Renesas RA Family

Getting Started with ADC Interleaved Mode Application

Introduction

This application note describes implementing an analog-to-digital converter (ADC) in an interleaved mode in an RA device by utilizing two ADC units that convert the same ADC input pin alternately to double the data sampling rate. The implementation uses a synchronous trigger from a General-Purpose Timer (GPT) to synchronize A/D conversion timing between the two ADC units.

The typical application for ADC interleaved mode requires a double data rate for faster signal processing.

This application note guides you through the steps to implement ADC interleaved mode, including:

- Board setup.
- Application overview.
- FSP/Driver configuration.
- Application design highlights.
- Interleaving ADC units for a higher sample rate using ADC interleave mode in RA MCUs.
- Setting up the ADC, GPT, and DMAC modules.

Required Resources

Development tools and software

- The e^2 studio IDE v2025-04.1
- Renesas Flexible Software Package (FSP) v6.0.0

Hardware

- Renesas EK-RA6M5 kit (RA6M5 MCU Group) (https://www.renesas.com/us/en/products/microcontrollers-microprocessors/ra-cortex-m-mcus/ek-ra6m5-evaluation-kit-ra6m5-mcu-group)
- Waveform generator

Reference Manuals

- RA Flexible Software Package Documentation
- Renesas RA6M5 Group User's Manual Rev.1.40
- EK-RA6M5-v1.0 Schematics

Contents

1. Introduction to Interleaving

ADC interleaving is a technique that enables two or more ADC cores to be operated together to achieve an effective sample rate that is higher than the rate that either ADC can produce on its own. It is a commonly used method to increase the system's overall performance and, when applied correctly, can have great benefits. However, the implementation has its challenges, and performance limitations should be considered carefully.

1.1 Sample Timing Control

Consider the case where two ADC cores are interleaved. In this case, we refer to the cores as the "odd" and "even" ADCs, as they will be used to successively sample the analog input signal and produce digital conversion results that are aligned with the odd-number data points and the even-number data points.

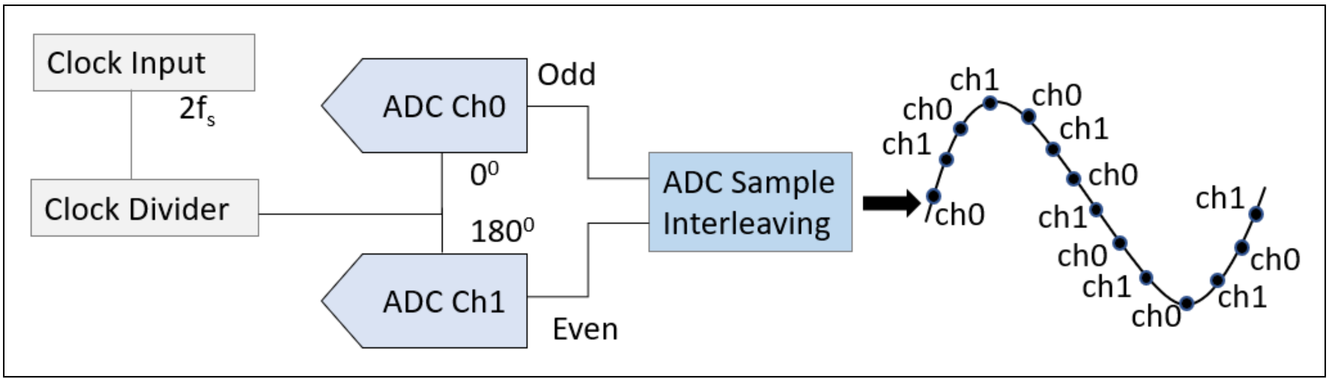

Figure 1. Simple Representation of Two Interleaved ADCs

From Figure 1, we can see the relationship between the sample clock applied to the odd and even ADC cores. The sample clock, oscillating at $2f_s$, is divided by two to ensure a 50:50 mark-space ratio and split into two signals with a 180° phase shift (by simple inversion in one of the outputs).

Each ADC core samples the analog signal on the rising edge of its sample clock, which, due to the divide by two, is now f_s. The odd ADC core produces the first result, followed by the even ADC core half a cycle later. The resulting output data stream, after the two converters' outputs have been combined, provides a sample rate that is back to $2f_s$.

For the interleaving to work correctly and provide a spurious-free output data stream, the phase relationship between the odd and even ADCs must be predictable, stable, and precisely 180°. The implementation of this phase relationship is the most important aspect of achieving correct interleaving.

For an AC system where the ADC is used to sample an input signal, if the phase relationship is not precisely 180°, spurs will appear in the output spectrum. These spurs will appear as a function of the sample rate f_s and the input frequency, so $f_s/2 \pm f_{in}$. See Figure 2.

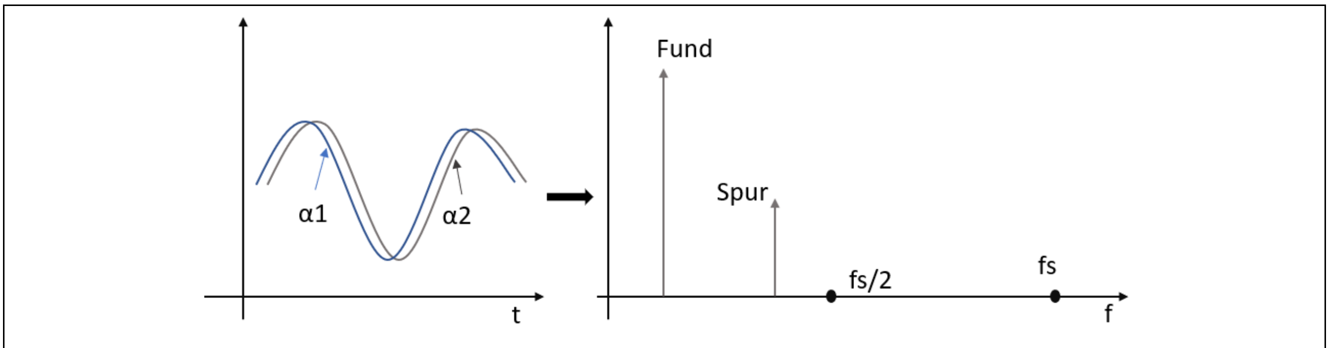

Figure 2. Sampled Data Spectrum due to Timing Error

For DC applications where a slow-moving signal is sampled and provided as feedback to a control loop, the phase angle error will create an amplitude error that can lead to uncertainty within the control loop if the timing irregularity is not considered.

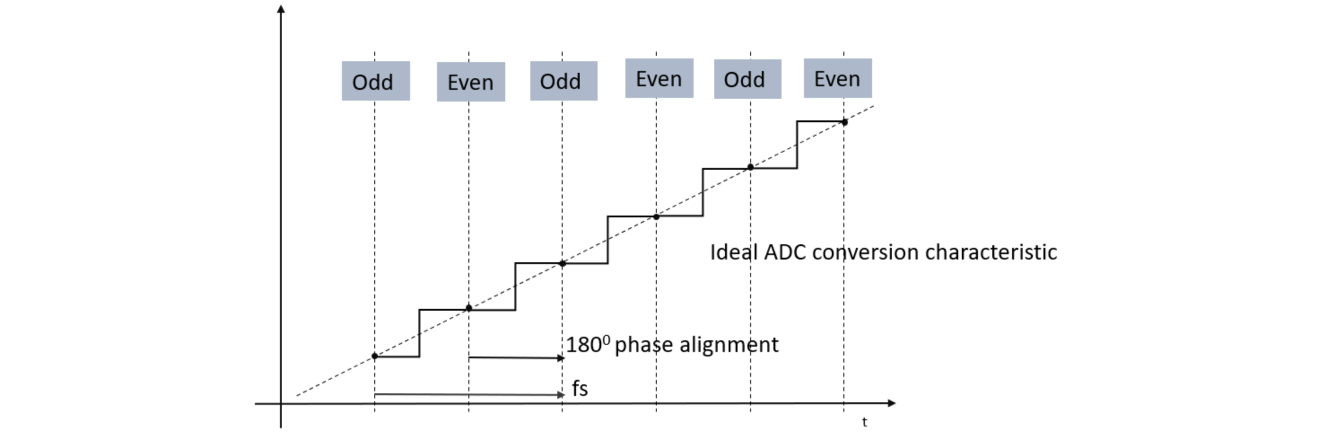

Figure 3. Ideal Sample Timing with 50:50 Phase Alignment

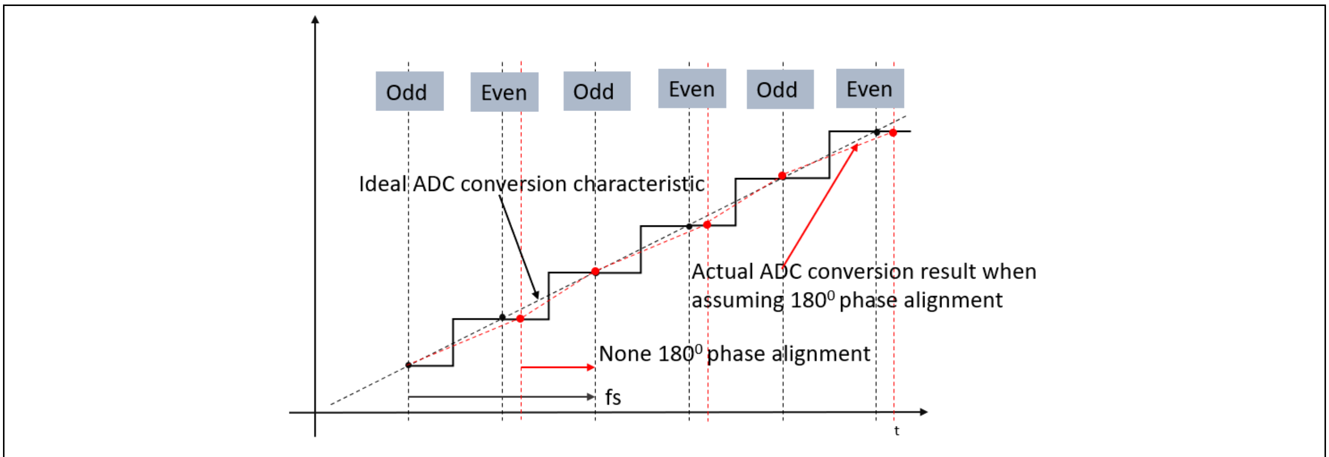

Figure 4. Sample Timing with Error with 60:40 Phase Alignment

1.2 Gain and Offset Mismatch

Other errors that can be present within the ADC output stream are related to gain and offset mismatch between the two ADC cores.

Gain mismatch can be present where individual PGAs are used prior to the odd and even ADC cores. Any mismatch between the absolute gain value of these two PGAs will result in a sample-by-sample error being present, causing the value created by the two ADC cores to be mismatched, diverging by an increasing amount as the signal approaches full scale. This effect can also be caused if individual or unstable voltage references are used. A spur will appear in the output spectrum at $f_s/2 \pm f_{in}$.

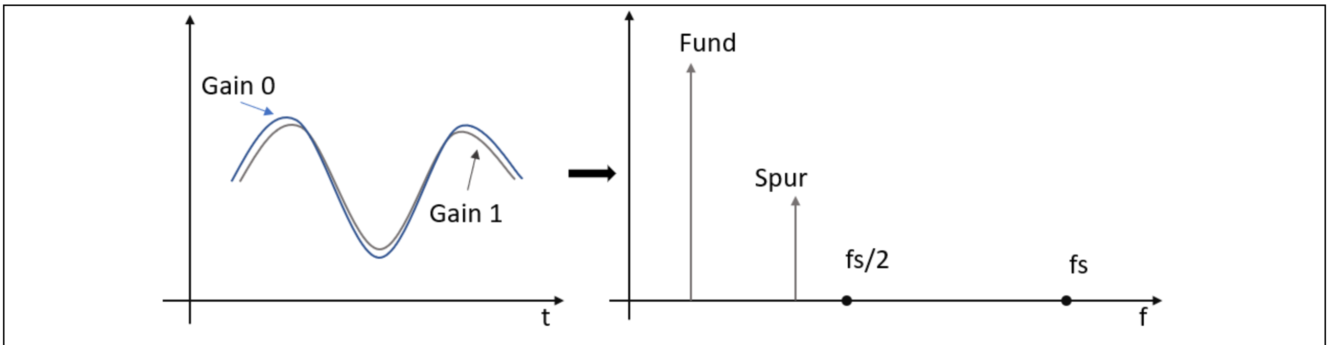

Figure 5. Sampled Data with Gain Mismatch

Offset mismatch can occur when the DC offset value of the odd and even ADC cores is not the same. Any mismatch results in a sample-by-sample error in the output values. In this case, a spur will appear at half of the combined sampling frequency, which is f_s. The magnitude of the spur is defined by the amplitude of the mismatch.

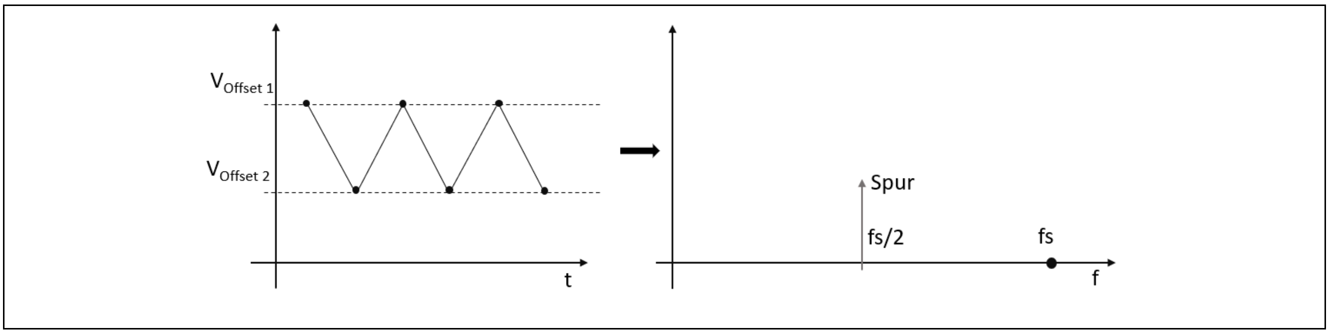

Figure 6. Sampled Data with Offset Mismatch

Typically, the causes of these effects are minimized on a monolithic device such as an MCU, where the two ADC cores are fabricated next to each other. If any effects are noticeable, they can be digitally compensated for in firmware quite simply with data manipulation for gain and offset.

2. Application Overview

The application project accompanying this document serves as a reference to create the ADC Interleave Mode in an RA device by utilizing two ADC units, ADC0 and ADC1. The ADC units convert the same ADC input pin alternately to double the data sampling rate, using a synchronous trigger from a GPT to synchronize A/D conversion timing between 2 channels. The conversion data is transferred to the same buffer using a DMAC.

Figure 7 shows the application diagram.

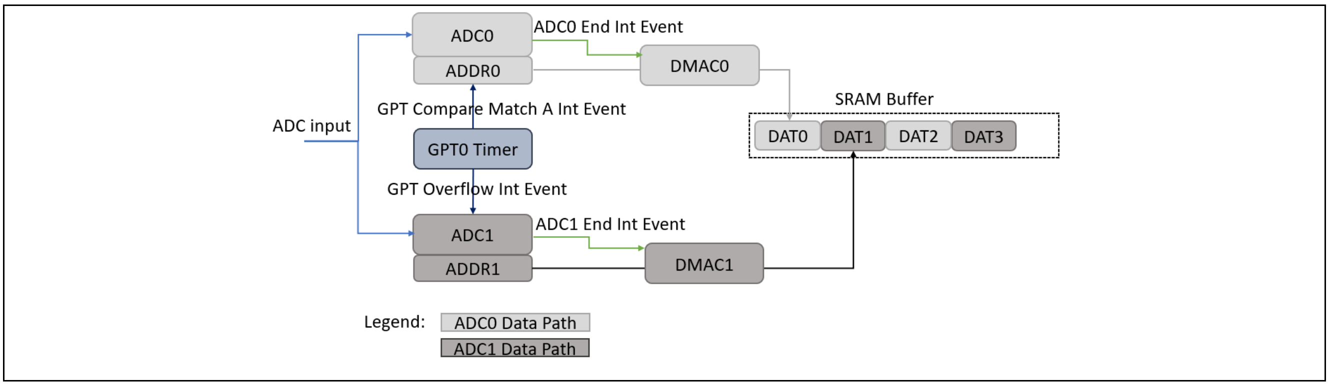

Figure 7. Block Diagram

3. Interleaving Two ADC Units for a Higher Sample Rate

ADC Interleave mode in the RA6M5 MCU allows the use of two identical ADC units to process regular sample data series at a faster rate than the operating sample rate of each individual data converter.

As shown in the Figure 8, two ADC units convert the same ADC input pin alternately to double the data sampling rate.

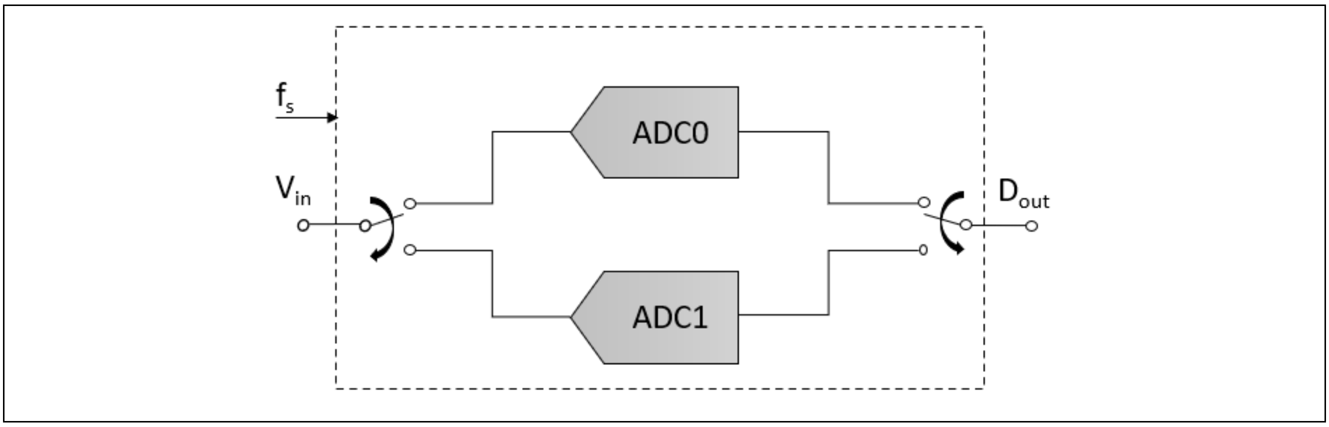

Figure 8. Interleaving 2 Channels of 2 ADC Units

4. 12-Bit ADC Converter (ADC12)

The RA6M5 MCU includes 12-bit successive approximation A/D converter (ADC12) units. In unit 0, up to 13 analog input channels are selectable. In unit 1, up to 16 analog input channels, temperature sensor output, and internal reference voltage can be selected for conversion in respective units.

The A/D conversion accuracy is selectable among 12-bit, 10-bit, and 8-bit, making it possible to optimize the trade-off between speed and resolution in generating a digital value.

The ADC12 supports the following operating modes and features:

- Single scan mode to convert analog inputs of selected channels in ascending order of channel number.
- Continuous scan mode to continuously convert analog inputs of selected channels in ascending order of channel number.
- Group scan mode to divide analog inputs of channels into two groups (group A and group B) and convert the analog inputs of selected channels for each group in ascending order of channel number.
- Conversion time is 0.4 microseconds/channel (when 12-bit A/D conversion clock PCLKC (ADCLK) is operating at 50 MHz).
- Support for interleave function.

Refer to the ADC12 chapter in the RA6M5 MCU User's Manual for more details.

Figure 9 and Figure 10 show the timing in interleaved operation.

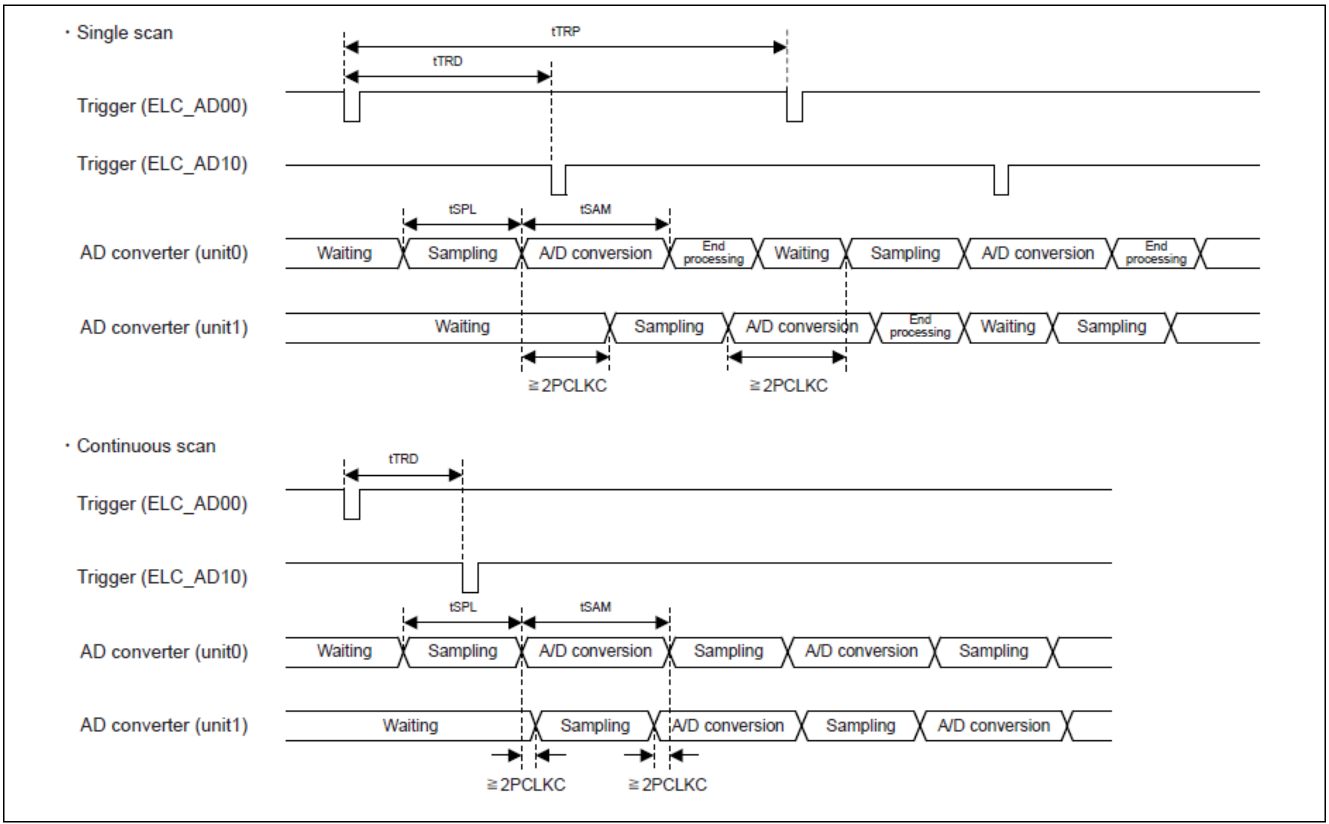

Figure 9. Trigger Input Timing in Interleaved Operation

Parameter		Symbol	Min	Max
Trigger issuance delay period	Single scan	t_{TRD}	t_{SPL} + 2PCLKC	t_{TRP} - t_{SPL} - 2PCLKC
	Continuous scan	t_{TRD}	t_{SPL} + 2PCLKC	t_{SAM} - 2PCLKC
Trigger issuance period		t_{TRP}	tSCAN	—

Figure 10. Setting for Trigger Input Timing in Interleaved Operation

5. FSP Configuration

One of the first things you must do when writing an FSP application is to configure the FSP. Refer to the *Renesas Flexible Software Package (FSP) User's Manual* for more information.

In this application, the FSP configuration is stored in a file named `configuration.xml`. Double-clicking on this file opens the project's RA Configuration tab.

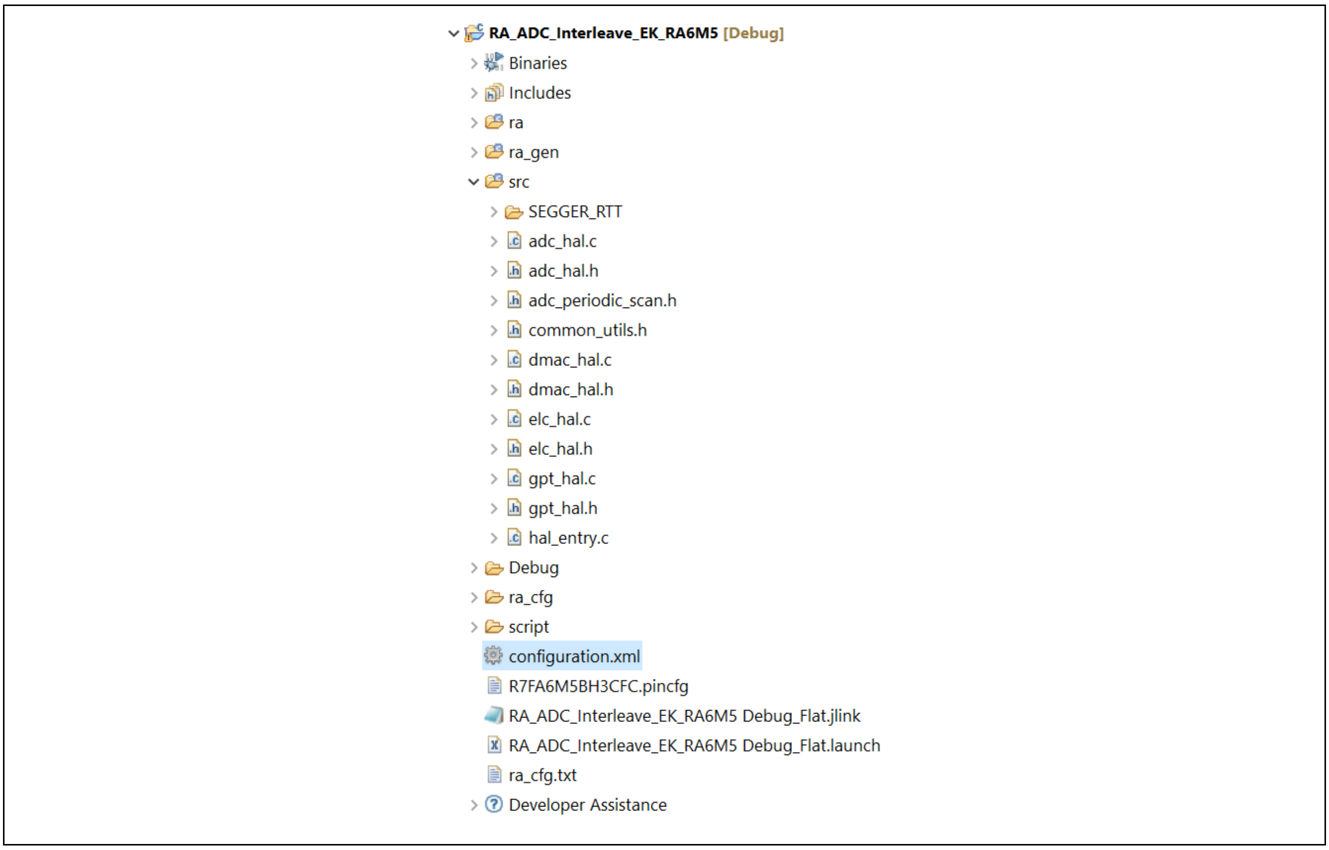

Figure 11. configuration.xml on the Project Plane

5.1 Stacks Tab

The **Stacks** tab allows you to add and configure the threads that the FSP automatically creates for your application. You define a new thread by clicking the button and then entering a unique name for your new thread. Once you add a new thread, you must define the Modules that the thread will use along with any thread objects that will be used by your thread.

As an example, if you click the **HAL/Common**, you should see something like the screen capture shown in Figure 12. This shows that the project requires multiple modules, such as the r_gpt driver, which is used to control the GPT unit.

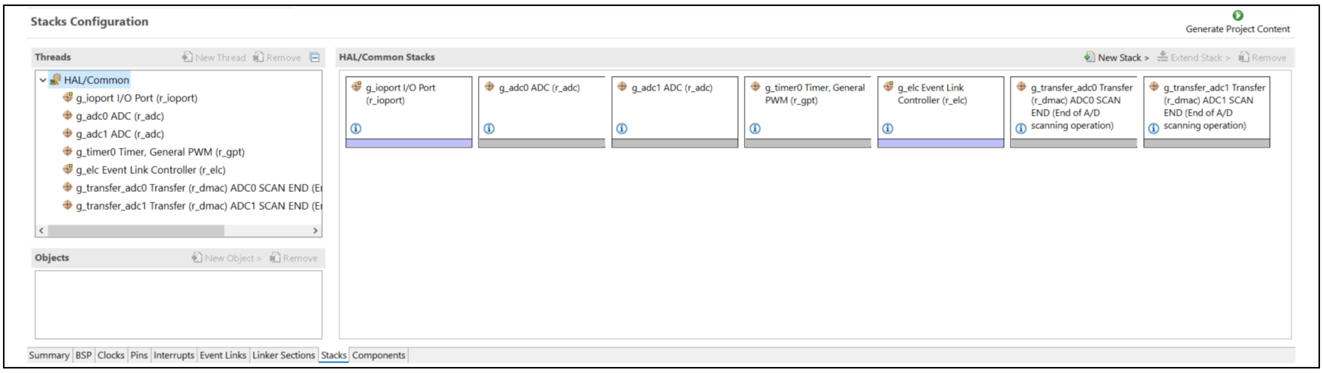

Figure 12. Modules Using for the Application

You can add modules to any thread by clicking the New Stack button. If you have chosen the appropriate components prior to adding Modules to your threads, you should not receive any errors. As an example, Figure 13 shows you how to add a GPT timer to the HAL/Common. The timer is added by choosing **New Stack** > **Search** > **r_gpt**.

If you pick a module for which you have not preselected the appropriate component first, the FSP automatically selects the component for you. If the FSP detects errors with the module addition, it prefaces the module with an error. You can examine the errors by hovering over the module name.

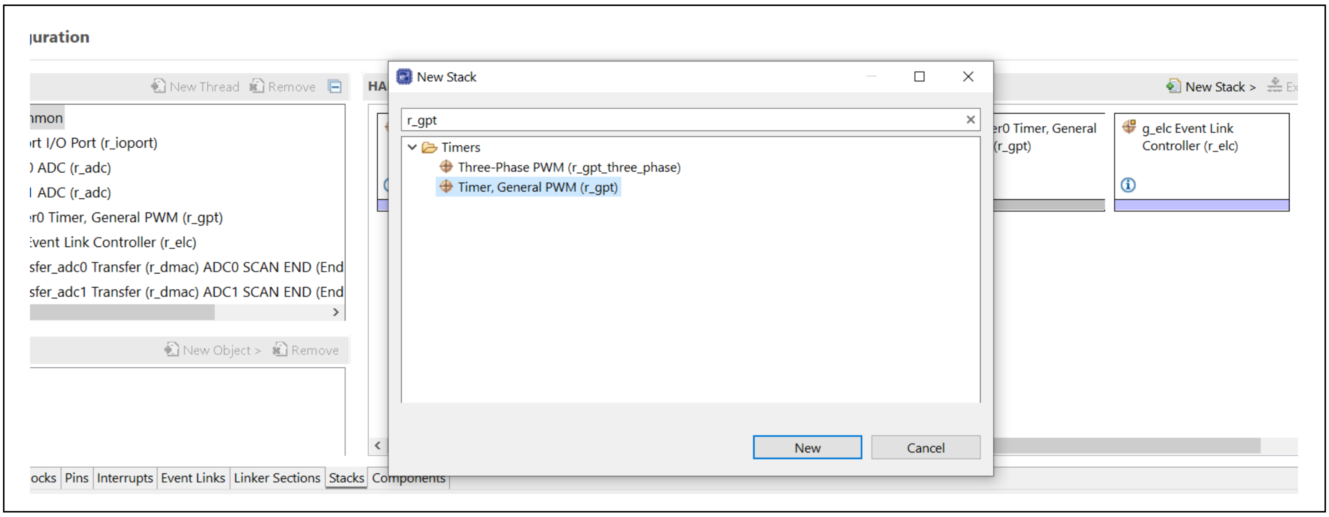

Figure 13. Adding r_gpt Driver

5.2 Module Configuration

Once you have added a module to your project, you need to configure its properties. The properties depend on the module(s) that you have added. Use the **Properties** tab to configure them.

5.2.1 ADC Input

In this application note, P000 is used as a single input pin for both ADC channels (AN000 and AN100) in interleave mode. The interleave function is also available with AN001/AN101 or AN002/AN102.

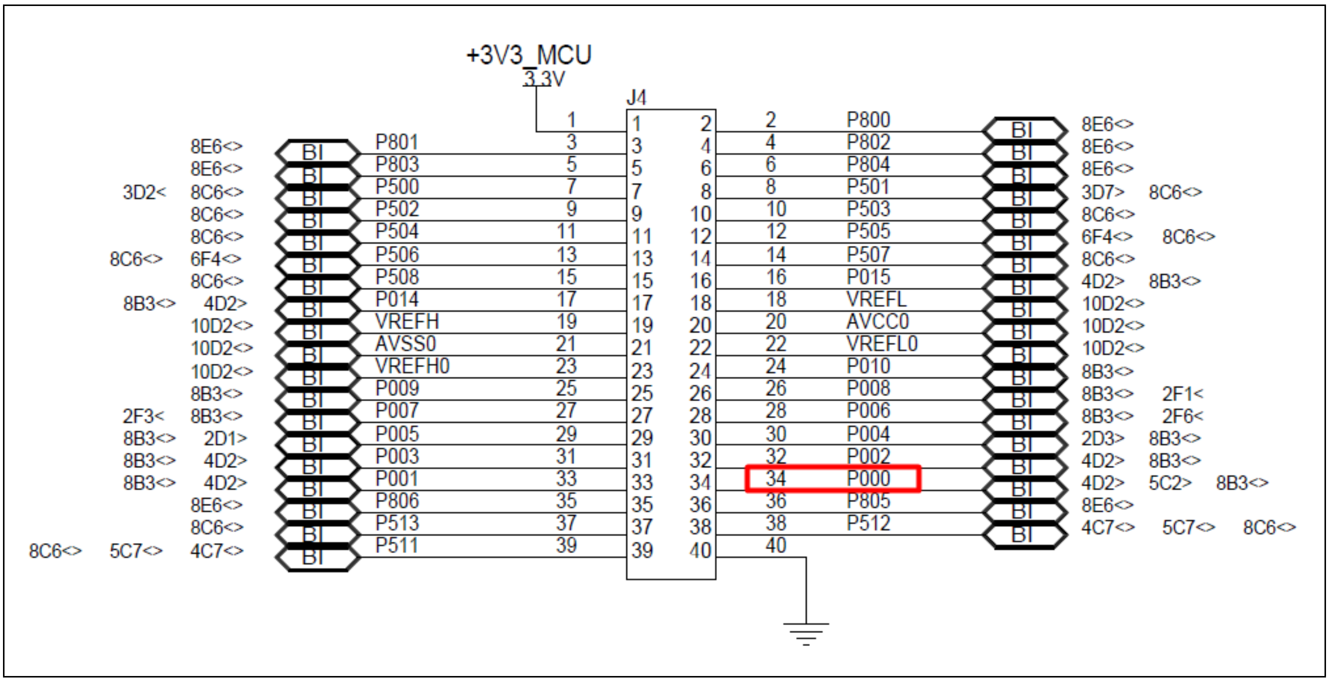

Figure 14. ADC Input Pin

5.2.2 ADC, Pin Configuration

In **Pins** tab, navigate to **Peripherals** > **Analog:ADC** at the **Pin Selection** window. Then, in the **Pin Configuration** window, set P000 as both inputs AN000 and AN100, as shown in the following figures:

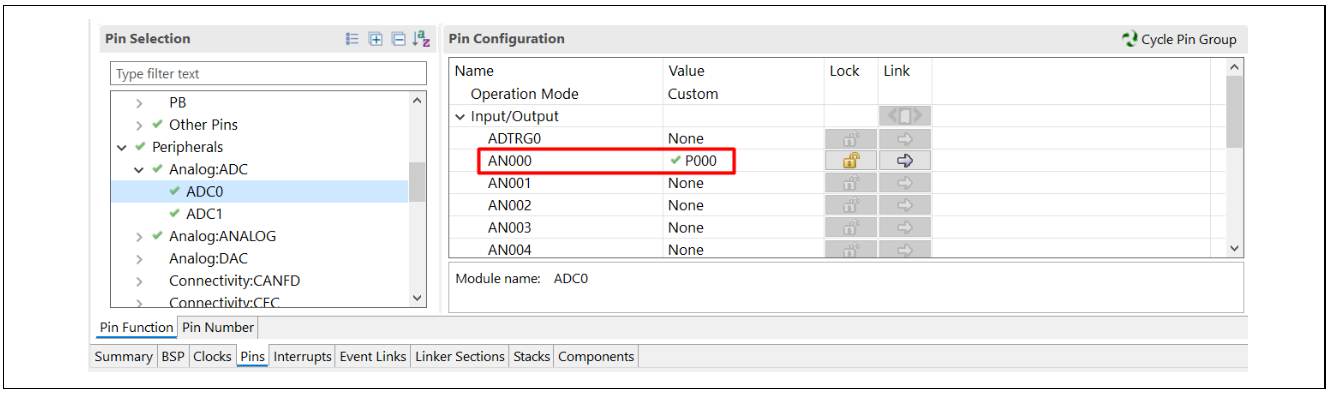

Figure 15. Configure P000 as AN000

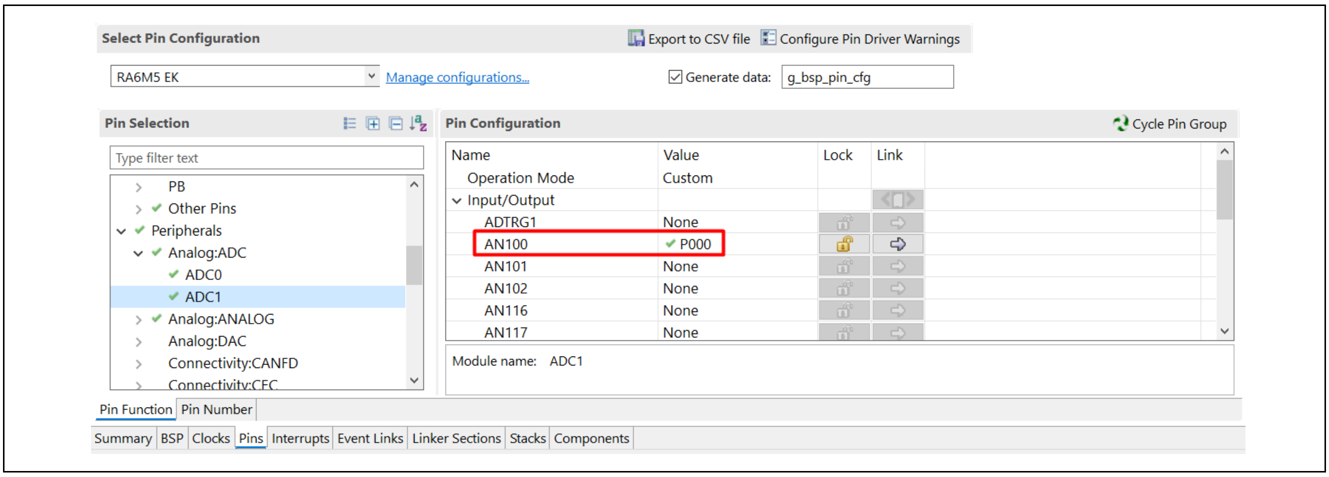

Figure 16. Configure P000 as AN100

5.2.3 GPT Configuration

The following figures show the setup for ADC continuous scan mode. In the r_gpt configuration, enable Capture A Interrupt and Overflow interrupt to trigger ADC 0 and ADC1 conversions.

The timer is configured in One-Shot mode since it only needs to trigger once in ADC continuous scan mode.

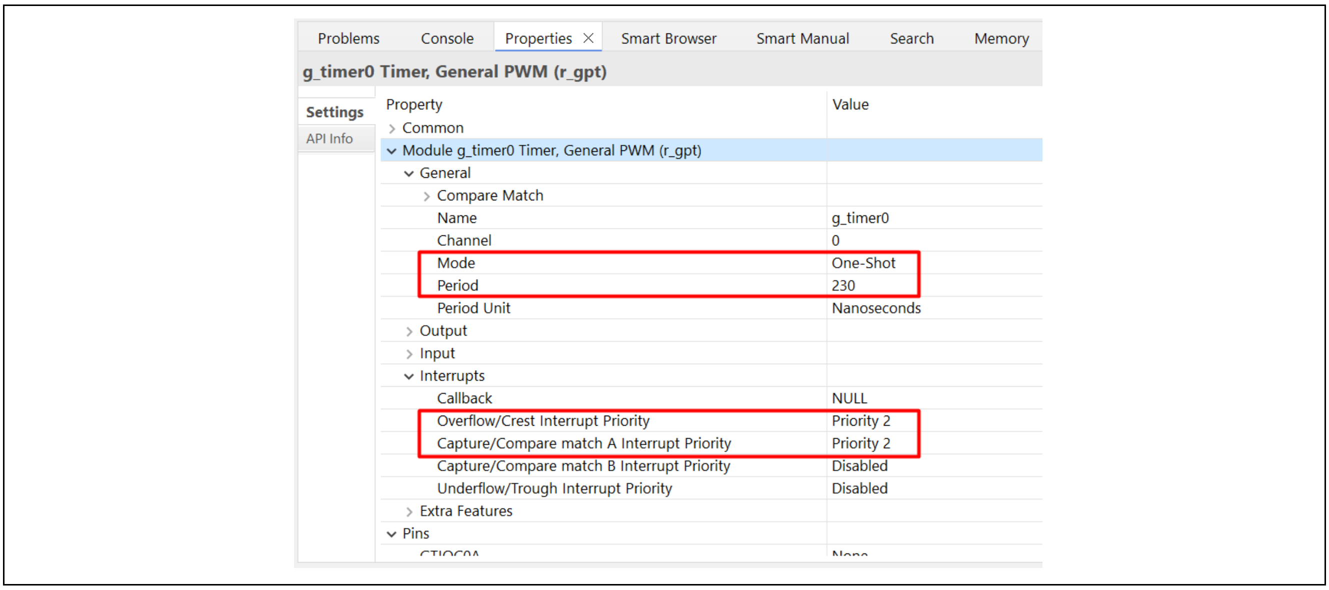

Figure 17. GPT Driver Configuration

The GPT timer's timing setting is the trigger issuance delay period (t_{TRD}) mentioned in Figure 10. The t_{SPL}, T_{SAM} and t_{SCAN} in Figure 10 can be found in Table 43.27 of the *RA6M5 User's Manual* (R01UH0891).

The minimum setting of t_{TRD} is as follows:

t_{TRD} = t_{SPL} + 2PCLKC, where t_{SPL} = ADSSTRn (initial value = 0Bh) x ADCLK + 0.5 ADCLK.

t_{TRD} = 11ADCLK + 0.5 ADCLK = 11.5 ADCLK (where ADCLK = 50 MHz).

t_{TRD} = 230 ns.

5.2.4 ADC Configuration

The ADC0 and ADC1 units will be configured in continuous scan mode, triggered by GPT0 Compare Match A and Overflow interrupts. ADC0 and ADC1 Scan End interrupts are used to trigger DMAC transfer from A/D data registers to a buffer in SRAM.

The GPT Compare Match A interrupt triggers at half the timer cycle, while the GPT Overflow interrupt triggers at the full-timer cycle.

Note: Since the ADC Scan End interrupt is always generated by hardware, it is unnecessary to enable it in the ADC driver configuration to trigger DMAC transfer.

The detailed settings are shown in the following figures.

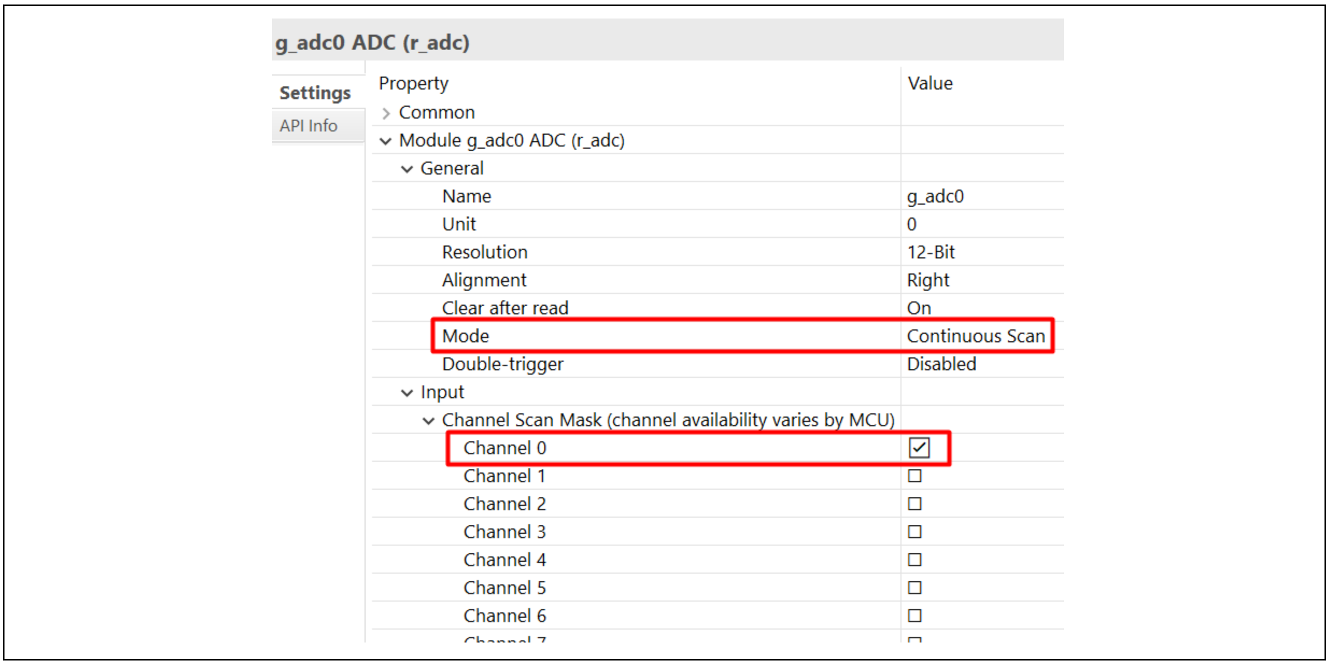

Figure 18. Configure ADC0 in Continuous Scan Mode, Channel 0 as Input

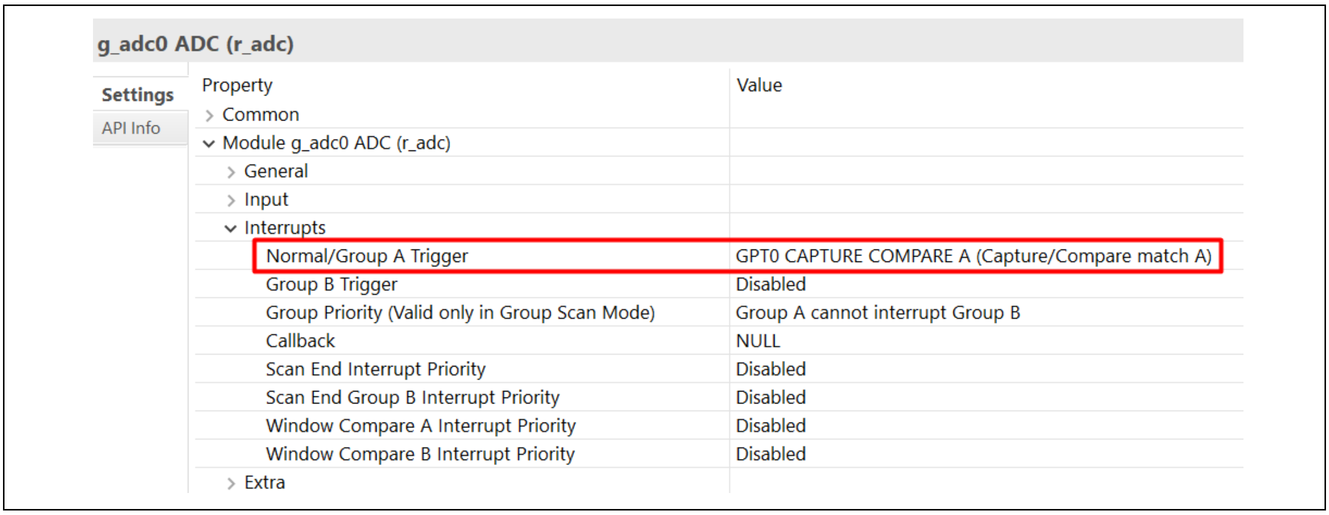

Figure 19. Set Trigger to Start ADC0 Conversion

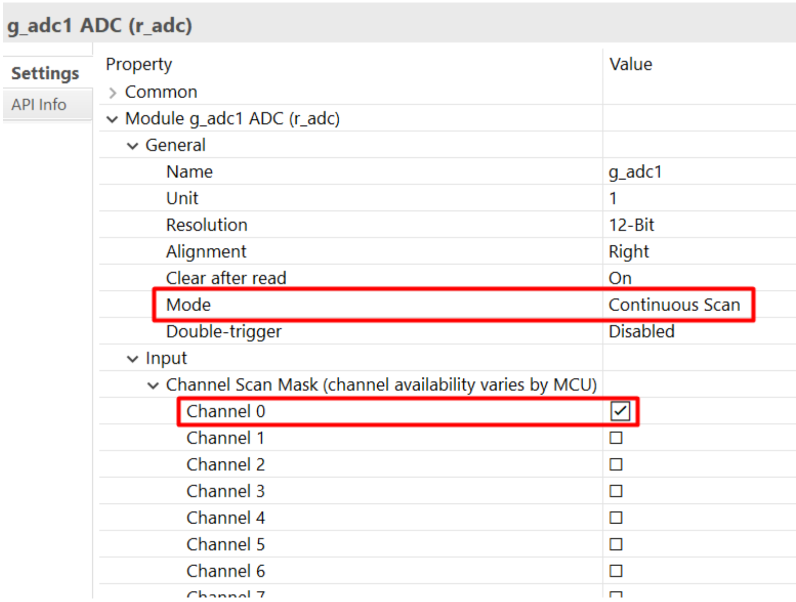

Figure 20. Configure ADC1 in Continuous Scan Mode, Channel 0 as Input

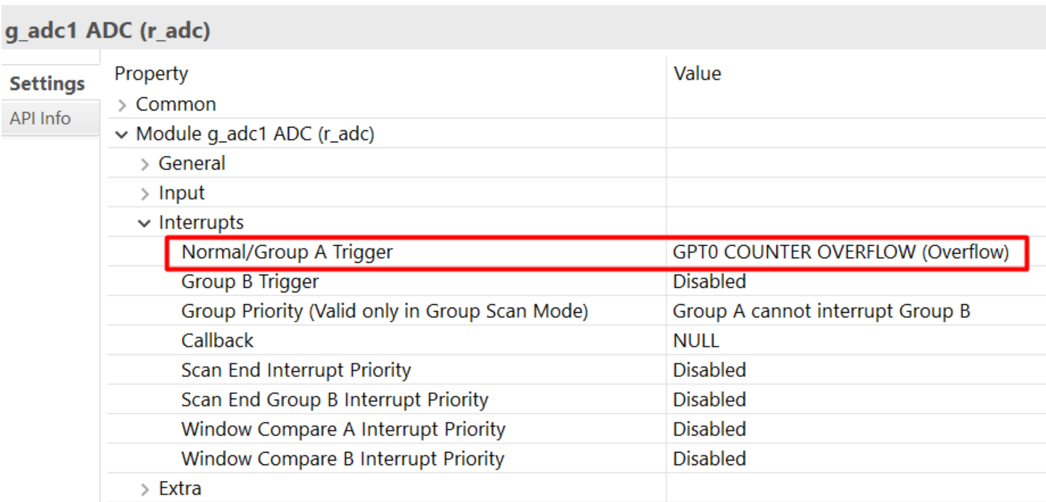

Figure 21. Set Trigger to Start ADC1 Conversion

6. DMAC Controller Setup

The ADC0 Scan End interrupt, and the ADC1 Scan End interrupt trigger DMAC0 and DMAC1 transfers, respectively.

The following figures show the detailed settings.

g_transfer_adc0 Transfer (r_dmac) ADC0 SCAN END (End of A/D scanning operation)

Settings | API Info

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_transfer_adc0 Transfer (r_dmac) ADC0 SCAN END (End of A/D scanning operation)	
Name	g_transfer_adc0
Channel	0
Mode	Repeat
Transfer Size	2 Bytes
Destination Address Mode	Offset addition
Source Address Mode	Fixed
Repeat Area (Unused in Normal Mode)	Source
Number of Transfers	512
Number of Blocks (Valid only in Repeat,Block or Repeat-Block Mode)	2
Activation Source	ADC0 SCAN END (End of A/D scanning operation)
Callback	g_transfer_adc0_cb
Transfer End Interrupt Priority	Priority 4
Interrupt Frequency	Interrupt after each block, or repeat size is transferred
Offset value (Valid only when address mode is \'Offset\')	4
Source Buffer Size	2

Figure 22. DMAC0 Configuration

g_transfer_adc1 Transfer (r_dmac) ADC1 SCAN END (End of A/D scanning operation)

Settings | API Info

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_transfer_adc1 Transfer (r_dmac) ADC1 SCAN END (End of A/D scanning operation)	
Name	g_transfer_adc1
Channel	1
Mode	Repeat
Transfer Size	2 Bytes
Destination Address Mode	Offset addition
Source Address Mode	Fixed
Repeat Area (Unused in Normal Mode)	Source
Number of Transfers	512
Number of Blocks (Valid only in Repeat,Block or Repeat-Block Mode)	2
Activation Source	ADC1 SCAN END (End of A/D scanning operation)
Callback	g_transfer_adc1_cb
Transfer End Interrupt Priority	Priority 5
Interrupt Frequency	Interrupt after each block, or repeat size is transferred
Offset value (Valid only when address mode is \'Offset\')	4
Source Buffer Size	2

Figure 23. DMAC1 Configuration

7. Application Code Highlights

This section details the highlights of the code in this application project. Figure 24 and Figure 25 show the code needed to initialize and configure DMAC channels. Both channels are configured to transfer converted data to the same buffer named g_buffer_adc.

```
/* Initialize DMAC instance and reconfigure for instance unit 0 */
err = init_hal_dmac(&g_transfer_adc0_ctrl, &g_transfer_adc0_cfg);
handle_error(err,"\r\n** init_hal_damc for unit 0 failed ** \r\n", ELC_DMAC0);
/* Update the DMAC ch 0 settings */
p_info_tmp = g_transfer_adc0_cfg.p_info;
p_info_tmp->p_src = (void const*) &R_ADC0->ADDR[ZERO];
p_info_tmp->p_dest = (void*) &g_buffer_adc[ZERO];
err = dmac_hal_reconfigure(&g_transfer_adc0_ctrl, p_info_tmp);
handle_error(err,"\r\n** dmac reconfiguration for unit 0 failed ** \r\n", ELC_DMAC0);
```

Figure 24. Initialize and Configure DMAC0 Transfer

```
/* Initialize DMAC instance and reconfigure for instance unit 1 */
err = init_hal_dmac(&g_transfer_adc1_ctrl, &g_transfer_adc1_cfg);
handle_error(err,"\r\n** dmac_init for unit 1 failed ** \r\n", ELC_DMAC01);
/* Update the DMAC ch 1 settings */
p_info_tmp = g_transfer_adc1_cfg.p_info;
p_info_tmp->p_src = (void const*) &R_ADC1->ADDR[ZERO];
p_info_tmp->p_dest = (void*) ((&g_buffer_adc[ZERO])+1);
err = dmac_hal_reconfigure(&g_transfer_adc1_ctrl, p_info_tmp);
handle_error(err,"\r\n** dmac reconfiguration for unit 1 failed ** \r\n", ELC_DMAC01);
```

Figure 25. Initialize and Configure DMAC1 Transfer

8. Board Setup

The EK-RA6M5 kit has a few switch settings that must be configured prior to running the application associated with this application note. In addition to these switch settings, the boards also contain a USB debug port and connectors to access the J-Link® programming interface.

Table 1. Switch settings for EK-RA6M5

Switch	Setting
J8	Jumper on pins 1-2
J9	Open

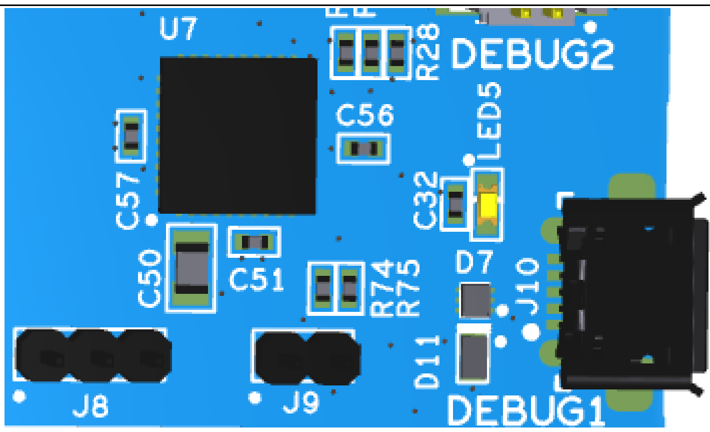

Figure 26. J8 and J9 on EK-RA6M5

Figure 27 shows a picture of the EK-RA6M5 kit.

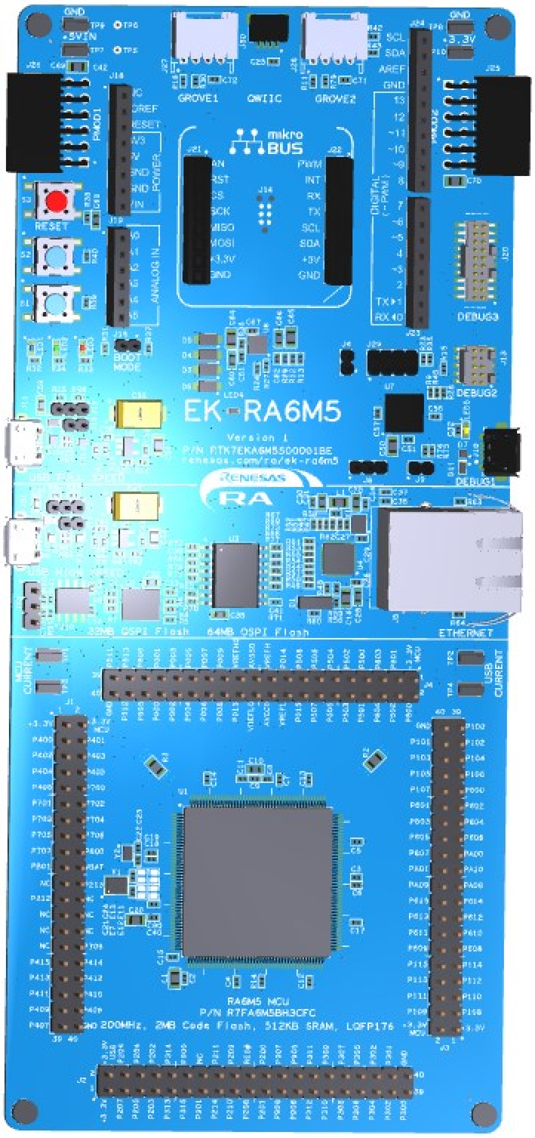

Figure 27. EK-RA6M5 Kit

9. Importing and Building the Project

To bring the application into the e² studio, follow these steps:

1. Launch e² studio.
2. In the workspace launcher, browse to the workspace location of your choice.
3. Close the Welcome window.
4. In e² studio, select **File > Import**.
5. In the Import Dialog Box, pick **Existing Projects into Workspace**.
6. Select the archive file bundled with this document.
7. Select the project and click **Finish**.
8. Open configuration .xml.
9. Click on **Generate Project Content** on the FSP configurator window.
10. Build the project.

10. Downloading the Executable to the EK-RA6M5 Kit

To connect and run the code, follow these steps:

1. Connect input signal to P000 (J4 - Pin 34). We used a waveform generator to generate input signals for testing.
2. Connect your PC to the USB port labeled DEBUG using a USB cable.
3. Go to **Run > Debug configurations.**
4. Click **Debug**. The program will break at the reset handler.
5. Click “Switch” to the **Debug perspective** when prompted by e^2 studio.
6. Click **Run** > **Resume**.

11. Verify ADC Conversion

After running the project, ensure that the **Memory** tab is open in the Console window, normally located at the bottom of the screen in Debug view. Click the small green plus (+) sign in the Monitors Pane to add a memory monitor. You should see a Monitor Memory dialog as shown in Figure 28. Enter the ADC buffer start address **&g_buffer_adc** and click OK.

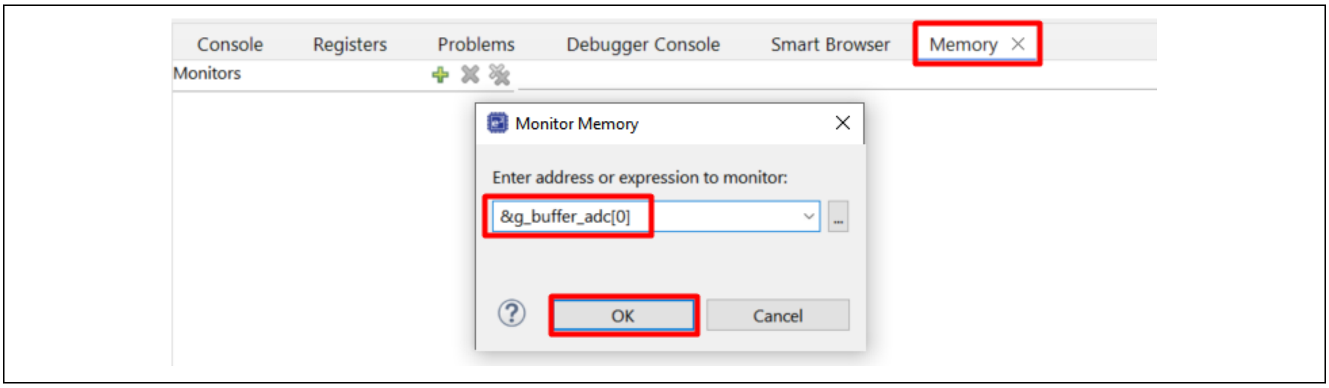

Figure 28. Memory Tab in e^2 studio

The Memory viewer shows the ADC buffer.

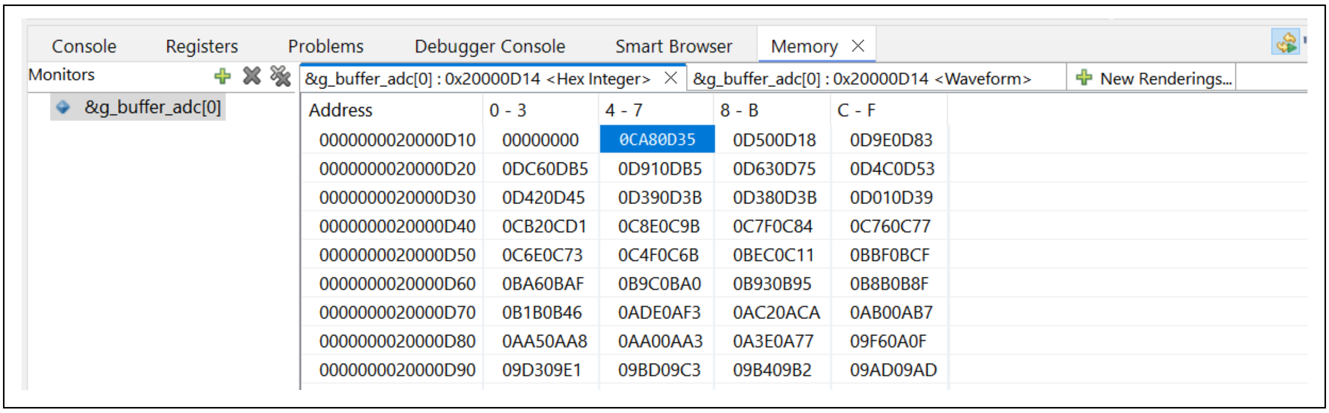

Figure 29. ADC Buffer in Hexadecimal

Click the (+) New Rendering and select Waveform rendering as shown in Figure 30.

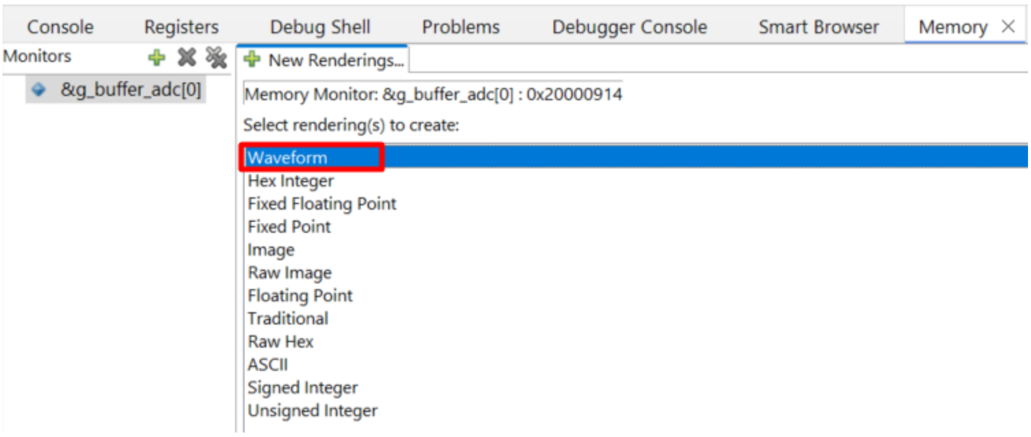

Figure 30. Waveform Rendering

Configure the waveform properties.

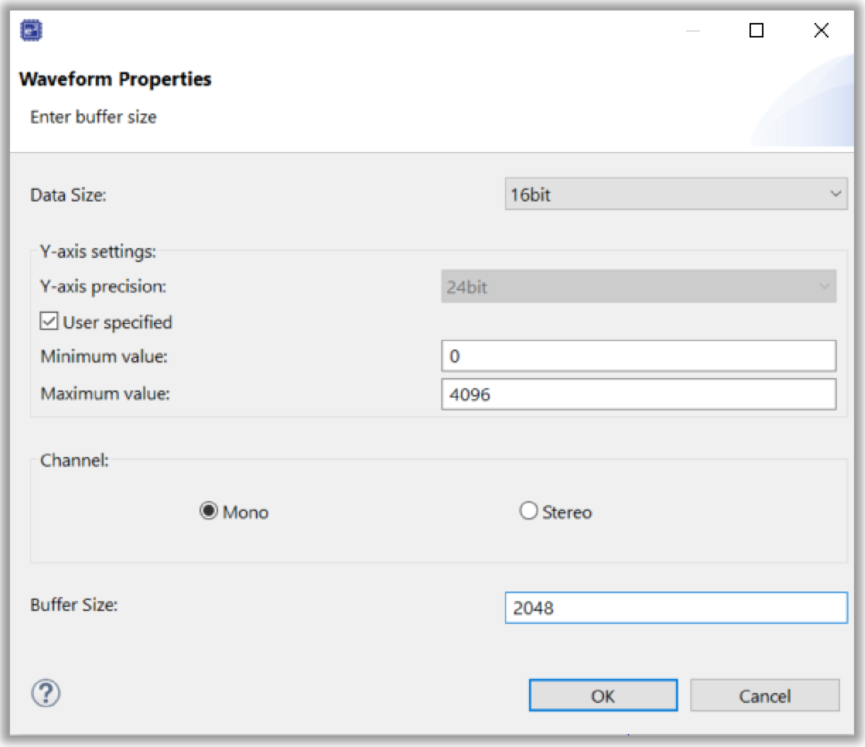

Figure 31. Configure Waveform Rendering

11.1 Rendering Sinewave on e² studio

The following figures show the captured Sinewave input from a waveform generator.

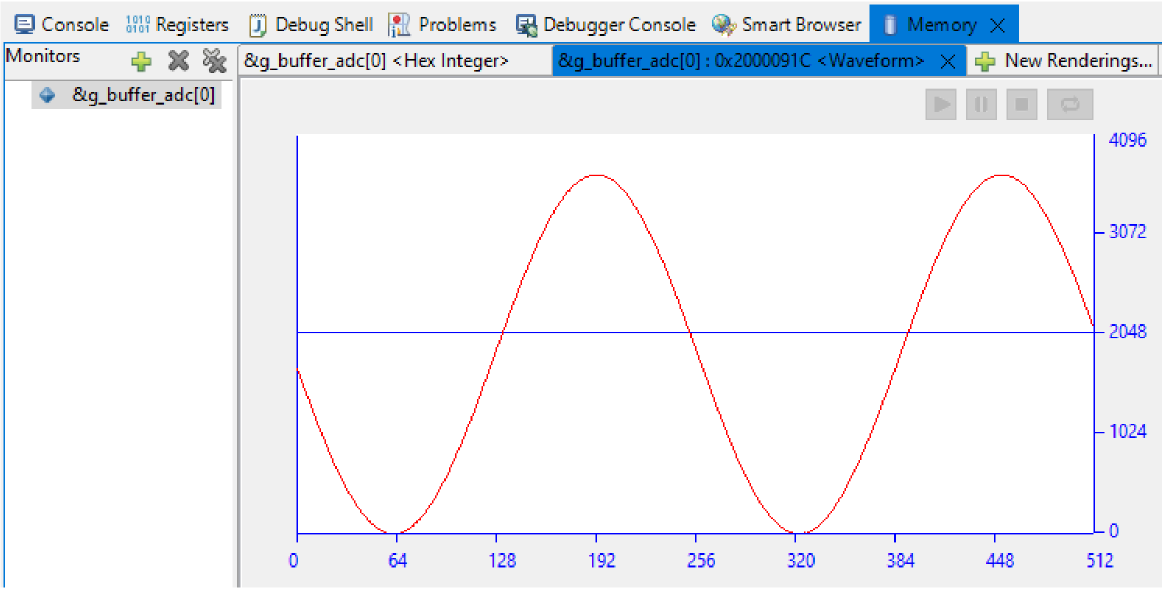

Figure 32. Rendering of Sinewave Input

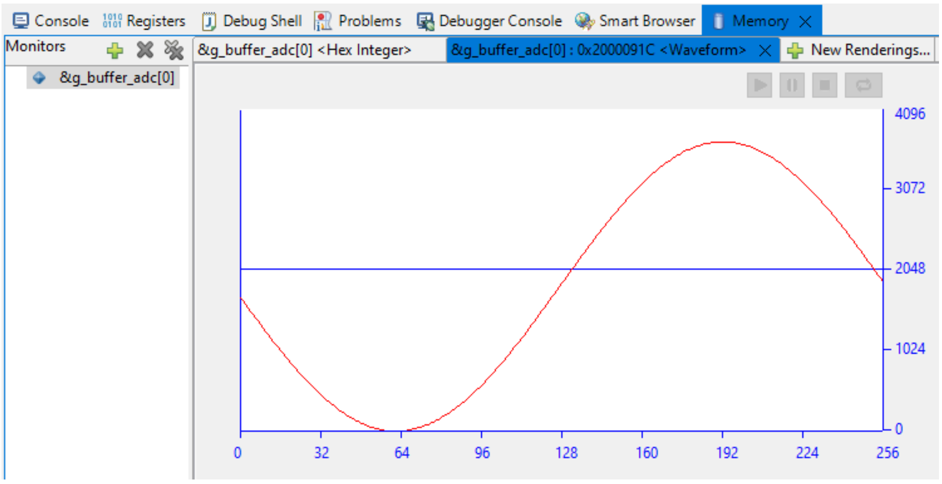

Figure 33. Rendering of Sinewave Input (Zoom in: a complete cycle)

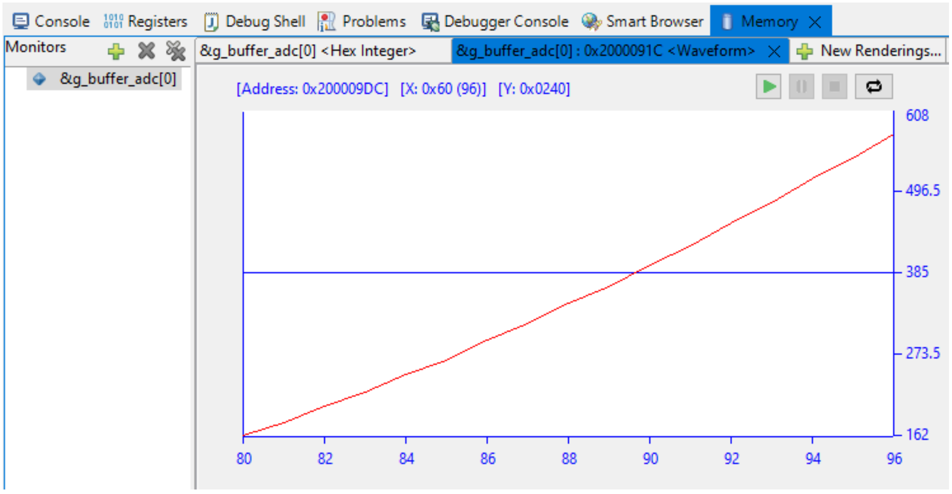

Figure 34. Rendering of Sinewave Input (Zoom in: 16-point data)

11.2 Rendering Ramp Input on e^2 studio

The following figures show captured Ramp input from a waveform generator.

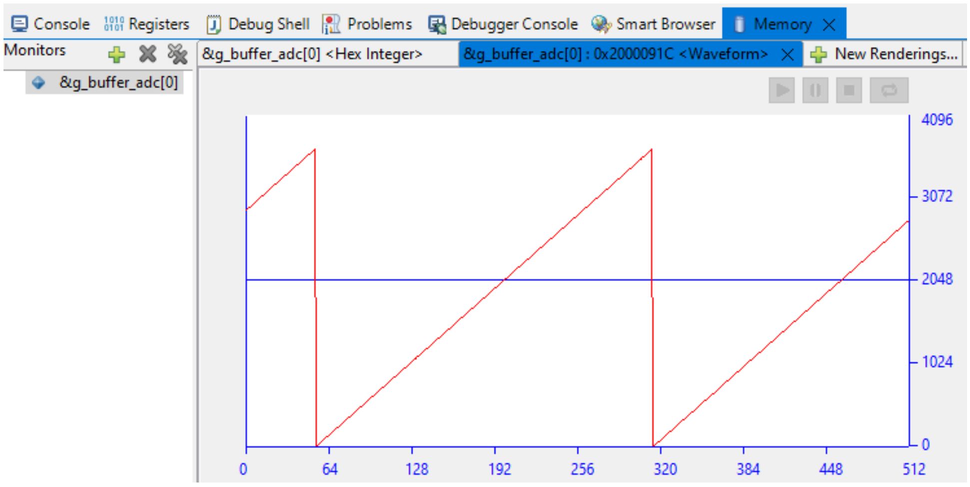

Figure 35. Rendering of Ramp Input

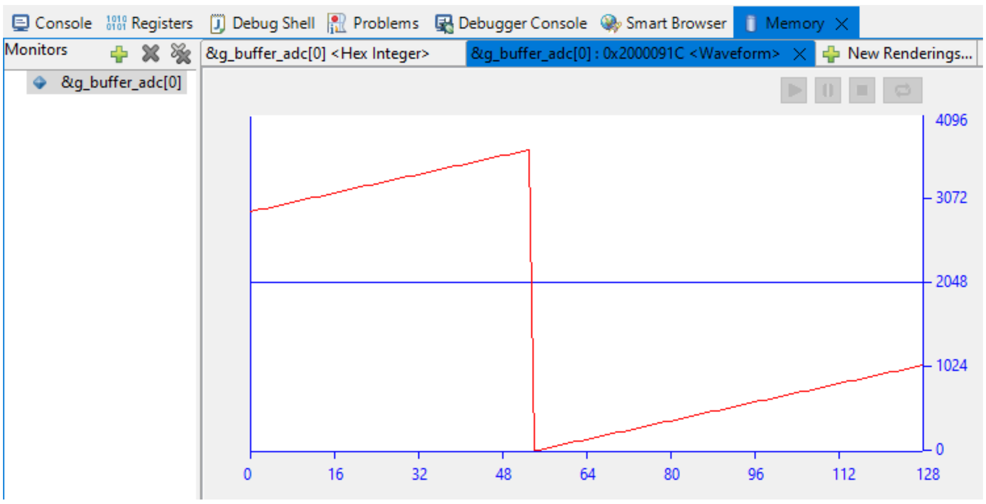

Figure 36. Rendering Ramp Input (Zoom in: a Full Cycle)

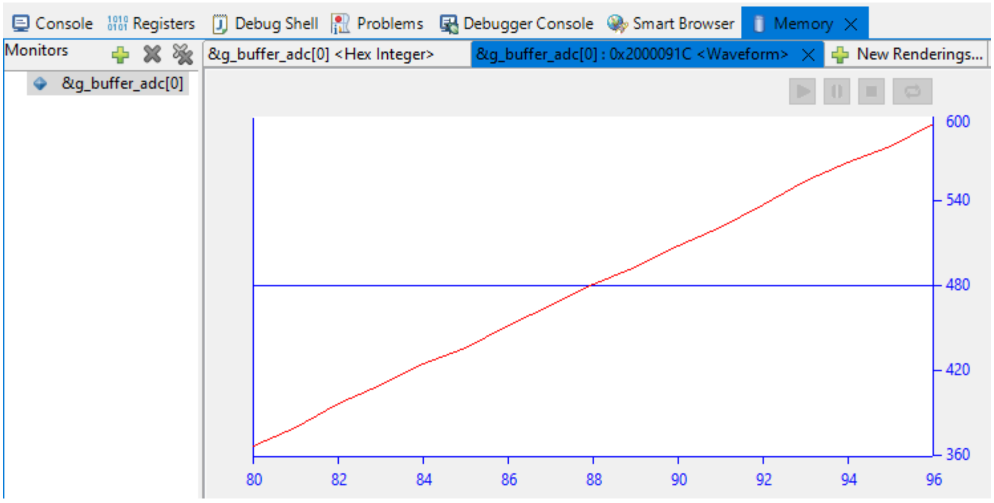

Figure 37. Rendering Ramp Input (Zoom in: 16 Data Points)

You can export the ADC buffer to various formats for further processing. Click the Export button marked in red in Figure 38 and select the data format.

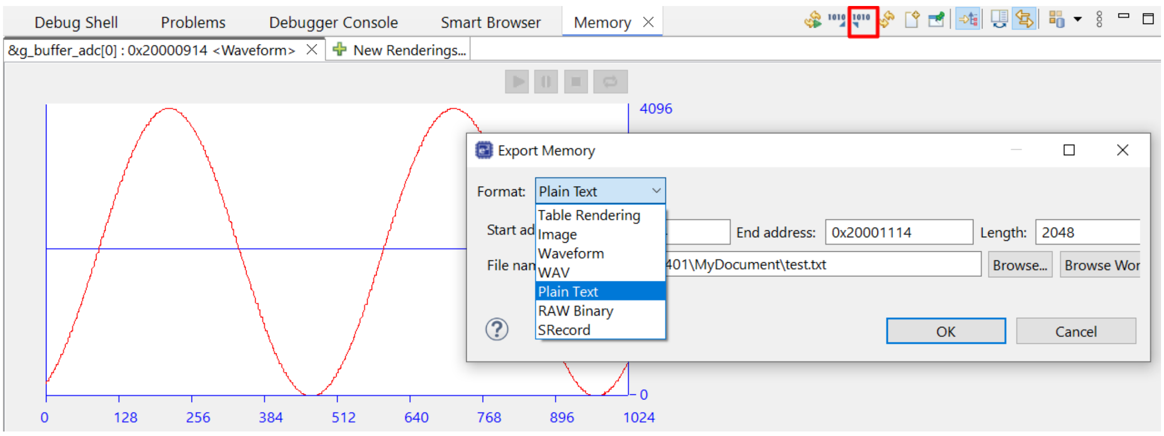

Figure 38. Select Data Format to Export Rendering Data

11.3 Set up J-Link RTT Viewer Output

To show information on the RTT Viewer, configure the SEGGER J-Link RTT Viewer as shown in Figure 39.

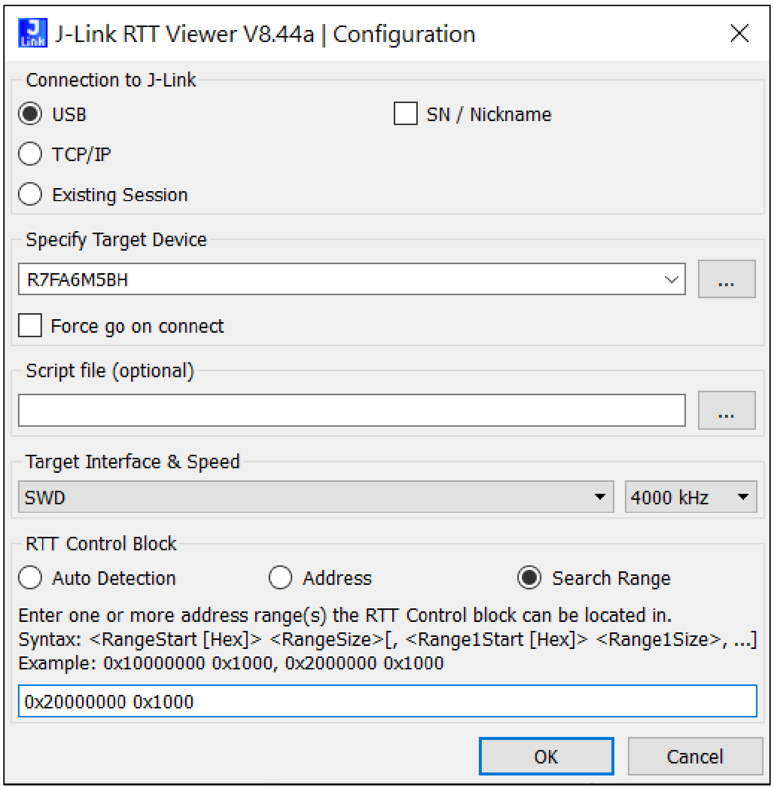

Figure 39. Configure SEGGER RTT Viewer

After completing the conversion of 2048 samples, the RTT Viewer stops and prints out the messages shown in Figure 40.

```
00>
00> ************************************************************************
00> *   Renesas FSP Application Project for ADC Interleave Mode            *
00> *   Application Project Version 1.0                                    *
00> *   Flex Software Pack Version  6.0.0                                  *
00> ************************************************************************
00> ADC Interleave Mode scan started...
00>
00> Scan completed. You can now open waveform rendering of memory in e2studio to observe the ADC output
```

Figure 40. SEGGER RTT Viewer Output

12. Change Project Configuration to Support ADC Single Scan Mode

By default, the application project is configured to work in ADC continuous scan mode. You can change it to work in ADC single scan mode by changing the GPT timer to periodic mode since ADC units need to be triggered repeatedly. Both ADC channels are configured in single scan mode.

The following figures show the settings of the GPT timer, ADC0, and ADC1 needed for ADC single scan mode.

g_timer0 Timer, General PWM (r_gpt)

Settings | API Info

Property	Value
⌄ Common	
Parameter Checking	Default (BSP)
Pin Output Support	Enabled
Write Protect Enable	Disabled
⌄ Module g_timer0 Timer, General PWM (r_gpt)	
⌄ General	
› Compare Match	
Name	g_timer0
Channel	0
Mode	One-Shot
Period	230
Period Unit	Nanoseconds
› Output	
› Input	
› Interrupts	
› Extra Features	
⌄ Pins	
GTIOC0A	None
GTIOC0B	None

Figure 41. GPT Timer in Periodic Mode

g_adc0 ADC (r_adc)

Settings | API Info

Property	Value
⌄ Common	
Parameter Checking	Default (BSP)
⌄ Module g_adc0 ADC (r_adc)	
⌄ General	
Name	g_adc0
Unit	0
Resolution	12-Bit
Alignment	Right
Clear after read	On
Mode	Continuous Scan
Double-trigger	Disabled
› Input	
› Interrupts	
› Extra	

Figure 42. ADC0 Configuration in Single Scan Mode

g_adc1 ADC (r_adc)

Settings | API Info

Property	Value
⌄ Common	
Parameter Checking	Default (BSP)
⌄ Module g_adc1 ADC (r_adc)	
⌄ General	
Name	g_adc1
Unit	1
Resolution	12-Bit
Alignment	Right
Clear after read	On
Mode	Continuous Scan
Double-trigger	Disabled
› Input	
› Interrupts	
› Extra	
⌄ Pins	

Figure 43. ADC1 Configuration in Single Scan Mode

13. Conclusion

You can utilize two ADC units that convert the same ADC input pin alternately to greatly increase the system's overall performance.

14. Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov.11.22	-	Initial version
1.01	Jun.28.23	-	Updated to FSP v4.4.0
1.02	May.01.24	-	Updated to FSP v5.2.0
1.03	Jul.09.25	-	Updated to FSP v6.0.0

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Corporate Headquarters

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

Contact information

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