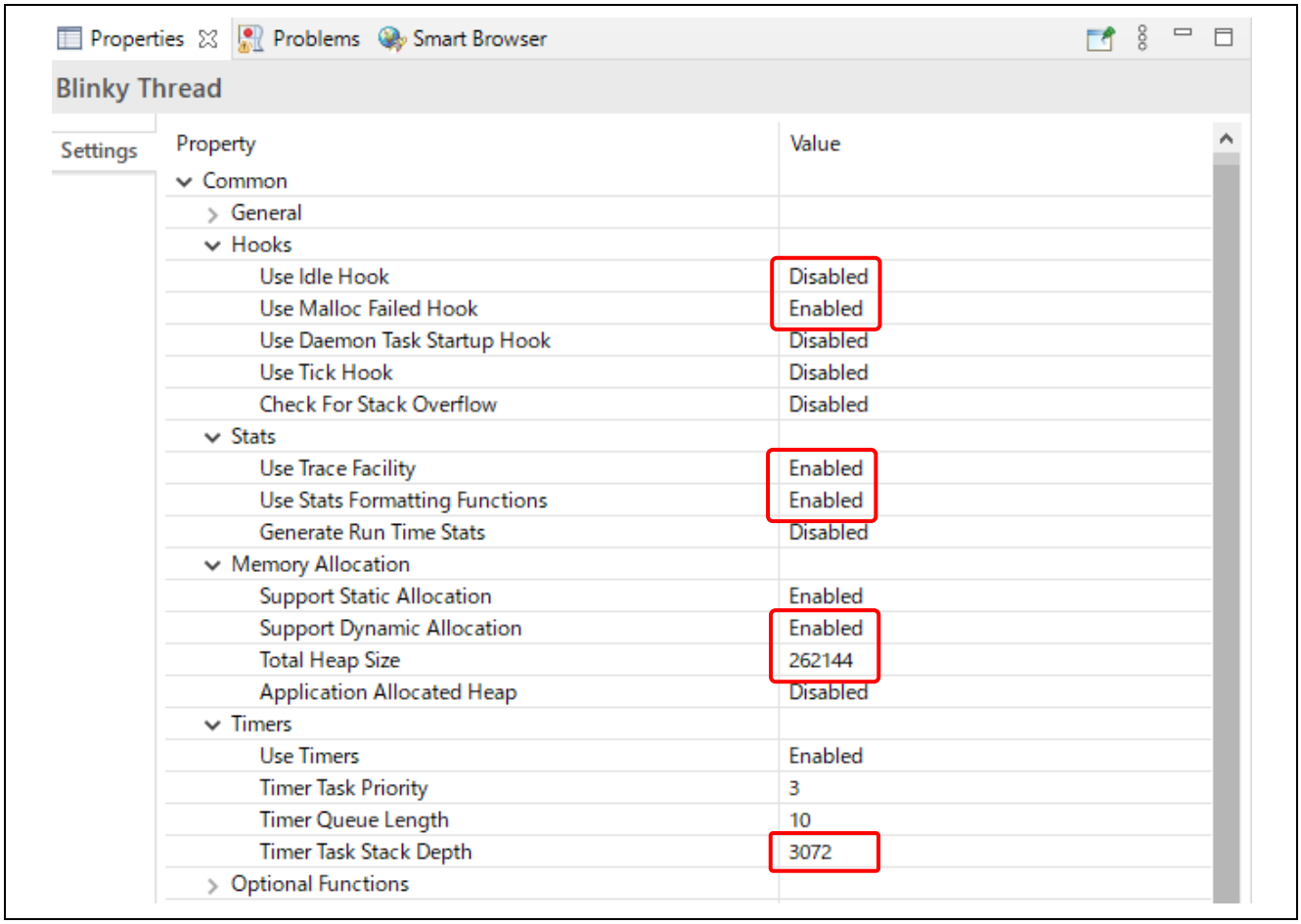



**Figure 15. Blinky Thread Properties 2**

Change the Properties (Optional Functions, RA, Logging) on Blinky Thread as follows.

- uxTaskGetStackHighWaterMark() Function: **Enabled**
- eTaskGetState() Function: **Enabled**
- xTimerPendFunctionCall() Function: **Enabled**
- xTaskAbortDelay() Function: **Enabled**
- Hardware Stack Monitor: **Enabled**
- Logging Include Time and Task Name: **Enabled**

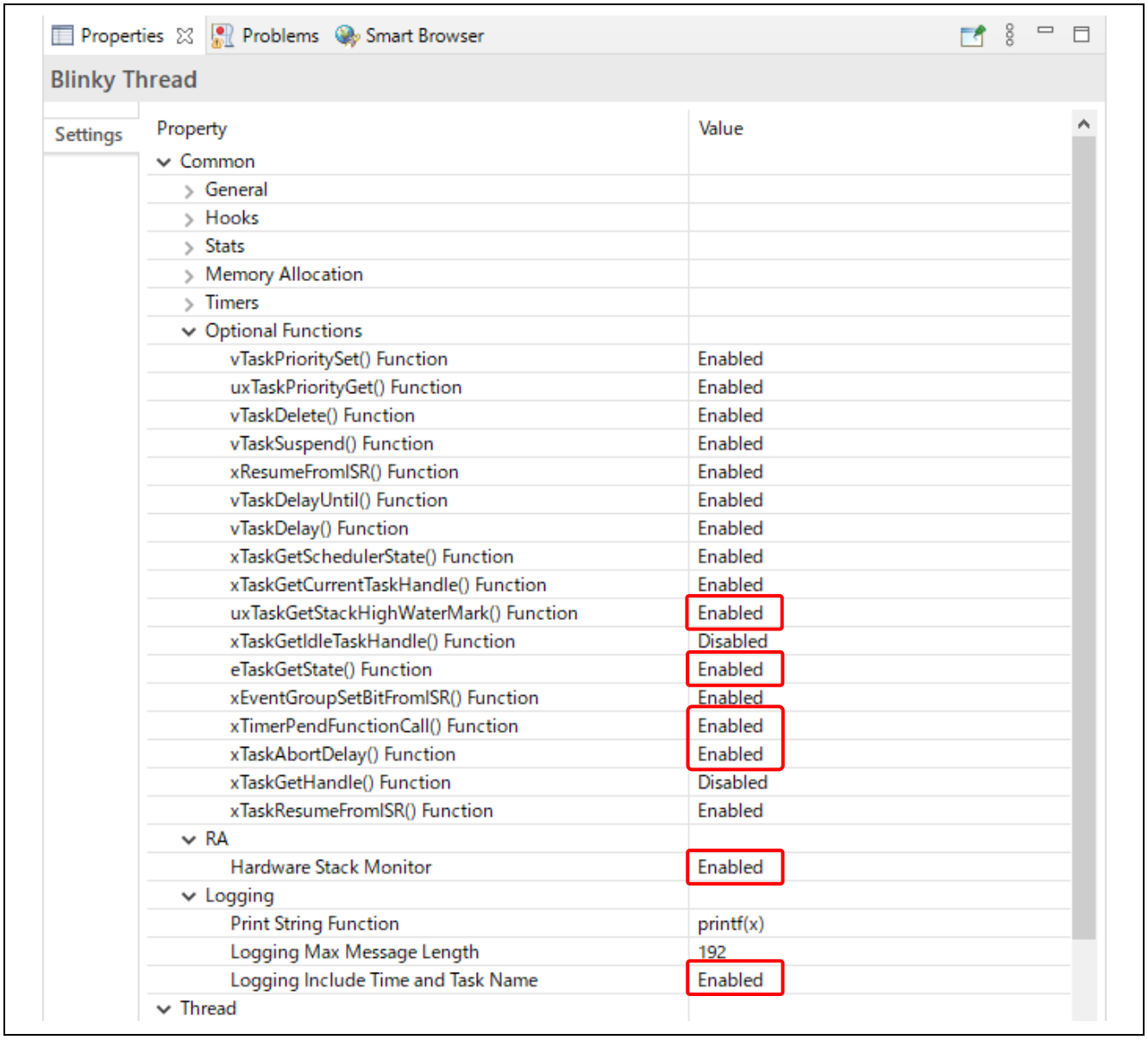
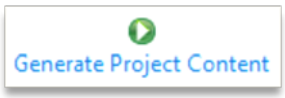



**Figure 16. Blinky Thread Properties 3**

### 4.2.3 Generate Project Content

Click on the 

button to generate the source files.

## 4.3 Code editing for Tracealyzer® connections

### 4.3.1 Create folder and file for UART

- Create the **uart** folder and `trcSteamingPort.c` and `trcStreamingPort.h` files, for example.
- Make `trcSteamingPort.c` and `trcStreamingPort.h` as shown in Figure 18 and Figure 19.

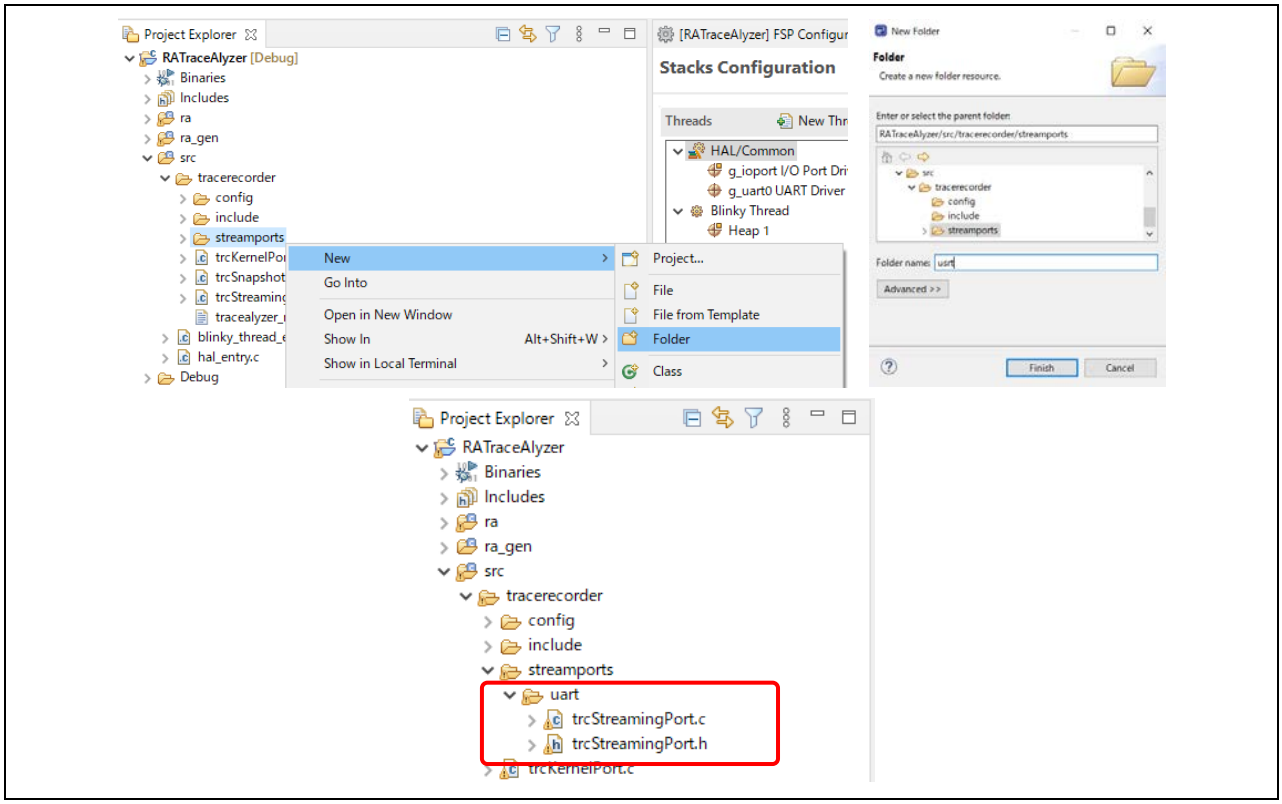



**Figure 17. Create folder and file**

— Create file: `trcStreamingPort.c`

```
#include "bsp_api.h"
#include "trcRecorder.h"
#include "r_sci_uart.h"
#include "r_uart_api.h"
#include <string.h>
#include "semphr.h"
#if (TRC_CFG_RECORDER_MODE == TRC_RECORDER_MODE_STREAMING)
#if (TRC_USE_TRACEALYZER_RECORDER == 1)
static uint8_t s_u8_string[1024];
static signed portBASE_TYPE xHigherPriorityTaskWoken;
static uint8_t sci_buffer[1024];
static uint32_t sci_current_received_size = 0;
extern sci_uart_instance_ctrl_t g_uart0_ctrl;
extern SemaphoreHandle_t semaphore_handle_1;

int32_t trcUartWrite(void* data, uint32_t size, int32_t *ptrBytesWritten)
{
    fsp_err_t err   = FSP_SUCCESS;
    int32_t error_code = -1;
    if(size < sizeof(s_u8_string))
    {
            memcpy(s_u8_string, data, size);
            /* Writing to terminal */
            err = R_SCI_UART_Write (&g_uart0_ctrl, s_u8_string, size);
            if(err == FSP_SUCCESS)
            {
                    xSemaphoreTake( semaphore_handle_1, portMAX_DELAY );
                    *ptrBytesWritten = size;
                    error_code = 0;
            }
    }
    return error_code;
}

int32_t trcUartRead(void* data, uint32_t size, int32_t *ptrBytesRead)
{
    if(sci_current_received_size == size)
    {
            memcpy(data, sci_buffer, sci_current_received_size);
            *ptrBytesRead = sci_current_received_size;
            sci_current_received_size = 0;
    }
    return 0;
}

void sci_callback_tracealyzer(uart_callback_args_t *p_args)
{
    if(UART_EVENT_RX_CHAR == p_args->event)
    {
            sci_buffer[sci_current_received_size] = (uint8_t ) p_args->data;
            if(sci_current_received_size == (sizeof(sci_buffer) - 1)) /* -1 means string
terminator after "\n" */
            {
                    sci_current_received_size = 0;
            }
            else
            {
                    sci_current_received_size++;
            }
    }
    else if(UART_EVENT_TX_COMPLETE == p_args->event)
    {
            xHigherPriorityTaskWoken = pdFALSE;
            xSemaphoreGiveFromISR(semaphore_handle_1, &xHigherPriorityTaskWoken);
            portYIELD_FROM_ISR( xHigherPriorityTaskWoken );
    }
    else
    {
    }
}
#endif /*(TRC_USE_TRACEALYZER_RECORDER == 1)*/
#endif /*(TRC_CFG_RECORDER_MODE == TRC_RECORDER_MODE_STREAMING)*/
```

**Figure 18. Create trcStreamingPort.c**

— Create file: `trcStreamingPort.h`

```
#ifndef TRC_STREAMING_PORT_H
#define TRC_STREAMING_PORT_H

#define TRC_STREAM_PORT_READ_DATA(_ptrData, _size, _ptrBytesRead) trcUartRead(_ptrData, _size,
_ptrBytesRead)
#define TRC_STREAM_PORT_WRITE_DATA(_ptrData, _size, _ptrBytesSent) trcUartWrite(_ptrData, _size,
_ptrBytesSent)

#endif
```

**Figure 19. Create the trcStreamingPort.h**

### 4.3.2 Add include file to task.c and timers.c

- `#include "trcRecorder.h"`

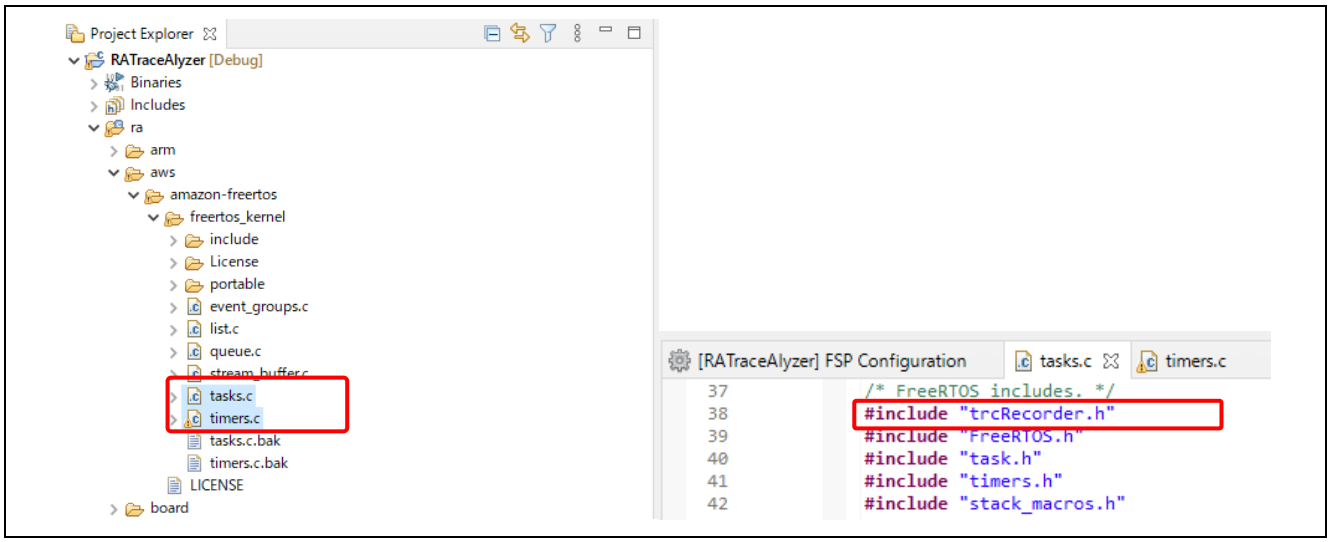



**Figure 20. Add include to freertos_kernel**

### 4.3.3 Add include files to trcKernelPort.c and trcStreamingRecorder.c

- `#include "bsp_api.h"`
- `#include "trcRecorder.h"`

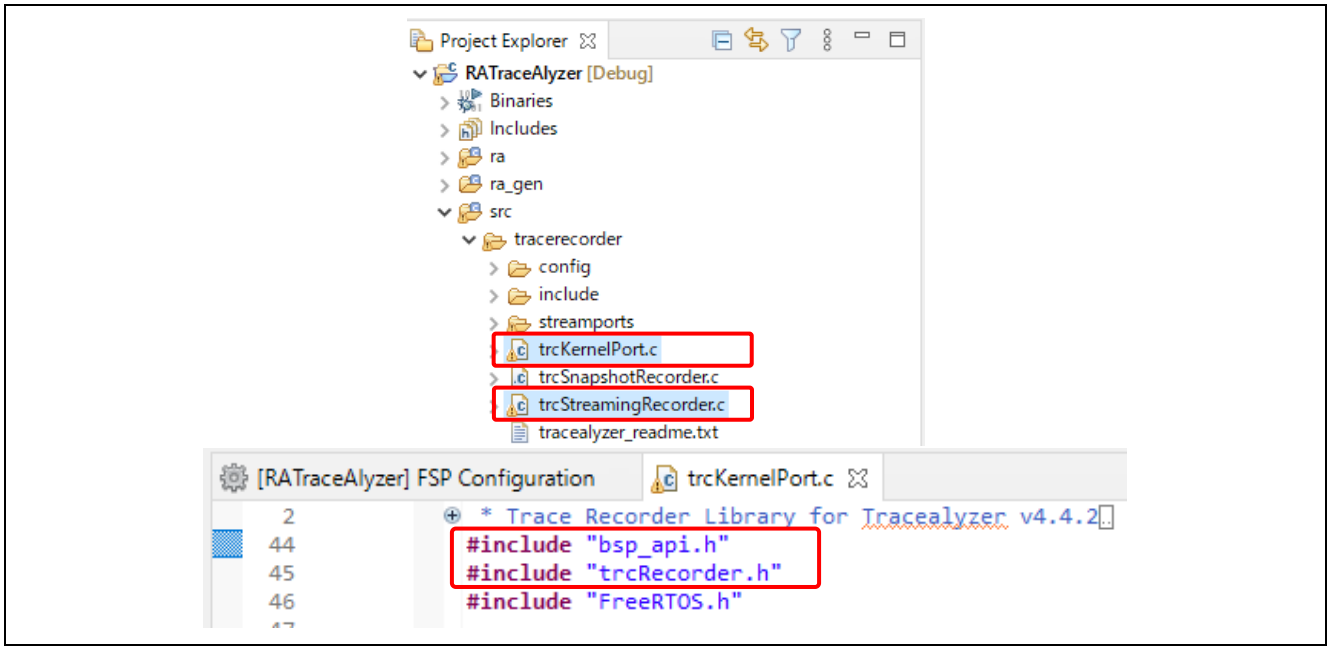



**Figure 21. Add include tracerecorder**

### 4.3.4 Change macro definitions in trcConfig.h

Change only the red part of the macro definitions in `trcConfig.h` as follows.

- `#include “bsp_api.h”`
- `//#error "Trace Recorder: Please include your processor's header file here and remove this line."`
- `#define TRC_CFG_HARDWARE_PORT TRC_HARDWARE_PORT_ARM_Cortex_M`
- `#define TRC_CFG_RECORDER_MODE TRC_RECORDER_MODE_STREAMING`
- `#define TRC_CFG_FREERTOS_VERSION TRC_FREERTOS_VERSION_10_4_1`

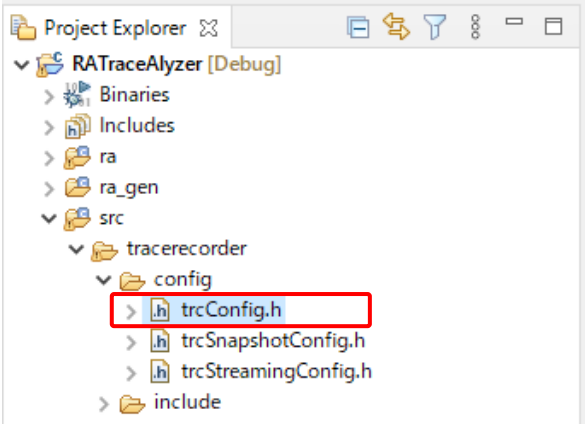



**Figure 22. Change the define in trcConfig.h**

### 4.3.5 Add Include path to e² studio properties

Select menu **Project > Properties**, then click **Settings > Includes** to add the Include path.

- `"${workspace_loc:/${ProjName}/src/tracerecorder}"`
- `"${workspace_loc:/${ProjName}/src/tracerecorder/config}"`
- `"${workspace_loc:/${ProjName}/src/tracerecorder/include}"`
- `"${workspace_loc:/${ProjName}/src/tracerecorder/streamports/uart}"`

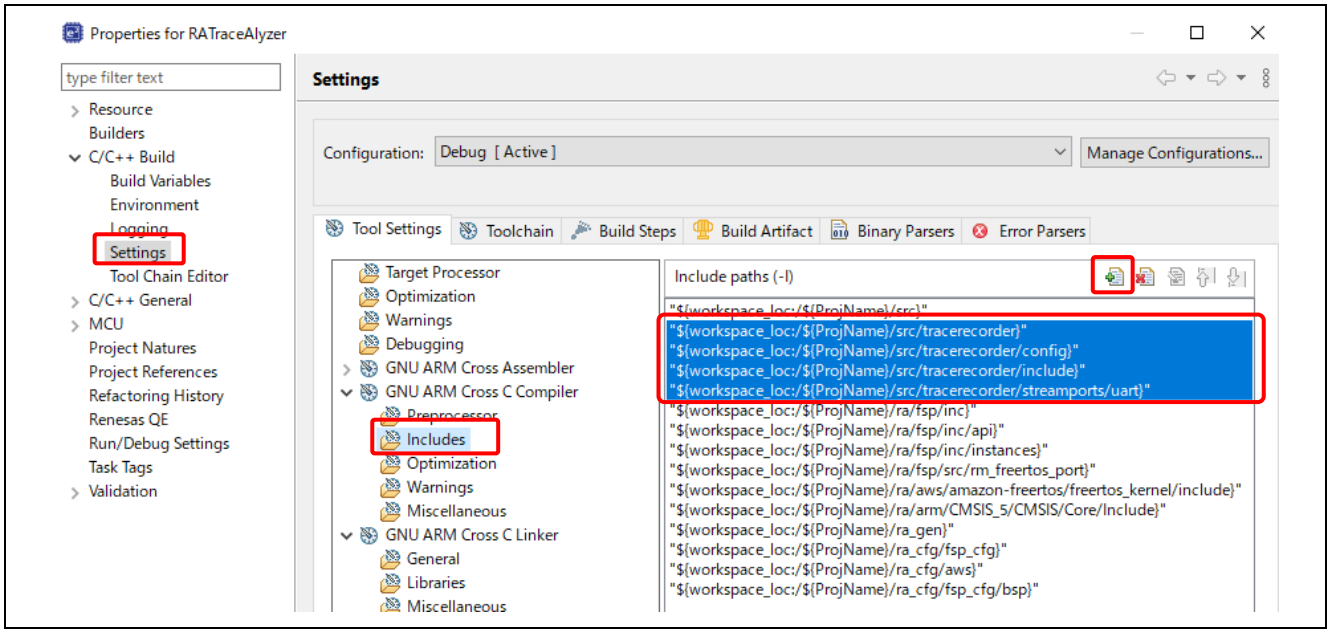



**Figure 23. Setting Include paths in project properties**

### 4.3.6 Add code to hal_entry.c

Add source code in red to `hal_entry.c`.

```
#include "bsp_api.h"
#include "trcRecorder.h"
#include "FreeRTOS.h"
#include "semphr.h"
#include "hal_data.h"

SemaphoreHandle_t semaphore_handle_1;
StaticSemaphore_t semaphore_handle_1_memory;
void R_BSP_WarmStart(bsp_warm_start_event_t event);

/*******************************************************************************************
********************//**
 * This function is called at various points during the startup process.  This implementation
uses the event that is
 * called right before main() to set up the pins.
 *
 * @param[in]  event    Where at in the start up process the code is currently at

*******************************************************************************************
**********************/
void R_BSP_WarmStart (bsp_warm_start_event_t event)
{
    if (BSP_WARM_START_RESET == event)
    {
#if BSP_FEATURE_FLASH_LP_VERSION != 0

        /* Enable reading from data flash. */
        R_FACI_LP->DFLCTL = 1U;

        /* Would normally have to wait tDSTOP(6us) for data flash recovery. Placing the enable
here, before clock and
         * C runtime initialization, should negate the need for a delay since the initialization
will typically take more than 6us. */
#endif
    }

    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);
    }

    fsp_err_t err = FSP_SUCCESS;

    /* Initialize UART channel with baud rate 115200 */
    err = R_SCI_UART_Open (&g_uart0_ctrl, &g_uart0_cfg);
    if (FSP_SUCCESS != err)
    {
    }
    semaphore_handle_1 = xSemaphoreCreateBinaryStatic (&semaphore_handle_1_memory);
    vTraceEnable(TRC_INIT);
}
```

**Figure 24. Change the UART Streaming hal_entry.c**

### 4.3.7 Build the project

Right-click on the project and select **Build Project**.

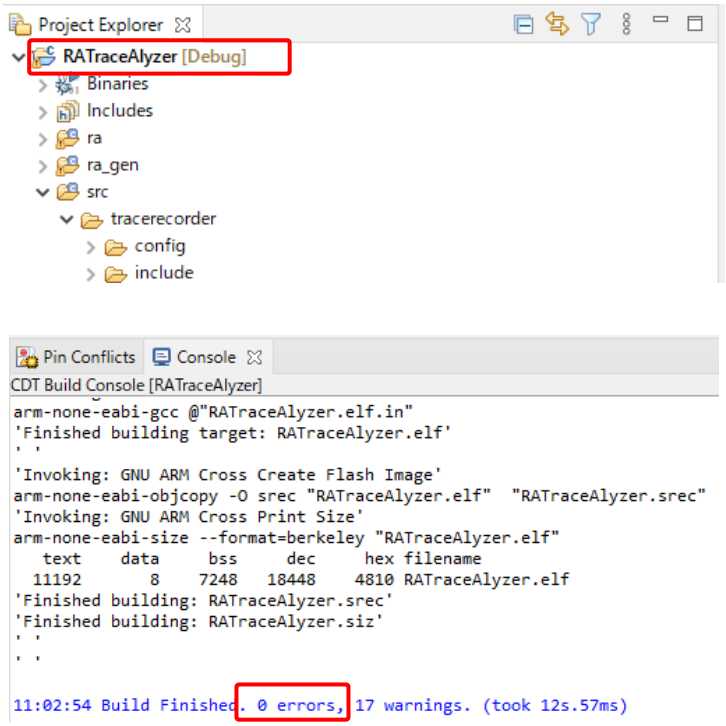



**Figure 25. Build Project**

## 4.4 Connect PC and EK-RA6M3 Board

The figure below shows the connection between the host PC and the EK-RA6M3 board.

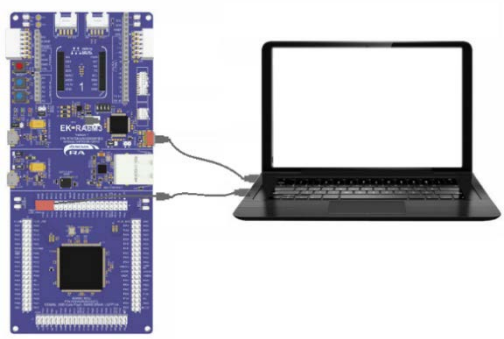


**Figure 26. EK-RA6M3 Board Connection**

The hardware settings are as follows:

**Table 1. Jumper Connection Summary for Different Debug Modes**

| Debug Modes | J8 | J9 | J29 |
|---|---|---|---|
| Debug on-board | Jumper on pins 1-2 | Open | Jumpers on pins 1-2, 3-4, 5-6, 7-8 |

Terminal connection (UART):

- PC port TXD: ORANGE - EK-RA6M3 board P410 (RXD_MISO)
- PC port RXD: YELLOW - EK-RA6M3 board P411 (TXD_MOSI)
- PC port GND: BLACK - EK-RA6M3 board GND

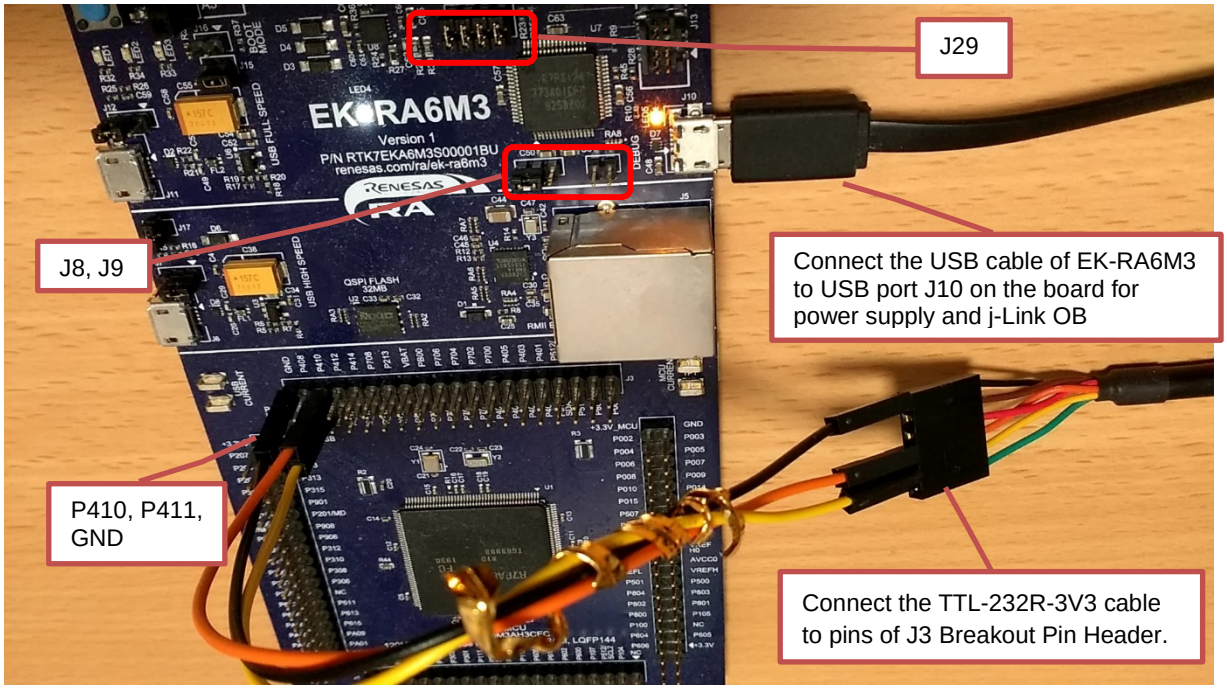



**Figure 27. Connection between PC and EK-RA6M3 Board**

## 4.5 Using the RTOS Resource View

The e² studio has an RTOS resource view function that displays the state of FreeRTOS resources. This procedure describes how to use the RTOS resource view.

### 4.5.1 Displaying the RTOS Resources View

Because the **RTOS Resources** view functions only with the debugger running, start the debugger and select **Renesas Views > Partner OS > RTOS Resources**. When the **Select OS** dialog box is displayed, select FreeRTOS as shown in Figure 28. The **RTOS Resources** view appears view appears as shown in Figure 29.

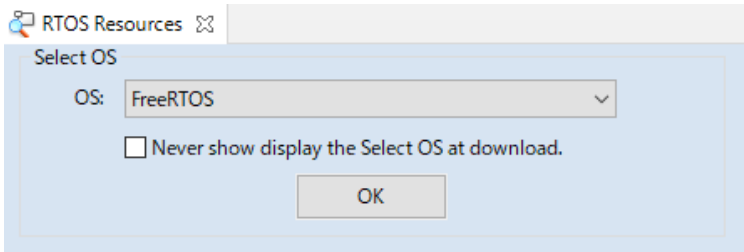



**Figure 28. Selecting the OS**

RTOS Resources

Stack | Task | Queue | Timer

| No. | TaskName | Base/ActualPriority | State | EventObject | TotalTickCount | DeltaTickCount |
|---|---|---|---|---|---|---|
| 1 | Blinky Thread | 1/1 | BLOCKED | None | -(-%) | -(-%) |
| 2 | IDLE | 0/0 | READY | None | -(-%) | -(-%) |
| 3 | Tmr Svc | 3/3 | SUSPENDED | None | -(-%) | -(-%) |
| 4 | | | | | | |
| 5 | | | | | | |

**Figure 29. RTOS Resources view**

### 4.5.2 Context menu

Display the context menu by right-clicking the mouse on the RTOS Resources view.

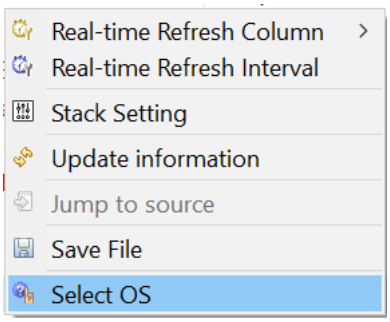



**Figure 30. Context menu**

Explanation:

- **Real-time Refresh Column:**
  Allows real-time display for the displayed items.
  This is not valid while the program is running.
- **Real-time Refresh Interval:**
  Specifies interval time for updating of the real-time display. The specifiable range is 500ms to 10000ms.
  This is not valid while the program is running.
- **Stack Setting:**
  Enables/disables Stack Loading and stack threshold setting for stack alert function.
  This is not valid while a program is running.
- **Update information:**
  Updates the information.
- **Jump to source:**
  Opens an editor view displaying the source code of the task/thread or handler. Double-clicking the task/thread or handler also opens an editor view.
  This is not valid while the program is running.
- **Save File:**
  Saves the data of the current tab in the text file (`*.txt`).
  This is not valid while the program is running.
- **Select OS:**
  Opens the **Select OS** dialog box.
  This is not valid while the program is running.

### 4.5.3 Stack setting

Enable load stack data and set stack threshold.

1. Open the context menu and select **Stack Setting**.
2. To load stack data to the **RTOS Resource** view, check **Enable loading Stack data** checkbox in the **Stack Setting** dialog. If this option is not enabled, stack data will not be loaded in the next debugging session.

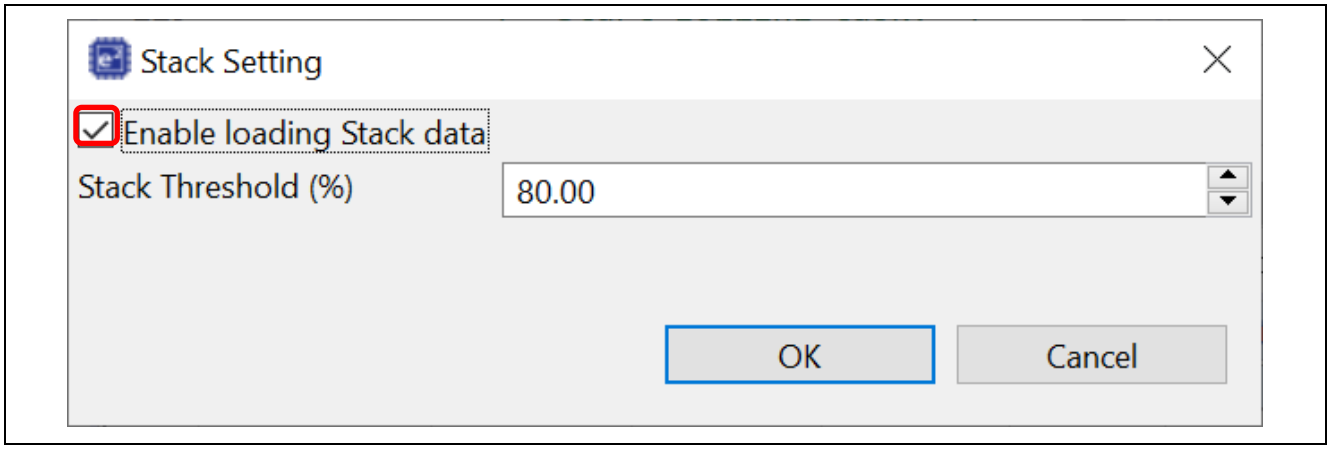



**Figure 31. Enable loading stack data**

3. The desired threshold value can be set in the **Stack Threshold (%)** textbox. Click **OK** to save the setting.

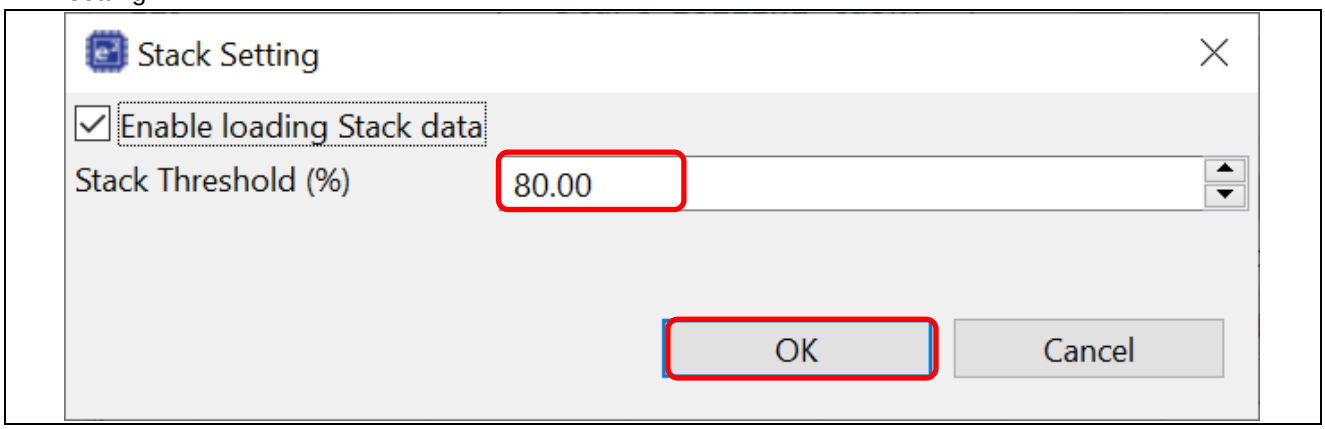



**Figure 32. Set up threshold value**

4. Run then suspend the target project to load stack data. The stack threshold warning will pop up if the threshold set is met.

There are 2 types of warning popup: **Stack Threshold Warning** (list of threads which reached stack threshold value set as above) and **Stack Overflow Warning** (reached 100%).

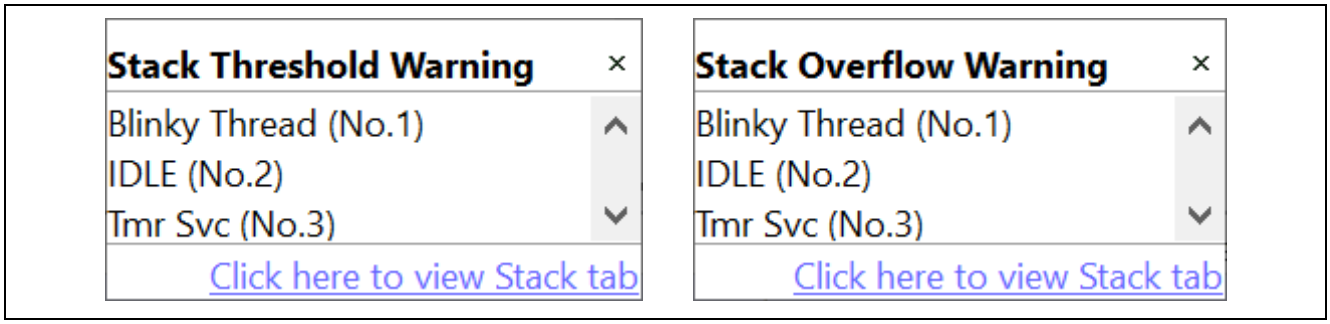



**Figure 33. Stack Threshold Warning popup (left) and Stack Overflow Warning popup (right)**

### 4.5.4 Tab menu

Table 2 show display items for each tab.

**Table 2. Contents of each tabbed window**

| Name of tabbed window in the RTOS Resources view | Displayed information andselections | Information to be displayed |
|---|---|---|
| Stack | No. | Row index |
| | TaskName | The name assigned to the task upon creation |
| | StartOfStack | The address of the start of stack |
| | EndOfStack | The address of the end of stack |
| | TopOfStack | The address of the top of the stack where it is last written to when the context of the stack was saved |
| | StackSize(bytes) | Total stack size |
| | StackUsageSize | Stack usage at high water mark |
| | StackUsageRatio | Percentage of usage at high water mark relative to total stack size |
| Task | No. | Row index |
| | TaskName | The name assigned to the task upon creation |
| | Base/ActualPriority | The base priority used by the priority inheritance mechanism/The actual priority used by the task |
| | State | State of the task which includes “RUNNING”, “READY”, “BLOCKED” and “SUSPENDED” |
| | EventObject | The name of the queue which causes the task to be blocked |
| | TotalTickCount | The total number of tick count for the task to be active |
| | DeltaTickCount | The number of tick count for the task to be active since previous suspend event |
| Queue | No. | Row index |
| | Name (Type) | The name assigned to the queue upon registration and its type (Queue, Semaphore, or Mutex) |
| | Address | The address of the queue handle |
| | MaxLength | Size per item in the queue (in bytes) |
| | ItemSize | Message size |
| | CurrentLength | Number of items currently stored in the queue |
| | #WaitingTx | Number of tasks blocked while waiting to send to the queue |
| | #WaitingRx | Address where the message queue starts |
| | EndAddress | Address where the message queue ends |
| Timer | No. | Row index |
| | Name | The current period of the timer in system ticks |
| | Period | Automatic reload enable/disable. “On” when auto reload is enabled, which resets the timer each time it expires, “Off” when auto reload is disabled which does nothing when the timer expires |
| | CallbackFn | Address and <Name> of the callback function which executes each time the timer ends |
| | TimerID | The numeric ID of the timer assigned in hexadecimal format when it was created |

## 4.6 Start debugging a project with Tracealyzer®

### 4.6.1 Launch debugger on e² studio

Select menu **Run > Debug** to launch the debugger.

### 4.6.2 Launch Tracealyzer®

Launch installed Tracealyer 4 on PC.

### 4.6.3 Set Recording Settings

Click **Recording settings** on Tracealyzer and select **PSF Streaming Settings** on Tracealyzer. Set the following.

- Device: **COM8** (user pc system port)
- Data bits: **8**
- Data rate: **921600**
- Handshake: **None**
- Parity: **None**
- Stop bits: **One**

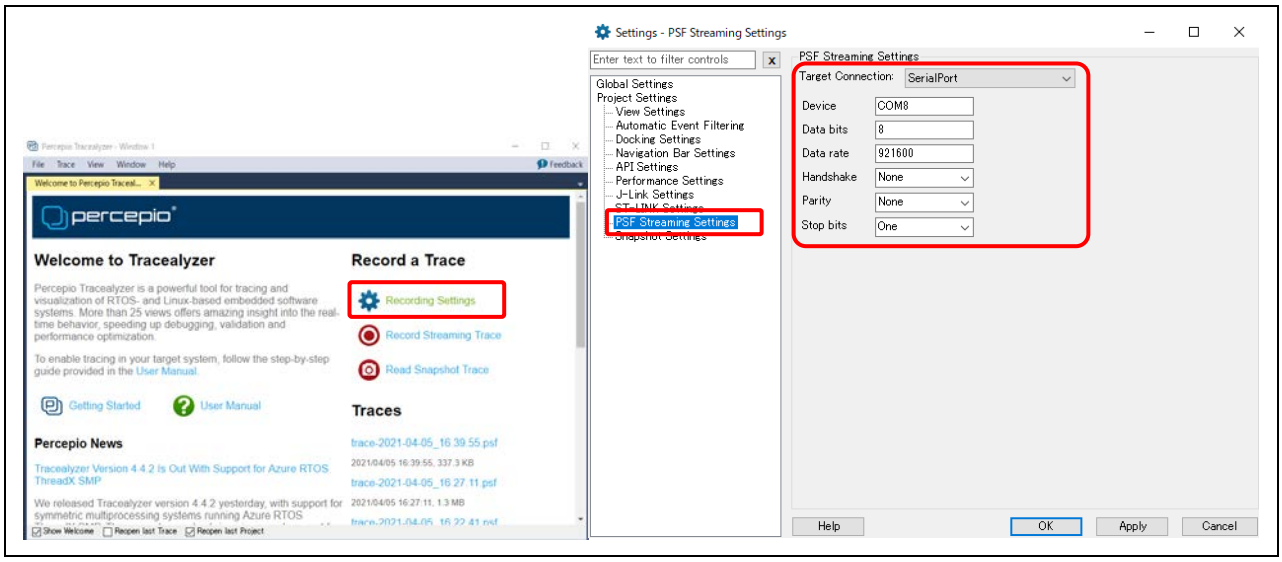



**Figure 34. Set to Record UART Settings**

### 4.6.4 Start Recording a Trace

Click **Record Streaming Trace** to start recording a Trace.

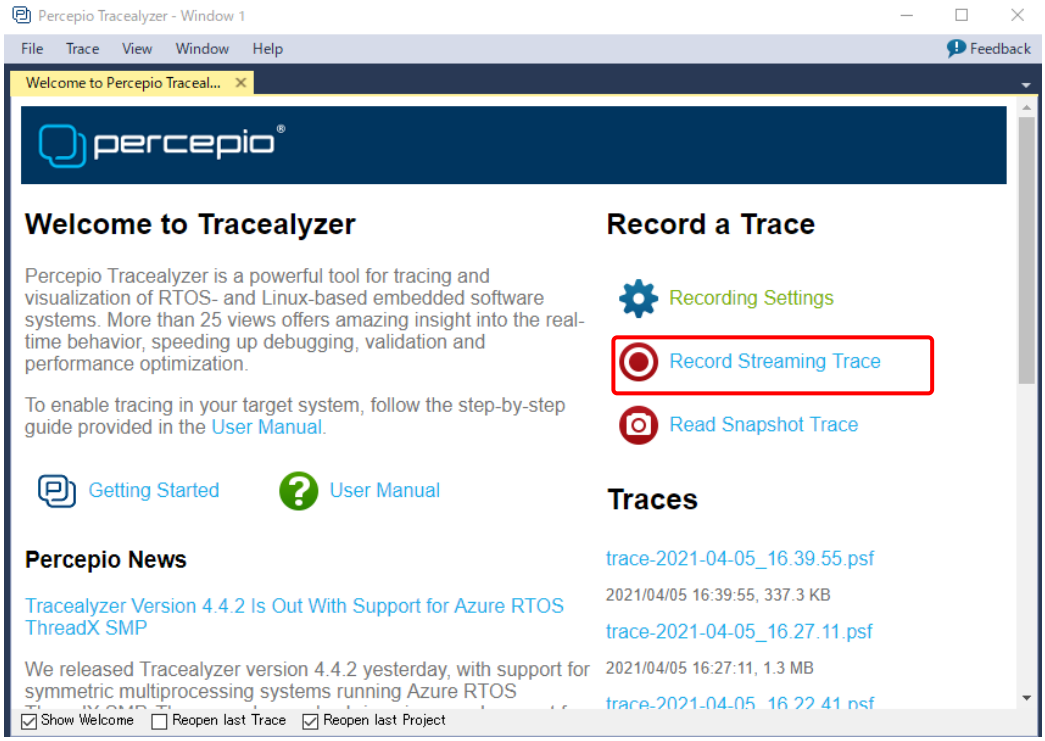



**Figure 35. Start Record Streaming Trace**

### 4.6.5 Trace information is displayed

Various analysis modes are provided. For more information, see **Help** tab.

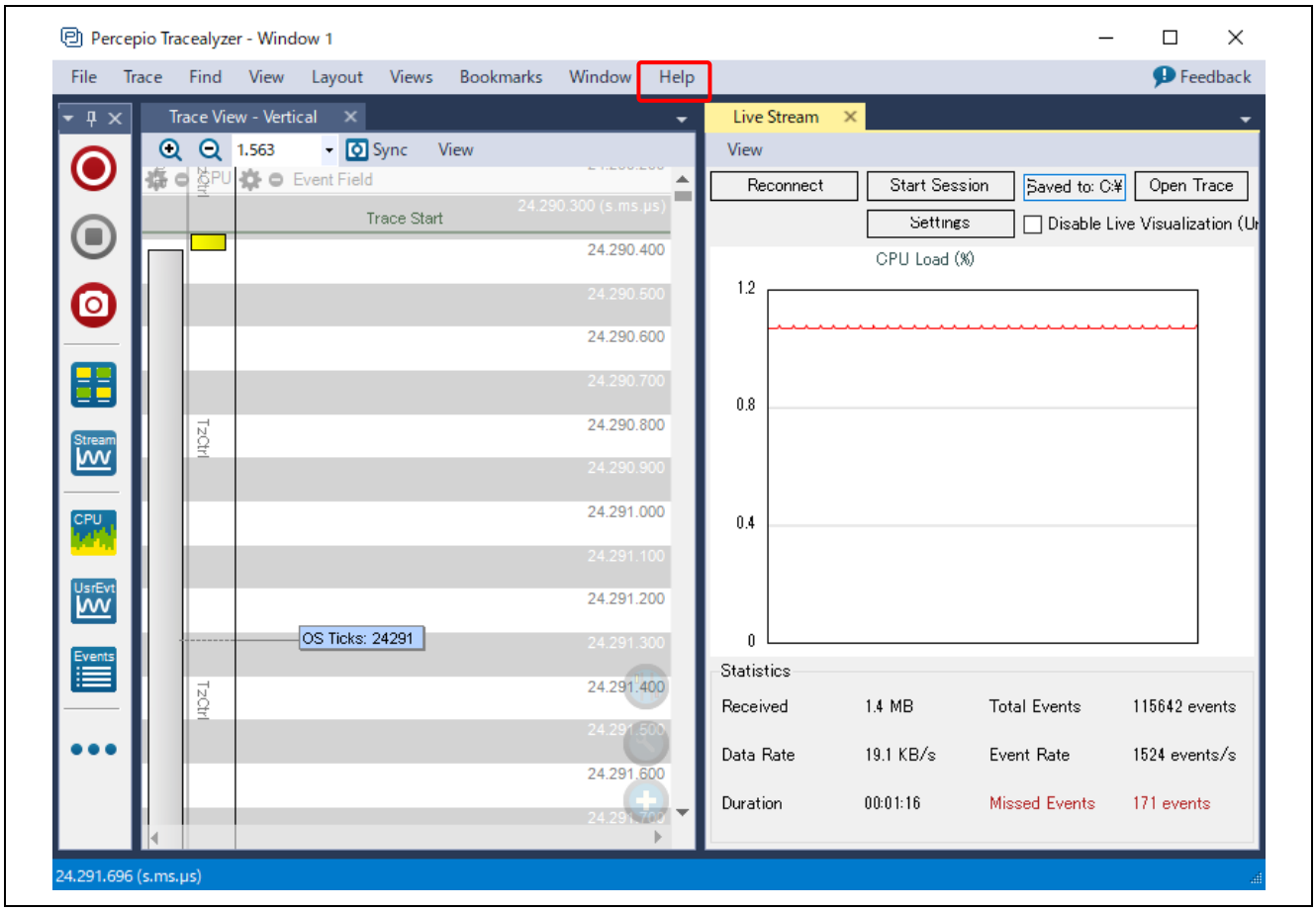



**Figure 36. Displayed Trace information**

# 5. Debugging via J-Link RTT with Tracealyzer®

This section describes how to use Tracealyzer® in J-Link RTT.

## 5.1 Copy and Remove Tracealyzer® for FreeRTOS into a project

### 5.1.1 Copy Tracealyzer® for FreeRTOS source under the Tracealyzer® installation folder

Copy the `Program Files\Percecio\Tracealyzer 4\FreeRTOS\TraceRecorder` folder into workspace folder `src` using File Explorer.

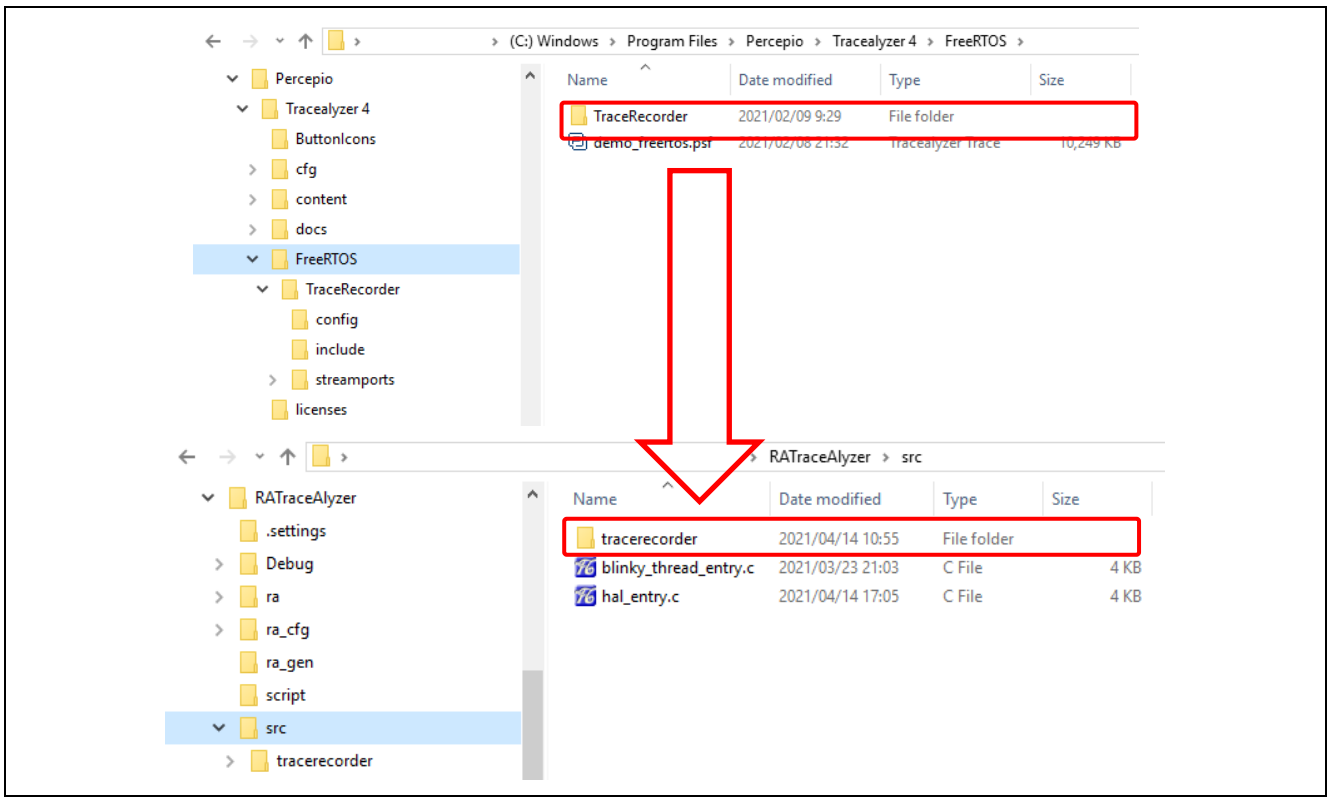



**Figure 37. Copy Folder**

### 5.1.2 Remove unnecessary folders

1. Remove sub-folders marked in red in workspace folder `src/TraceRecorder/streamports` as shown in Figure 5-2.

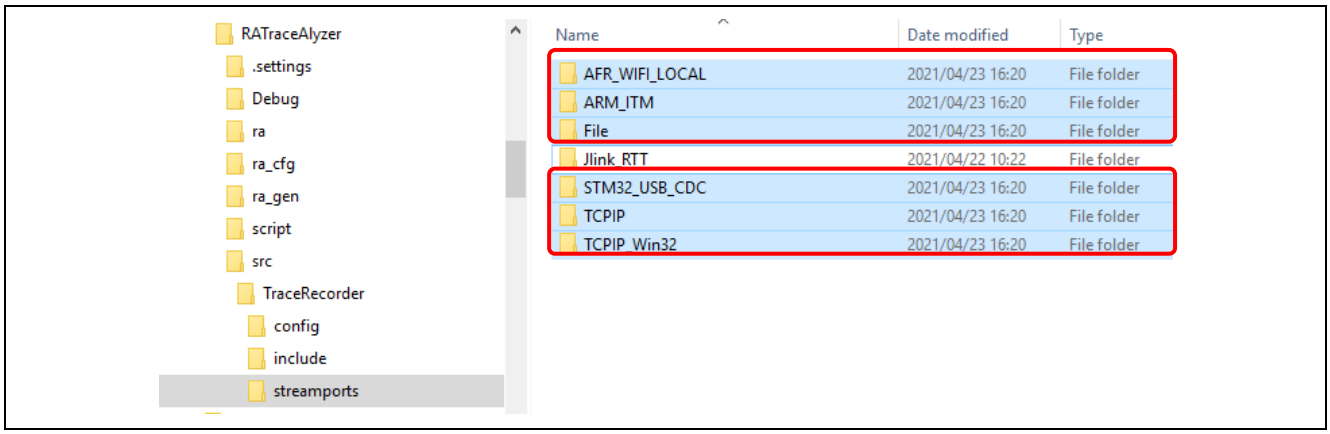



**Figure 38. Remove Folder (J-Link RTT)**

2. Copy J-Link RTT files in EK-RA6M3 Example Project Bundle - Sample Code.
   Overwrite the files in the `\ek_ra6m3\sci_uart\sci_uart_ek_ra6m3_ep\e2studio\src\SEGGER_RTT` folder into workspace folder `src/TraceRecorder/streamports`" using File Explorer.
   - File `SEGGER_RTTc`
   - File `SEGGER_RTT_printfc`
   - File `SEGGER_RTTh`
   - File `SEGGER_RTT_Confh`

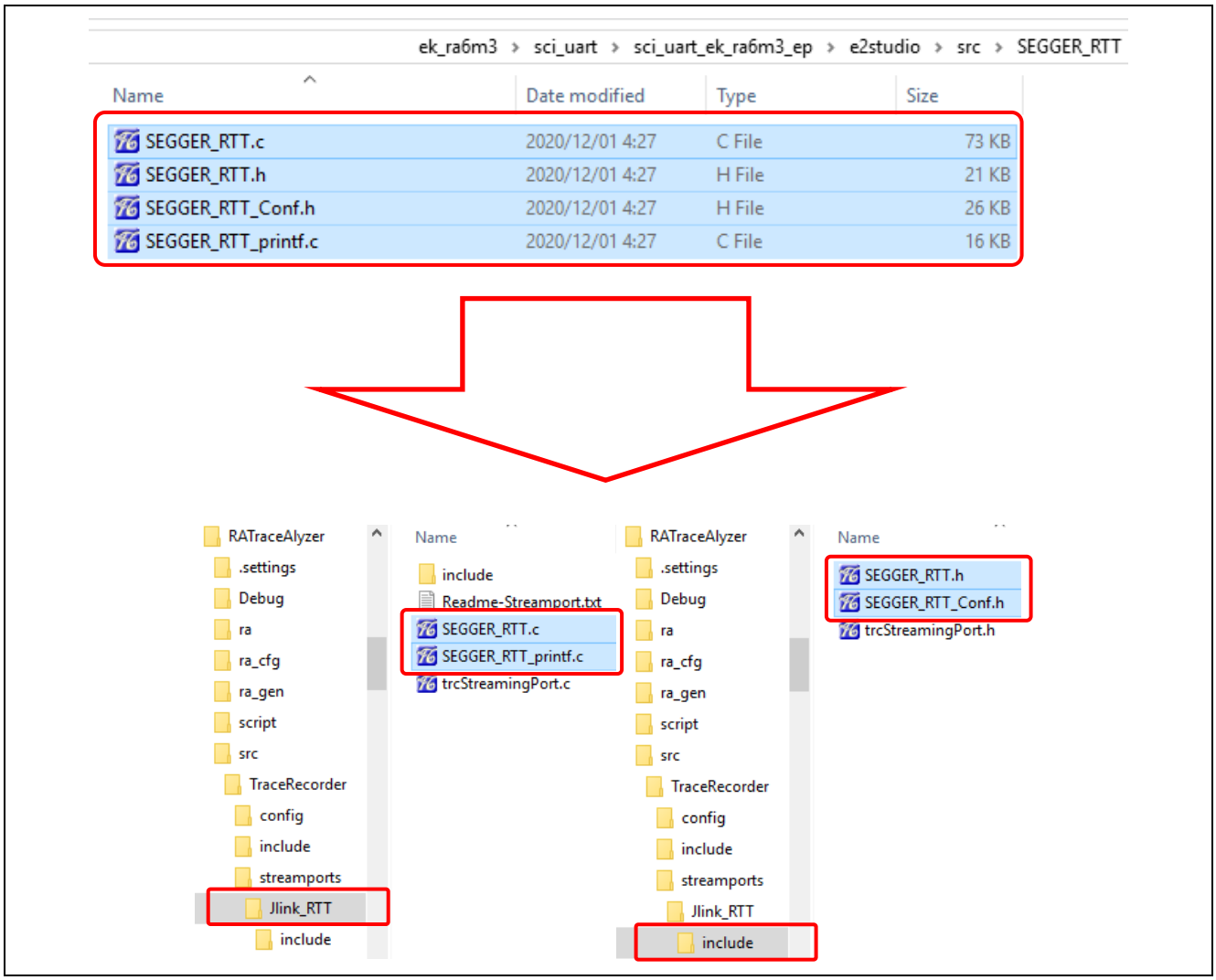



**Figure 39. Copy the SEGGER_RTT files**

## 5.2 FSP Configuration

### 5.2.1 Blinky Thread Settings

Add the Heap to the Blinky Thread as follows.

- Move to the **Stacks** tab.
- Select **Blinky Thread**.
- Click **New Stack > FreeRTOS > Memory Management > Heap 1**.

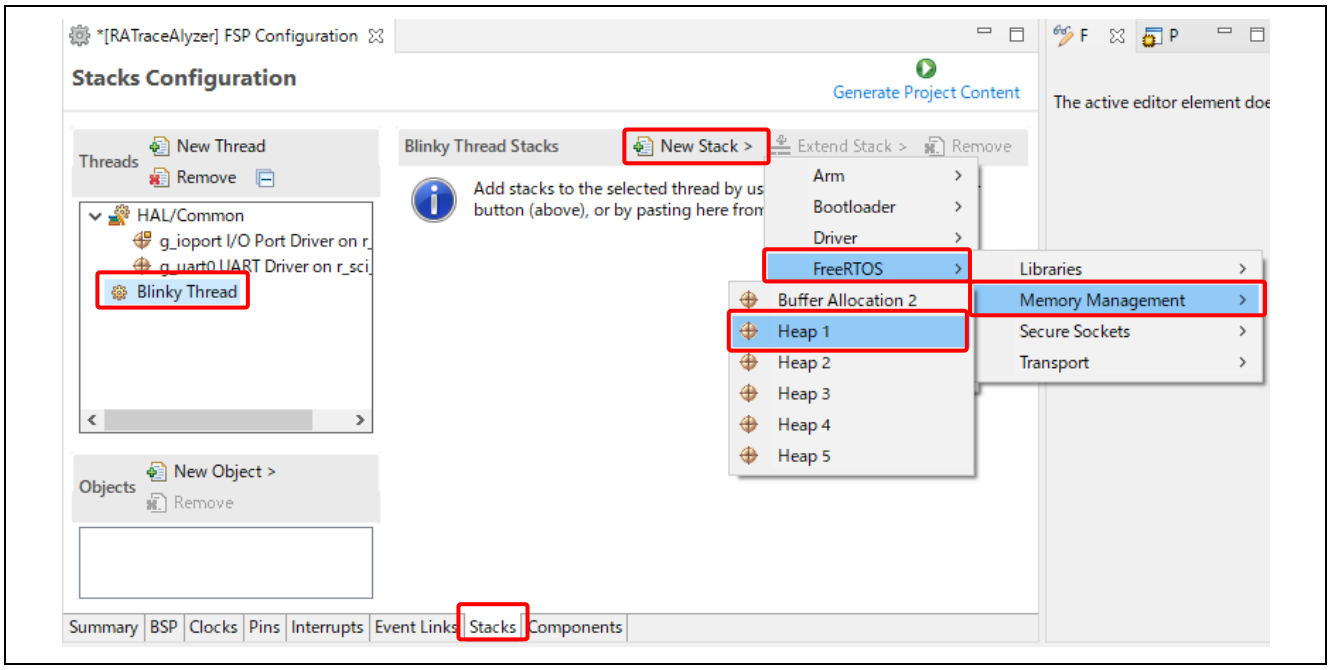



**Figure 40. Add Heap on Stacks Configuration**

Change the **Properties > General** on **Blinky Thread** as follows.

- Minimal Stack Size: **512**
- Use Mutexes: **Enabled**
- Use Recursive Mutexes: **Enabled**
- Use Queue Sets: **Enabled**
- Enable Backward Compatibility: **Enabled**

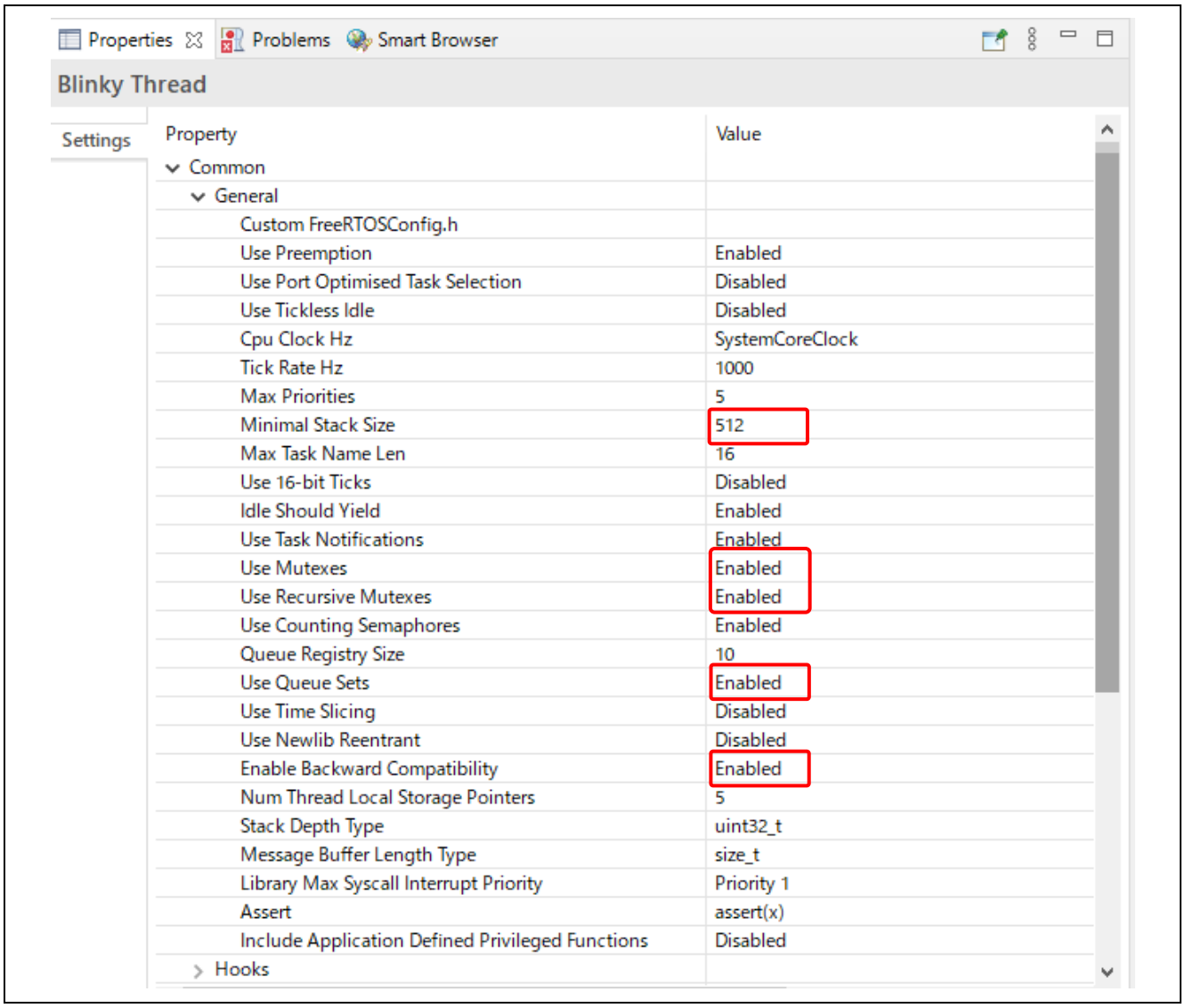



**Figure 41. Blinky Thread Properties 1**

Change the **Properties > Hooks**, **Stats**, **Memory Allocation**, **Timers** on **Blinky Thread** as follows.

- Use Idle Hook: Disabled
- Use Malloc Failed Hook: Enabled
- Use Trace Facility: Enabled
- Use Stats Formatting Functions: Enabled
- Support Dynamic Allocation: Enabled
- Total Heap Size: 262,144 (256 * 1,024)
- Timer Task Static Depth: 3,072 (1024 * 3)

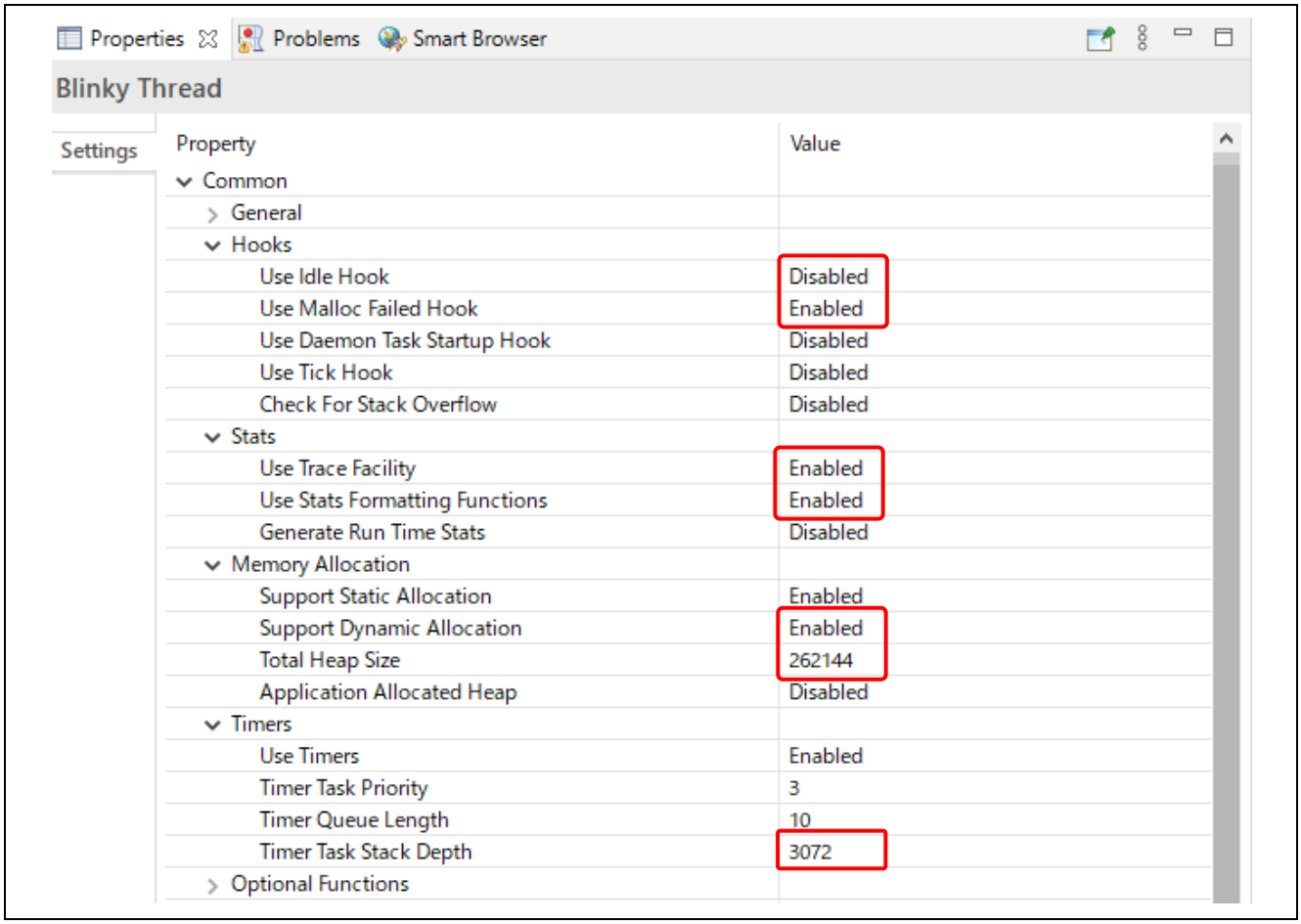



**Figure 42. Blinky Thread Properties 2**

Change the **Properties > Optional Functions**, **RA**, **Logging** on **Blinky Thread** as follows.

- uxTaskGetStackHighWaterMark() Function: **Enabled**
- eTaskGetState() Function: **Enabled**
- xTimerPendFunctionCall() Function: **Enabled**
- xTaskAbortDelay() Function: **Enabled**
- Hardware Stack Monitor: **Enabled**
- Logging Include Time and Task Name: **Enabled**

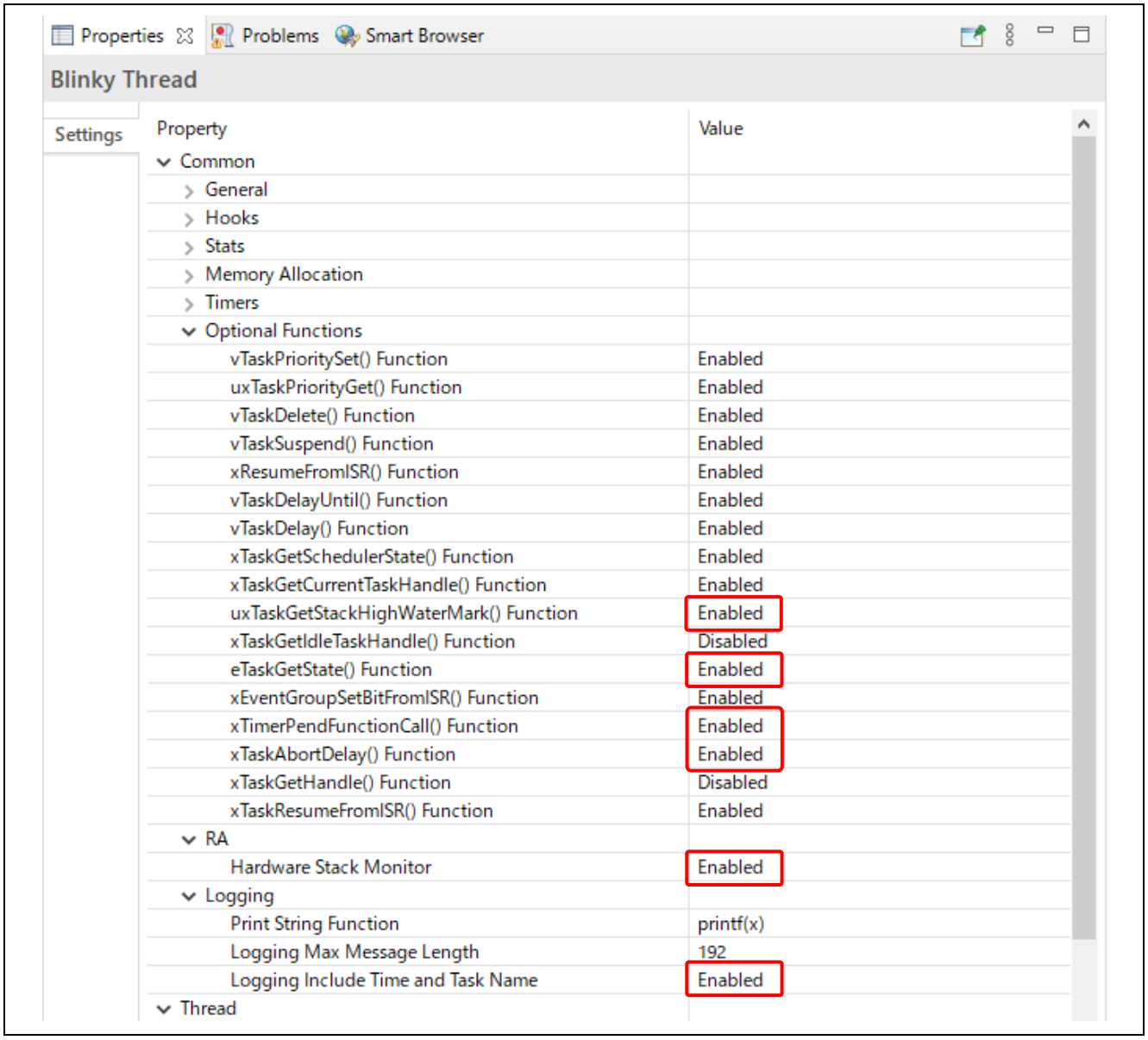
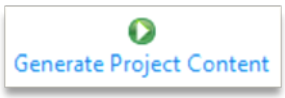



**Figure 43. Blinky Thread Properties 3**

### 5.2.2 Generate Project Content

Click on the 

button to generate the source files.

## 5.3 Code editing for Tracealyzer® connections

### 5.3.1 Add include file to task.c and timers.c

- `#include "trcRecorder.h"`

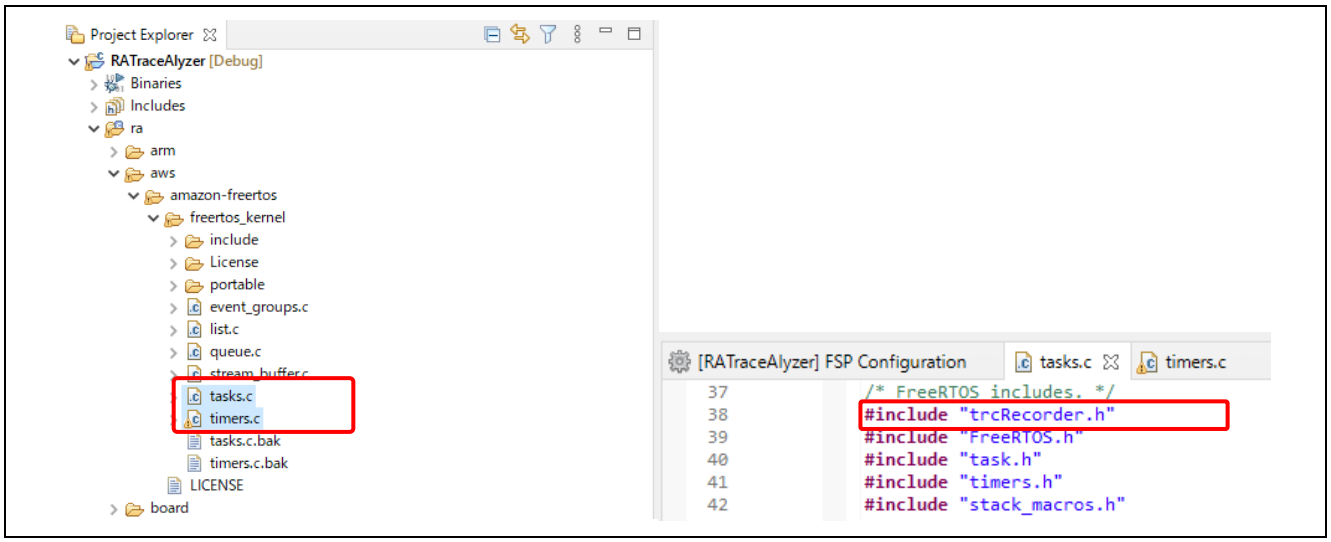



**Figure 44. Add include to freertos_kernel**

### 5.3.2 Add include files to trcKernelPort.c and trcStreamingRecorder.c

- `#include "bsp_api.h"`
- `#include "trcRecorder.h"`

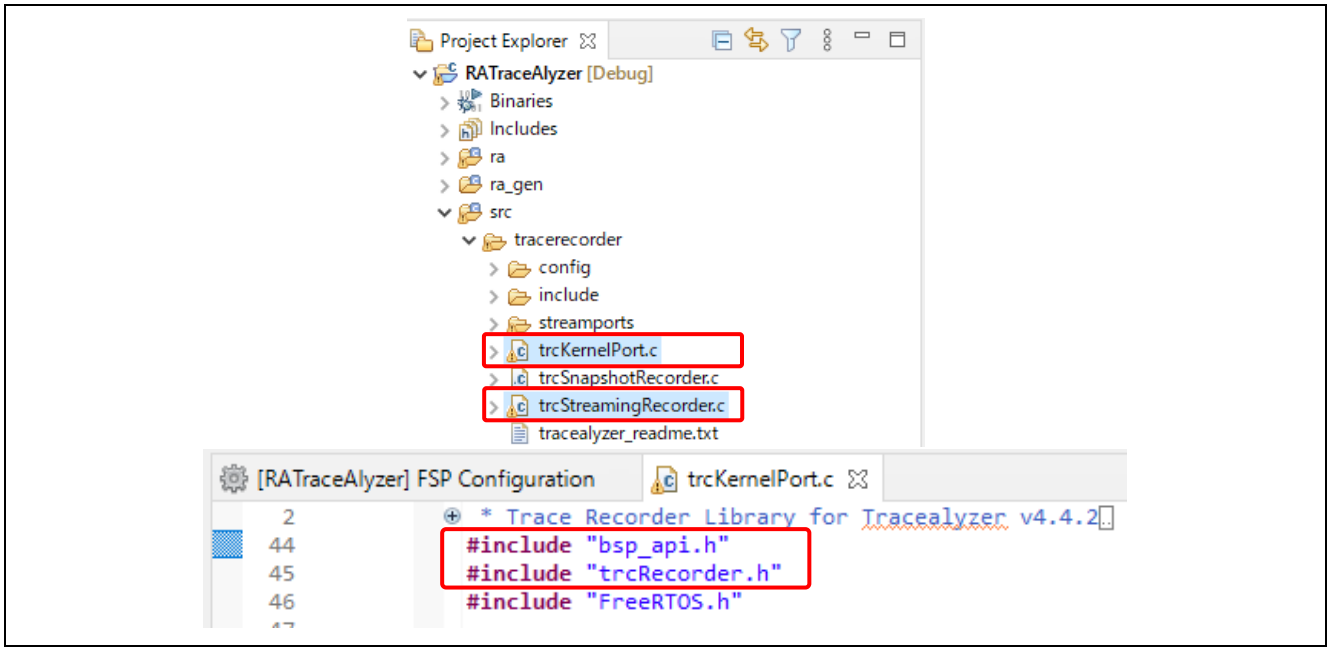



**Figure 45. Add include tracerecorder**

### 5.3.3 Change macro definitions in trcConfig.h

Change only the red part of macro definitions `trcConfig.h` as follows.

- #include “bsp_api.h”.
- //#error "Trace Recorder: Please include your processor's header file here and remove this line.".
- #define TRC_CFG_HARDWARE_PORT TRC_HARDWARE_PORT_ARM_Cortex_M.
- #define TRC_CFG_RECORDER_MODE TRC_RECORDER_MODE_STREAMING.
- #define TRC_CFG_FREERTOS_VERSION TRC_FREERTOS_VERSION_10_4_1.

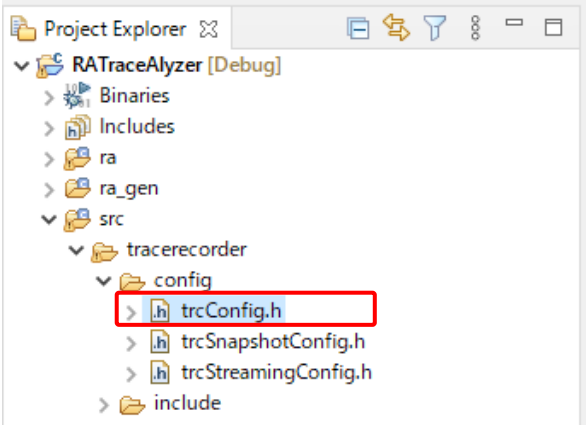



**Figure 46. Change the define in trcConfig.h**

### 5.3.4 Add Include path on e² studio properties

Select menu **Project > Properties**, then click **Settings > Includes** to add the Include path .

- `"${workspace_loc:/${ProjName}/src/tracerecorder}"`
- `"${workspace_loc:/${ProjName}/src/tracerecorder/config}"`
- `"${workspace_loc:/${ProjName}/src/tracerecorder/include}"`
- `"${workspace_loc:/${ProjName}/src/tracerecorder/streamports/Jlink_RTT/include}"`

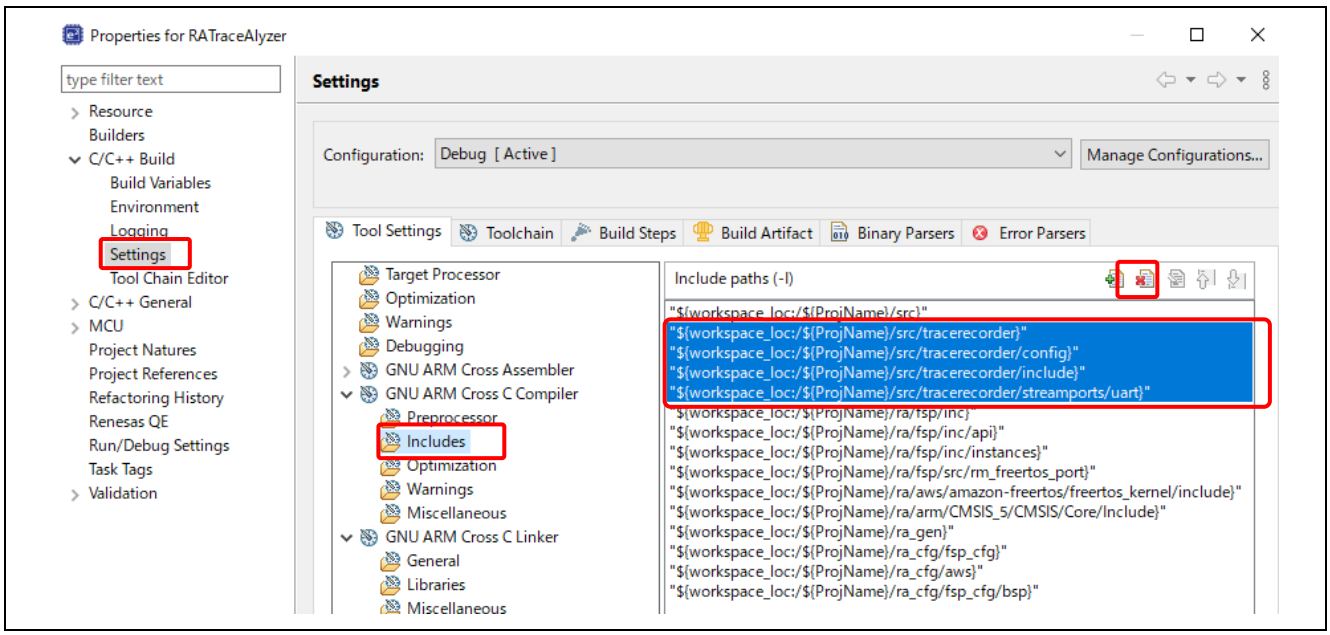



**Figure 47. Settings Include paths of project properties**

### 5.3.5 Add code to hal_entry.c

Add source code in red to `hal_entry.c`.

```
#include "bsp_api.h"
#include "trcRecorder.h"
#include "FreeRTOS.h"
#include "semphr.h"
#include "hal_data.h"

void R_BSP_WarmStart(bsp_warm_start_event_t event);

/*******************************************************************************************
********************//**
 * This function is called at various points during the startup process.  This implementation
uses the event that is
 * called right before main() to set up the pins.
 *
 * @param[in]  event    Where at in the start up process the code is currently at

*******************************************************************************************
**********************/
void R_BSP_WarmStart (bsp_warm_start_event_t event)
{
    if (BSP_WARM_START_RESET == event)
    {
#if BSP_FEATURE_FLASH_LP_VERSION != 0

        /* Enable reading from data flash. */
        R_FACI_LP->DFLCTL = 1U;

        /* Would normally have to wait tDSTOP(6us) for data flash recovery. Placing the enable
here, before clock and
         * C runtime initialization, should negate the need for a delay since the initialization
will typically take more than 6us. */
#endif
    }

    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);
    }

    vTraceEnable(TRC_INIT);
}
```

**Figure 48. Change the J-Link RTT Streaming hal_entry.c**

### 5.3.6 Build the project

Right-click on the project and select **Build Project**.

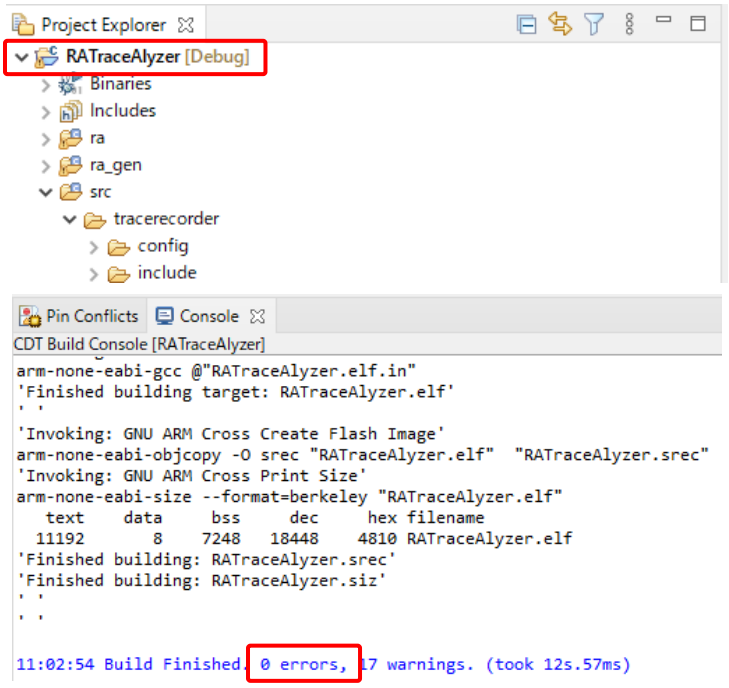



**Figure 49 Build Project**

## 5.4 Connect PC and EK-RA6M3 Board

The picture below shows the connection between the host PC and the EK-RA6M3 board.

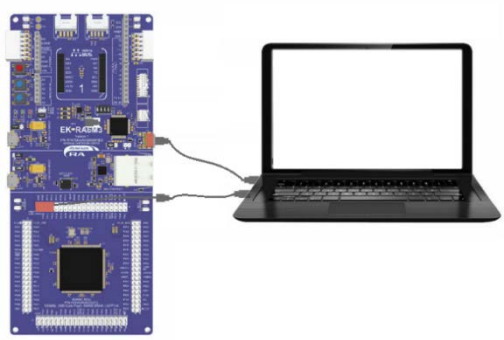


**Figure 50. EK-RA6M3 Board Connection**

The hardware settings are as follows:

**Table 3. Jumper Connection Summary for Different Debug Modes**

| Debug Modes | J8 | J9 | J29 |
|---|---|---|---|
| Debug on-board | Jumper on pins 1-2 | Open | Jumpers on pins 1-2, 3-4, 5-6, 7-8 |

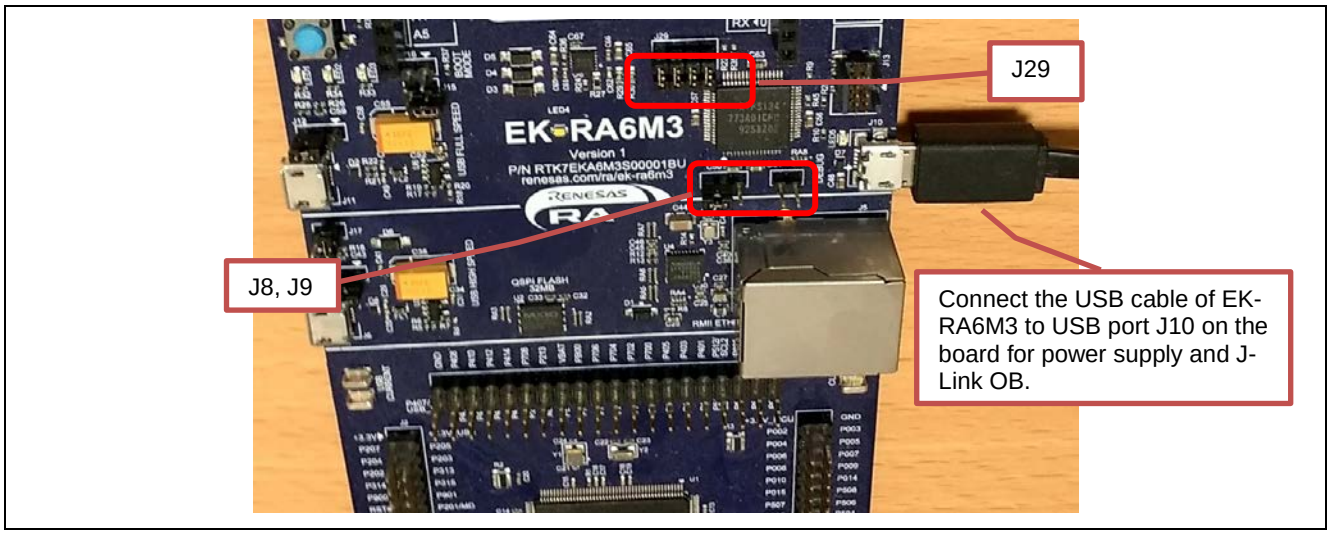



**Figure 51. Connection between PC and EK-RA6M3 Board**

## 5.5 Using the RTOS Resource View

Refer to section 4.5, Using the RTOS Resource View.

## 5.6 Start debugging a project with Tracealyzer®

### 5.6.1 Launch debugger on e² studio

Select menu **Run > Debug** to launch the debugger.

### 5.6.2 Launch Tracealyzer®

Launch installed Tracealyer 4 on PC.

### 5.6.3 Set to Recording Settings

Refer to Figure 52 to get the RTT control block address in the map file, See code in red.

```
*(COMMON)
 COMMON          0x1ffe2fa4
0xa8 ./src/TraceRecorder/streamports/Jlink_RTT/SEGGER_RTT.o
                                0x1ffe2fa4                _SEGGER_RTT
```

**Figure 52. RATraceAlyzer.map (map file)**

Click **Recording settings** on Tracealyzer, then select **J-Link Settings** and **PSF Streaming Settings** on Tracealyzer. Set the following.

- Select Target Device: **R7FA6M3AH**. (EK-RA6M3)
- Set RTT Control Block Address: **0x1FFE2FA4**.See **Error! Reference source not found.**

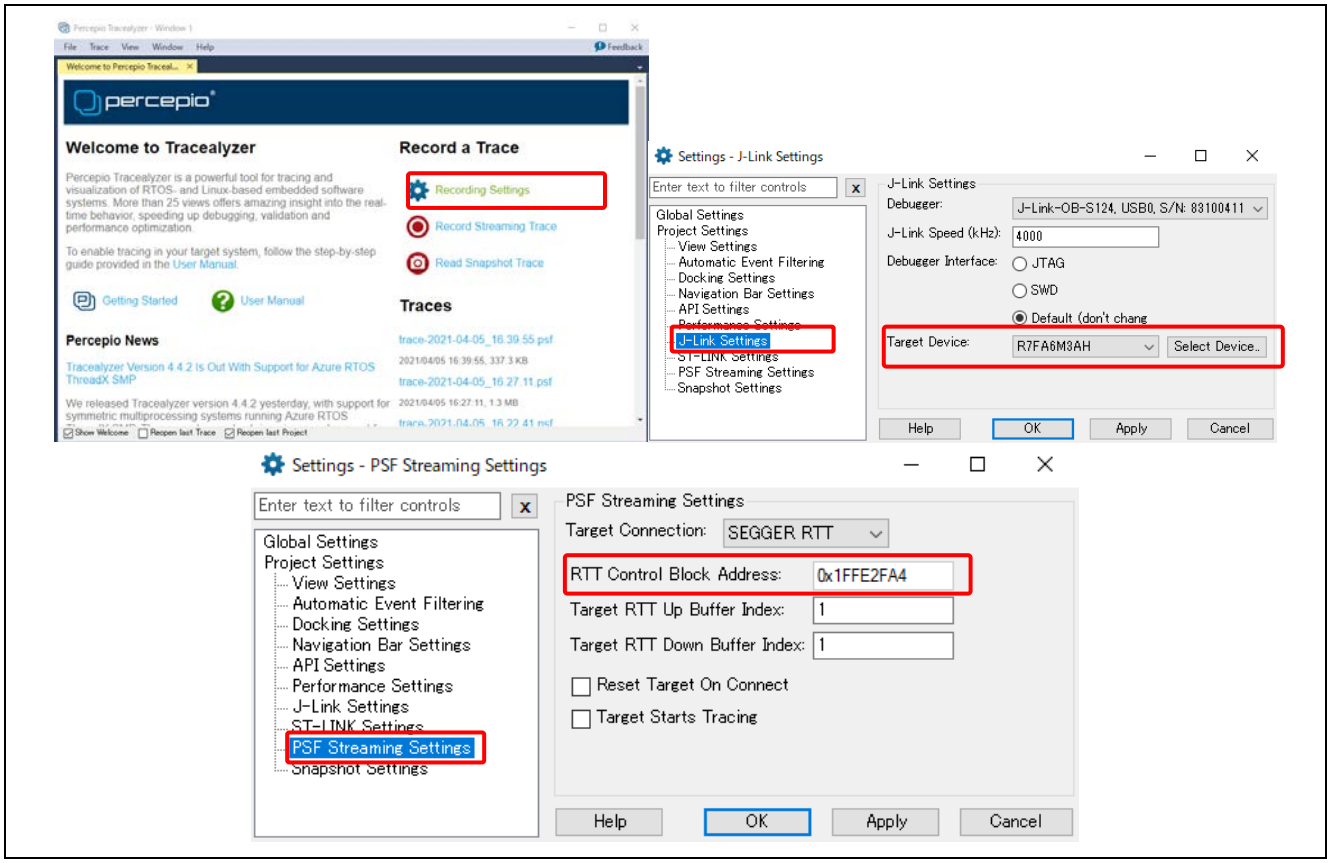



**Figure 53. Set to Recording J-Link RTT Settings**

### 5.6.4 Start Recording a Trace

Click Record Streaming Trace to start recording a trace.

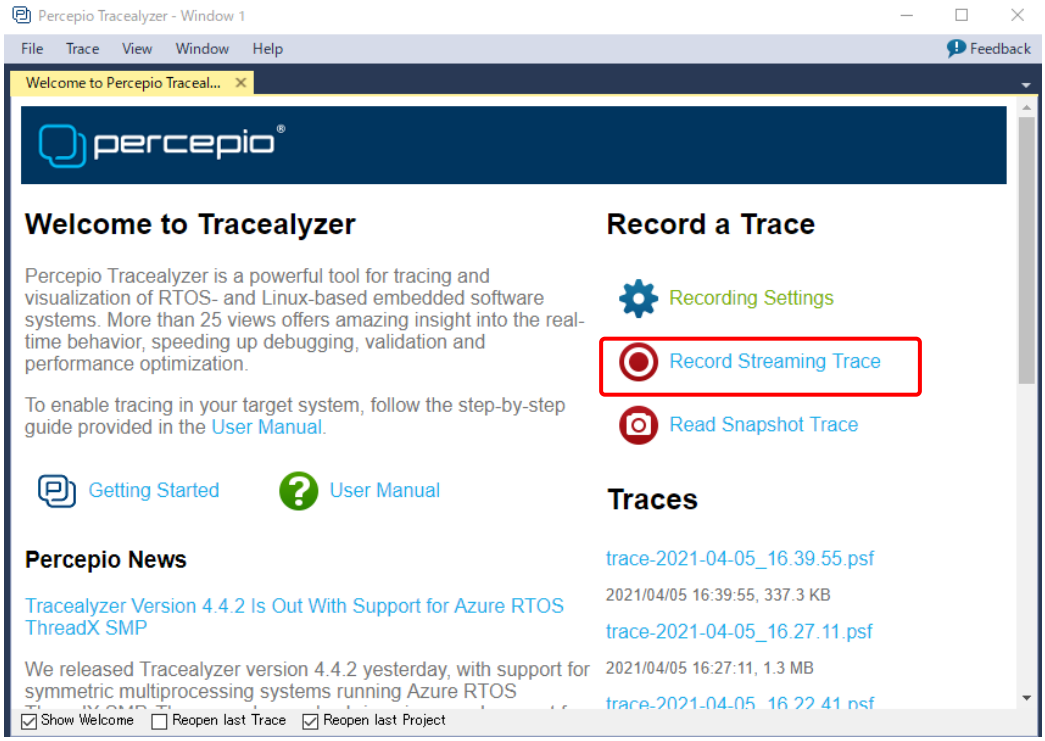



**Figure 54. Start Record Streaming Trace**

### 5.6.5 Trace information is displayed

Various analysis modes are provided. For more information, see the **Help** tab.

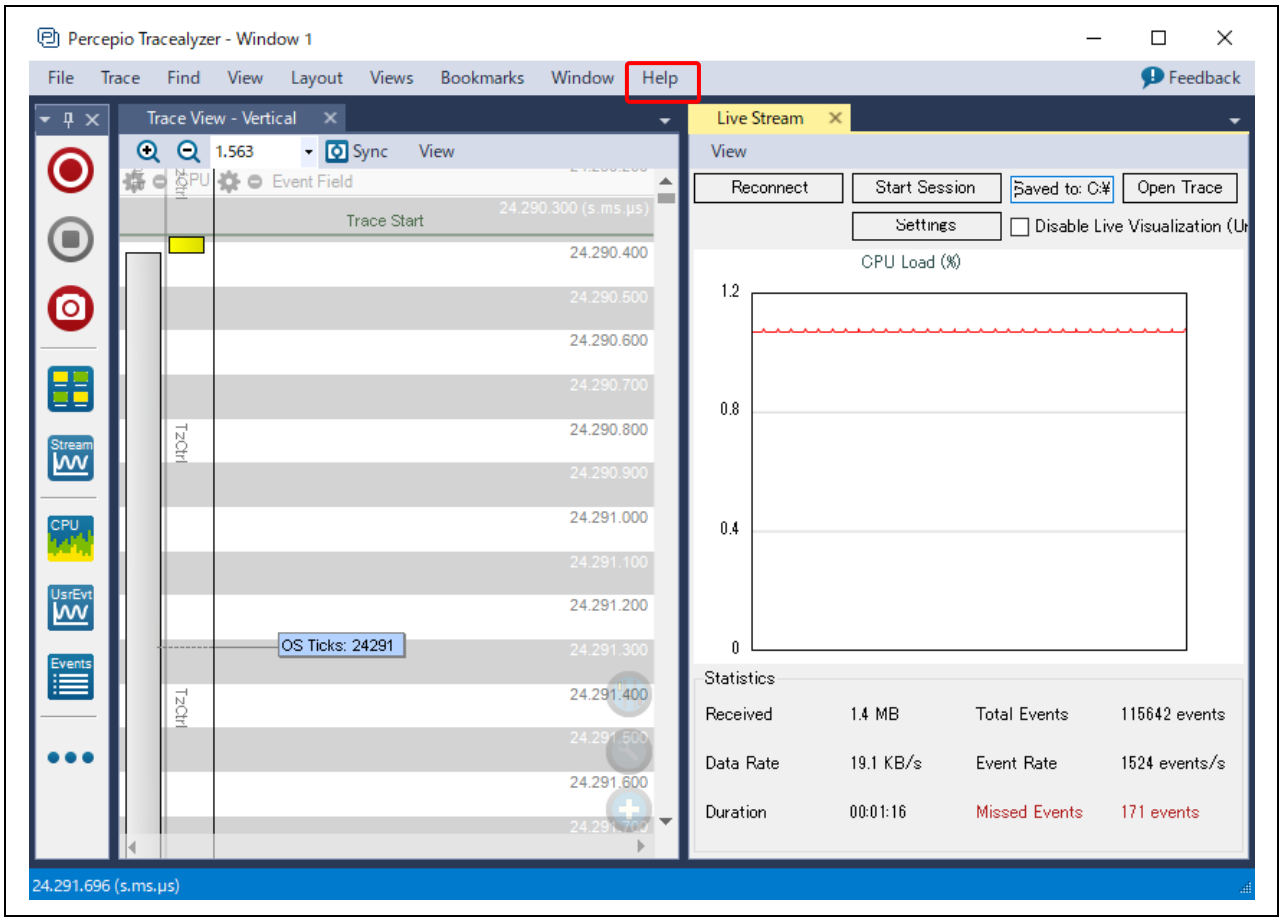



**Figure 55. Displayed Trace information**

## Website and Support

Visit the following URLs to learn about key elements of the RA family, download components, and related documentation, and get support.

| | |
|---|---|
| RA Product Information | www.renesas.com/ra |
| RA Product Support Forum | www.renesas.com/ra/forum |
| RA Flexible Software Package | www.renesas.com/FSP |
| Renesas Support | www.renesas.com/support |

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Jul.22.21 | — | First release document |
| | | | |

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)


## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com


## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.