RX63N Group, RX631 Group

Determining Pitch Notations Using the FFT Algorithm

R01AN1265EJ0100
Rev. 1.00
Aug. 1, 2013

Abstract

This document describes the system to determine pitch notations using the DSP library in the RX63N and RX631 Groups.

The features of the system are described below.

- Determination range is between do (C4) and ti (B4).
- Determine a pitch notation for a sound or pitch notations for a chord that consists of less than or equal to three sounds including a semitone.
- Use the FFT algorithm in the DSP library to convert sound data to frequency data.
- Display a pitch notation for an input sound on an LCD.

Products

- RX63N Group 177-pin and 176-pin packages with a ROM size between 768 KB and 2 MB
- RX63N Group 145-pin and 144-pin packages with a ROM size between 768 KB and 2 MB
- RX63N Group 100-pin package with a ROM size between 768 KB and 2 MB
- RX631 Group 177-pin and 176-pin packages with a ROM size between 256 KB and 2 MB
- RX631 Group 145-pin and 144-pin packages with a ROM size between 256 KB and 2 MB
- RX631 Group 100-pin package with a ROM size between 256 KB and 2 MB

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

Contents

1. Specifications

With this system, a sound or chord in the range from do (C4) to ti (B4) is input, and a pitch notation or pitch notations for the input sound or chord is displayed on an LCD.

Processing consists of the following four operations:

- Sound input: Convert a sound input through a microphone to digital data (hereinafter referred to as sound data) by A/D conversion.
- FFT algorithm: Convert sound data to frequency data.
- Pitch notation determination: Convert the frequency data to power spectrums and determine their corresponding pitch notations.
- Displaying the result: Display the determined pitch notations on an LCD.

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the Block Diagram of Pitch Notation Determination System.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
MTU2a channel 0 (MTU0)	Generate a sampling period for a sound input
12-bit A/D converter (S12AD)	Convert a sound to sound data.
DMACA	Transfer the A/D conversion result to the buffer for the FFT algorithm.

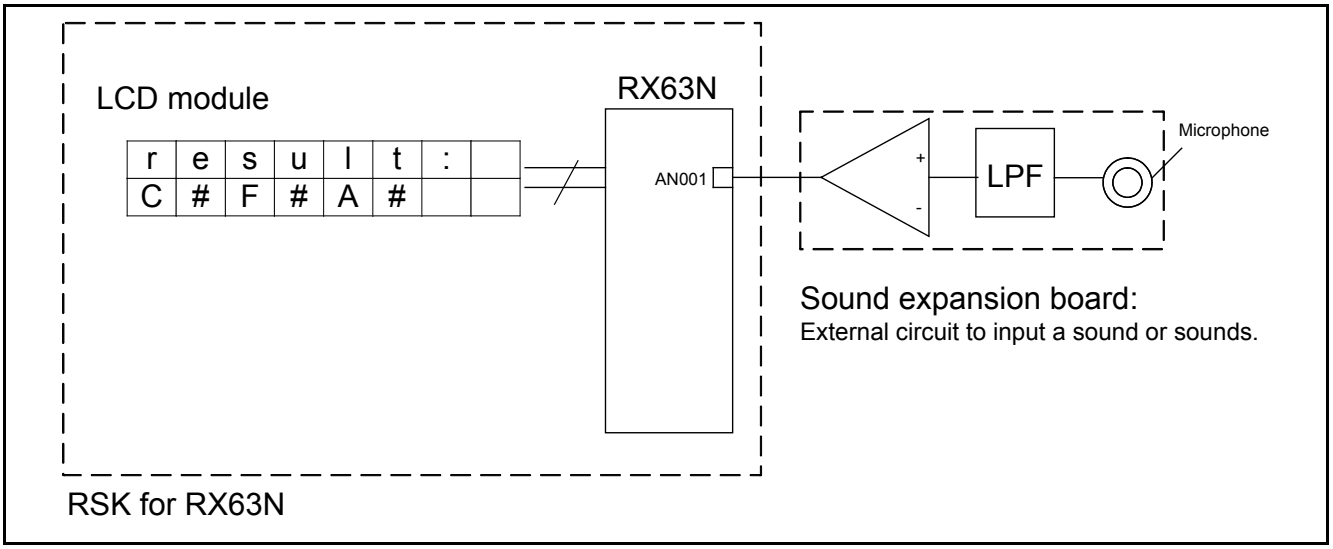

Figure 1.1 Block Diagram of Pitch Notation Determination System

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F563NBDDFC (RX63N Group)
Operating frequencies	- Main clock: 12 MHz - PLL: 192 MHz (main clock divided by 1 and multiplied by 16) - System clock (ICLK): 96 MHz (PLL divided by 2) - Peripheral module clock A (PCLKA): 96 MHz (PLL divided by 2) - Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4) - External bus clock (BCLK): 48 MHz (PLL divided by 4) - FlashIF clock (FCLK): 48 MHz (PLL divided by 4) - IEBUS clock (IECLK): 48 MHz (PLL divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01
	Compile options -cpu=rx600 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo (The default setting is used in the integrated development environment.)
iodefine.h version	Version 1.50
Endian	Little endian (1)
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit+ for RX63N (product part no.: R0K50563NC000BE)

Note:
1. The sample code in this application note does not support big endian.

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.

- RX600 Family RX DSP Library API (R01AN1244ES0100)
- RX63N Group, RX631 Group Initial Setting Rev. 1.00 (R01AN1245EJ0100)

The functions in the reference application notes are used in the sample code in this application note. The revision numbers of the reference application notes are the ones when this application note was made. However the latest version is always recommended. Visit the Renesas Electronics Corporation website to check and download the latest version.

4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows an Hardware Configuration Example including a circuit of sound expansion board.

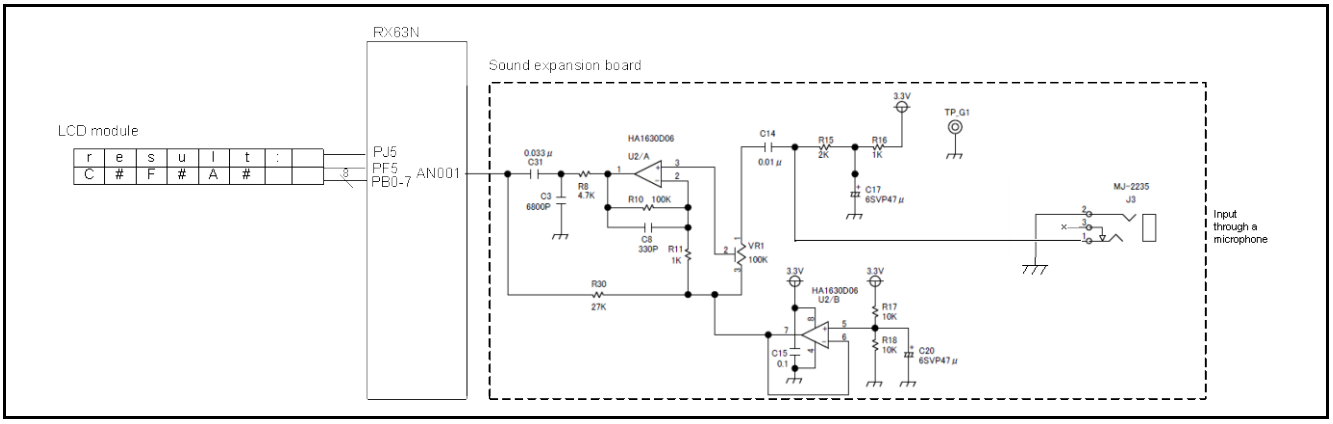

Figure 4.1 Hardware Configuration Example

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
AN001	Input	Pin to input an analog voltage through a microphone.
PF_5	Output	Display enable pin
PJ_5	Output	Debug control pin
PB_0 to PB_7	Output	Data output pin

5. Software

5.1 Operation Overview

The sample code consists of the following four operations.

(1) Sound input: Convert a sound input through a microphone to sound data by A/D conversion.

(2) FFT algorithm: Convert sound data to frequency data.

(3) Pitch notation determination: Convert the frequency data to power spectrums and determine their pitch notations.

(4) Displaying the result: Display the determined pitch notations on an LCD.

In the following sections, operations are described in the order shown in Figure 5.1.

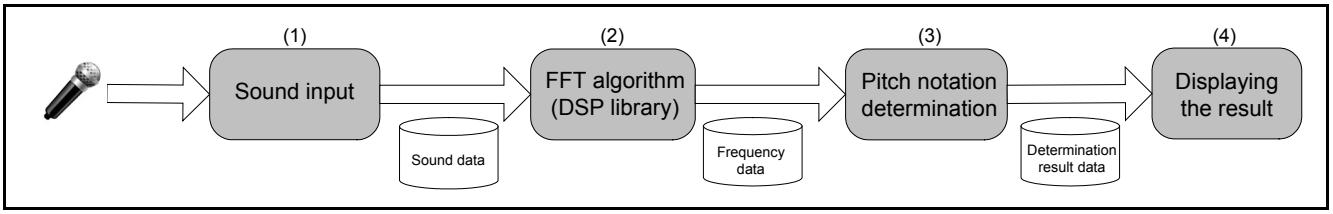

Figure 5.1 Operation Overview

5.2 Sound Input

The sound input is performed as follows:

(1) An input sound is converted to sound data by A/D conversion at certain intervals (sampling period).

(2) The A/D conversion result is stored in the buffer using DMA transfer.

5.2.1 describes Calculating Values Used for a Sound Input and 5.2.2 describes Peripheral Function Settings.

5.2.1 Calculating Values Used for a Sound Input

To input a sound, the sampling frequency, number of data, FFT reference frequency, and time required for a sound input need to be calculated. Procedures to calculate these values for the system in this application note are described below.

(1) Calculate the sampling frequency to be used as the condition based on the sampling theorem (condition A).
Refer to 1 in Column 1.
Calculation: The upper frequency limit 'f' of consecutive signals is ti that is B4 and 493 Hz.
Thus sampling frequency fs ≥ 2f = 493 × 2. The sampling frequency fs becomes approximately 1.0 kHz.

(2) Calculate the FFT reference frequency to be used as the condition based on intervals between notes (condition B).
Refer to 2 in Column 1
Calculation: The narrowest interval between notes is from do (C) to do# (C#) which is approximately 16 Hz. The FFT reference frequency is calculated by dividing 16 Hz by 3. Thus the condition becomes approximately 5 Hz.

(3) Calculate the number of data based on conditions A and B, and given calculation method. Refer to 3 in Column 1.
Calculation: Sampling frequency (1 kHz) ÷ reference frequency (5 Hz) ≤ number of data (200). Thus the number of data is 256 which is an approximate value of the Nth power of 2 (2^N).

(4) Calculate the sampling frequency to be used for calculation based on the number of data in (3) and condition B.
When the calculated result satisfies condition A, the value is determined as the sampling frequency.
Calculation: Number of data (256) × reference frequency (5 Hz) = sampling frequency (1.28 kHz)
The calculated value is valid as it satisfies condition A.

(5) Calculate the sampling period based on the sampling frequency in (4).
Calculation: 1 ÷ sampling frequency (1.28 kHz) = sampling period (0.78125 ms)

(6) Calculate time required for a sound input based on the sampling period in (5) and the number of data in (3).
Calculation: Sampling period (0.78125 ms) × number of data (256) = 0.2 seconds

The setting values used in the sample code are listed in Table 5.1.

Table 5.1 Setting Values Used in the Sample Code

Item	Setting Value
Sampling frequency	1.28 kHz
Sampling period	0.78125 ms
Number of data	256
FFT reference frequency	5 Hz
Time required for a sound input	0.2 seconds

Column 1: Basic Knowledge Required for the Sound Input

1. Condition of sampling frequency

The sampling frequency must satisfy the sampling theorem (1) to avoid noise (aliasing) that was not in the original signals.

Note:

1. Definition of the sampling theorem: To supply frequency components included in consecutive signals as correct sampling values, the sampling frequency must be more than or equal to twice the upper frequency limit of consecutive signals. That is, when the upper limit of frequency components in consecutive signals is f, the sampling frequency fs must be 'fs ≥ 2f'.

2. Condition of FFT reference frequency (spectrum interval)

The frequency of the narrowest interval between two successive notes must be considered for accurate determination.
Higher spectrums appear before and after the expected degrees than spectrums with no sound input. Therefore three spectrum intervals are required to determine adjacent notes correctly.

For example, adjacent notes which have the narrowest interval are do (C) and do# (C#), and the frequency between them is approximately 16 Hz. The number of the higher spectrum intervals appeared between do (C) and do# (C#) are three as shown in Figure 5.2. Thus 16 Hz ÷ 3 ≈ 5 Hz. The FFT reference frequency is approximately 5 Hz.

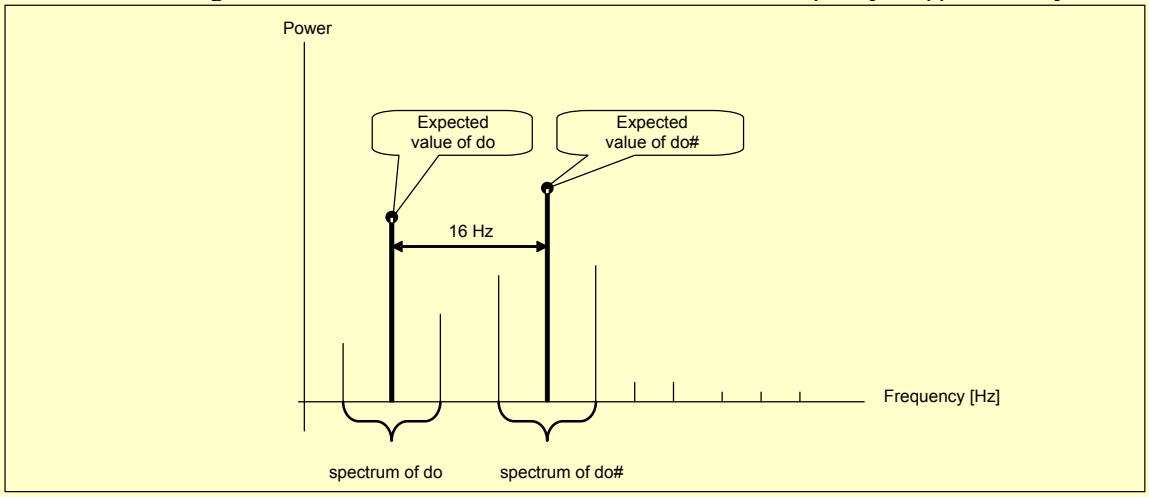

Figure 5.2 Spectrum Intervals Between Notes (Example: Between do and do#)

3. Calculation method for the number of data

The number of data indicates the number of samples for input signals and is to be the Nth power of 2 (2^N).
Calculation: Sampling frequency * ÷ FFT reference frequency * ≤ number of data (the Nth power of 2)
* Frequencies shall satisfy conditions in 1 and 2 above.

4. Sampling frequency

The sampling frequency is determined based on the number of data in 3 and the FFT reference frequency in 2 above.
Calculation: number of data × FFT reference frequency = sampling frequency

5. Others

To expand a range for determination, increase the number of data. Note that when the number of data is increased, execution time of the FFT algorithm becomes longer.

When the number of data is 256:
Determination range is up to 0.64 kHz which is half of '256 × 5 Hz = 1.28 kHz'.
When the number of data is 512:
Determination range is up to 1.28 kHz which is half of '512 × 5 Hz = 2.56 kHz'.

5.2.2 Peripheral Function Settings

A sound input is performed using values calculated in 5.2.1. An input sound is A/D converted at intervals of the sampling period, and the conversion result is transferred using the DMAC. The multi-function timer pulse unit 2 (MTU2a) is used as the A/D conversion start trigger. Table 5.2 to Table 5.4 list settings of MTU2a, 12-bit A/D converter, and DMA controller used for a sound input.

Table 5.2 Settings of MTU2a Channel 0 (MTU0)

Item	Setting
Operating mode	Normal mode
Counter clear source	TGRA compare match
Count source	Internal clock (PCLK/4)
Timer operation period	0.78125 ms

Table 5.3 Settings of 12-Bit A/D Converter (S12AD) Channel 1 (AN001)

Item	Setting
A/D conversion start trigger	A/D conversion starts by a compare match between MTU0.TGRA and TCNT (TRG0AN_0).
Operating mode	Single-cycle scan mode

Table 5.4 Settings of DMA Controller Channel 0 (DMAC0)

Item	Setting
DMAC activation source	12-bit A/D conversion scan end interrupt (S12ADI0) request
Transfer data	A/D conversion result
Transfer data length	16 bits
Transfer source address	ADDR1 register for storing the A/D conversion result
Transfer destination address	Buffer in the on-chip RAM: g_adc_buf1[], g_adc_buf2[] (1)
Transfer mode	Normal transfer mode
Number of transfers	256 times

Note:
1. Buffers g_adc_buf1[] and g_adc_buf2[] are described in 5.3.

This section describes operations of peripheral functions used for the sound input. Figure 5.3 shows correlation among major signals. Numbers (1) to (4) in the figure correspond to numbers in the following description.

(1) When the MTU0 counter value matches the TGRA register value, a compare match signal is output and the ADST bit becomes 1 (starts A/D conversion process).
When the ADST bit becomes 1, A/D conversion is started, and when the A/D conversion is completed, the ADST bit becomes 0 (stops A/D conversion process).

(2) When A/D conversion is completed, a scan end interrupt request is generated and the DMAC is activated.

(3) When DMA transfers have been performed 256 times, the DMA transfer end flag becomes 1 and the FFT algorithm is started.

(4) The transfer destination for the A/D conversion result is changed to another buffer and A/D conversion is performed continuously. DMA transfer and the FFT algorithm are performed in parallel using two buffers alternately.

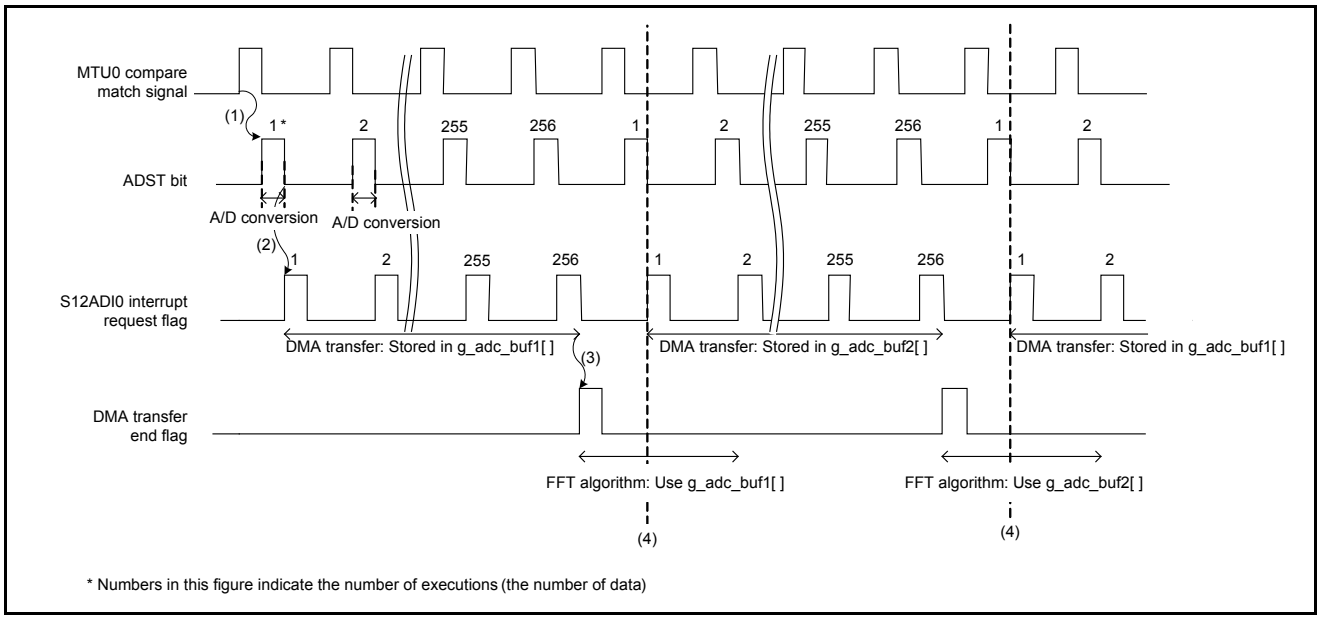

Figure 5.3 Operations of Peripheral Functions when the Sound Input is Performed

5.3 FFT Algorithm

The sample code uses the FFT algorithm in the RX600 Series DSP library (R_DSP_FFT_i16ci16).
Two buffers (array) are used alternately to store A/D conversion results and perform the FFT algorithm. Figure 5.4 shows the Data Flow of the FFT Algorithm and Figure 5.5 shows Timing/Period of Buffer Usage by the FFT Algorithm.

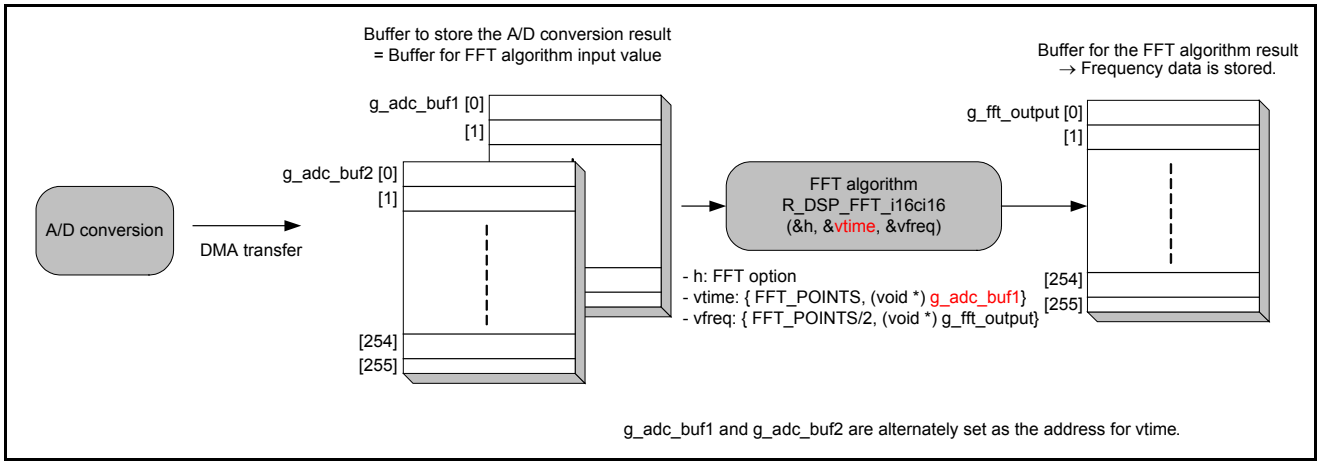

Figure 5.4 Data Flow of the FFT Algorithm

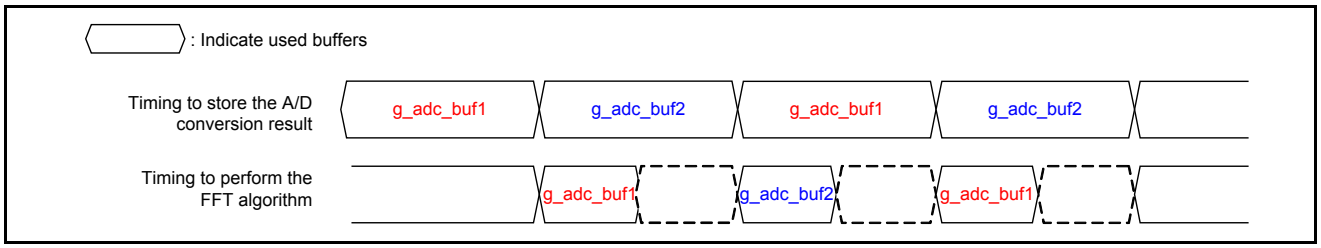

Figure 5.5 Timing/Period of Buffer Usage by the FFT Algorithm

5.4 Pitch Notation Determination

The sample code determines a pitch notation for a sound or pitch notations for a chord that consists of less than or equal to three sounds in the range from do (C4) to ti (B4). The procedures are as follows:

(1) Convert frequency data computed by the FFT algorithm to power spectrums (refer to Column 2).

(2) Extract power spectrums that meet the conditions of input sound (see below) from the converted power spectrums (output power spectrums). The extracted power spectrums are considered as valid input sounds.

(3) Sort the valid input sounds in descending order and pick up the highest three spectrums. And pitch notations will be determined for them.

(4) A pitch notation is determined based on the degree of the sound. Table 5.5 lists Relation Between Sounds and Degrees.

(5) When multiple sounds are input, a pitch notation which has lower frequency is displayed first on the LCD.

Conditions of Input Sound

A power spectrum that meets the following three conditions is considered as a valid input sound. Figure 5.6 shows Determining Pitch Notations (When Do, Re, and Fa# Are Valid Input Sounds).

- The spectrum is equal to or more than an average value of all output power spectrums.
- The spectrum is the peak point (higher than power spectrums before and after itself).
- The spectrum is not white noise (1).

Note:
1. White noise varies depending on the environment when inputting a sound. Change the value of the WHITENOISE constant accordingly.

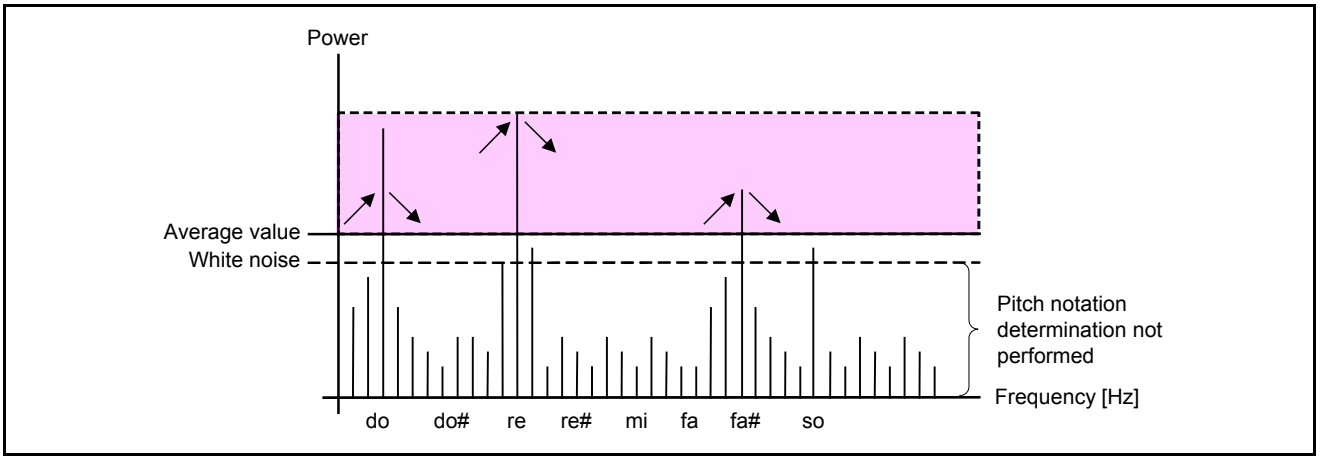

Figure 5.6 Determining Pitch Notations (When Do, Re, and Fa# Are Valid Input Sounds)

Relation Between Sounds and Degrees

Pitch notation determination is performed by relating the degree of each valid input sound to pitch notations. Table 5.5 shows Relation Between Sounds and Degrees. The value in the table is calculated with the formula 'frequency ÷ FFT reference frequency'.

Table 5.5 Relation Between Sounds and Degrees

Pitch Notation	Note	Frequency (Hz)	Value	Degree of Sound	Pitch Notation	Note	Frequency (Hz)	Value	Degree of Sound
C4	Do	261.63	52.36	50 to 54	F4#	Fa#	369.99	74.00	73 to 76
C4#	Do#	277.18	55.44	55 to 57	G4	So	392.0	78.40	77 to 80
D4	Re	293.66	58.73	58 to 60	G4#	So#	415.3	83.06	81 to 86
D4#	Re#	311.13	62.22	61 to 64	A4	La	440.0	88.00	87 to 91
E4	Mi	329.63	65.93	65 to 68	A4#	La#	466.16	93.23	92 to 96
F4	Fa	349.02	69.80	69 to 72	B4	Ti	493.83	98.77	97 to 102

Column 2: Power Spectrum (Xn)

Definitions of power spectrum in the application note are as follows:
- It is ratio of energy value within each frequency component and indicates loudness.
- It is equivalent to an amplitude spectrum squared.
- It is calculated by sum of sine wave (real part: an) squared and cosine wave (imaginary part: bn) squared of the frequency data computed by the FFT algorithm.

Xn^2 = an^2 + bn^2 (n = degree)

5.5 Displaying the Result

The following three are display patterns on an LCD.

Initial view

i	n	p	u	t	:		

Result view: When valid sounds are present, their pitch notations are displayed.

The following is an example when the valid sounds are do# (C4#), fa# (F4#), and la (A4).

r	e	s	u	l	t	:	
C	#	F	#	A	#		

Result view: When no valid sound is present, ‘silent’ is displayed.

r	e	s	u	l	t	:	
s	i	l	e	n	t		

The result is displayed according to the following specifications of the LCD display:

- When the input sound is valid, the corresponding pitch notation is displayed.
  (C to B displayed by the sample code indicate C4 to B4.)
- While the input sound is valid, i.e. the power spectrum is higher than white noise and more than or equal to an average of output power spectrums, the result keeps being displayed.
- The initial view is being displayed from the system start up to the first pitch notation determination.
- When the result for a chord is output, a pitch notation which has lower frequency is displayed first.

5.6 File Composition

Table 5.6 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 5.6 Files Used in the Sample Code

File Name	Outline	Remarks
RX_DSP_Little.lib	FFT processing	DSP library
main.c	Main processing	
r_init_stop_module.c	Stop processing for active peripheral functions after a reset	
r_init_stop_module.h	Header file for r_init_stop_module.c	
r_init_non_existent_port.c	Nonexisting port initialization	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Clock initialization	
r_init_clock.h	Header file for r_init_clock.c	
peripheral_init.c	Initialization for the MTU2a, A/D conversion and DMAC	
peripheral_init.h	Header file for peripheral_init.c	
port_init.c	Initialization for ports that are not used for peripheral functions	
port_init.h	Header file for port_init.c	
global.c	Definitions of global variables	
global.h	External reference file for macro definitions and global variables	
lcd.c	Processing for LCD display	
lcd.h	Header for lcd.c	
sound_analyze.c	Pitch notation determination	
sound_analyze.h	Header for sound_analyze.c	
windowtbl_256.h	256 data of the window function	
r_dsp_transform.h	Header file incorporated in the DSP library	
r_dsp_types.h	Common header file for the DSP library	

5.7 Option-Setting Memory

Table 5.7 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.7 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	The IWDT is stopped after a reset. The WDT is stopped after a reset.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

5.8 Constants

Table 5.8 lists the Constants Used in the Sample Code.

Table 5.8 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
FFT_POINTS	256	Number of data for sampling
WHITENOISE	1.0E + 3	Upper limit of white noise (1)
FALG_COMP	1	Flag set status
BUFF1	1	When the A/D conversion result is stored in g_adc_buf1[]
BUFF2	0	When the A/D conversion result is stored in g_adc_buf2[]

Note:
1. White noise varies depending on the environment when inputting a sound. Change the value of the WHITENOISE constant accordingly.

5.9 Structure/Union List

Figure 5.7 shows the Structure/Union Used in the Sample Code.

```
typedef struct
{
    uint16_t   n;                                   /* Number of data */
    void *   data;                                  /* Store input or output value of the FFT algorithm */
} vector_t;

vector_t vtime = {FFT_POINTS, (void *)g_adc_buf1};      /* Information of the FFT algorithm input value */
vector_t vfreq = {FFT_POINTS/2, (void *)g_fft_output};  /* Information of the FFT algorithm output value*/
```

Figure 5.7 Structure/Union Used in the Sample Code

5.10 Variables

Table 5.9 lists the Global Variables and Table 5.10 lists the static Variables.

Table 5.9 Global Variables

Type	Variable Name	Contents	Function Used
int16_t	g_adc_buf1 [256]	Buffer 1 to store the A/D conversion result that is used as the FFT input value	main, init_dmac, Excep_DMACA_DMAC0
int16_t	g_adc_buf2 [256]	Buffer 2 to store the A/D conversion result that is used as the FFT input value	main, init_dmac, Excep_DMACA_DMAC0
int16_t	g_fft_output [256]	Store the FFT output result	main, sound_analyze
int8_t	g_dmac_flag	DMA transfer end flag 0: Processing 1: Completed	main, Excep_DMACA_DMAC0
int8_t	g_buf_flag	A/D conversion result storage buffer flag 0: Store in g_adc_buf2 1: Store in g_adc_buf1	main, Excep_DMACA_DMAC0
char	g_result [8]	Buffer to store the determination result	sound_analyze, Display_LCD

Table 5.10 static Variables

Type	Variable Name	Contents	Function Used
vector_t	vtime	Input data for the FFT algorithm	R_DSP_FFT_i16ci16
vector_t	vfreq	Output data for the FFT algorithm	R_DSP_FFT_i16ci16
int32_t	status	Error check for the DSP library	main
int32_t	fft_bitrev[56]	Buffer for bit-reversed	main
int16_t	fft_twiddle[FFT_POINT +FFT_POINTS/2]	Buffer for twiddle factor	main
int16_t	r_fft_tbl_window	Window coefficient	main
uint32_t	input_cnt	Number of DMA transfer completions	Excep_DMACA_DMAC0
uint32_t	real	Real part of the frequency data squared	sound_analyze
uint32_t	image	Imaginary part of the frequency data squared	sound_analyze
uint32_t	power[128]	Buffer to store power spectrum	sound_analyze
uint32_t	sum	Sum of power spectrums	sound_analyze
float	avg	Average of power spectrums	sound_analyze
uint8_t	peak[10]	Buffer to store valid input sounds	sound_analyze
const char	clear[8]	Buffer used for clearing the result	sound_analyze
const char	silent[8]	Buffer used when no sound is displayed	sound_analyze

5.11 Functions

Table 5.11 lists the Functions Used in the Sample Code and Table 5.12 lists the API Functions for the DSP Library.

Table 5.11 Functions Used in the Sample Code

Function Name	Outline
main	Main processing
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
port_init	Port initialization
peripheral_init	Peripheral function initialization
mtu_init	MTU0 initialization
s12ad_init	S12AD initialization
dmac_init	DMAC initialization
lcd_init	LCD initialization
Display_LCD	LCD display
sound_analyze	Pitch notation determination
Excep_DMACA_DMAC0	DMA transfer end interrupt handling
error	Error processing

Table 5.12 API Functions for the DSP Library

Function Name	Outline
R_DSP_FFT_i16ci16	Real FFT algorithm
R_DSP_FFT_BufSize_i16ci16	Obtain an area size required for the FFT algorithm
R_DSP_FFT_Init_i16ci16	Initialization for the FFT algorithm

5.12 Function Specifications

The following tables list the sample code function specifications.

5.12.1 Functions Used in the Sample Code

main	
Outline	Main processing
Header	sound_analyze.h, lcd.h, r_dsp_transform.h, windowtbl_256.h, r_init_clock.h, r_init_non_existent_port.h, port_init.h, peripheral_init.h
Declaration	void main (void)
Description	Call following functions; port initialization, clock initialization, peripheral function initialization (initialization for MTU0, S12AD, and DMAC), and LCD initialization. Start the MTU0 count and enable DMA transfer.
Arguments	None
Return Value	None

R_INIT_StopModule	
Outline	Stop processing for active peripheral functions after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	Configure the setting to enter the module-stop state.
Arguments	None
Return Value	None
Remarks	Transition to the module-stop state is not performed in the sample code. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details on this function.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Initialize port direction registers for ports that do not exist in products with less than 176 pins.
Arguments	None
Return Value	None
Remarks	The number of pins in the sample code is set for the 176-pin package (PIN_SIZE=176). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details on this function.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Initialize the clock.
Arguments	None
Return Value	None
Remarks	The sample code selects processing which uses PLL as the system clock without using the sub-clock. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details on this function.

port_init	
Outline	Port initialization
Header	None
Declaration	void port_init (void)
Description	Initialize ports used for an LCD.
Arguments	None
Return Value	None

peripheral_init	
Outline	Peripheral function initialization
Header	lcd.h
Declaration	void peripheral_init (void)
Description	Initialize the following peripheral functions; MTU0, S12AD and DMAC.
Arguments	None
Return Value	None

mtu_init	
Outline	MTU0 initialization
Header	global.h
Declaration	void mtu_init (void)
Description	Initialize MTU0 and the interrupt used.
Arguments	None
Return Value	None

s12ad_init	
Outline	S12AD initialization
Header	global.h
Declaration	void s12ad_init (void)
Description	Initialize the 12-bit A/D converter, pins used, and the interrupt used.
Arguments	None
Return Value	None

dmac_init	
Outline	DMAC initialization
Header	global.h
Declaration	void dmac_init (void)
Description	Transfer 16-bit data 256 times using the ADDR1 register as the DMA transfer source and the on-chip RAM (g_adc_buf1[] or g_adc_buf2[]) as the DMA transfer destination.
Arguments	None
Return Value	None

lcd_init	
Outline	LCD initialization
Header	lcd.h, global.h
Declaration	void lcd_init (void)
Description	Initialize the LCD display to display results on the LCD.
Arguments	None
Return Value	None

Display_LCD	
Outline	LCD display (for displaying the initial view and the determination result)
Header	lcd.h, global.h
Declaration	void Display_LCD (unsigned char position, char * string)
Description	Display the second argument contents on the line specified by the first argument.
Arguments	- position: Position of the string - string: Pointer to the string to be displayed
Return Value	None

sound_analyze	
Outline	Pitch notation determination (for outputting power spectrum, extracting valid input sounds, and determining pitch notations)
Header	global.h, r_dsp_types.h
Declaration	void sound_analyze (void)
Description	With the FFT algorithm result ‘g_fft_output[ ]’ (frequency data), convert frequency components in the range from 250 Hz to 520 Hz to power spectrums by degrees and store the results in the power[]. Determine the pitch notations for the three highest power spectrums that meet the conditions of the input sound in the power[], and store the result in the result variable.
Arguments	None
Return Value	None

Excep_DMACA_DMAC0	
Outline	DMA transfer end interrupt
Header	global.h
Declaration	void Excep_DMACA_DMAC0 (void)
Description	Select the g_adc_buf1 and g_adc_buf2 alternately as the DMAC transfer destination, and enable DMA transfer.
Arguments	None
Return Value	None

error	
Outline	Error processing
Header	global.h
Declaration	void error (void)
Description	When the return value from the API in the DSP library is other than “R_DSP_STATUS_OK”, enter an infinite loop
Arguments	None
Return Value	None

5.12.2 API Functions for the DSP Library

R_DSP_FFT_Bufsize_i16ci16	
Outline	Obtain an area required for the FFT algorithm
Header	r_dsp_transform_internal.h, r_dsp_types.h
Declaration	int32_t R_DSP_FFT_Bufsize_i16ci16(r_dsp_fft_t *h, size_t * numTwiddleBytes, size_t * numBitRevBytes)
Description	Obtain sizes of both the twiddle factor array and bit-reverse table, given the length, form, and input/output data types of a Fast Fourier Transform function.
Arguments	- h: Pointer to the FFT handle - numTwiddleBytes: Pointer to the twiddle factor - numBitRevBytes: Pointer to the bit-reverse table
Return Value	Error determination R_DSP_STATUS_OK: No issues encountered R_DSP_ERR_HANDLE_NULL: Pointer to the handle is NULL. R_DSP_INVALID_INPUT_SIZE: The order of the transform is too small or too large. R_DSP_ERR_INVALID_OPTIONS: Options value specified is invalid.

R_DSP_FFT_Init_i16ci16	
Outline	Initialization for the FFT algorithm
Header	r_dsp_transform_internal.h, r_dsp_types.h
Declaration	int32_t R_DSP_FFT_Init_i16ci16(r_dsp_fft_t * handle)
Description	Initialize the data structure for the FFT algorithm.
Arguments	- handle: Instance used for FFT processing
Return Value	Error determination R_DSP_STATUS_OK: Initialization completed. R_DSP_ERR_HANDLE_NULL: Handle is NULL. R_DSP_ERR_INPUT_NULL: Twiddle table or bit-reverse table is NULL.

R_DSP_FFT_i16ci16	
Outline	Real FFT algorithm
Header	r_dsp_transform_internal.h, r_dsp_types.h
Declaration	int32_t R_DSP_FFT_i16ci16 (r_dsp_fft_t *handle, cpmst vector_h *src, vector_t *dst)
Description	Compute the FFT algorithm, given the first argument that is necessary for FFT processing and the second argument, and input the result to the third argument.
Arguments	- handle: Instance used for FFT processing - src: Pointer to the address that stores an input value - dst: Pointer to the address that stores an output value
Return Value	Error determination R_DSP_STATUS_OK: FFT algorithm successfully completed R_DSP_ERR_HANDLE_NULL: Pointer to the handle is NULL. R_DSP_ERR_INPUT_NULL: Pointer to input data is NULL. R_DSP_ERR_OUTPUT_NULL: Pointer to output data is NULL. R_DSP_INVALID_INPUT_SIZE: Input data size is invalid. R_DSP_INVALID_OUTPUT_SIZE: Output data size is invalid. R_DSP_STATUS_OVERFLOW: Data overflow.
Remarks	h.window that is included in the handle structure indicates the window function. Refer to Column 3 for details on the window function.

Column 3: Window Function

- The function is to preprocess for decreasing a margin of error in the FFT algorithm. Windowing of an input data for the FFT algorithm makes the first and last amplitude of sampling waveforms become small and brings the data close to the ideal data (exactly one cycle or integral multiple of the reference frequency).
- The calculation formula for the Hanning window used in the sample code is as follows:

$$0.5 - 0.5 \times \cos(2 \times \pi \times n \div (\text{number of data} - 1))$$

5.13 Flowcharts

5.13.1 Main Processing

Figure 5.8 shows the Main Processing.

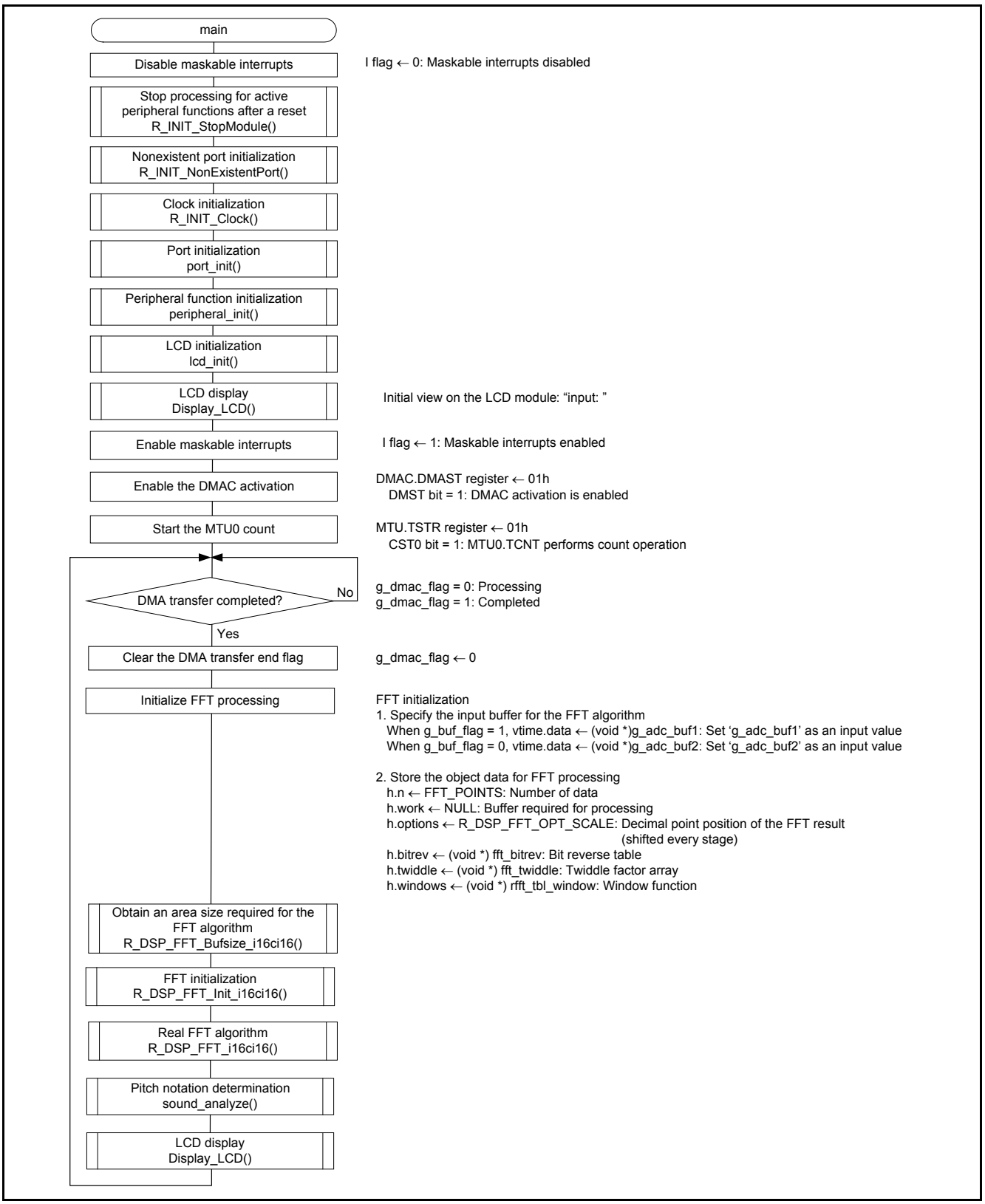

Figure 5.8 Main Processing

5.13.2 Port Initialization

Figure 5.9 shows the Port Initialization.

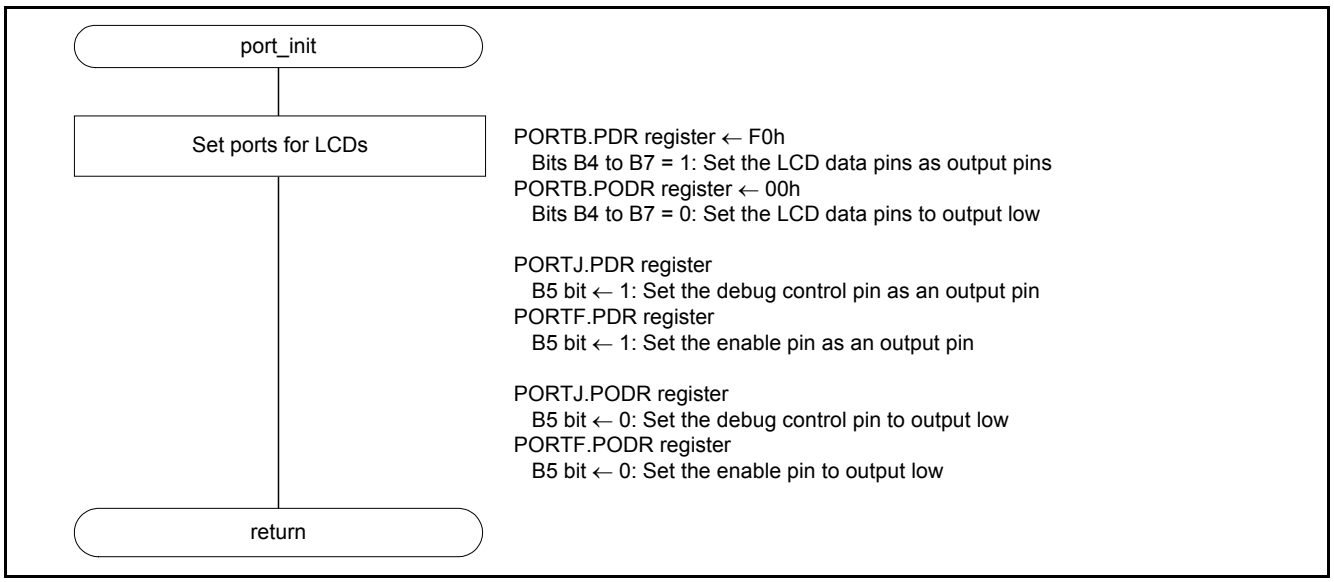

Figure 5.9 Port Initialization

5.13.3 Peripheral Function Initialization

Figure 5.10 shows the Peripheral Function Initialization.

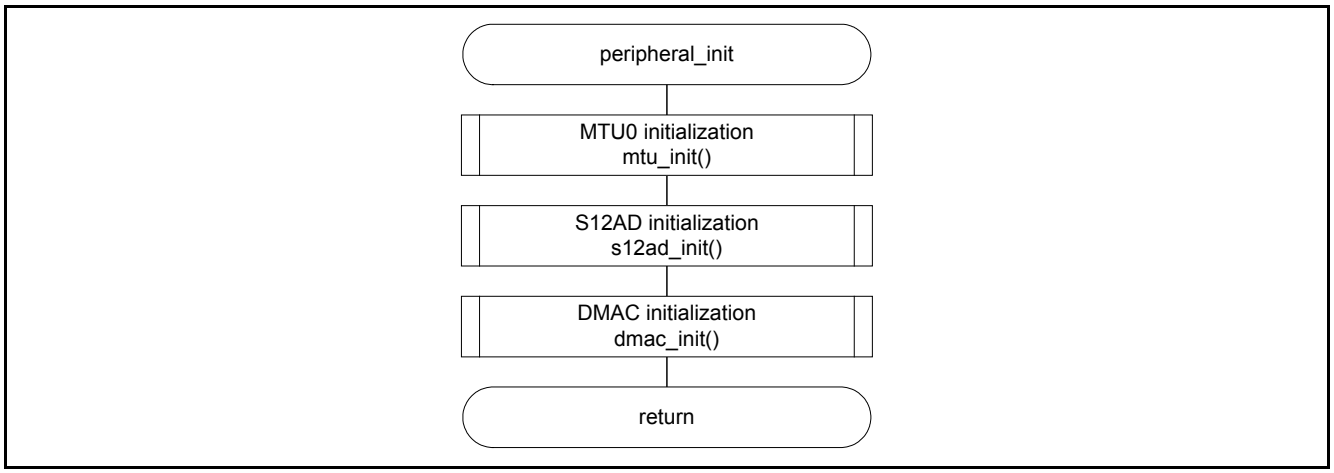

Figure 5.10 Peripheral Function Initialization

5.13.3.1 MTU0 Initialization

Figure 5.11 shows the MTU0 Initialization.

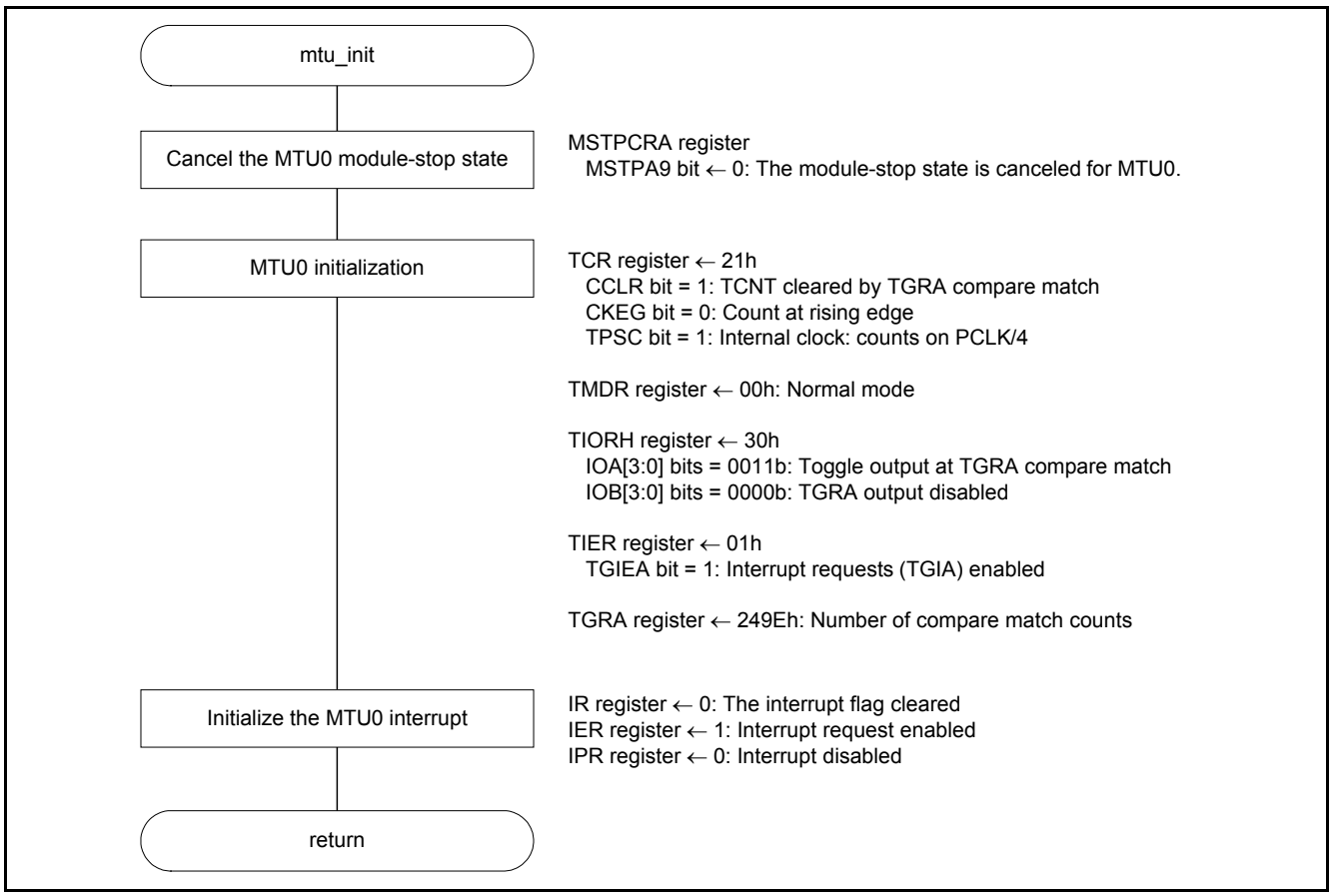

Figure 5.11 MTU0 Initialization

5.13.3.2 S12AD Initialization

Figure 5.12 shows the S12AD Initialization.

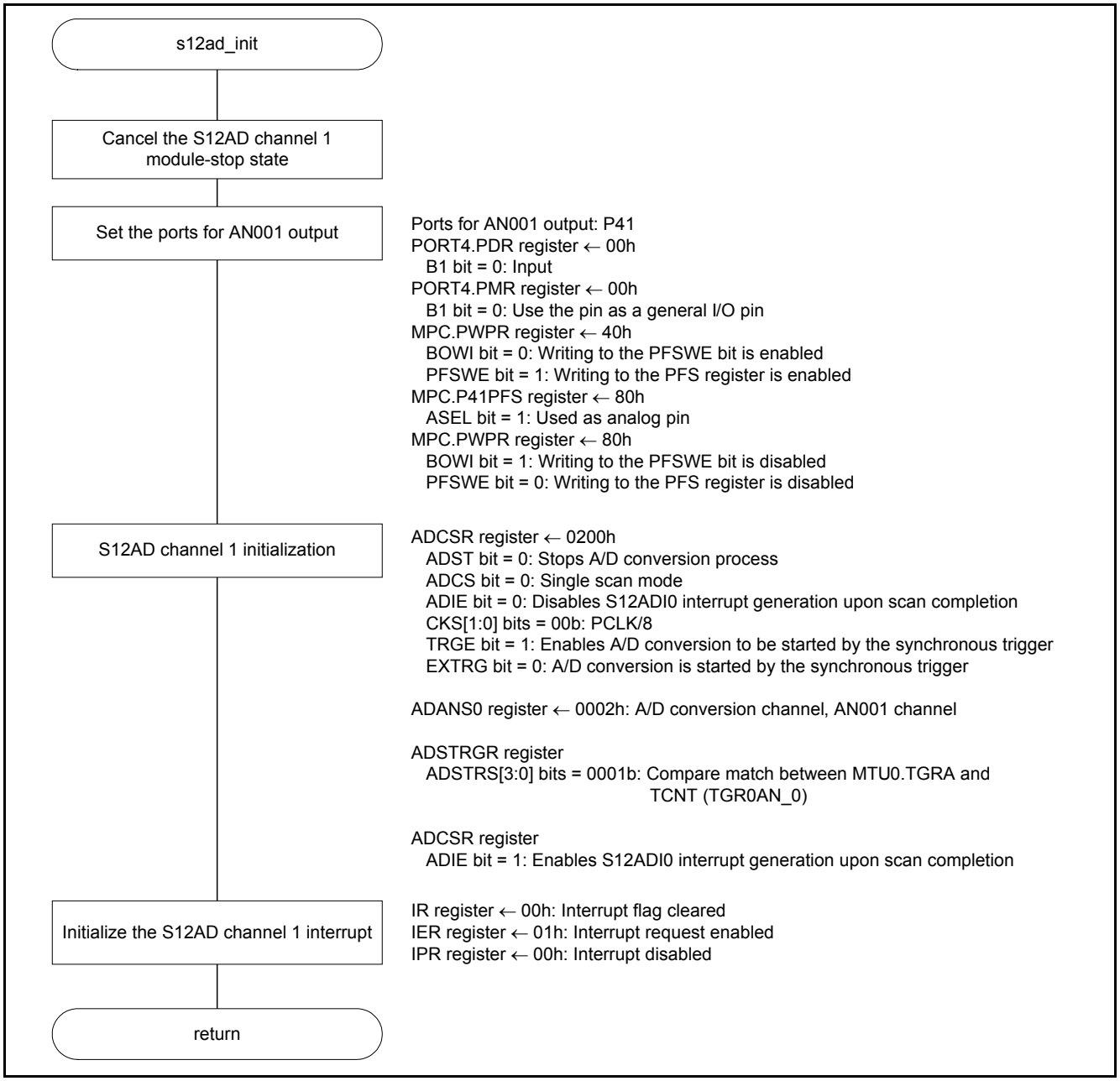

Figure 5.12 S12AD Initialization

5.13.3.3 DMAC Initialization

Figure 5.13 shows the DMAC Initialization.

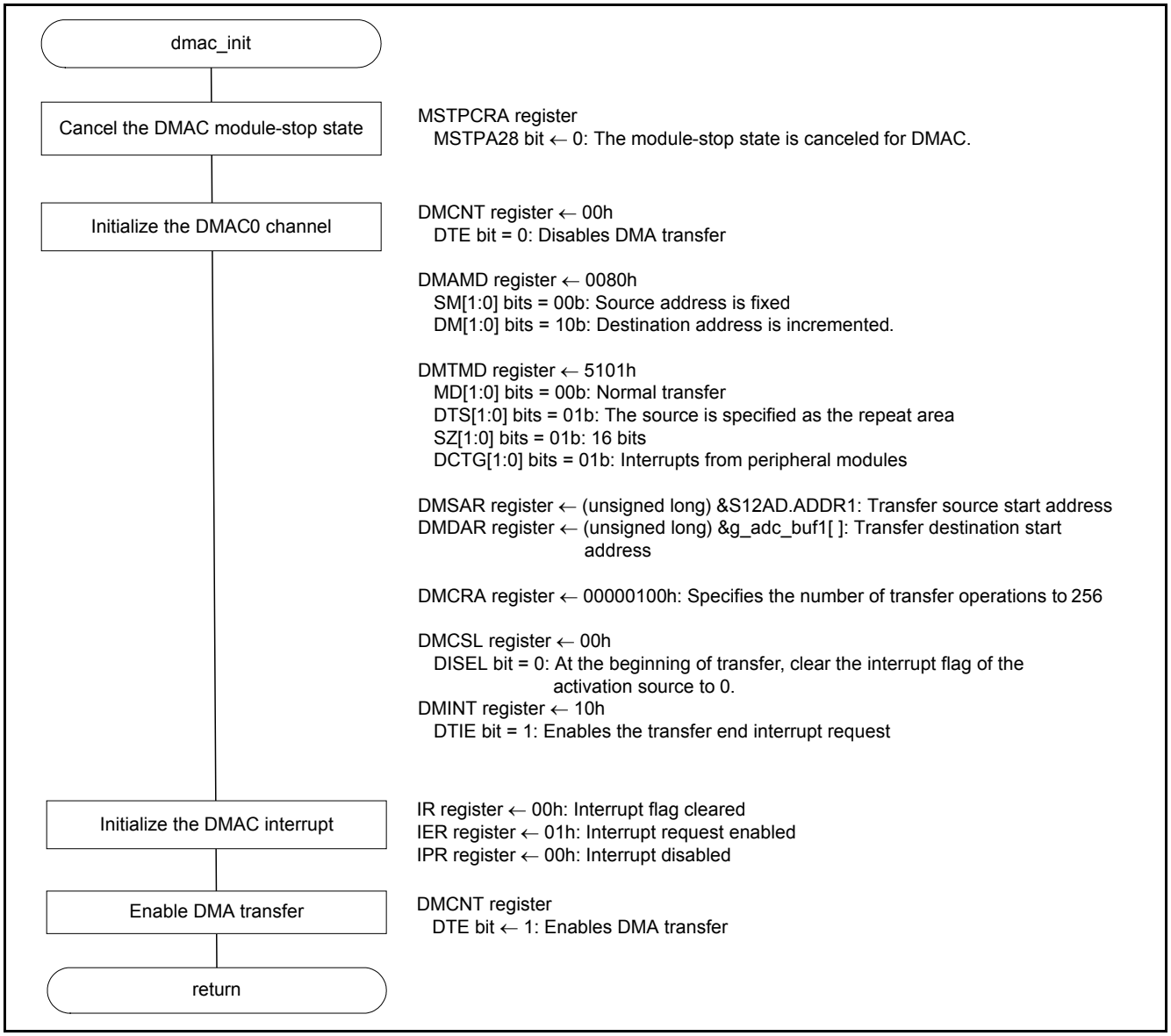

Figure 5.13 DMAC Initialization

5.13.4 DMA Transfer End Interrupt Handling

Figure 5.14 shows the DMA Transfer End Interrupt Handling.

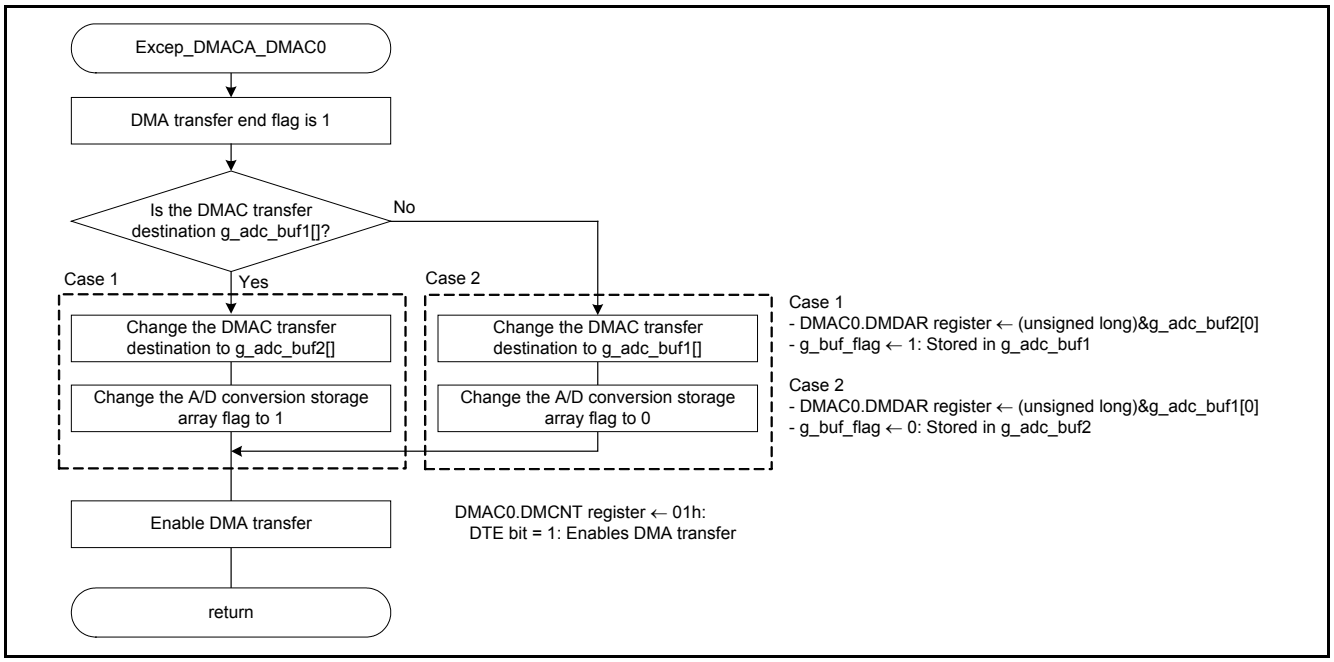

Figure 5.14 DMA Transfer End Interrupt Handling

5.13.5 Pitch Notation Determination

Figure 5.15 shows the Pitch Notation Determination.

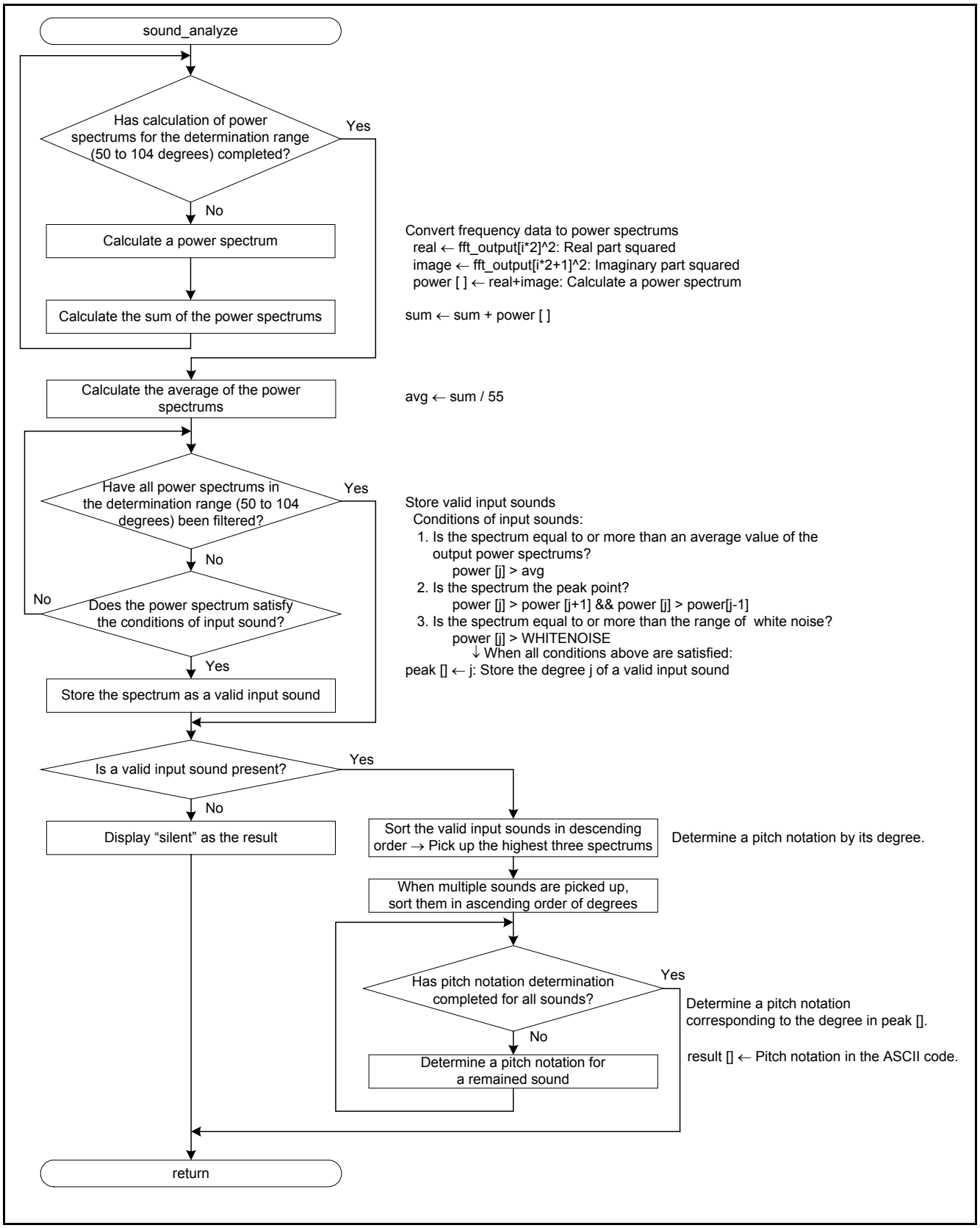

Figure 5.15 Pitch Notation Determination

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User's Manual: Hardware

RX63N Group, RX631 Group User's Manual: Hardware Rev.1.50 (R01UH0041EJ)
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

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

User's Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (R20UT0570EJ)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website

http://www.renesas.com

Inquiries

http://www.renesas.com/contact/

REVISION HISTORY	RX63N Group, RX631 Group Application Note Determining Pitch Notations Using the FFT Algorithm

Rev.	Date	Description	
		Page	Summary
1.00	Aug. 1, 2013	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. Handling of Unused Pins

   Handle unused pins in accord 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 /-7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.2