Quick Setup Example on AIK-RA6M3 Solution Kit

Renesas Advanced (RA) Family – RA6 Series

Description

Welcome to Quick Setup Example for Renesas RA using AIK-RA6M3 Solution Kit! The objective of this workshop is to build a basic Renesas RA application utilizing Renesas tools.

You will start by setting up the display with the basic operations project. The application used in this lab is built to run on AIK-RA6M3 Solution Kit. A foundation Display project will be created from scratch and populated with several HAL drivers provided by the Flexible Software Package (FSP). Accelerometer and Ethernet demo projects are also added.

Objectives
• Configure AIK-RA6M3 kit to run display with the basic operations project
• Implement Accelerometer demo
• Implement Ethernet demo
	Prerequisites
	• Renesas AIK-RA6M3 VUI Solution Kit
	• Renesas Flexible Software Package 4.5.0 platform installation, which includes:
	  • e² studio 2023-10 or newer
	  • FSP 4.5.0 or newer
	  • GCC Arm Embedded 10.3.1
	• PC running Windows 10 64-bit with at least one USB port.
	• Serial terminal software such as PuTTY or TeraTerm (provided with the workshop)
	• J-Link RTT Viewer
	• Router with Ethernet connection

Skill Level • Basic familiarity with embedded electronics • Basic understanding of C language • Understanding of how to import projects into e² studio (optional – for use with ready checkpoint projects).	Time • 2 hours to complete

Workshop Sections

0 Setting up the hardware

Procedural Steps

0.1	To begin working with AIK-RA6M3 platform you will need the following:  AIK-RA6M3 Solution Kit  Display  OV2640 Camera module  Renesas ICM-42670-P PMOD Board  Display patch module  USB micro-B cable (included with the kit)  Ethernet Cable
0.2	Connect the AIK-RA6M3 kit to the PC using USB micro-B cable and J-Link OB USB port (J10) in the bottom left corner of the board. Connect Display patch module to PMOD 3 and the other end to the display. Also connect Renesas ICM-42670-P PMOD Board to PMOD1.
0.3	Verify that: • The blue LED2 (POWER) near the ethernet port is on. • The orange LED4 (LED OB) near the bottom edge of the board is on and not flashing.
0.4	Your kit and operating environment are now set-up and ready for evaluation and development.

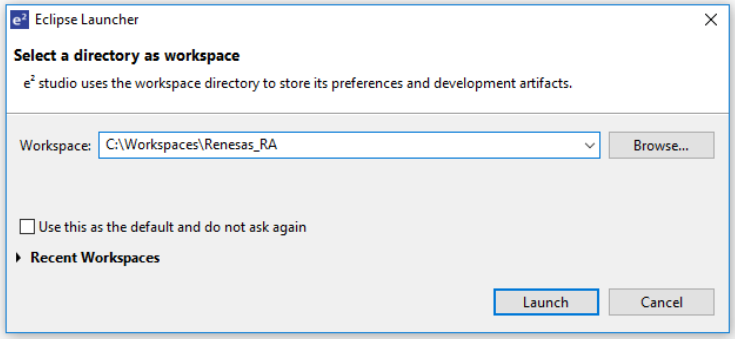
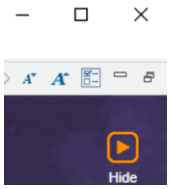
END OF SECTION

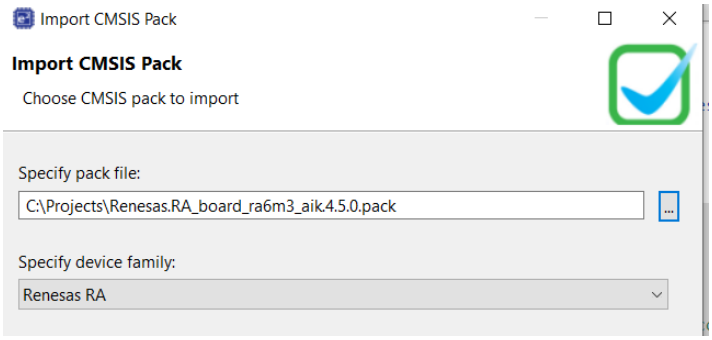
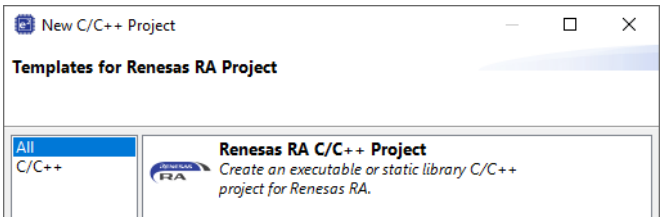
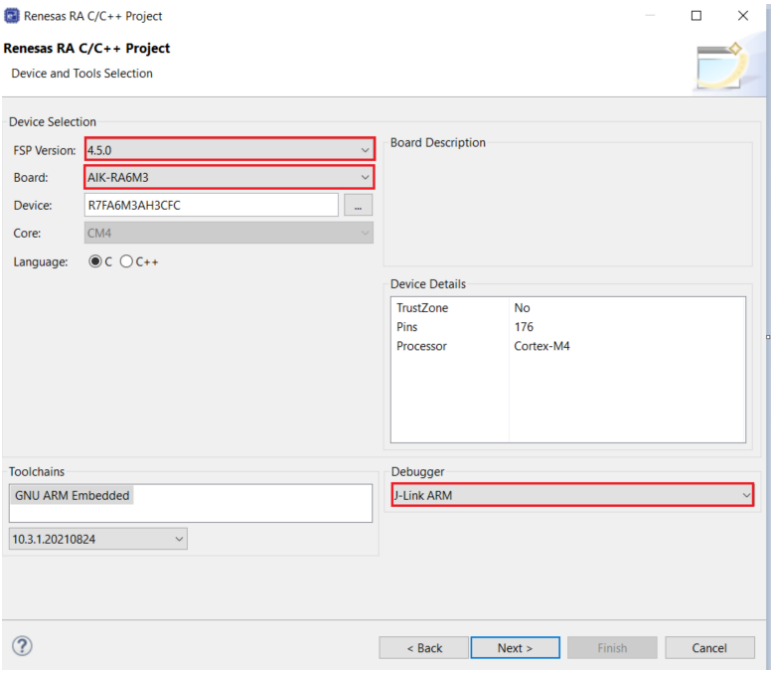
1 Implementing Display with the basic operations demo

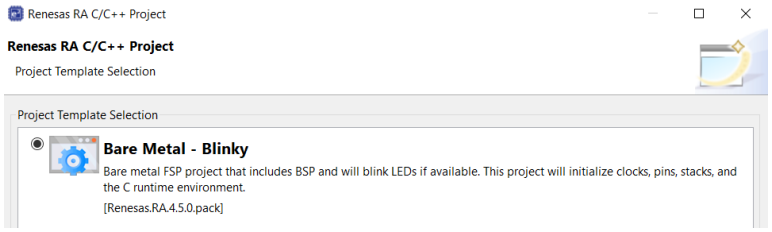
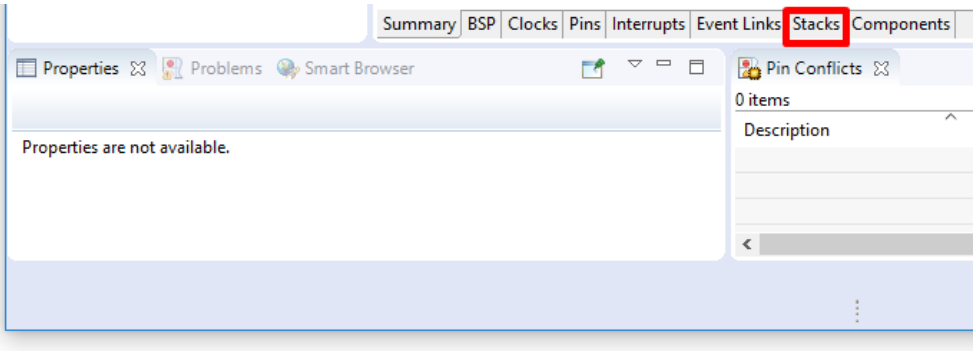
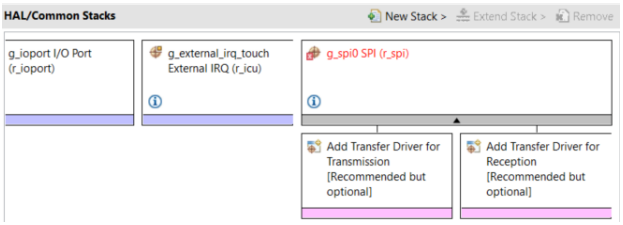
Overview

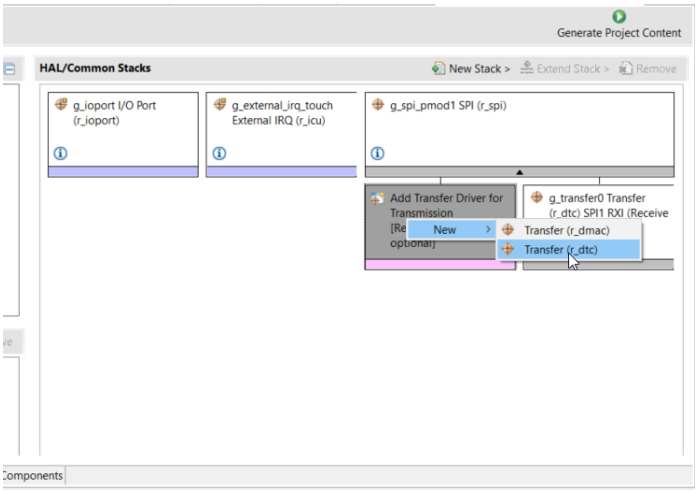
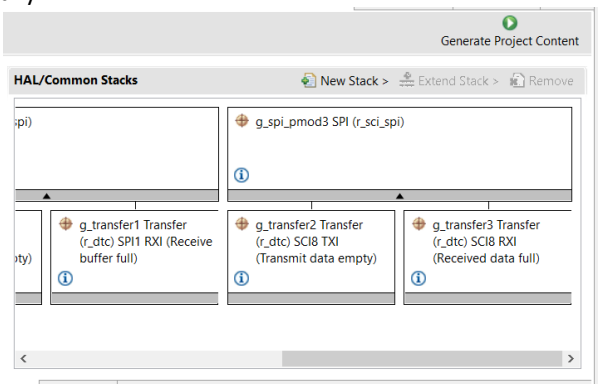
Following section describes in details steps required to create an e² studio workspace and set up a Display with basic operations-based project for AIK RA6M3 kit.

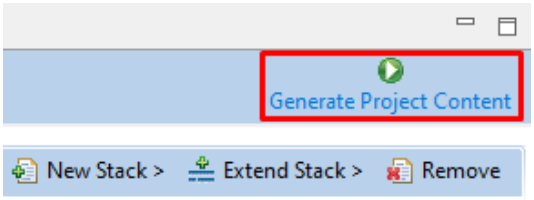
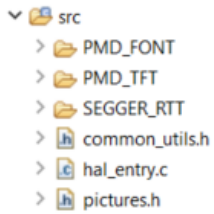
Procedural Steps

1.1	Launch e² studio. e² studio can be launched from the Windows start menu or directly from the installation folder. If you have multiple versions of e² studio installed, please make sure to launch the version of e² studio that was specified on the first page.
1.2	In the Eclipse Launcher window, specify the destination for the new workspace. It is recommended to keep the path simple and avoid using spaces. 
1.3	Click **Launch** to start e² studio in the specified path. If prompted, press **Apply** to dismiss pop up window asking for permission to log and report usage (it will remain disabled).
1.4	The welcome screen will show inside the new workspace. It can be dismissed by clicking on the Hide button in the top-right corner. 
1.5	If you already have installed the BSP for AIK-RA6M3 kit, proceed directly to step 2.8. Otherwise, go to **File -> Import** and Select **General -> CMSIS Pack**.

1.6	In the Import CMSIS Pack window, click **…** to browse for the .pack file containing BSP for AIK-RA6M3 kit (Renesas.RA_board_RA6M3.<version>.pack). Select **Renesas RA** from the drop-down box under Specify device family and click **Finish**. 
1.7	Click **OK** in the pop-up window confirming successful pack file import.
1.8	Go to **File -> New** and select **Renesas C/C++ Project**, then **Renesas RA**.
1.9	In the new project wizard window, select **Renesas RA C/C++ Project** and click **Next**. 
1.10	Specify a project name and Click **Next**.
1.11	Select FSP version matching your BSP and FSP installation (e.g., **4.5.0**) and set Board to **AIK-RA6M3**. Verify that the Debugger is set to **J-Link ARM** and click **Next**. 
1.12	On the next window, leave **Executable** and **No RTOS** selected. Click **Next**.

1.13	On the final page of the new project wizard select **Bare Metal – Blinky** and click **Finish**. 
1.14	When prompted to open the **FSP Configuration perspective**, click **Open Perspective**. The project is now set up to begin evaluation and development using the AIK kit.
1.15	Once new project is created, e^2 studio will switch to a layout optimized for developing Renesas RA projects. Select the **Stacks** tab at the bottom of the **FSP Configuration** pane visible in the middle. 
1.16	Access the **New Stack** menu again and select **Input -> External IRQ (r_icu)**. Use **Properties** tab to configure following properties for this new module: • Name g_external_irq_touch • Channel 6 • Trigger Failing • Digital Filtering Enable • Callback pmd_touch_irq_ep_callback
1.17	Open the **New Stack** menu (near the top-left corner) and navigate to **Connectivity -> SPI (r_spi)**.
1.18	A SPI driver module will be added inside the **HAL/Common group**. DTC modules will need to be populated. 

1.19	Click on **g_spi0 SPI (r_spi)**, go to the **Properties** tab and apply the following settings. You may need to expand the chevrons to access all of the properties: • Name g_spi_pmod1 • Channel 1 • Receive interrupt Priority Priority 2 • Transmit buffer Empty Interupt Priority Priority 2 • Transmit Complete Interupt Priority Priority 2 • Error Interupt Priority Priority 2 • Operating Mode Master • Callback spi_pmod1_callback • Bitrate 15000000 • SPI Mode Clock Synchronous Operation
1.20	In the SPI driver module, in the first DTC module press on the icon -> New and choose Transfer (r_dtc). Press the second DTC module -> New and choose Transfer (r_dtc). The fields are populated automatically. 
1.21	Open the **New Stack** menu (near the top-left corner) and navigate to **Connectivity -> SPI (r_sci_spi)**.
1.22	A second SPI driver module will be added inside the **HAL/Common group**. This time, DTC modules will be populated automatically. 

1.23	Click on **g_spi0 SPI (r_sci_spi)**, go to the **Properties** tab and apply the following settings. You may need to expand the chevrons to access all of the properties: • Name g_spi_pmod3 • Channel 8 • Receive interrupt Priority Priority 2 • Transmit buffer Empty Interupt Priority Priority 2 • Transmit Complete Interupt Priority Priority 2 • Error Interupt Priority Priority 2 • Callback sci_spi_pmod3_callback • Bitrate 15000000
1.24	RA Configuration for this section is complete. Apply changes to the project source by clicking the **Generate Project Content** button in the top-right corner of the Configurator window. When prompted to *Proceed with save and generate*, tick the box next to **Always save and generate without asking** and click **Proceed**. 
1.25	The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the **Properties** tab.
1.26	In the **Project Explorer** pane, expand the **src** folder in the project and add the following folders and files that can be found in the demo folder: • PMD_FONT • PMD_TFT • SEGGER_RTT • common_utils.h • pictures.h
1.27	In the **Project Explorer** pane, expand the **src** folder in the project and open **hal_entry.c**. 
1.28	**hal_entry.c** contains user application entry point (hal_entry function) for RTOS-less projects. The `R_BSP_WarmStart` callback is provided for the user to specify additional functions to be called during the FSP initialization sequence (e.g., pin configuration).
1.29	Add #include statement to include common_utils.h, PMD_TFT/pmd_tft.h, PMD_TFT/pmd_text.h, pictures.h near the top of the file.

1.30 Add static uint16_t buffer_rgb565[RENESAS_110_17_WIDTH*RENESAS_110_17_HEIGHT + (RENESAS_110_17_WIDTH*RENESAS_110_17_HEIGHT +1)/2]; after the includes.
After extern bsp_leds_t g_bsp_leds; add uint16_t act_state=0 ; uint16_t tft_ori_cnt=0;

1.31 **hal_entry.c** can be used to exercise API of the various modules configured inside FSP Configurator using Developer Assist or by writing code manually.

Following code can be used to completely replace contents of hal_entry.c to perform basic operations using the display for the AIK-RA6M3 board:

```
#include "hal_data.h"

// START USER include

//some definitions for RTT
#include "common_utils.h"
//some definitions for TFT
#include "PMD_TFT/pmd_tft.h"
#include "PMD_TFT/pmd_text.h"
// END USER include

//picture input by array
#include "pictures.h"
//

static uint16_t buffer_rgb565[RENESAS_110_17_WIDTH*RENESAS_110_17_HEIGHT +
                              (RENESAS_110_17_WIDTH*RENESAS_110_17_HEIGHT +1)/2
                             ];

extern bsp_leds_t g_bsp_leds;
uint16_t act_state=0 ;
uint16_t tft_ori_cnt=0;
// End user

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER


void hal_entry (void)
{
#if BSP_TZ_SECURE_BUILD

    /* Enter non-secure code */
    R_BSP_NonSecureEnter();
#endif

    /* Define the units to be used with the software delay function */
    const bsp_delay_units_t bsp_delay_units = BSP_DELAY_UNITS_MILLISECONDS;

    /* Set the blink frequency (must be <= bsp_delay_units */
    const uint32_t freq_in_hz = 2;

    /* Calculate the delay in terms of bsp_delay_units */
    const uint32_t delay = bsp_delay_units / freq_in_hz;

    /* LED type structure */
    bsp_leds_t leds = g_bsp_leds;

    /* If this board has no LEDs then trap here */
    if (0 == leds.led_count)
    {
        while (1)
        {
            ;                           // There are no LEDs on this board
        }
    }

    /* Holds level to set for pins */
    bsp_io_level_t pin_level = BSP_IO_LEVEL_LOW;

    // START USER include
```

```
    /* set the TFT orientation */
    tft_set_ori_set (TFT_R90);

    /* init the TFT */
    tft_configure ();

    /* draw some shapes */
    tft_set_draw_color(0xff0000);
    tft_draw_rect((int16_t)20,(int16_t)20,
                  (int16_t)(tft_get_act_width()/2-1),(int16_t)(tft_get_act_height()/2-1));
    tft_set_draw_color(0x00ff00);
    tft_draw_rect((int16_t)(tft_get_act_width()/2), (int16_t)(tft_get_act_height()/2),
                  (int16_t)(tft_get_act_width()-1-20),(int16_t)(tft_get_act_height()-1-20));

    // START USER include
    pmd_text_init();


    while (1)
    {
        /* Enable access to the PFS registers. If using r_ioport module then register
protection is automatically
         * handled. This code uses BSP IO functions to show how it is used.
         */
        R_BSP_PinAccessEnable();

        /* Update all board LEDs */
        for (uint32_t i = 0; i < leds.led_count; i++)
        {
            /* Get pin to toggle */
            uint32_t pin = leds.p_leds[i];

            /* Write to this pin */
            R_BSP_PinWrite((bsp_io_port_pin_t) pin, pin_level);
        }

        /* Protect PFS registers */
        R_BSP_PinAccessDisable();

        /* Toggle level for next write */
        if (BSP_IO_LEVEL_LOW == pin_level)
        {
            pin_level = BSP_IO_LEVEL_HIGH;
        }
        else
        {
            pin_level = BSP_IO_LEVEL_LOW;
        }

        /* some output on the Display */
        //act_state=0 ;
        switch (act_state++)
        {
            case 0 :
                if (tft_ori_cnt > 3)
                    tft_ori_cnt = 0;
                if (tft_ori_cnt == 0)
                    tft_set_ori (TFT_R0);
                if (tft_ori_cnt == 1)
                    tft_set_ori (TFT_R90);
                if (tft_ori_cnt == 2)
                    tft_set_ori (TFT_R180);
                if (tft_ori_cnt == 3)
                    tft_set_ori (TFT_R270);

                //clear the screen
                tft_cls(CLS_COLOR) ;
                /* draw some shapes */
                tft_set_draw_color(0x0000ff);
                tft_draw_rect(5,5, 15,15);  //This top left corner

                tft_set_draw_color(0xff0000);
                tft_draw_rect(20,20,
                              (int16_t)(tft_get_act_width()/2-1),
```

```
                                (int16_t)(tft_get_act_height()/2-1));
                    tft_set_draw_color(0x00ff00);
                    tft_draw_rect((int16_t)(tft_get_act_width()/2),
                                (int16_t)(tft_get_act_height()/2),
                                (int16_t)(tft_get_act_width()-1-20),
                                (int16_t)(tft_get_act_height()-1-20));

                    pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
                    pmd_text_set_f_color(0xff0000ff); // A - is 255 use foreground color blue
                    pmd_text_set_font(1) ;
                    pmd_text_set_rotation(TFT_TXT_R0);
                    if (tft_ori_cnt == 0)
                        pmd_draw_string("TFT_R0",20,3);
                    if (tft_ori_cnt == 1)
                        pmd_draw_string("TFT_R90",20,3);
                    if (tft_ori_cnt == 2)
                        pmd_draw_string("TFT_R180",20,3);
                    if (tft_ori_cnt == 3)
                        pmd_draw_string("TFT_R270",20,3);
                    tft_ori_cnt++;
                    break ;

                case 2:
                    // output a RGB565 picture at x(centered horizontal) y(bottom in vertical)
                    // background of picture has color 0x00000 we want to keep the LCD background
so alpha -is 0x00
                    // --> color = 0x00000000u
                    // foreground of picture we want to blend E7
                    // foreground RGB is set to 0x00000 (no use)
                    // --> color = 0xE7000000u

                    tft_blit_copy_blend ((uint16_t*) buffer_rgb565, (uint16_t*)
picture_renesas_110_17_rgb565,
                                         (int16_t) ((tft_get_act_width () - RENESAS_110_17_WIDTH) /
2),
                                         (int16_t) ((tft_get_act_height () - RENESAS_110_17_HEIGHT
- 1)),
                                         RENESAS_110_17_WIDTH,
                                         RENESAS_110_17_HEIGHT,
                                         0x00000000u,  // BG color is RGB 0x000000 and alpha will
be 0x00 so keep background on LCD
                                         0xE7000000u); // FG color is unused only alpha channel
will be used for blend

                case 4:
                    pmd_text_set_rotation(TFT_TXT_R0);
                    pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
                    pmd_text_set_f_color(0xff000000); //black
                    pmd_text_set_font(1) ;
                    pmd_draw_string("Hey\f   Renesas\r\n",
                                    (int16_t)(tft_get_act_width()/2),
                                    (int16_t)(tft_get_act_height()/2));
                    pmd_text_set_font(0) ;
                    pmd_draw_string(" TFT_TXT_R0 \r\n",
                                    pmd_text_get_cursor_x(),       // please take care for correct
offsets to screen and text start point
                                    pmd_text_get_cursor_y() + 10); // please take care for correct
offsets to screen and text start point
                    break;

                case 6:
                    pmd_text_set_rotation(TFT_TXT_R90);
                    pmd_text_set_b_color(0x38eff0ef); // A lets'use some alpha blending
                    pmd_text_set_f_color(0xff0000bf); //blue ARGB
                    pmd_text_set_font(1) ;
                    pmd_draw_string("Hey\f   Renesas\r\n",
                                    (int16_t)(tft_get_act_width()/2),
                                    (int16_t)(tft_get_act_height()/2));
                    pmd_text_set_font(0) ;
                    pmd_draw_string(" TFT_TXT_R90 \r\n",
                                    pmd_text_get_cursor_x() + 10 , // please take care for correct
offsets to screen and text start point
```

```
                                pmd_text_get_cursor_y());      // please take care for correct
offsets to screen and text start point
                break;

            case 8:
                pmd_text_set_rotation(TFT_TXT_R180);
                pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
                pmd_text_set_f_color(0xffffffff); //white ARGB
                pmd_text_set_font(1) ;
                pmd_draw_string("Hey\f   Renesas\r\n",
                                (int16_t)(tft_get_act_width()/2),
                                (int16_t)(tft_get_act_height()/2));
                pmd_text_set_font(0) ;
                pmd_draw_string(" TFT_TXT_R180 \r\n",
                                pmd_text_get_cursor_x() ,       // please take care for correct
offsets to screen and text start point
                                pmd_text_get_cursor_y() -10 );  // please take care for correct
offsets to screen and text start point
                break;

            case 10:
                pmd_text_set_rotation(TFT_TXT_R270);
                pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
                pmd_text_set_f_color(0xff0000ff); //blue ARGB
                pmd_text_set_font(1) ;
                pmd_draw_string("Hey\f   Renesas\r\n",
                                (int16_t)(tft_get_act_width()/2),
                                (int16_t)(tft_get_act_height()/2));
                pmd_text_set_font(0) ;
                pmd_draw_string(" TFT_TXT_R270 \r\n",
                                pmd_text_get_cursor_x() -10, // please take care for correct
offsets to screen and text start point
                                pmd_text_get_cursor_y() );   // please take care for correct
offsets to screen and text start point
                break;

            case 12:
                tft_set_draw_color (0x8f8f8f);
                tft_draw_rect (20, 20, (int16_t) (tft_get_act_width () - 1 - 20),
                               (int16_t) (tft_get_act_height () - 1 - 20));
            break;

            case 13:
            {
                int16_t xt = 30;
                int16_t x = (int16_t) ((tft_get_act_width () - 0 ) / 2 + 25 );
                int16_t y2 = 40;

                pmd_text_set_rotation(TFT_TXT_R0);
                pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
                pmd_text_set_f_color(0xffb0ffb0); //  ARGB
                pmd_text_set_font(1) ;
                pmd_draw_string("circle",
                                xt,
                                (int16_t)(y2-pmd_font_get_height()/2));

                tft_set_draw_color(0x2020ff);
                tft_draw_circle(x + 6, y2, 11);
                tft_draw_circle(x + 6, y2, 9);
            }
                break ;

            case 14:
            {
                int16_t xt = 30;
                int16_t x = (int16_t) ((tft_get_act_width () - 0 ) / 2 + 25 );
                int16_t y3 = 70;

                pmd_text_set_rotation(TFT_TXT_R0);
                pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
                pmd_text_set_f_color(0xffb0ffb0); //  ARGB
                pmd_text_set_font(1) ;
                // circle does not support line width
                pmd_draw_string("filled circle",
                                xt,
```

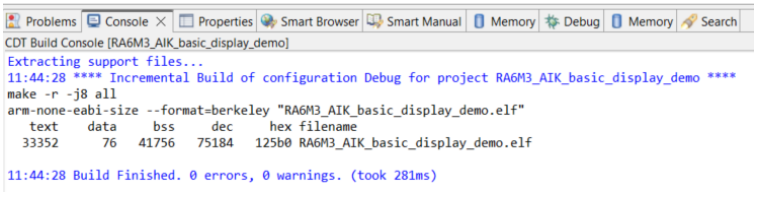
```
                                (int16_t)(y3-pmd_font_get_height()/2));

            tft_set_draw_color(0x2020ff);
            tft_draw_filled_circle(x + 6, y3, 11);
        }
            break ;

        case 15:
        {
            int16_t xt = 30;
            int16_t x = (int16_t) ((tft_get_act_width () - 0 ) / 2 + 25);
            int16_t y1 = 100 -26/2;

            pmd_text_set_rotation(TFT_TXT_R0);
            pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
            pmd_text_set_f_color(0xffb0ffb0); //  ARGB
            pmd_text_set_font(1) ;

            pmd_draw_string("frame",
                            xt,
                            (int16_t)(y1-pmd_font_get_height()/2+13));

            tft_set_draw_color(0x2020ff);
            pmd_set_linesize (1);
            pmd_draw_frame(x, y1, 60, 26);

            }

            break ;


        case 16:
        {
            int16_t xt = 30;
            int16_t x = (int16_t) ((tft_get_act_width () - 0 ) / 2 + 25);
            int16_t y2 = 130 -26/2;

            pmd_text_set_rotation(TFT_TXT_R0);
            pmd_text_set_b_color(0x0000ff00); // A - is 0 keep background
            pmd_text_set_f_color(0xffb0ffb0); //  ARGB
            pmd_text_set_font(1) ;

            pmd_draw_string("text frame",
                            xt,
                            (int16_t)(y2-pmd_font_get_height()/2+13));

            pmd_set_linesize (3);
            pmd_text_set_font(0) ;

            pmd_text_set_b_color(0xff0000Af); // overwrite background
            pmd_text_set_f_color(0xffffffff); //  ARGB
            pmd_draw_text_frame("IN_BOX\f42", x, y2, 60, 26);

            }

            break ;


        case 17:
        {
            int16_t xt = 30;
            int16_t x = (int16_t) ((tft_get_act_width () - 0) / 2 + 25);
            int16_t y3 = 160 + 10;

            pmd_text_set_rotation (TFT_TXT_R0);
            pmd_text_set_b_color (0x0000ff00); // A - is 0 keep background
            pmd_text_set_f_color (0xffb0ffb0); //  ARGB
            pmd_text_set_font (1);
            pmd_draw_string ("line", xt, (int16_t)(y3 - pmd_font_get_height () / 2 + 0) );

            tft_draw_line (0 + x, 20 + y3, 10 + x, 0 + y3);
            tft_draw_line (0 + x, 20 + y3, -10 + x, 0 + y3);

            tft_draw_line (-10 + x, 0 + y3, 0 + x, -20 + y3);
            tft_draw_line (+10 + x, 0 + y3, 0 + x, -20 + y3);
```

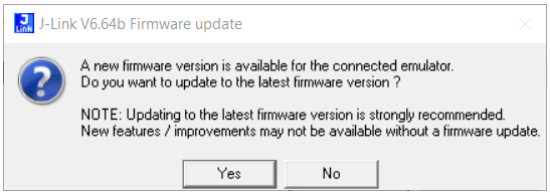
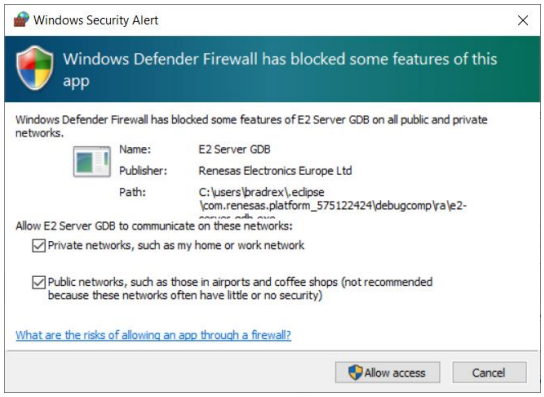
```
                tft_draw_v_line (x + 20, y3 - 10, y3 + 10, 3);
                tft_draw_h_line (x + 30, y3, x + 30 + 20, 3);

                }

                break ;


            case 18:
            {
                int16_t xt = 30;
                int16_t x = (int16_t) ((tft_get_act_width () - 0) / 2 + 25);
                int16_t x2 = x ;
                int16_t y4 = 190 + 10 ;

                if ( tft_get_ori() == TFT_R0 || tft_get_ori() == TFT_R180)
                    x2 -= 40 ;

                pmd_text_set_rotation (TFT_TXT_R0);
                pmd_text_set_b_color (0x0000ff00); // A - is 0 keep background
                pmd_text_set_f_color (0xffb0ffb0); //  ARGB
                pmd_text_set_font (1);
                pmd_draw_string ("picture", xt, (int16_t)(y4 + (RENESAS_110_17_HEIGHT -
pmd_font_get_height ()) / 2 + 0) );

                // output a RGB565 picture at x(centered horizontal) y(bottom in vertical)
                // background of picture has color 0x00000 we want to keep the LCD background
so alpha -is 0x00
                // --> color = 0x00000000u
                // foreground of picture we want to blend E7
                // foreground RGB is set to 0x00000 (no use)
                // --> color = 0xE7000000u

                tft_blit_copy_blend ((uint16_t*) buffer_rgb565, (uint16_t*)
picture_renesas_110_17_rgb565,
                                     x2,
                                     y4,
                                     RENESAS_110_17_WIDTH,
                                     RENESAS_110_17_HEIGHT,
                                     0x00000000u,  // BG color is RGB 0x000000 and alpha will
be 0x00 so keep background on LCD
                                     0xff000000u); // FG color is unused only alpha channel
will be used for blend

                }


                break ;


            case 24:
                act_state = 0 ;
                break ;

            default:
               break ;

        }

        /* Delay */
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
    }
}

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;
```

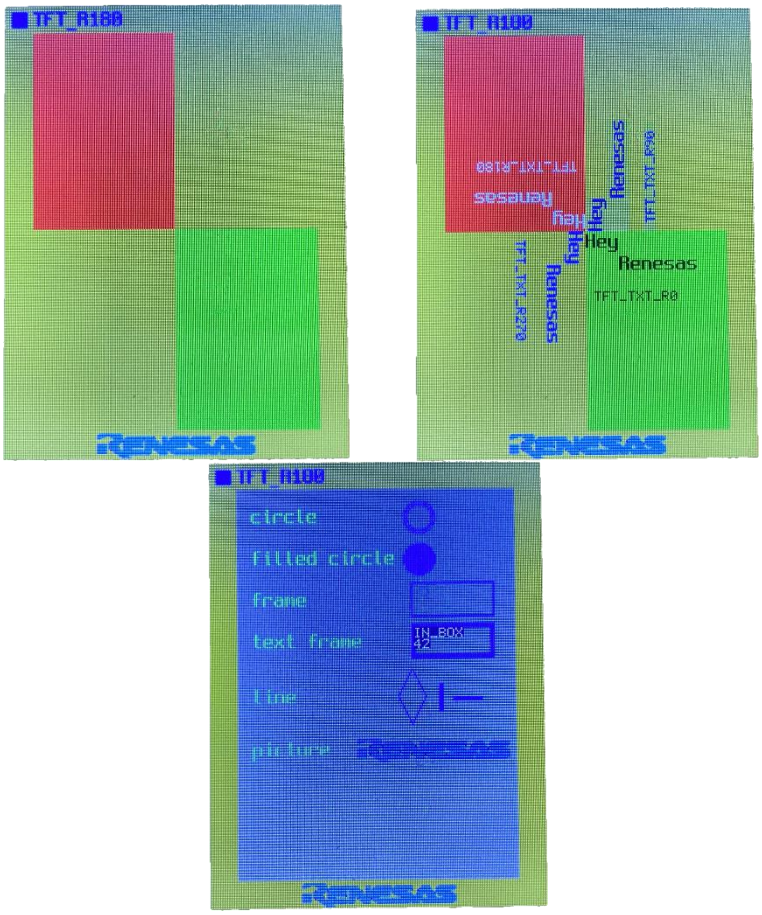
```
        /* Would normally have to wait tDSTOP(6us) for data flash recovery. Placing the enable
here, before clock and
         * C runtime initialization, should negate the need for a delay since the
initialization will typically take more than 6us. */
#endif
    }

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

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


#if BSP_TZ_SECURE_BUILD

FSP_CPP_HEADER
BSP_CMSE_NONSECURE_ENTRY void template_nonsecure_callable ();

/* Trustzone Secure Projects require at least one nonsecure callable function in order to build
(Remove this if it is not required to build). */
BSP_CMSE_NONSECURE_ENTRY void template_nonsecure_callable ()
{

}
FSP_CPP_FOOTER

#endif
```

1.32 The project is now ready to compile. Press the “hammer” icon to start building the project.

1.33 Once the build has finished, the **Console** pane in the lower-right corner of e² studio will report zero errors :



1.34	Check that the Display is connected to PMOD3 as seen below. 
1.35	The application is now ready to be programmed and run on the AIK kit. Press the “bug” icon to begin the debug session. 
1.36	You may be prompted to update the J-Link debugger firmware. You can click **Yes** to update. It will take a few moments to complete. 
1.37	Windows could also prompt you to allow the GDB server through your firewall. Click the checkbox to allow it through private networks, then **Allow** access. 
1.38	e^2 studio will perform flash programming routines and prompt to switch to **Debug** perspective. Select the check box by **Remember my decision** and click **Switch**.
1.39	The debug session is now started, and the application is paused at its entry function (`SystemInit()` in `Reset_Handler`). At this point, you can set up additional debug features such as variable and expressions views before the program is executed.

1.40	Click the Resume button or press F8 on the keyboard to start the application. 
1.41	The Program will stop again, this time at the start of the main function. Low-level initialization routines are now completed. Press Resume or F8 again to resume the application and begin executing user code.
1.42	Go back to the devices display to observe the output from the AIK kit. The display will show the Renesas banner in the middle bottom, “TFT_Rxx” in the top left corner. Also it will show multiple “Hey Renesas” & “TFT_TXT_Rxx” with xx being the degrees and will also have basic geometrical figures as shown in the pictures bellow. 
1.43	Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session. 
1.44	Full project is also available in GitHub under https://github.com/renesas/aiot-ready/tree/master/kits/ra6m3_aik/qse

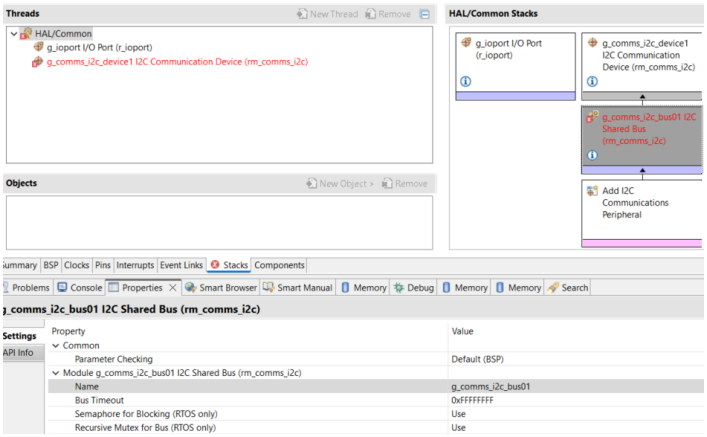
END OF SECTION

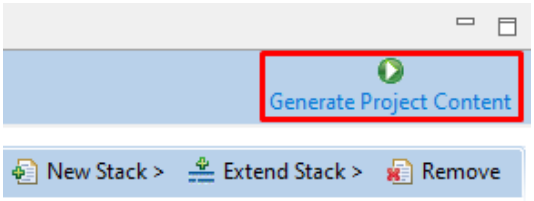
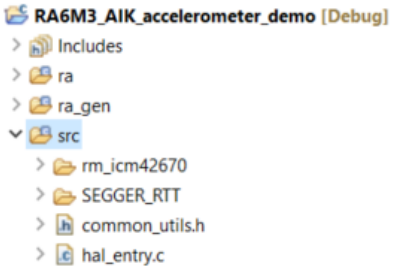
2 Implementing Accelerometer demo

Overview

Following section describes in details steps required to set up an accelerometer demo project for AIK RA6M3 kit.

Procedural Steps

2.1	Create a new project and follow the steps described from 1.1 to 1.15.
2.2	Access the **New Stack** menu and select **Connectivity -> I2C Communication Device (rm_comms_i2c)**. Use **Properties** tab to configure following properties for this new module: • Name g_comms_i2c_device1 • Slave Address 0x68 • Callback rm_icm42670_comms_i2c_callback
2.3	In the I2C Communication Device module, in **g_comms_i2c_bus01 I2C Shared Bus (rm_comms_i2c)** Use **Properties** tab to configure following properties for this new module: • General -> Name g_comms_i2c_bus01 
2.4	In the I2C Communication Device module, in **Add I2C Communications peripherals** press on the icon -> New and choose **I2C Master (r_iic_master)**. Use **Properties** tab to configure following properties for this new module: • Common -> DTC on Transmission and Reception Enabled • Name g_i2c_master1 • Channel 1 • Slave Address 0x68
2.5	In the I2C Communication Device module, in the first **Add DTC Driver for Transmission [optional]** press on the icon -> New and choose **Transfer (r_dtc)**.
2.6	Access the **New Stack** menu again and select **Input -> External IRQ (r_icu)**. Use **Properties** tab to configure following properties for this new module: • Name g_external_irq6_pmod1 • Channel 6

2.7	Access the **New Stack** menu again and select **Connectivity -> UART (r_sci_uart)**. Use **Properties** tab to configure following properties for this new module: • Common -> DTC Support — Enable • General -> Name — g_uart4_pmod5 • General -> Channel — 4 • Baud -> Max Error (%) — 2 • Interrupts -> Callback — rm_uart_callback • General -> Channel — 4
2.8	In the UART module, in **Add DTC Driver for Transmission [optional]** press on the icon -> New and choose **Transfer (r_dtc)**.
2.9	RA Configuration for this section is complete. Apply changes to the project source by clicking the **Generate Project Content** button in the top-right corner of the Configurator window. When prompted to *Proceed with save and generate*, tick the box next to **Always save and generate without asking** and click **Proceed**. 
2.10	The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the **Properties** tab.
2.11	In the **Project Explorer** pane, expand the **src** folder in the project and add the following folders and files than can be found in the demo folder: • rm_icm42670 • SEGGER_RTT • common_utils.h
2.12	In the **Project Explorer** pane, expand the **src** folder in the project and open **hal_entry.c**. 
2.13	**hal_entry.c** contains user application entry point (hal_entry function) for RTOS-less projects. The `R_BSP_WarmStart` callback is provided for the user to specify additional functions to be called during the FSP initialization sequence (e.g., pin configuration).
2.14	At the beginning of hal_entry.c before “void R_BSP_WarmStart(bsp_warm_start_event_t event);” add the #include statement for the following: • `#include <stdio.h>` • `#include <string.h>` • `#include "common_utils.h"` • `#include "rm_icm42670/rm_icm42670_hal_data.h"` • `#include "rm_icm42670/rm_common_uart.h"`

- #define RM_ICM42670_EXAMPLE_DELAY_50MS 50
- #define RM_ICM42670_EXAMPLE_DELAY_1US 10
- #define RM_ICM42670_EXAMPLE_IRQ_ENABLE 1

2.15

hal_entry.c can be used to exercise API of the various modules configured inside FSP Configurator using Developer Assist or by writing code manually.
Following code can be used to completely replace contents of hal_entry.c to perform basic operations using the display for the AIK-RA6M3 board:

```
#include "hal_data.h"

#include <stdio.h>
#include <string.h>
#include "common_utils.h"
#include "rm_icm42670/rm_icm42670_hal_data.h"
#include "rm_icm42670/rm_common_uart.h"

#define RM_ICM42670_EXAMPLE_DELAY_50MS 50
#define RM_ICM42670_EXAMPLE_DELAY_1US 10
#define RM_ICM42670_EXAMPLE_IRQ_ENABLE 1

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
//void __attribute__((optimize("O0"))) init_i2c_comm(void) ;
void init_i2c_comm(void) ;
fsp_err_t rm_icm42670_irq_open (rm_icm42670_ctrl_t * const p_api_ctrl);
FSP_CPP_FOOTER

#ifdef RTT_DEBUG_ON
char segBuf1[16] ;
char segBuf2[16] ;
#endif

volatile rm_comms_i2c_bus_extended_cfg_t * p_extend ;
volatile i2c_master_instance_t * p_driver_instance ;

/***
 * @brief initialize the comms bus interface
 */
void init_i2c_comm(void)
{
    fsp_err_t err = FSP_SUCCESS;
    /* Open the I2C bus if it is not already open. */
    p_extend = (rm_comms_i2c_bus_extended_cfg_t*) g_icm42670_sensor0_cfg.p_comms_instance-
>p_cfg->p_extend;
    p_driver_instance = (i2c_master_instance_t*) p_extend->p_driver_instance;
    p_driver_instance->p_ctrl = &g_icm42670_sensor0_ctrl;
    err = p_driver_instance->p_api->open (p_driver_instance->p_ctrl, p_driver_instance->p_cfg);
    if(err != FSP_SUCCESS){__BKPT(0);}

#if BSP_CFG_RTOS
    /* Create a semaphore for blocking if a semaphore is not NULL */
    if (NULL != p_extend->p_blocking_semaphore)
    {
 #if BSP_CFG_RTOS == 1                // AzureOS
       tx_semaphore_create(p_extend->p_blocking_semaphore->p_semaphore_handle,
                           p_extend->p_blocking_semaphore->p_semaphore_name,
                           (ULONG) 0);
 #elif BSP_CFG_RTOS == 2              // FreeRTOS
       *(p_extend->p_blocking_semaphore->p_semaphore_handle) =
           xSemaphoreCreateCountingStatic((UBaseType_t) 1,
                                          (UBaseType_t) 0,
                                          p_extend->p_blocking_semaphore->p_semaphore_memory);
 #endif
    }

    /* Create a recursive mutex for bus lock if a recursive mutex is not NULL */
    if (NULL != p_extend->p_bus_recursive_mutex)
    {
 #if BSP_CFG_RTOS == 1                // AzureOS
       tx_mutex_create(p_extend->p_bus_recursive_mutex->p_mutex_handle,
                       p_extend->p_bus_recursive_mutex->p_mutex_name,
                       TX_INHERIT);
```

```
 #elif BSP_CFG_RTOS == 2                // FreeRTOS
        *(p_extend->p_bus_recursive_mutex->p_mutex_handle) =
            xSemaphoreCreateRecursiveMutexStatic(p_extend->p_bus_recursive_mutex-
>p_mutex_memory);
 #endif
    }
#endif

}


/***********************************************************************************************
*********************//**
 * main() is generated by the RA Configuration editor and is used to generate threads if an
RTOS is used.  This function
 * is called by main() when no RTOS is used.

***********************************************************************************************
**********************/
void hal_entry(void)
{
    /* TODO: add your own code here */

#if BSP_TZ_SECURE_BUILD
    /* Enter non-secure code */
    R_BSP_NonSecureEnter();
#endif

    fsp_err_t err = FSP_SUCCESS;
    rm_icm42670_raw_data_t raw_data;
    rm_icm42670_accel_data_t icm42670_accel_data;
    rm_icm42670_gyro_data_t icm42670_gyro_data;
    rm_icm42670_temp_data_t icm42670_temp_data;

#if 0 == RM_ICM42670_EXAMPLE_IRQ_ENABLE
    rm_icm42670_device_status_t device_status;
#endif

    /* Enable access to the PFS registers. If using r_ioport module then register protection is
automatically
     * handled. This code uses BSP IO functions to show how it is used.
     */
    R_BSP_PinAccessEnable ();
    R_BSP_PinWrite(LED1_BLUE, BSP_IO_LEVEL_HIGH);

    /* Open the uart bus if it is not already open. */

    err = rm_uart_initialize ();
    if ( err != FSP_SUCCESS)
    {
        R_BSP_PinWrite(LED1_RED, BSP_IO_LEVEL_HIGH);
    }
    else
    {
        R_BSP_PinWrite(LED1_GREEN, BSP_IO_LEVEL_HIGH);
    }
    R_BSP_PinWrite(LED1_BLUE, BSP_IO_LEVEL_LOW);

    /* cursor home */
    printf ("%c[H", 27);

#ifdef RTT_DEBUG_ON
    // RTT seems not to support cursor home
    //APP_PRINT("\x1B[H");
#endif

    /* cls  terminal clear screen */
    printf ("%c[2J", 27);
#ifdef RTT_DEBUG_ON
    APP_PRINT ("RTT_CTRL_CLEAR");

    APP_PRINT (BANNER_INFO);
#endif

    printf ("UART                : initialized\r\n");
```

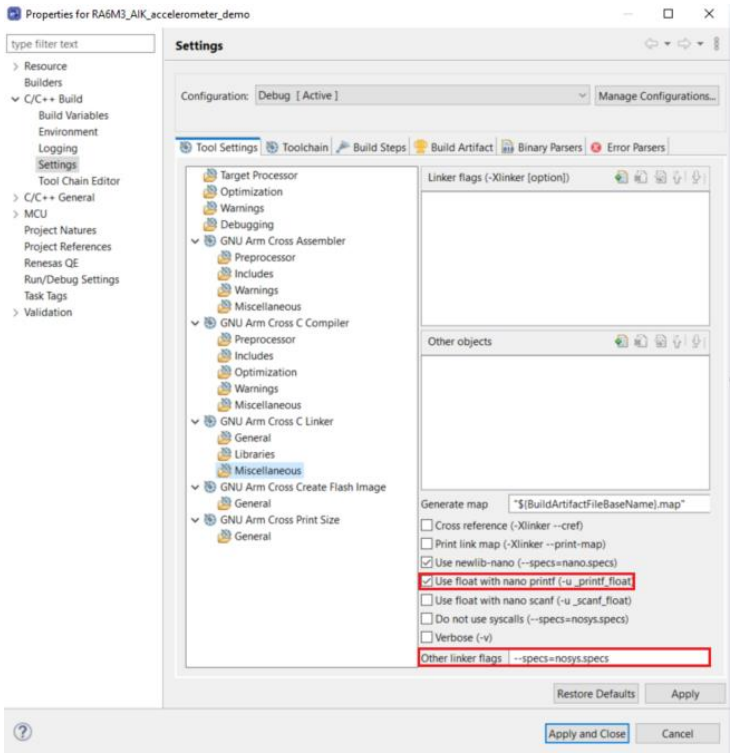
```
#ifdef RTT_DEBUG_ON
    APP_PRINT ("UART                : initialized\r\n");
#endif

    /* init the i2c comm interface */
    init_i2c_comm ();
    printf ("I2c common interface : initialized\r\n");
#ifdef RTT_DEBUG_ON
    APP_PRINT ("I2c common interface : initialized\r\n");
#endif

    /* After reset unlock the Open ICM42670 state */
    g_icm42670_sensor0_ctrl.open = 0;
    /* Open ICM42670 */
    err = RM_ICM42670_Open (&g_icm42670_sensor0_ctrl, &g_icm42670_sensor0_cfg);
    if (err != FSP_SUCCESS)
    {
        R_BSP_PinWrite(LED1_RED, BSP_IO_LEVEL_HIGH);
        __BKPT(0);
    }
    printf ("ICM42670 module      : initialized\r\n");
#ifdef RTT_DEBUG_ON
    APP_PRINT ("ICM42670 module      : initialized\r\n");
#endif

    g_icm42670_interrupt_cfg.int_config |= 0x01 ; // use active high (or change
g_external_irq6_pmod1 raising to falling)
    err = RM_ICM42670_DeviceInterruptCfgSet (&g_icm42670_sensor0_ctrl,
g_icm42670_interrupt_cfg);
    if (err != FSP_SUCCESS)
    {
        R_BSP_PinWrite(LED1_RED, BSP_IO_LEVEL_HIGH);
        __BKPT(0);
    }
    printf ("ICM42670 interrupts  : initialized\r\n");
#ifdef RTT_DEBUG_ON
    APP_PRINT ("ICM42670 interrupts  : initialized\r\n");
#endif

   /* Start measurement in data ready mode */
    err = RM_ICM42670_MeasurementStart (&g_icm42670_sensor0_ctrl);
    if (err != FSP_SUCCESS)
    {
        R_BSP_PinWrite(LED1_RED, BSP_IO_LEVEL_HIGH);
        __BKPT(0);
    }
    printf ("ICM42670 measurement : started\r\n");
#ifdef RTT_DEBUG_ON
    APP_PRINT ("ICM42670 measurement : started\r\n");
#endif

    /* Open external IRQ */
    err = rm_icm42670_irq_open (&g_icm42670_sensor0_ctrl);
    if (err != FSP_SUCCESS)
    {
        R_BSP_PinWrite(LED1_RED, BSP_IO_LEVEL_HIGH);
        __BKPT(0);
    }
    printf ("ICM42670 interrupt   : opened\r\n");
#ifdef RTT_DEBUG_ON
    APP_PRINT ("ICM42670 interrupt   : opened\r\n");
#endif

    err = R_ICU_ExternalIrqEnable (&g_external_irq6_pmod1_ctrl);
    if (err != FSP_SUCCESS)
    {
        R_BSP_PinWrite(LED1_RED, BSP_IO_LEVEL_HIGH);
        __BKPT(0);
    }
    printf ("ICM42670 interrupt   : enabled\r\n");
#ifdef RTT_DEBUG_ON
    APP_PRINT ("ICM42670 interrupt   : enabled\r\n");
#endif

    /*
```

```
     * Example :
     * Device interrupt : data ready mode
     */

    R_BSP_SoftwareDelay(1500, BSP_DELAY_UNITS_MILLISECONDS);

    //cls  terminal clear screen
    printf ("%c[2J", 27);
#ifdef RTT_DEBUG_ON
    APP_PRINT (RTT_CTRL_CLEAR);
#endif

    while (true)
    {
#if RM_ICM42670_EXAMPLE_IRQ_ENABLE

        /* Wait IRQ callback */
        while (0 == g_irq_flag)
        {
            /* Wait callback */
        }
        g_irq_flag = 0;
#else
        do
        {
            RM_ICM42670_DeviceStatusGet (&g_icm42670_sensor0_ctrl, &device_status);
            rm_icm42670_device_status_check (&g_icm42670_sensor0_ctrl);
        }
        while (false == device_status.data_ready);
#endif
#if 1
        /* cursor home */
        printf ("%c[H", 27);
#ifdef RTT_DEBUG_ON
        // RTT seems not to support cursor home
        //APP_PRINT("\x1B[H");
#endif

        /* Read Temperature data */
        RM_ICM42670_TempRead (&g_icm42670_sensor0_ctrl, &raw_data);
        /* Calculate Temperature data */
        RM_ICM42670_TempDataCalculate (&g_icm42670_sensor0_ctrl, &raw_data,
&icm42670_temp_data);

        /* Output Tempature data to console */
        printf ("                    \r\n");
        printf ("Temperature: %3.1f [%+3d] degrees Celsius\r\n",
icm42670_temp_data.temp_data_float,
                icm42670_temp_data.temp_data);

#ifdef RTT_DEBUG_ON
        snprintf(segBuf1,sizeof(segBuf1)-1,"%3.1f",icm42670_temp_data.temp_data_float);
        snprintf(segBuf2,sizeof(segBuf2)-1,"%+3d",icm42670_temp_data.temp_data);

        APP_PRINT ("\r\n");
        APP_PRINT ("Temperature: %s [%s] degrees Celsius\r\n", segBuf1, segBuf2);
#endif


        /* Read Accel data */
        RM_ICM42670_AccelRead (&g_icm42670_sensor0_ctrl, &raw_data);

        /* Calculate Accel data */
        RM_ICM42670_AccelDataCalculate (&g_icm42670_sensor0_ctrl, &raw_data,
&icm42670_accel_data);

        /* Output Accel data to console */
        printf ("                    \r\n");
        printf ("Acc_x:  %10.3f\r\n", icm42670_accel_data.accel_x);
        printf ("Acc_y:  %10.3f\r\n", icm42670_accel_data.accel_y);
        printf ("Acc_z:  %10.3f\r\n", icm42670_accel_data.accel_z);

#ifdef RTT_DEBUG_ON
        APP_PRINT ("\r\n");
        snprintf(segBuf1,sizeof(segBuf1)-1,"%10.3f",icm42670_accel_data.accel_x);
```

```
        APP_PRINT ("Acc_x:  %s\r\n", segBuf1);
        snprintf(segBuf1,sizeof(segBuf1)-1,"%10.3f",icm42670_accel_data.accel_y);
        APP_PRINT ("Acc_y:  %s\r\n", segBuf1);
        snprintf(segBuf1,sizeof(segBuf1)-1,"%10.3f",icm42670_accel_data.accel_z) ;
        APP_PRINT ("Acc_z:  %s\r\n", segBuf1);
#endif

        /* Read Gyro data */
        RM_ICM42670_GyroRead (&g_icm42670_sensor0_ctrl, &raw_data);
        /* Calculate Gyro data */
        RM_ICM42670_GyroDataCalculate (&g_icm42670_sensor0_ctrl, &raw_data,
&icm42670_gyro_data);

        /* Output Gyro data to console */
        printf ("                  \r\n");
        printf ("Gyro_x: %10.3f\r\n", icm42670_gyro_data.gyro_x);
        printf ("Gyro_y: %10.3f\r\n", icm42670_gyro_data.gyro_y);
        printf ("Gyro_z: %10.3f\r\n", icm42670_gyro_data.gyro_z);

#ifdef RTT_DEBUG_ON
        APP_PRINT ("\r\n");
        snprintf(segBuf1,sizeof(segBuf1)-1,"%10.3f",icm42670_gyro_data.gyro_x);
        APP_PRINT ("Gyro_x: %s\r\n", segBuf1);
        snprintf(segBuf1,sizeof(segBuf1)-1,"%10.3f",icm42670_gyro_data.gyro_y);
        APP_PRINT ("Gyro_y: %s\r\n", segBuf1);
        snprintf(segBuf1,sizeof(segBuf1)-1,"%10.3f",icm42670_gyro_data.gyro_z);
        APP_PRINT ("Gyro_z: %s\r\n", segBuf1);
#endif

        {
            static uint16_t mode = BSP_IO_LEVEL_HIGH ;
            mode = mode == BSP_IO_LEVEL_HIGH ?  BSP_IO_LEVEL_LOW : BSP_IO_LEVEL_HIGH ;
            R_BSP_PinWrite(LED1_GREEN, mode);
        }
#endif
    }

}

/*******************************************************************************************
*********************//**
 * 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. */
        IOPORT_CFG_OPEN (&IOPORT_CFG_CTRL, &IOPORT_CFG_NAME);
    }
}

#if BSP_TZ_SECURE_BUILD
```

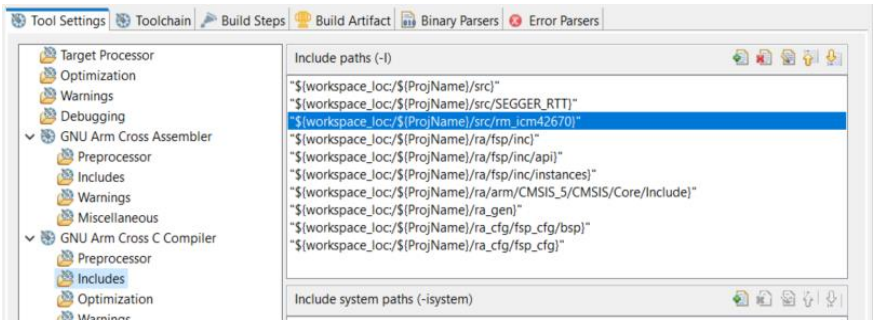
```
FSP_CPP_HEADER
BSP_CMSE_NONSECURE_ENTRY void template_nonsecure_callable ();

/* Trustzone Secure Projects require at least one nonsecure callable function in order to build
(Remove this if it is not required to build). */
BSP_CMSE_NONSECURE_ENTRY void template_nonsecure_callable ()
{

}
FSP_CPP_FOOTER

#endif
```

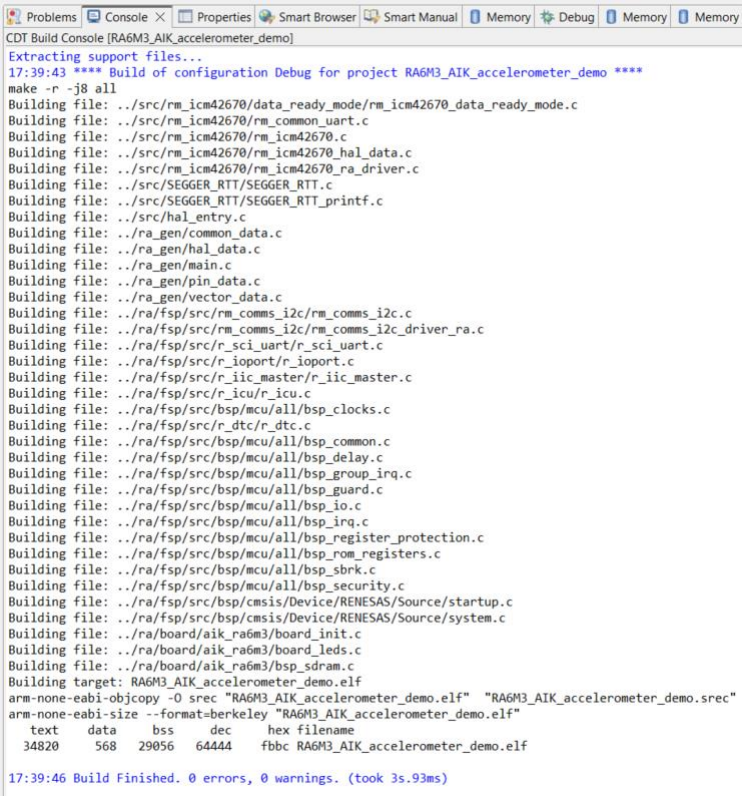
2.16 Right-click the project in the **Project Explorer** and select **Properties** form the context menu, then navigate to **C/C++ Build -> Settings**. Make sure you're on the tool **Setting -> GNU Arm Cross Linker -> Miscellaneous** tab and click on the **Use float with nano printf (-u_printf_float)** also change **Other Linker flags** field to **–specs=nosys.specs**.

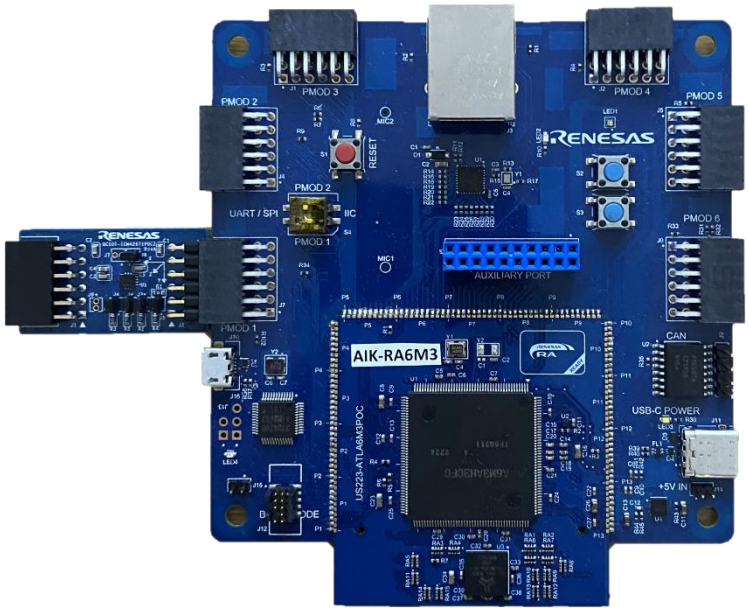
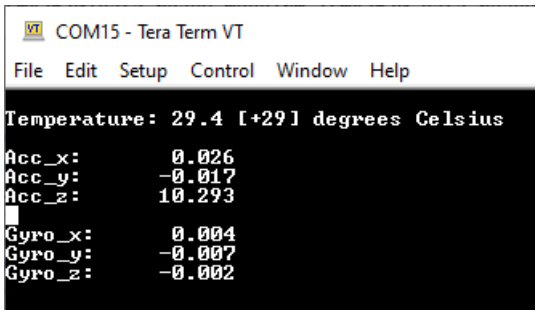


2.17 Navigate to **Setting -> GNU Arm Cross C Compiler -> includes** tab and click on the **Add** button to Include paths and add :

- **"${workspace_loc:/${ProjName}/src/SEGGER_RTT}"**
- **"${workspace_loc:/${ProjName}/src/rm_icm42670}"**



2.18	The project is now ready to compile. Press the “hammer” icon to start building the project. 
2.19	Once the build has finished, the **Console** pane in the lower-right corner of e^2 studio will report zero error and warnings: 
2.20	Connect PMOD5 Pin2 & Pin 3 with the USB2Serial TX & RX pins of the dongle respectively to enable UART output through Teraterm. **Pin** / **Signal/Bus SPI** / **Description UART** 2 / P900 / TXD 3 / P315 / RXD

2.21	Check that the Accelerometer is connected to PMOD1 as seen below. 
2.22	The application is now ready to be programmed and run on the AIK kit. Press the “bug” icon to begin the debug session. 
2.23	Bring up the serial terminal window to observe the debug output from the AIK kit.
2.24	Click the **Resume** button or press **F8** on the keyboard to start the application. Press **Resume** or **F8** again to resume the application and begin executing user code. 
2.25	Go back to the serial terminal window which should now be populated with debug output from the AIK kit. First line shows the temperature in degrees Celsius , the rest lines show the data of the accelerometers x,y,z axis and the data from the gyroscope x,y,z axis. 
2.26	Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session. 
2.27	Full project is also available in GitHub under https://github.com/renesas/aiot-ready/tree/master/kits/ra6m3_aik/qse

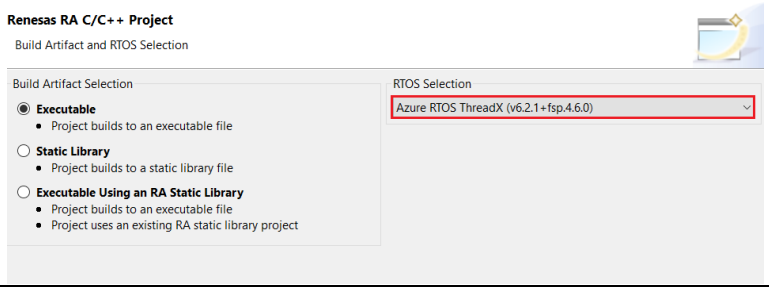
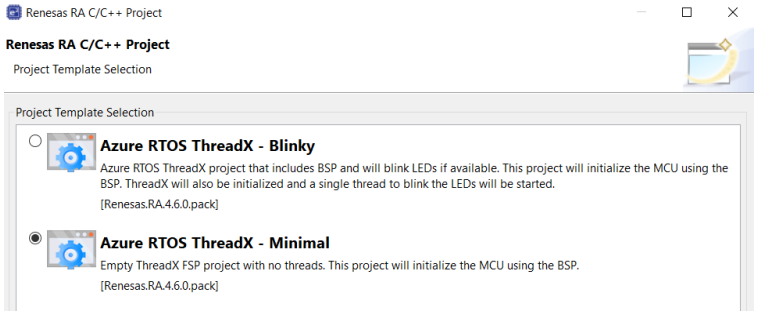
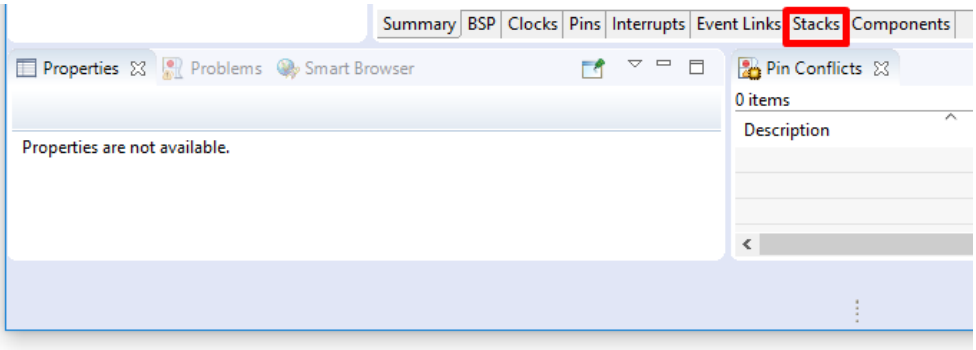
END OF SECTION

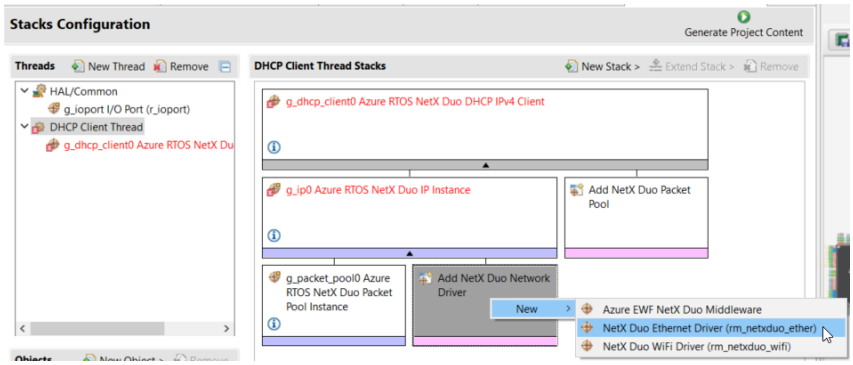
3 Implementing Ethernet demo

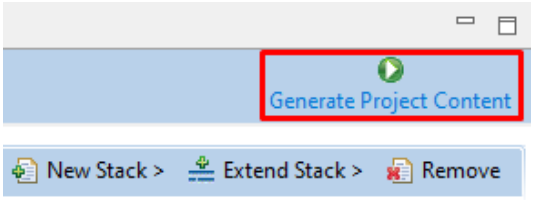
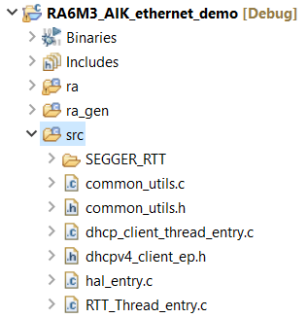
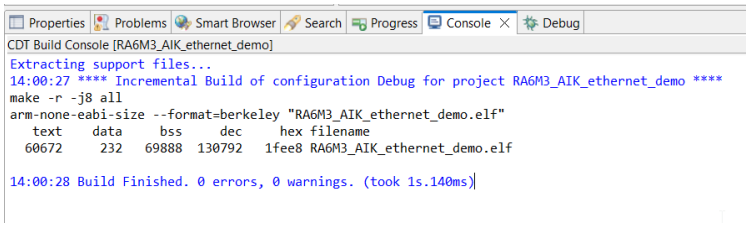
Overview

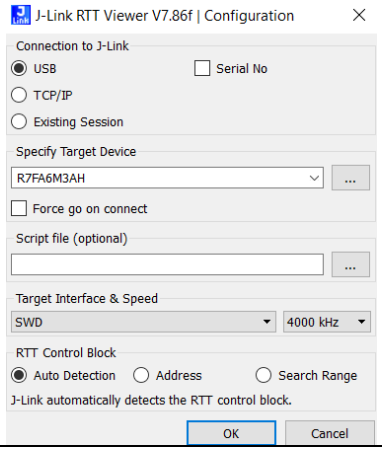
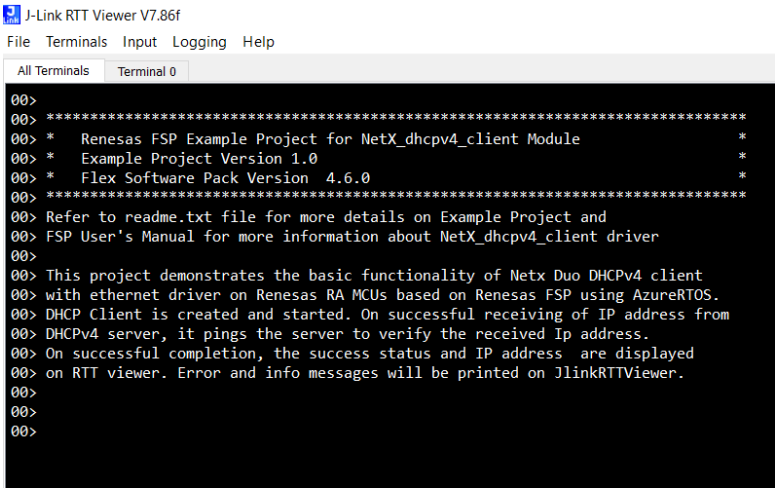
Following section describes in details steps required to set up an Ethernet demo project for AIK RA6M3 kit.

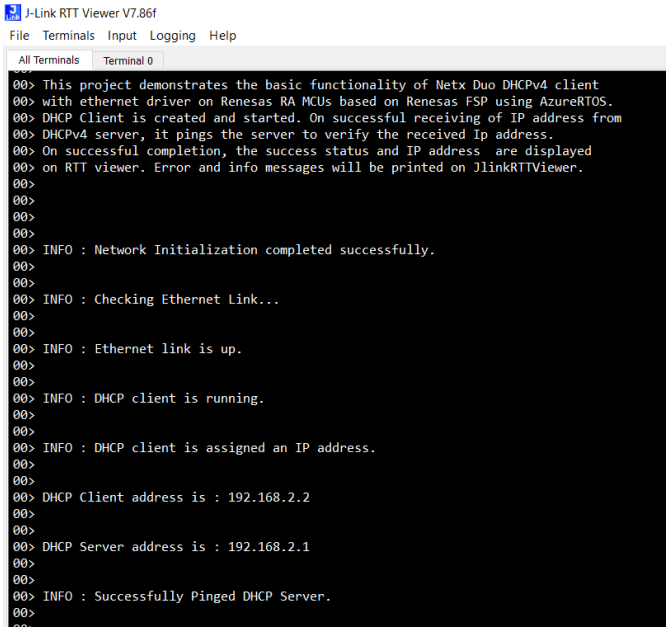
Procedural Steps

3.1	Prerequisites for this demo are: • Router with ethernet connection • Ethernet Cable
3.2	Create a new project and follow the steps described from 1.1 to 1.12.
3.3	On the next window, leave **Executable** and **Azure RTOS ThreadX** selected. Click **Next**.
3.4	When prompted to open the **FSP Configuration perspective**, click **Open Perspective**. The project is now set up to begin evaluation and development using the AIK kit. 
3.5	On the final page of the new project wizard select **Azure RTOS ThreadX – Minimal** and click **Finish**. 
3.6	Once new project is created, e^2 studio will switch to a layout optimized for developing Renesas RA projects. Select the **Stacks** tab at the bottom of the **FSP Configuration** pane visible in the middle. 
3.7	In **Threads** tab remove **Azure RTOS TreadX Port (rm_threadx_port).**

3.8	Access the **New Thread** menu again and select **New Thread**. Use **Properties** tab to configure following properties for this new module: • Thread -> Symbol dhcp_client_thread • Thread -> Name DHCP Client Thread • Thread -> Stack size (byte) 2048 • Thread -> Priority 3
3.9	Access the **New Stack** menu and select **Networking -> Azure RTOS NetX Duo DHCP IPv4 Client**. Use **Properties** tab to configure following properties for this new module: • DHCP -> Client -> IPv4 -> Persistent client state Enable • DHCP -> Client -> IPv4 -> Internal thread priority 1 • HTTP -> Client -> Minimum packet size (bytes) 300 • SNTP -> Client -> Maximum time adjustment allowed to local clock time (milliseconds) 10800000 • FTP -> Server -> Binary left shift as multiplier for next retry duration 1
3.10	In the Azure RTOS NetX Duo DHCP IPv4 Client module, press **Add NetX Duo Network Driver -> New -> NetX Duo Ethernet Driver (rm_netxduo_ether)**. 
3.11	In the Azure RTOS NetX Duo DHCP IPv4 Client module, press **Add NetX Duo Packet Pool -> Use -> g_packet_pool0 Azure RTOS NetX Duo Packet Pool Instance**.
3.12	In the **g_ether_phy0 Ethernet (r_ether_phy)** module. Use **Properties** tab to configure following properties for this new module: • Common -> ICS1894 target Enable • Module- g_ether_phy0 Ethernet (r_ether_phy) ->PHY-LSI Address 7
3.13	Access the **New Thread** menu again and select **New Thread**. Use **Properties** tab to configure following properties for this new module: • Thread -> Symbol rtt_thread • Thread -> Name RTT_Thread • Thread -> Priority 4
3.14	RA Configuration for this section is complete. Apply changes to the project source by clicking the **Generate Project Content** button in the top-right corner of the Configurator window. When prompted

	to *Proceed with save and generate*, tick the box next to **Always save and generate without asking** and click **Proceed**. 
3.15	The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the **Properties** tab.
3.16	In the **Project Explorer** pane, expand the **src** folder in the project, remove rtt_thread_entry.c file and add the following folders and files that can be found in the demo folder: • SEGGER_RTT • common_utils.h • common_utils.c • dhcp_client_thread_entry.c • dhcpv4_client_ep.h • RTT_Thread_entry.c
3.17	In the **Project Explorer** pane, expand the **src** folder in the project and open **hal_entry.c**. 
3.18	The project is now ready to compile. Press the “hammer” icon to start building the project. 
3.19	Once the build has finished, the **Console** pane in the lower-right corner of e² studio will report zero error and warnings: 
3.20	The application is now ready to be programmed and run on the AIK kit. Press the “bug” icon to begin the debug session. 

3.21	Bring up the J-Link RTT Viewer window to observe the debug output from the AIK kit. Go to File -> Connect and make the following changes: • Connection to J-Link USB • Specify Target Device R7FA6M3AH • Target interface & Speed SWD 4000 kHz 
3.22	Click the **Resume** button or press **F8** on the keyboard to start the application. Press **Resume** or **F8** again to resume the application and begin executing user code. 
3.23	Go back to the terminal window which should now be populated with debug output from the AIK kit. 

3.24	Insert the Ethernet cable to the AIK kit and the terminal prints the following information: • Network Initialization completed successfully. • Checking Ethernet Link... • Ethernet link is up. • DHCP client is running. • DHCP client is assigned an IP address • DHCP Client address is : 192.168.2.2 • DHCP Server address is : 192.168.2.1 • INFO : Successfully Pinged DHCP Server. **Note**: Values in DHCP Client/Server address may vary. 
3.25	Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session. 
3.26	Full project is also available in GitHub under https://github.com/renesas/aiot-ready/tree/master/kits/ra6m3_aik/qse

END OF SECTION