RX Family

Sample Program Using FIR Filter to Perform Frequency Band Judgement

Introduction

This application note describes an example of the use of the FIR filter API from the RX Family DSP library.

The sample program described in this application note is configured as shown in Figure 1. The analog signal input to the RX140 undergoes A/D conversion and normalization, after which it is divided into three channels for processing by the FIR filter. The results of FIR filter processing are then used to make a judgement of the frequency band of the input signal. The judgement result is indicated by LEDs.

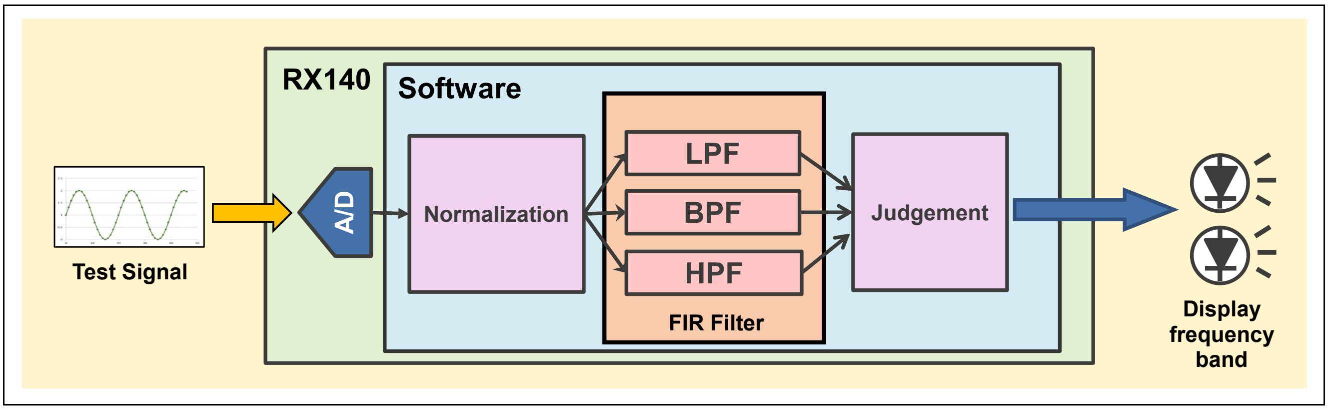


Figure 1 System Outline Diagram

The sample program implements FIR filter processing using 16-bit fixed-point data, 32-bit fixed-point data, or single-precision floating-point data. The description below applies to the default setting, which performs processing using 16-bit fixed-point data. To use the sample program to process 32-bit fixed-point or single-precision floating-point data it is necessary to change the configuration settings, and some restrictions apply.

The sample program environment and usage procedure, and details of the sample program, are described in the pages that follow.

Target Device

RX140 Group

Operation Confirmation Board

Target Board for RX140

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. System Overview

Figure 1.1 shows an overview of the system described in this application note.
This system uses a single RX140 MCU for all processing, from sampling of the input signal to judgement result output control.

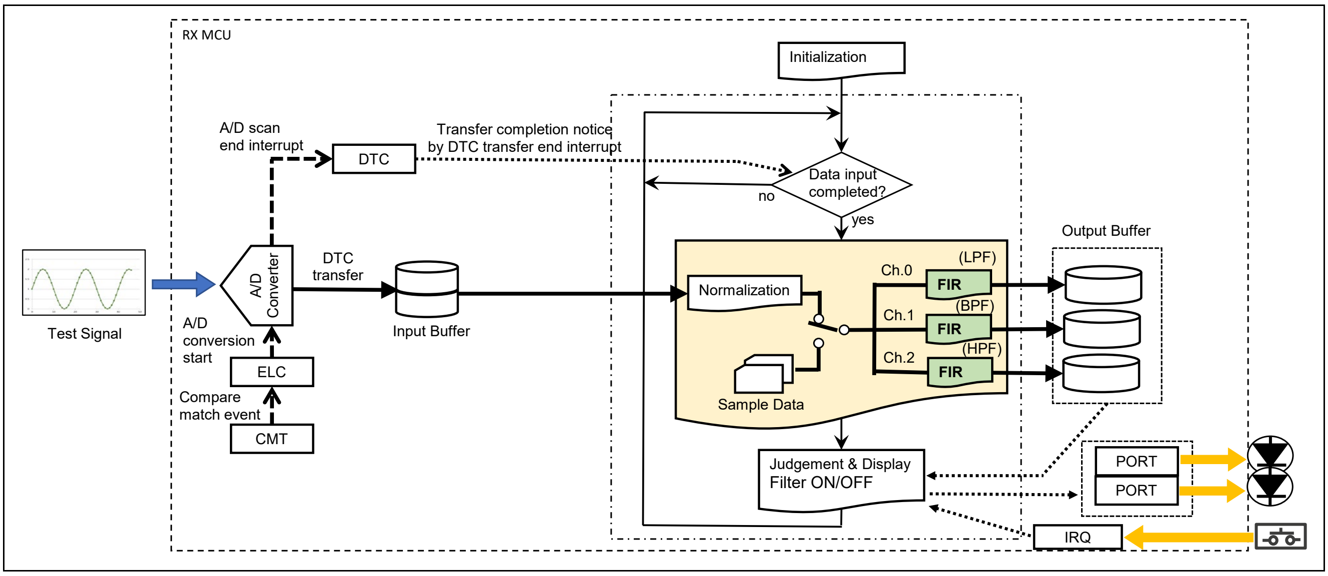


Figure 1.1 System Overview

The system performs the following processing:

1. A/D conversion
   The 12-bit A/D converter (S12AD), compare match timer (CMT), and event link controller (ELC) are used to perform A/D conversion at a sampling frequency of approximately 10 kHz. First, the CMT generates compare match events with a period of approximately 100 μs, and these events are passed by the ELC to the S12AD as A/D conversion start triggers. The converted data is transferred to the input buffer by the DTC controller (DTC). After 256 DTC transfers have completed, a DTC transfer end interrupt is generated.

2. Normalization
   The input signal A/D converted by the S12AD is stored in the input buffer as 12-bit (unsigned) data. The 12-bit data stored in the input buffer is normalized to 15-bit (signed) format (by bias processing and scaling).

3. FIR filter
   The normalized data undergoes low-pass filter (LPF), band-pass filter (BPF), and high-pass filter (HPF) FIR filter processing, and the results are stored in the output buffer. In Figure 1.1, Ch.0 is the LPF, Ch.1 is the BPF, and Ch.2 is the HPF. The filter can be turned on or off by means of switch input.

4. Judgement
   A judgment is made as to which band the frequency components of the input signal correspond to, and the result is indicated by changing the illumination pattern of the LEDs.

1.1 File Structure Associated with This Application Note

Figure 1.2 shows the file structure associated with this application note. When the contents of the ZIP file in which this application note is distributed are extracted, a folder is created with the same name as the ZIP file. The "workspace_fir_example" folder within this folder contains an e² studio workspace that includes projects in e² studio format. As shown in Figure 1.3, the project folder contains the sample program source code files as well as e² studio configuration files and this application note.

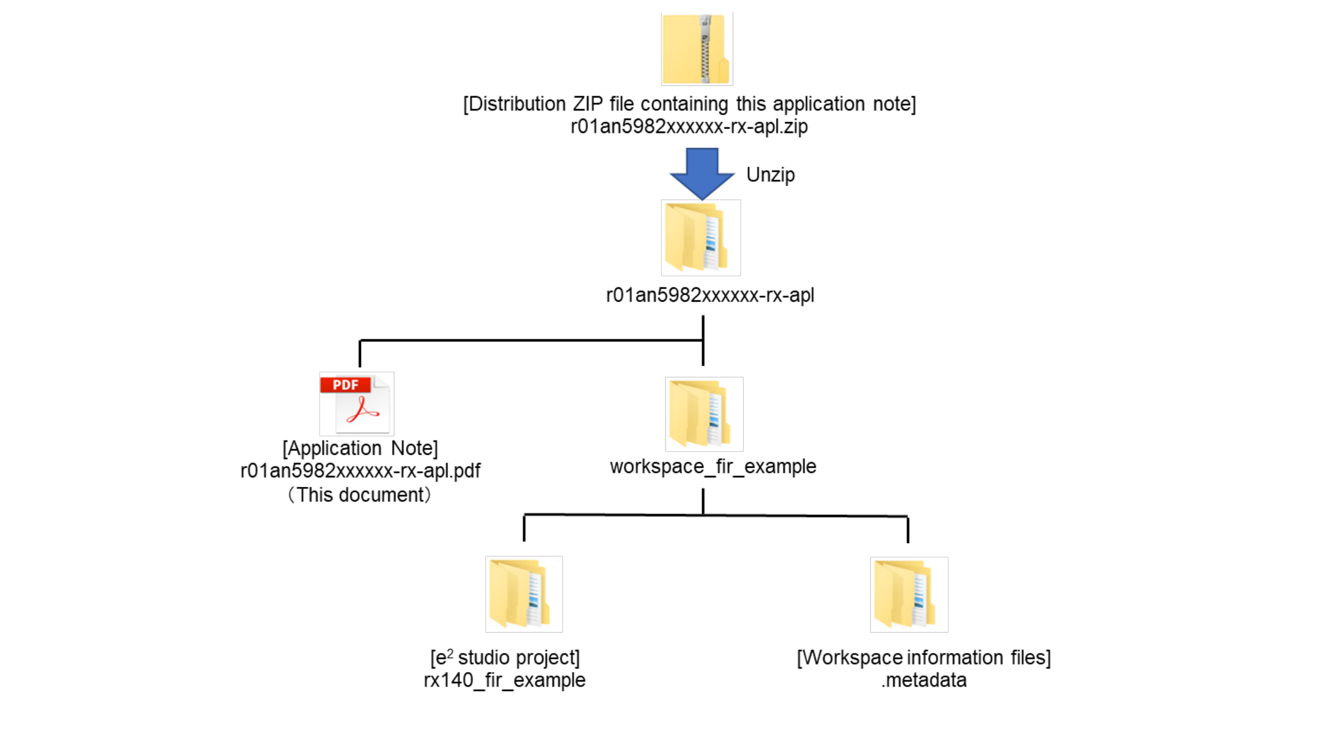


Figure 1.2 File Structure Associated with This Application Note

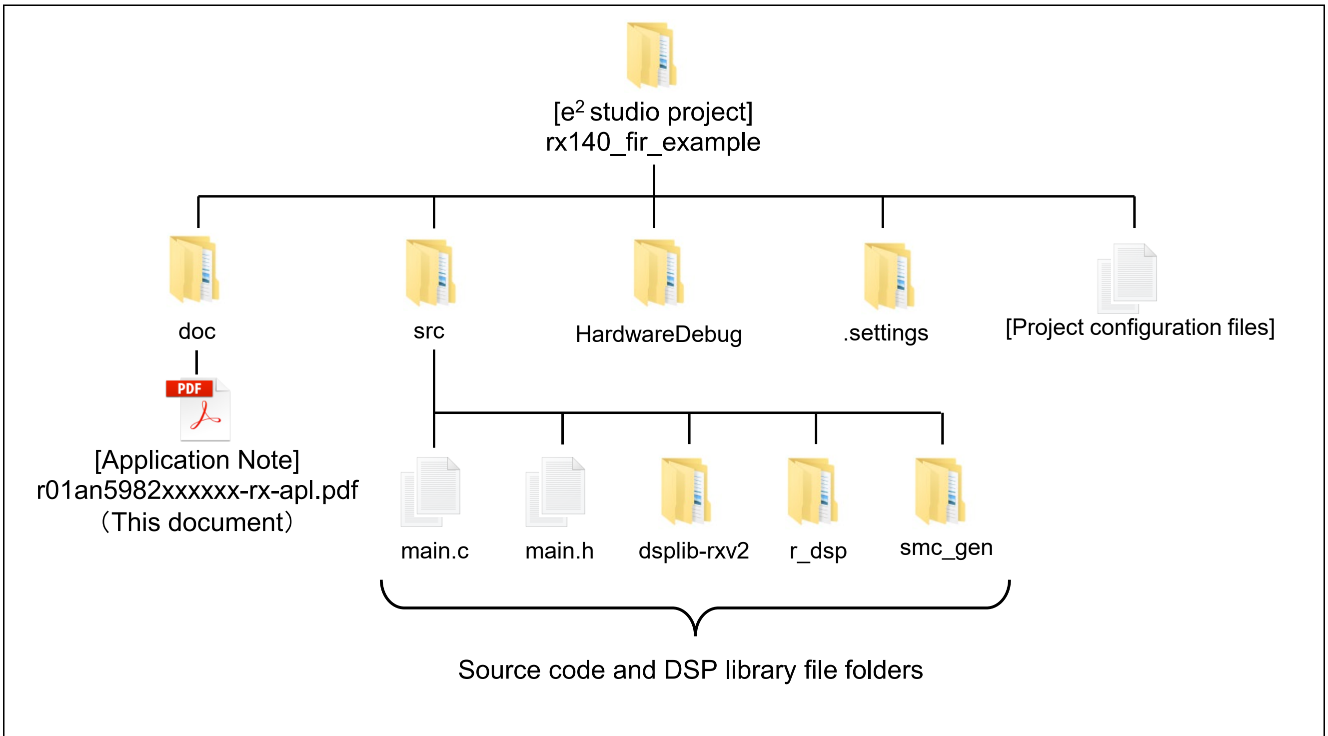


Figure 1.3 Folder Structure of Sample Project

1.2 Structure of Sample Program

Figure 1.4 shows the structure of the sample program and Table 1.1 lists the software modules used. The FIT modules and DSP library can be obtained from the Renesas website. Driver software for the other peripheral functions is generated by using the Code Generator function of e² studio. For details of each software module, refer to the associated application note or the e² studio help system.

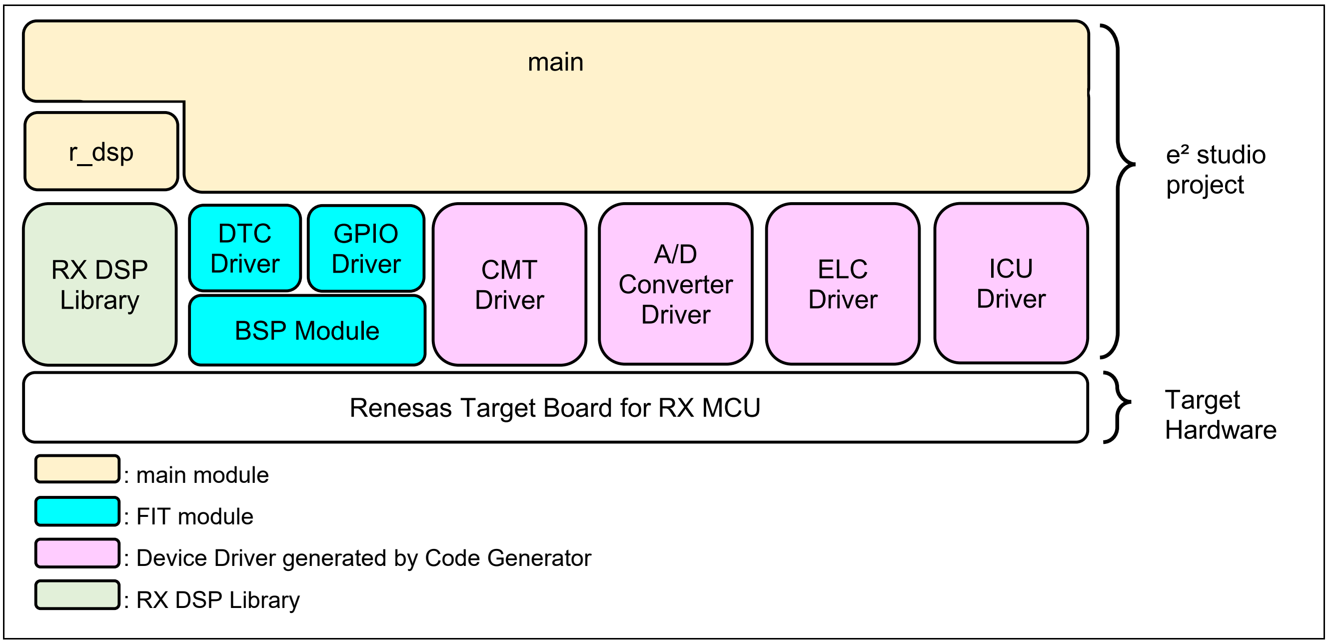


Figure 1.4 Structure of Sample Program

Table 1.1 List of Software Modules Used

Module	Document Title	Document No.	Category
main	—	—	Module containing the main function developed for the program described in this application note
r_dsp	—	—	DSP library operation module developed for the program described in this application note
BSP	RX Family Board Support Package Module Using Firmware Integration Technology	R01AN1685	FIT module
DTC	RX Family DTC Module Using Firmware Integration Technology	R01AN1819	FIT module
GPIO	RX Family GPIO Module Using Firmware Integration Technology	R01AN1721	FIT module
S12AD	—	—	Driver function generated by Code Generator
CMT	—	—	
ELC	—	—	
ICU	—	—	
RX DSP library	RX Family DSP Library Version 5.0	R01AN4359	DSP library

1.3 Operating Environment

The operation of the sample program described in this application note has been confirmed under the conditions listed in Table 1.2.

Table 1.2 Operation Confirmation Conditions

Item		Description
MCU		R5F51403ADFM (RX140 Group)
	Operating frequency	• HOCO clock: 48 MHz • System clock (ICLK): 48 MHz (HOCO clock × 1) • FlashIF clock (FCLK): 48 MHz (HOCO clock × 1) • Peripheral module clock (PCLKB): 24 MHz (HOCO clock × 1/2) • Peripheral module clock (PCLKD): 48 MHz (HOCO clock × 1)
	Operating voltage	3.3 V
	Operating mode	Single-chip mode
	Processor mode	Supervisor mode
Integrated development environment		Renesas Electronics e² studio 2022-01
C compiler		Renesas Electronics RX Compiler CC-RX V3.04.00
		Compiler options • -lang = c99 • -save_acc
Endian order		• Data: Little endian • Debug tool setting: Little endian
iodefine.h		Version 1.00A
Sample program		Version 1.01
Evaluation board		Renesas Electronics: Renesas Target Board for RX140 (RTK5RX1400C00000BJ) • On-board MCU: See above. • Power supply: Supplied via USB.
Function generator		Signal generator with analog signal output terminal to output sine waveforms. Output signal set to 1.65 V bias relative to ground (GND) and amplitude of 3.0 Vpp. • Analog signal output (+) is connected to the CN3.60 pin on the target board. The signal applied to the CN3.60 pin is input to AN000 of the S12AD. • Analog signal output (GND) is connected to the CN3.40 pin on the target board. The CN3.40 pin is connected to GND on the target board.

2. Running the Sample Program

The procedure for running the program described in this application note is shown below.

2.1 Launching the Workspace

Extract the ZIP file containing the project described in this application note to a location of your choice, and make sure the path of the destination does not contain any Japanese or other double-byte characters. Next, launch e² studio and, when Eclipse Launcher window appears, select the workspace (workspace_fir_example) described in this application note.

If Eclipse Launcher window does not appear when e² studio is launched, make the following selection after launching e² studio:

[File] >> [Switch Workspace] >> [Other]

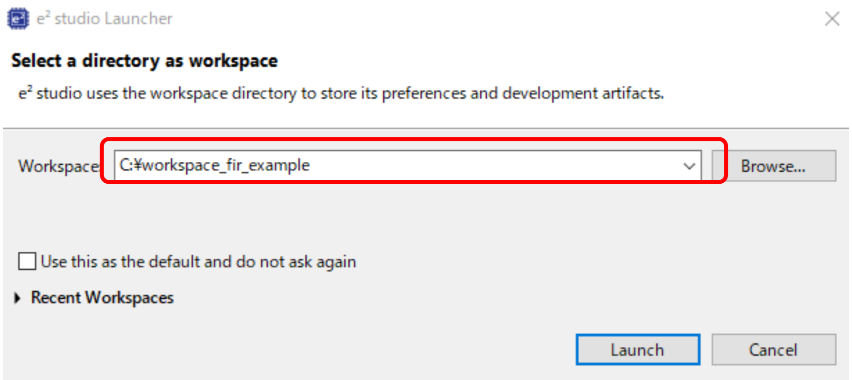
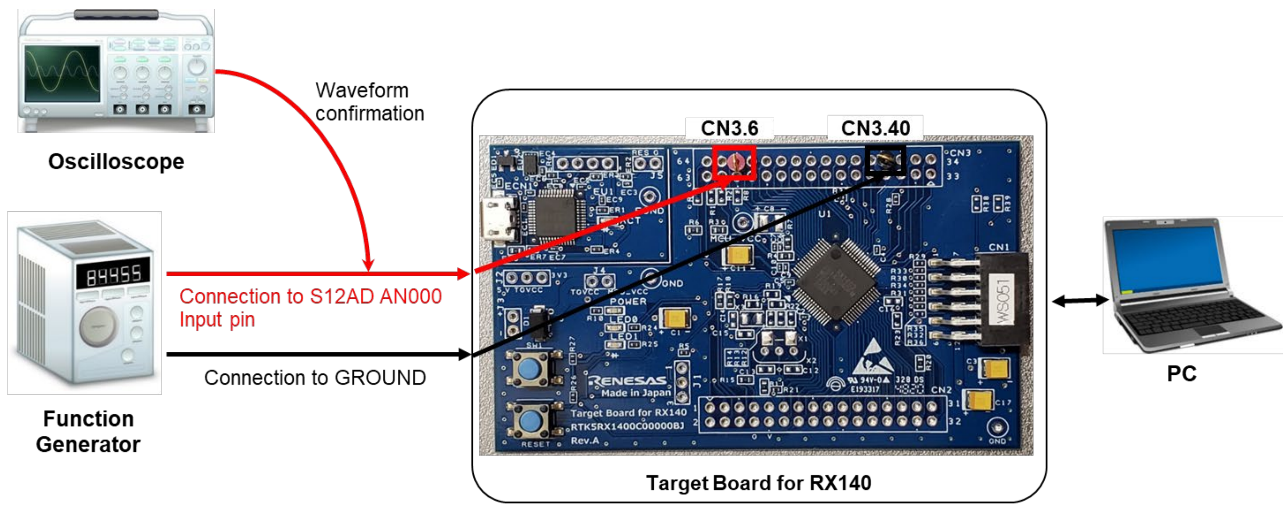


Figure 2.1 Eclipse Launcher

2.2 Connecting Equipment

Make connections as shown in Figure 2.2.



Note: The signal output terminals of some function generators may require terminating resistors. Refer to the manual of the function generator for confirmation.

Figure 2.2 Connecting Equipment

2.3 Running the Sample Program and Checking Operation

Using e2 studio, the procedure for connecting the debug tool and running the sample program is as follows.

1. Click Build button to build the sample program in e^2 studio
2. Click Debug button to download it to the MCU.
3. Click Reset button to reset the MCU.
4. Click Resume button to MCU and then run the program.
5. A break occurs at the start of the main function, so click to continue running the program.
   After resuming execution, the program starts.

2.3.1 Frequency Band Indication on LEDs

The sample program makes a judgement as to the frequency band of the input signal based on the three FIR filter outputs. The judgement result is indicated by the illumination pattern of LED0 and LED1 on the target board, as seen in Figure 2.3. The illumination pattern is updated approximately once per second. Refer to 3.4.5, Frequency Band Judgement Based on FIR Filter Processing Results, for a list of the LED illumination patterns.

In addition, you can toggle the FIR filter on and off by pressing a switch on the target board (SW1) while the program is running.

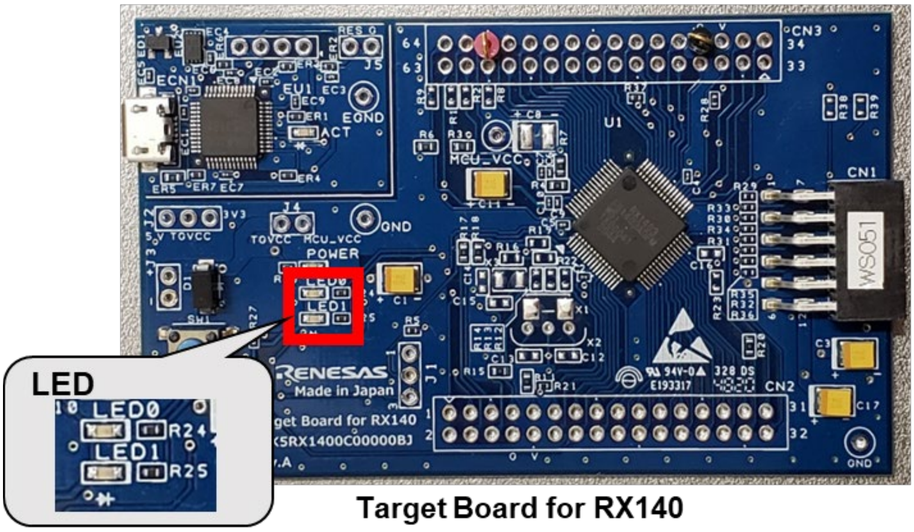


Figure 2.3 LEDs on Target Board for RX140

2.3.2 Using e^2 studio Functions to Monitor FIR Filter Operation

e^2 studio has many debugging functions. The sample program described in this application note is implemented a perspective (e2 studio screen configuration) for monitoring FIR filter outputs with the Waveform rendering function.

After connecting the debug tool, click the **FIR_Filter** button among the available perspectives (Figure 2.4). You can switch among the perspectives in this way.

Note that this perspective is included in the workspace setting information. Follow the procedure described in 2.1, Launching the Workspace, to use the workspace included in the distribution package with this application note.

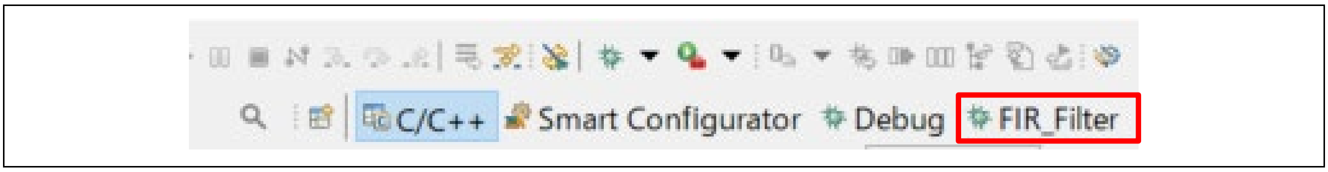


Figure 2.4 Switching among Perspectives

The FIR_Filter perspective (e² studio screen configuration) appears as shown in Figure 2.5.

Figure 2.5 FIR_Filter Perspective

Enable **Real-time Refresh** in the **Memory** view to update the display at the specified interval (Figure 2.6).

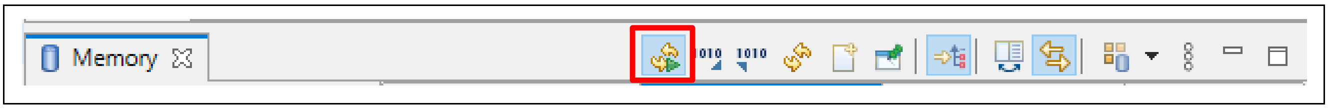


Figure 2.6 Enabling Real-Time Refresh

For information on how to use the **Waveform rendering** function in the **Memory** view, which is used in the FIR_Filter perspective, refer to 5.1, Monitoring Signal Processing in e² studio, and the help system in e² studio.

Help → Help Contents → e² studio User Guide → Debugging Projects → Views

- Memory → Waveform Memory Rendering

2.4 Modifiable Definitions

Table 2.1 and Table 2.2 list the system settings that can be changed by the user.

Table 2.1 Changeable Settings (main.h)

Function/Definition Name	Description	Default Value
Input data switching		
SAMPLE_DATA_MODE	Selects the signal source for FIR filter processing. 0: External input signal 1: Sample data	0
SELECT_SAMPLE_DATA	Selects the sample data used. Setting value: 1: A mix of three sine waves dominant of 156.25Hz 2: A mix of three sine waves dominant of 1250Hz 3: A mix of three sine waves dominant of 4000Hz	1
Sleep mode execution		
SLEEP_MODE	Selects whether or not to execute sleep mode. 0: Do not execute sleep mode. 1: Execute sleep mode.	0

Table 2.2 Changeable Settings (r_dsp_fir_config.h)

Function/Definition Name	Description	Default Value
FIR filter operation word length selection		
FIR_OPERATION*1	Selects the operation word length used for FIR filter processing. Setting value: 0: FIR_I16I16 1: FIR_I32I32 2: FIR_F32F32	0
FIR filter number of channels		
NUM_FIR_PROC	Selects the number of channels used for FIR filter processing. Setting value: 1, 2, 3	3
FIR filter setting values		
FIR_UNIT_INPUT*1	Specifies the processing unit sample count for FIR filter processing. Setting range: 16 to 1,024 (a power of 2)	256
FIR_TAPS*1	Number of FIR filter taps. Setting range: 64 to 256 (a power of 2)	64

Note: 1. The default settings of the sample program were decided based on the RX140 as the target device. Changing these setting values could prevent the program from operating properly due to insufficient RAM or excessive CPU processing load. Make sure to carefully estimate the resources that will be required before changing the settings.

3. Description of Sample Program

3.1 Overview of Sample Program

Table 3.1 lists the processing performed by the sample program.

Table 3.1 Roles of Processing

Processing	Role
Main processing	• Initializes peripheral functions and FIR filter processing. • Starts the first DTC transfer. • Notifies DTC transfer end interrupt processing of next DTC transfer destination address. • Performs normalization processing on input data. • Performs FIR filter processing. • Performs judgement of FIR filter processing results and LED illumination pattern display processing.
DTC transfer end interrupt processing	Makes settings for second and subsequent DTC transfers and starts transfer.

3.2 Processing Sequence

The processing sequence of the sample program consists primarily of the main processing and the DTC transfer end interrupt processing. Figure 3.1 shows the sequence for the main processing and DTC transfer end interrupt processing.

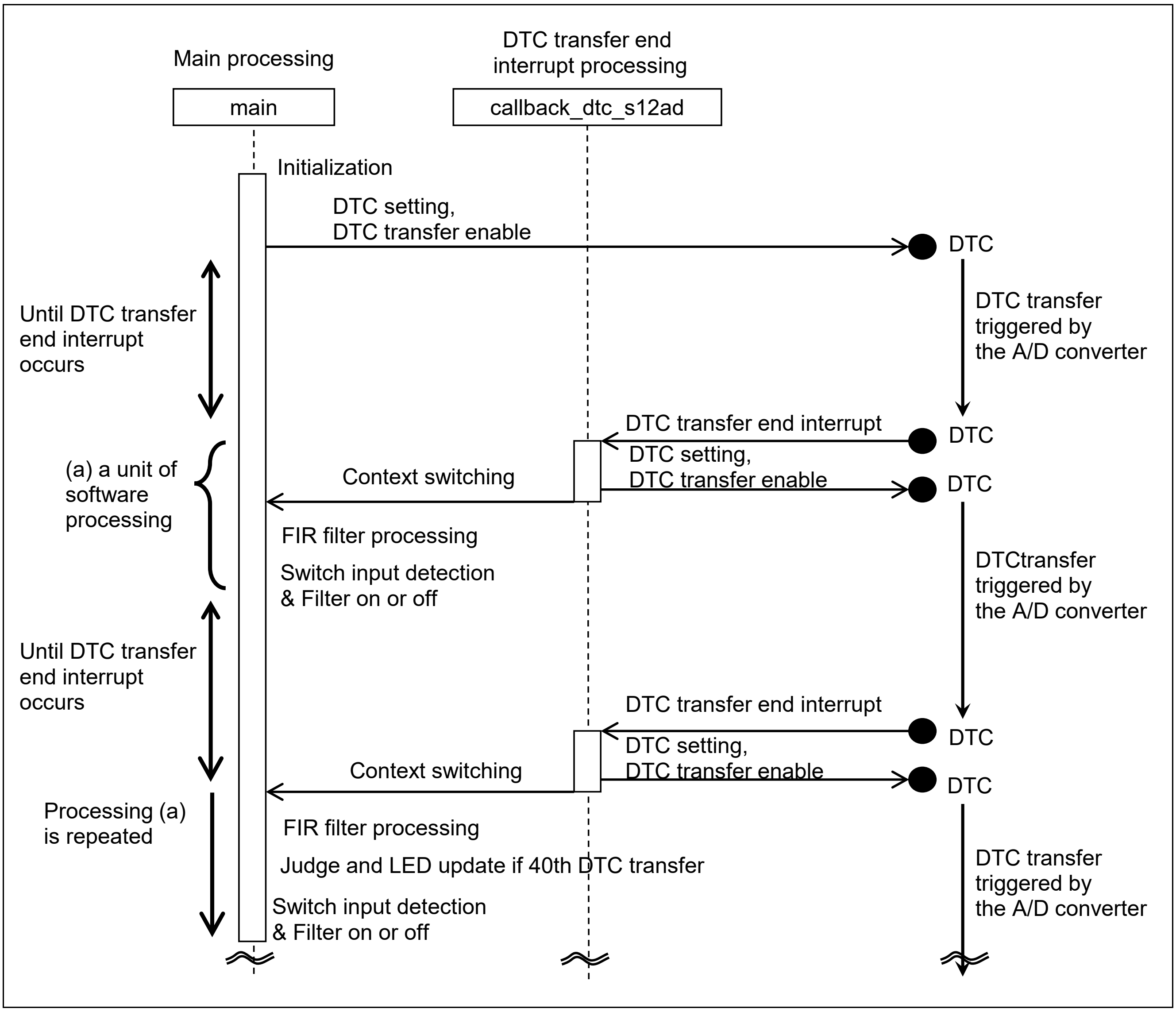


Figure 3.1 Sample Program Processing Sequence (Main Processing and DTC Transfer End Interrupt Processing)

As shown in the figure, the first DTC transfer is enabled in the main processing when A/D conversion ends. When DTC transfer of the specified number of samples finishes, a DTC transfer end interrupt request occurs. This triggers execution of the DTC transfer end interrupt processing and main processing, in that order. Thereafter, the same processing sequence is executed repeatedly.

After DTC transfer end interrupt processing has occurred 40 times (approximately 1 second), the Frequency Band judgement is performed, and the frequency band of the input signal is indicated by the two LEDs.

3.3 Processing Flowchart

Figure 3.2 shows a flowchart of the processing of the sample program.

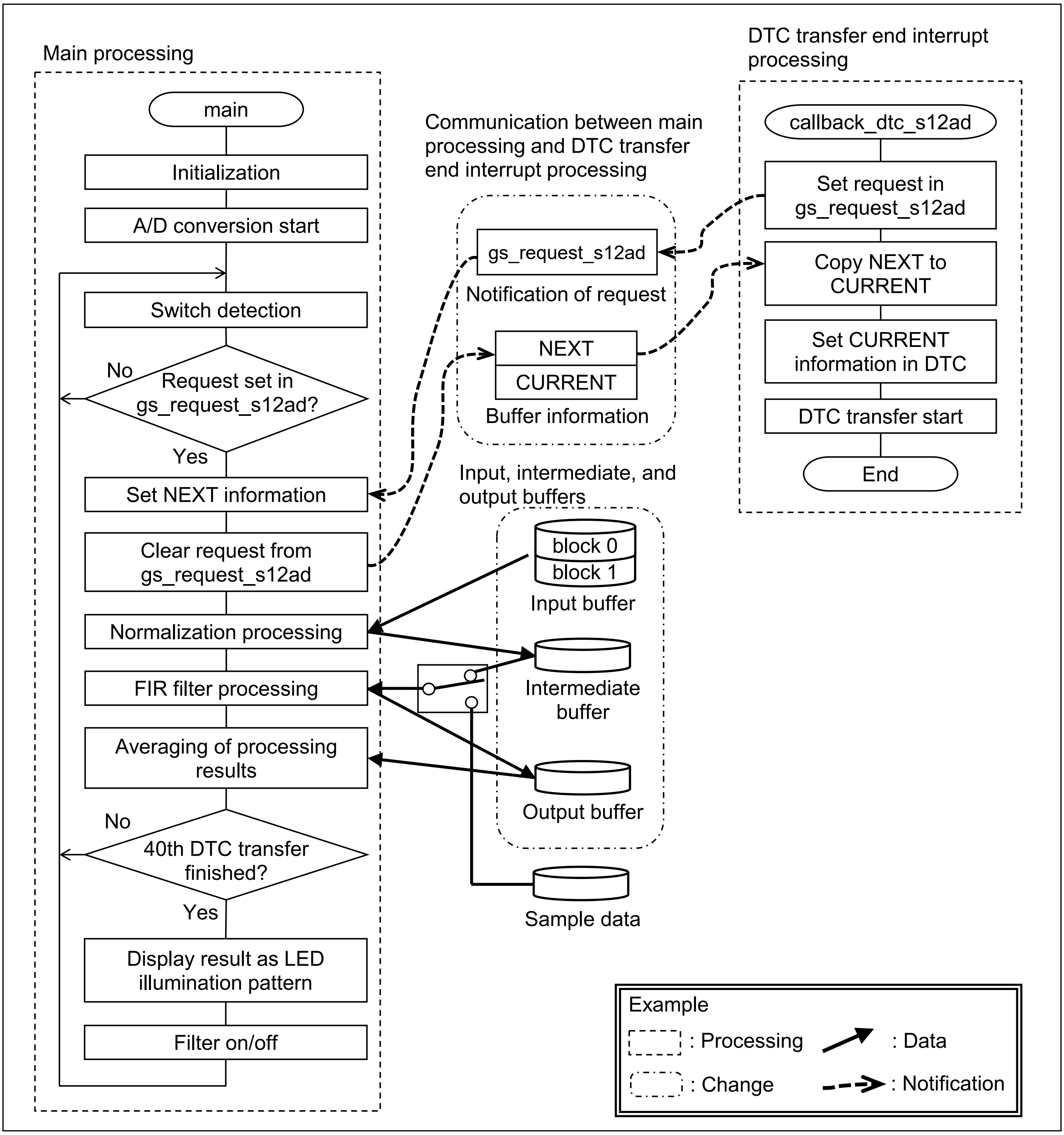


Figure 3.2 Processing Flowchart

The elements of Figure 3.2 are described below.

- Main processing
  Initialization is performed first.
  After initialization, repeat the following process.
  - Switch (SW1) operation detection
  - Determine if the A / D conversion result is stored in the buffer by DTC
    If the A / D conversion result is stored, perform the following processing.
    - Set the next DTC transfer destination and perform FIR filter processing.
    - If a switch operation is detected, the FIR filter is switched ON / OFF.

- DTC transfer end interrupt processing
  A request is sent to the main processing to update the next DTC transfer destination, and the current DTC transfer is started.

- Communication between main processing and DTC transfer end interrupt processing
  Specific variables are used for communication between the main processing and the DTC transfer end interrupt processing when the main processing notifies the DTC transfer end interrupt processing of the next DTC transfer destination and when the DTC transfer end interrupt processing requests the main processing to update the next DTC transfer destination. Table 3.2 lists the variables used for communication.

- Input buffer
  The DTC stores the A/D conversion result in the input buffer, and it is read as input data for FIR filter processing. The input buffer is configured as two blocks to avoid access conflicts between the CPU (FIR filter processing) and the DTC. Switching between the buffer blocks accessed by the CPU and DTC is triggered by the DTC transfer end interrupt. As shown in Figure 3.3, the DTC transfer end interrupt is used as a trigger for switching between the buffer blocks. Figure 3.4 shows the configuration of the input buffer.

- Intermediate buffer
  The intermediate buffer stores the results of normalization processing. Figure 3.4 shows the configuration of the intermediate buffer.

- Output buffer
  The output buffer stores the results of FIR filter processing. FIR filter processing is divided into three channels. Figure 3.4 shows the configuration of the output buffer.

- Sample data
  Sample data is used when macro definition SAMPLE_DATA_MODE is set to 1. In this case, sample data is used as the input for FIR filter processing instead of external signal input data.

Table 3.2 Variables for Communication Between Main and DTC Transfer End Interrupt Processing

Type		Description
(a) Request notification		The DTC transfer end interrupt processing uses this variable to request the main processing to update the buffer information (b). The request is set by the DTC transfer end interrupt processing. When the variable is set, the main processing updates the NEXT plane of buffer information (b) and clears the request. The request is first cleared by the main processing during initialization.
(b) Buffer information		The start address of the DTC transfer destination buffer and the number of data units it contains are stored in this variable. The buffer consists of two planes, NEXT and CURRENT, to avoid access conflicts between different types of processing. The initial values of NEXT and CURRENT are set by the main processing.
	NEXT	The main processing stores the start address and the data count in this plane in response to request notification (a). The initial values are as follows: • Start address: Data unit 0 of block 1 in the input buffer • Data count: 256
	CURRENT	The DTC transfer end interrupt processing refers to the start address and data count stored in this plane to make settings in the DTC. The DTC transfer end interrupt processing itself copies the information in NEXT to CURRENT. The initial values are as follows: • Start address: Data unit 0 of block 0 in the input buffer • Data count: 256

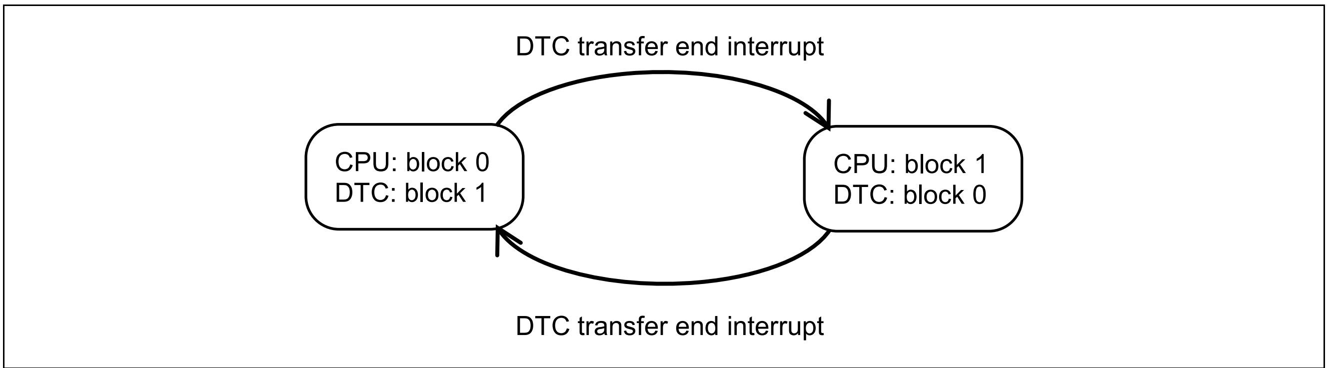


Figure 3.3 Switching Input Buffer Blocks Accessed by CPU and DTC

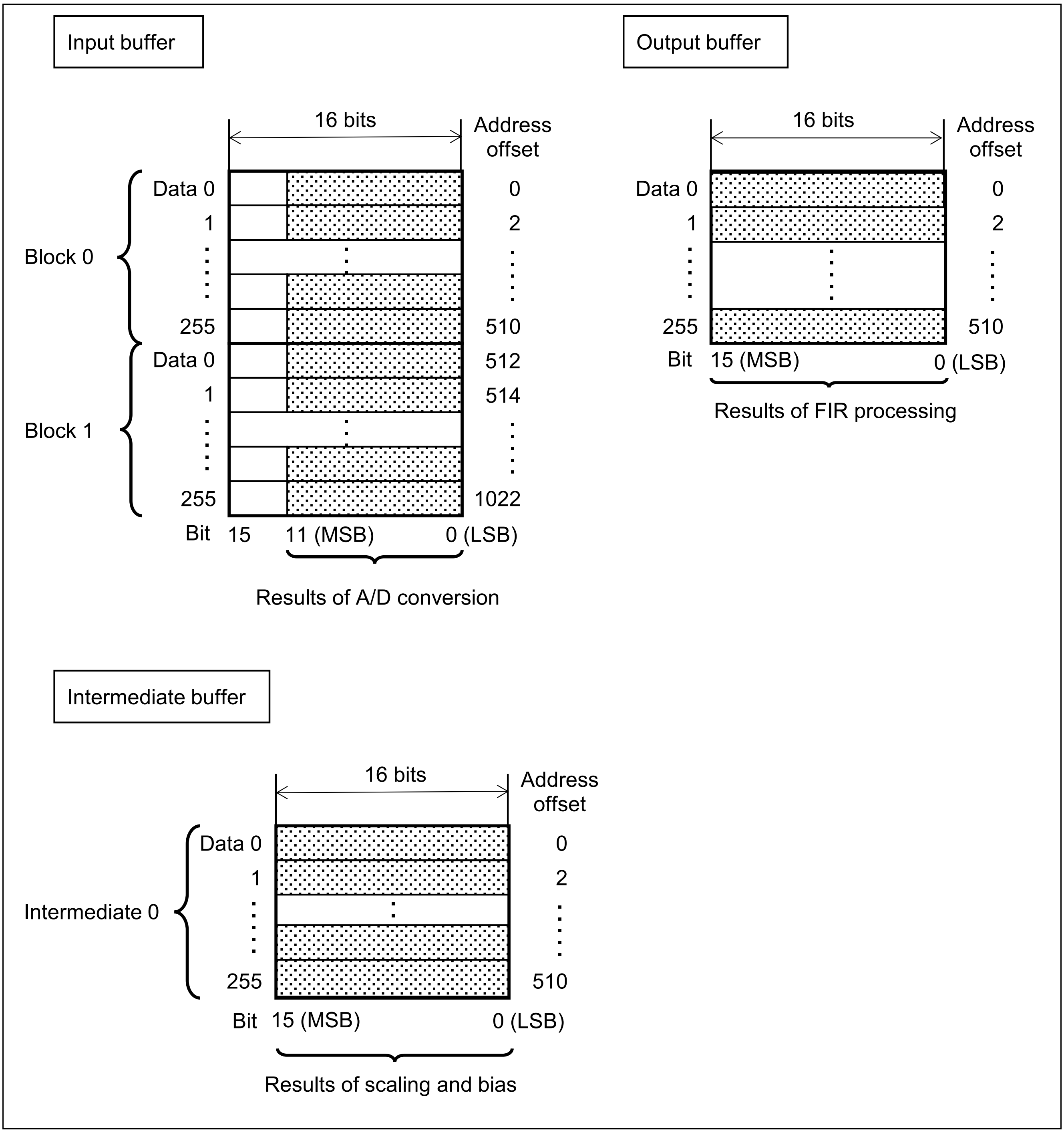


Figure 3.4 Input, Intermediate, and Output Buffers

3.4 Details

3.4.1 Initialization

The sample program performs initialization processing in the following sequence.

1. Initialization of peripheral functions
   The CMT, S12AD, and ELC are initialized.
2. Initialization of FIR filter processing
   FIR filter processing is initialized and coefficients are set to the each channels.
3. Initialization of variables for communication between main processing and DTC transfer end interrupt processing
   The variables listed in Table 3.2, Variables for Communication Between Main and DTC Transfer End Interrupt Processing, are initialized.
4. Enabling of DTC transfers
   Settings including the activation source, transfer mode, transfer source address, transfer destination address, and transfer data count are configured, and DTC transfers are enabled.

After this, DTC transfers begin when A/D conversion operation starts.

3.4.2 Normalization Processing

The sample program performs the normalization processing (bias processing and scaling) shown in Figure 3.5.

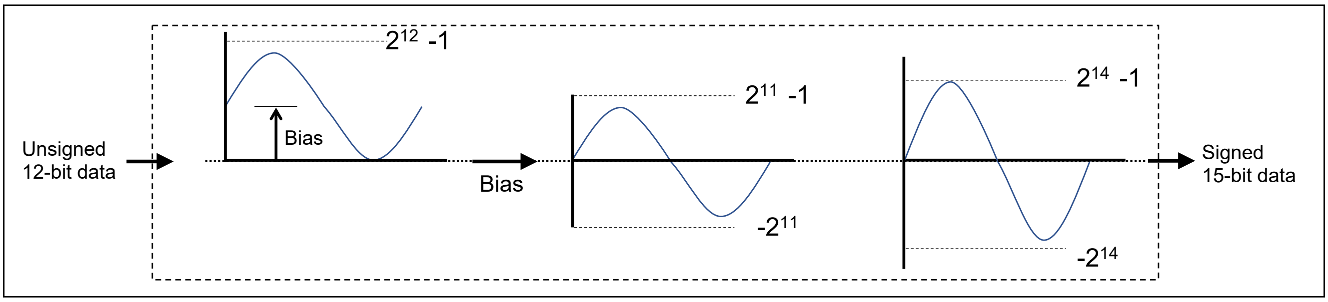


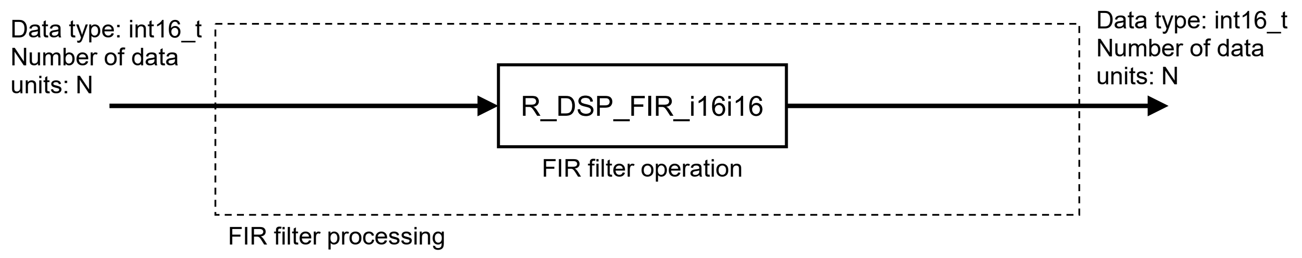
Figure 3.5 Normalization Processing

- Components of normalization processing
  The data input by the S12AD is in 12-bit (unsigned) format, so it must be normalized to 15-bit (signed) format in order to obtain adequate operation results from FIR filter processing. The normalization processing consists of bias processing and scaling.

- Normalization processing of 32-bit integer type and single-precision floating-point data
  With 32-bit integer type data, normalization processing converts 12-bit (unsigned) integer data output by the A/D converter into 31-bit (signed) data. The range of values is -2^{30} to $2^{30} - 1$.
  With single-precision floating-point data, normalization processing converts 12-bit (unsigned) integer data output by the A/D converter into single-precision floating-point data. The range of values is −1.0 to 1.0.

3.4.3 FIR Filter Processing

The sample program uses an API function from the DSP library to perform filter processing on the input signal as shown in Figure 3.6.



R_DSP_FIR_i16i16	FIR filter operation function
int16_t	16-bit signed integer type
N	FIR filter sample count (256)

Figure 3.6 Flow of Data in FIR Filter Processing

- API function comprising FIR filter processing
  FIR filter processing is performed by the FIR filter operation function R_DSP_FIR_i16i16. R_DSP_FIR_i16i16 outputs operation results according to the filter coefficient settings provided to the function. The sample program has three presets of filter characteristics: LPF, BPF, and HPF. These are assigned to Ch.0, Ch.1, and Ch.2, respectively.
- Input and output data
  The sample program inputs 256 samples of data as type int16_t for FIR filter processing, and also outputs data as type int16_t.

For details of the API specifications, refer to RX DSP Library Version 5.0 API User's Manual: Software.

3.4.4 Averaging of FIR Filter Processing Results

Average values for each of the preset filter characteristics from the FIR filter processing results are stored in the output buffer (gs_output_buffer). For each 256 samples, the data stored in the output buffer is averaged as described below. The data stored in the output buffer is signed, so absolute values are used in the calculations. This processing is performed by the get_average_value function in main.c.

The processing is performed on the FIR filter processing results on all three channels.

Averaging = $\sum_{i=0}^{255} |gs\_output\_buffer[i]| \,/256$

3.4.5 Frequency Band Judgement Based on FIR Filter Processing Results

Figure 3.7 illustrates the frequency band judgement.

The frequency band of the input signal is determined every second using FIR filter processing and averaging for 3 channels. 40 times of averaging process performed per second. Because the sampling frequency is 10kHz and the processing unit is 256 samples.

The averaged results for 40 times are all added up, the values of Ch.0, Ch.1, and Ch.2 are compared, and the largest value becomes the filter frequency band result, which is indicated by the LED illumination pattern. Comparison processing is performed by evaluate_max_values in main.c, and LED illumination processing is performed by display_led in main.c.

The illumination patterns, which correspond to the filter characteristics, are listed in Table 3.3.

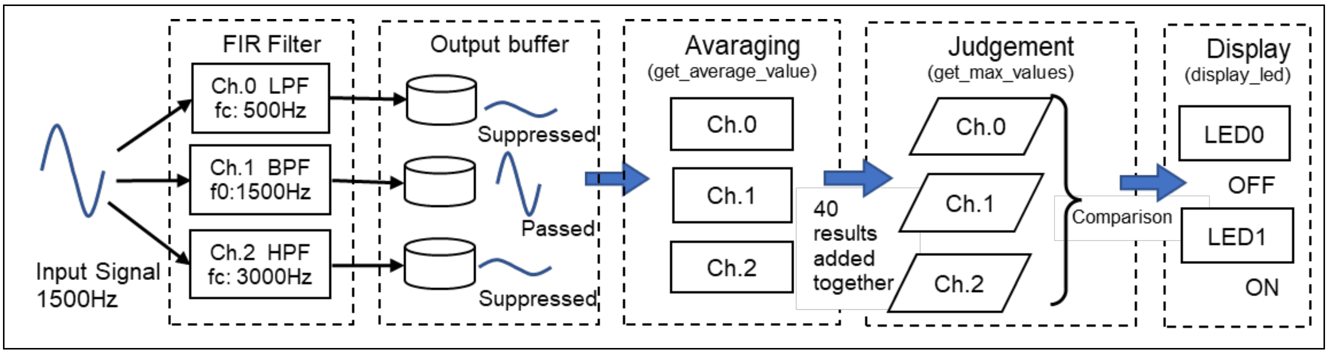


Figure 3.7 Illustration of Frequency Band Judgement

Table 3.3 LED Illumination Patterns for Display of Results of FIR Filter Processing

	LED0	**LED1**
Ch.0 (LPF)	ON	OFF
Ch.1 (BPF)	OFF	ON
Ch.2 (HPF)	ON	ON
Judgement not possible	OFF	OFF

3.5 File Structure

Table 3.4 shows the file structure of the software modules implemented in the sample program. In addition, Table 3.5 to Table 3.9 list the functions in the source files. For details of the other modules, refer to their respective application notes.

Table 3.4 File Structure of Software Modules

Module/File		Description
Main		Main processing of sample program
	main.c	Main processing, DTC transfer end interrupt processing, etc.
	main.h	Header file of main.c
r_dsp		FIR-related processing
	r_normalize.c	Normalization processing
	r_normalize.h	Header file of r_normalize.c
	r_dsp_fir_i16i16.c	Initialization and filtering of FIR filter processing for 16-bit signed integer
	r_dsp_fir_i32i32.c	Initialization and filtering of FIR filter processing for 32-bit signed integer
	r_dsp_fir_f32f32.c	Initialization and filtering of FIR filter processing for 32-bit floating-point
	r_dsp_fir.h	Header file of r_dsp_fir_i16i16.c, r_dsp_fir_i132i32.c, and r_dsp_fir_f32f32.c
	r_dsp_fir_config.h	FIR filter configuration file
	r_coef_lpf_*.c	LPF coefficient files Cutoff frequency: 500 [Hz] Note: Substitute i16, i32, or f32 depending on the operation word length
	r_coef_bpf_*.c	BPF coefficient files Center frequency: 1,500 [Hz] Note: Substitute i16, i32, or f32 depending on the operation word length
	r_coef_hpf_*.c	HPF coefficient files Cutoff frequency: 3,000 [Hz] Note: Substitute i16, i32, or f32 depending on the operation word length
	r_coef_flat_*.c	Coefficient file with flat characteristics Note: Substitute i16, i32, or f32 depending on the operation word length
	r_wave_sample_*_lpf.c	Sample waveform files to check the LPF A mix of three sine waves dominant of 156.25Hz Note: Substitute i16, i32, or f32 depending on the operation word length
	r_wave_sample_*_bpf.c	Sample waveform files to check the BPF A mix of three sine waves dominant of 1250Hz Note: Substitute i16, i32, or f32 depending on the operation word length
	r_wave_sample_*_hpf.c	Sample waveform files to check the HPF A mix of three sine waves dominant of 4000Hz Note: Substitute i16, i32, or f32 depending on the operation word length

Table 3.5 main.c File Functions

Function Name	Description
main	• Initializes peripheral devices. • Starts first DTC transfer. • Notifies DTC transfer end interrupt processing of next DTC transfer destination. • Performs FIR filter processing. • Displays judgement of processing results on LEDs.
set_buf_info	Sets the next DTC transfer destination address and data count in the variables used for communication between the main processing and the DTC transfer end interrupt processing.
dtc_init	Initializes the DTC with S12AD channel 0 as the transfer source.
dtc_start	Stores transfer information in the DTC and starts the DTC.
callback_dtc_s12ad	The callback function registered in the DTC module by the main processing • Performs DTC transfer end interrupt processing. • Sets the next transfer destination in the DTC and enables the DTC transfer. • Requests the next transfer destination information from the main processing.
get_average_value	Fetches the average sample values stored in the output buffer.
evaluate_max_values	Compares the three average values output by get_average_value and makes a judgement as to the largest value.
display_led	Updates the LED illumination pattern based on the result of evaluate_max_values.

Table 3.6 r_normalize.c File Functions

Function Name	Description
R_Normalize_Operation	Performs normalization processing.

Table 3.7 r_dsp_fir_i16i16.c File Functions

Function Name	Description
R_DSP_FIR_Init	Initializes FIR filter processing.
R_DSP_FIR_Change_Coef	Sets FIR filter coefficients.
R_DSP_FIR_Operation	Executes FIR filter processing.

Table 3.8 r_dsp_fir_i32i32.c File Functions

Function Name	Description
R_DSP_FIR_Init	Initializes FIR filter processing.
R_DSP_FIR_Change_Coef	Sets FIR filter coefficients.
R_DSP_FIR_Operation	Executes FIR filter processing.

Table 3.9 r_dsp_fir_f32f32.c File Functions

Function Name	Description
R_DSP_FIR_Init	Initializes FIR filter processing.
R_DSP_FIR_Change_Coef	Sets FIR filter coefficients.
R_DSP_FIR_Operation	Executes FIR filter processing.

4. Usage Notes

4.1 Error in FIR Filter Results Due to HOCO Clock Error

The RX140 on the target board operates using HOCO as the clock source. HOCO has a maximum error of ±2%, and this is the sampling frequency error of the A/D conversion processing performed by the sample program. This sampling frequency error can show up as error in the FIR filter cutoff frequency, etc. For applications requiring a system capable of more accurate FIR filter processing than the sample program, use a highly accurate oscillator as the clock source of the RX140 and make corresponding changes to the clock settings of the sample program.

4.2 Aliasing

The evaluation board used with the sample program described in this application note does not incorporate any measures to deal with aliasing in the A/D conversion input signal of the RX140. Aliasing occurs when the frequency of the A/D conversion input is higher than one-half the sampling frequency. When designing a system with this application note as a reference, consider adding an external antialiasing filter to the RX140 as necessary.

5. Reference

5.1 Monitoring Signal Processing in e2 studio

You can use the waveform rendering function of e2 studio to monitor the signal input to the RX140 and the FIR filter processing results.

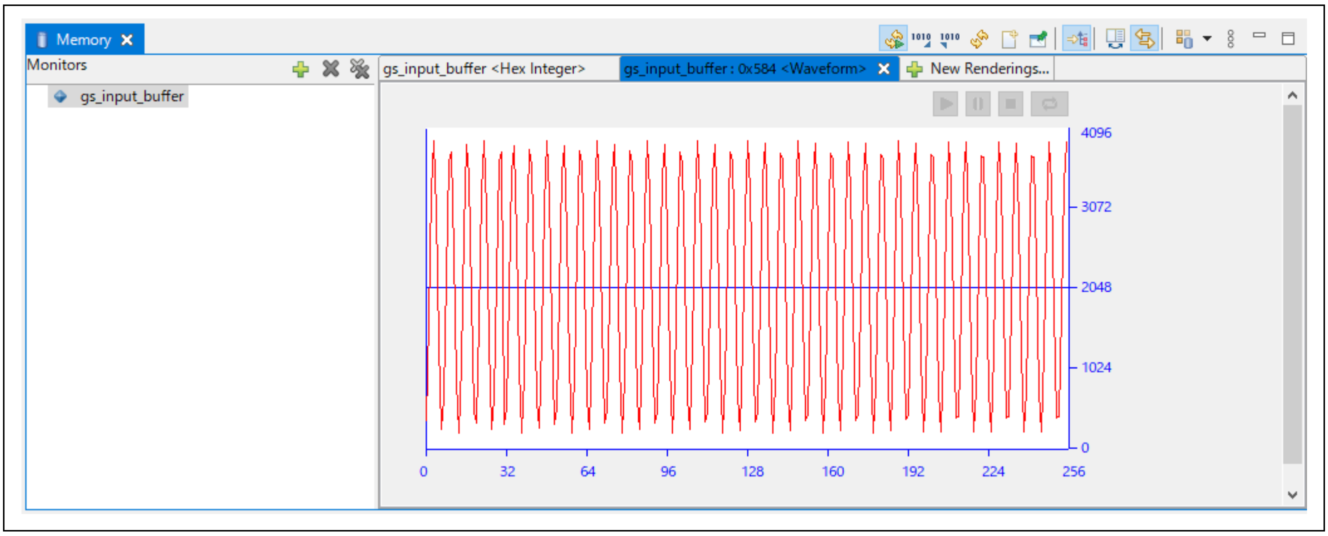


Figure 5.1 Waveform Rendering Display Example (Data Stored in Input Buffer)

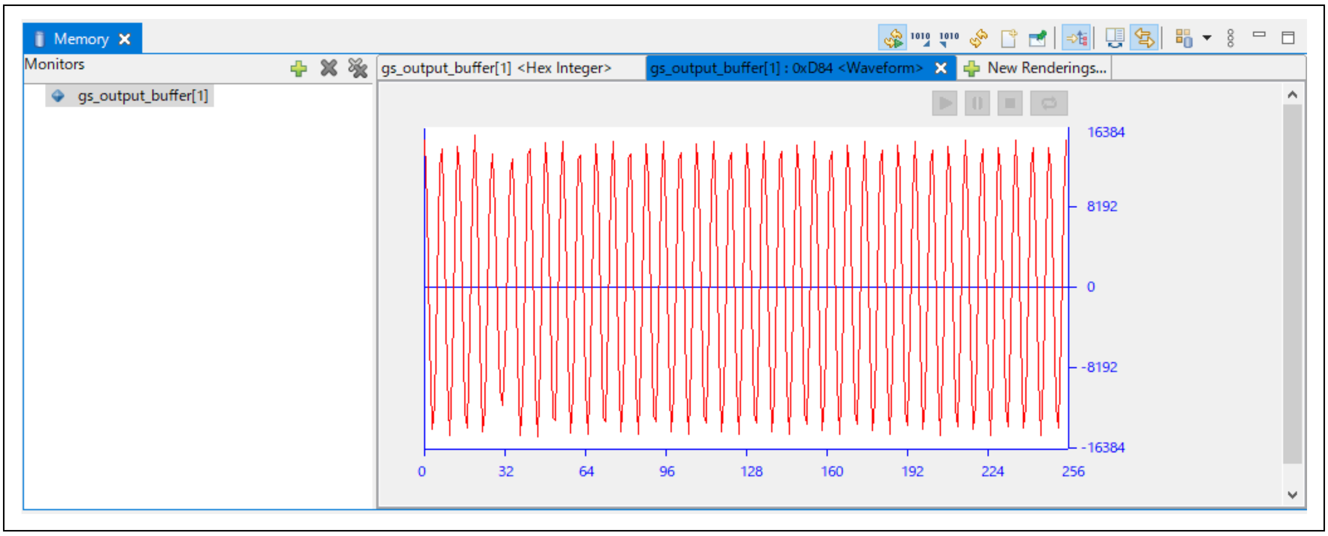


Figure 5.2 Waveform Rendering Display Example
(Frequency Magnitude Characteristics Produced by FIR Filter Processing)

After connecting the RX140 to the debugger, select **Window** → **Show View** → **Memory** to display the **Memory** view. When the **Memory** view appears, specify the input buffer (gs_input_buffer) as the variable to monitor. After adding the input buffer, specify the output buffer (gs_output_buffer) in the same manner.

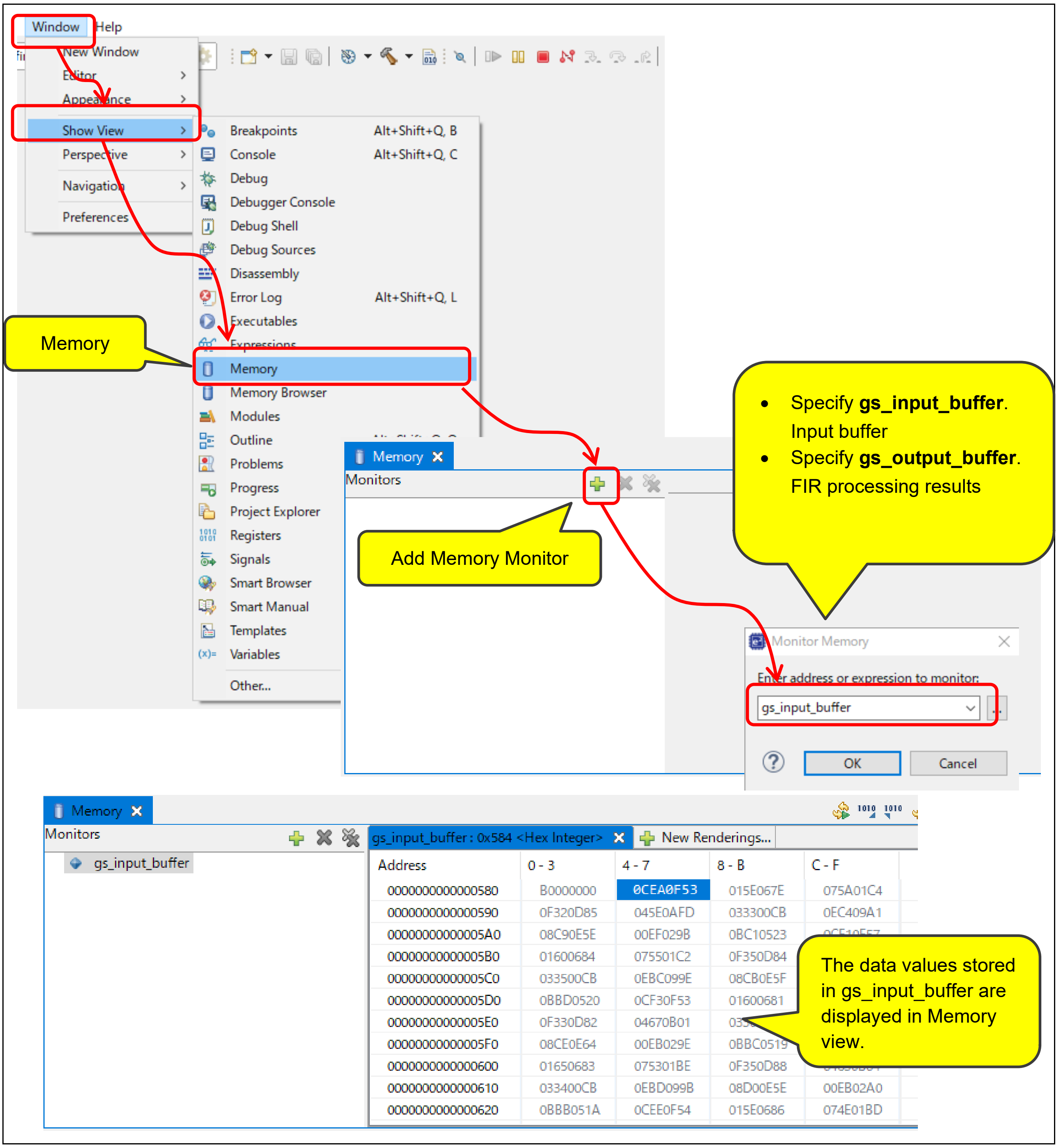


Figure 5.3 Memory View Display Procedure

Next, select **New Renderings...** → **Waveform** → **Add Rendering(s)** to choose a variable to be displayed graphically. When the **Waveform Properties** window appears, enter the necessary settings for the variable, and finally click the **OK** button to show the graphical display.

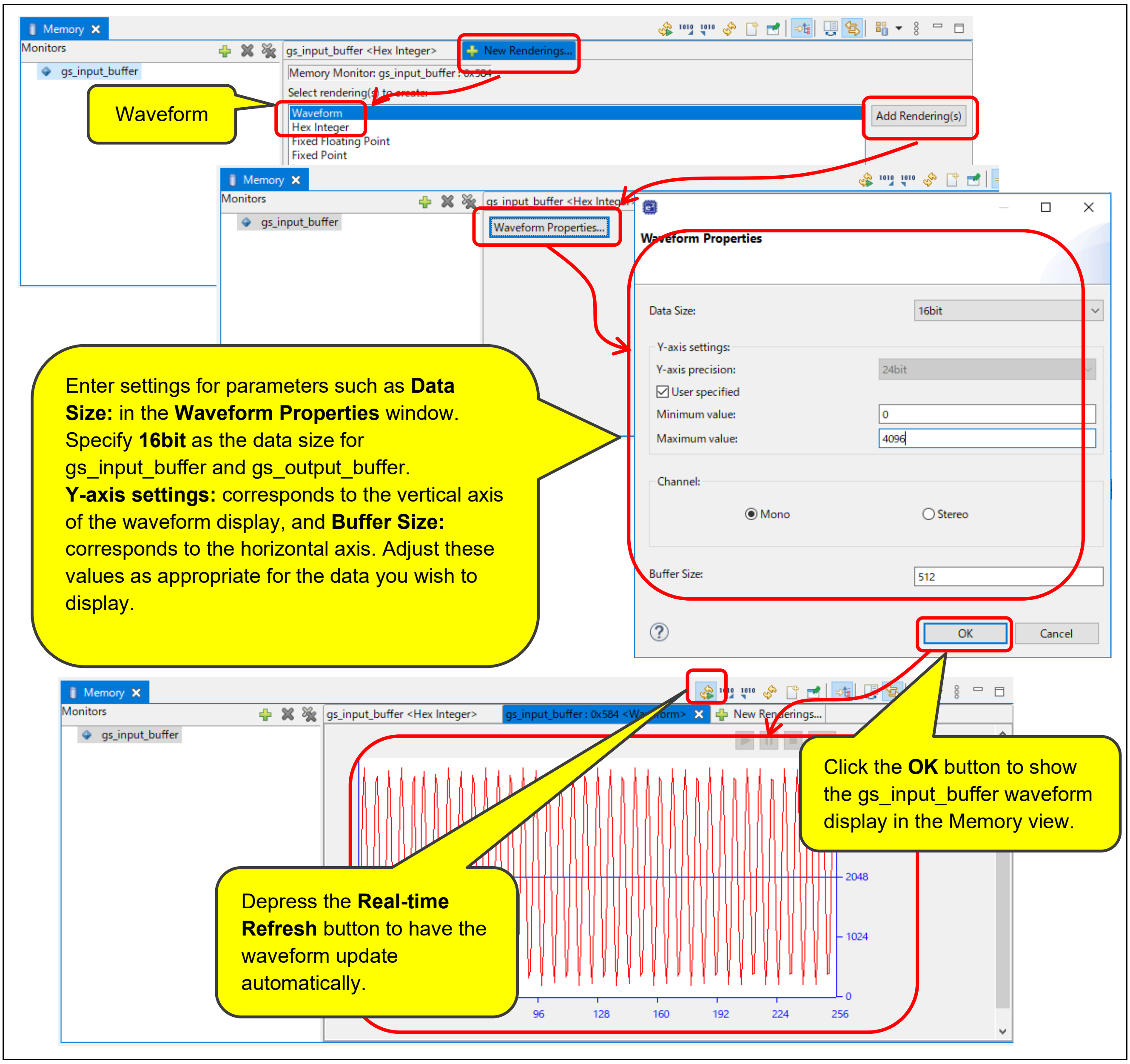


Figure 5.4 Waveform Rendering Display Procedure

5.2 Memory Usage

Table 5.1 lists the memory usage of the sample program.

The figures for ROM and RAM were calculated based on a .map file generated under the conditions listed in Table 5.1. The figures for user stack and interrupt stack were obtained by measuring the actual stack memory usage while the sample program was running.

Table 5.1 Sample Program Memory Usage (Reference)

Item	Measured Value [Bytes]	Description
ROM	13801	Sample program ROM usage
RAM	12603	Sample program RAM usage
User stack	116	Sample program stack memory usage
Interrupt stack	128	

5.3 Resources Consumed by FIR Filter Processing and Hints on Selecting MCUs

5.3.1 Cycle Count, RAM Consumption, CPU Usage, and RAM Usage

Table 5.2 to Table 5.6 list examples of the resource consumption of the sample program for FIR filter processing on one channel under various conditions.

- Items such as CPU usage and RAM usage are indices used to determine whether or not system implementation is possible. Depending on conditions such as operation word length, sampling frequency, and number of taps, the RX140 may be insufficient in areas such as the execution cycle count or RAM. Refer to Table 5.2 to Table 5.6 to determine whether or not system implementation is possible and select an appropriate Renesas MCU, paying particular attention to points such as CPU usage and RAM usage.
- Table 5.2 lists resource consumption using the default settings of the sample program, except for the number of channels. We recommend that Table 5.2 be compared with the other tables that follow. By comparing the tables, you can get an understanding of how resource consumption increases or decreases when each condition changes.
- The measurement values in each table include the elements essential for FIR filter processing, such as the buffer memory defined in main.c, in addition to the functions, variables, constants, etc., in the source files r_dsp_fir_i16i16.c, r_dsp_fir_i32i32.c, and r_dsp_fir_f32f32.c.
- The values listed in the tables are measurement values for the RX140. The items that are not listed are sample program settings.
  For details of the measurement conditions, refer to 5.3.2, Resource Consumption Measurement Conditions, below.

Table 5.2 Resource Consumption of FIR Filter Processing (1 Channel)

	Channels: 1, Taps: 64, Fs: 10 KHz, Unit Samples: 256		
	r_dsp_fir_i16i16.c	**r_dsp_fir_i32i32.c**	**r_dsp_fir_f32f32.c**
Cycle count (CPU usage)	59497 cycles (4.84 %)	87713 cycles (7.14 %)	135317 cycles (11.01 %)
RAM (RAM usage)	3108 bytes (18.97 %)	6180 bytes (37.72 %)	6180 bytes (37.72 %)
Stack	SU : 108 bytes SI : 0 bytes	SU : 120 bytes SI : 0 bytes	SU : 120 bytes SI : 0 bytes
ROM (ROM usage)	823 bytes (1.26 %)	2626 bytes (4.01 %)	1142 bytes (1.74 %)

Table 5.3 Resource Consumption of FIR Filter Processing (3 Channels)

	Channels: 3, Taps: 64, Fs: 10 KHz, Unit Samples: 256		
	r_dsp_fir_i16i16.c	**r_dsp_fir_i32i32.c*1**	**r_dsp_fir_f32f32.c*1**
Cycle count (CPU usage)	180554 cycles (14.69 %)	-	-
RAM (RAM usage)	8300 bytes (50.66 %)	-	-
Stack	SU : 112 bytes SI : 0 bytes	-	-
ROM (ROM usage)	1207 bytes (1.84 %)	-	-

Note: 1. The description is omitted because the RX140 (RAM: 16 KB) has insufficient RAM.

Table 5.4 Resource Consumption of FIR Filter Processing (Taps: 256)

	Channels: 1, Taps: 256, Fs: 10 KHz, Unit Samples: 256		
	r_dsp_fir_i16i16.c	**r_dsp_fir_i32i32.c**	**r_dsp_fir_f32f32.c**
Cycle count (CPU usage)	213098 cycles (17.34 %)	308899 cycles (25.14 %)	503446 cycles (40.97 %)
RAM (RAM usage)	3108 bytes (18.97 %)	6180 bytes (37.72 %)	6180 bytes (37.72 %)
Stack	SU : 108 bytes SI : 0 bytes	SU : 120 bytes SI : 0 bytes	SU : 120 bytes SI : 0 bytes
ROM (ROM usage)	825 bytes (1.26 %)	2627 bytes (4.01 %)	1143 bytes (1.74 %)

Table 5.5 Resource Consumption of FIR Filter Processing (Unit Processing Sample Count: 64)

	Channels: 1, Taps: 64, Fs: 10 KHz, Unit Samples: 64		
	r_dsp_fir_i16i16.c	**r_dsp_fir_i32i32.c**	**r_dsp_fir_f32f32.c**
Cycle count (CPU usage)	14946 cycles (4.87 %)	21987 cycles (7.16 %)	34003 cycles (11.07 %)
RAM (RAM usage)	804 bytes (4.91 %)	1572 bytes (9.59 %)	1572 bytes (9.59 %)
Stack	SU : 108 bytes SI : 0 bytes	SU : 120 bytes SI : 0 bytes	SU : 120 bytes SI : 0 bytes
ROM (ROM usage)	819 bytes (1.25 %)	2622 bytes (4.00 %)	1138 bytes (1.74 %)

Table 5.6 Execution Cycle Count of FIR Filter Processing (By Sampling Frequency)*1

	Channels: 1, Taps: 64, Unit Samples: 256		
Fs [Hz]	**r_dsp_fir_i16i16.c**	**r_dsp_fir_i32i32.c**	**r_dsp_fir_f32f32.c**
1,000 (CPU usage)	59497 cycles (0.48 %)	87713 cycles (0.71 %)	135059 cycles (1.01 %)
10,000 (CPU usage)	59497 cycles (4.84 %)	87713 cycles (7.14 %)	135317 cycles (11.01 %)
16,000 (CPU usage)	59497 cycles (7.75 %)	95394 cycles (12.42 %)	135317 cycles (17.62 %)
24,000 (CPU usage)	59497 cycles (11.62 %)	95394 cycles (18.63 %)	135317 cycles (26.43 %)

Note: 1. Refer to Table 5.2 for usage of resources of such as RAM.

5.3.2 Resource Consumption Measurement Conditions

The resource consumption measurement conditions underlying the values in Table 5.2 to Table 5.6 are listed below.

- The measurement targets are the processing, memory usage, etc., associated with execution of R_DSP_FIR_Operation. (Refer to Table 5.1, Sample Program Memory Usage (Reference), for the overall resource consumption of the sample program.)

- The sample program was run in the following conditions, with the settings in r_dsp_fir_config.h changed as indicated, and the results were measured.
  Main measurement conditions：
  - Integrated development environment: Renesas Electronics e2 studio 2021-10
  - C compiler Renesas Electronics RX Compiler: CC-RX V3.03.00
  - Sample program: Version 1.00

  For other conditions, refer to Table 1.2 Operation Confirmation Conditions.

- Cycle count and CPU usage
  The execution cycle of R_DSP_FIR_Operation was measured. CPU usage is the ratio of the execution cycle count of R_DSP_FIR_Operation to the execution cycle count for data input of 256 samples at a sampling frequency or 10 kHz.
  Example: CPU usage of r_dsp_fir_i16i16.c in Table 5.2
  (a) Processing execution cycle count of R_DSP_FIR_Operation: 59,497 [cycles]
  (b) Cycle count for storage of unit processing sample count: 1,228,800 [cycles]
  (c) CPU usage: 4.84 [%]= (a) / (b) × 100[%]

- RAM and RAM usage
  RAM refers to the total RAM consumption of r_dsp_fir_i16i16.c, r_dsp_fir_i32i32.c, or r_dsp_fir_f32f32.c as well as gs_intemediate_buffer and gs_output_buffer. RAM usage refers to the RAM consumption of FIR filter processing as a percentage of the total RAM capacity of the RX140.
  Example: RAM consumption percentage of r_dsp_fir_i16i16.c in Table 5.2 (RAM required for FIR filter processing)
  (a) RAM required for FIR filter processing: 3,108 [bytes]
  (b) RAM capacity of RX140: 16,384 [bytes]
  (c) RAM usage: 18.97 [%]= (a) / (b) × 100[%]

- Stack
  The maximum amount of stack memory used by R_DSP_FIR_Init and R_DSP_FIR_Operation.

- ROM and ROM usage
  ROM refers to the ROM consumption of r_dsp_fir_i16i16.c, r_dsp_fir_i32i32.c, or r_dsp_fir_f32f32 as well as one coefficient file and the DSP library. ROM usage refers to the ROM consumption of FIR filter processing as a percentage of the total ROM capacity of the RX140.
  Example: ROM usage of r_dsp_fir_i16i16.c in Table 5.2 (ROM required for FIR filter processing)
  (a) ROM required for FIR filter processing: 823 [bytes]
  (b) ROM capacity of RX140: 65,536 [bytes]
  (c) ROM usage: 1.26 [%]= (a) / (b) × 100[%]

5.4 Reducing CPU Load

Once configured, the S12AD, CMT, ELC, and DTC operate without the intervention of the CPU. You can reduce the CPU load by taking advantage of this feature, running the CPU in normal operating mode when using software to perform processing and transitioning the CPU to sleep mode by means of the WAIT instruction at all other times.

To make use of this function, refer to Table 2.1 and change the setting value of the macro SLEEP_MODE defined in main.h. Note that the waveform rendering function may not work properly when in sleep mode. Do not use sleep mode if you need to use the waveform rendering function.

5.5 Software Module Settings

Table 5.7 to Table 5.12 list the FIT module settings, e^2 studio Smart Configurator settings, and DSP library settings used in the sample program. For Smart Configurator settings, the items and setting details match those displayed on the setting menu. For details of the software modules, refer to the application notes listed in 8, Reference Document.

Table 5.7 BSP Module Settings

Category	Item	Setting/Description
Smart Configurator >> Components >> r_bsp		Other than the changes listed below, properties are left in the default settings.
	Parameter checking	Disabled
	Heap size	0x900
Smart Configurator >> Clock		The following settings are made on the “Clocks” tab and reflected in r_bsp_config.h.
	VCC setting	3.3 (V)
	(Sample project for target board) Main clock settings	Operation: Unchecked.
	Sub-clock oscillator setting	Operation: Unchecked.
	(Sample project for target board) HOCO clock settings	Operation: Checked. Frequency: 48 (MHz)
	LOCO clock settings	Operation: Unchecked.
	(Sample project for target board) System clock settings	Clock source: HOCO System clock (ICLK): ×1 48 (MHz) Peripheral module clock (PCLKB): ×1/2 24 (MHz) Peripheral module clock (PCLKD): ×1 48 (MHz) FlashIF clock (FCLK): ×1 48 (MHz)
	IWDT dedicated clock settings	Operation: Unchecked.

Table 5.8 DTC Module Settings

Category	Item	Setting/Description
r_dtc_rx_config.h		Other than the changes listed below, default settings are used.
	DTC_CFG_USE_DMAC_FIT_MODULE	This setting specifies whether the DMAC module is used while the DTC module is in use. Changed to DTC_DISABLE because the RX140 does not have a DMAC module.

Table 5.9 Smart Configurator Settings (S12AD)

Category	Item	Setting
Smart Configurator >> Components >> Single Scan Mode S12AD (Config_S12AD0)		Code is generated using the settings below.
	Analog input mode setting	Double trigger mode: Unchecked.
	Analog input channel setting	Only AN000 checked.
	Conversion start trigger setting	Start trigger source: Trigger from ELC
	Interrupt settings	Enable A/D conversion end interrupt (S12ADI0) checked. Priority: Level 0 (disabled)
	Add/average AD conversion value setting	AN000 unchecked.
	A/D conversion select	High-speed
	High-potential reference voltage select	AVCC0
	Low-potential reference voltage select	ACSS0
	Self-diagnosis setting	Mode: Unused.
	Disconnection detection assist setting	Charge setting: Unused.
	Data register settings	Data placement: Right-alignment Automatic clearing: Disable automatic clearing Addition/average mode select: Addition mode Addition count: 1-time
	Data storage buffer setting	Disable
	Window function setting	Disable
	Window A/B operation settings	Enable comparison window A: Unchecked. Enable comparison window B: Unchecked.
	Input sampling time setting	AN000: 0.407 (μs)
	Event link control setting	ELC scan end event generation condition: On completion of all scans

Table 5.10 Smart Configurator Settings (CMT1)

Category	Item	Setting
Smart Configurator >> Components >> Compare Match Timer (Config_CMT1)		Code is generated using the settings below.
	Count clock settings	PCLK/8
	Compare match setting	Interval value: 100 μs (actual value: 100) Register value (CMCOR): 299 Compare match interrupt (CMI1): Unchecked.

Table 5.11 Smart Configurator Settings (ICU)

Category	Item	Setting
Smart Configurator >> Components >> Interrupt Controller (Config_ICU)		Code is generated using the settings below.
	IRQ0	IRQ0: Checked. Detection type: Falling edge Digital filter: PCLK/64 Priority: Level 13

Table 5.12 Smart Configurator Settings (GPIO)

Category	Item	Setting
Smart Configurator >> Components >> I/O Ports (r_gpio_rx)		Code is generated using the settings below.
	Parameter checking	System Default

6. Obtaining the Development Environment

6.1 e^2 studio

Visit the following URL and download e^2 studio.

https://www.renesas.com/products/software-tools/tools/ide/e2studio.html

This document assumes that version 2022-01 or later of e^2 studio is used. If a version earlier than 2022-01 is used, some e^2 studio functions may not be supported. Make sure to download the latest version of e^2 studio on the website.

6.2 Compiler Package

Visit the following URL and download the RX Family C/C++ Compiler Package.

https://www.renesas.com/products/software-tools/tools/compiler-assembler/compiler-package-for-rx-family.html

7. Additional Information

7.1 Notes on Using the Evaluation Version of C/C++ Compiler Package for RX Family

The evaluation version of C/C++ Compiler Package for RX Family can only be used for a limited duration and other usage limitations apply. When the evaluation period expires, the size of linkable objects is reduced to 128 KB or less, which may cause incorrect generation of the load module.

For details, refer to the following software tool page for evaluation versions on the Renesas website:

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

7.2 RX Family DSP Library

The sample program uses the DSP library to perform DSP processing (FFT, etc.).

For detailed information and to download the DSP library, visit the RX Family DSP Library webpage on the Renesas website at the following URL:

https://www.renesas.com/software-tool/dsp-library-rx-family

8. Reference Documents

- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- e^2 studio Code Generator User’s Manual: RX API Reference (R20UT2864)
- RX Smart Configurator User’s Guide: e^2 studio (R20AN0451)
- RX140 Group User’s Manual: Hardware (R01UH0905)

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 19, 2021	—	1.00
1.01	Mar. 18, 2022	Whole the document	・ Correct typographical errors, omissions, and confusing sentences. ・ Correction of font and appearance
		6	・ Clerical error correction: Removed CMT FIT modules from Table 1.1
		7	・ Table 1.2 Version update: Integrated development environment, C compiler, Sample program ・ Clerical error correction: Corrected version of iodefine.h to 1.00A
		8	・ 2.2 Connecting Equipment Updated figure: Figure 2.2
		9	・ 2.3 Running the Sample Program and Checking Operation Improved description of e^2 studio
		9, 10	・ 2.3.2 Using e2 studio Functions to Monitor FIR Filter Operation Correct the title Simplify the explanation Renamed: Perspective "fir_demo" to "FIR_Filter" Updated figure: Figure 2.3, Figure 2.4 and Figure 2.5
		18	・ 3.4.1 Initialization Correction of description: 2. Initialization of FIR filter processing
		20	・ 3.4.5 Frequency Band Judgement Based on FIR Filter Processing Results Simplification of explanation Figure update: Figure 3.7, missing characters correction and text enlargement
		21	・ Table 3.4 Correction of description: r_dsp_fir_i16i16.c, r_dsp_fir_i32i32.c, r_dsp_fir_f32f32.c
		27	・ Table 5.1 Sample Program Memory Usage (Reference) Updated due to revision of sample program
		28	・ Table 5.3 Clerical error correction: Table note
		30	・ Resource Consumption Measurement Conditions Supplementary information on measurement conditions
		Sample Program	・ Correct the discontinuity of FIR filter output signal ・ r_dsp_fir_i16i16.c, r_dsp_fir_i32i32.c, r_dsp_fir_f32f32.c ・ Version update: Integrated development environment, C compiler, FIT modules, etc., ・ e^2 studio **FIR_Filter** perspective ・ Renamed: Changed to "FIR_Filter" ・ Correction: Display order, Expression of **Monitor Memory**

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

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/.

Trademarks

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