Renesas Synergy™ Platform

ADC Periodic Framework Module Guide

Introduction

This module guide will enable you to effectively use a module in your own design. Upon completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application and write code, using the included application project code as a reference and efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are available on the Renesas Synergy Knowledge Base (as described in the References section at the end of this document) and should be valuable resources for creating more complex designs.

The ADC Periodic Framework provides a high-level API for signal processing applications. The module configures the ADC/SDADC to sample any of the available channels (using the single-scan mode) at a configurable rate and buffers the data for a configurable number of sampling iterations before notifying the application. The ADC Periodic Framework uses the ADC/SDADC, GPT or AGT and DTC peripherals on a Renesas Synergy™ Microcontroller. A user-defined callback can be created to process the data each time a new sample is available.

Contents

1. ADC Periodic Framework Module Features

- 24-bit Sigma-Delta A/D Converter (S1JA only).
- 16-bit A/D Converter (S1JA)
- 14-Bit A/D Converter (S3A7, S3A6, S3A3, S124, S128)
- 12-Bit A/D Converter (S7G2, S5D9, S5D5)
- Multiple Operation Modes
  - Single Scan
  - Group Scan
  - Continuous Scan
- Multiple Channels
  - 1 channel (S1JA)
  - 13 channels (unit 0), 12 channels (unit 1) (S7G2 and S5D9)
  - 13-channels (unit 0), 9 channels (unit 1) (S5D5)
  - 18 channels (S124)
  - 21 channels (S128)
  - 25 channels (S3A6)
  - 28 channels (S3A7)
  - Temperature sensor channel
  - Voltage sensor channel

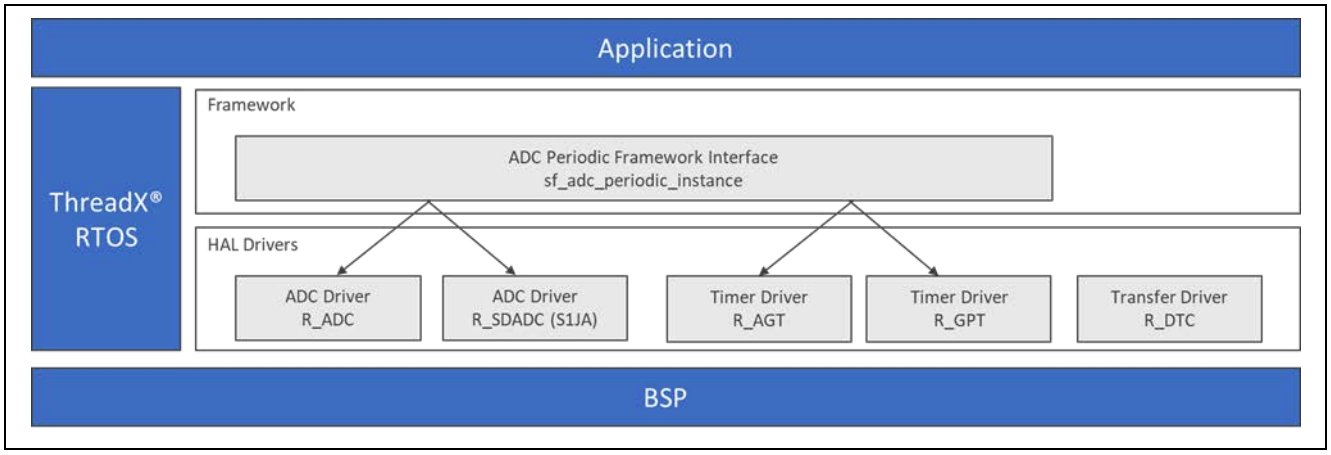

Figure 1. ADC Periodic Framework Module Block Diagram

2. ADC Periodic Framework Module APIs Overview

The ADC Periodic Framework defines APIs for opening, closing, starting, and stopping the ADC scans. A complete list of the available APIs, an example API call and a short description of each can be found in the following table. A table of status return values follows the API summary table.

Table 1. ADC Periodic Framework Module API Summary

Function Name	Example API Call and Description
open	`g_sf_adc_periodic.p_api->open(g_sf_adc_periodic.p_ctrl, g_sf_adc_periodic.p_cfg);` Acquires mutex, then initializes module at the HAL layer
start	`g_sf_adc_periodic.p_api->start(g_sf_adc_periodic.p_ctrl);` Starts the scan.
stop	`g_sf_adc_periodic.p_api->stop(g_sf_adc_periodic.p_ctrl);` Stops the hardware trigger (timer) from triggering any more ADC scans.
close	`g_sf_adc_periodic.p_api->close(g_sf_adc_periodic.p_ctrl);` Releases channel mutex and closes channel at HAL layer.
versionGet	`g_sf_adc_periodic.p_api->versionGet(&version);` Retrieve the API version using the version pointer.

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures, and function variables, review the *SSP User's Manual* API References for the associated module.

Table 2. Status Return Values

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_UNSUPPORTED	Command not found in the current menu
SSP_ERR_NOT_OPEN	Driver control block not valid. Call `SF_ADC_PERIODIC_Open` to configure.
SSP_ERR_ASSERTION	Version get error- `p_version` was NULL
SSP_ERR_INTERNAL	An internal ThreadX® error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.

Note: Lower-level drivers may return common error codes. Refer to the *SSP User's Manual* API References for the associated module for a definition of all relevant status return values.

3. ADC Periodic Framework Module Operational Overview

The ADC Periodic Framework module samples and buffers ADC data. The Framework notifies the application once the configured number of samples are buffered. The ADC Periodic Framework works as follows:

- After initial configuration and the scan process is started, the framework uses a hardware timer to trigger an ADC scan in one-shot mode. Each scan can consist of one or more channels and when each scan is completed, the ADC interrupt is intercepted by the DTC, which moves the result of the scan into the user buffer.
- Each scan is defined as a sampling iteration, and the number of samples generated for each scan is equal to the number of channels. If the channels are sequential (for example, channels 1, 2, 3, 4), the data is captured in order. If the channels are not in sequence (for example, channels 1, 3, 4, 5), then the samples generated by each scan also include data from the unused channels in between. Therefore, in the second example, five samples are stored to the user buffer each time.
- The user specifies the total number of sample iterations that need to occur before being notified. When the specified number of sampling iterations have occurred, and the data for each iteration has been stored into the user buffer, the user is notified via a callback with an index for the valid data in the buffer, and an event indicating that sampling for the specified number of iterations is complete.
- Unless the user stops the scan process, the scan continues to be triggered by the timer (using AGT or GPT), and data will be written into the user buffer, which is treated by the Framework as a circular buffer. The name and length of the buffer are specified via the ISDE configurator.

3.1 ADC Periodic Framework Module Important Operational Notes and Limitations

3.1.1 ADC Periodic Framework Module Operational Notes

1. At least one channel must be chosen while configuring the ADC/SDADC HAL module to avoid an API return error.
2. When configuring the scan rate for the ADC Periodic Framework (the GPT or AGT timer period), make sure that the period is long enough to accommodate scanning of all selected channels (about 2 microseconds for each channel conversion on a Synergy S7G2 MCU Group device).
3. The ADC Periodic Framework stores data for all the channels from each scan into the user-specified buffer. When the specified number of sample iterations is completed, the user is notified. If five channels are selected (channels 1, 2, 3, 4, 5) and the sample count is set to 3, the user will be notified when (5 x 3 =) 15 samples are available. The samples are ordered as follows:

Figure 2. ADC Periodic Operation – Sample Order

When selecting the data buffer length in the ADC Periodic Framework configuration, make sure that the buffer length is at least twice the length of the number of samples that will be generated (15 x 2 = 30 in this example). This is because once the user application is notified that the data is available, the framework keeps buffering in new data at the sample rate. Since the buffer is treated as a circular buffer, you can inadvertently overwrite the data. If the size is not larger than the number of samples generated, the data is overwritten before the application can use it.

The application callback has an index into the appropriate location in the buffer where valid data is present.

3.1.2 ADC Periodic Framework Module Limitations

- The ADC Periodic Framework does not currently support the following features:
  - The use of group-scan mode
  - The use of DMA
- When configuring the ADC channels to be used with this framework, the temperature and voltage sensors must not be selected if any of the other available channels are also selected. It is possible to use only the temperature sensor, only the voltage sensor, or any number of the regular ADC channels.
- ADC Periodic framework does not support DTC transfer when lower lever driver is SDADC.
- When using ADC Periodic framework with lower level SDADC of 24-bit, user should not access output data through "p_args" in callback function. User should access output data only through user defined buffer only.
- Refer to the most recent *SSP Release Notes* for any additional operational limitations for this module.

4. Including the ADC Periodic Framework Module in an Application

This section describes how to include the ADC Periodic Framework module in an application using the SSP configurator.

Note: It is assumed you are familiar with creating a project, adding threads, adding a stack to a thread, and configuring a block within the stack. If you are unfamiliar with any of these items, refer to the *Getting Started Guide for SSP* given in the References section at the end of this document to learn how to manage each of these important steps in creating SSP-based applications.

To add the ADC Periodic Framework to your application, simply add it to a project thread using the stacks selection sequence given in the following table. (The default name for the ADC Periodic Framework is `g_sf_adc_periodic0`. This name can be changed in the associated **Properties** window.)

Table 3. ADC Periodic Framework Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
`g_sf_adc_periodic` ADC Periodic Framework	Threads	New Stack> Framework> Graphics> ADC Periodic Framework on `sf_adc_periodic`

When the ADC Periodic Framework on `sf_adc_periodic` is added to the thread stack as shown in the following figure, the configurator automatically adds any needed lower-level modules. Any drivers that need additional configuration information will have box text highlighted in Red. Modules with a Gray band are individual modules that stand alone.

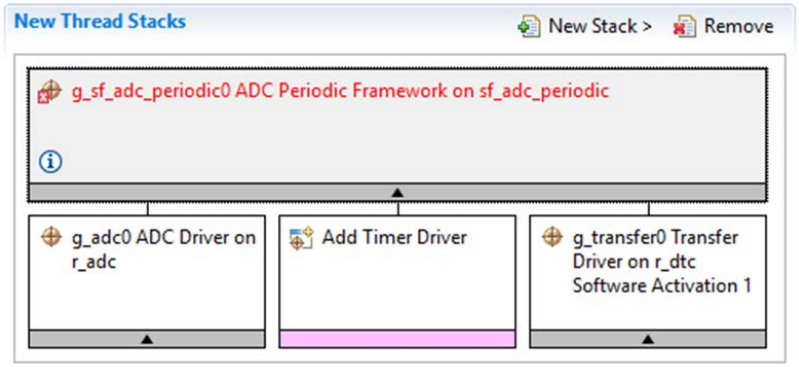

Figure 3. ADC Periodic Framework Module Stack

5. Configuring the ADC Periodic Framework Module

The ADC Periodic Framework Module must be configured by the user for the desired operation. The available configuration settings and defaults for all the user-accessible properties are given in the properties tab within the SSP configurator and are shown in the following tables for easy reference. Only properties that can be changed without causing conflicts are available for modification. Other properties are locked and not available for changes and are identified with a lock icon for the locked property in the Properties window in the ISDE. This approach simplifies the configuration process and makes it much less error-prone than previous manual approaches to configuration. The available configuration settings and defaults for all the user-accessible properties are given in the Properties tab within the SSP Configurator and are shown in the following tables for easy reference.

Note: You may want to open your ISDE, create the module and explore the property settings in parallel with looking over the following configuration table settings. This will help orient you and can be a useful hands-on approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for the ADC Periodic Framework Module on sf_adc_periodic

Parameter	Value	Description
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	Selects if code for parameter checking is to be included in the build
Name	Default: `g_sf_adc0_periodic0`	ADC Periodic Framework module name
Name of the data-buffer to store samples	Default: `g_user_buffer`	Name of the 16-bit data buffer to store samples.
Length of the data-buffer	Default: 128	Length of the buffer to which data is to be stored.
Number of sampling iterations	Default: 10	Number of samples captured per iteration

Parameter	Value	Description
Callback	Default: `g_adc_framework_user_callback`	User function that will be called once the number of sampling iterations of data has been buffered.
Name of generalized initialization function	sf_adc_periodic_init0	Name of initialization function
Auto Initialization	Enable, Disable (Default: Enable)	Auto initialization setting

Note: The example values and defaults are for a project using the Synergy S7G2 MCU Group. Other MCUs may have different default values and available configuration settings.

5.1 Configuration Settings for the ADC HAL Module

Table 5. Configuration Settings for the ADC HAL Module on r_adc

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	If selected code for parameter checking is included in the build.
Name	g_adc0	Module name
Unit	0, 1 (S7G2 Only) Default: 0	Specify the ADC Unit to be used. The S7G2 has two units; 0 and 1.
Resolution	14-Bit (S3A7/S124 Only), 12-Bit, 10-Bit (S7G2) Default: 8-Bit (S7G2 Only)	Specify the conversion resolution for this unit.
Alignment	Right, Left Default: Right	Specify the conversion result alignment.
Clear after read	Off, On Default: On	Specify if the result register must be automatically cleared after the conversion result is read. Note: If this is enabled, then watching the result register using a debugger always results in a 0.
Mode	Single Scan	The ADC Framework preconfigures and locks this field.
Channels 0-6	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 7-10 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 11-15 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 16-20	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.

ISDE Property	Value	Description
Channel 21 (Unit 0 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 22 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 23-27 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Temperature Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Temperature sensor use selection for Channel Scan Mask
Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Voltage sensor use selection for Channel Scan Mask
Normal/Group A Trigger	ELC Event	The ADC Framework preconfigures and locks this field.
Group B Trigger (Valid Only in Group Scan Mode)	ELC Event (The only valid trigger for either group in Group Scan Mode)	The ADC Framework preconfigures and locks this field.
Group Priority (Valid only in Group Scan Mode)	Group A cannot interrupt Group B, Group A can interrupt Group B; Group B scan restarts at next trigger, Group A can interrupt Group B; Group B scan restarts immediately, Group A can interrupt Group B; Group B scan restarts immediately and scans continuously Default: Group A cannot interrupt Group B	Do not use with ADC Framework since the mode is locked to Single Scan Mode.
Add/Average Count	Disabled, Add two samples, Add three samples, Add four samples, Add sixteen samples, Average two samples, Average four samples Default: Disabled	Specify if addition or averaging needs to be done for any of the channels in this unit. The actual channels are specified by using a channel mask adc_channel_cfg_t::add_mask.
Channels 0-27	Disabled, Enabled Default: Disabled	This field is valid only if adc_cfg_t::add_average_count is enabled. This field determines what channels results are to be averaged or summed.
Temperature Sensor	Disabled, Enabled Default: Disabled	Temperature sensor use selection for Addition/Averaging Mask
Voltage Sensor	Disabled, Enabled Default: Disabled	Voltage sensor use selection for Addition/Averaging Mask

ISDE Property	Value	Description
Channels 0-2	Disabled, Enabled Default: Disabled	Determines which of channels 0, 1 and 2 are using the updated sample-and-hold states value specified in adc_channel_cfg_t::sample_hold_states. This field must only be set if it is desired to modify the default sample and hold count value for channels 0, 1 and 2.
Sample Hold States (Applies only to the 3 channels selected above)	24	Specifies the updated sample-and-hold count for the channel dedicated sample-and-hold circuit. This field is valid only if adc_channel_cfg_t::sample_hold_mask is not 0. Only channels 0, 1 and 2 have dedicated sample and hold circuits. Note: Use this to modify the default number of states (24) for which the value is sampled. Each state is equal to 1/ADCLK time.
Callback	NULL	The ADC Framework uses the callback internally.
Scan End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Interrupt Priority selection
Scan End Group B Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Group B Interrupt Priority selection

Note The example settings and defaults are for a project using the Synergy S7G2 MCU Family. Other MCUs may have different default values and available configuration settings.

Table 6. Configuration Settings for the SDADC HAL Module on r_sdadc (only for S1JA)

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_adc0	Module name.
Mode	Single Scan, Continuous Scan Default: Continuous Scan	In single scan mode, all channels are converted once per start trigger, and conversion stops after all enabled channels are scanned. In continuous scan mode, conversion starts after a start trigger, then continues until stopped in software.
Resolution	16 Bit, 24 Bit Default: 24 Bit	Select 24-bit or 16-bit resolution.
Alignment	Right, Left Default: Right	Select left or right alignment.
Trigger	ELC Hardware Event, Software Default: Software	Select conversion start trigger. Conversion can be started in software, or conversion can be started when a hardware event occurs if the hardware event is linked to the SDADC peripheral using the ELC API.
Vref Source	Internal, External Default: Internal	Vref can be sourced internally and output on the SBIAS pin, or Vref can be input from VREFI.
Vref Voltage	0.8V, 1.0V, 1.2V, 1.4V, 1.6V, 1.8V, 2.0V, 2.2V, 2.4V Default: 1.0V	Select Vref voltage. If Vref is input externally, the voltage on VREFI must match the voltage selected within 3%.

ISDE Property	Value	Description
Internal Calibration During Open()	Enabled, Disable Default: Enabled	Calibration is required for all channels configured for differential input. Internal calibration is performed automatically during open for these channels unless it is disabled here.
Callback	NULL	Enter the name of the callback function to be called when conversion completes or a scan ends.
Conversion End Interrupt Priority	Priority 0 (highest), Priority 1, Priority 2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Select the interrupt priority for the conversion end interrupt. [Required]
Scan End Interrupt Priority	Priority 0 (highest), Priority 1, Priority 2, Priority 3 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for the scan end interrupt. [Required]
Calibration End Interrupt Priority	Priority 0 (highest), Priority 1, Priority 2, Priority 3 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for the calibration end interrupt. [Required]

Table 7. Configuration Settings for the AGT HAL Module on r_agt

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables parameter checking.
Name	g_timer0	Module name.
Channel	0	Physical hardware channel.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Microseconds	See Timer Period Calculation
Auto Start	False	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	The clock source for the AGT counter.
AGTO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTO pin). Set to false for no output of the timer signal.
AGTIO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTIO pin). Set to false for no output of the timer signal.
Output Inverted	True, False Default: True	Set to false to start the output signal low. Set to true to start the output signal high.
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Underflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Timer interrupt priority. 0 is the highest priority.

Note The example settings and defaults are for a project using the Synergy S7G2 MCU Family. Other MCUs may have different default values and available configuration settings.

Table 8. Configuration Settings for the GPT HAL Module on r_gpt

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	The ADC Framework preconfigures and locks this field based on channel selected in the ADC Framework.
Mode	Periodic	The ADC Framework preconfigures and locks this field.
Period Value	10	Configure timer period to trigger ADC scans.
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Milliseconds	Configure units of the timer period set above.
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	False	The ADC Framework preconfigures and locks this field.
GTIOCA Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	The ADC Framework preconfigures and locks this field.
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

Note The example settings and defaults are for a project using the Synergy S7G2 MCU Family. Other MCUs may have different default values and available configuration settings.

Table 9. Configuration Settings for the DTC HAL Module on r_dtc Software Activation

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	1	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

Note The example settings and defaults are for a project using the Synergy S7G2 MCU Family. Other MCUs may have different default values and available configuration settings.

5.2 ADC Periodic Framework Module Pin Configuration

To access a channel, ADC channels must be set in the Pins tab of the ISDE. The following table illustrates the method for selecting the pins within the SSP configuration window.

Table 10. Pin Selection for the ADC HAL Module

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select Peripherals > Analog ADC>ADC0\1>AN_XX
SDADC	Pins	Select Peripherals > Analog: SDADC > SDADC0 > AN_XX

Note: In the cases of the internal temperature sensor and the internal voltage sensor, there are no pin configurations required.

6. Using the ADC Periodic Framework Module in an Application

The key elements in constructing a simple ADC Periodic Framework module are selecting and configuring the stack, selecting a timer (GPT or AGT), defining the body of the callback function sf_adc_periodic_cfg_t::p_callback, initializing the framework, scanning the defined ADC channels, operating on the captured data as required by the application, rescanning periodically, and then stopping if the ADC measurements have been completed. The typical steps in using the ADC Periodic Framework module in an application are:

1. Initialize the ADC using the `open` API
2. Start the scan of channels using the `start` API
3. Stop the scan with the `stop` API
4. Read the results of the conversion using the `read` API
5. Close the instance using the `close` API

Often the scan is repeated periodically or stopped between scan operations. The following diagram shows common steps in a typical operational flow:

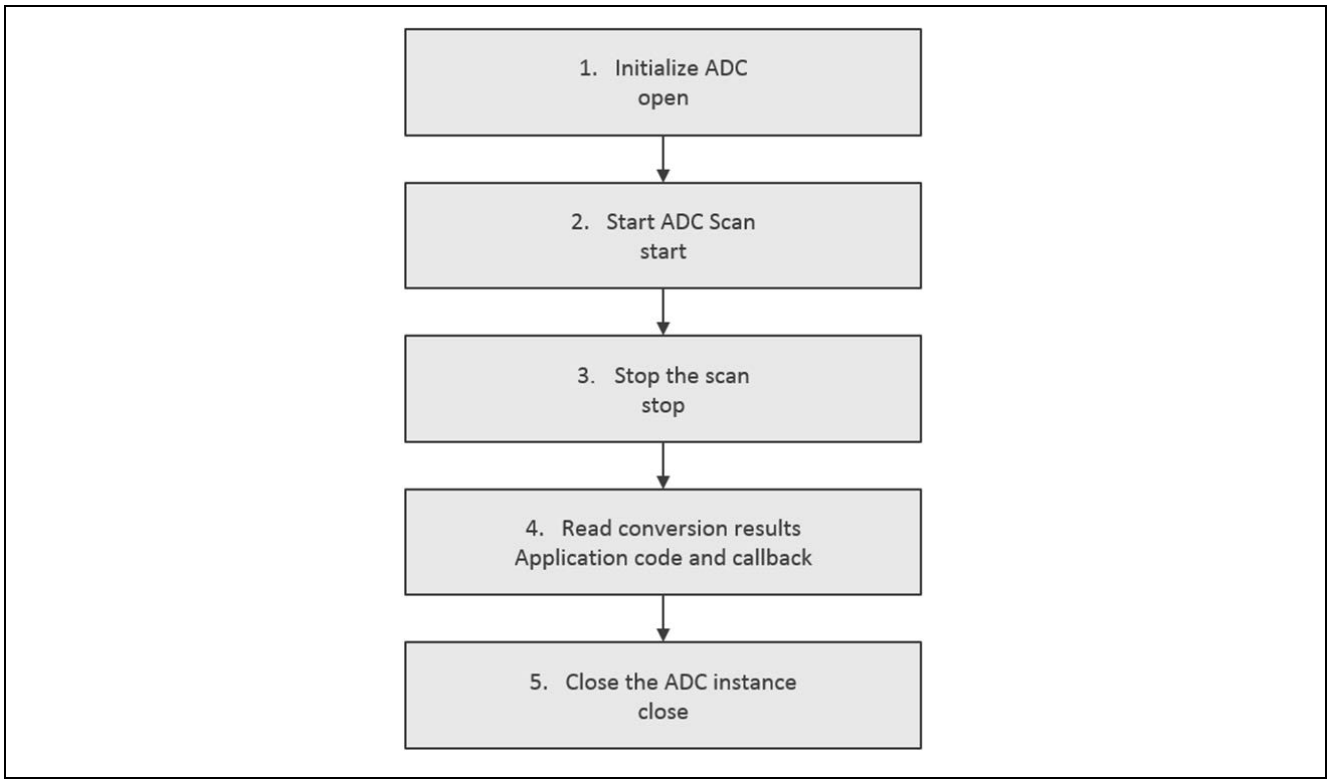

Figure 4. ADC Periodic Framework Module Application Project Flow Diagram

7. The ADC Periodic Framework Module Application Project

The application project associated with this module guide demonstrates the steps in a full design. The project can be found as described in the References section at the end of this document. You may want to import and open the application project within the ISDE and view the configuration settings for the ADC Periodic Framework module.

The complete application project can be found as described in the References section at the end of this document. The `adc_example_entry.c` file is located in the project once it has been imported into the ISDE. You can open this file within the ISDE and follow along with the description provided to help identify key uses of APIs.

The application project demonstrates the typical use of the ADC Periodic Framework APIs. The application project main thread entry initializes the ADC Periodic Framework and periodically scans the temperature sensor; the scan result is placed in a user-specified buffer. A user callback function is entered when the scan result is available; the user-specified callback function prints the result on the Debug Console using the common semi-hosting function.

Table 11. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	5.3.1 or later	Integrated Solution Development Environment
SSP	1.2.0 or later	Synergy Software Platform
IAR EW for Synergy	7.71.2 or later	IAR Embedded Workshop® for Renesas Synergy™
SSC	5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

A simple flow diagram of the application project is given in the following figure:

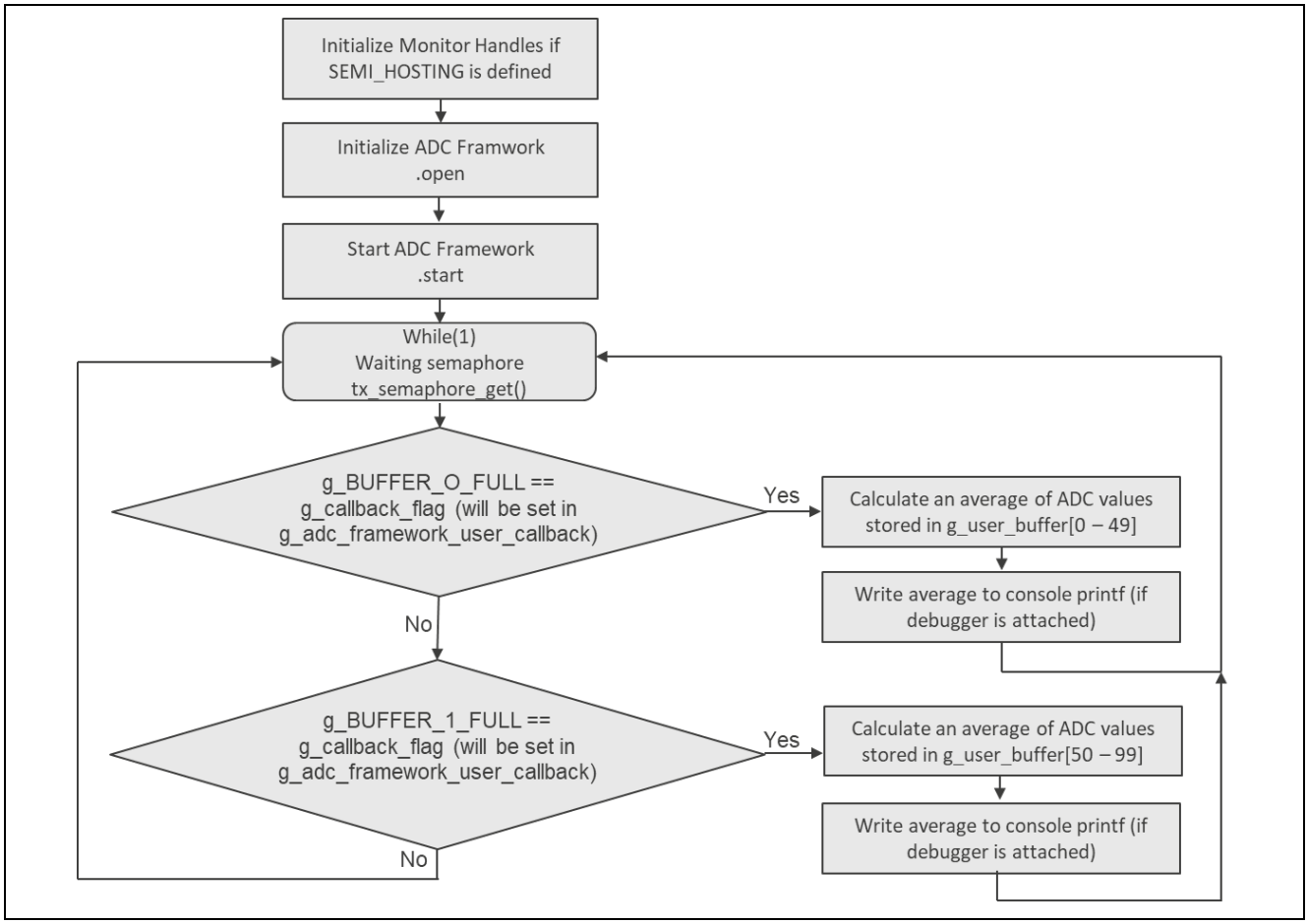

Figure 5. ADC Periodic Framework Module Application Project Flow Diagram

The file `adc_framework_mg_entry.c` file can be located in the project once it has been imported into the ISDE. Opening this file within the ISDE and following the description provided will identify the use of ADC PERIODIC FRAMEWORK APIs.

The user defined application code begins with the inclusion of the header files which reference the ADC instance structure and external function declaration. Following this, global variables that are required by the application are defined. These are `bsp_leds_t g_LEDs;` used in determining the LEDs of the S7G2-SK board and `g_tsn_temperature,` used for the calculation of the device temperature.

A basic error handling function (**`error_trap`**) is then defined. This error handling function will be called if any of the ADC PERIODIC FRAMEWORK API do not return successfully. Execution of the application will stop in this error handling function, via the software break instruction `__BKPT (1);` If the use of SEMI_HOSTING has been defined then an error message will be displayed in the virtual debug window (e² studio) / terminal I/O window (IAR-EW).

The application function `void adc_framework_mg_entry(void)` is now defined. If SEMI_HOSTING and the GCC tool chain are defined, and if the On-chip debugger is attached, the function `initialise_monitor_handles();` will be called.

Next, the API to initialize the ADC PERIODIC FRAMEWORK is called, with the corresponding control and configuration pointers being passed in: `g_sf_adc_periodic0.p_api->open(g_sf_adc_periodic0.p_ctrl, g_sf_adc_periodic0.p_cfg);` If the .open does not open successfully then the error_trap function will be called.

Next, three API that are application specific are called. These API are not required for the ADC PERIODIC FRAMEWORK to operate but enable the user to interact with the application. Two API for opening external IRQ are called and an API to identify the LEDs of the development board. If these API do not return successfully then the error_trap function will be called.

Next, the API to start the ADC PERIODIC FRAMEWORK is called, with the corresponding control pointer being passed in: `g_sf_adc_periodic0.p_api->start(g_sf_adc_periodic0.p_ctrl);` If the .open does not open successfully then the error_trap function will be called.

The application code now enters a `while(1)` processing loop. The application will wait indefinitely for an event flag: `tx_event_flags_get(&g_buffer_button_event_flags, ALL_EVENT_FLAGS, TX_OR_CLEAR, &actual_flags, TX_WAIT_FOREVER);`

The possible event flags (defined in the file `#include "adc_framework_mg_entry.h"`) are set in either the ADC PERIODIC FRAMEWORK callback functions, indicating that a data buffer has valid data, or the external IRQ callback functions indicating that a push button has been pressed.

A switch statement is used to determine which event flag has been set.

If the user push button S4 has been pressed, the event flag `EVENT_FLAG_0x1_S4_BUTTON` will be set and the API to stop the ADC PERIODIC FRAMEWORK is called: `g_sf_adc_periodic0.p_api->stop(g_sf_adc_periodic0.p_ctrl);`

If the user push button S5 has been pressed, the event flag `EVENT_FLAG_0x2_S5_BUTTON` will be set and the API to start the ADC PERIODIC FRAMEWORK is called: `g_sf_adc_periodic0.p_api->start(g_sf_adc_periodic0.p_ctrl);`

If the ADC PERIODIC FRAMEWORK buffer has valid data, the callback function will set an event flag indicating which buffer to process. The callback will set `EVENT_FLAG_0x4_BUFFER_0_49_FULL` or `EVENT_FLAG_0x8_BUFFER_50_99_FULL.` From this data the temperature of the device is calculated and if SEMI_HOSTING is enabled, the temperature result is displayed in the virtual debug window (e2 Studio) / terminal window (IAR-EW). The temperature value based upon each set of buffer data is shown. If the push buttons are pressed a message indicating the state will be displayed. The temperature format of C of F can be set by the define in `adc_framework_mg_entry.h`

Note: It is assumed you are familiar with using `printf()` with the Debug Console in the Synergy Software Package. If you are unfamiliar with this, refer to the How do I Use `Printf()` with the Debug Console in the Synergy Software Package available as described in the References section at the end of this document. The user can see results via the watch variables in the debug mode.

A few key properties are configured in this application project to support the required operations and the physical properties of the target board and MCU. The properties with the values set for this specific project are listed in the following tables. You can also open the application project and view these settings in the Properties window as a hands-on exercise.

Table 12. ADC Periodic Framework Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_sf_adc_periodic0
Name of the data buffer to store samples	g_user_buffer
Length of the buffer	100
Number of Sampling iterations	50
Callback	g_adc_framework_user_callback
Name of generated initialization function	sf_adc_periodic_init0
Auto Initialization	Disable

Table 13. g_adc0 ADC Driver on r_adc Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_adc0
Unit	0
Resolution	12-bit
Alignment	Right
Clear after read	On
Internal Calibration During Open()	Disabled
Channel Scan Mask >Temperature Sensor	Used in Normal/ Group A
Scan End Interrupt Priority	Priority 8

Table 14. g_timer0 Timer Driver on r_gpt

ISDE Property	Value Set
Name	g_timer0
Channel	0
Period Value	20
Period Unit	Milliseconds

Table 15. g_transfer0 Transfer Driver on r_dtc ADC COMPARE MATCH

ISDE Property	Value Set
Name	g_timer0

Table 16. g_external_irq10 External IRQ Driver on r_icu

ISDE Property	Value Set
Name	g_external_irq10
Channel	10
Trigger	Falling
Digital Filtering	Enabled
Digital Filtering Sample Clock	PCLK / 64
Interrupt enabled after initialization	True
Callback	cb_external_irq10
Pin Interrupt Priority	Priority 10

Table 17. Pin Configuration Settings for g_external_irq10 External IRQ Driver on r_icu

Pin Configuration Property	Value
Ports ⟶ P0 ⟶ P005	
Mode	Input Mode
IRQ	IRQ10-DS

Table 18. g_external_irq11 External IRQ Driver on r_icu

ISDE Property	Value Set
Name	g_external_irq11
Channel	11
Trigger	Falling
Digital Filtering	Enabled
Digital Filtering Sample Clock	PCLK / 64
Interrupt enabled after initialization	True
Callback	cb_external_irq10
Pin Interrupt Priority	Priority 12

Table 19. Pin Configuration Settings for g_external_irq11 External IRQ Driver on r_icu

Pin Configuration Property	Value
Ports ⟶ P0 ⟶ P006	
Mode	Input Mode
IRQ	IRQ11-DS

8. Customizing the ADC Periodic Framework Module for a Target Application

Some configuration settings will normally be changed by the developer from those shown in the application project; for example, the user can easily change the configuration settings for the ADC clock by updating the PCLKC in the **Clocks** tab. The user can also change the ADC port pins to select the desired analog input; this can be done using the **Pins** tab in the configurator. The ADC application project uses the on-chip temperature sensor to read the die temperature.

9. Running the ADC Periodic Framework Module Application Project

To run the ADC Periodic Framework application project and to see it executed on a target kit, you can simply import it into your ISDE, compile and run debug.

To implement the ADC Periodic Framework application in a new project, follow the steps for defining, configuring, auto-generating files, adding code, compiling and debugging on the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical, while just reading over the guide will tend to be more theoretical.

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters in the *SSP User's Manual* available as described in the References section at the end of this document.

To create and run the ADC Periodic Framework application project, simply follow these steps:

1. Refer to *Importing a Renesas Synergy Project* (11an0023eu0116-synergy-ssp-import-guide.pdf, included in this package) for instructions on importing the project into e^2 studio ISDE or IAR Embedded Workbench® for Renesas Synergy™ and building/running the application.
   NOTE: If using the GCC compiler, as the application example sends floating point numbers to the virtual debug window via printf, ensure that the Linker setting "Use float with nano printf (-u _printf_float)" is selected, as shown in figure 7.
2. Connect to the host PC via a micro USB cable to J19 on the SK-S7G2 Kit.
3. Start to debug the application.

The output can be viewed in the Debug Console. You can place your thumb on the chip or blow air over the chip to see variation in temperature readings.

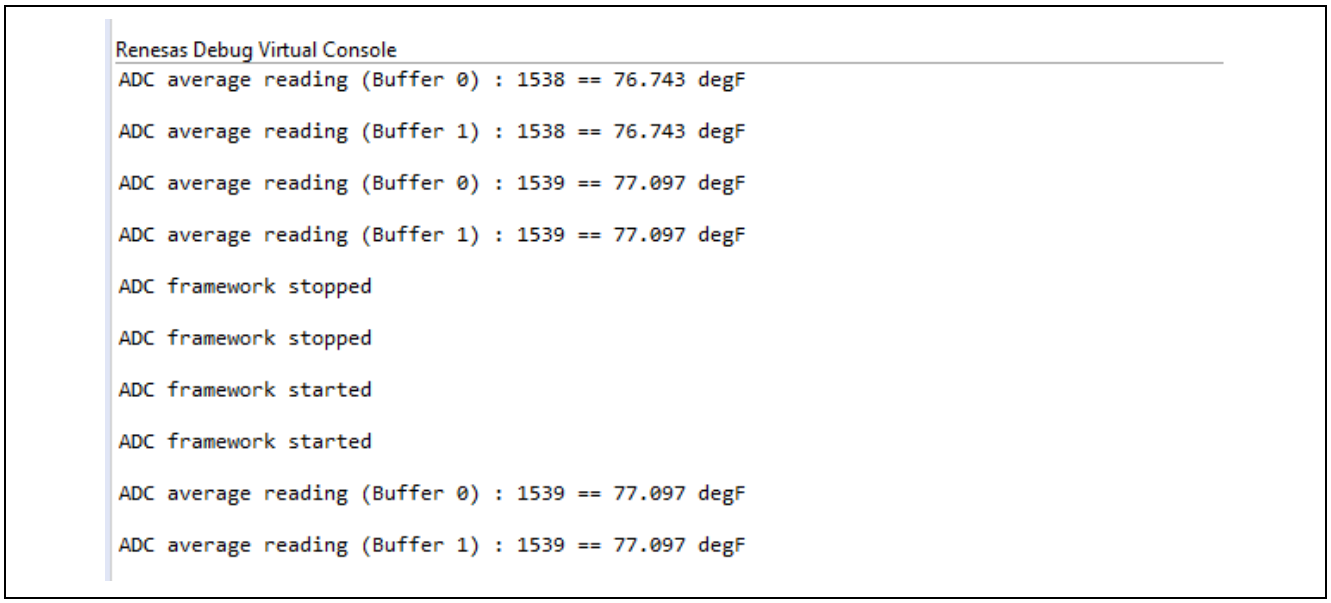

Figure 6. Example Output from ADC Periodic Framework Application Project (E2Studio, °F selected)

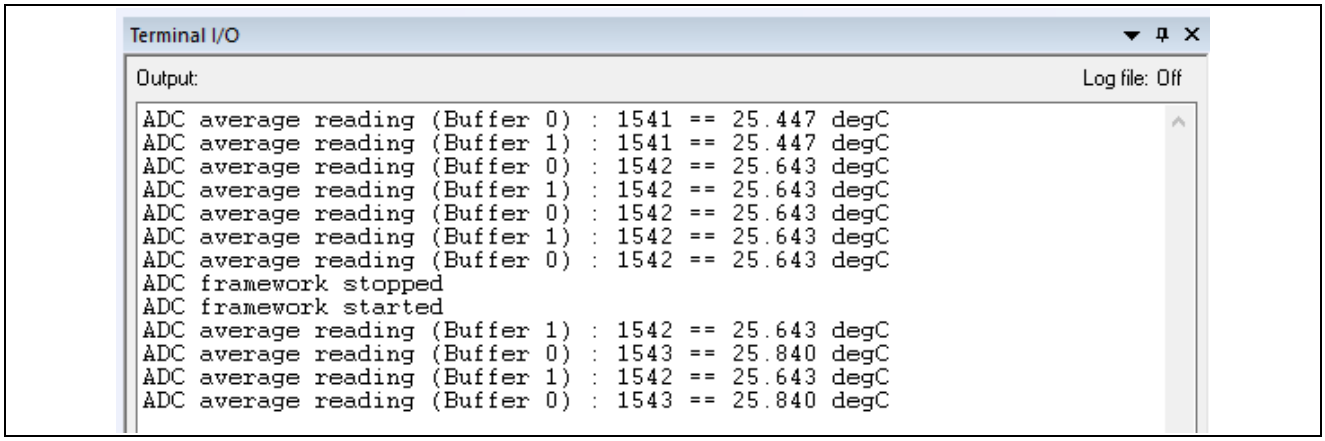

Figure 7. Example Output from ADC Periodic Framework Application Project (IAR-EW, °C selected)

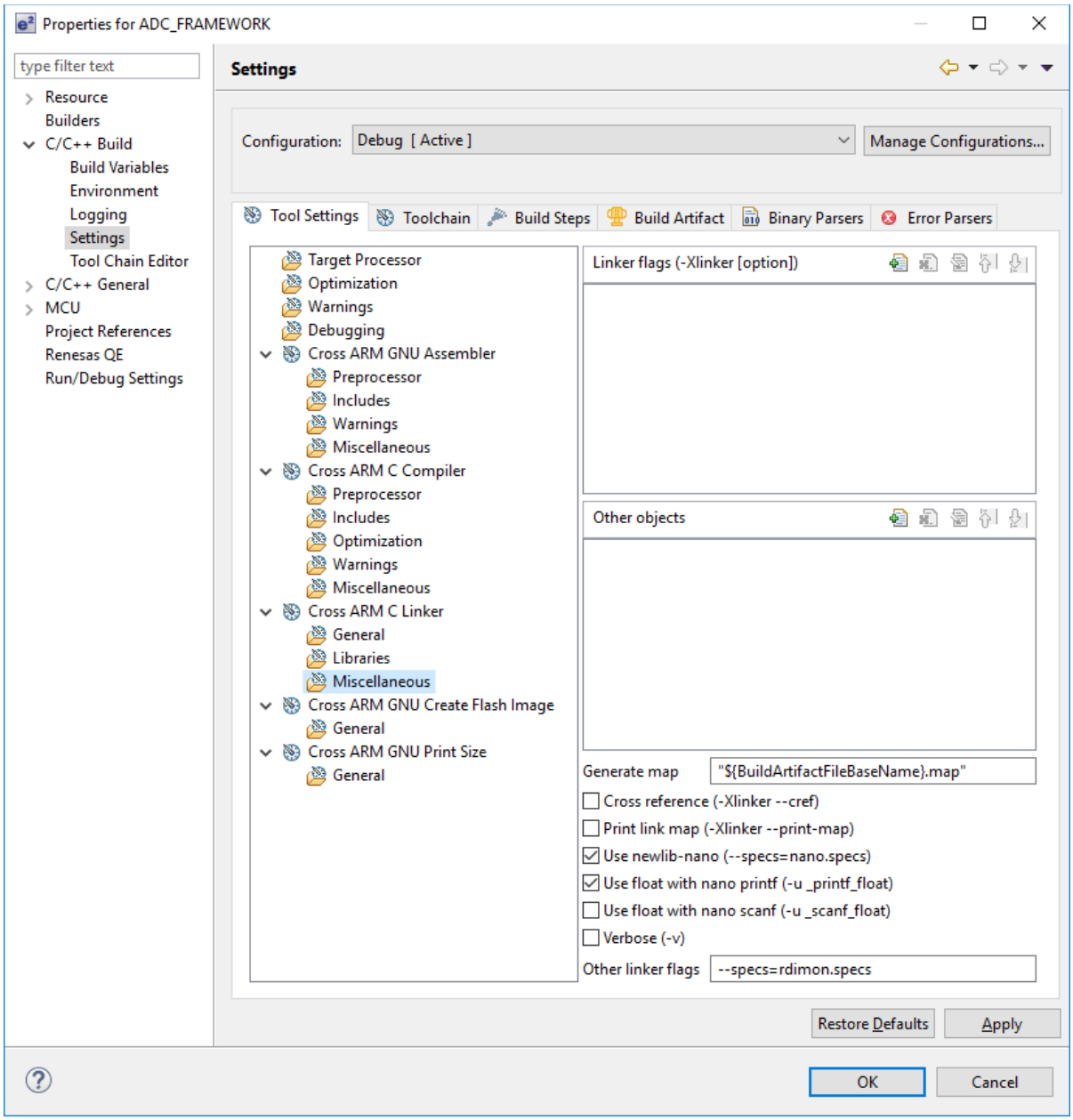

Figure 8. Enabling float with nano printf

10. ADC Periodic Framework Module Conclusion

This module guide has provided all the background information needed to select, add, configure, and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy™ Platform makes these steps much less time consuming and removes the common errors, like conflicting configuration settings or the incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrates additional development time savings by allowing work to begin at a high level and avoiding the time required in older development environments to use or, in some cases, create, lower-level drivers.

11. ADC Periodic Framework Module Next Steps

After you have mastered an ADC Periodic Framework module project, you may want to review a different example. You may find that the ADC HAL module is a better fit for your target application. The ADC HAL module guide illustrates the use of the ADC in a different implementation. Other application projects and application notes that demonstrate ADC use are available as described in the References section at the end of this document.

12. ADC Periodic Framework Module Reference Information

SSP User Manual: Available in html format in the SSP distribution package and as a pdf from the Synergy Gallery.

Links to all the most up-to-date sf_adc_periodic module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977526.

Website and Support

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

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.22.17	-	Initial Release
1.01	Aug.30.17		Update to Hardware and Software Resources Table
1.02	Mar.02.19		Updated application project and descriptions

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

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

Contact information

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