Quick Setup Example on VK-RA8M1 Solution Kit

Renesas Advanced (RA) Family – RA8 Series

Description

Welcome to Quick Setup Example for Renesas RA using VK-RA8M1 Solution Kit! The objective of this workshop is to build a basic Renesas RA application utilizing Renesas tools.

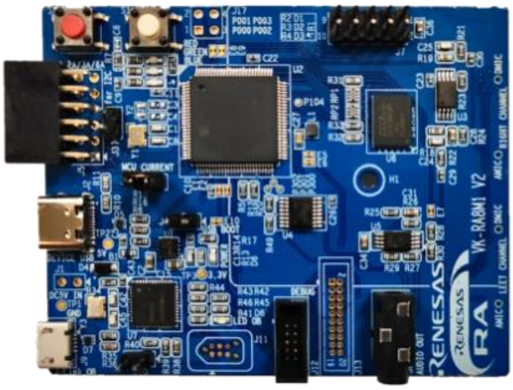
The applications used in this lab is built to run on VK-RA8M1 Solution Kit. A foundation project will be created from scratch and populated with several HAL drivers provided by the Flexible Software Package (FSP).

Objectives • Configure RA8M1-VK to run UART demo • Implement AMIC demo • Implement DMIC demo • Implement DAC demo	Prerequisites • Renesas VK-RA8M1 VUI Solution Kit • Renesas Flexible Software Package 5.2.0 platform installation, which includes: • e^2 studio 2024-1 or newer • FSP 5.2.0 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)
Skill Level • Basic familiarity with embedded electronics • Basic understanding of C language • Understanding of how to import projects into e^2 studio (optional – for use with ready checkpoint projects).	Time • 3 hours to complete

Workshop Sections

0 Setting up the hardware

Procedural Steps

0.1	To begin working with VK-RA8M1 platform you will need the following:  VK-RA8M1 Solution Kit MIC-Board USB micro-B cable (included with the kit)
0.2	Connect the VK-RA8M1 kit to the PC using USB micro-B cable and J-Link OB USB port (J9) in the bottom left corner of the board. When advised so, connect MIC Board J14 and the other end to the J7 of the motherboard, before powering up the solution kit.
0.3	Verify that: • The blue LED (POWER) is on. • The Yellow LED (LED OB) is on and not flashing.
0.4	Your kit and operating environment are now set-up and ready for evaluation and development.

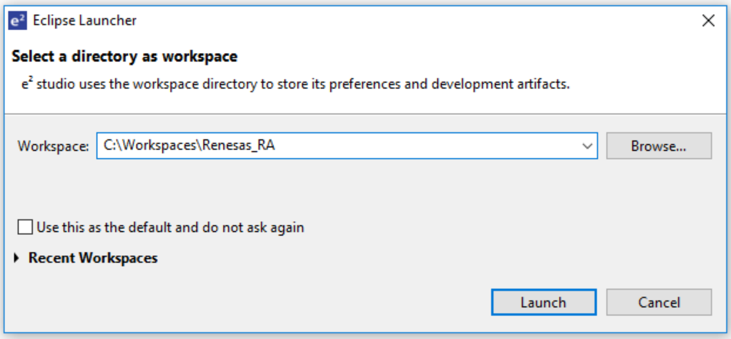
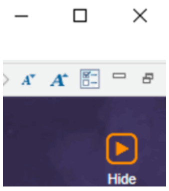
END OF SECTION

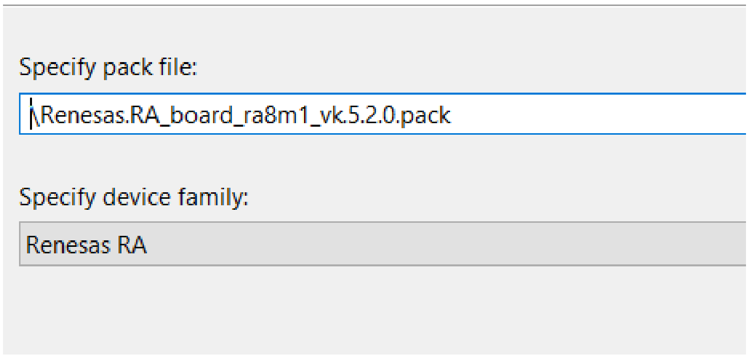
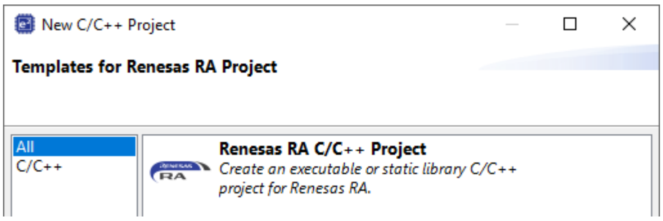
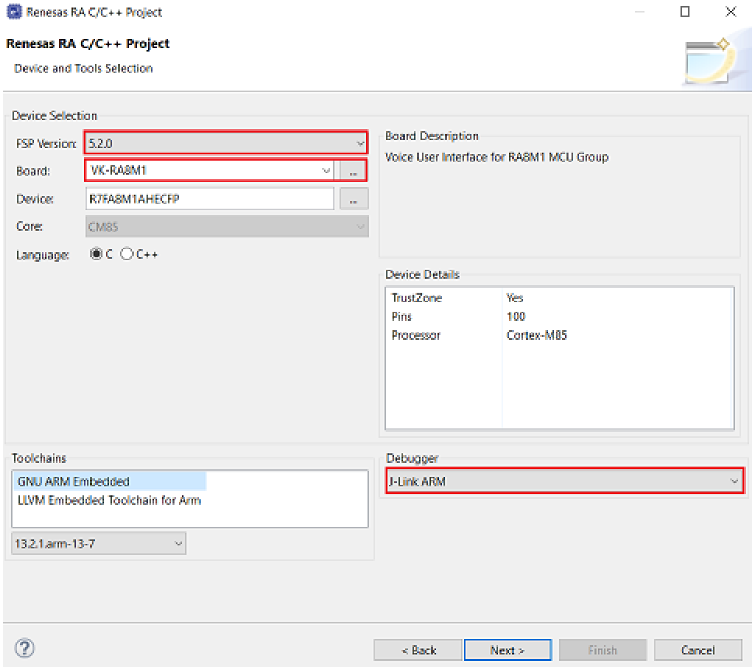
1 Implementing UART demo

Overview

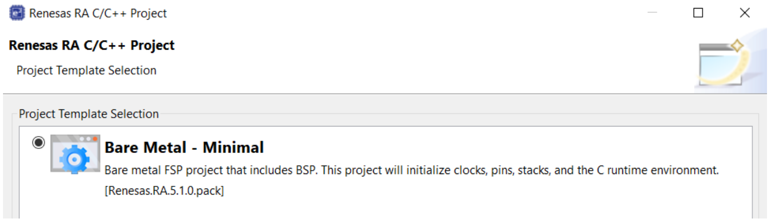
The following section describes in details steps required to create an e^2 studio workspace with basic operations-based project for RA8M1Voice Kit.

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	In the Eclipse Launcher window, specify the destination for the new workspace. It is recommended to keep the path simple and avoid using spaces.  Eclipse Launcher **Select a directory as workspace** e^2 studio uses the workspace directory to store its preferences and development artifacts. Workspace: C:\Workspaces\Renesas_RA Browse... ☐ Use this as the default and do not ask again ▸ **Recent Workspaces** Launch Cancel
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	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 VK-RA8M1 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 VK-RA8M1 kit (Renesas.RA_board_RA8M3.<version>.pack). Select **Renesas RA** from the drop-down box under Specify device family and click **Finish**.  Specify pack file: \Renesas.RA_board_ra8m1_vk.5.2.0.pack Specify device family: Renesas RA
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**.  New C/C++ Project **Templates for Renesas RA Project** All C/C++ **Renesas RA C/C++ Project** *Create an executable or static library C/C++ project for Renesas RA.*
1.10	Specify a project name and Click **Next**.
1.11	Select FSP version matching your BSP and FSP installation (e.g., **5.1.0**) and set Board to **VK-RA8M1**. Verify that the Debugger is set to **J-Link ARM** and click **Next**.  Renesas RA C/C++ Project **Renesas RA C/C++ Project** Device and Tools Selection Device Selection FSP Version: 5.2.0 Board: VK-RA8M1 Device: R7FA8M1AHECFP Core: CM85 Language: ● C ○ C++ Board Description Voice User Interface for RA8M1 MCU Group Device Details TrustZone Yes Pins 100 Processor Cortex-M85 Toolchains GNU ARM Embedded LLVM Embedded Toolchain for Arm 13.2.1.arm-13-7 Debugger J-Link ARM < Back Next > Finish Cancel
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** and click **Finish**.



Renesas RA C/C++ Project

Renesas RA C/C++ Project

Project Template Selection

Project Template Selection

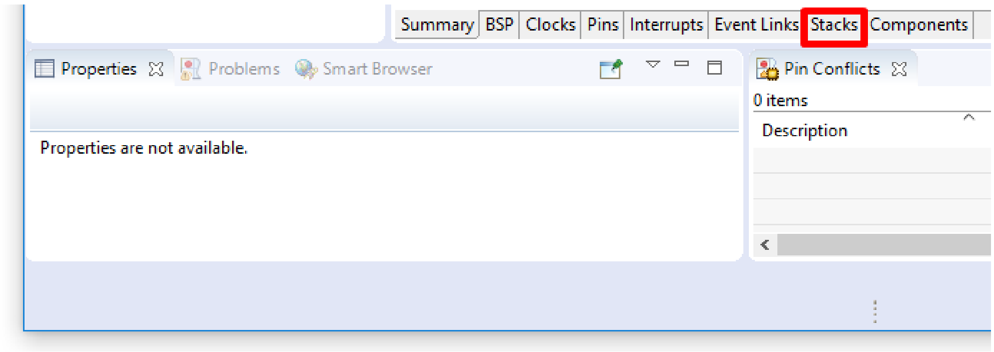
● **Bare Metal - Minimal**

Bare metal FSP project that includes BSP. This project will initialize clocks, pins, stacks, and the C runtime environment.

[Renesas.RA.5.1.0.pack]

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 Voice 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.

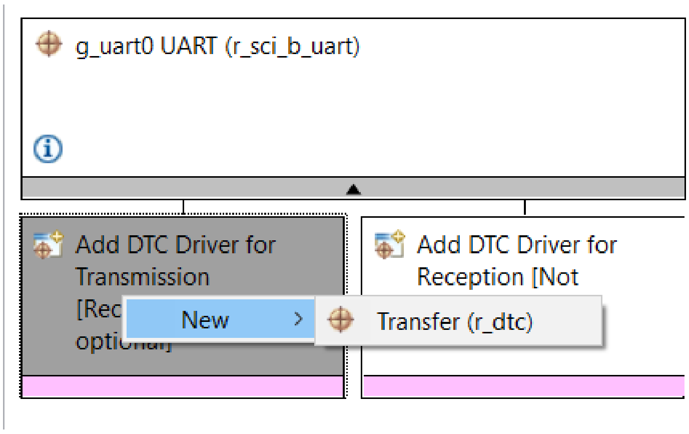


Properties are not available.

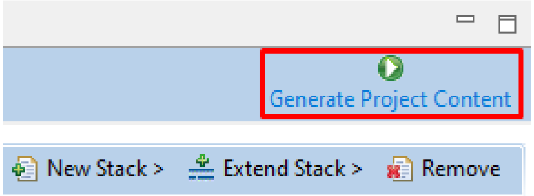
1.16 Access the **New Stack** menu again and select **Connectivity > g_uart_ds (r_sci_b_uart)**. Use **Properties** tab to configure following properties for this new module:

Common→ FIFO Support	Enable
Common→ DTC Support	Enable
General→ Name	g_uart0
General→ Channel	0
Baud→Baud Rate	460800
Interrupts→Callback	g_uart0_cb
Pins→RXD0	P610
Pins→TXD0	P609

1.17 In the **g_uart_ds (r_sci_b_uart) Add DTC Driver**.

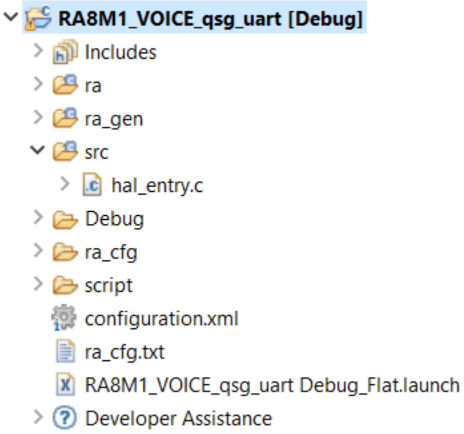

g_uart0 UART (r_sci_b_uart)
Add DTC Driver for Transmission [Rec optional]
Add DTC Driver for Reception [Not
New
Transfer (r_dtc)

1.18 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**.



1.19 The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the **Properties** tab.

1.20 In the **Project Explorer** pane, expand the **src** folder in the project and open **hal_entry.c**.



1.21 **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).

1.22 **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.

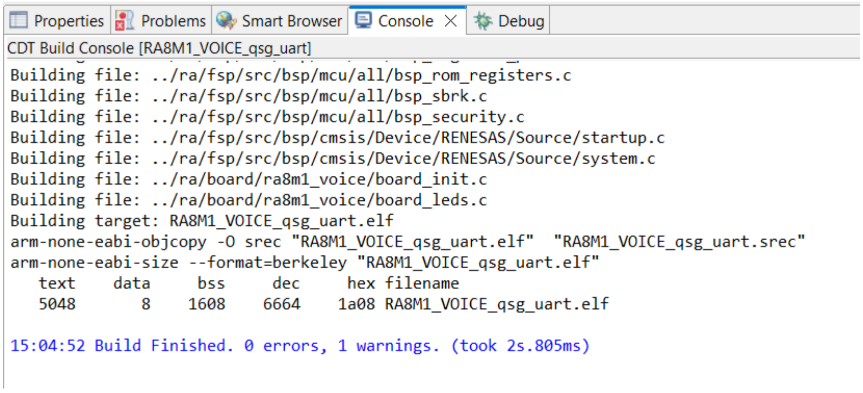
Following code can be used to completely replace contents of hal_entry.c to perform basic operations using the display for the VK-RA8M1 board:

```
#include "hal_data.h"
#include "stdio.h"

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

static volatile bool uart_done;
static volatile char uart_rec;

void hal_entry(void)
{
    fsp_err_t err;

    /* Initialize SCI peripheral in UART mode */
    err = R_SCI_B_UART_Open(&g_uart0_ctrl, &g_uart0_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Perform UART write */
    err = R_SCI_B_UART_Write(&g_uart0_ctrl, (void *) "Hello from Renesas
VOICE kit\r\n", 30);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Wait for interrupt & check for completion */
    while (false == uart_done)
        __WFI();

    uart_done = false;

    while (1)
    {
        /* Wait for interrupt & check for received data */
        while ('\0' == uart_rec)
            __WFI();

        char text_buf[32] = {0};
        snprintf(text_buf, 32, "Received character: '%c'\r\n", uart_rec);

        uart_rec = '\0';

        /* Perform UART write */
        err = R_SCI_B_UART_Write(&g_uart0_ctrl, (void *) text_buf,
strlen(text_buf));
        if (FSP_SUCCESS != err)
        {
            __BKPT(0);
        }

        /* Wait for interrupt & check for completion */
        while (false == uart_done)
            __WFI();

        uart_done = false;
    }
}

void g_uart0_cb(uart_callback_args_t * p_args)
{
    if (UART_EVENT_TX_COMPLETE == p_args->event)
    {
        uart_done = true;
    }

    else if (UART_EVENT_RX_CHAR == p_args->event)
    {
        uart_rec = (char) p_args->data;
    }

    else
    {}
}
```

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

        /* Configure pins. */
        R_IOPORT_Open (&g_ioport_ctrl, g_ioport.p_cfg);
    }
}
```

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

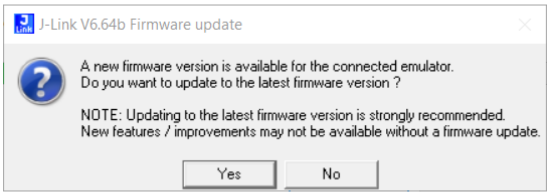
1.24 Once the build has finished, the **Console** pane in the lower-right corner of e^2 studio will report zero errors :



```
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_rom_registers.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_sbrk.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_security.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/system.c
Building file: ../ra/board/ra8m1_voice/board_init.c
Building file: ../ra/board/ra8m1_voice/board_leds.c
Building target: RA8M1_VOICE_qsg_uart.elf
arm-none-eabi-objcopy -O srec "RA8M1_VOICE_qsg_uart.elf"  "RA8M1_VOICE_qsg_uart.srec"
arm-none-eabi-size --format=berkeley "RA8M1_VOICE_qsg_uart.elf"
   text    data     bss     dec     hex filename
   5048       8    1608    6664    1a08 RA8M1_VOICE_qsg_uart.elf

15:04:52 Build Finished. 0 errors, 1 warnings. (took 2s.805ms)
```

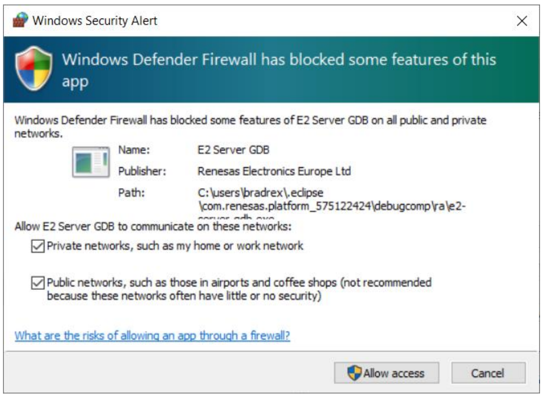
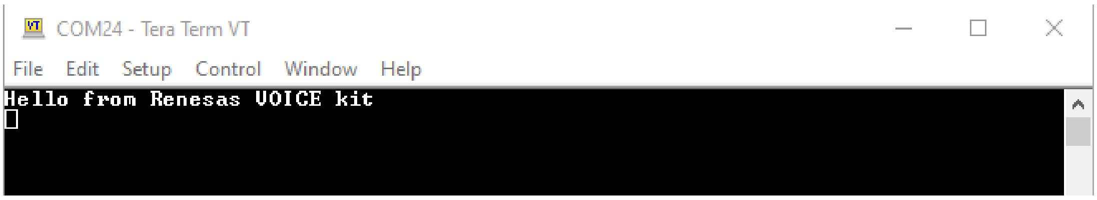
1.25 The application is now ready to be programmed and run on the Voice kit. Press the “bug” icon to begin the debug session.

1.26 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.


J-Link V6.64b Firmware update

A new firmware version is available for the connected emulator.
Do you want to update to the latest firmware version ?

NOTE: Updating to the latest firmware version is strongly recommended.
New features / improvements may not be available without a firmware update.

1.27	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.  Windows Defender Firewall has blocked some features of this app Windows Defender Firewall has blocked some features of E2 Server GDB on all public and private networks. Name: E2 Server GDB Publisher: Renesas Electronics Europe Ltd Path: C:\users\bradrex\.eclipse \com.renesas.platform_575122424\debugcomp\ra\e2- Allow E2 Server GDB to communicate on these networks: ☑ Private networks, such as my home or work network ☑ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security) What are the risks of allowing an app through a firewall?
1.28	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**.
1.29	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.
1.30	Click the Resume button or press F8 on the keyboard to start the application.
1.31	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.
1.32	Go to Serial Terminal and observe the printed message.  `Hello from Renesas VOICE kit`
1.33	Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session.

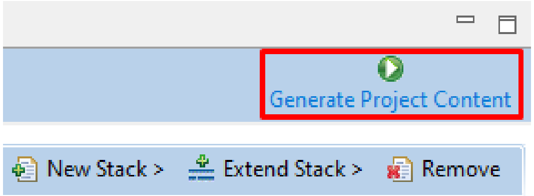
END OF SECTION

2 Implementing AMIC demo

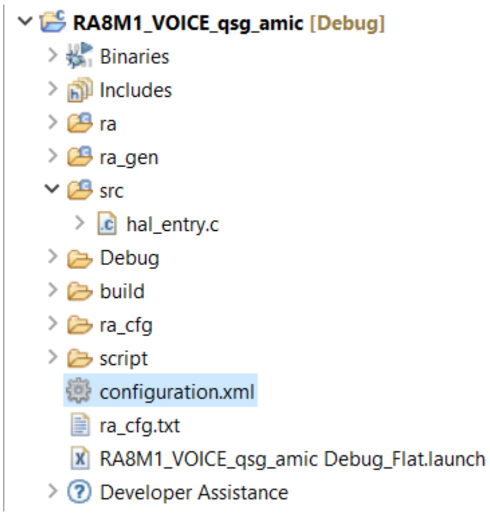
Overview

Following section describes in details steps required to set up an analog microphone demo project for RA8M1 Voice 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 **Timers -> Timer, General PWM (r_gpt)**. Use **Properties** tab to configure following properties for this new module: • Channel 2 • Period 16000 • Period Unit Hertz
2.3	Access the **New Stack** menu again and select **Analog -> ADC (r_adc)**. Use **Properties** tab to configure following properties for this new module: • Input, Channel & Channel 1 enable • Interrupts, Normal/Group A Trigger GPT2 COUNTER OVERFLOW (Overflow) • AN000 P004 • AN001 P005
2.4	Access the **New Stack** menu again and select **System -> Event Link Controler (r_elc)**. Use **Properties** tab to configure following properties for this new module: • Name g_elc
2.5	Access the **New Stack** menu again and select **Transfer -> Transfer (r_dmac)**. Use **Properties** tab to configure following properties for this new module: • Transfer Size 4 • Destination Address Mode Incremented • Activation Source ADC0 SCAN END (End of A/D scanning operation) • Callback g_transfer0_cb • Transfer End Interrupt Priority Priority 11
2.6	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**.  Generate Project Content New Stack > Extend Stack > Remove
2.7	The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the **Properties** tab.

2.8 In the **Project Explorer** pane, expand the **src** folder in the project and open **hal_entry.c**.



```
RA8M1_VOICE_qsg_amic [Debug]
  Binaries
  Includes
  ra
  ra_gen
  src
    hal_entry.c
  Debug
  build
  ra_cfg
  script
  configuration.xml
  ra_cfg.txt
  RA8M1_VOICE_qsg_amic Debug_Flat.launch
  Developer Assistance
```

2.9 **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).

2.10 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 enable the analog microphones for the VK-RA8M1 board:

```
#include "hal_data.h"

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

#define AMIC_BUF_SIZE   (8000)

static uint32_t amic_buf[2][AMIC_BUF_SIZE];
static volatile uint8_t amic_idx;

static volatile bool amic_done;

void hal_entry(void)
{
    fsp_err_t err;

    /* Initialize ELC peripheral */
    err = R_ELC_Open(&g_elc_ctrl, &g_elc_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Enabled configured ELC links */
    err = R_ELC_Enable(&g_elc_ctrl);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Initialize the ADC peripheral */
    err = R_ADC_Open(&g_adc0_ctrl, &g_adc0_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Enable ADC scanning on microphone channels */
    err = R_ADC_ScanCfg(&g_adc0_ctrl, &g_adc0_channel_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Enable ADC scanning */
    err = R_ADC_ScanStart(&g_adc0_ctrl);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Initialize the DMA peripheral */
    err = R_DMAC_Open(&g_transfer0_ctrl, &g_transfer0_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Set the DMA to capture from ADC registers into amic_buf */
    err = R_DMAC_Reset(&g_transfer0_ctrl, (void *) R_ADC0->ADDR,
amic_buf[amic_idx], AMIC_BUF_SIZE);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Initialize timer used to limit the sampling rate */
    err = R_GPT_Open(&g_timer0_ctrl, &g_timer0_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Start the timer */
    err = R_GPT_Start(&g_timer0_ctrl);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    while (1)
    {
        /* Wait for interrupt & check for event */
        while (false == amic_done)
            __WFI();

        amic_done = false;

        /** Data in amic_buf[amic_idx ^ 1] can be used at this point */

        /* Toggle green LED to indicate buffer received */
```

<table>
<tr><td></td><td><pre>
        bsp_io_level_t level;
        R_IOPORT_PinRead(&g_ioport_ctrl, BSP_IO_PORT_04_PIN_00, &level);
        R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_04_PIN_00, !level);
    }
}

void g_transfer0_cb(dmac_callback_args_t * p_args)
{
    /* Change index of the active write buffer */
    amic_idx ^= 1;

    /* Start subsequent ADC capture */
    R_DMAC_Reset(&g_transfer0_ctrl, (void *) R_ADC0->ADDR,
amic_buf[amic_idx], AMIC_BUF_SIZE);

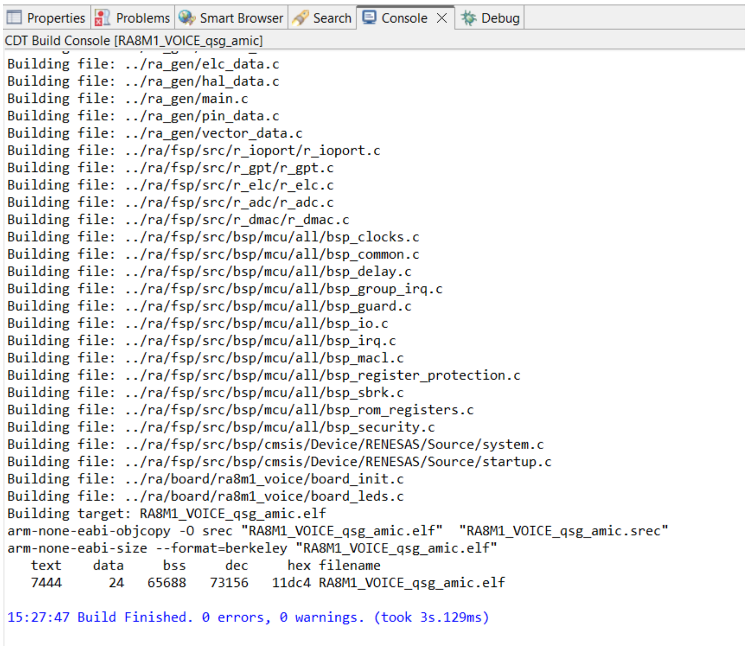
    amic_done = true;

    /* Suppress compiler warning for unused p_args */
    FSP_PARAMETER_NOT_USED(p_args);
}

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

        /* Configure pins. */
        R_IOPORT_Open (&g_ioport_ctrl, g_ioport.p_cfg);
    }
}
</pre></td></tr>
<tr><td>2.11</td><td>The project is now ready to compile. Press the “hammer” icon to start building the project.</td></tr>
</table>

2.12 Once the build has finished, the **Console** pane in the lower-right corner of e^2 studio will report zero error and warnings:



```
Building file: ../ra_gen/elc_data.c
Building file: ../ra_gen/hal_data.c
Building file: ../ra_gen/main.c
Building file: ../ra_gen/pin_data.c
Building file: ../ra_gen/vector_data.c
Building file: ../ra/fsp/src/r_ioport/r_ioport.c
Building file: ../ra/fsp/src/r_gpt/r_gpt.c
Building file: ../ra/fsp/src/r_elc/r_elc.c
Building file: ../ra/fsp/src/r_adc/r_adc.c
Building file: ../ra/fsp/src/r_dmac/r_dmac.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_clocks.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_common.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_delay.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_group_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_guard.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_io.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_macl.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_register_protection.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_sbrk.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_rom_registers.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_security.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/system.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.c
Building file: ../ra/board/ra8m1_voice/board_init.c
Building file: ../ra/board/ra8m1_voice/board_leds.c
Building target: RA8M1_VOICE_qsg_amic.elf
arm-none-eabi-objcopy -O srec "RA8M1_VOICE_qsg_amic.elf"  "RA8M1_VOICE_qsg_amic.srec"
arm-none-eabi-size --format=berkeley "RA8M1_VOICE_qsg_amic.elf"
   text    data     bss     dec     hex filename
   7444      24   65688   73156   11dc4 RA8M1_VOICE_qsg_amic.elf

15:27:47 Build Finished. 0 errors, 0 warnings. (took 3s.129ms)
```

2.13 The application is now ready to be programmed and run on the Voice kit. Press the “bug” icon to begin the debug session.

2.14 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.

2.15 Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session.

END OF SECTION

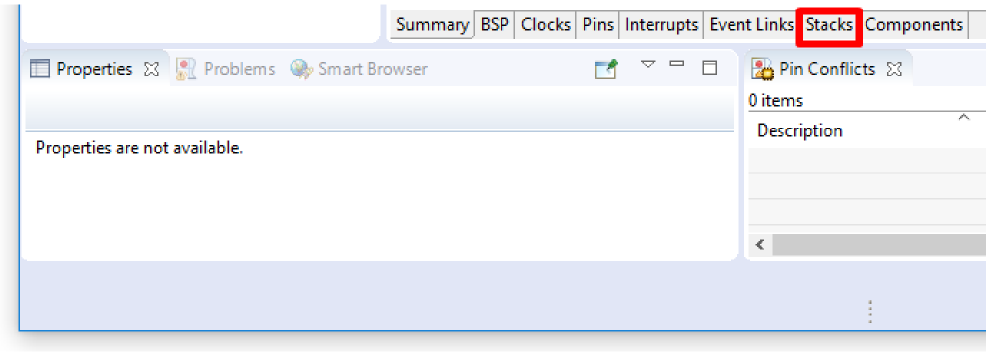
3 Implementing DMIC demo

Overview

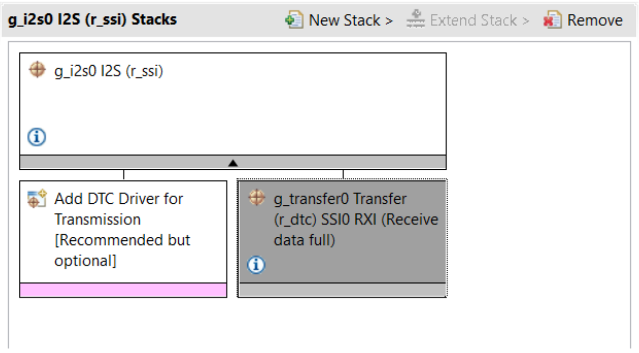
Following section describes in details steps required to set up a Digital Microphone demo project for RA8M1 Voice Kit.

Procedural Steps

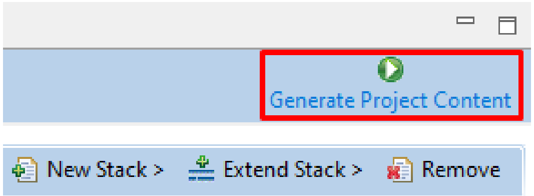
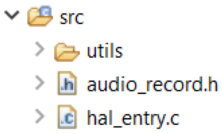
<table>
<tr><td>3.1</td><td>Create a new project and follow the steps described from 1.1 to 1.15.</td></tr>
<tr><td>3.2</td><td>Once new project is created, e^2 studio will switch to a layout optimized for developing Renesas RA projects. Select the <b>Stacks</b> tab at the bottom of the <b>FSP Configuration</b> pane visible in the middle.



Properties are not available.</td></tr>
<tr><td>3.3</td><td>Access the <b>New Stack</b> menu and select <b>Connectivity -> I2S (r_ssi)</b>.
Use <b>Properties</b> tab to configure following properties for this new module:
<table>
<tr><td>DTC Support</td><td>Enable</td></tr>
<tr><td>Name</td><td>g_i2s0</td></tr>
<tr><td>Word length</td><td>32 Bits</td></tr>
<tr><td>Bit Clock Source(available only in Master mode)</td><td>Internal AUDIO_CLK</td></tr>
<tr><td>Callback</td><td>g_audio_cb</td></tr>
<tr><td>Transmit Interrupt Priority</td><td>Disabled</td></tr>
<tr><td>Receive Interrupt Priority</td><td>Priority 2</td></tr>
<tr><td>Idle/Error Interrupt Priority</td><td>Priority 2</td></tr>
</table></td></tr>
<tr><td>3.4</td><td>In the I2S module, press <b>Add DTC Driver for Trasmission -> New -> Transfer (r_dtc)</b>


g_i2s0 I2S (r_ssi) Stacks
New Stack >
Extend Stack >
Remove
g_i2s0 I2S (r_ssi)
Add DTC Driver for Transmission [Recommended but optional]
g_transfer0 Transfer (r_dtc) SSI0 RXI (Receive data full)

</td></tr>
</table>

3.5	Access the **New Stack** menu and select **Timers -> Timer, General PWM (r_gpt)**. Use **Properties** tab to configure following properties for this new module: • Name g_i2s_clock • Channel 2 • Period 1024000 • Output→GTIOCA Output Enabled True
3.6	Access the **New Stack** menu and select **Connectivity -> I2C Master (r_iic_master)**. Use **Properties** tab to configure following properties for this new module: • Name g_i2c_da7218 • Channel 1 • Slave Address 0x1A • Timeout During SCL Low Disabled • Callback i2c_master_callback • Pins→SCL1 P205 • Pins→SDA1 P206
3.7	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**.  Generate Project Content New Stack > Extend Stack > Remove
3.8	The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the **Properties** tab.
3.9	In the **Project Explorer** pane, expand the **src** folder in the project, and add the following folders and files: • utils • audio_record.h
3.10	In the **Project Explorer** pane, expand the **src** folder in the project and open **hal_entry.c**. 
3.11	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).
3.12	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 enable the digital microphones for the VK-RA8M1 board:

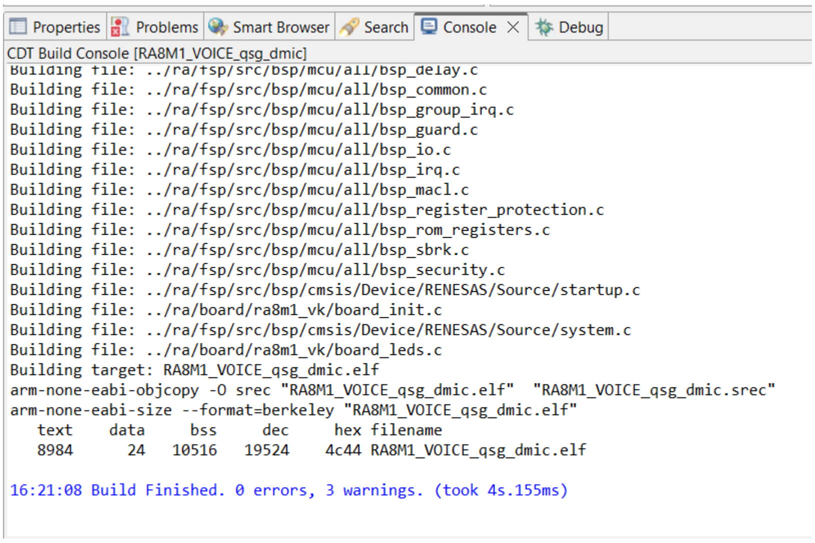
```
#include "hal_data.h"
#include "audio_record.h"
#include "utils/ringbuffer.h"
#include "utils/da7218.h"
FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

typedef unsigned int                    UINT;
#define I2S_BUFFER_SIZE               256
#define RING_BUFFER_SIZE              I2S_BUFFER_SIZE*16     //I2S_BUFFER_SIZE * 16 slots
static ring_buffer_t rb_hdl;
static char audio_rb[RING_BUFFER_SIZE];
static char i2s_buf[I2S_BUFFER_SIZE];
/***********************************************************************************
********************************//**
 * 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 */
   fsp_err_t  err = FSP_SUCCESS;

   ring_buffer_init(&rb_hdl, audio_rb, RING_BUFFER_SIZE);

   err = R_SSI_Open(&g_i2s0_ctrl, &g_i2s0_cfg);
      if (err != FSP_SUCCESS) {
         __BKPT(0);
      }

   err = R_GPT_Open(&g_i2s_clock_ctrl, &g_i2s_clock_cfg);
      if (err != FSP_SUCCESS) {
         __BKPT(0);
      }
         audio_record_start();


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

fsp_err_t audio_record_start(void)
{
   fsp_err_t  err = FSP_SUCCESS;

   err = R_GPT_Start(&g_i2s_clock_ctrl);
   if (err != FSP_SUCCESS) {
      __BKPT(0);
   }
```

```
    err = R_SSI_Read(&g_i2s0_ctrl, i2s_buf, I2S_BUFFER_SIZE);
    if (err != FSP_SUCCESS) {
        __BKPT(0);
    }
    return err;
}

fsp_err_t audio_record_stop(void)
{
    fsp_err_t  err = FSP_SUCCESS;

    err = R_GPT_Stop(&g_i2s_clock_ctrl);
    if (err != FSP_SUCCESS) {
        __BKPT(0);
    }

    return err;
}

void g_audio_cb(i2s_callback_args_t *p_args)
        {
            fsp_err_t err = FSP_SUCCESS;

            switch (p_args->event) {
                case I2S_EVENT_IDLE: ///< Communication is idle
                    err = R_SSI_Read(&g_i2s0_ctrl, i2s_buf, I2S_BUFFER_SIZE);
                    if (err != FSP_SUCCESS) {
                        __BKPT(0);
                    }
                    break;

                case I2S_EVENT_TX_EMPTY: ///< Transmit buffer is below FIFO trigger level
                    //DBG_UART_TRACE("I2S_EVENT_TX_EMPTY.\r\n");
                    break;

                case I2S_EVENT_RX_FULL: ///< Receive buffer is above FIFO trigger level
                    ring_buffer_queue_arr(&rb_hdl, i2s_buf, I2S_BUFFER_SIZE);
                    err = R_SSI_Read(&g_i2s0_ctrl, i2s_buf, I2S_BUFFER_SIZE);
                    if (err != FSP_SUCCESS) {
                        break;
                    }
                    break;

                default:
                    break;
            }
        }

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

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

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

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

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

      /* Configure pins. */
      R_IOPORT_Open (&g_ioport_ctrl, &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
```

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

3.14 Once the build has finished, the **Console** pane in the lower-right corner of e^2 studio will report zero error and warnings:



```
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_delay.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_common.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_group_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_guard.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_io.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_macl.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_register_protection.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_rom_registers.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_sbrk.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_security.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.c
Building file: ../ra/board/ra8m1_vk/board_init.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/system.c
Building file: ../ra/board/ra8m1_vk/board_leds.c
Building target: RA8M1_VOICE_qsg_dmic.elf
arm-none-eabi-objcopy -O srec "RA8M1_VOICE_qsg_dmic.elf"  "RA8M1_VOICE_qsg_dmic.srec"
arm-none-eabi-size --format=berkeley "RA8M1_VOICE_qsg_dmic.elf"
   text    data     bss     dec     hex filename
   8984      24   10516   19524    4c44 RA8M1_VOICE_qsg_dmic.elf

16:21:08 Build Finished. 0 errors, 3 warnings. (took 4s.155ms)
```

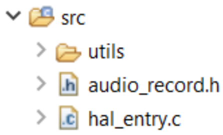
3.15 The application is now ready to be programmed and run on the VK kit. Press the “bug” icon to begin the debug session.

3.16 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.

3.17 Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session.

3.18 The user can also use the digital microphones from the MIC Board, for this we will need before powering up the solution kit, to connect MIC Board J14 and the other end to the J7 of the motherboard.

3.19 In the **Project Explorer** pane, expand the **src** folder in the project and open again **hal_entry.c**.



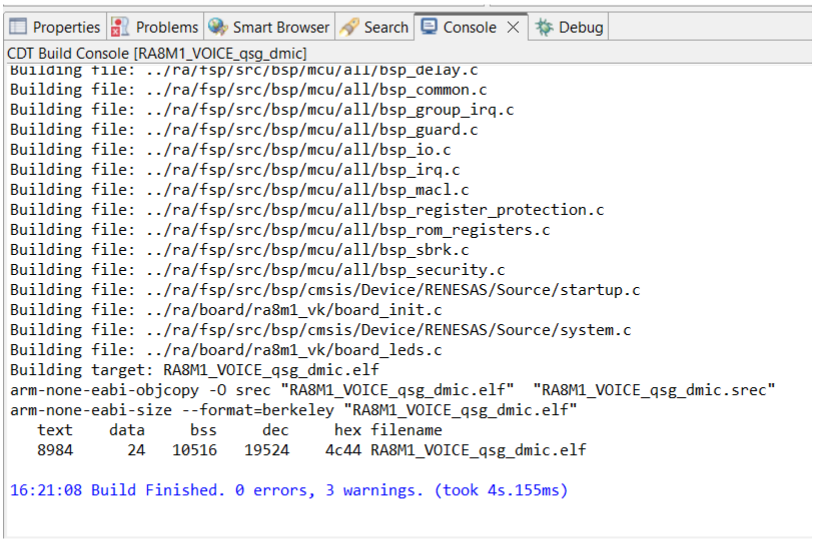
```
src
  utils
  audio_record.h
  hal_entry.c
```

3.20 Below `ring_buffer_init(&rb_hdl, audio_rb, RING_BUFFER_SIZE);` **add:**

```
    if (enable_da7218()) {
            err = init_da7218();
    }
```

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

3.22 Once the build has finished, the **Console** pane in the lower-right corner of e^2 studio will report zero error and warnings:



CDT Build Console [RA8M1_VOICE_qsg_dmic]

```
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_delay.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_common.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_group_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_guard.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_io.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_macl.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_register_protection.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_rom_registers.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_sbrk.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_security.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.c
Building file: ../ra/board/ra8m1_vk/board_init.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/system.c
Building file: ../ra/board/ra8m1_vk/board_leds.c
Building target: RA8M1_VOICE_qsg_dmic.elf
arm-none-eabi-objcopy -O srec "RA8M1_VOICE_qsg_dmic.elf"  "RA8M1_VOICE_qsg_dmic.srec"
arm-none-eabi-size --format=berkeley "RA8M1_VOICE_qsg_dmic.elf"
   text    data     bss     dec     hex filename
   8984      24   10516   19524    4c44 RA8M1_VOICE_qsg_dmic.elf

16:21:08 Build Finished. 0 errors, 3 warnings. (took 4s.155ms)
```

3.23 The application is now ready to be programmed and run on the VK kit. Press the “bug” icon to begin the debug session.

3.24 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.

3.25 Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session.

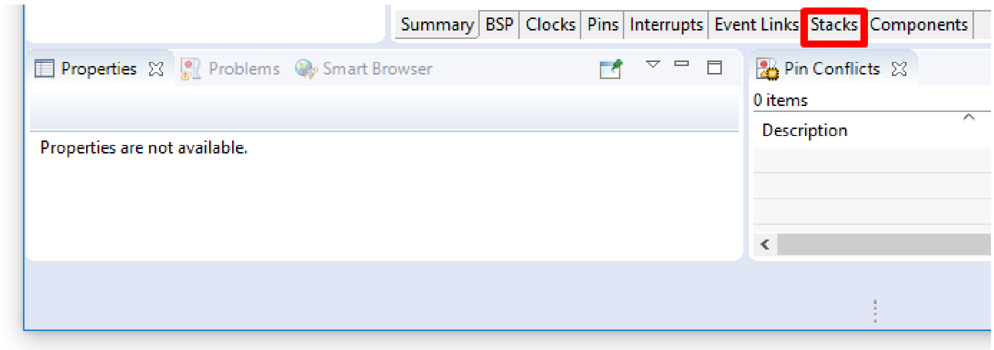
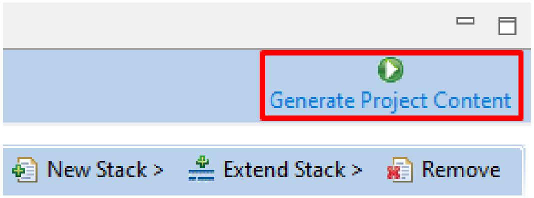
END OF SECTION

4 Implementing DAC demo

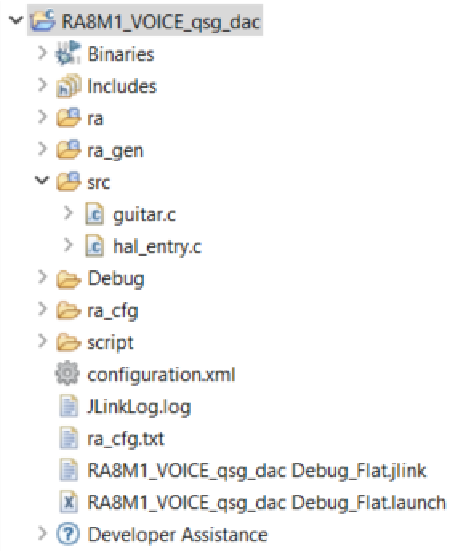
Overview

Following section describes in details steps required to set up a DAC project for RA8M1 Voice Kit.

Procedural Steps

4.1	Create a new project and follow the steps described from 1.1 to 1.15.
4.2	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.  Summary BSP Clocks Pins Interrupts Event Links Stacks Components Properties Problems Smart Browser Pin Conflicts 0 items Description Properties are not available.
4.3	Access the **New Stack** menu again and select **Analog > DAC (r_dac)**.
4.4	Access the **New Stack** menu again and select **Transfer > Transfer (r_rdmac)**. Use **Properties** tab to configure following properties for this new module: • Name g_transfer_dac • Channel 1 • Source Address Mode Incremented • Activation Source GPT5 COUNTER OVERFLOW (Overflow) • Callback g_transfer_dac_cb • Transfer End Interrupt Priority Priority 13
4.5	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**.  Generate Project Content New Stack > Extend Stack > Remove
4.6	The FSP Configurator will extract all the necessary drivers and generate the code based on the configuration provided in the **Properties** tab.
4.7	In the **Project Explorer** pane, expand the **src** folder in the project, and add the following file: • guitar.c

4.8 In the **Project Explorer** pane, expand the **src** folder in the project and open **hal_entry.c**.



- RA8M1_VOICE_qsg_dac
  - Binaries
  - Includes
  - ra
  - ra_gen
  - src
    - guitar.c
    - hal_entry.c
  - Debug
  - ra_cfg
  - script
  - configuration.xml
  - JLinkLog.log
  - ra_cfg.txt
  - RA8M1_VOICE_qsg_dac Debug_Flat.jlink
  - RA8M1_VOICE_qsg_dac Debug_Flat.launch
  - Developer Assistance

4.9 **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).

4.10 **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 for the VK-RA8M1 board:

```
#include "hal_data.h"

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

extern uint8_t audio_samples[130032];

static volatile bool dac_done;

void hal_entry(void)
{
    fsp_err_t err;

    /* Initialize the DAC peripheral */
    err = R_DAC_Open(&g_dac0_ctrl, &g_dac0_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Enable DAC output */
    err = R_DAC_Start(&g_dac0_ctrl);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Initialize the DMA peripheral */
    err = R_DMAC_Open(&g_transfer_dac_ctrl, &g_transfer_dac_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Initialize the timer used to control the sampling rate */
    err = R_GPT_Open(&g_timer_dac_ctrl, &g_timer_dac_cfg);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    /* Start the timer */
    err = R_GPT_Start(&g_timer_dac_ctrl);
    if (FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    while (1)
    {
        /* Start playback by setting DMA to transfer audio samples to the
DAC */
        err = R_DMAC_Reset(&g_transfer_dac_ctrl, audio_samples, (void *)
R_DAC->DADR,
                           sizeof(audio_samples) / sizeof(uint16_t));
        if (FSP_SUCCESS != err)
        {
            __BKPT(0);
        }

        /* Wait for interrupt & check for event */
        while (false == dac_done)
            __WFI();

        dac_done = false;

        /* Wait before starting the playback again */
        R_BSP_SoftwareDelay(2, BSP_DELAY_UNITS_SECONDS);
    }
}

void g_transfer_dac_cb(dmac_callback_args_t * p_args)
{
    /* Use this callback to end the playback or restart the DMA
     * with more samples to play longer tracks */

    /* Signal that last sample has been sent to DAC */
    dac_done = true;

    /* Suppress compiler warning for unused p_args */
    FSP_PARAMETER_NOT_USED(p_args);
}


void R_BSP_WarmStart(bsp_warm_start_event_t event)
{
    if (BSP_WARM_START_POST_C == event)
    {
```

<table>
<tr><td></td><td>

```
        /* C runtime environment and system clocks are setup. */

        /* Configure pins. */
        R_IOPORT_Open (&g_ioport_ctrl, g_ioport.p_cfg);
    }
}
```

</td></tr>
<tr><td>4.11</td><td>

The example project provides guitar.c file which includes an example track stored as array of PCM inside const unsigned char audio_samples[130032]. You can replace this file with your own samples and/or buffer them in another array in on-chip SRAM. With DAC set to left-justified, the audio samples should be provided in unsigned 16-bit mono PCM format (regardless of whether the storage type in the code is 8, 16 or 32-bit). To convert any audio file to this format, use ffmpeg and execute the following:
ffmpeg.exe -i {input_file} -acodec pcm_u16le -f u16le -ac 1 -ar 16000 {output_file}
Where {input_file} and {output_file} are replaced by the path to input and output, respectively. "16000" after the "-ar" is the output sampling rate setting and should match timer rate set in step 5.5.

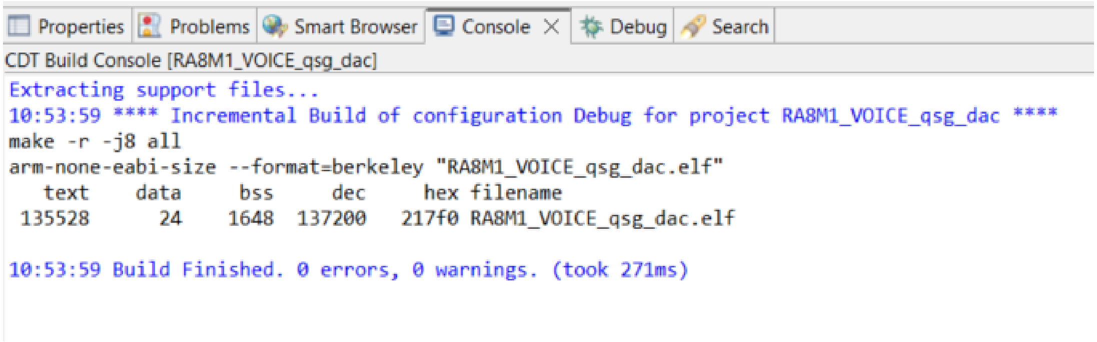
Raw audio files output by ffmpeg can be included in the project either by converting them to a C array or by creating an assembly file with .incbin directive to inline the file.

</td></tr>
<tr><td>4.12</td><td>

The project is now ready to compile. Press the "hammer" icon to start building the project.

</td></tr>
<tr><td>4.13</td><td>

Once the build has finished, the **Console** pane in the lower-right corner of e^2 studio will report zero errors:



CDT Build Console [RA8M1_VOICE_qsg_dac]

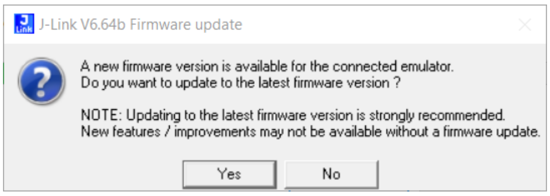
```
Extracting support files...
10:53:59 **** Incremental Build of configuration Debug for project RA8M1_VOICE_qsg_dac ****
make -r -j8 all
arm-none-eabi-size --format=berkeley "RA8M1_VOICE_qsg_dac.elf"
   text    data     bss     dec     hex filename
 135528      24    1648  137200   217f0 RA8M1_VOICE_qsg_dac.elf

10:53:59 Build Finished. 0 errors, 0 warnings. (took 271ms)
```

</td></tr>
<tr><td>4.14</td><td>

The application is now ready to be programmed and run on the Voice kit. Press the "bug" icon to begin the debug session.

</td></tr>
<tr><td>4.15</td><td>

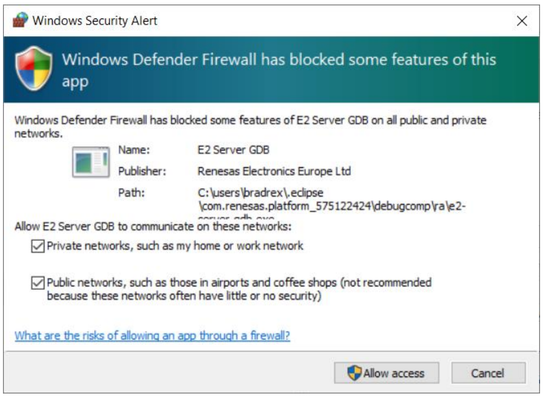
You may be prompted to update the J-Link debugger firmware. You can click **Yes** to update. It will take a few moments to complete.



A new firmware version is available for the connected emulator.
Do you want to update to the latest firmware version ?

NOTE: Updating to the latest firmware version is strongly recommended.
New features / improvements may not be available without a firmware update.

</td></tr>
</table>

4.16	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.  Windows Security Alert Windows Defender Firewall has blocked some features of this app Windows Defender Firewall has blocked some features of E2 Server GDB on all public and private networks. Name: E2 Server GDB Publisher: Renesas Electronics Europe Ltd Path: C:\users\bradrex\.eclipse \com.renesas.platform_575122424\debugcomp\ra\e2-server-gdb.exe Allow E2 Server GDB to communicate on these networks: ☑ Private networks, such as my home or work network ☑ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security) What are the risks of allowing an app through a firewall? Allow access Cancel
4.17	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**.
4.18	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.
4.19	Click the Resume button or press F8 on the keyboard to start the application.
4.20	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.
4.21	As application is executing. The sound capture is running continuously with each new data set being passed to the main loop approximately every 500ms (16000Hz sampling rate with 8000 samples per buffer). The sample code implements double buffering to allow for further processing of the data without breaking the data continuity. Example application can be easily extended to use the data captured, e.g. for voice recognition model or real-time streaming to another host.
4.22	Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session.

For further information and inquiries please contact: rai-cs@dm.renesas.com

END OF SECTION