RL78/G24

FAA Oversampling A/D Conversion Library Installation Guide

Introduction

This sample application describes the Oversampling A/D Conversion Library (hereinafter referred to as the Oversampling A/D Conversion Library) using the Flexible Application Accelerator (FAA). The FAA is built into the RL78/G24 microcontroller and operates as a processor independent of the CPU.

Target Device

RL78/G24

If applying this application note to other microcontrollers, modify it according to their specifications and evaluate thoroughly.

Related Documents

- RL78/G24 User's Manual: Hardware (R01UH0961)
- RL78 family User's Manual: Software (R01US0015)
- RL78/G24 Fast Prototyping Board User's Manual (R20UT5091)
- RL78 Smart Configurator User's Gude: CS+ (R20AN0580)
- RL78 Smart Configurator User's Gude: e2 studio (R20AN0579)
- Flexible Application Accelerator (FAA) Tool Guide: CS＋ (R01AN7094)
- Flexible Application Accelerator (FAA) Tool Guide: e2 studio (R01AN7095)

Contents

1. Features of the Oversampling A/D Conversion Library

This chapter explains the features of the Oversampling A/D Conversion Library.

1.1 Oversampling A/D Conversion Control

This library performs oversampling on a specified single analog input channel.

Oversampling is a technique to pseudo-enhance the resolution of an A/D converter. The smallest identifiable analog input voltage of an A/D converter, i.e., the ratio of analog input voltage per digital output bit, is called 1 LSB, which represents the resolution of A/D conversion. This library performs 12-bit A/D conversion, stores the conversion results in a buffer, and uses that buffer for oversampling processing to reduce noise, resulting in more accurate and higher resolution A/D conversion results. Additionally, parameters such as A/D conversion time and analog input channel switching are possible.

1.2 Flexible Application Accelerator (FAA)

The processing to achieve oversampling A/D conversion control is performed by the FAA. The FAA is built into the RL78/G24 microcontroller and operates as a processor independent of the CPU. This allows oversampling A/D conversion control processing to be performed without occupying the CPU.

For detailed information on the FAA, please refer to the "RL78/G24 User's Manual: Hardware (R01UH0961)".

2. Tools Available for the Oversampling A/D Conversion Library

This library is provided as source code through the code generation function of the Smart Configurator. The generated source code is added to the user project and built within the integrated development environment. Table 2-1 lists the tools available for using this library. For the usage procedure, refer to section 5 Procedure for Using the Oversampling A/D Conversion Library.

Table 2-1 Tools Available for the Oversampling A/D Conversion Library

Item	Description
integrated development environment (IDE)	e^2 studio CS+ for CC
Smart Configurator (SC)	Renesas Smart Configurator for RL78
Compiler	CC-RL

3. Folder Structure of the Oversampling A/D Conversion Library

The folder structure of this library is shown below. Each file is provided through the Smart Configurator's code generation feature.

Table 3-1 Folder Structure

Folder / File name	Description
smc_gen	Smart Configurator generated folder
\{ConfigName}	FAA component folder
{ConfigName}_common.c	Source file for FAA common functions
{ConfigName}_common.h	Header file for FAA common functions
{ConfigName}_common.inc	Include file for FAA common functions
{ConfigName}_ADC_OverSampling.c	Source file for the oversampling A/D conversion function
{ConfigName}_ADC_OverSampling.h	Header file for the oversampling A/D conversion function
{ConfigName}_src.dsp	FAA DSP assembler source file

4. Resource Used

4.1 Peripherals

This library uses the following peripherals:

- Flexible Application Accelerator (FAA)
- A/D Converter

4.2 ROM/RAM Size

For reference, the ROM/RAM sizes when built with the following options are shown below.

Compiler options

-cpu=S3 -character_set=utf8 –Odefault

Link options

-NOOPtimize

Table 4-1 ROM/RAM Size [byte]

ROM	RAM	FAACODE	FAADATA
1843	7	92	1060

5. Procedure for Using the Oversampling A/D Conversion Library

The procedure for using this library is described below.

5.1 CC-RL Environment

5.1.1 Integrated Development Environment

Please download e2 studio and CS+ from the Renesas website.

[Renesas Website]

https://www.renesas.com/en/products/software-tools/tools.html

For basic operations of e2 studio, refer to the following user's manual.

- e2 studio Integrated Development Environment User's Manual: Getting Started Guide (R20UT2858)

For basic operations of CS+, please refer to the following user's manual:

- CS+ Integrated Development Environment User's Manual: Project Operation (R20UT4691)

5.1.2 Smart Configurator

Please download "RL78 Smart Configurator" and "CS+ RL78 Smart Configurator Communication Plugin" from the URL below. The CS+ RL78 Smart Configurator Communication Plugin is required to register the source code generated by Smart Configurator into CS+.

https://www.renesas.com/rl78-smart-configurator

After launching the installer, please follow the installer's instructions to complete the installation. The installation must be performed with administrator privileges.

5.1.3 Oversampling A/D Conversion Library

This section describes the procedure for using the Oversampling A/D Conversion Library in Smart Configurator. For basic operating instructions of Smart Configurator, refer to the user guide listed below.

- RL78 Smart Configurator User's Guide: e^2 studio (R20AN0579)
- RL78 Smart Configurator User's Guide: CS+ (R20AN0580)

5.1.3.1 Launching Smart Configurator

Create a new project or load an existing project in CS+, then launch Smart Configurator from the CS+ screen.

5.1.3.2 Clock Settings

Select the "Clocks" page of the "Smart Configurator View" and set the "Timer clock" and " f_{CLK}" outputs as follows.

- Timer clock：96000[kHz]
- f_{CLK}：48000[kHz]

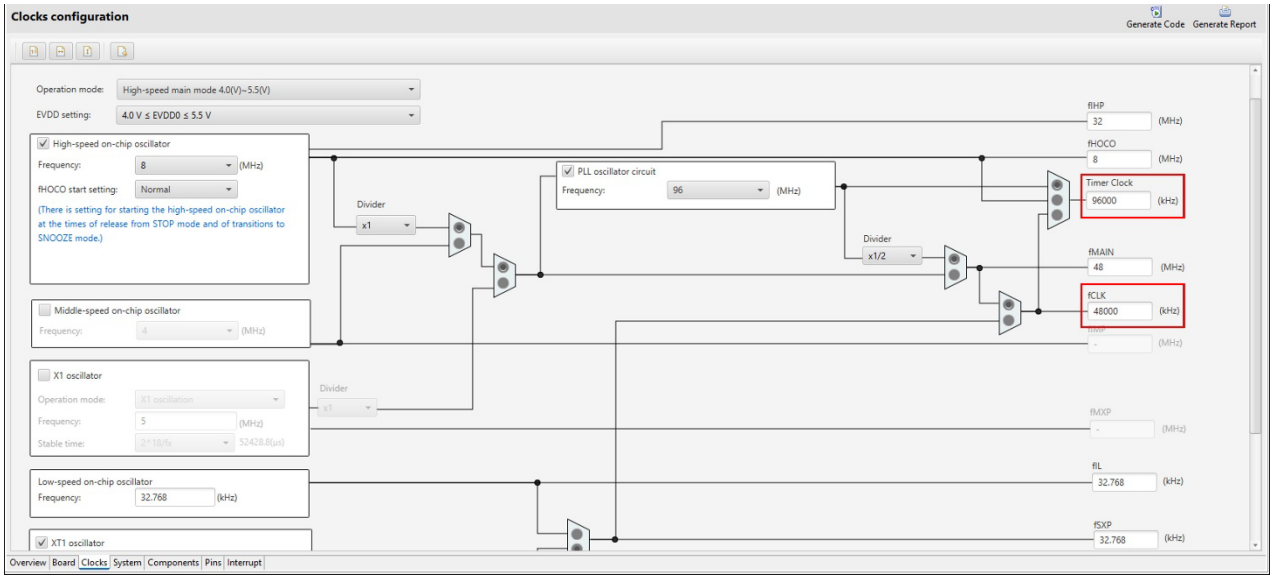


Figure 5-1 Clock Settings

5.1.3.3 Adding FAA Components

Select the "Components" page of the "Smart Configurator View" and click the "Add component" button.

Next, add the "Flexible Application Accelerator" component from the "Software Component Selection" page.

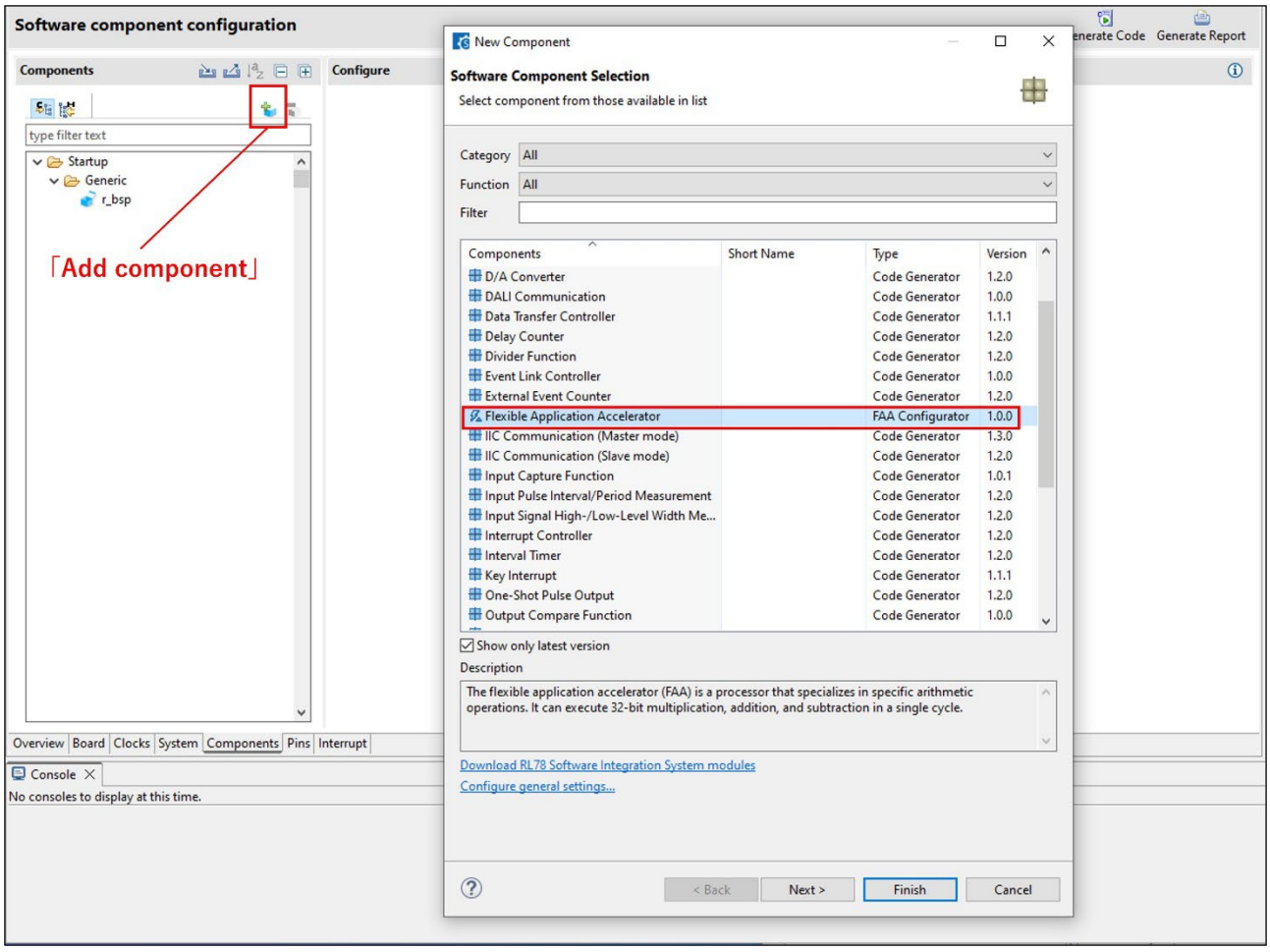
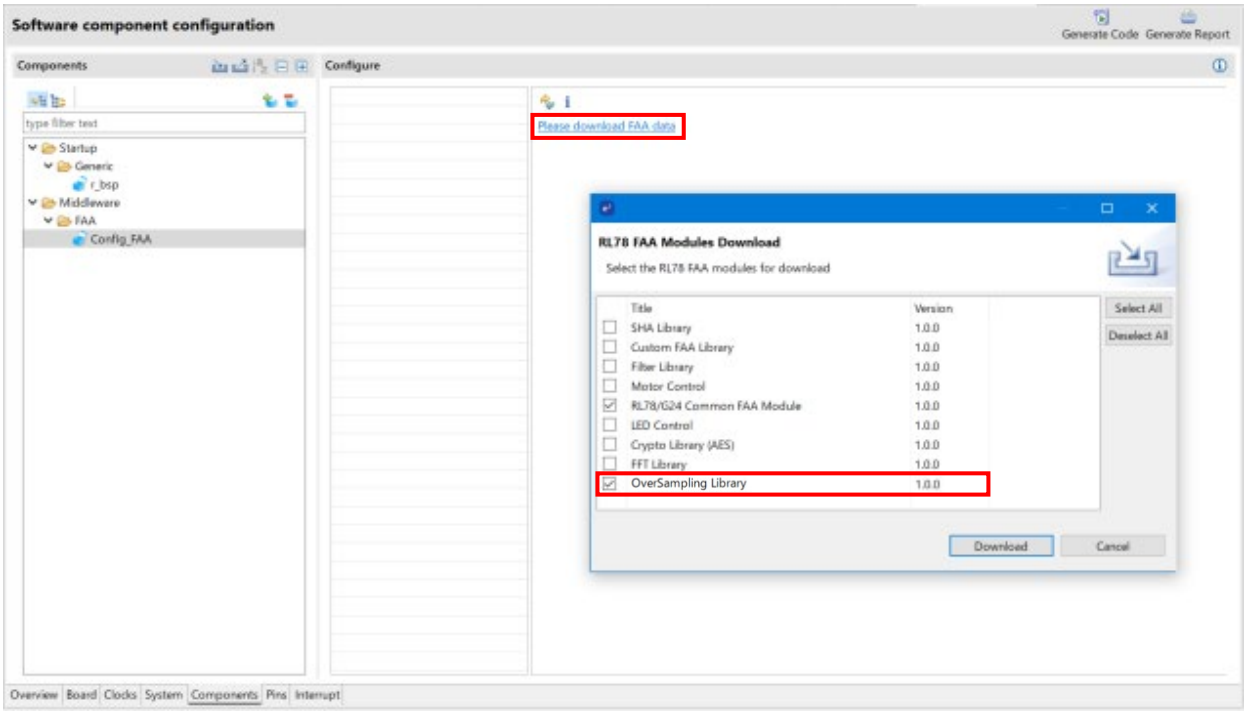


Figure 5-2 Adding FAA Components

5.1.3.4 Downloading FAA Modules

Click on "Please download FAA data" displayed on the screen to see the FAA modules available for download. Select "OverSampling Library " to download.



Figrue 5-3 Downloading FAA Modules

5.1.3.5 FAA Module Configuration

When you select the “ADC OverSampling” module from the list of downloaded FAA modules, the configuration screen is displayed.
Configure the settings according to your user environment. Table 5-1 shows the details of the configuration items.

Table 5-1 List of Smart Configurator Setting Items

Item	Value	Description
ADC VREF(+)	VDD/AVREFP	Selects the positive reference voltage of the A/D converter.
ADC VREF(-)	VSS/AVREFM	Selects the negative reference voltage of the A/D converter

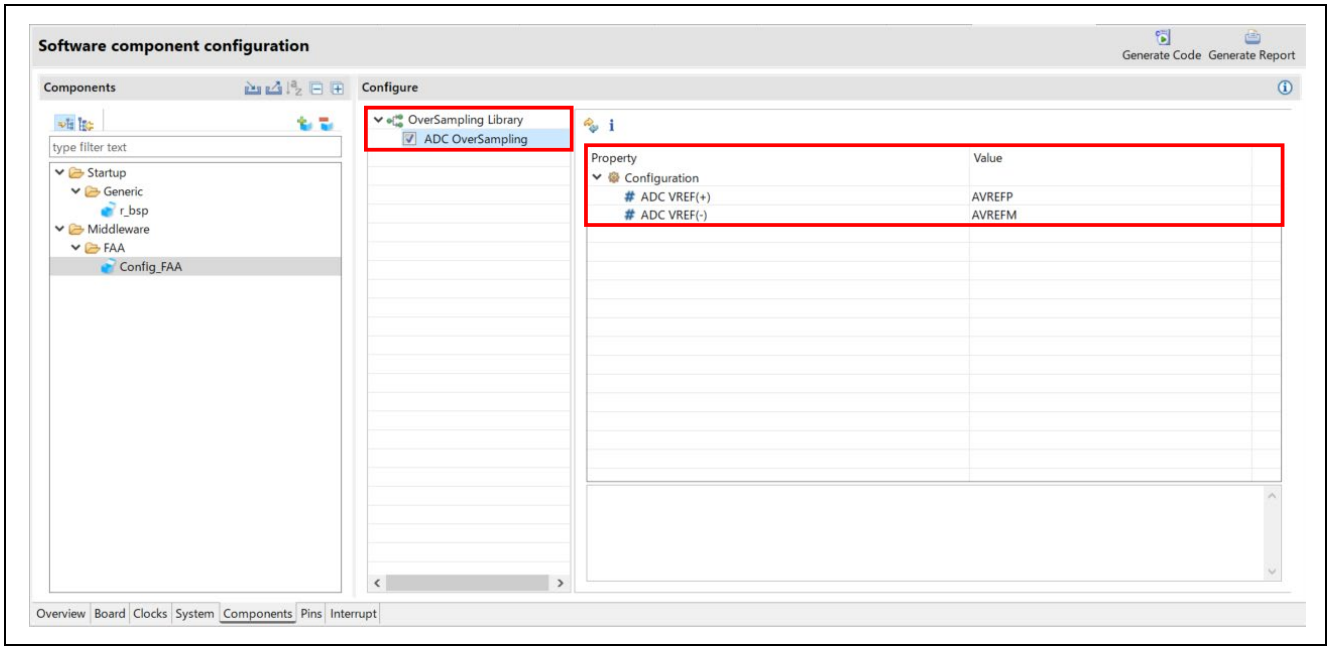


Figure 5-4 FAA Module Configuration

5.1.3.6 Adding the A/D Converter Component

This library uses the A/D converter component of Smart Configurator, so add the A/D converter component. Select “Advanced Mode” as the operating mode.

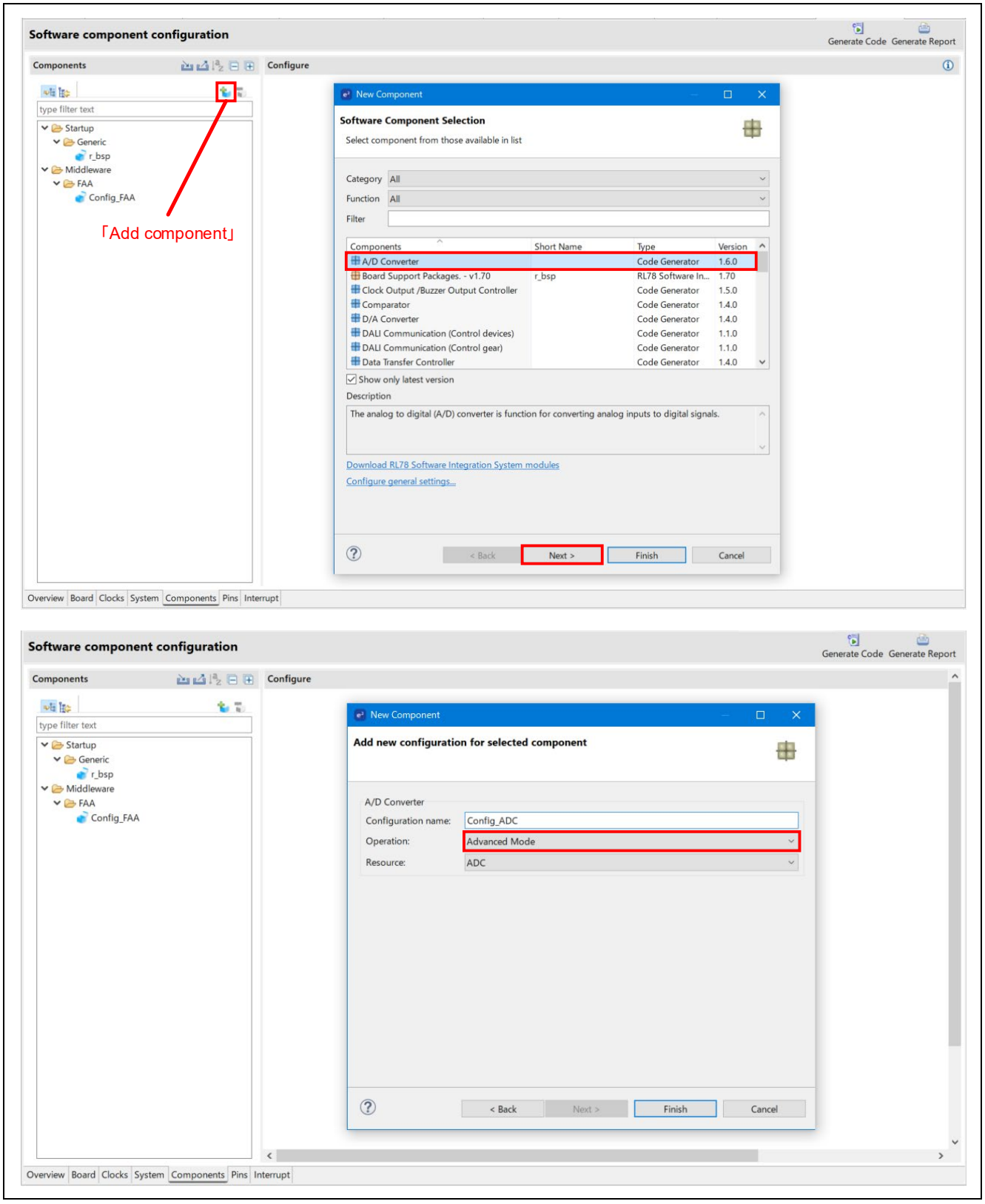


Figure 5-5 Adding the A/D Converter Component

5.1.3.7 Code Generation

After setting the configuration, click the "Generate code" button to generate the source code for the LED control library. The generated source code will be automatically registered in the user project on e^2 studio and CS+.

Implement the API function calls provided by the library in the user application. For details on the various API functions, see 7.4 API Function Specification.

6. Library Specifications (FAA Program)

This section describes the oversampling A/D conversion control provided by this library.
The control processing is implemented in the “{ConfigName}_src.dsp” file and is executed by the FAA.

6.1 List of Constants

Table 6-1 lists the constants provided by this library.

Table 6-1 Constants Used in the Library

Constant name	Setting Value	Description
_C_ADControl_FAAAP_ADM3	#ADM3_PTR （014H）	Setting value used when accessing the ADM3 register via the FAA address pointer.
_C_ADControl_ADTRSWT	80H	Setting value to generate a software trigger (ADM3 setting value).
_C_ADControl_FAAAP_ADINST	#ADINTST_PTR （029H）	Setting value used when accessing the ADINTST register via the FAA address pointer.
_C_ADControl_ADINTST_ST0S	02H	Value used to determine whether the A/D conversion completed successfully.
_C_ADControl_FAAAP_ADCR0	#ADCR0_PTR （020H, 021H）	Setting value used when accessing the ADCR0 register via the FAA address pointer.
_C_ADControl_ADINTST_ST0S_CLEAR	01H	Setting value used to clear the status of the A/D conversion result.

6.2 List of Variables

Table 6-2 lists the variables provided by this library.

Table 6-2 Variables Used in the Library

Size	Variable name	Description
4 Byte	_V_OverSampling_Result	Variable that stores the oversampled value calculated from 256 A/D conversion results.
4 Byte	_V_OverSampling_Start	Variable that stores A/D conversion values for 256 samples.
4 Byte	_V_OverSampling_End	Variable indicating the end of the buffer used to store 256 A/D conversion values.
4 Byte	_V_ADControl_ADINTST_ST0S	A/D conversion completion check.
4 Byte	_V_Buffer_Full	Address of the end of the buffer used to store A/D conversion values for 256 samples.

6.3 List of Functions

Table 6-3 lists the functions (process) provided by this library.

Table 6-3 List of Processes

Label name	Summary	Source file
_P_Pointer_Init	Initializes the address pointer.	Config_FAA_src.dsp
_P_ADControl	A/D conversion processing.	Config_FAA_src.dsp
_P_OverSampling	Oversampling processing.	Config_FAA_src.dsp

6.4 Function Specifications

The function specifications of this library are shown below.

FAA Program

[Label name] _P_Pointer_Init

Summary	Address pointer initialization
Header	Config_FAA_common.inc
Declaration	—
Description	Initializes the address pointer.
Arguments	None
Return value	None
Remarks	None

[Label name] _P_ADControl

Summary	A/D conversion processing
Header	Config_FAA_common.inc
Declaration	—
Description	Performs A/D conversion on the specified analog input channel.
Arguments	None
Return value	None
Remarks	None

[Label name] _P_OverSampling

Summary	Oversampling processing
Header	Config_FAA_common.inc
Declaration	—
Description	Performs oversampling using 256 A/D conversion results, stores the result in SHDMEM, and generates an FAA end interrupt.
Arguments	None
Return value	None
Remarks	None

6.5 Control Flow

The control flow of the oversampling A/D conversion is shown below.

6.5.1 Pointer Initialization Processing

Figure 6-1 shows the flowchart of the pointer initialization processing.
The pointer initialization processing is executed when the API function **"R_{ConfigName}_ADC_OverSampling_Start"** is called.

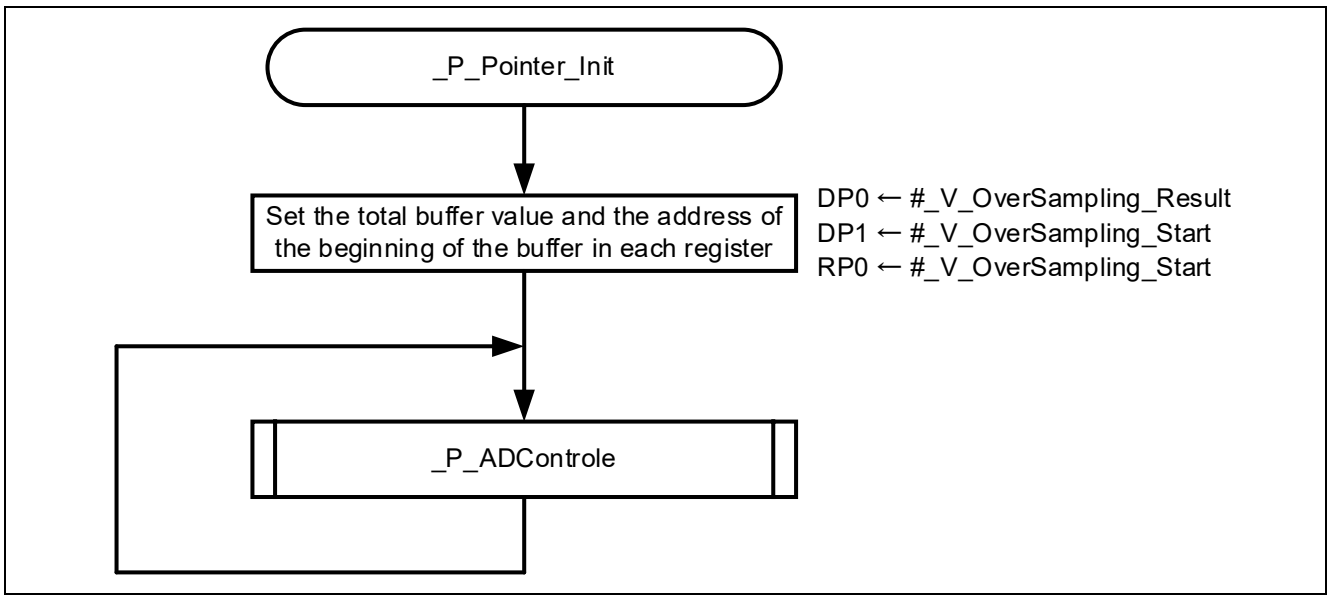


Figure 6-1 Pointer initialization processing

6.5.2 A/D Conversion Processing

Figure 6-2 shows the flowchart of the A/D conversion processing.

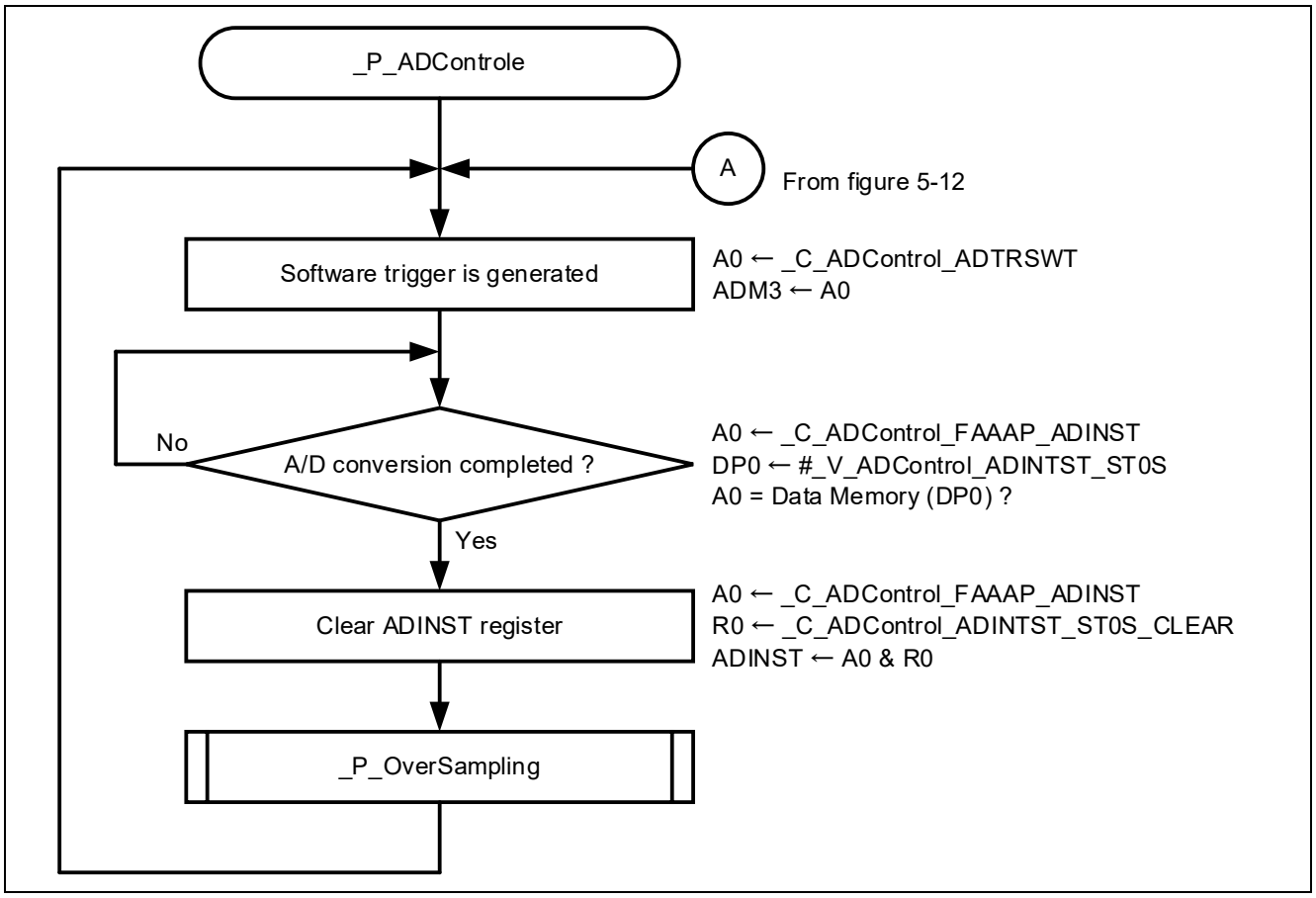


Figure 6-2 A/D Conversion Processing

6.5.3 Oversampling Processing

Figure 6-3 shows the flowchart of the oversampling processing.

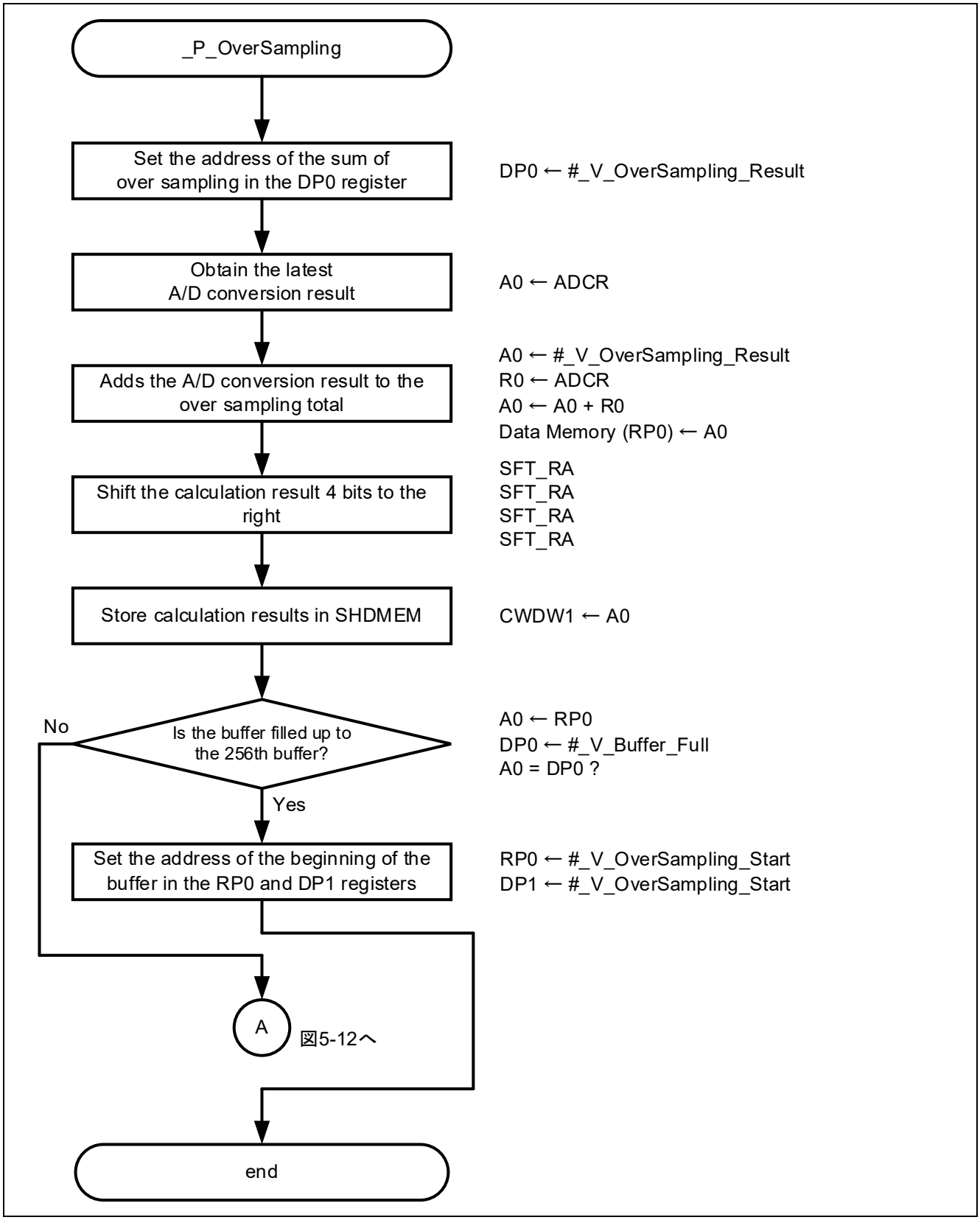


Figure 6-3 **Oversampling Processing**

7. API Information

7.1 7.1 List of Constants

Table 7-1 lists the constants provided by this library.

Table 7-1 Constants Used in the Library

Constant name	Setting value	Description
ENABLE_CALLBACK_FUNCTION	1	Enables callback function calls (0: disables callback function calls).

7.2 List of Variables

Table 7-2 lists the variables provided by this library.

Table 7-2 Variables Used in the Library

Valiable name	Type	Description	Used in
g_oversampling_result	uint16_t	Stores the oversampling result.	Config_FAA_ADC_OverSampling.c
g_faa_stop_flg	bool	Oversampling completion flag.	Config_FAA_ADC_OverSampling.c

7.3 API Typedef Definitions

This section describes the typedef definitions provided by this library.

7.3.1 e_ad_convert_time_t

This typedef defines the A/D conversion time. It is used to specify the A/D conversion time of the A/D converter.

```
typedef enum
{
    AD_CONVERT_TIME_1600 = 0x00,    // 1600/fCLK
    AD_CONVERT_TIME_800  = 0x08,    // 800/fCLK
    AD_CONVERT_TIME_400  = 0x10,    // 400/fCLK
    AD_CONVERT_TIME_200  = 0x18,    // 200/fCLK
    AD_CONVERT_TIME_100  = 0x20,    // 100/fCLK
    AD_CONVERT_TIME_50   = 0x28,    // 50/fCLK
} e_ad_convert_time_t;
```

7.3.2 e_oversampling_channel_t

This typedef defines the analog input channels. It is used to specify the channel to be oversampled by the A/D converter.

```
typedef enum
{
    OVERSAMPLING_CHANNEL_0,
    OVERSAMPLING_CHANNEL_1,
    OVERSAMPLING_CHANNEL_2,
. . .
    OVERSAMPLING_CHANNEL_6,
    OVERSAMPLING_CHANNEL_7,
    OVERSAMPLING_CHANNEL_16 = 16U,
    OVERSAMPLING_CHANNEL_17,
. . .
    OVERSAMPLING_CHANNEL_30
} e_oversampling_channel_t;
```

7.4 API Function Specification

This section describes the API functions provided by this library.
“{ConfigName}” included in an API function name indicates the configuration name of the FAA component set in Smart Configurator.

7.4.1 R_{ConfigName}_ADC_OverSampling_Create

Format

void R_{ConfigName}_ADC_OverSampling_Create (CallbackFunction callback)

Parameters

Callback	Pointer to the callback function. (Note) If you do not use a callback function, set this parameter to NULL.

Return Values

None

Properties

The prototype is declared in {ConfigName}_ADC_OverSampling_Create.h.

Description

This function initializes SHDMEM used by the FAA, sets the callback function to the function pointer, and configures the reference voltages of the A/D converter.

7.4.2 R_{ConfigName}_ADC_OverSampling_SetParameter

Format

void R_{ConfigName}_ADC_OverSampling_SetParameter (st_sampling_parameter_t *config)

Parameters

config.time A/D conversion time

config.channel Analog input channel to be oversampled

Return Values

None

Properties

The prototype is declared in **{ConfigName}_ADC_OverSampling.h**.

Description

This function sets the A/D conversion time and the channel to be oversampled.

Example

```
config.time    = AD_CONVERT_TIME_50;
config.channel = OVERSAMPLING_CHANNEL_2;
R_Config_FAA_ADC_OverSampling_SetParameter(&config);
```

Notes:

Set the conversion time within the range specified for the A/D converter. The A/D conversion times that can be configured in this library are as follows

System clock: 48 MHz:

1. For **"AD_CONVERT_TIME_1600"**
   Conversion time = $\frac{1600}{48\times10^6} \fallingdotseq$ 33.333μs
2. For **"AD_CONVERT_TIME_800"**
   Conversion time = $\frac{800}{48\times10^6} \fallingdotseq$ 16.667μs
3. For **"AD_CONVERT_TIME_400"**
   Conversion time = $\frac{400}{48\times10^6} \fallingdotseq$ 8.3333μs
4. For **"AD_CONVERT_TIME_200"**
   Conversion time = $\frac{200}{48\times10^6} \fallingdotseq$ 4.1667μs
5. For **"AD_CONVERT_TIME_100"**
   Conversion time = $\frac{100}{48\times10^6} \fallingdotseq$ 2.0833μs
6. For **"AD_CONVERT_TIME_50"**
   Conversion time = $\frac{50}{48\times10^6} \fallingdotseq$ 1.0417μs

For details, refer to "RL78/G24 User's Manual: Hardware (R01UH0961)", "Table 20-6 Selection of A/D Conversion Time (10/11)".

If **AVREFP** is used as the positive reference voltage of the A/D converter, do not select **ANI0** as the A/D conversion channel. Also, if **AVREFM** is used as the negative reference voltage of the A/D converter, do not select **ANI1** as the A/D conversion channel.

7.4.3 R_{ConfigName}_ADC_OverSampling_Start

Format

FAA_Status_t R_{ConfigName}_ADC_OverSampling_Start (void)

Parameters

None

Return Values

FAA_SUCCESS	FAA processing succeeded
FAA_ALREADY_RUNNING	FAA is already running

Properties

The prototype is declared in **{ConfigName}_ADC_OverSampling.h**.

Description

This function switches bus access to the A/D converter to the FAA and starts FAA processing.

7.4.4 R_{ConfigName}_ADC_OverSampling_Stop

Format

void R_{ConfigName}_ADC_OverSampling_Stop (void)

Parameters

None

Return Values

None

Properties

The prototype is declared in **{ConfigName}_ADC_OverSampling.h**.

Description

This function switches bus access to the A/D converter back to the CPU and stops FAA processing.

7.4.5 R_{ConfigName}_ADC_OverSampling_Get

Format

void R_{ConfigName}_ ADC_OverSampling_Get (void)

Parameters

None

Return Values

None

Properties

The prototype is declared in **{ConfigName}_ADC_OverSampling.h**.

Description

This function retrieves the oversampling result. The oversampling result is stored in **CWDW1L** in **SHDMEM**.

7.4.6 R_{ConfigName}_ADC_OverSampling_INTFAAE_interrupt

Format

static void __near R_Config_FAA_ADC_OverSampling_INTFAAE_interrupt(void)

Parameters

None

Return Values

None

Properties

The prototype is declared in **{ConfigName}_ADC_OverSampling.h**.

Description

This function is the oversampling processing completion interrupt handler, and it sets a flag within the function.

There are two ways to determine that the oversampling A/D conversion processing has completed:

1. **Poll the FAA completion flag**
   Poll the FAA completion flag (**g_faa_stop_flg**) that is set in the oversampling completion interrupt function from your user program. Determine completion by checking the flag state.
2. **Register a callback function**
   By enabling **ENABLE_CALLBACK_FUNCTION**, you can use a callback function. When the FAA completion interrupt occurs, the library calls the callback function. By adding user code to this callback function, you can execute the required processing when the operation completes.

7.5 Procedure for Calling the Library APIs

This section shows an example procedure for calling the API functions when using the oversampling A/D conversion library to perform oversampling on one analog input channel.
The library APIs are shown in **bold**.

Table 7-3 Example Procedure for Calling the API Functions (1/2)

No.	Operation	Description
Start processing		
(1)	EI();	Enables acceptance of interrupt requests by the CPU.
(2)	**R_Config_FAA_ADC_OverSampling_ Create(oversampling_callback);**	Initializes SHDMEM. If a callback function is used, sets the callback function.
(3)	st_sampling_parameter_t config = {AD_CONVERT_TIME_50, OVERSAMPLING_CHANNEL_2};	Declares a variable for setting the A/D conversion time and analog input channel.
(4)	**R_Config_FAA_ADC_OverSampling_SetParameter(&config);**	Sets the A/D conversion time and analog input channel.
(5)	R_Config_ADC_Set_OperationOn();	Sets the A/D converter to the A/D conversion standby state.
(6)	R_Config_ADC_Start();	Sets the A/D converter to the A/D trigger wait state.
(7)	**R_Config_FAA_ADC_OverSampling_Start();**	Switches bus access to the A/D converter to the FAA and starts FAA processing.
(8)	Wait for completion of oversampling A/D conversion by the FAA	**(a) When using a callback function:** After the FAA completes oversampling A/D conversion, an FAA end interrupt (INTFAAE) occurs and the callback function set in (2) is called. Until oversampling A/D conversion completes, the CPU can execute other tasks. **(b) When not using a callback function:** After oversampling A/D conversion completes, the g_faa_stop_flg flag is set. Detect that g_faa_stop_flg has been set in software, and after detection, clear the g_faa_stop_flg flag.
(9)	**R_Config_FAA_ADC_OverSampling_Get();**	Retrieves the oversampling A/D conversion result from SHDMEM and stores it in g_oversampling_result. To perform oversampling A/D conversion again, return to (7).

Table 7-4 Example Procedure for Calling the API Functions (2/2)

No.	Operation	Description
Stop processing		
(10)	**R_Config_FAA_ADC_OverSampling_Stop();**	Switches bus access to the A/D converter back to the CPU and stops FAA processing.
(11)	R_Config_ADC_Stop();	Sets the A/D converter to the A/D conversion standby state.
(12)	R_Config_ADC_Set_OperationOff();	Sets the A/D converter to the A/D conversion stopped state.

8. Sample Application

From this chapter onward, sample code using the oversampling A/D conversion library is described.

8.1 Specification Overview

In this sample application, two processors (CPU and FAA) are used. Oversampling is controlled by the FAA to achieve a resolution of 12 bits or higher. The A/D conversion time and the analog input channel can be configured/switched by the program.

In the CPU program, the A/D converter is initially configured. After setting the A/D conversion time and the analog input channel, FAA operation is started. When the FAA end interrupt occurs, the oversampling result stored in SHDMEM is retrieved. While waiting for FAA completion, a port is toggled.

In the FAA program, the A/D converter is controlled. After operation starts, A/D conversion is performed with an A/D conversion time of **50/fCLK** on the analog input channel **P22 / ANI2**. Oversampling processing is performed on 256 A/D conversion results, and the result is stored in SHDMEM. When oversampling completes, an FAA end interrupt is generated and the FAA stops.

On the CPU side, the settings are then switched to an A/D conversion time of **1600/fCLK** and the analog input channel **P120 / ANI19**, and oversampling processing is executed again.

Table 8-1 shows the peripheral functions used and their purposes.

Table 8-1 Peripherals Used and Their Purpose

Peripheral function	Purpose
Flexible Application Accelerator (FAA)	Controls the A/D converter and performs oversampling processing.
A/D Converter (Advanced Mode ON)	Converts the analog input voltage on the analog input pin to a digital value.
Ports (PORT)	Continuously toggles a port to indicate simultaneous operation of the CPU and FAA.

Figure 8-1 shows the Setting of the Address Bus Selection Register (ADBSEL).

	15	14	13	12	11	10	9	8
ADBSEL	FAADIVSEL	0	TRNGSEL	PORTSEL	TKB32SEL	TKB31SEL	TKB30SEL	TRGSEL
Setting value	0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
TRD0SEL	PWMOP SEL	TRXSEL	DACSEL	PGACMP SEL	ADCSEL	SAU0SEL	TAU0SEL
0	0	0	0	0	1	0	0

0	Bus access by the CPU is enabled.
1	Bus access by the FAA is enabled.

ADCSEL : A/D Converter

Figure 8-1 Setting of the Address Bus Selection Register (ADBSEL)

8.2 Operation Description

In this sample code, 256 A/D conversion results are collected and oversampled to achieve a resolution of 12 bits or higher.
A flag is set by the FAA end interrupt, and the CPU retrieves the oversampling result stored in SHDMEM.

1. [CPU program] Initializes SHDMEM.
2. [CPU program] Sets the A/D conversion time and the channel to be oversampled.
3. [CPU program] Sets bus access to the A/D converter to the FAA.
4. [CPU program] Sets the FAA stack pointer and the start address of the FAA program, and starts FAA operation.
5. [FAA program] Performs A/D conversion on P22 / ANI2, oversamples the 256 conversion results, and stores the result in SHDMEM. It then stops clock supply to the FAA and ends FAA operation.
6. [CPU program] The FAA end interrupt occurs; after setting the flag, the CPU retrieves the value stored in SHDMEM.
7. [CPU program] Switches the A/D conversion time and the channel to be oversampled.
8. [CPU program] Repeats steps 3 to 6.

9. Operating Conditions

The sample application has been evaluated to operate in the following environment.

Table 9.1 Operating Conditions for Evaluation

Item	Details
MCU Used	RL78/G24（R7F101GLG）
Board Used	RL78/G24 Fast Prototyping Board (RTK7RLG240C00000BJ)
Operating Frequency	High-speed On-Chip Oscillator Clock (fHOCO): 8MHz PLL Clock (fPLL): 96MHz CPU/Peripheral Hardware Clock (fCLK): 48MHz
Operating Voltage	3.3V (Can operate between 2.7V to 5.5V) LVD0 Operation (V_{LVD0}): Reset Mode Rising edge = TYP. 2.97V Falling edge = TYP. 2.91V
Integrated Development Environment (CS+)	Renesas Electronics CS+ for CC V8.12.0
C Compiler (CS+.)	Renesas Electronics CC-RL V1.12.00
Integrated Development Environment（e² studio）	Renesas Electronics e² studio 2024-10 (24.10.0)
C Compiler（e² studio）	Renesas Electronics CC-RL V1.14.00
Smart Configurator	V.1.11.0
Board Support Package（r_bsp）	V.1.70
Emulator	CS+ and e² studio：COM PORT

10. Hardware Description

10.1 Example of Hardware Configuration

Figure 10-1 shows an example of the hardware configuration used in the sample code of this sample application.

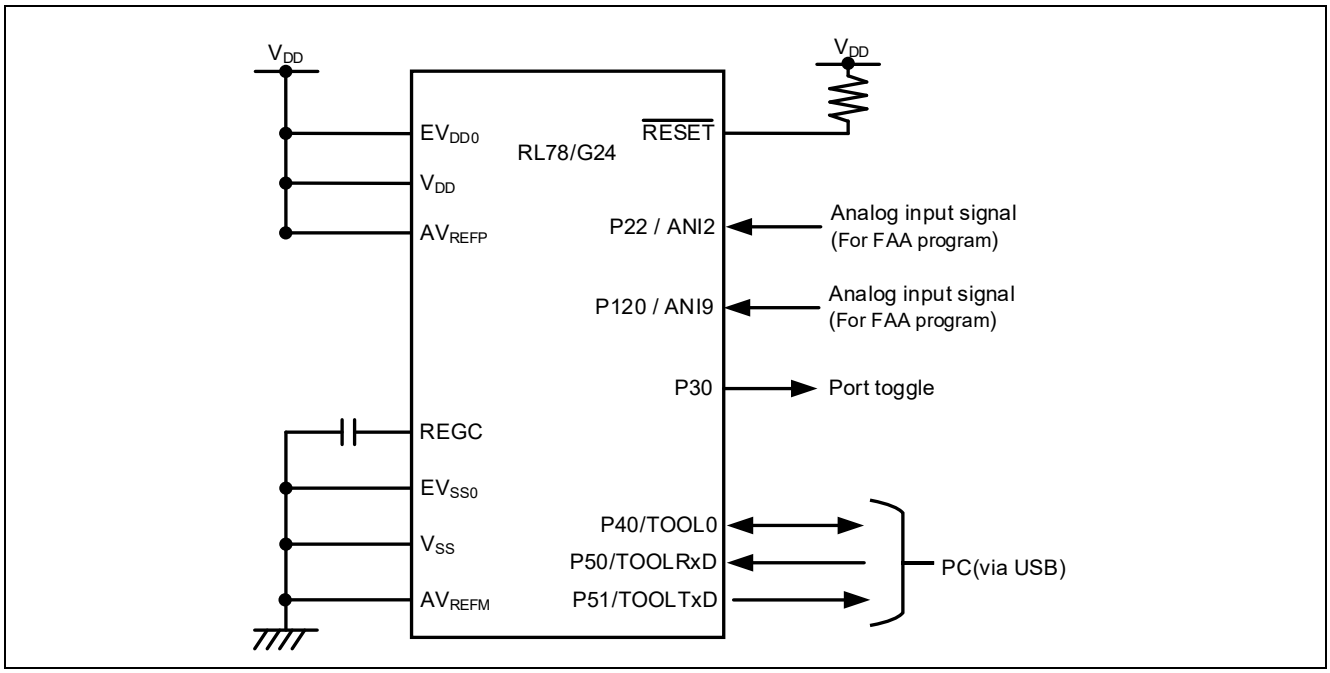


Figure 10-1 Example of Hardware Configuration

Note 1. This simplified circuit diagram was created to show an overview of connections only. When actually designing your circuit, make sure the design includes appropriate pin handling and meets electrical characteristic requirements (connect each input-only port to VDD or VSS through a resistor).

Note 2. Connect any pins whose name begins with EVSS to VSS, and any pins whose name begins with EVDD to VDD, respectively.

Note 3. VDD must not be lower than the reset release voltage (VLVD0) that is specified for the LVD0.

10.2 List of Pins Used

Table 10-1 shows the pins used and their functions.

Table 10-1 List of Pins Used

Pin name	I/O	Function
P22 / ANI2	Input	A/D converter analog input (for FAA program)
P120 / ANI19	Input	A/D converter analog input (for FAA program)
P30	Output	Port toggle

Note This sample application sets unused pins to “output” for noise countermeasures. For input-only pins, you need to configure the corresponding bits or connect them individually to VDD or VSS through a resistor. For details, refer to “2.3 Connection of Unused Pins” in the RL78/G24 User’s Manual: Hardware (R01UH0961). When actually designing your circuit, make sure the design includes appropriate pin handling and meets electrical characteristic requirements.

11. Software Description

11.1 Smart Configurator Setting

The Smart Configurator (SC) settings in this sample code are shown below. The items and settings in each SC settings table are explained using the description on the settings screen.

11.1.1 System Setting

Figure 11-1 shows the system settings used in this sample code (e2 studio, CS+). When performing COM port debugging on the RL78/G24 Fast Prototyping Board (RTK7RLG240C00000BJ), you need to configure the settings appropriately within the integrated development environment (e2 studio, CS+). For details, refer to “7.1 Using COM Port Debugging with the e2 studio” and “7.2 Using COM Port Debugging with CS+” in the RL78/G24 Fast Prototyping Board User’s Manual (R20UT5091).

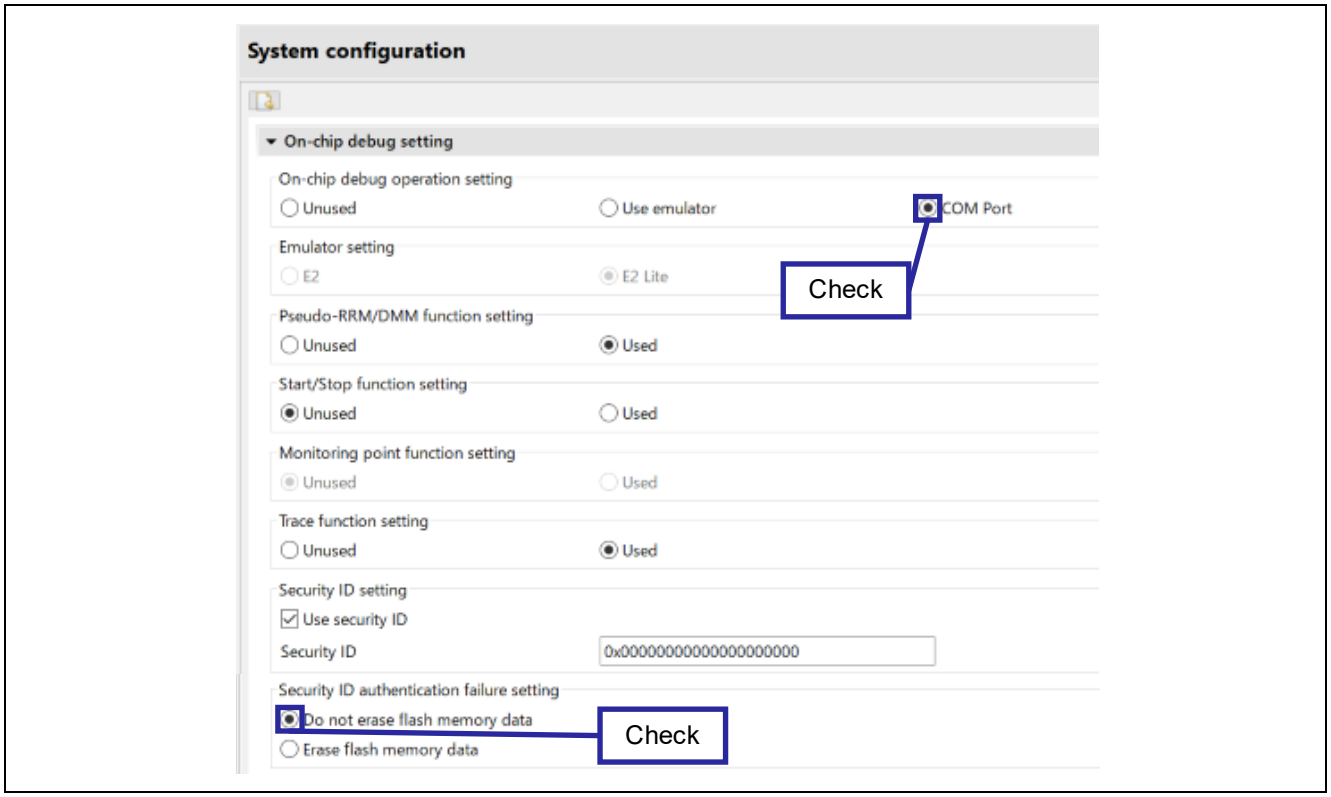


Figure 11-1 System Configuration（e² studio, CS+）

11.1.2 Component Settings

The settings of the components used in this sample code are shown below.

Table 11-1 Component Settings (FAA)

Item	Details
Component	Flexible Application Accelerator
Configuration Name	Config_FAA
Recouce	FAA

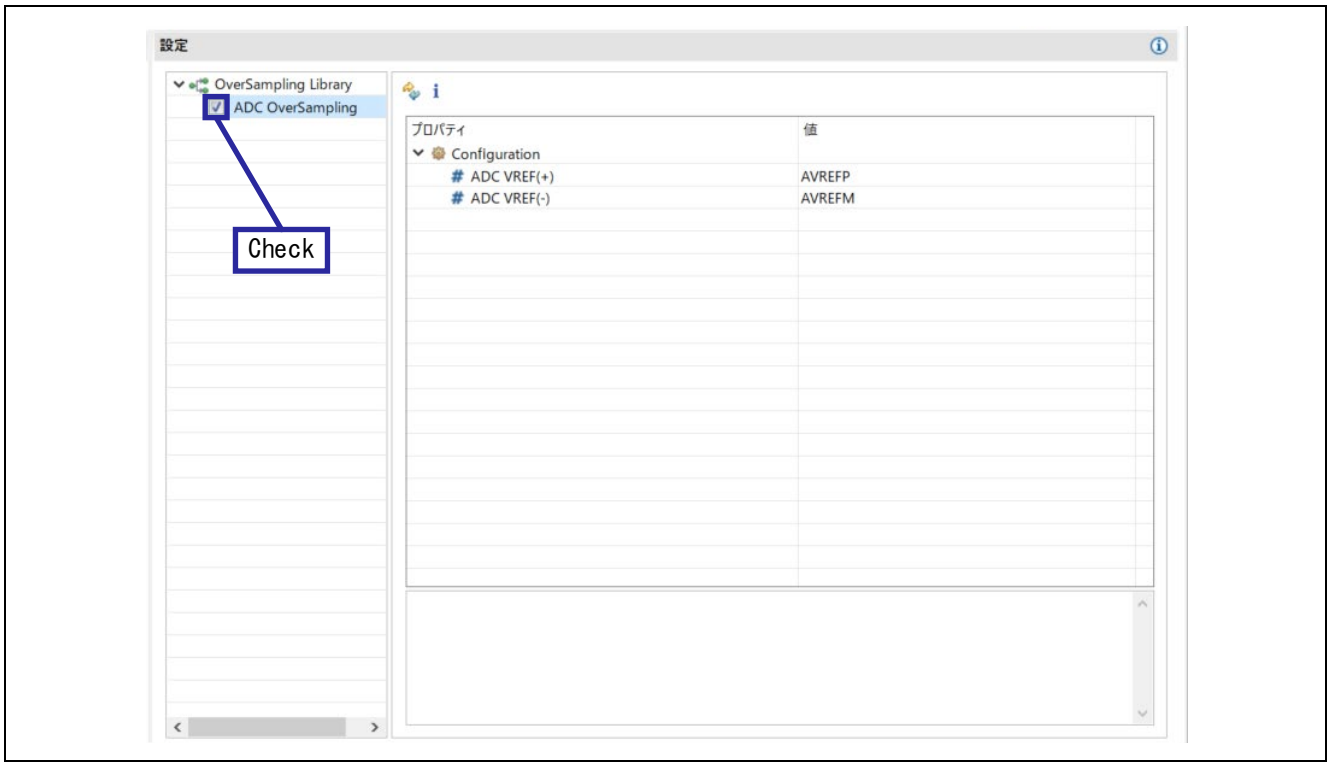


Figure 11-2 FAA Settings

Note If the FAA library is not displayed after loading the sample code, refer to No.11 in “2.3.1 Adding the FAA Component” in the “Flexible Application Accelerator (FAA) Tool Guide: CS+/e2studio” and download the FAA library.

Table 11-2 Component Settings (A/D Converter)

Item	Details
Component	A/D Converter
Configuration Name	Config_ADC
Recouce	ADC
Operation Mode	Advanced Mode

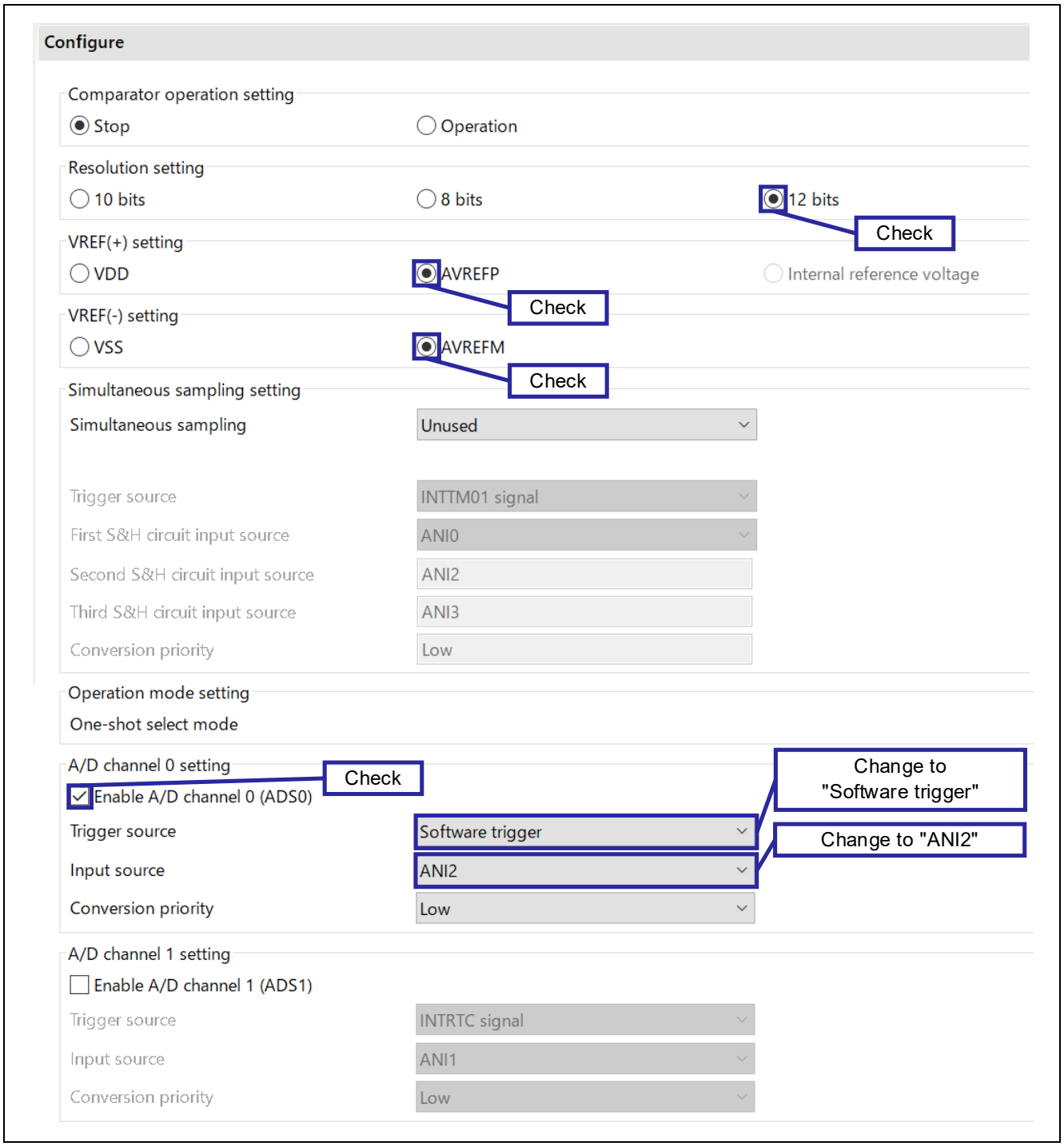


Figure 11-3 A/D Converter Settings (1/2)

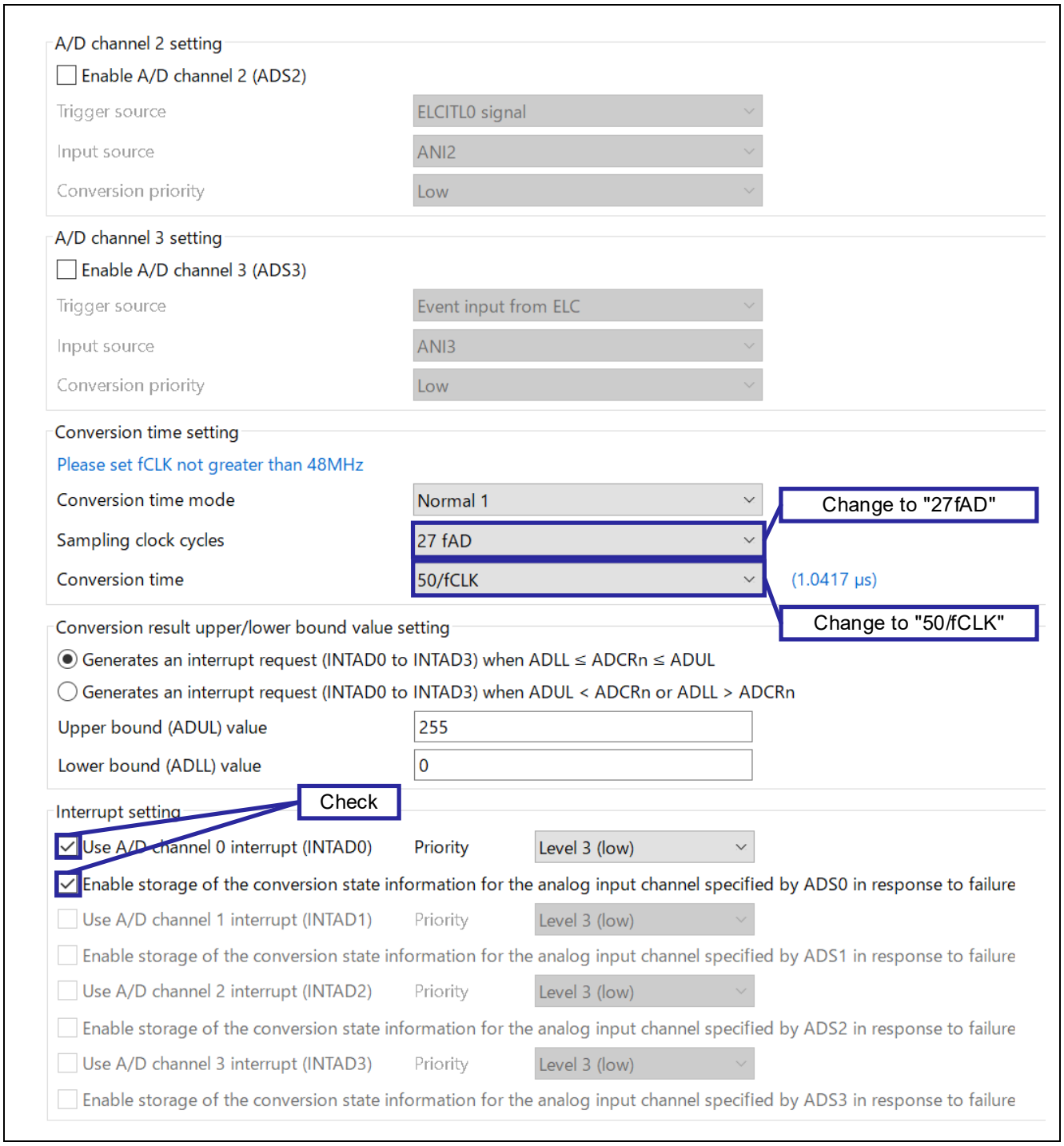


Figure 11-4 A/D Converter Settings (2/2)

Table 11-3 Component Settings (Port)

Item	Details
Component	PORT
Configuration Name	Config_PORT
Recouce	PORT

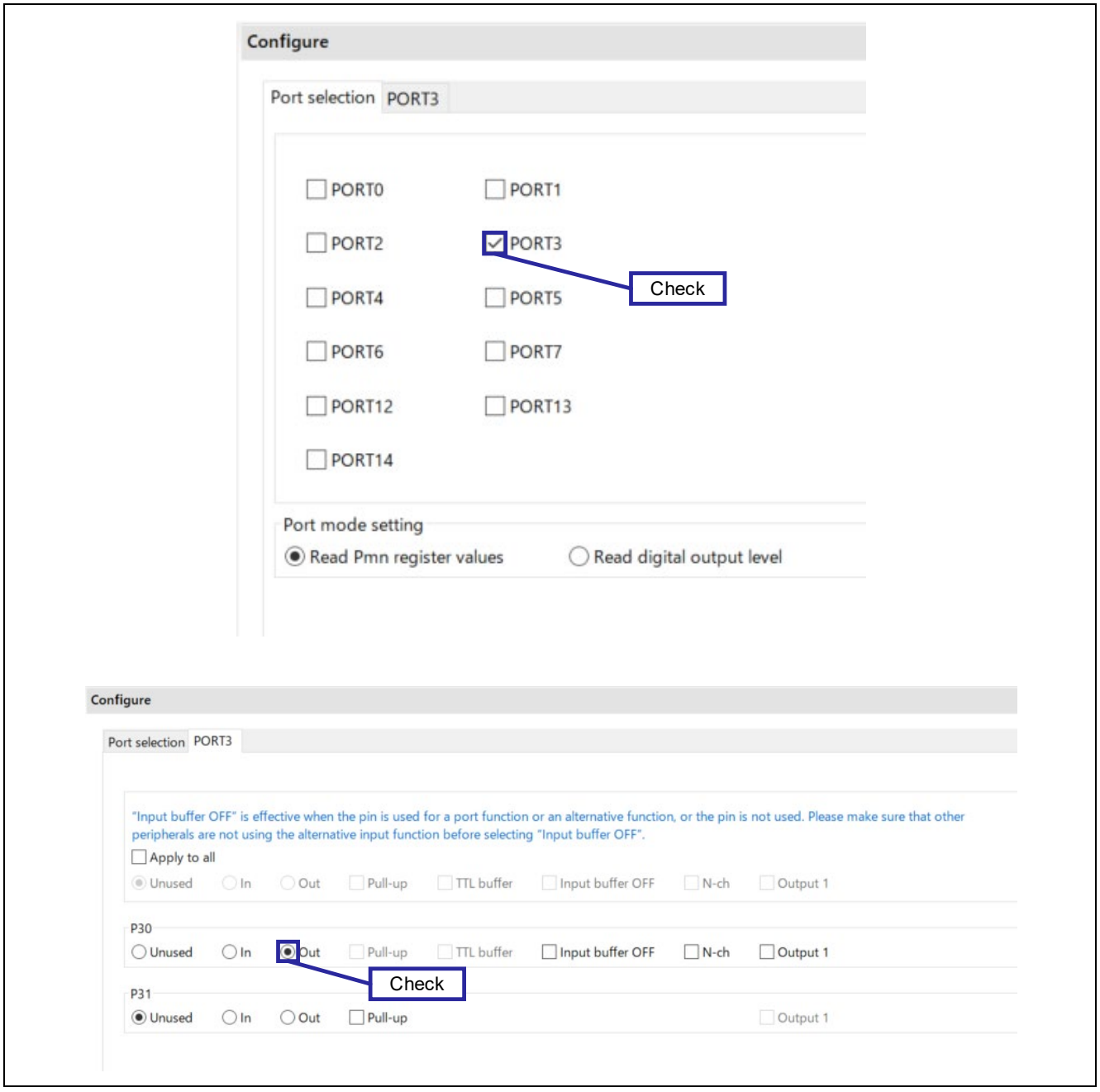


Figure 11-5 Port Settings

Note: In the sample application, unused pins are fixed as outputs for the purpose of reducing the impact of noise on A/D conversion. For input-only pins, appropriate unused-pin handling must be performed, such as enabling the internal pull-up resistor or connecting the pin individually to VDD or VSS through an external resistor. For details, refer to “2.3 Connection of Unused Pins” in RL78/G24 User’s Manual: Hardware (R01UH0961). In this sample program, unused-pin handling for an input-only pin is required only for P137.

11.2 Folder Structure

Table 11-4 shows the structure of the source files and header files used in the sample code.

Files automatically generated by the integrated development environment, as well as files related to the BSP environment, are excluded.

Figure 11-4 Folder Structure

Folder / File Name	Description	Smart Configurator Used
\r01an7576_faa_adc_oversampling<DIR>	Sample code folder	
\src<DIR>	Folder for program source files	
main.c	Sample code source file	
main.h	Sample code header file	
faa_oversampling.c	FAA source file	
faa_oversampling.h	FAA header file	
\smc_gen<DIR>	Smart Configurator generated folder	√
\Config_ADC<DIR>	Folder for ADC-related programs	√
Config_ADC.c	ADC source file	√
Config_ADC.h	ADC header file	√
Config_ADC_user.c	ADC interrupt source file	√Note 1
\Config_FAA<DIR>	Folder for FAA-related programs	√
Config_FAA_ADC_OverSampling.c	Over-sampling function source file	√Note 2
Config_FAA_ADC_OverSampling.h	Over-sampling function header file	√
Config_FAA_common.c	Common FAA module source file	√
Config_FAA_common.h	Common FAA module header file	√
Config_FAA_common.inc	FAA include file	√
Config_FAA_src.dsp	FAA assembler source file	√
\Config_PORT<DIR>	Folder for PORT-related programs	√
Config_PORT.c	PORT source file	√
Config_PORT.h	PORT header file	√
Config_PORT_user.c	PORT interrupt source file	√Note 1
¥general<DIR>	Folder for initialization and common programs	
¥r_bsp<DIR>	Folder for BSP programs	
¥r_config<DIR>	Folder for configuration programs	

Remark: ”<DIR>” indicates a directory.

Note 1: This file is not used in this sample code.

Note 2: Code has been added to the user code area of the Smart Configurator.

11.3 Option Byte Settings

Table 11-5 shows the option byte settings.

Table 11-5 Option Byte Settings

Address	Setting Value	Description
000C0H/040C0H	1110 1111B (EFH)	Watchdog timer operation disabled (counting stops after reset release)
000C1H/040C1H	1111 1011B (FBH)	LVD0 reset mode Detection voltage: rising 2.97 V / falling 2.91 V
000C2H/040C2H	1110 1010B (EAH)	Flash operation mode: High-speed main mode High-speed on-chip oscillator frequency: 8 MHz
000C3H/040C3H	1000 0101B (85H)	On-chip debug operation enabled

11.4 Constant List

No constants are used in this sample code.

11.5 Variable List

No variables are used in this sample code.

11.6 Function List

Table 11-6 lists the functions used in the sample code.
Functions generated by the Smart Configurator that have not been modified are excluded.

Table 11-6 Function List

Function Name	Description	Source File
main	Main processing	main.c
oversampling_callback	Callback processing	main.c

11.7 Function Specifications

This section describes the function specifications of the sample code.

[Function Name] main

Description	Main Processing
Header	r_smc_entry.h、main.h、Config_FAA_ADC_OverSampling.h
Format	void main (void);
Overview	This function performs the initial configuration of the A/D converter. After setting the A/D conversion time and the oversampling target channel, it starts FAA operation. Completion of oversampling is detected by polling, and the conversion result is retrieved from SHDMEM. The function then repeats this sequence: switching the A/D conversion time and the oversampling target channel, and restarting FAA operation.
Arguments	None
Return Value	None
Remarks	None

[Function Name] oversampling_callback

Description	Callback processing
Format	Config_FAA_common.h、main.h、Config_FAA_ADC_OverSampling.h
Declaration	void oversampling_callback(void)
Overview	User code should be written in this function. In the sample code, a NOP instruction is executed.
Arguments	None
Return Value	None
Remarks	None

11.8 Flowchart

11.8.1 Main Processing

Figure 11-6 shows the flowchart of the main processing.

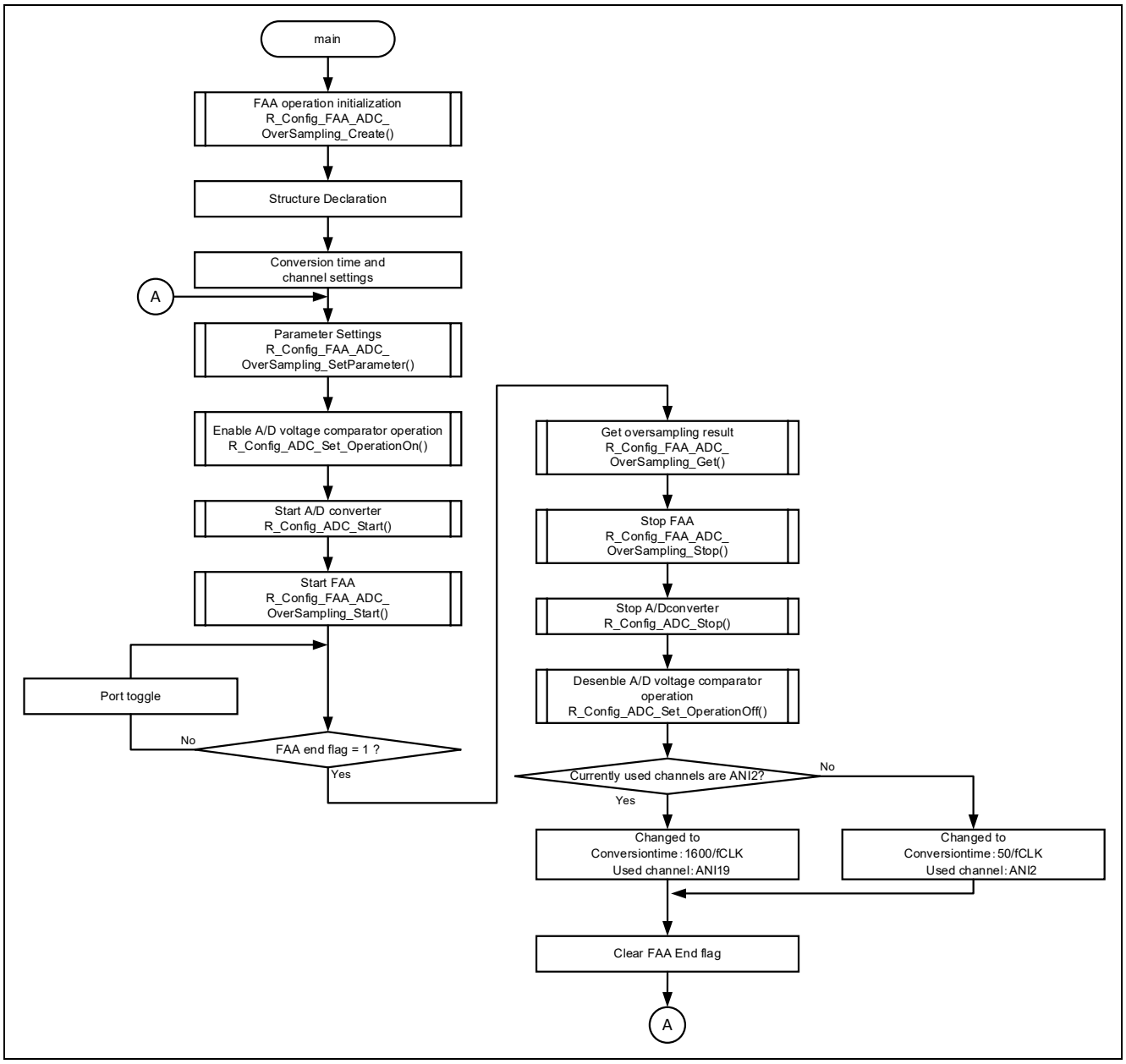


Figure 11-6 Main Processing

11.8.2 oversampling_callback Function

Figure 11-7 shows the flowchart of the oversampling_callback function.

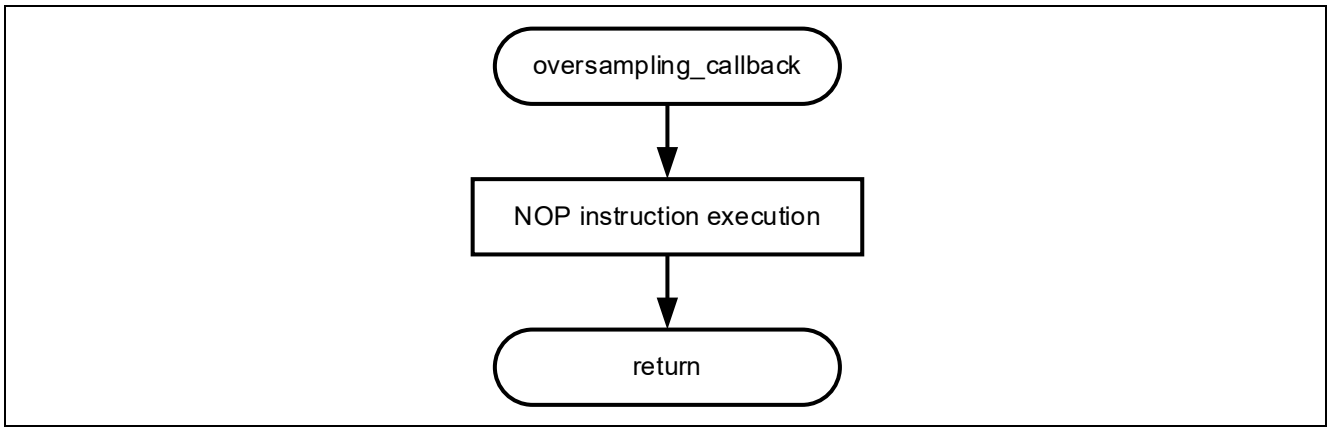


Figure 11-7 oversampling_callback Function

12. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

13. Reference Documents

RL78/G24 User's Manual: Hardware (R01UH0961)

RL78 family User's Manual: Software (R01US0015)

RL78/G24 Fast Prototyping Board User's Manual (R20UT5091)

RL78 Smart Configurator User's Gude: CS+ (R20AN0580)

RL78 Smart Configurator User's Gude: e2 studio (R20AN0579)

Flexible Application Accelerator (FAA) Tool Guide: CS＋ (R01AN7094)

Flexible Application Accelerator (FAA) Tool Guide: e2 studio (R01AN7095)

(The latest version can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News

(The latest version can be downloaded from the Renesas Electronics website.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jan. 27th, 2026	—	First Edition

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.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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/.