

Quick Setup Example on AIK-RA4E1 Solution Kit

Renesas Advanced (RA) Family – RA6 Series

Description

Welcome to Quick Setup Example for Renesas RA using AIK-RA4E1 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-RA4E1 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.

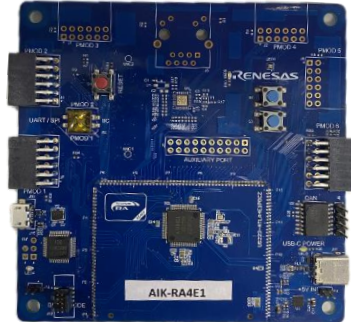




Objectives <ul style="list-style-type: none"> • Configure AIK-RA4E1 kit to run display with the basic operations project • Implement Accelerometer demo 	Prerequisites <ul style="list-style-type: none"> • Renesas AIK-RA4E1 VUI Solution Kit • Renesas Flexible Software Package 4.5.0 platform installation, which includes: <ul style="list-style-type: none"> • 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.
Skill Level <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 2 hours to complete

Workshop Sections

0	Setting up the hardware	2
1	Implementing Display with the basic operations demo	3
2	Implementing Accelerometer demo	17

0 Setting up the hardware

Procedural Steps

0.1	<p>To begin working with AIK-RA4E1 platform you will need the following:</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>AIK-RA4E1 Solution Kit</p> </div> <div style="text-align: center;">  <p>Display</p> </div> </div> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>Renesas ICM-42670-P PMOD Board</p> </div> <div style="text-align: center;">  <p>Display patch module</p> </div> <div style="text-align: center;">  <p>USB micro-B cable (included with the kit)</p> </div> </div>
0.2	<p>For the Accelerometer Demo a USB2Serial dongle will be needed.</p>
0.3	<p>Connect the AIK-RA4E1 kit to the PC using USB micro-B cable and J-Link OB USB port (J10) in the bottom left corner of the board.</p> <p>Connect Display patch module to PMOD 2 and the other end to the display. Also connect Renesas ICM-42670-P PMOD Board to PMOD1.</p>
0.4	<p>Verify that:</p> <ul style="list-style-type: none"> The blue LED2 (POWER) near the Renesas logo is on. The orange LED4 (LED OB) near the bottom edge of the board is on and not flashing.
0.5	<p>Your kit and operating environment are now set-up and ready for evaluation and development.</p>

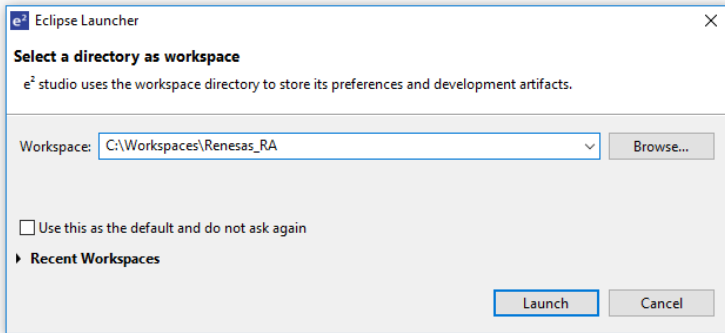
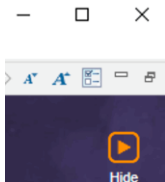
END OF SECTION

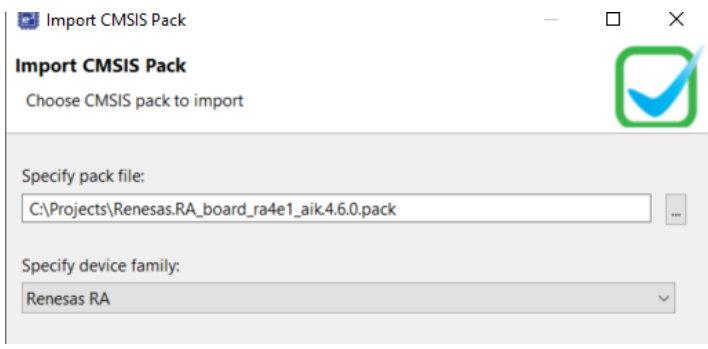
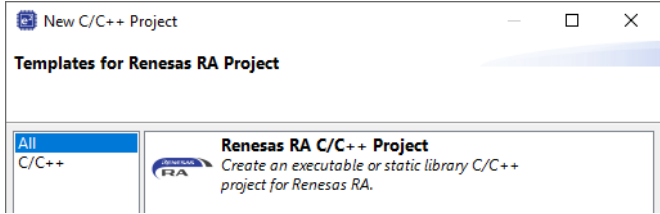
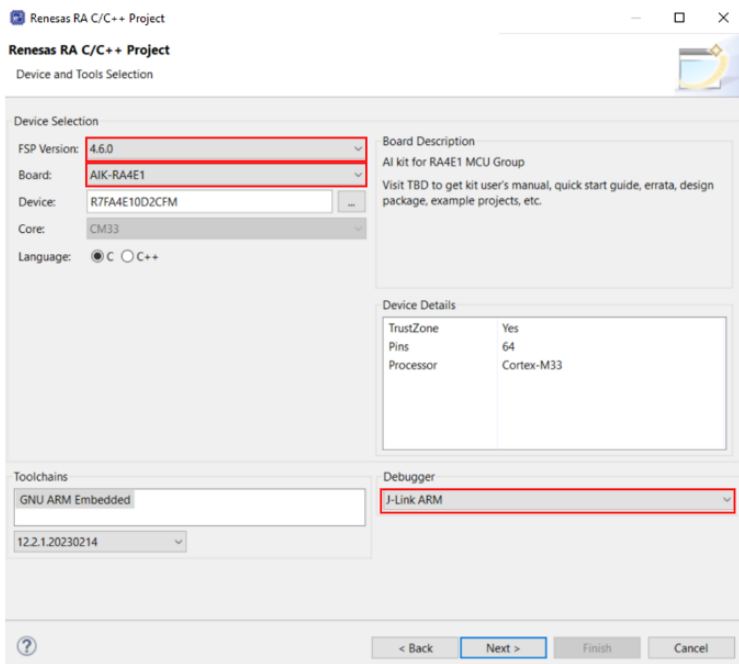
1 Implementing Display with the basic operations demo

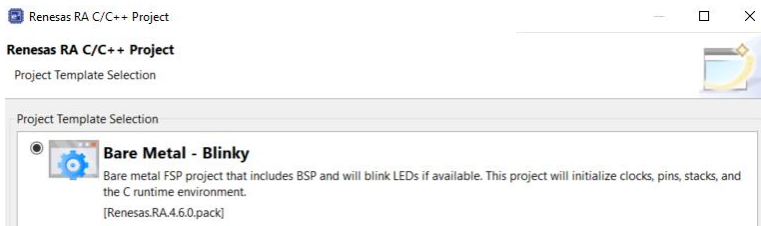
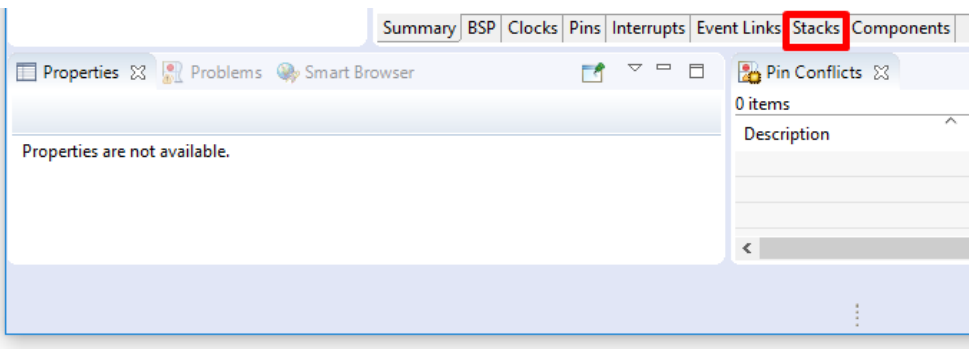
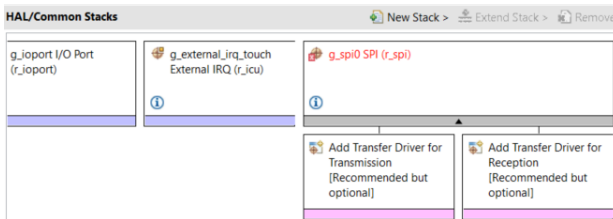
Overview

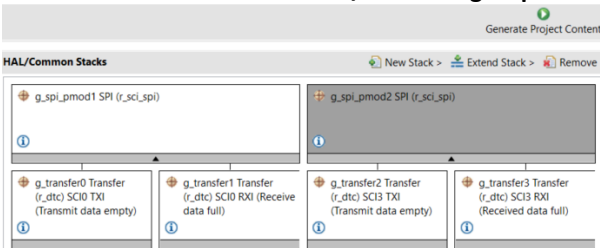
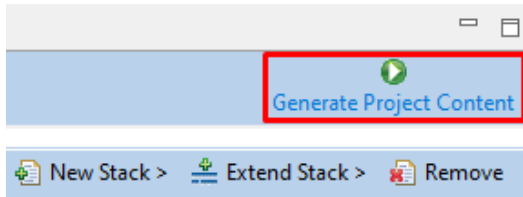
Following section describes in details steps required to create an e² studio workspace and set up a Display with basic operations-based project for AIK RA4E1 kit.

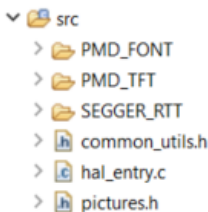
Procedural Steps

1.1	Launch e ² studio. e ² studio can be launched from the Windows start menu or directly from the installation folder. If you have multiple versions of e ² studio installed, please make sure to launch the version of e ² studio that was specified on the first page.
1.2	<p>In the Eclipse Launcher window, specify the destination for the new workspace. It is recommended to keep the path simple and avoid using spaces.</p> 
1.3	Click Launch to start e ² 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	<p>The welcome screen will show inside the new workspace. It can be dismissed by clicking on the Hide button in the top-right corner.</p> 
1.5	If you already have installed the BSP for AIK-RA4E1 kit, proceed directly to step 2.8. Otherwise, go to File -> Import and Select General -> CMSIS Pack .

1.6	<p>In the Import CMSIS Pack window, click ... to browse for the .pack file containing BSP for AIK-RA4E1 kit (Renesas.RA_board_RA4E1.<version>.pack). Select Renesas RA from the drop-down box under Specify device family and click Finish.</p> 
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	<p>In the new project wizard window, select Renesas RA C/C++ Project and click Next.</p> 
1.10	Specify a project name and Click Next .
1.11	<p>Select FSP version matching your BSP and FSP installation (e.g., 4.6.0) and set Board to AIK-RA4E1. Verify that the Debugger is set to J-Link ARM and click Next.</p> 
1.12	On the next window, leave Executable and No RTOS selected. Click Next .

1.13	<p>On the final page of the new project wizard select Bare Metal – Blinky and click Finish.</p> 
1.14	<p>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.</p>
1.15	<p>Once new project is created, e² 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.</p> 
1.16	<p>Access the New Stack menu again and select Input -> External IRQ (r_icu). Use Properties tab to configure following properties for this new module:</p> <ul style="list-style-type: none"> Name: g_external_irq_touch Channel: 5 Trigger: Failing Digital Filtering: Enable Callback: pmd_touch_irq_ep_callback Pins / IRQ5: P302
1.17	<p>Open the New Stack menu (near the top-left corner) and navigate to Connectivity -> SPI (r_sci_spi).</p>
1.18	<p>A SPI driver module will be added inside the HAL/Common group. DTC modules will need to be populated.</p> 

1.19	<p>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:</p> <ul style="list-style-type: none"> Name: g_spi_pmod1 Channel: 0 Receive interrupt Priority: Priority 2 Transmit buffer Empty Interrupt Priority: Priority 2 Transmit Complete Interrupt Priority: Priority 2 Error Interrupt Priority: Priority 2 Operating Mode: Master Callback: spi_pmod1_callback Bitrate: 15000000
1.20	Open the New Stack menu (near the top-left corner) and navigate to Connectivity -> SPI (r_spi) .
1.21	<p>A second SPI driver module will be added inside the HAL/Common group.</p> 
1.22	<p>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:</p> <ul style="list-style-type: none"> Name: g_spi_pmod2 Channel: 3 Receive interrupt Priority: Priority 2 Transmit buffer Empty Interrupt Priority: Priority 2 Transmit Complete Interrupt Priority: Priority 2 Error Interrupt Priority: Priority 2 Callback: sci_spi_pmod2_callback Bitrate: 15000000
1.23	<p>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 <i>Proceed with save and generate</i>, tick the box next to Always save and generate without asking and click Proceed.</p> 
1.24	The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the Properties tab.

1.25	<p>In the Project Explorer pane, expand the src folder in the project and add the following folders and files:</p> <ul style="list-style-type: none"> • PMD_FONT • PMD_TFT • SEGGER_RTT • common_utils.h • pictures.h
1.26	<p>In the Project Explorer pane, expand the src folder in the project and open hal_entry.c.</p> 
1.27	<p>hal_entry.c contains user application entry point (hal_entry function) for RTOS-less projects. The <code>R_BSP_WarmStart</code> callback is provided for the user to specify additional functions to be called during the FSP initialization sequence (e.g., pin configuration).</p>
1.28	<p>Add <code>#include</code> statement to include <code>common_utils.h</code>, <code>PMD_TFT/pmd_tft.h</code>, <code>PMD_TFT/pmd_text.h</code>, <code>pictures.h</code> near the top of the file.</p>
1.29	<p>Add static <code>uint16_t buffer_rgb565[RENESAS_110_17_WIDTH*RENESAS_110_17_HEIGHT + (RENESAS_110_17_WIDTH*RENESAS_110_17_HEIGHT +1)/2]</code>; after the includes. After <code>extern bsp_leds_t g_bsp_leds</code>; add <code>uint16_t act_state=0 ; uint16_t tft_ori_cnt=0</code>;</p>
1.30	<p>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.</p> <p>Following code can be used to completely replace contents of <code>hal_entry.c</code> to perform basic operations using the display for the AIK-RA6M3 board:</p> <pre> #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 </pre>

```

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 0x000000 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 0x000000 (no use)
            // --> color = 0xE7000000u

            tft_blit_copy_blend ((uint16_t*) buffer_rgb565, (uint16_t*)
picture_renesas_110_17_rgb565,

```

```

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

    break;

    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 0x000000 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


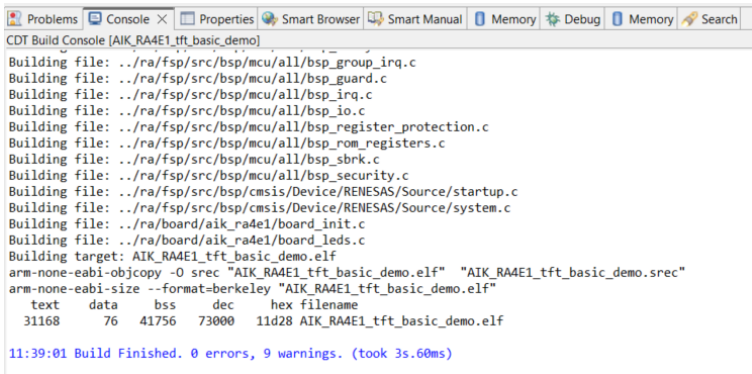
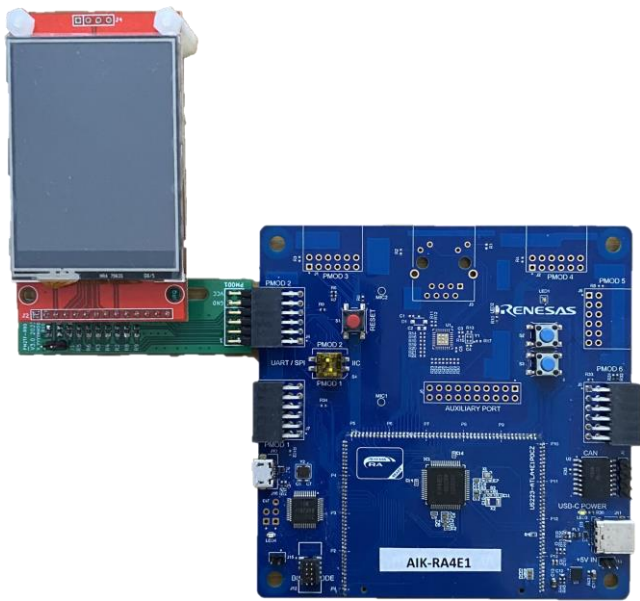

        FSP_CPP_HEADER
        BSP_CMSE_NONSECURE_ENTRY void template_nonsecure_callable ();

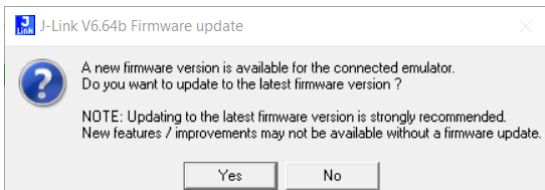
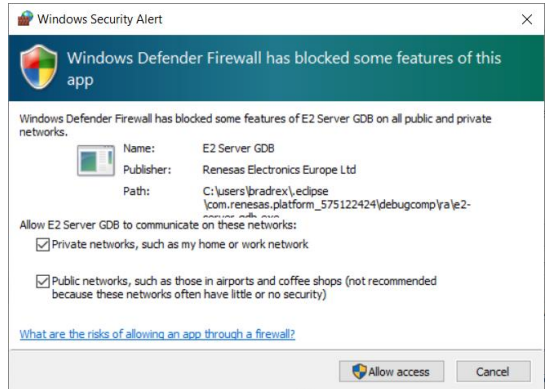

        /* Trustzone Secure Projects require at least one nonsecure callable function in order to build
(Remove this if it is not required to build). */
        BSP_CMSE_NONSECURE_ENTRY void template_nonsecure_callable ()
        {

        }

        FSP_CPP_FOOTER

```

	#endif
1.31	<p>The project is now ready to compile. Press the “hammer” icon to start building the project.</p> 
1.32	<p>Once the build has finished, the Console pane in the lower-right corner of e² studio will report zero errors :</p> 
1.33	<p>Check that the Display is connected to PMOD2 as seen below.</p> 
1.34	<p>The application is now ready to be programmed and run on the AIK kit. Press the “bug” icon to begin the debug session.</p> 

1.35	<p>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.</p>  <p>A screenshot of a 'J-Link V6.64b Firmware update' dialog box. It contains a question mark icon and the text: 'A new firmware version is available for the connected emulator. Do you want to update to the latest firmware version?'. Below this is a note: 'NOTE: Updating to the latest firmware version is strongly recommended. New features / improvements may not be available without a firmware update.' At the bottom are 'Yes' and 'No' buttons.</p>
1.36	<p>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.</p>  <p>A screenshot of a 'Windows Security Alert' dialog box. The title bar says 'Windows Security Alert'. The main text says: 'Windows Defender Firewall has blocked some features of this app'. Below this, it says: 'Windows Defender Firewall has blocked some features of E2 Server GDB on all public and private networks.' It lists details: Name: E2 Server GDB, Publisher: Renesas Electronics Europe Ltd, Path: C:\users\bradrex\eclipse\com.renesas.platform_575122424\debugcomp\ja\e2-server-gdb.exe. It then asks to 'Allow E2 Server GDB to communicate on these networks:' with two checkboxes: 'Private networks, such as my home or work network' (checked) and 'Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)'. At the bottom are 'Allow access' and 'Cancel' buttons.</p>
1.37	<p>e² studio will perform flash programming routines and prompt to switch to Debug perspective. Select the check box by Remember my decision and click Switch.</p>
1.38	<p>The debug session is now started, and the application is paused at its entry function (<code>SystemInit()</code> in <code>Reset_Handler</code>). At this point, you can set up additional debug features such as variable and expressions views before the program is executed.</p>
1.39	<p>Click the Resume button or press F8 on the keyboard to start the application.</p>  <p>A small icon of a green play button with a yellow bar, representing the 'Resume' button in a debugger.</p>
1.40	<p>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.</p>

1.41	<p>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 below.</p> <div data-bbox="454 347 1220 1265"> </div>
1.42	<p>Click the Terminate button or press Ctrl + F2 on the keyboard to stop the application and terminate the debug session.</p> <div data-bbox="798 1377 869 1444"> </div>

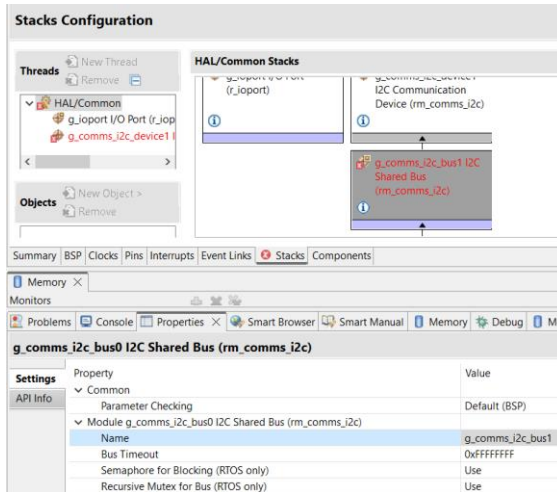
END OF SECTION

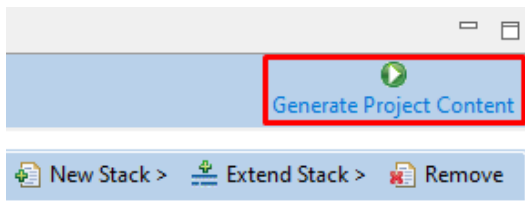
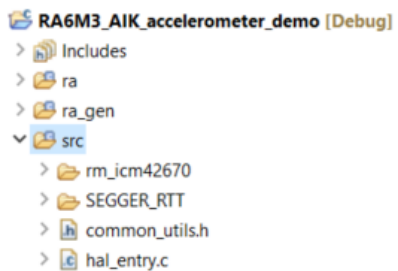
2 Implementing Accelerometer demo

Overview

Following section describes in details steps required to set up an accelerometer demo project for AIK RA4E1 kit.

Procedural Steps

2.1	Create a new project and follow the steps described from 1.1 to 1.15.
2.2	<p>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:</p> <ul style="list-style-type: none"> Name: g_comms_i2c_device1 Slave Address: 0x68 Callback: rm_icm42670_comms_i2c_callback
2.3	<p>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:</p> <ul style="list-style-type: none"> General -> Name: g_comms_i2c_bus1 
2.4	<p>In the I2C Communication Device module, in Add I2C Communications peripherals press on the icon -> New and choose I2C Master (r_sci_master).</p> <p>Use Properties tab to configure following properties for this new module:</p> <ul style="list-style-type: none"> Common -> DTC on Transmission and Reception: Enabled Name: g_i2c_master1 Channel: 4 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	<p>Access the New Stack menu again and select Input -> External IRQ (r_icu). Use Properties tab to configure following properties for this new module:</p> <ul style="list-style-type: none"> Name g_external_irq6_pmod1 Channel 1
2.7	<p>Access the New Stack menu again and select Connectivity -> UART (r_sci_uart). Use Properties tab to configure following properties for this new module:</p> <ul style="list-style-type: none"> Common -> DTC Support Enable General -> Name g_uart3_pmod2 General -> Channel 3 Baud -> Max Error (%) 5 Interrupts -> Callback rm_uart_callback
2.8	<p>In the UART module, in Add DTC Driver for Transmission [optional] press on the icon -> New and choose Transfer (r_dtc).</p>
2.9	<p>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 <i>Proceed with save and generate</i>, tick the box next to Always save and generate without asking and click Proceed.</p> 
2.10	<p>The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the Properties tab.</p>
2.11	<p>In the Project Explorer pane, expand the src folder in the project and add the following folders and files:</p> <ul style="list-style-type: none"> rm_icm42670 SEGGER_RTT common_utils.h
2.12	<p>In the Project Explorer pane, expand the src folder in the project and open hal_entry.c.</p> 
2.13	<p>hal_entry.c contains user application entry point (hal_entry function) for RTOS-less projects. The <code>R_BSP_WarmStart</code> callback is provided for the user to specify additional functions to be called during the FSP initialization sequence (e.g., pin configuration).</p>

2.14	<p>Athe the beginning of hal_entry.c before "void R_BSP_WarmStart(bsp_warm_start_event_t event);" add the #include statement for the following:</p> <ul style="list-style-type: none"> • #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	<p>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-RA8M3 board:</p> <pre> #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 ; 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 } #endif } </pre>

```

/* 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

}

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

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)
    {
#ifdef 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
#ifdef 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
}

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. */

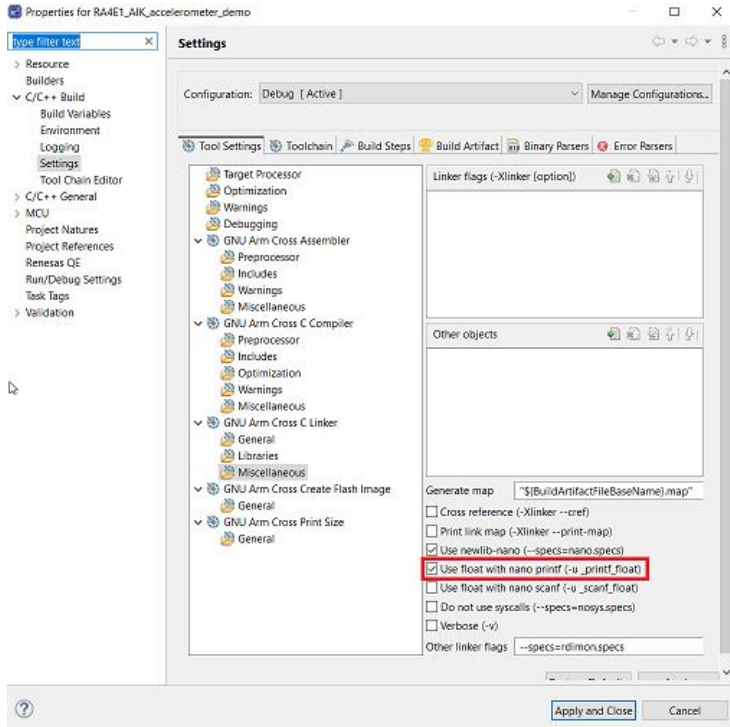
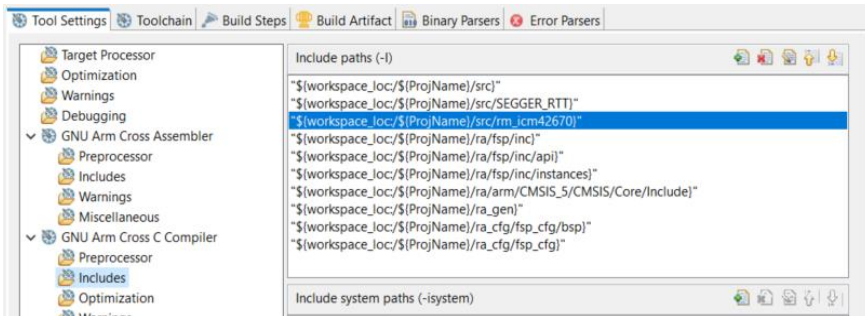

            /* Configure pins. */
            R_IOPORT_Open (&g_ioport_ctrl, &IOPORT_CFG_NAME);
        }
    }

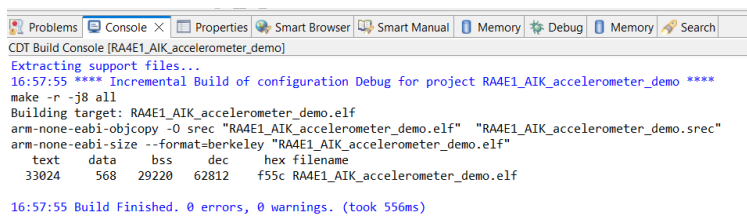
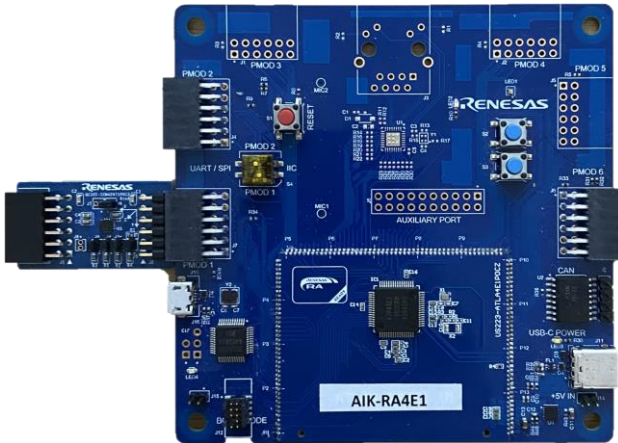


    #if BSP_TZ_SECURE_BUILD

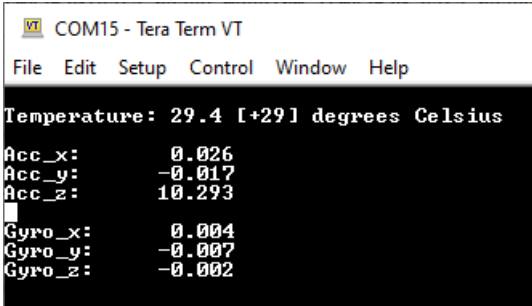

        FSP_CPP_HEADER
        BSP_CMSE_NONSECURE_ENTRY void template_nonsecure_callable ();

        /* 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 ()
        {
    }
}

```

	<p>FSP_CPP_FOOTER</p> <p>#endif</p>
2.16	<p>Right-click the project in the Project Explorer and select Properties from 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).</p> 
2.17	<p>Navigate to Setting -> GNU Arm Cross C Compiler -> includes tab and click on the Add button to Include paths and add :</p> <ul style="list-style-type: none"> • <code>"\${workspace_loc}/\${ProjName}/src/SEGGER_RTT"</code> • <code>"\${workspace_loc}/\${ProjName}/src/rm_icm42670"</code> 
2.18	<p>The project is now ready to compile. Press the "hammer" icon to start building the project.</p> 

2.19	<p>Once the build has finished, the Console pane in the lower-right corner of e² studio will report zero error and warnings:</p> <div></div>									
2.20	<p>Connect PMOD2 Pin2 & Pin 3 with the USB2Serial TX & RX pins of the dongle respectively to enable UART output through Teraterm.</p> <table><tr><th>Pin</th><th>Signal/Bus SPI</th><th>Description UART</th></tr><tr><td>2</td><td>P707</td><td>TXD</td></tr><tr><td>3</td><td>P706</td><td>RXD</td></tr></table>	Pin	Signal/Bus SPI	Description UART	2	P707	TXD	3	P706	RXD
Pin	Signal/Bus SPI	Description UART								
2	P707	TXD								
3	P706	RXD								
2.21	<p>Check that the Accelerometer is connected to PMOD1 as seen below.</p> <div></div>									
2.22	<p>The application is now ready to be programmed and run on the AIK kit. Press the “bug” icon to begin the debug session.</p> <div></div>									
2.23	<p>Bring up the serial terminal window to observe the debug output from the AIK kit.</p>									
2.24	<p>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.</p> <div></div>									

2.25	<p>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.</p> 
2.26	<p>Click the Terminate button or press Ctrl + F2 on the keyboard to stop the application and terminate the debug session.</p> 

END OF SECTION

