RL78/G23

Audio Playback of ADPCM Data in NOR FLASH by Using I^{2}S

Introduction

This application note describes how to play back ADPCM data stored in NOR FLASH by using the I^{2}S master function described in the RL78/G23 Application Note I^{2}S Communication with ELCL and SPI (R01AN6420). The RL78 Family Serial NOR Flash Memory Control Module Software Integration System (SIS) is used for NOR FLASH control, and the M3S-S2-Tiny middleware is used for the ADPCM encoder/decoder.

Target Device

RL78/G23

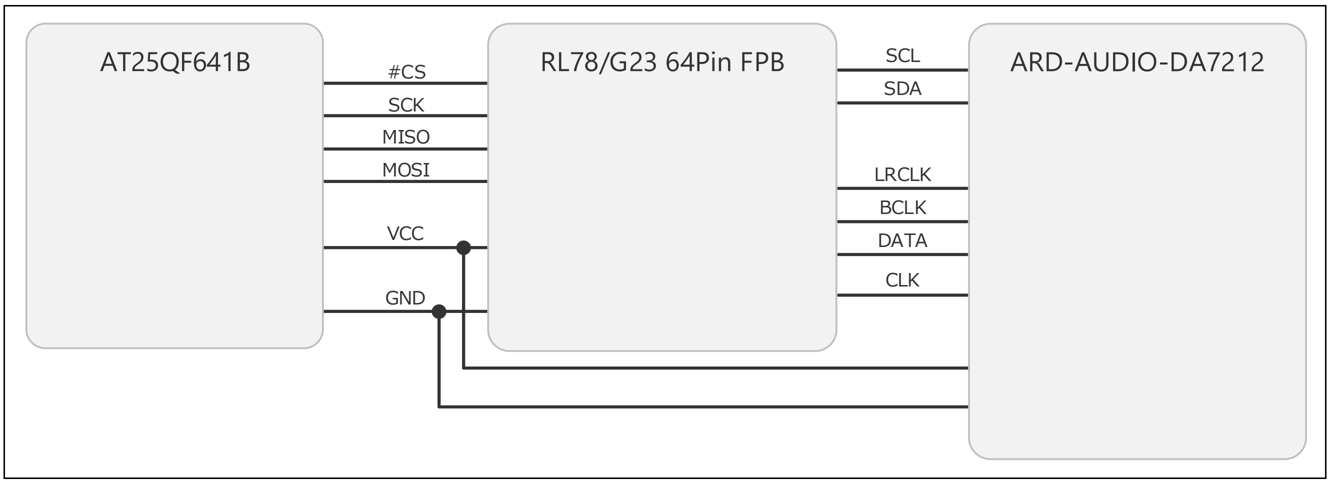
When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Specification

Figure 1-1 shows the I^{2}S communication configuration supported by this application and Table 1 1 shows the specifications.

Figure 1-1 I^{2}S Communication Configuration



Note: For pin details, see Figure 3-1

Table 1-1 I^{2}S Communication Format

Items	Contents
Sound data format	PCM
Function	Master
	BCLK frequency: 64 fs (fs: Sampling frequency)
	Sampling frequency: 8 kHz, 11.025 kHz, 16 kHz, or 22.05 kHz
	PCM data size: 16 bits

2. Conditions for Operation Confirmation Test

The sample code with this application note runs properly under the condition below.

Table 2-1 Operation Confirmation Conditions

Items	Contents
MCU	RL78/G23（R7F100GLG）
Operating frequencies	• High-speed on-chip oscillator clock: 32MHz • CPU/peripheral hardware clock: 32MHz
Operating voltage	• 3.3V • LVD0 operations (V_{LVD0}): Reset mode Rising edge TYP. 1.90V Falling edge TYP. 1.86V
Integrated development environment (CS+)	CS+ for CC V8.12.00 from Renesas Electronics Corporation
C compiler (CS+)	CC-RL V1.14 from Renesas Electronics Corporation
Integrated development environment (e2 studio)	e² studio 2024-07 (24.07.0) from Renesas Electronics Corporation
C compiler (e2 studio)	CC-RL V1.14 from Renesas Electronics Corporation
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V5.10.3 from IAR Systems
C compiler (IAR)	
Smart Configurator	V.1.11.0
Board support package (r_bsp)	V.1.62
Emulator	COM port
Board	RL78/G23-64p Fast Prototyping Board (RL78/G23-64p FPB) (RTK7RLG230CLG000BJ)
	ARD-AUDIO-DA7212 (190-03-B1-001)
	NOR FLASH AT25QF641B mounted breadboard

3. Configuration

3.1 Used Pins

Table 3-1 shows list of used pins and assigned functions.

Table 3-1 List of Pins and Functions

Pin name	Input/Output	Function
P60/SCLA0	Output	Serial clock
P61/SDAA0	Output	Serial data
P50/EO50	Output	LRCLK
P51/EO51	Output	BCLK
P73/SO01	Output	SDIN
P31/PCLBUZ0	Output	Clock output for DA7212
P16/GPIO	Output	SPI CS for NOR FLASH
P15/SCK20	Output	SPI clock for NOR FLASH
P14/SI20	Input	SPI MISO for NOR FLASH
P13/SO20	Output	SPI MOSI for NOR FLASH
P12/TXD	Output	Operating conditions display and flow control (Xon/Xoff)
P11/RXD	Input	Command and WAVE data input

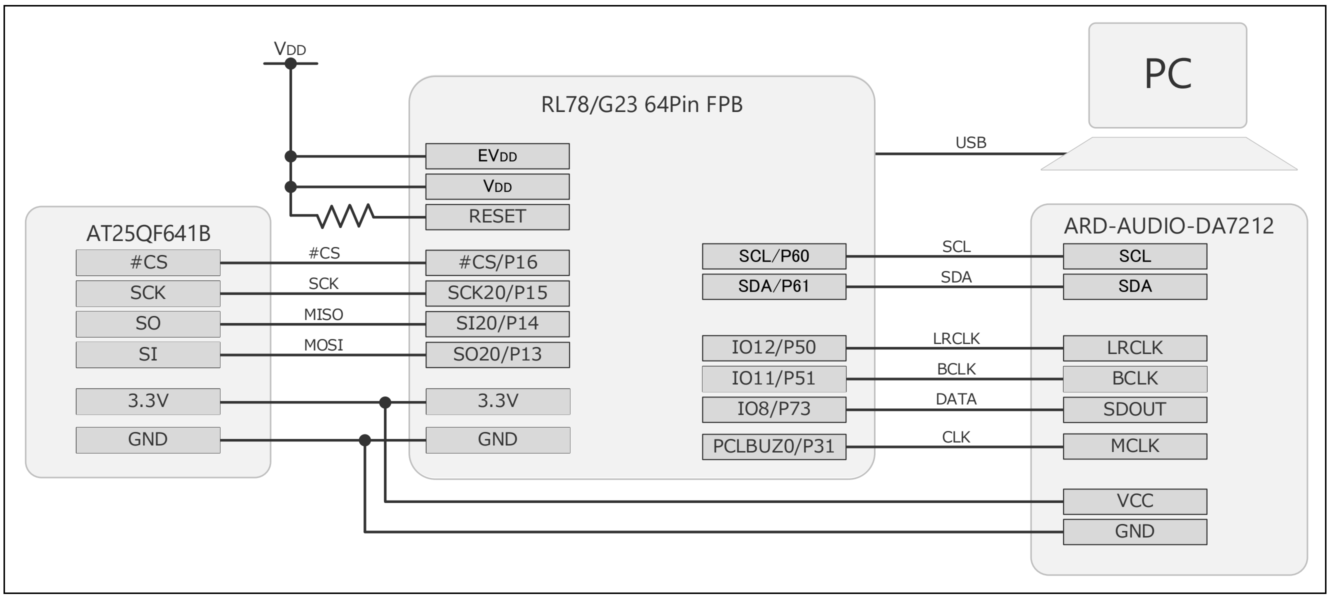
Caution. In this application note, only the used pins are processed. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

3.2 Example of Hardware Configuration

Figure 3-1 shows an example of the hardware configuration used by the sample code in this application note.

Figure 3-2 shows an example of actual connection, and Figure 3-3 and Figure 3-4 show the enlarged views of connections.

Figure 3-1 Hardware Configuration Example



Note: For I²C communication, see 5.8.1 Overview of operation.

Caution 1. This simplified circuit diagram was created to show an overview of connections only. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements. (Connect each input-only port to V_{DD} or V_{SS} through a resistor).

Caution 2. Connect any pins whose name begins with EV_{SS} to V_{SS}, and any pins whose name begins with EV_{DD} to V_{DD}.

Caution 3. V_{DD} must be held at not lower than the reset release voltage (V_{LVD0}) specified as LVD0.

Figure 3-2 Example of Actual Connection

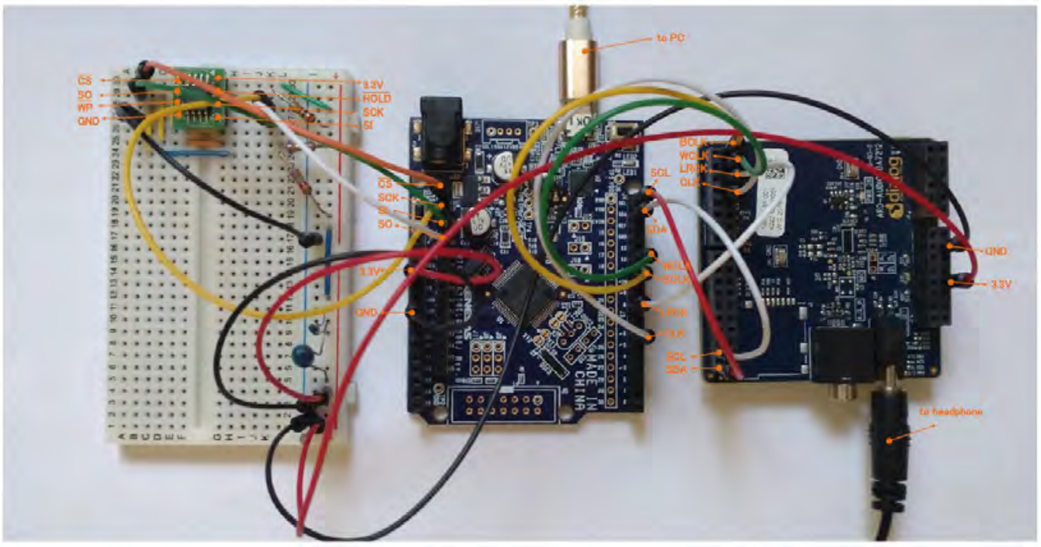


Figure 3-3 Enlarged View of Connection between RL78/G23 and ARD-AUDIO-DA7212

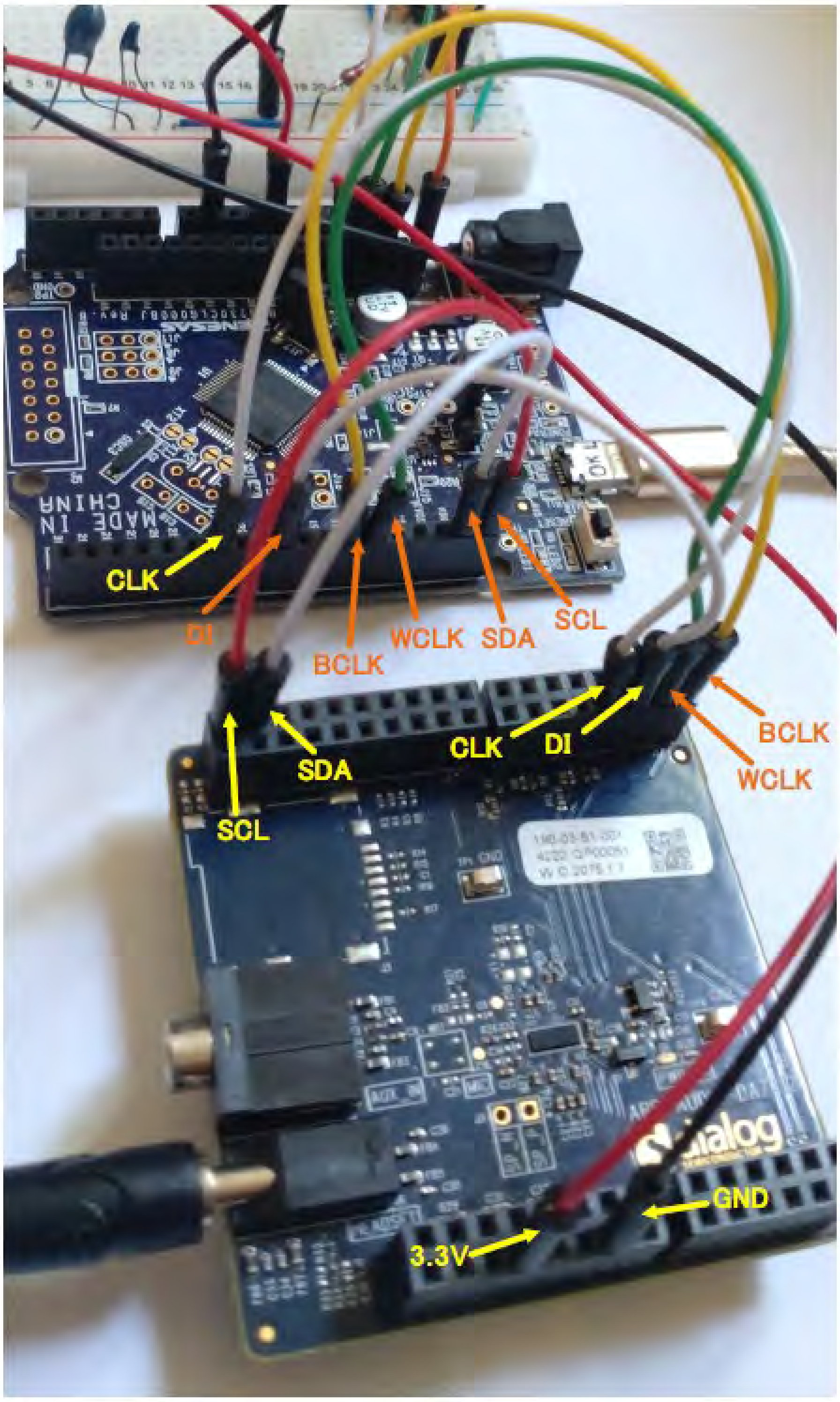
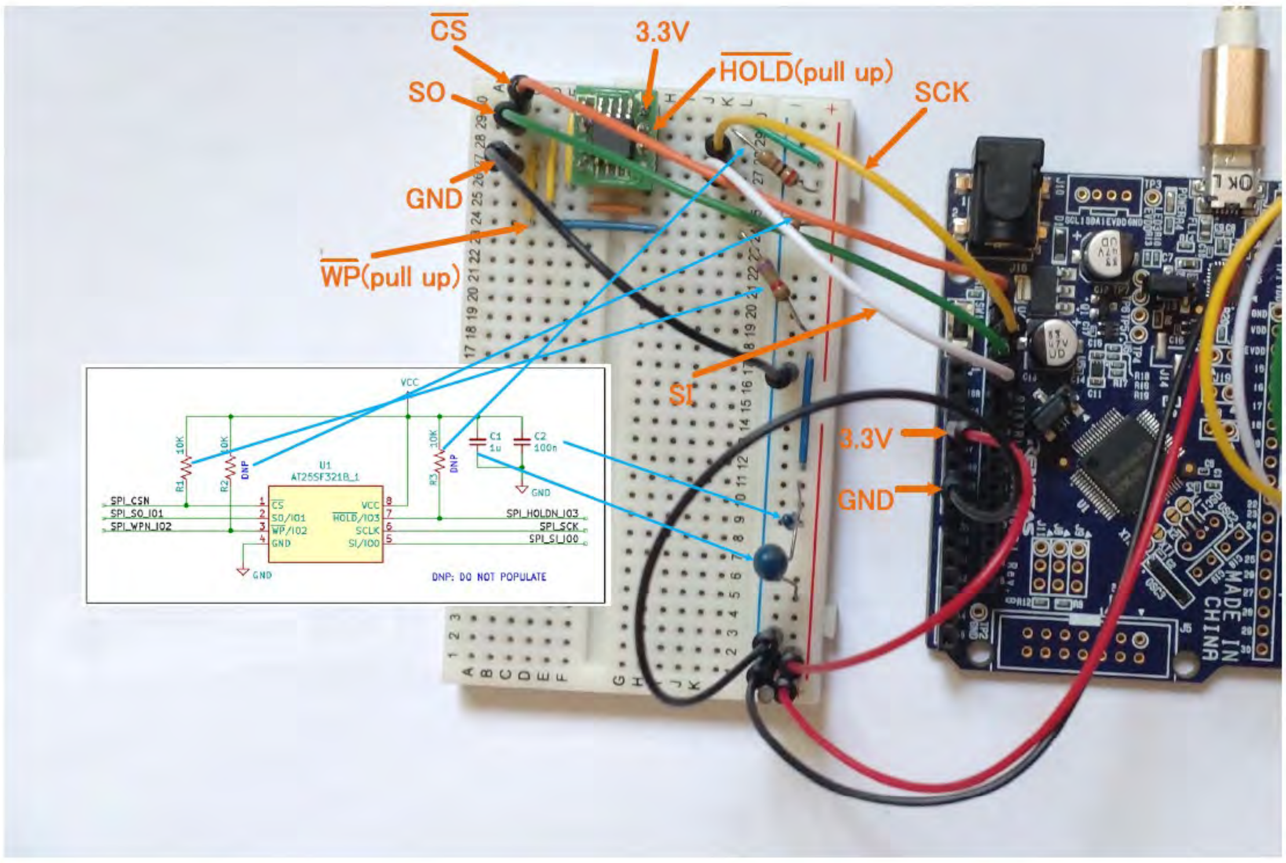


Figure 3-4 Enlarged View of Connection between RL78/G23 and NOR FLASH



Note: R2 and R3 are mounted on the preceding breadboard.

3.3 Software configuration

This section describes the software configuration.

The software used in this application note consists of two projects: ADPCM writing project (ra01an7416_adpcm_writer) and ADPCM playback project (ra01an7416_adpcm_player). The ADPCM writing project reads a WAVE (WAV) file, which has a standard audio file format used by Windows operating systems, encodes it to ADPCM data, and then writes the encoded data to NOR FLASH. The ADPCM playback project reads ADPCM data from NOR FLASH, performs ADPCM decoding, and then regenerates the decoded data through I^{2}S.

Table 3-2 Project Names and Functions

Name	Description
ra01an7416_adpcm_writer	ADPCM writing project
ra01an7416_adpcm_player	ADPCM playback project

3.4 APN configuration (File configuration)

The following shows the folder configuration of the application note.

Table 3-3 Folder configuration

Folder/File configuration	Outline
ra01an7416xx0100-rl78g23-Communication Interface<DIR>	Application note folder
├ workspace<DIR>	Workspace for storing projects
├ ├ player<DIR>	Project for audio playback
├ ├ ├ CS+<DIR>	For CS+
├ ├ ├ ├ ra01an7416_adpcm_player_cs<DIR>	Workspace for CS+
├ ├ ├ e2studio<DIR>	For e^2 studio
├ ├ ├ ├ ra01an7416_adpcm_player<DIR>	Workspace for e^2 studio
├ ├ ├ IAR<DIR>	For IAR
├ ├ ├ ├ ra01an7416_adpcm_player_iar<DIR>	Workspace for IAR
├ ├ ├ ra01an7416_adpcm_player.mot	Pre-built binary (Motorola S-record)
├ ├ writer<DIR>	Project for audio writing
├ ├ ├ CS+<DIR>	For CS+
├ ├ ├ ├ ra01an7416_adpcm_writer_cs<DIR>	Workspace for CS+
├ ├ ├ e2studio<DIR>	For e^2 studio
├ ├ ├ ├ ra01an7416_adpcm_writer<DIR>	Workspace for e^2 studio
├ ├ ├ IAR<DIR>	For IAR
├ ├ ├ ├ ra01an7416_adpcm_writerr_iar<DIR>	Workspace for IAR
├ ├ ├ ra01an7416_adpcm_writer.mot	Pre-built binary (Motorola S-record)
├ wave_files<DIR>	For audio samples
├ ├ wave_8KHz<DIR>	8 kHz audio sample
├ ├ wave_11.025KHz<DIR>	11.025 kHz audio sample
├ ├ wave_16KHz<DIR>	16 kHz audio sample
├ ├ wave_22.05KHz<DIR>	22.05 kHz audio sample
├ RA01AN7416EJ0100.pdf	This application note

4. Writing Audio Data (ADPCM) to NOR FLASH

4.1 Software Description

First, you need to write ADPCM data to NOR FLASH. To do this, use the project for the RL78/G23-64p FPB (ra01an7416_adpcm_writer) supplied with this application note. This project uses a WAVE file as input to perform ADPCM encoding with the RL78/G23-64p FPB, and then writes the data to NOR FLASH. A maximum of ten units of ADPCM data can be written.

For WAVE files, you can use the sample data supplied with this application note or use any .wav file. Note that the restrictions in Table 4-1 are applied.

Table 4-1 Restrictions on WAVE Files

Name	Restrictions
Sampling rate	8 kHz, 11.025 kHz, 16 kHz, or 22.05 kHz
Bit depth	16 bits only
Number of channels	Monaural only
Audio format	PCM only

Note: The maximum audio data length supported is 10 seconds.

4.2 Hardware Connection

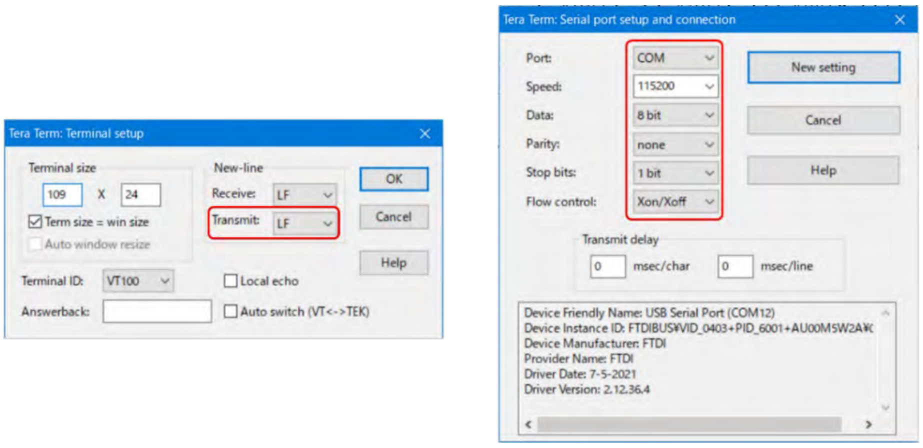
For details on hardware connection, see Figure 3-1 Hardware Configuration Example.

Note: The SPI communication rate between RL78/G23-64p FPB and NOR FLASH for ra01an7416_adpcm_writer is assumed to be 1 MHz.
This is because consideration is required for waveform rounding in connections that use a breadboard or jump wires.

4.3 Terminal software

You can operate ra01an7416_adpcm_writer by using commands through terminal software such as TeraTerm. This application note describes an example when using TeraTerm.
For details on terminal software settings, see the following (specify [Xon/Xoff] for [Flow control]).

Figure 4-1 TeraTerm Settings


Tera Term: Terminal setup
OK
Cancel
Help
Tera Term: Serial port setup and connection
New setting
Cancel
Help

Terminal size: 109 X 24
☑ Term size = win size
☐ Auto window resize
New-line
Receive: LF
Transmit: LF
Terminal ID: VT100
Answerback:
☐ Local echo
☐ Auto switch (VT<->TEK)

Port: COM
Speed: 115200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: Xon/Xoff
Transmit delay
0 msec/char 0 msec/line

Device Friendly Name: USB Serial Port (COM12)
Device Instance ID: FTDIBUS¥VID_0403+PID_6001+AU00MSW2A¥0
Device Manufacturer: FTDI
Provider Name: FTDI
Driver Date: 7-5-2021
Driver Version: 2.12.36.4

4.4 Using a Debugger

ra01an7416_adpcm_writer cannot use a debugger because USB is used for communication with terminal software. After writing the project to the RL78/G23-64p FPB by using, for example, the Renesas Flash Programmer, start it in standalone mode.

4.5 Commands

Table 4-2 lists the commands. The wr command writes ADPCM data to the location with a given number in NOR FLASH. You can specify any number in the range from 0 to 9. You can also specify a comment of up to 20 characters in the wr command. The de command deletes ADPCM data with a given number in NOR FLASH. You can specify any number in the range from 0 to 9. The li command displays information written in NOR FLASH.

Table 4-2 Commands

Command	Description
wr␣<num>␣<comment>	Writes ADPCM data to the location with the number indicated by num in NOR FLASH (write).
de␣<num>	Deletes data with the number indicated by num in NOR FLASH (del).
li	Displays information written in NOR FLASH (list).

Note: ␣ indicates a space.

4.6 Operation Methods

First, use the Renesas Flash Programmer to write ra01an7416_adpcm_writer.mot (supplied with this application note) on the RL78/G23-64p FPB.

4.6.1 Writing data to NOR FLASH

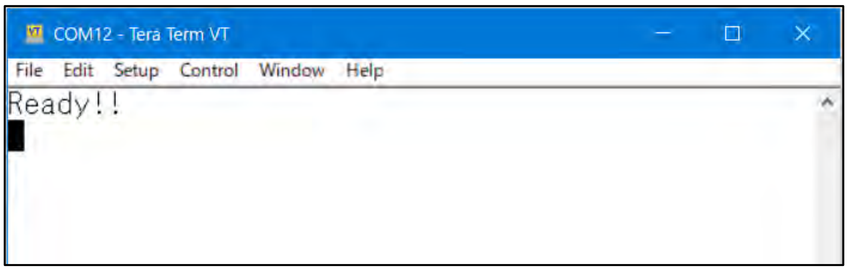
Use the wr command to write data. In the wr command, specify the write number (0 to 9) and a comment of up to 20 characters as arguments.

(1) Startup

Start the terminal software and then press the reset switch on the RL78/G23-64p FPB.

“Ready” appears on the terminal window.

Figure 4-2 Terminal Software Window: Startup Window

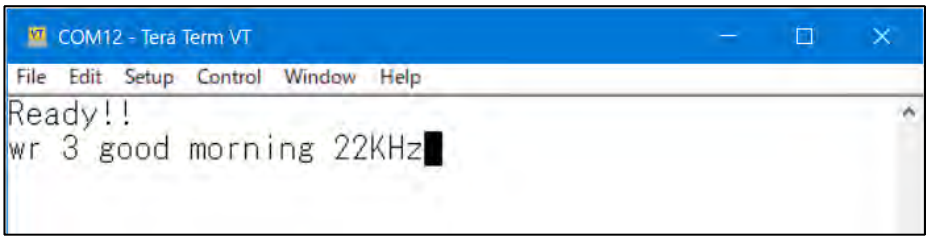


```
Ready!!
```

(2) Write command

Enter “wr” as the write command. After “wr”, specify the write number (0 to 9) and a comment of up to 20 characters.

Figure 4-3 Example of Entering the Write Command

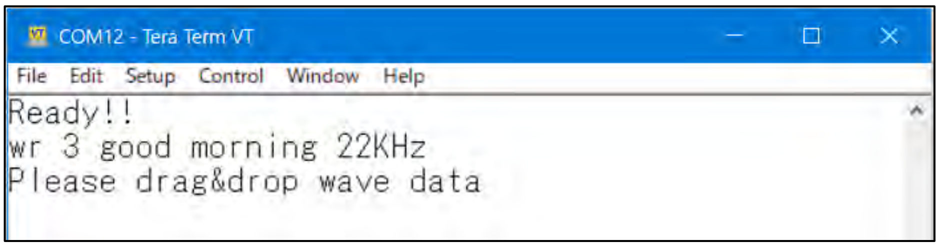


```
Ready!!
wr 3 good morning 22KHz
```

(3) WAVE data input

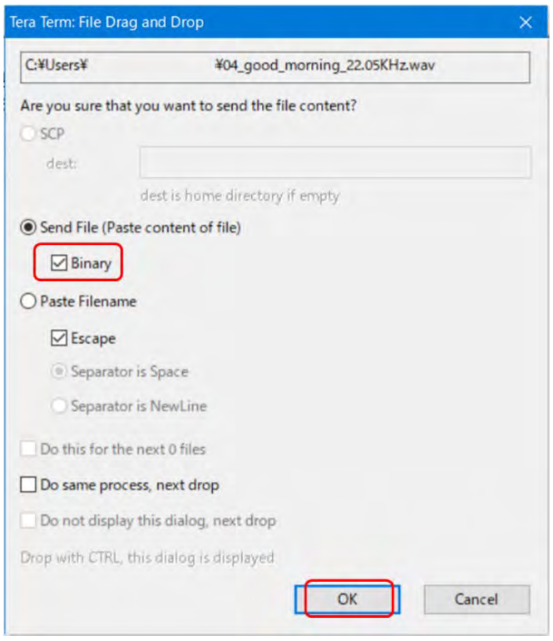
After entering the write command, press the Enter key. When “Please drag&drop wave data” appears as shown in Figure 4-4, drag and drop the WAVE file into the terminal. After dropping the WAVE file, the confirmation window appears as shown in Figure 4-5. Then, select the [Binary] check box, and then click [OK].

Figure 4-4 WAVE Input Request Window



```
Ready!!
wr 3 good morning 22KHz
Please drag&drop wave data
```

Figure 4-5 WAVE Setting Window



C:¥Users¥ ¥04_good_morning_22.05KHz.wav

Are you sure that you want to send the file content?

- ○ SCP
  - dest:
  - dest is home directory if empty
- ● Send File (Paste content of file)
  - ☑ Binary
- ○ Paste Filename
  - ☑ Escape
  - ● Separator is Space
  - ○ Separator is NewLine

☐ Do this for the next 0 files

☐ Do same process, next drop

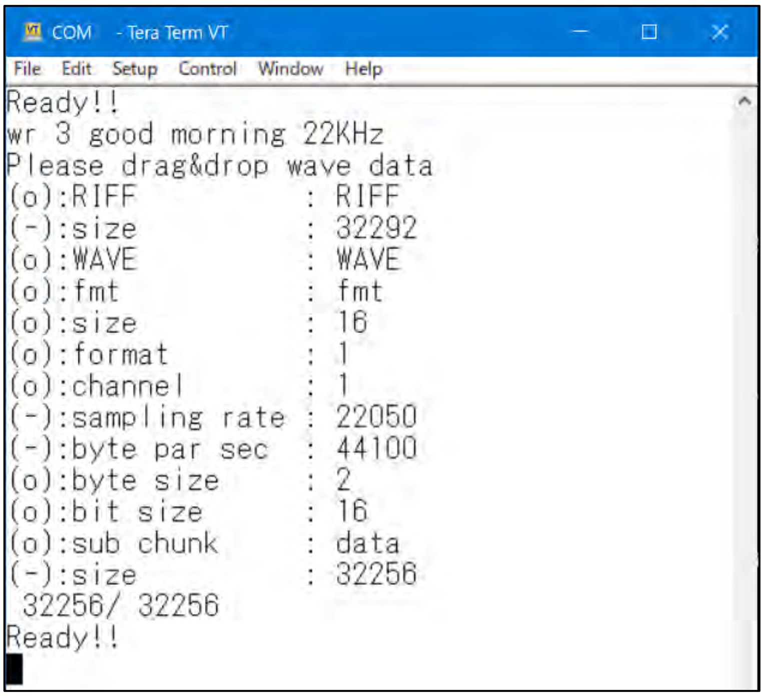
☐ Do not display this dialog, next drop

Drop with CTRL, this dialog is displayed

(4) Completion of writing

When writing is complete, WAVE information is displayed and then “Ready!!” appears again as shown in Figure 4-6.

Figure 4-6 Write Completion Window



```
Ready!!
wr 3 good morning 22KHz
Please drag&drop wave data
(o):RIFF            : RIFF
(-):size            : 32292
(o):WAVE            : WAVE
(o):fmt             : fmt
(o):size            : 16
(o):format          : 1
(o):channel         : 1
(-):sampling rate   : 22050
(-):byte par sec    : 44100
(o):byte size       : 2
(o):bit size        : 16
(o):sub chunk       : data
(-):size            : 32256
 32256/ 32256
Ready!!
```

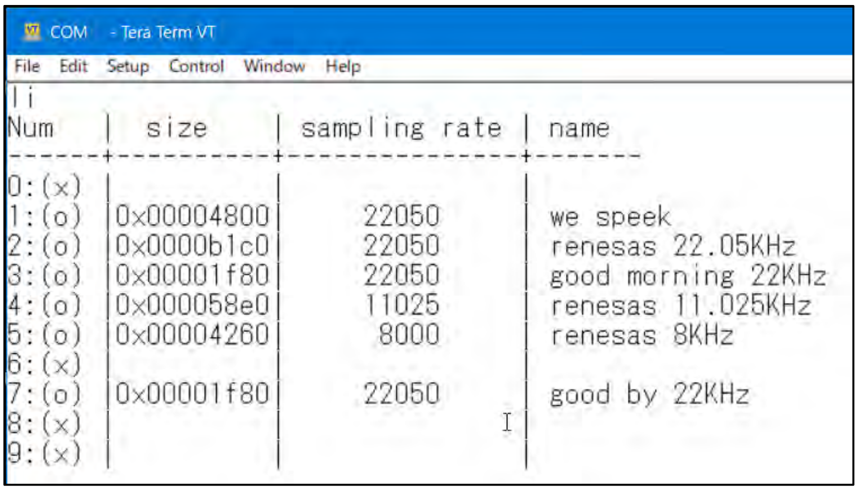
Note: 1. If a cross mark (x) is displayed beside WAVE information, it indicates the WAVE file has a format error. In this case, data cannot be written normally (no error handling is performed).

2. If data larger than the file size is sent from a terminal software such as TeraTerm to the RL78/G23 due to communication quality issues, the final Ready message will not be displayed and a deadlock will occur. In this case, reset the RL78/G23 and try writing again.

4.6.2 Displaying NOR FLASH contents

You can display the contents of NOR FLASH by using the li command.Figure 4-7 shows an example of using the li command.

Figure 4-7 Example of Using the li Command

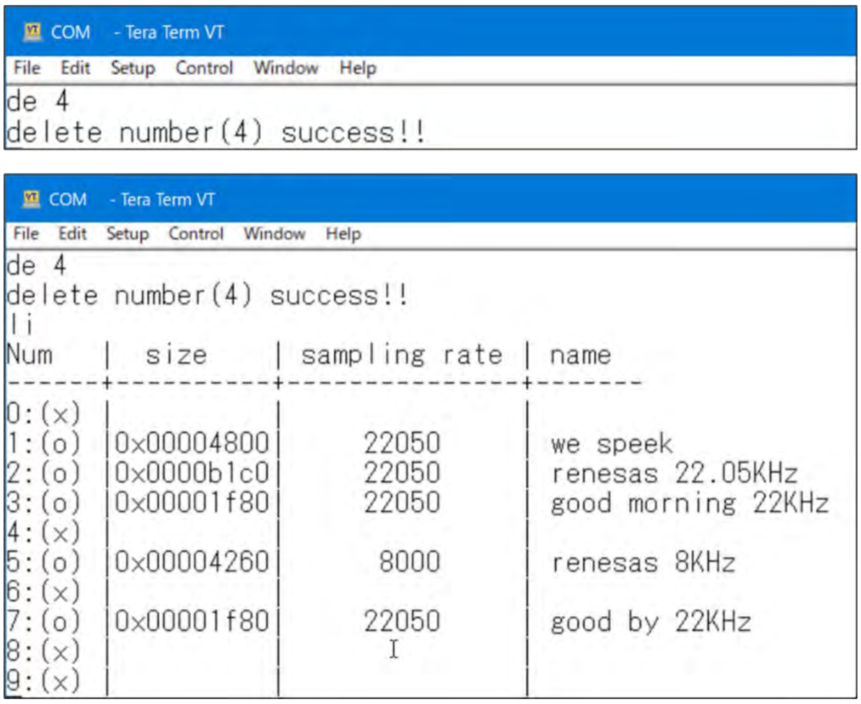


```
li
Num   |  size     | sampling rate | name
------+-----------+---------------+-------
0:(x) |           |               |
1:(o) |0x00004800|     22050      | we speek
2:(o) |0x0000b1c0|     22050      | renesas 22.05KHz
3:(o) |0x00001f80|     22050      | good morning 22KHz
4:(o) |0x000058e0|     11025      | renesas 11.025KHz
5:(o) |0x00004260|      8000      | renesas 8KHz
6:(x) |           |               |
7:(o) |0x00001f80|     22050      | good by 22KHz
8:(x) |           |               |
9:(x) |           |               |
```

4.6.3 Deleting NOR FLASH

You can delete the contents of NOR FLASH by using the de command. Figure 4-8 shows an example of using the de command. Executing the li command after the de command allows you to verify that number 4 data has been deleted.

Figure 4-8 Example of Using the de Command


COM - Tera Term VT
File Edit Setup Control Window Help
COM - Tera Term VT
File Edit Setup Control Window Help

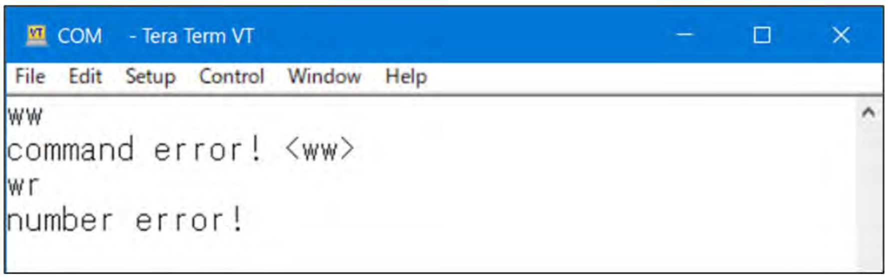
```
de 4
delete number(4) success!!
```

```
de 4
delete number(4) success!!
li
Num   |  size     | sampling rate | name
------+-----------+---------------+-------
0:(x) |           |               |
1:(o) |0x00004800|     22050      | we speek
2:(o) |0x0000b1c0|     22050      | renesas 22.05KHz
3:(o) |0x00001f80|     22050      | good morning 22KHz
4:(x) |           |               |
5:(o) |0x00004260|      8000      | renesas 8KHz
6:(x) |           |               |
7:(o) |0x00001f80|     22050      | good by 22KHz
8:(x) |           |               |
9:(x) |           |               |
```

4.7 Error Display

An error is displayed if the command cannot be executed or if no number is specified. Figure 4-9 shows an example of error display.

Figure 4-9 Error Display Example



```
ww
command error! <ww>
wr
number error!
```

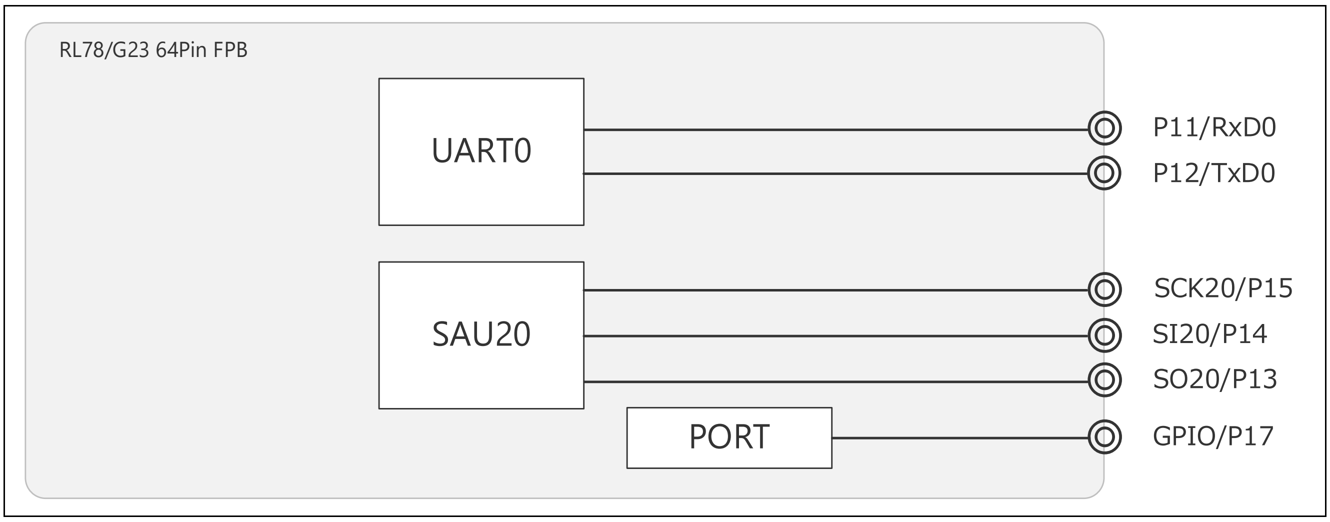
4.8 Detailed Software Description

4.8.1 Overview of operation

This sample code is a program that converts a WAVE file to ADPCM data and writes the converted data to Renesas AT25QF641B (NOR FLASH) through SPI communication (with SAU20 and PORT). You can operate this sample code through terminal software such as TeraTerm by using three commands (wr, li, and de). It cannot use a debugger because UART0 is used for terminal communication with terminal software (operating in standalone mode). This sample code only requires the configuration inFigure 4-10, but it can also be used in the same configuration as Figure 3-1.

Figure 4-10 shows the system configuration of the sample code.

Figure 4-10 System Configuration of the Sample Code



4.8.2 Folder configuration

Table 4-3 shows the configuration of source files and header files used by this sample code. Note that files that are automatically generated by an integrated development environment and those in a bsp environment are excluded.

Table 4-3 Folder configuration

Folder/File configuration	Outline	Created by Smart configurator
r01an7416_adpcm_writer<DIR>Note 1	Sample code folder	
└ src<DIR>	Program folder	
├ s2<DIR>	M3S-S2-Tiny（ADPCM encoder/decoder）	
├ smc_gen<DIR>	Folder generated by the Smart Configurator	√
├ ├ Config_CSI20	Folder for CSI20 program	√
├ ├ ├ Config_CSI20.c	Source file for CSI20	√
├ ├ ├ Config_CSI20.h	Header file for CSI20	√
├ ├ └ Config_CSI20_user.c	Interrupt source file for CSI20	√
├ ├ Config_UART0	Folder for UART0 program	√
├ ├ ├ Config_UART0.c	Source file for UART0	√
├ ├ ├ Config_UART0.h	Header file for UART0	√
├ ├ └ Config_UART0_user.c	Interrupt source file for UART0	√
├ ├ general<DIR>	Folder for initialization and common program	
├ ├ r_bsp<DIR>	Folder for BSP program	
├ ├ r_config<DIR>	Program folder	
├ ├ r_nor_flash_rl78<DIR>	Folder for storing the NOR FLASH libraryNote 2	
├ ├ r_pincfg<DIR>	Folder for storing pincfg files	
├ main.c	Main program	
├ main.h	Header file	
├ r_adpcm_writer_main.c	Main program for NOR FLASH write	
├ r_adpcm_writer_main.h	Header file	
├ r_flash_write.c	NOR FLASH write process	
├ r_flash_write.h	Header file	
├ r_wave_decoder.c	WAVE decode process	
├ r_wave_decoder.h	Header file	

Note: <DIR> means a directory.

Note 1. The sample code of the IAR versions contains ra01an7416_adpcm_writer_iar.ipcf. For the ipcf file, refer to the RL78 Smart Configurator User's Guide: IAR (R20AN0581).

Note 2. Download the RL78 Family Serial NOR Flash Memory control module Software Integration System by using the New Component icon of the Smart Configurator.

4.8.3 Option Byte Settings

Table 4-4 shows the option byte settings.

Table 4-4 Option Byte Settings

Address	Setting Value	Contents
000C0H/040C0H	1110 1111B (EFH)	Operation of Watchdog timer is stopped (counting is stopped after reset)
000C1H/040C1H	1111 1110B (FEH)	LVD0 operating mode: reset mode Detection voltage: Rising edge 1.90V Falling edge 1.86V
000C2H/040C2H	1110 1000B (E8H)	Flash operating mode: HS mode High-speed on-chip oscillator clock: 32MHz
000C3H/040C3H	1111 1011B (EBH)	On-chip debugging is disabled

4.8.4 ROM/RAM Size

Table 4-5 shows the ROM and RAM sizes used by this sample code when the CC-RL optimization level is -Olite (partial optimization). (The size varies depending on the optimization level.)

Table 4-5 ROM/RAM Size

Sample project name	ROM Size	RAM Szie
ra01an7416_adpcm_writer	20631Byte	1031Byte

4.8.5 Contents

Table 4-6 shows the constants that are used in this sample code.

Table 4-6 Constants used in the sample code

Constant Name	Setting Value	Contents	File
WAVE_DATA_BUF_SIZE	256	PCM buffer size	r_flash_writer_main.h

4.8.6 Variables

Table 4-7 shows the global variables used in this sample code.

Table 4-7 Global variables used in the sample code

Type	Variable name	Contents	Functions used in
st_flow_ctrl_t	g_flow_ctrl	Flow control structure	Config_UART0_user.c r_flash_writer_main.c r_wave_decoder.c
st_wave_data_t	g_wave_data	Structure for ADPCM	Config_UART0_user.c r_flash_writer_main.c r_wave_decoder.c
e_flow_state_t	g_flow_state	State	Config_UART0_user.c r_flash_writer_main.c
st_nor_flash_info_t	g_nor_flash_info_p	Flash write structure	r_flash_writer_main.c r_wave_decoder.c
st_nor_flash_erase_info_t	g_nor_flash_info_e	Flash erasure structure	r_flash_writer_main.c r_wave_decoder.c

4.8.7 Functions

Table 4-8 shows the functions used in the sample code. However, the unchanged functions generated by the Smart Configurator are excluded.

In addition, this application note only covers the functions that have been changed since Application Note I^2S Communication with ELCL and SPI. Refer to Application Note I^2S Communication with ELCL and SPI.

Table 4-8 Functions

	Function name	Outline	Source File
g	main	Main process	main.c
g	r_adpcm_writer_main	Main process of adpcm_writer	r_adpcm_writer_main.c
s	init	adpcm_writer initialization	r_adpcm_writer_main.c
s	state_ctrl	State control of adpcm_writer	r_adpcm_writer_main.c
s	check_command	Processing of commands received from the PC	r_adpcm_writer_main.c
s	get_flow_read_next	Returns the next state of flow_read.	r_adpcm_writer_main.c
s	get_flow_enc_adpcm	Returns the next state of flow_enc.	r_adpcm_writer_main.c
s	status_init	Status initialization	r_adpcm_writer_main.c
g	wait_send_end	Waiting for end of transmission to the PC	r_adpcm_writer_main.c
g	wait_and_print	Waiting for end of transmission to the PC and printing	r_adpcm_writer_main.c
g	wait_and_print_len	wait_and_print with the transmission character length	r_adpcm_writer_main.c
s	adpcm_encode_8byte	ADPCM encode process	r_adpcm_writer_main.c
s	write_nor_flash	NOR FLASH write process	r_adpcm_writer_main.c
s	check_last	Last ADPCM conversion and write	r_adpcm_writer_main.c
s	parse_command	Analyzing commands received from the PC	r_adpcm_writer_main.c
s	write_header_nor_flash	Writing header information to NOR FLASH	r_adpcm_writer_main.c
s	check_success_and_busy	Waiting for completion of writing to NOR FLASH	r_nor_flash_write.c
s	init	r_nor_flash_write initialization	r_nor_flash_write.c
g	r_nor_flash_data_write	Writing data to NOR FLASH	r_nor_flash_write.c
g	r_nor_flash_data_read	Reading data from NOR FLASH	r_nor_flash_write.c
g	r_wave_decoder_parse	WAVE format check	r_wave_decoder.c

Note 1. The first letter “g” indicates a global function and “s” indicates a static function.

4.8.8 Function Specifications

This part describes function specifications of the sample code.

Note that this application note only covers the functions that have been changed since Application Note I^{2}S Communication with ELCL and SPI. Refer to Application Note I^{2}S Communication with ELCL and SPI.

[Function name] main

Outline	Main process
Header	stdint.h, stdbool.h, main.h, r_adpcm_writer_main.h,
Declaration	void main (void);
Description	This is the main function. It calls r_adpcm_writer_main.
Arguments	None
Return value	None
Remarks	None

[Function name] r_adpcm_writer_main

Outline	Main process of adpcm_writer
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	void r_adpcm_writer_main (void);
Description	This is the main function of adpcm_writer.
Arguments	None
Return value	None
Remarks	None

[Function name] init

Outline	adpcm_writer initialization
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void init (void);
Description	This function initializes adpcm_writer.
Arguments	None
Return value	None
Remarks	None

[Function name] state_ctrl

Outline	State control of adpcm_writer
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void state_ctrl (void);
Description	This function controls the states of adpcm_writer.
Arguments	None
Return value	None
Remarks	None

[Function name] check_command

Outline	Processing of commands received from the PC
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void check_command (void);
Description	This function performs processing of commands received from the PC.
Arguments	None
Return value	None
Remarks	None

[Function name] get_flow_read_next

Outline	Returns the next state of flow_read.
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void get_flow_read_next (void);
Description	This function returns the next state of flow_read.
Arguments	None
Return value	None
Remarks	None

[Function name] get_flow_enc_adpcm

Outline	Returns the next state of flow_enc.
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void get_flow_enc_adpcm (void);
Description	This function returns the next state of flow_enc.
Arguments	None
Return value	None
Remarks	None

[Function name] status_init

Outline	Status initialization
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void status_init (void);
Description	This function initializes the status.
Arguments	None
Return value	None
Remarks	None

[Function name] wait_send_end

Outline	Waiting for end of transmission to the PC
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	void wait_send_end (void);
Description	This function waits for the end of transmission to the PC.
Arguments	None
Return value	None
Remarks	None

[Function name] wait_and_print

Outline	Waiting for end of transmission to the PC and printing
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	void wait_and_print (void);
Description	This function waits for the end of transmission to the PC, and then continues transmission to the PC.
Arguments	None
Return value	None
Remarks	None

[Function name] wait_and_print_len

Outline	wait_and_print with the transmission character length
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	void wait_and_print_len (void);
Description	This is the wait_and_print function with the transmission character length.
Arguments	None
Return value	None
Remarks	None

[Function name] adpcm_encode_8byte

Outline	ADPCM encode process
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void adpcm_encode_8byte (void);
Description	This function performs the ADPCM encode process (from 8 bytes to 2 bytes).
Arguments	None
Return value	None
Remarks	None

[Function name] write_nor_flash

Outline	NOR FLASH write process
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void write_nor_flash (void);
Description	This function performs processing to write to NOR FLASH.
Arguments	None
Return value	None
Remarks	None

[Function name] check_last

Outline	Last ADPCM conversion and write
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void check_last (void);
Description	This function converts less than 256 bytes of PCM data to ADPCM data and writes the converted data to NOR FLASH.
Arguments	None
Return value	None
Remarks	None

[Function name] parse_command

Outline	Analyzing commands received from the PC
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void parse_command (void);
Description	This function analyzes commands received from the PC.
Arguments	None
Return value	None
Remarks	None

[Function name] write_header_nor_flash

Outline	Writing header information to NOR FLASH
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void write_header_nor_flash (void);
Description	This function writes the presence or absence of ADPCM and the size to NOR FLASH.
Arguments	None
Return value	None
Remarks	None

[Function name] check_success_and_busy

Outline	Waiting for completion of writing to NOR FLASH
Header	stdint.h, stdbool.h, string.h, r_adpcm_writer_main.h, r_nor_flash_write.h, r_wave_decoder.h
Declaration	static void check_success_and_busy (void);
Description	This function waits for completion of writing to NOR FLASH.
Arguments	None
Return value	None
Remarks	None

[Function name] init(r_now_flash_write.c)

Outline	r_nor_flash_write initialization
Header	stdint.h, stdbool.h, string.h, r_smc_entry.h, r_adpcm_writer_main.h, r_nor_flash_write.h
Declaration	static void init (void);
Description	This function initializes r_nor_flash_write.
Arguments	None
Return value	None
Remarks	None

[Function name] r_nor_flash_data_write

Outline	Writing data to NOR FLASH
Header	stdint.h, stdbool.h, string.h, r_smc_entry.h, r_adpcm_writer_main.h, r_nor_flash_write.h
Declaration	void r_nor_flash_data_write (void);
Description	This function writes data to NOR FLASH.
Arguments	None
Return value	None
Remarks	None

[Function name] r_nor_flash_data_read

Outline	
Header	stdint.h, stdbool.h, string.h, r_smc_entry.h, r_adpcm_writer_main.h, r_nor_flash_write.h
Declaration	void r_nor_flash_data_read (void);
Description	
Arguments	None
Return value	None
Remarks	None

[Function name] r_wave_decoder_parse

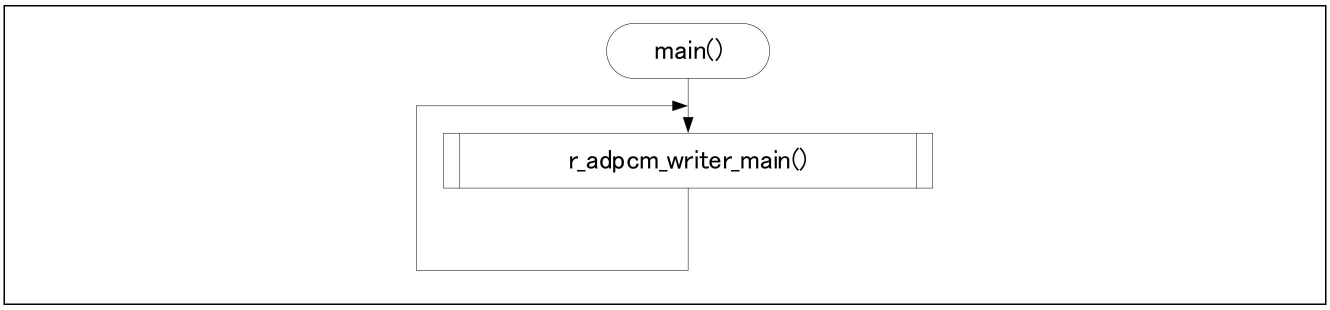
Outline	WAVE format check
Header	stdint.h, stdbool.h, r_adpcm_writer_main.h
Declaration	void r_wave_decoder_parse (void);
Description	This function checks (analyzes) the WAVE format.
Arguments	None
Return value	None
Remarks	None

4.8.9 Flow Charts

4.8.9.1 main()

Figure 4-11 shows the flowchart of main().

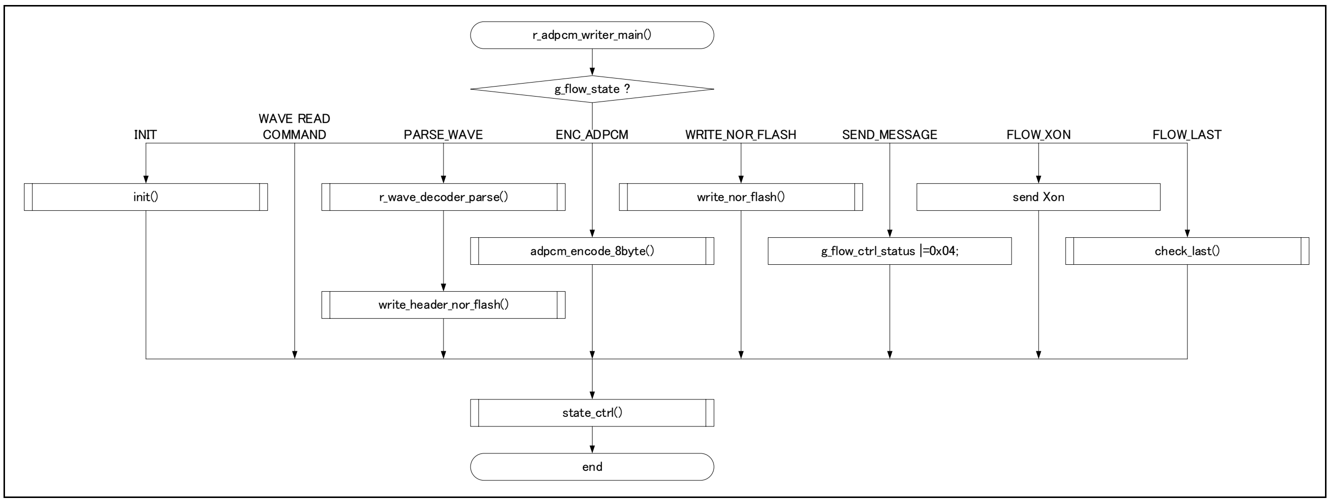
Figure 4-11 main()



4.8.9.2 r_adpcm_writer_main()

Figure 4-12 the flowchart of r_adpcm_writer_main().

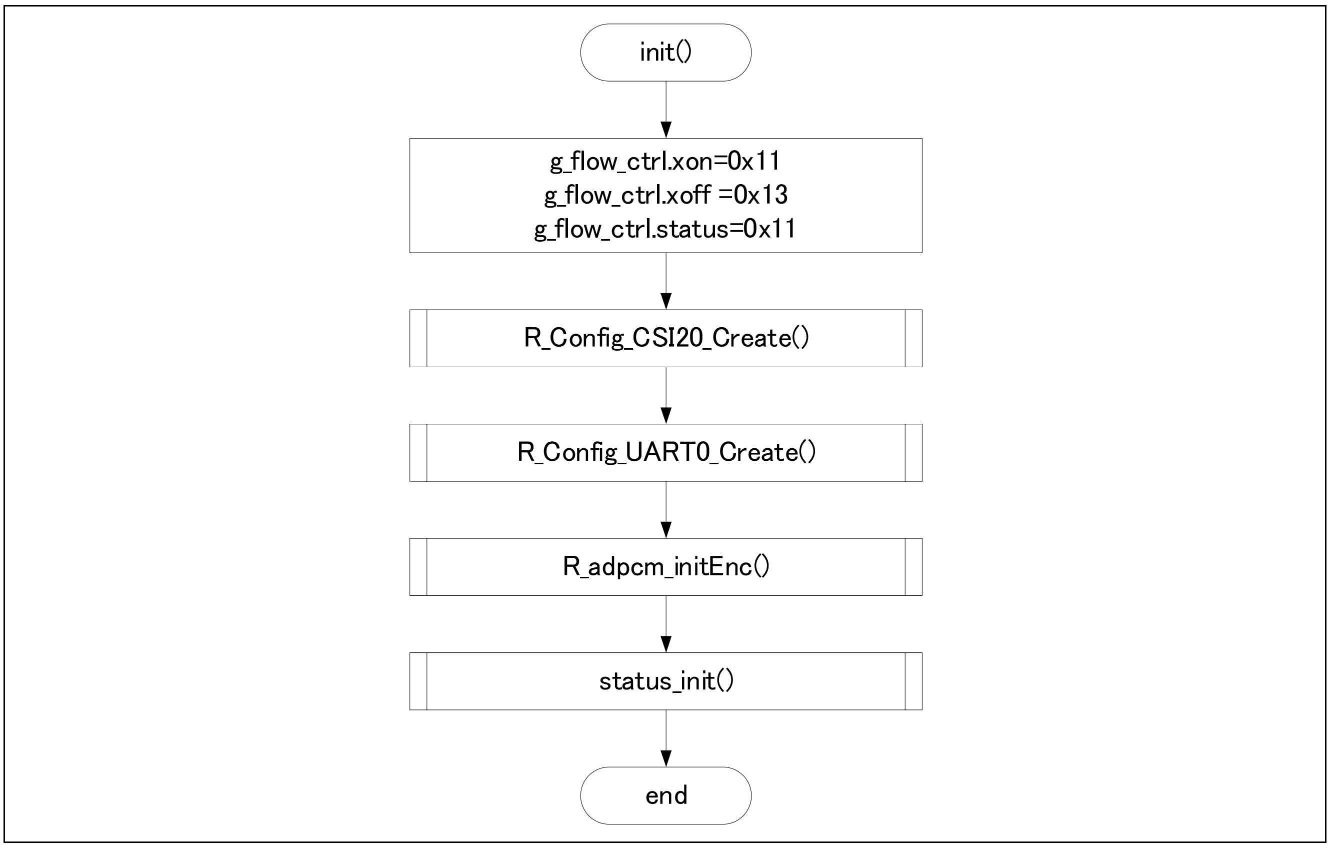
Figure 4-12 r_adpcm_writer_main()



4.8.9.3 init()

Figure 4-13 shows the flowchart of init ().

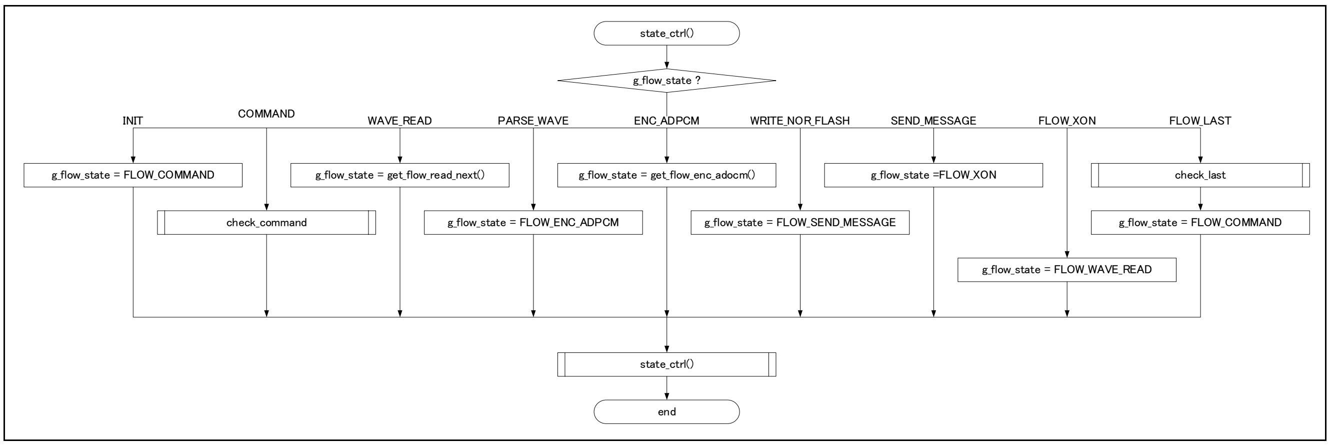
Figure 4-13 init()



4.8.9.4 state_ctrl()

Figure 4-14 shows the flowchart of state_ctrl().

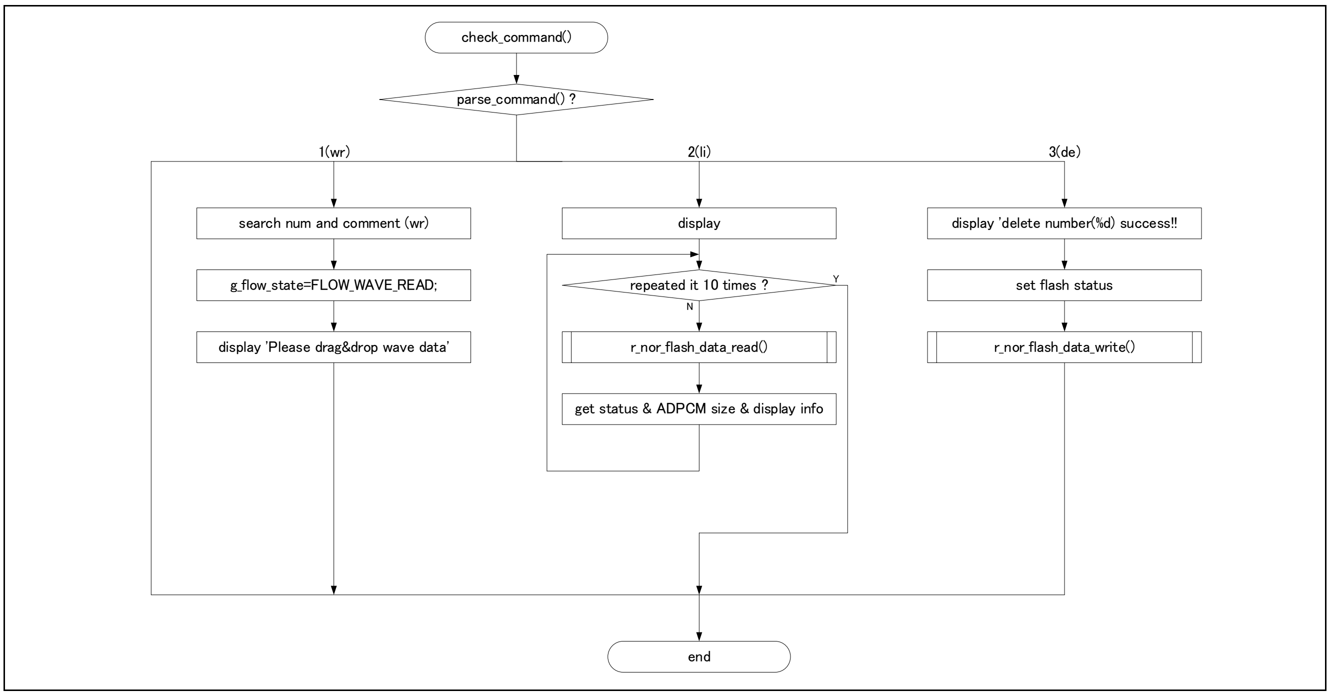
Figure 4-14 state_ctrl()


state_ctrl()
g_flow_state ?
INIT
COMMAND
WAVE_READ
PARSE_WAVE
ENC_ADPCM
WRITE_NOR_FLASH
SEND_MESSAGE
FLOW_XON
FLOW_LAST
g_flow_state = FLOW_COMMAND
check_command
g_flow_state = get_flow_read_next()
g_flow_state = FLOW_ENC_ADPCM
g_flow_state = get_flow_enc_adocm()
g_flow_state = FLOW_SEND_MESSAGE
g_flow_state =FLOW_XON
g_flow_state = FLOW_WAVE_READ
check_last
g_flow_state = FLOW_COMMAND
state_ctrl()
end

4.8.9.5 check_command()

Figure 4-15 shows the flowchart of check_command().

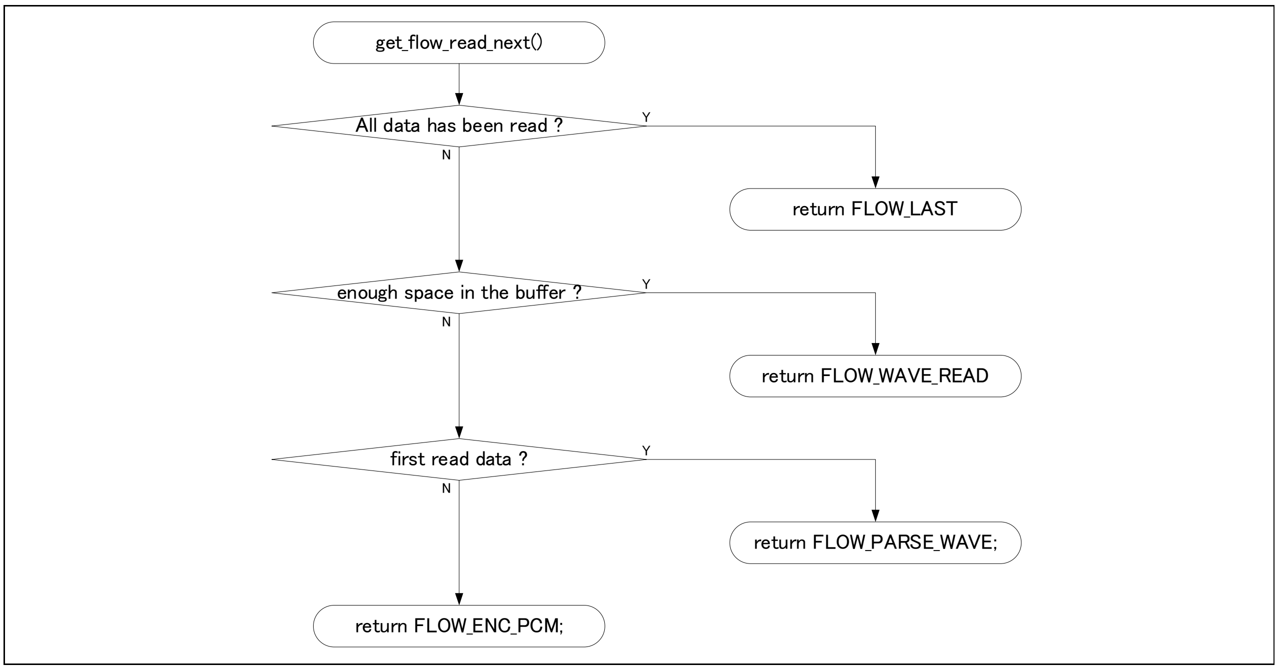
Figure 4-15 check_command()


check_command()
parse_command() ?
1(wr)
2(li)
3(de)
search num and comment (wr)
g_flow_state=FLOW_WAVE_READ;
display 'Please drag&drop wave data'
display
repeated it 10 times ?
Y
N
r_nor_flash_data_read()
get status & ADPCM size & display info
display 'delete number(%d) success!!
set flash status
r_nor_flash_data_write()
end

4.8.9.6 get_flow_read_next ()

Figure 4-16 shows the flowchart of get_flow_read_next().

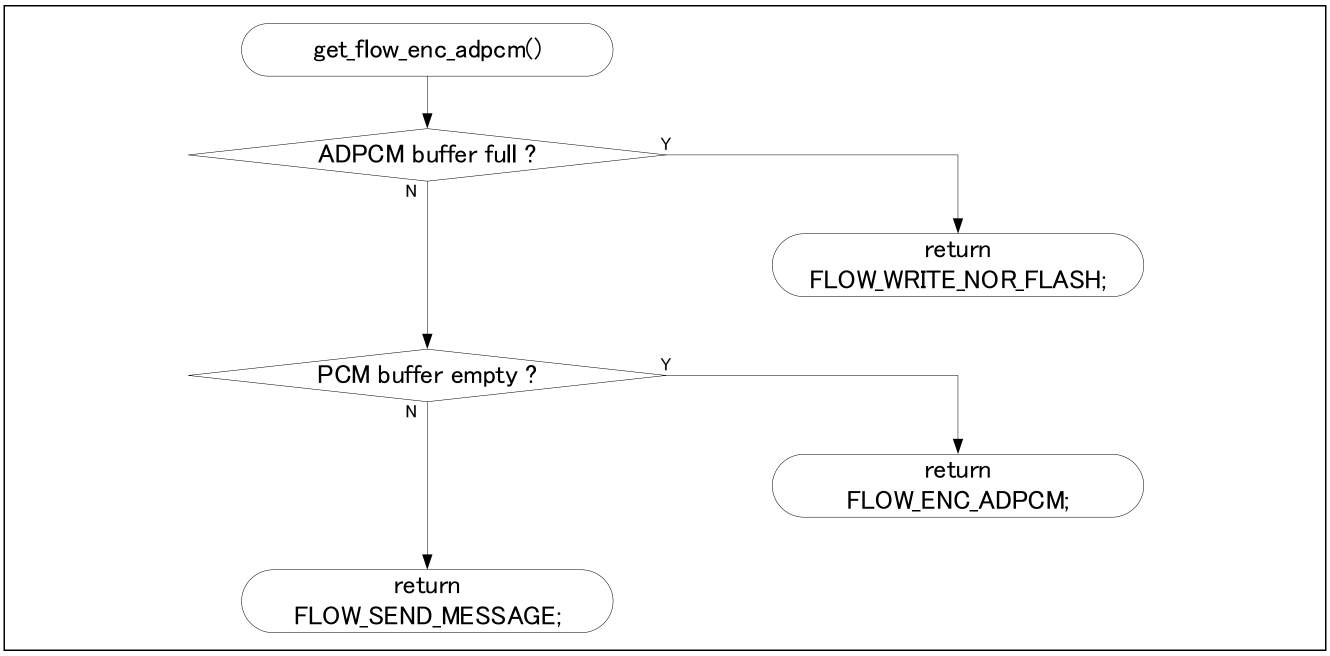
Figure 4-16 get_flow_read_next()



4.8.9.7 get_flow_enc_adpcm ()

Figure 4-17 shows the flowchart of get_flow_enc_adpcm().

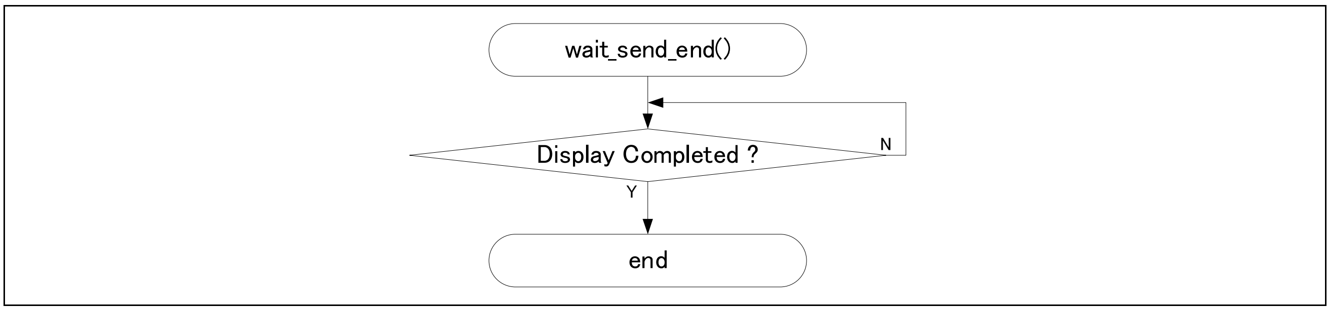
Figure 4-17 get_flow_enc_adpcm()



4.8.9.8 wait_send_end()

Figure 4-18 shows the flowchart of wait_send_end ().

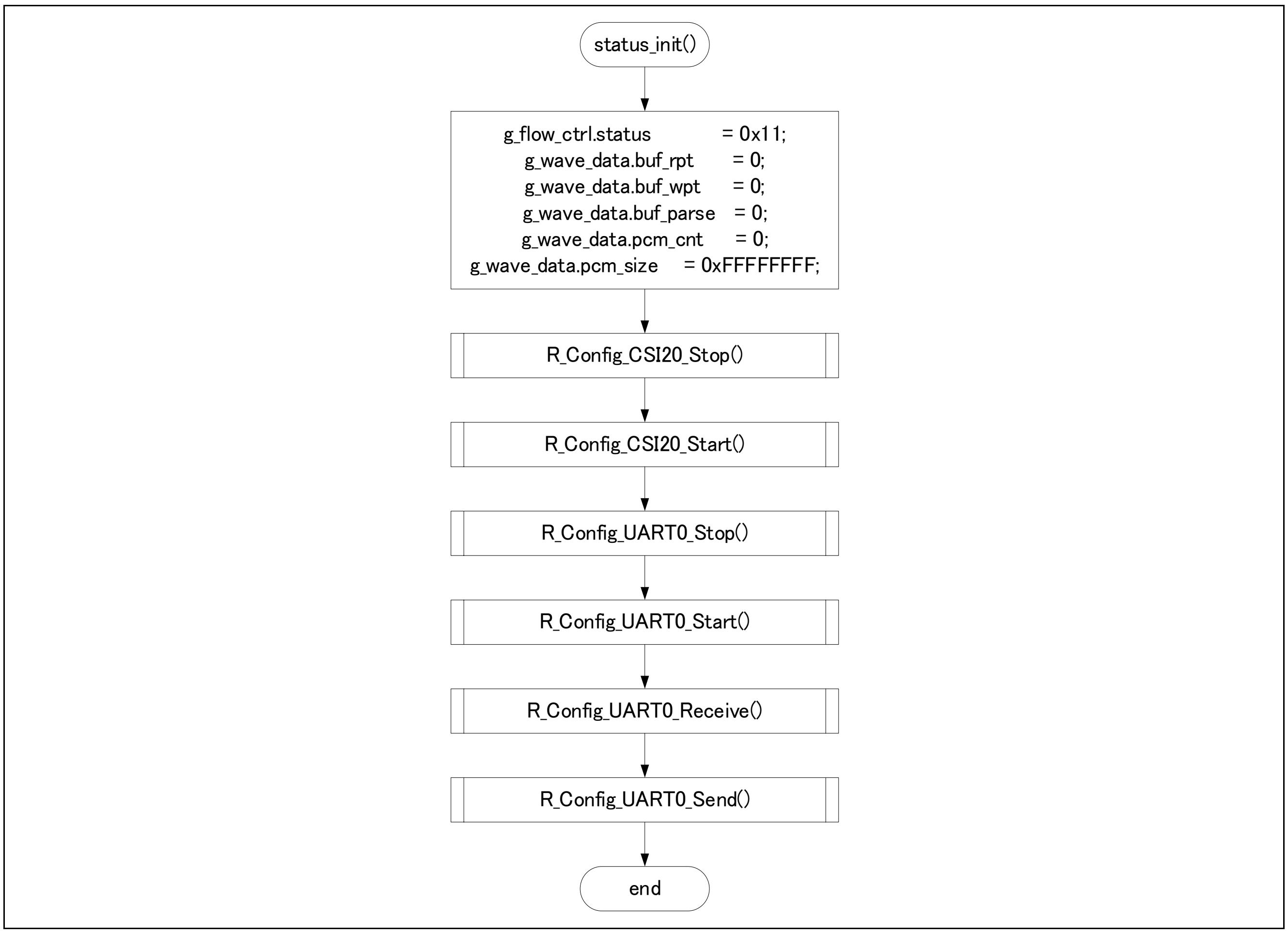
Figure 4-18 wait_send_end ()



4.8.9.9 status_init()

Figure 4-19 shows the flowchart of status_init().

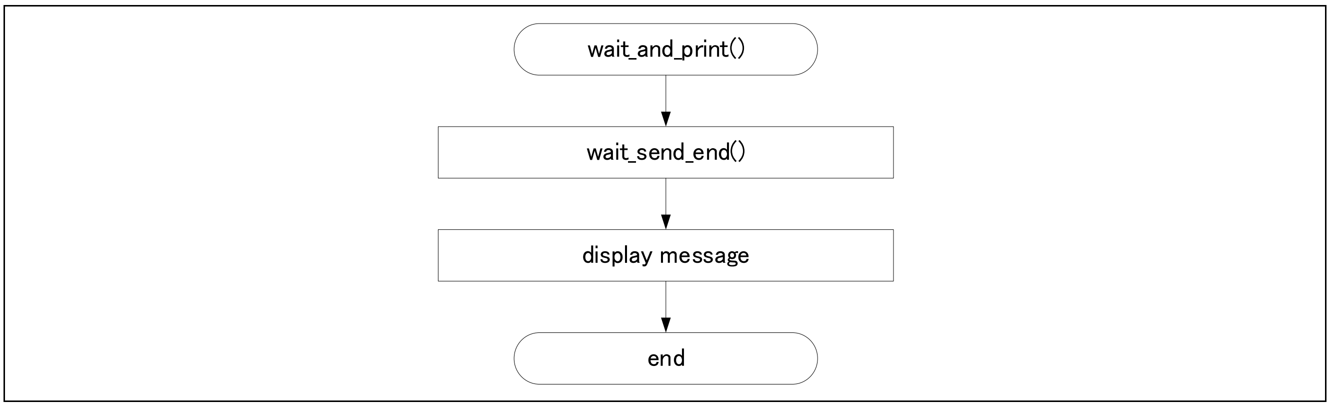
Figure 4-19 status_init()



4.8.9.10 wait_and_print ()

Figure 4-20 shows the flowchart of wait_and_print ().

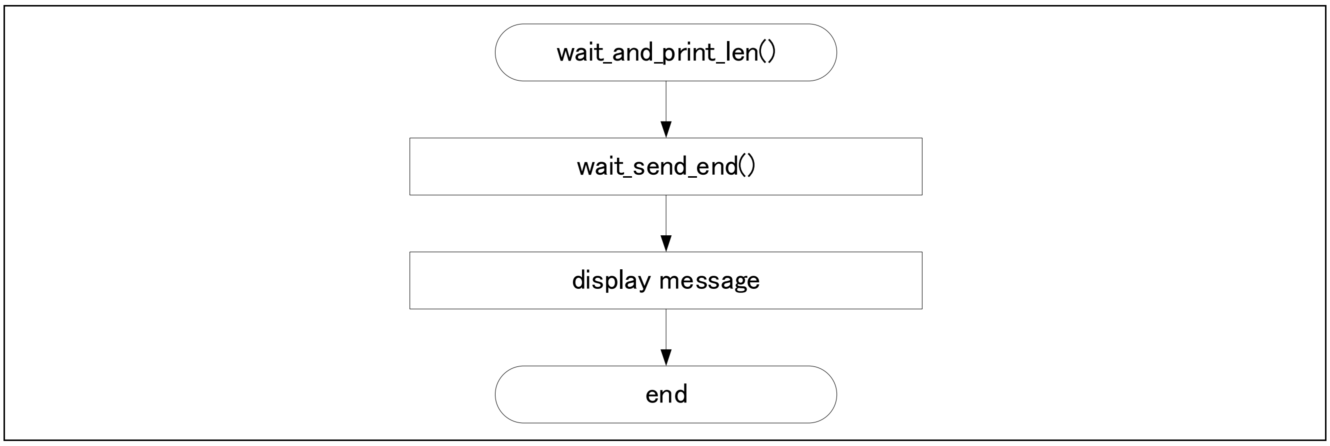
Figure 4-20 wait_and_print ()



4.8.9.11 wait_and_print_len()

Figure 4-21 shows the flowchart of wait_and_print_len().

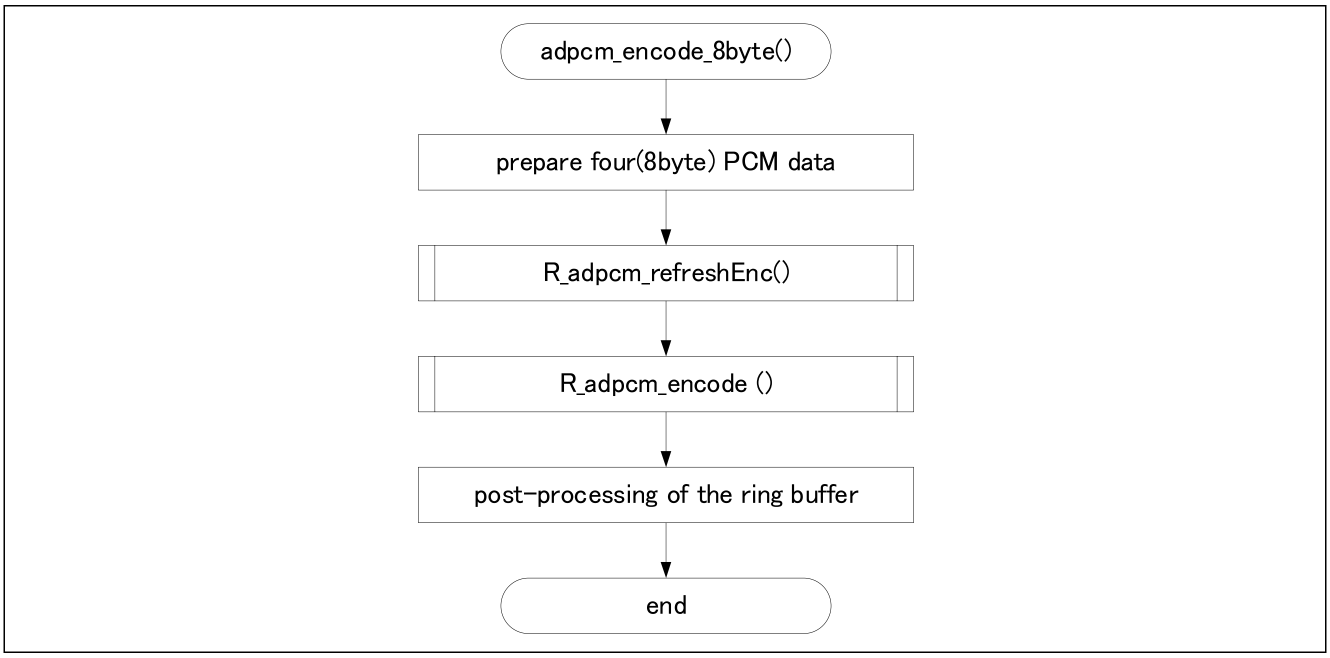
Figure 4-21 wait_and_print_len()



4.8.9.12 adpcm_encode_8byte()

Figure 4-22 shows the flowchart of adpcm_encode_8byte().

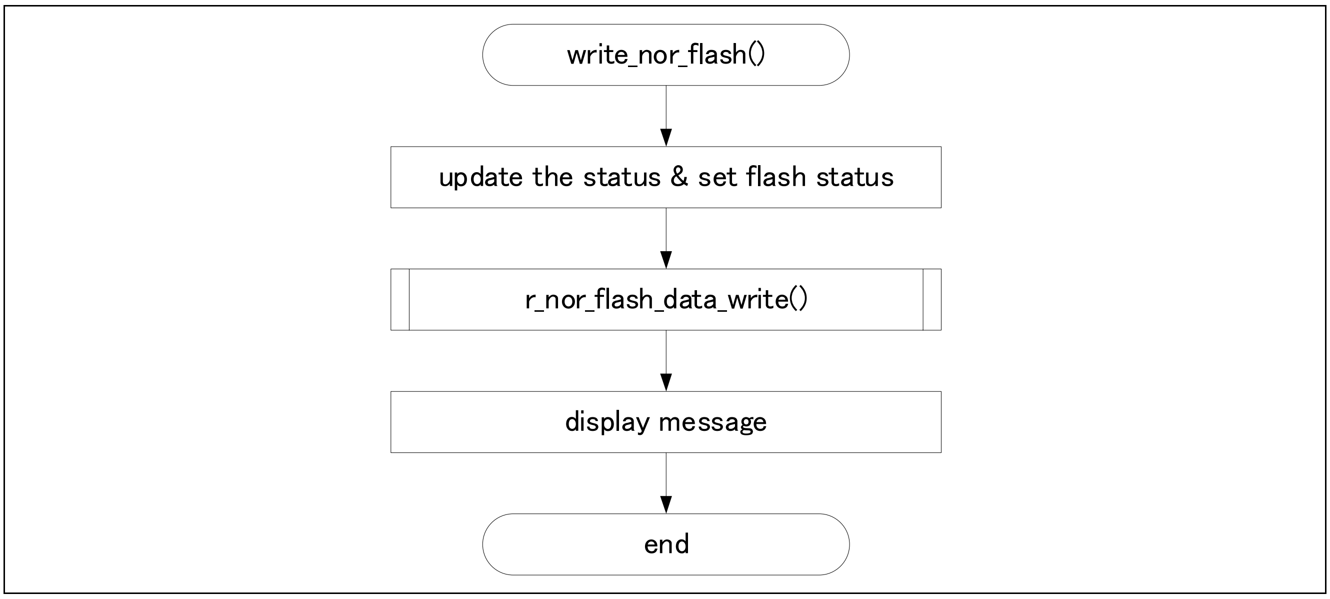
Figure 4-22 adpcm_encode_8byte()


adpcm_encode_8byte()
prepare four(8byte) PCM data
R_adpcm_refreshEnc()
R_adpcm_encode ()
post-processing of the ring buffer
end

4.8.9.13 write_nor_flash()

Figure 4-23 shows the flowchart of write_nor_flash().

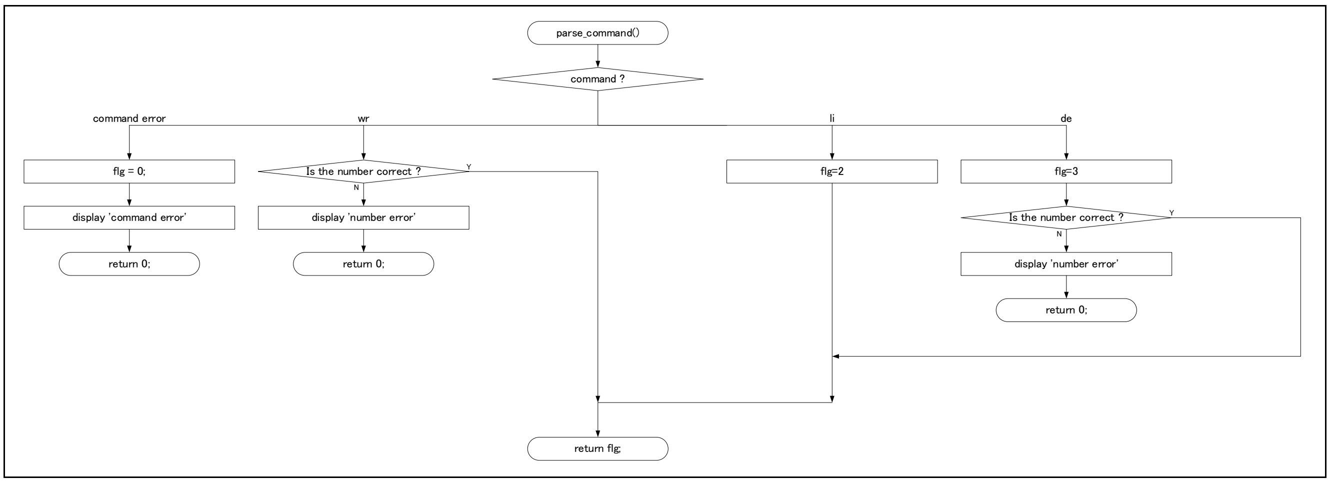
Figure 4-23 write_nor_flash()



4.8.9.14 parse_command()

Figure 4-24 shows the flowchart of write_nor_flash().

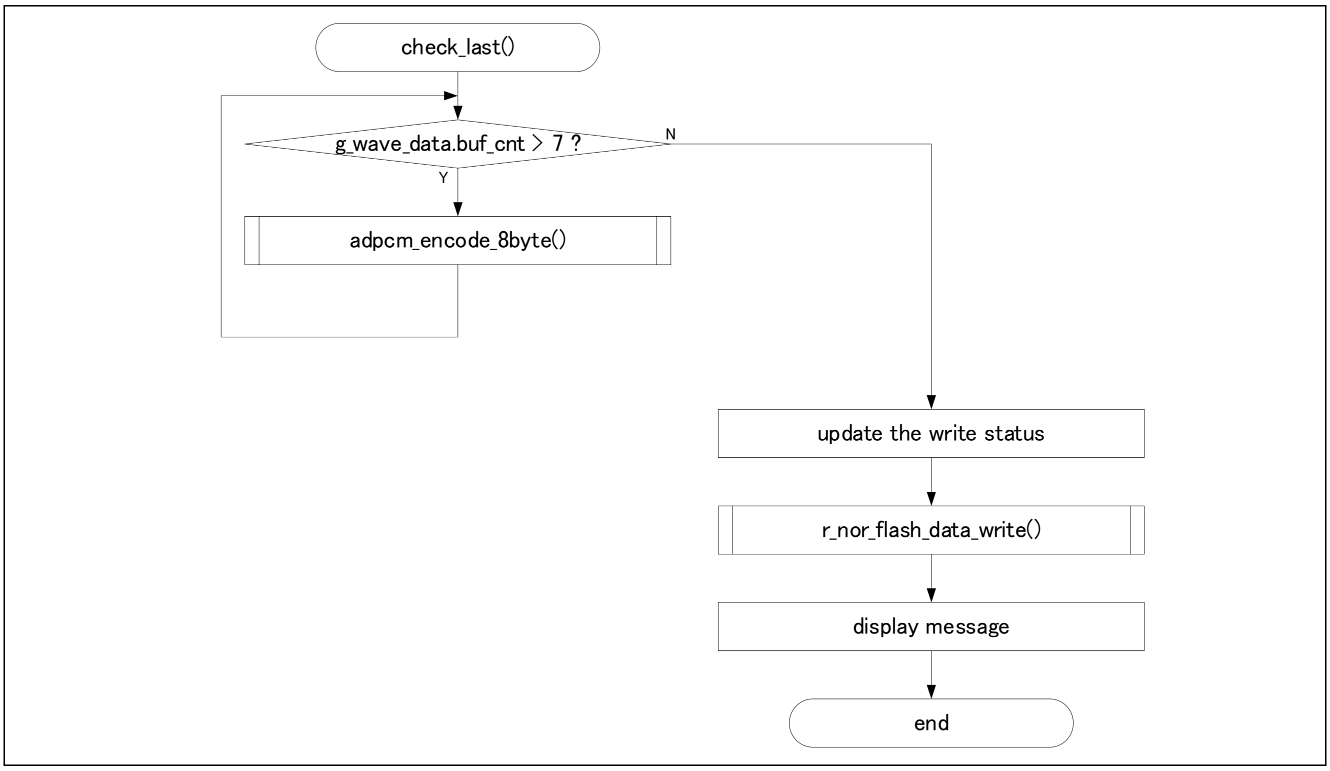
Figure 4-24 parse_command()



4.8.9.15 check_last()

Figure 4-25 shows the flowchart of check_last().

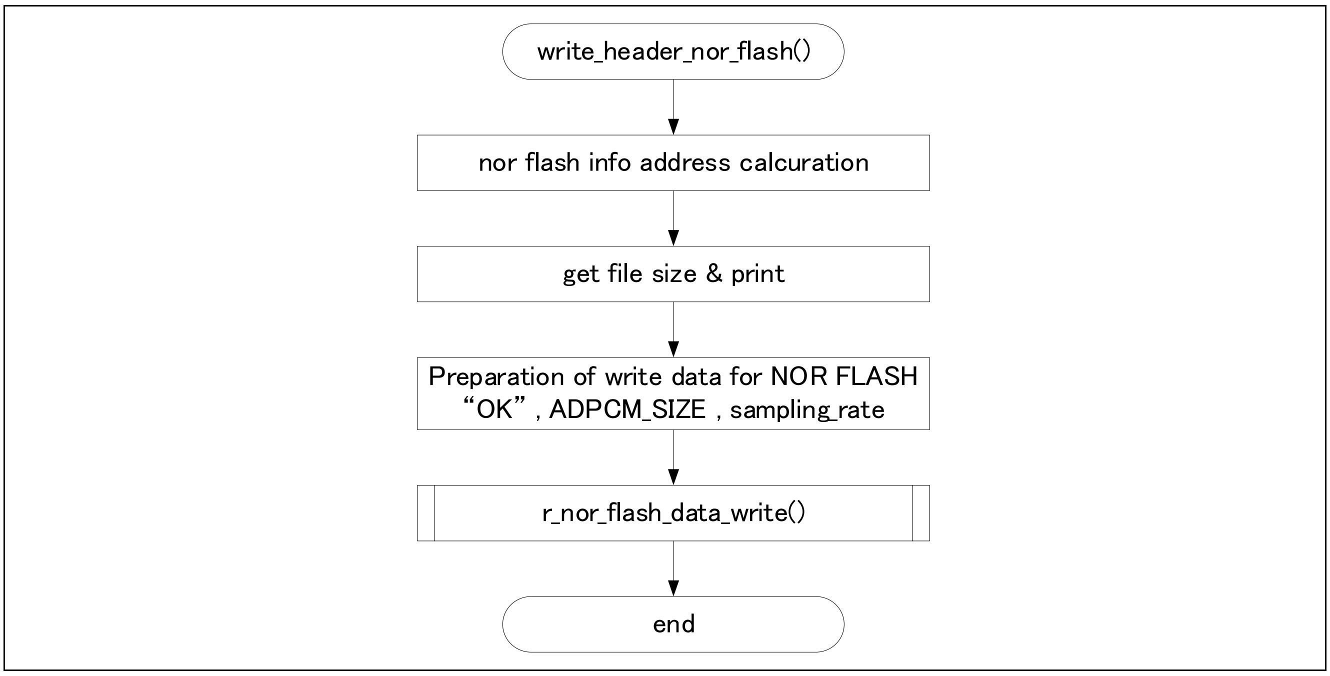
Figure 4-25 check_last()



4.8.9.16 write_header_nor_flash ()

Figure 4-26 shows the flowchart of check_last().

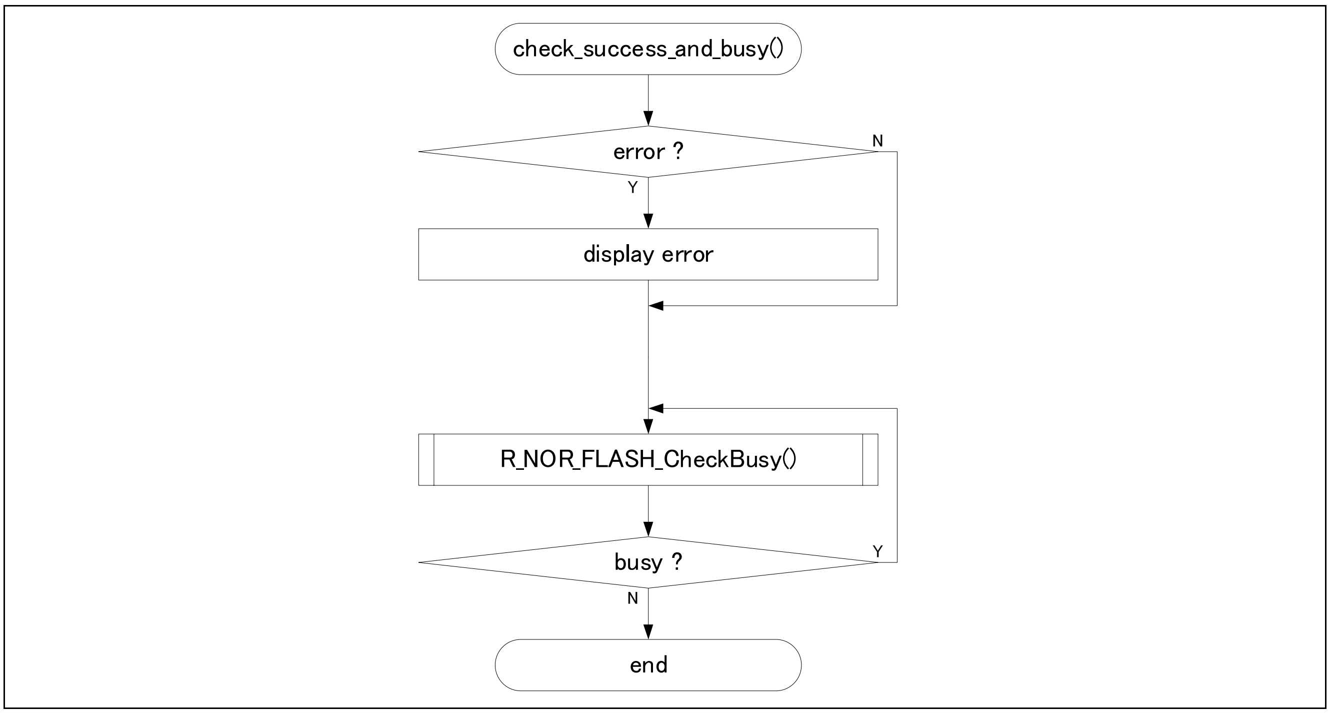
Figure 4-26 write_header_nor_flash ()


write_header_nor_flash()
nor flash info address calcuration
get file size & print
Preparation of write data for NOR FLASH
"OK" , ADPCM_SIZE , sampling_rate
r_nor_flash_data_write()
end

4.8.9.17 check_success_and_busy()

Figure 4-27 shows the flowchart of check_success_and_busy().

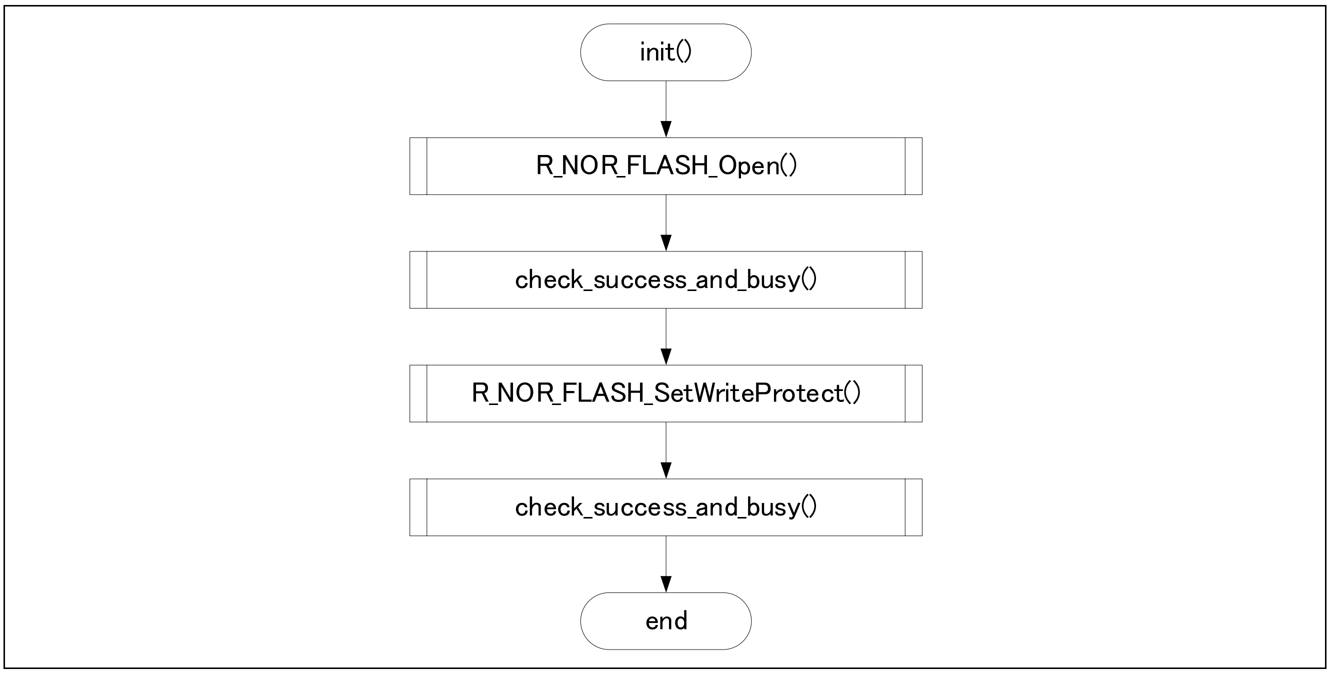
Figure 4-27 check_success_and_busy()



4.8.9.18 init()

Figure 4-28 shows the flowchart of init ().

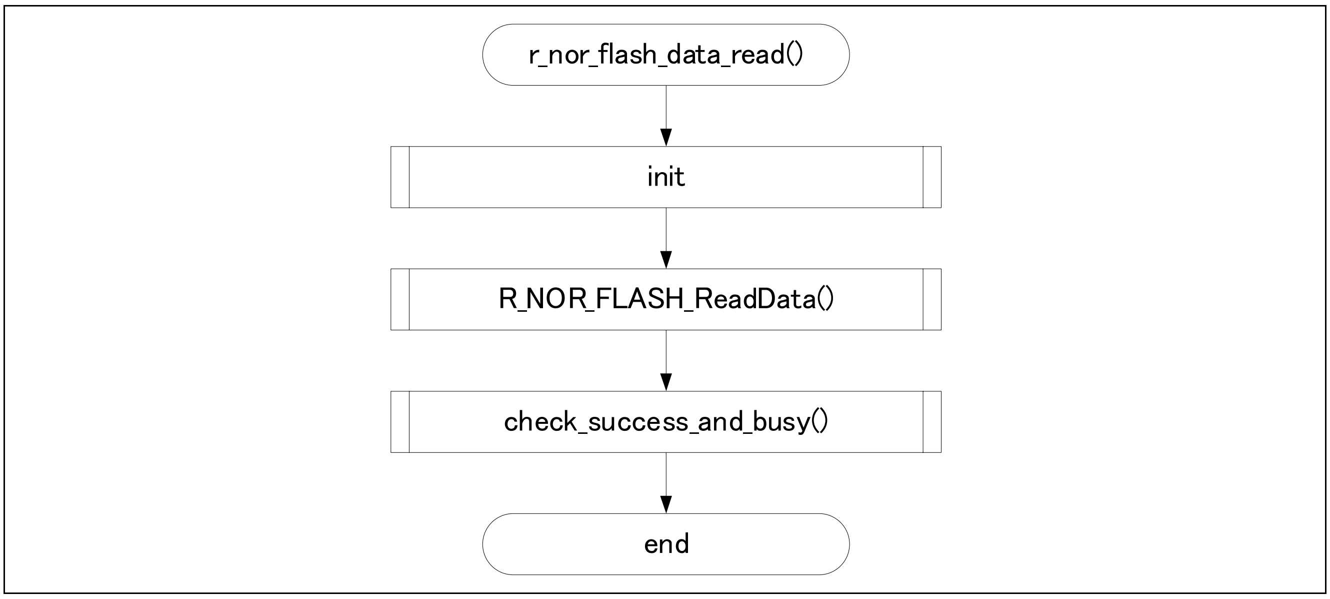
Figure 4-28 init()



4.8.9.19 r_nor_flash_data_read()

Figure 4-29 shows the flowchart of r_nor_flash_data_read().

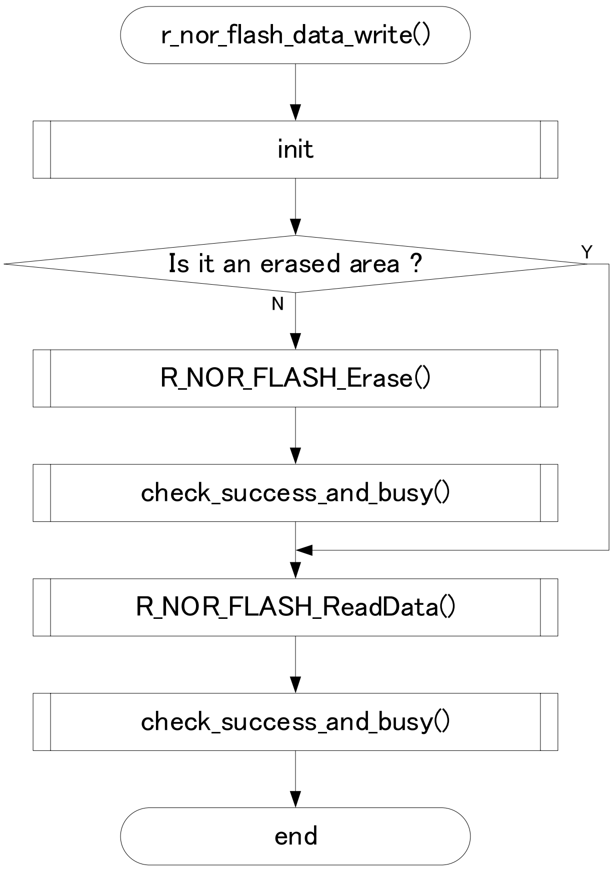
Figure 4-29 r_nor_flash_data_read()



4.8.9.20 r_nor_flash_data_write()

Figure 4-30 shows the flowchart of r_nor_flash_data_write().

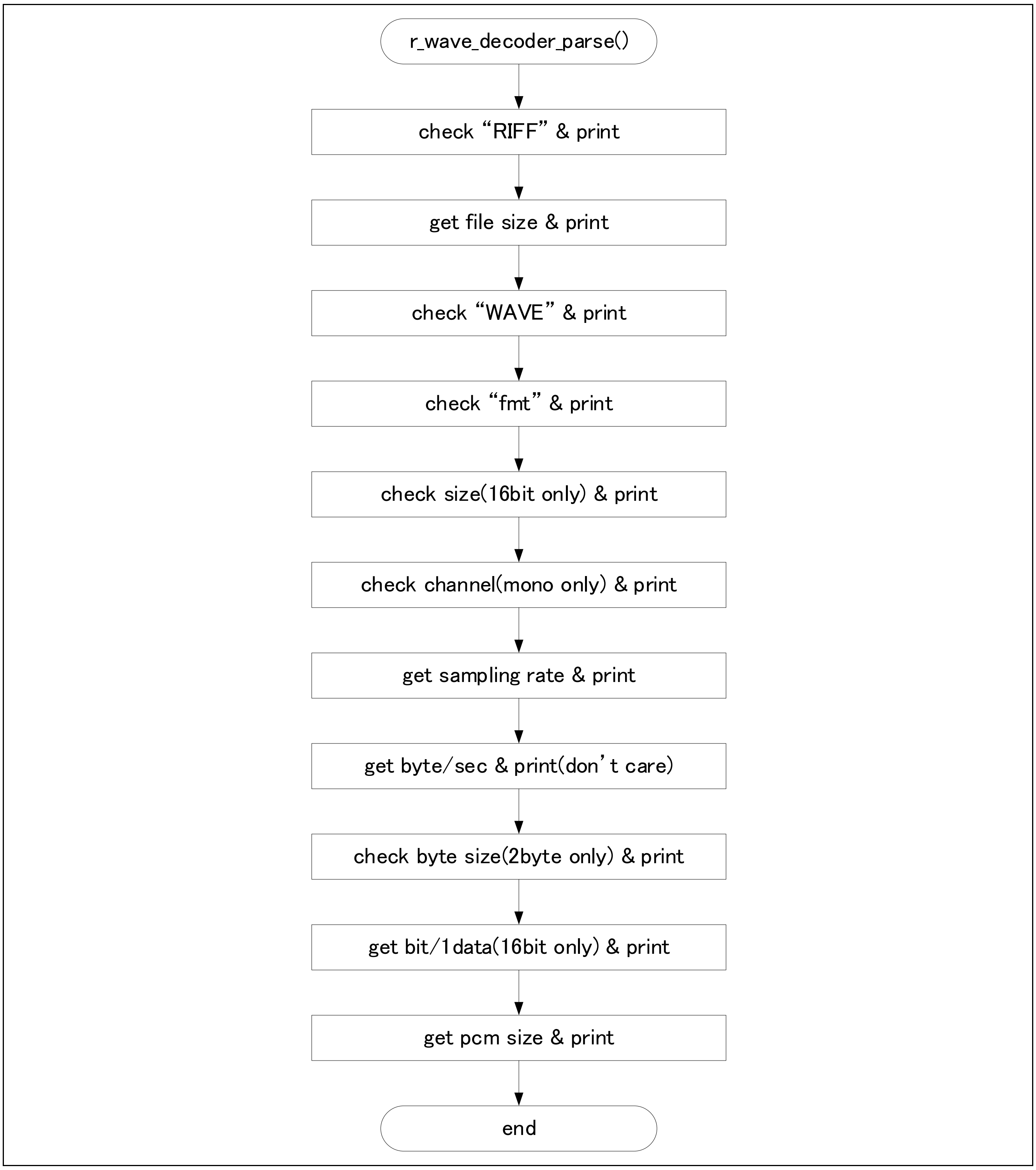
Figure 4-30 r_nor_flash_data_write()



4.8.9.21 r_wave_decoder_parse()

Figure 4-31 shows the flowchart of r_wave_decoder_parse().

Figure 4-31 r_wave_decoder_parse()


r_wave_decoder_parse()
check “RIFF” & print
get file size & print
check “WAVE” & print
check “fmt” & print
check size(16bit only) & print
check channel(mono only) & print
get sampling rate & print
get byte/sec & print(don’t care)
check byte size(2byte only) & print
get bit/1data(16bit only) & print
get pcm size & print
end

4.8.10 r01an7416_adpcm_writer.scfg

This is the Smart Configurator configuration file used in the sample code. It contains all the features configured in the Smart Configurator. The sample code settings are as follows.

Table 4-9 Parameters of Smart Configurator

Tag name	Component	Contents
Clocks	-	Operation mod: High-speed main mode 2.4 (V) to 5.5 (V) EV_{DD} setting:1.8V≦EV_{DD0}＜5.5V High-speed on-chip oscillator: 32MHz f_{IHP}: 32MHz f_{CLK}: 32000kHz (High-speed on-chip oscillator) f_{SXP}: 32.768kHz (Low-speed on-chip oscillator)
System	-	On-chip debug operation setting: Unused Pseudo-RRM/DMM function setting: - Start/Stop function setting: - Trace function setting: - Security ID setting: Use security ID Security ID: 0x00000000000000000000 Security ID authentication failure setting: -
Components	r_bsp	Start up select: Enable (use BSP startup) Control of invalid memory access detection: Disable RAM guard space (GRAM0-1): Disabled Guard of control registers of port function (GPORT): Disabled Guard of registers of interrupt function (GINT): Disabled Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC): Disabled Data flash access control (DFLEN): Disables Initialization of peripheral functions by Code Generator/Smart Configurator: Enable API functions disable: Enable Parameter check enable: Enable Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode: High-speed Enable user warm start callback (PRE): Unused Enable user warm start callback (POST): Unused Watchdog Timer refresh enable: Unused
	Config_LVD0	Operation mode setting: Reset mode Voltage detection setting: Reset generation level (V_{LVD0}): 1.86 (V)

Table 4-10 Parameters of Smart Configurato

Tag name	Component	Contents
Components	Config_CSI20	Transfer clock mode: Internal clock (master) Operation clock:CK10 Clock source: f_{CLK} Transfer mode setting: Single transfer mode Data length setting: 8 bits Transfer direction setting: MSB Data transmission/reception timing setting: Type1 Baud rate: 1000000bps Transmit end interrupt priority (INTST0): Level 3(low) Callback function setting: Transmission end, reception end, Overrun error
	Config_UART0	Operation clock:CK00 Clock source: f_{CLK}/2 Transfer mode setting: Single transfer mode Data length setting: 8 bits Transfer direction setting: LSB Parity setting: None Stop bit length setting: 1bit Transfer data level setting: Non-reverse Transfer rate setting:115200bps Transmit end interrupt priority (INTST0): Level 3(low) Callback function setting: Transmission end
	r_nor_flash	Parameter check: use system default ENABLE CHECKING OF THE WEL BIT.: Check WEL bit Select serial flash memory device: Renesas Electronics AT25QF Select serial flash memory capacity: 64M bit CS Port Number: PORT1 CS Bit Number: BIT7 Data transfer mode: CPU transfer (Software transfer) SPI(CSI) channel number: Channel 4 DTC Control Data Number: 0

4.8.10.1 Clocks

Set the clock used in the sample code. (For the settings, seeTable 4-9.)

4.8.10.2 System

Set the on-chip debug of the sample code. (The on-chip debug is not used).

The settings of “On-chip debug operation setting” and “Security ID authentication failure setting” affect the “On-chip debugging is enabled” setting in Table 5-3 Option Byte Settings. Care must be taken when changing the settings.

4.8.10.3 r_bsp

Set the startup of the sample code.

4.8.10.4 Config_LVD0

Set the power management of the sample code.

This setting affects the LVD0 setting in Table 5-3 Option Byte Settings. Care must be taken when changing the setting.

4.8.10.5 Config_CSI20

Used to control SPI for NOR FLASH.

4.8.10.6 Config_UART0

Used for communication with the PC.

4.8.10.7 r_nor_flash

This is middleware for NOR FLASH.

5. Audio Playback Method

5.1 Software Description

The project for the RL78/G23-64p FPB (ra01an7416_adpcm_player) supplied with this application note is used to play sound. This project reads ADPCM data written in NOR FLASH, and then sends decoded PCM data to Renesas ARD-AUDIO-DA7212 that plays sound.

Each time the user switch (SW1) is pressed, a sound assigned a number from 0 to 9 is reproduced sequentially.

5.2 Hardware Connection

For details on hardware connection, see Figure 3-1 Hardware Configuration Example.

Note: The SPI communication rate between RL78/G23-64p FPB and NOR FLASH for ra01an7416_adpcm_player is assumed to be 1 MHz.
This is because consideration is required for waveform rounding in connections that use a breadboard or jump wires.

5.3 Terminal software

No terminal software is used with ra01an7416_adpcm_player.

5.4 Using a Debugger

ra01an7416_adpcm_player can use a debugger because no terminal software is used.

Like ra01an7416_adpcm_writer, operation in standalone mode is also possible.

5.5 Commands

No commands are used.

5.6 Operation Methods

First, use the Renesas Flash Programmer to write ra01an7416_adpcm_player.mot (supplied with this application note) on the RL78/G23-64p FPB.

Each time the user switch (SW1) of the RL78/G23-64p FPB is pressed, a sound assigned a number from 0 to 9 is reproduced.

5.7 Error Display

No error display function is provided.

5.8 Detailed Software Description

5.8.1 Overview of operation

This sample code uses the Renesas ARD-AUDIO-DA7212. It also uses I^{2}C communication for register settings in the audio CODEC and I^{2}S communication for sound data transmission processing.

BCLK uses the Through content of ELCL and the output pin is changed to P51.

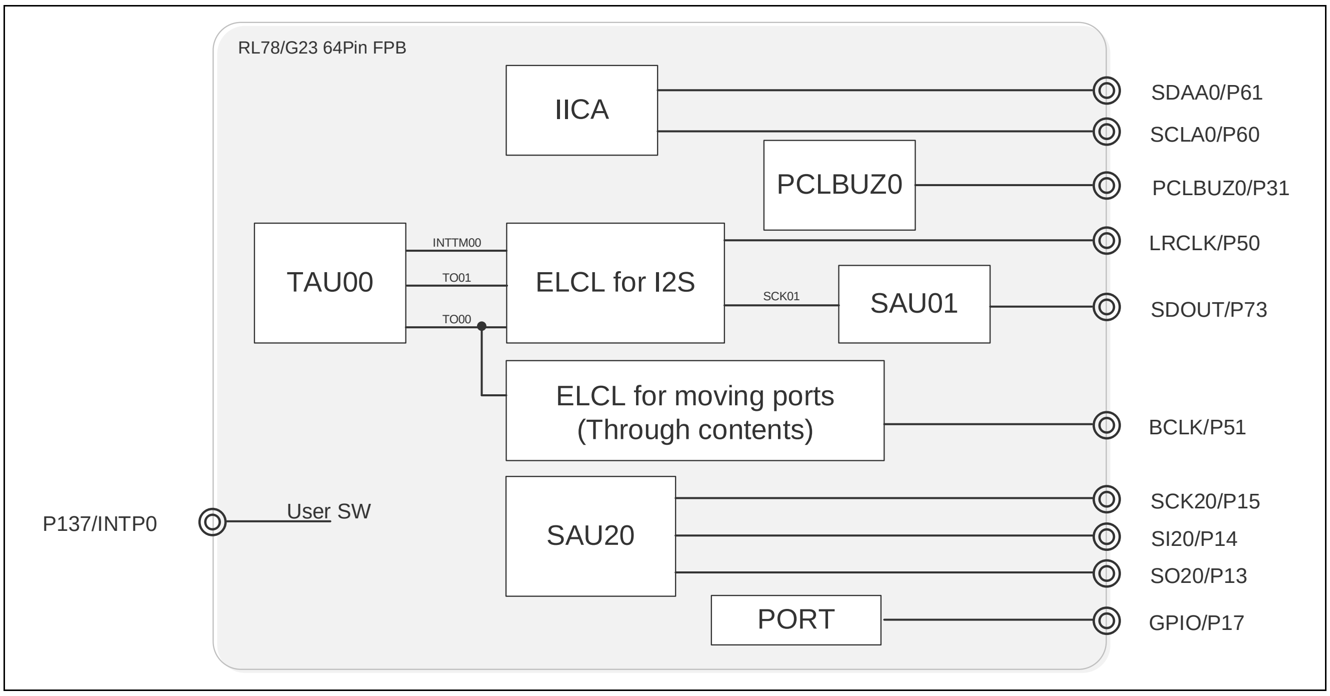
In addition, SPI communication is used for processing to read from Renesas AT25QF641B (NOR FLASH).

The sample code works as follows.

(1) When the user switch (SW1) is pressed, IICA for I^{2}C communication starts operation and sets registers of ARD-AUDIO-DA7212.
(2) ADPCM data is read from NOR FLASH through SPI communication.
(3) CSI01 for I^{2}S communication starts operation (waiting for SCK01).
(4) TAU0 for BCLK and LRCLK starts operation and data transmission starts.
(5) CSI01 and TAU0 stops, and then the operation returns to (1).
Note: Step (2) is repeated until audio playback is complete.

Figure 5-1 shows the system configuration of the sample code.

Figure 5-1 System Configuration of the Sample Code



5.8.2 Folder configuration

Table 5-1 and Table 5-2 show the configuration of the source files and header files used by the sample code. Note that files that are automatically generated by an integrated development environment and those in a bsp environment are excluded.

Table 5-1 Folder configuration (1/2)

Folder/File configuration	Outline	Created by Smart configurator
r01an7416_adpcm_player<DIR>Note 3	Sample code folder	
└ src<DIR>	Program folder	
├ csi01<DIR>	Folder for CSI01 program	
├ ├ csi01.c	Source file for CSI01	
├ └ csi01.h	Header file for CSI01	
├ elcl<DIR>	Folder for ELCL program	
├ ├ elcl.c	Source file for ELCL	
├ └ elcl.h	Header file for ELCL	
├ iica0<DIR>	Folder for IICA0 program	
├ ├ iica0.c	Source file for IICA0	
├ └ iica0.h	Header file for IICA0	
├ s2<DIR>	M3S-S2-Tiny (ADPCM encoder/decoder)	
├ smc_gen<DIR>	Folder generated by the Smart Configurator	√
├ ├ Config_CSI20	Folder for CSI20 program	√
├ ├ ├ Config_CSI20.c	Source file for CSI20	√
├ ├ ├ Config_CSI20.h	Header file for CSI20	√
├ ├ └ Config_CSI20_user.c	Interrupt source file for CSI20	√
├ ├ Config_INTC<DIR>	Folder for INTC program	√
├ ├ ├ Config_INTC.c	Source file for INTC	√
├ ├ ├ Config_INTC.h	Header file for INTC	√
├ ├ └ Config_INTC_user.c	Interrupt source file for INTC	√
├ ├ Config_PCLBUS0<DIR>	Folder for PCLBUZ0 program	√
├ ├ ├ Config_PCLBUZ0.c	Source file for PCLBUZ0	√
├ ├ ├ Config_PCLBUS0.h	Header file for PCLBUZ0	√
├ ├ └ Config_PCLBUS0_user.c	Interrupt source file for PCLBUZ0	√Note 2
├ ├ Config_PORT<DIR>	Folder for PORT program	√
├ ├ ├ Config_PORT.c	Source file for PORT	√
├ ├ ├ Config_PORT.h	Header file for PORT	√
├ ├ └ Config_PORT_user.c	Interrupt source file for PORT	√Note 1

Note: <DIR> means a directory.

Note 1: Not used in this sample code.

Note 2: Added the initial settings to the file generated by the Smart Configurator.

Note 3. The sample code of the IAR versions contains buildinfo.ipcf. For the ipcf file, refer to the RL78 Smart Configurator User's Guide: IAR (R20AN0581).

Table 5-2 Folder configuration (2/2)

Folder/File configuration	Outline	Created by Smart configurator
├ ├ Config_TAU0_1<DIR>	Folder for TAU00 program	√
├ ├ ├ Config_TAU0_0.c	Source file for TAU00	√
├ ├ ├ Config_TAU0_0.h	Header file for TAU00	√
├ ├ └ Config_TAU0_0_user.c	Interrupt source file for TAU01	√Note 1
├ ├ Config_TAU0_1<DIR>	Folder for TAU01 program	√
├ ├ ├ Config_TAU0_1.c	Source file for TAU01	√
├ ├ ├ Config_TAU0_1.h	Header file for TAU01	√
├ ├ └ Config_TAU0_1_user.c	Interrupt source file for TAU01	√Note 1
├ ├ Config_TAU0_7<DIR>	Folder for TAU07 program	√
├ ├ ├ Config_TAU0_7.c	Source file for TAU07	√
├ ├ ├ Config_TAU0_7.h	Header file for TAU07	√
├ ├ └ Config_TAU0_7_user.c	Interrupt source file for TAU07	√Note 4
├ ├ Config_Through<DIR>	Folder for Through programNote 5	√
├ ├ ├ Config_Through.c	Source file for Through	√
├ ├ ├ Config_Through.h	Header file for Through	√
├ ├ └ Config_Through_user.c	Interrupt source file for Through	√
├ ├ general<DIR>	Folder for initialization and common program	
├ ├ r_bsp<DIR>	Folder for BSP program	
├ ├ r_config<DIR>	Program folder	
├ ├ r_nor_flash_rl78<DIR>	Folder for storing the NOR FLASH libraryNote 6	
├ ├ r_pincfg<DIR>	Folder for storing pincfg files	
├ main.c	Main program	
├ main.h	Header file	
├ r_flash_adpcm_player.c	ADPCM playback main program	
├ r_flash_adpcm_player.h	Header file	
├ r_codec.c	DA7212 settings	
├ r_codec.h	Header file	

Note: <DIR> means a directory.
Note 4: Added the interrupt handling routine to the file generated by the Smart Configurator.
Note 5. In the New Component dialog box of the Smart Configurator, click the [Download ELCL modules] link and download ELCL Through.
Note 6. Download the RL78 Family Serial NOR Flash Memory control module and Software Integration System module by using the New Component dialog box of the Smart Configurator.

Since I^{2}S communication uses CSI, it does not follow the original CSI standard and overrun errors occur. Determination of error type[*1] and unused functions are removed from the code generated by the Smart Configurator.

IICA also removed unnecessary functions[*2] from the code generated by the Smart Configurator.

*1: err_type = (uint8_t)(SSR01 & _0001_SAU_OVERRUN_ERROR);
if (1U != err_type) are removed
Functions related to Config_CSI01_User.c
*2: Functions related to Config_IICA0_User.c are removed.

5.8.3 Option Byte Settings

Table 5-3 shows the option byte settings.

Table 5-3 Option Byte Settings

Address	Setting Value	Contents
000C0H/040C0H	1110 1111B (EFH)	Operation of Watchdog timer is stopped (counting is stopped after reset)
000C1H/040C1H	1111 1110B (FEH)	LVD0 operating mode: reset mode Detection voltage: Rising edge 1.90V Falling edge 1.86V
000C2H/040C2H	1110 1000B (E8H)	Flash operating mode: HS mode High-speed on-chip oscillator clock: 32MHz
000C3H/040C3H	1000 0101B (85H)	On-chip debugging is enabled

5.8.4 ROM/RAM Size

Table 5-4 shows the ROM and RAM sizes used by this sample code when the CC-RL optimization level is -Olite (partial optimization). (The size varies depending on the optimization level.)

Table 5-4 ROM/RAM Size

Sample Project Name	RON Size	RAM Size
ra01an7416_adpcm_player	9019 Byte	919 Byte

5.8.5 Constants

Table 5-5 shows the constants that are used in this sample code.

Table 5-5 Constants used in the sample code

Constant Name	Setting Value	Contents	File
WAIT_TIME	100	IICA0 communication wait time	r_codec.h
SENSOR_ADD	0x34	Audio CODEC IC address	r_codec.h

5.8.6 Variables

Table 5-6 shows the global variables used in this sample code.

Table 5-6 Global variables used in the sample code

Type	Variable name	Contents	Functions used in
st_sound_buf_t	gp_recv_sb	Pointer to the ADPCM receiving structure	recv_sb_status_clear status_init check_recv_sb adpcm2pcm check_dec_sb play_start
st_intp0_t	g_intp0	Structure related to user switch (SW1) interrupts	r_Config_INTC_intp0_interrupt wait_sw play_start r_nor_flash_operation
volatile uint16_t	g_ms_timer	Wait process count value	r_ms_delay r_Config_TAU0_7_interrupt
volatile uint8_t	g_tx_done_flag	Transmission end flag	r_nor_flash_operation r_csi01_callback_sendend
volatile uint8_t	g_sample_mode	I^{2}C communication status	r_codec_init send_i2c r_codec_stop r_iica0_callback_master_sende nd

5.8.7 Functions

Table 5-7 shows the functions used in the sample code. However, the unchanged functions generated by the Smart Configurator are excluded.

In addition, this application note only covers the functions that have been changed since Application Note I^{2}S Communication with ELCL and SPI. Refer to Application Note I^{2}S Communication with ELCL and SPI.

Table 5-7 Functions

	Function name	Outline	Source File
g	main	Main process	main.c
g	r_adpcm_player_operation	Main process of adpcm_player	r_adpcm_player.c
s	init	Initialization process of adpcm_player	r_adpcm_player.c
s	wait_sw	Determination of whether the user switch (SW1) is pressed	r_adpcm_player.c
s	play_start	adocm playback start	r_adpcm_player.c
s	check_recv_sb	Receive buffer determination	r_adpcm_player.c
s	check_dec_sb	adpcm decode buffer determination	r_adpcm_player.c
s	status_init	adpcm_writer status initialization	r_adpcm_player.c
s	recv_sb_status_clear	Clear the status of reception-related structure	r_adpcm_player.c
s	dec_sb_status_clear	Clear the status of decoding-related structure	r_adpcm_player.c
s	i2s_sb_status_clear	Clear the status of playback-related structure	r_adpcm_player.c
s	get_i2s_data	I^{2}S transfer data acquisition	r_adpcm_player.c
s	adpcm2pcm	ADPCM decoding	r_adpcm_player.c
s	change_i2s_buf	Playback structure change	r_adpcm_player.c
g	r_codec_init	I^{2}C open process	r_codec.c
s	send_i2c	I^{2}C transmission	r_codec.c
g	r_codec_start	DA7212 initialization	r_codec.c
g	r_codec_change_sampling_rate	DA7212 playback rate change	r_codec.c
g	r_codec_stop	I^{2}C stop confirmation	r_codec.c

Note 1. The first letter “g” indicates a global function and “s” indicates a static function.

5.8.8 Function Specifications

This part describes function specifications of the sample code.

Note that this application note only covers the functions that have been changed since Application Note I^{2}S Communication with ELCL and SPI. Refer to Application Note I^{2}S Communication with ELCL and SPI.

[Function name] main

Outline	Main process
Header	stdint.h, stdbool.h, main.h, r_adpcm_player.h
Declaration	void main (void);
Description	This is the main function. It calls r_adpcm_player_operation.
Arguments	None
Return value	None
Remarks	None

[Function name] r_adpcm_player_operation(r_adpcm_player.c)

Outline	Main process of adpcm_player
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	void r_adpcm_player_operation(void);
Description	This function performs the main process of adpcm_player.
Arguments	None
Return value	None
Remarks	None

[Function name] init(r_adpcm_player.c)

Outline	Initialization process of adpcm_player
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void init(void);
Description	This function performs the initialization process of adpcm_player.
Arguments	None
Return value	None
Remarks	None

[Function name] wait_sw(r_adpcm_player.c)

Outline	User switch (SW1) determination process
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void wait_sw(void);
Description	This function performs the user switch (SW1) determination process.
Arguments	None
Return value	None
Remarks	None

[Function name] play_start(r_adpcm_player.c)

Outline	Playback start process
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void play_start(void);
Description	This function identifies the data stored in NOR FLASH based on the number of times the user switch (SW1) is pressed, and then performs the playback start process. This function also identifies the data size and specifies the sampling rate for DA7212.
Arguments	None
Return value	None
Remarks	None

[Function name] check_recv_sb (r_adpcm_player.c)

Outline	ADPCM reception process
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void check_recv_sb (void);
Description	This function performs processing to read from NOR FLASH if the ADPCM receive buffer is empty.
Arguments	None
Return value	None
Remarks	None

[Function name] check_dec_sb (r_adpcm_player.c)

Outline	ADPCM decode process
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void check_dec_sb (void);
Description	This function performs the ADPCM decode process if ADPCM reception has finished.
Arguments	None
Return value	None
Remarks	None

[Function name] status_init (r_adpcm_player.c)

Outline	Status initialization in r_adpcm_player
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void status_init (void);
Description	This function performs processing to initialize the status in r_adpcm_player.
Arguments	None
Return value	None
Remarks	None

[Function name] recv_sb_status_clear (r_adpcm_player.c)

Outline	Clear the status of reception-related structure
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void recv_sb_status_clear (void);
Description	This function initializes the status of the reception-related structure.
Arguments	None
Return value	None
Remarks	None

[Function name] dec_sb_status_clear (r_adpcm_player.c)

Outline	Clear the status of decoding-related structure
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void dec_sb_status_clear (void);
Description	This function initializes the status of the ADPCM decoding-related structure.
Arguments	None
Return value	None
Remarks	None

[Function name] i2s_sb_status_clear (r_adpcm_player.c)

Outline	Clear the status of I^{2}S-related structure
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void i2s_sb_status_clear (void);
Description	This function initializes the status of the I^{2}S-related structure.
Arguments	None
Return value	None
Remarks	None

[Function name] get_i2s_data (r_adpcm_player.c)

Outline	I^{2}S transfer data acquisition
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void get_i2s_data (void);
Description	This function acquires data to be sent to I^{2}S.
Arguments	None
Return value	None
Remarks	None

[Function name] adpcm2pcm (r_adpcm_player.c)

Outline	ADPCM decoding
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void adpcm2pcm (void);
Description	This function performs ADPCM decoding.
Arguments	None
Return value	None
Remarks	None

[Function name] change_i2s_buf (r_adpcm_player.c)

Outline	Playback structure change
Header	stdint.h, stdbool.h, r_smc_entry.h, csi01.h, elcl.h, r_adpcm.h, r_adpcm_player.h, r_nor_flash_rl78_if, r_codec.h
Declaration	static void change_i2s_buf (void);
Description	This function replaces the I^{2}S playback buffer with the ADPCM decoded buffer.
Arguments	None
Return value	None
Remarks	None

[Function name] r_codec_init (r_codec.c)

Outline	I^{2}C open process
Header	stdint.h, stdbool.h, iica0.h, r_codec.h, Config_TAU0_7.h
Declaration	void change_i2s_buf (void);
Description	This function performs the I^{2}C open process.
Arguments	None
Return value	None
Remarks	None

[Function name] send_i2c (r_codec.c)

Outline	I^{2}C transmission
Header	stdint.h, stdbool.h, iica0.h, r_codec.h, Config_TAU0_7.h
Declaration	static void send_i2c (void);
Description	This is an I^{2}C transmission function.
Arguments	None
Return value	None
Remarks	None

[Function name] r_codec_start (r_codec.c)

Outline	DA7212 initialization
Header	stdint.h, stdbool.h, iica0.h, r_codec.h, Config_TAU0_7.h
Declaration	static void r_codec_start (void);
Description	This function initializes DA7212.
Arguments	None
Return value	None
Remarks	None

[Function name] r_codec_change_sampling_rate (r_codec.c)

Outline	DA7212 playback rate change
Header	stdint.h, stdbool.h, iica0.h, r_codec.h, Config_TAU0_7.h
Declaration	static void r_codec_change_sampling_rate (void);
Description	This function changes the sampling rate of DA7212.
Arguments	None
Return value	None
Remarks	None

[Function name] r_codec_stop (r_codec.c)

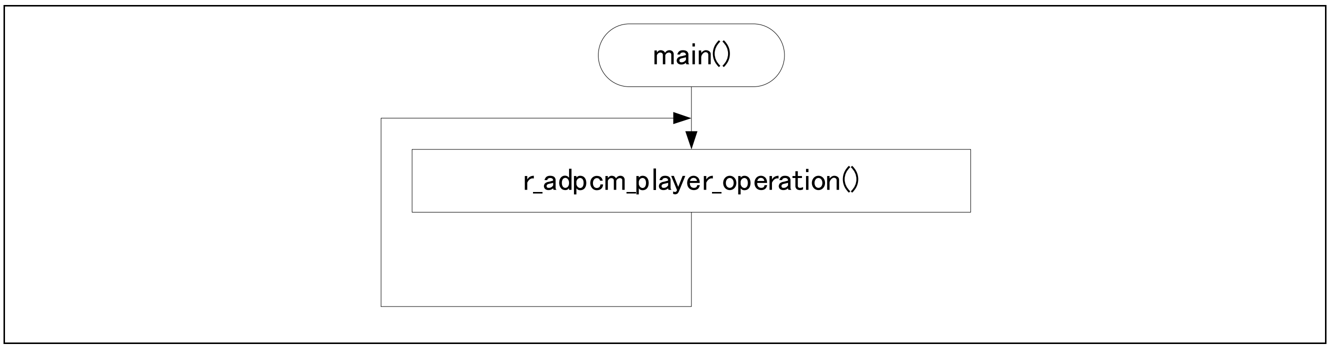
Outline	I^{2}C stop confirmation
Header	stdint.h, stdbool.h, iica0.h, r_codec.h, Config_TAU0_7.h
Declaration	static void r_codec_stop (void);
Description	This function confirms that I^{2}C has stopped and waits for termination for 100 ms.
Arguments	None
Return value	None
Remarks	None

5.8.9 Flow Charts

5.8.9.1 main()

Figure 5-2 shows the flowchart of the main process.

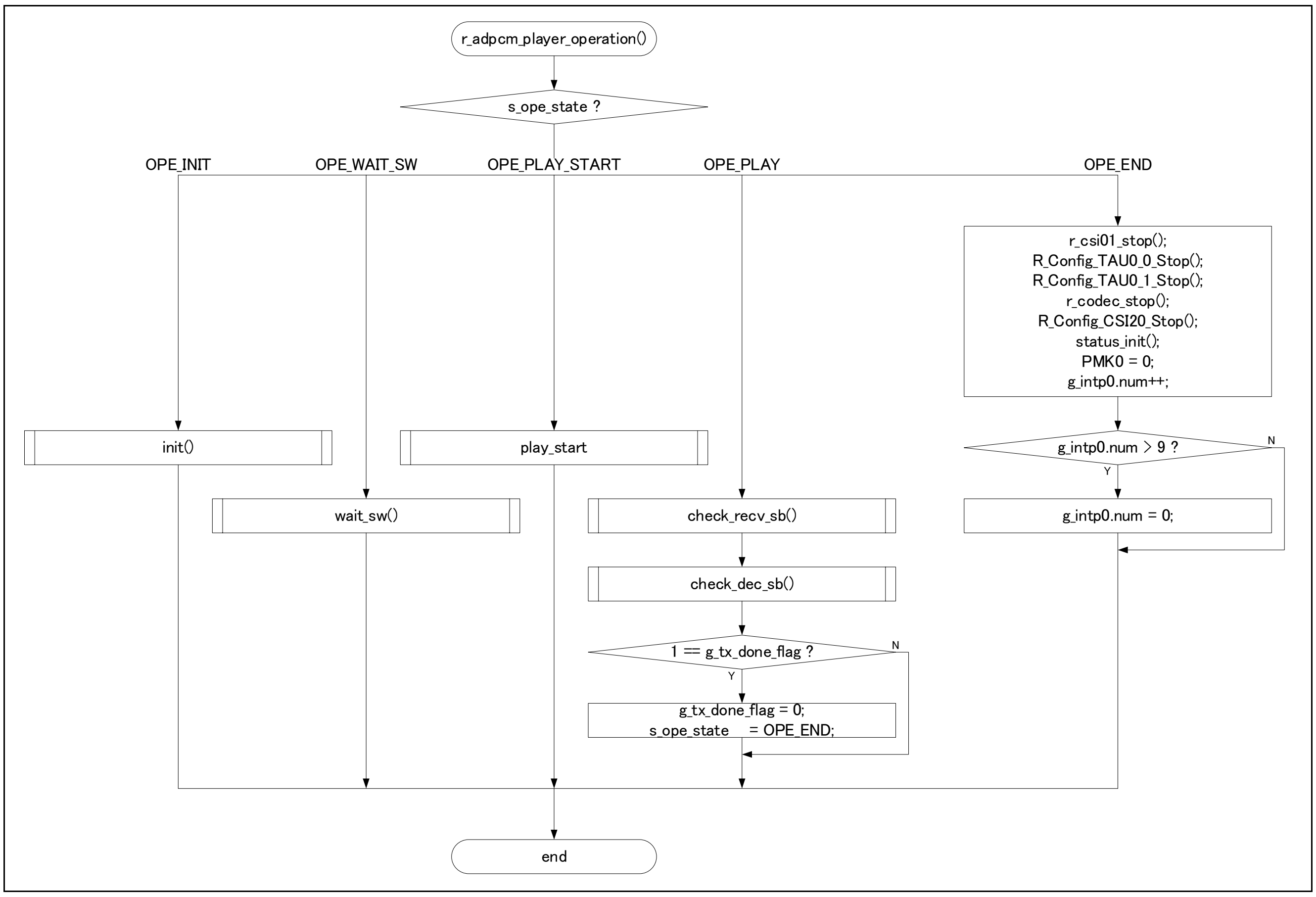
Figure 5-2 Main Process



5.8.9.2 r_adpcm_player_operation ()

Figure 5-3 shows the flowchart of r_adpcm_player_operation ().

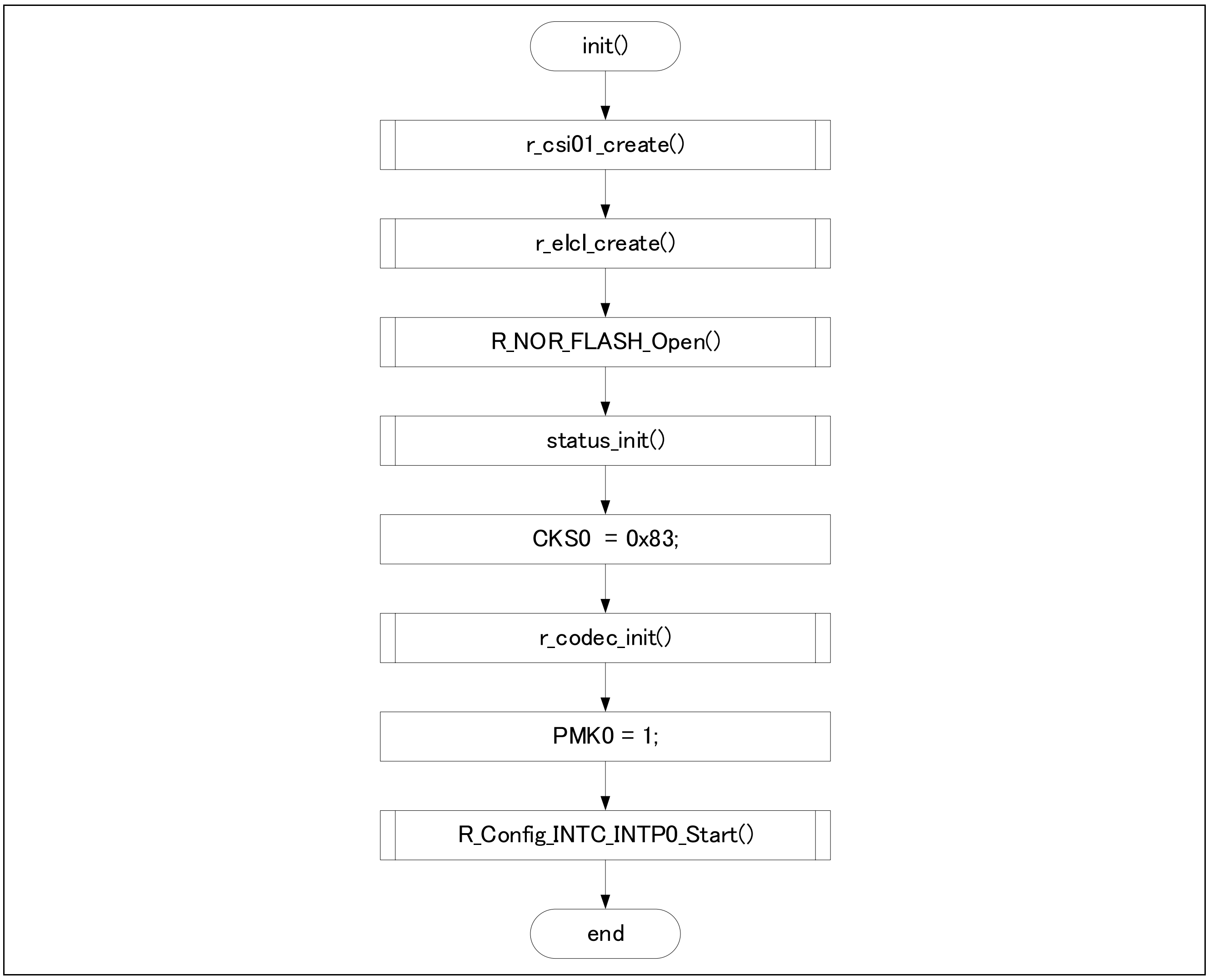
Figure 5-3 r_adpcm_player_operation ()



5.8.9.3 init ()

Figure 5-4 shows the flowchart of init ().

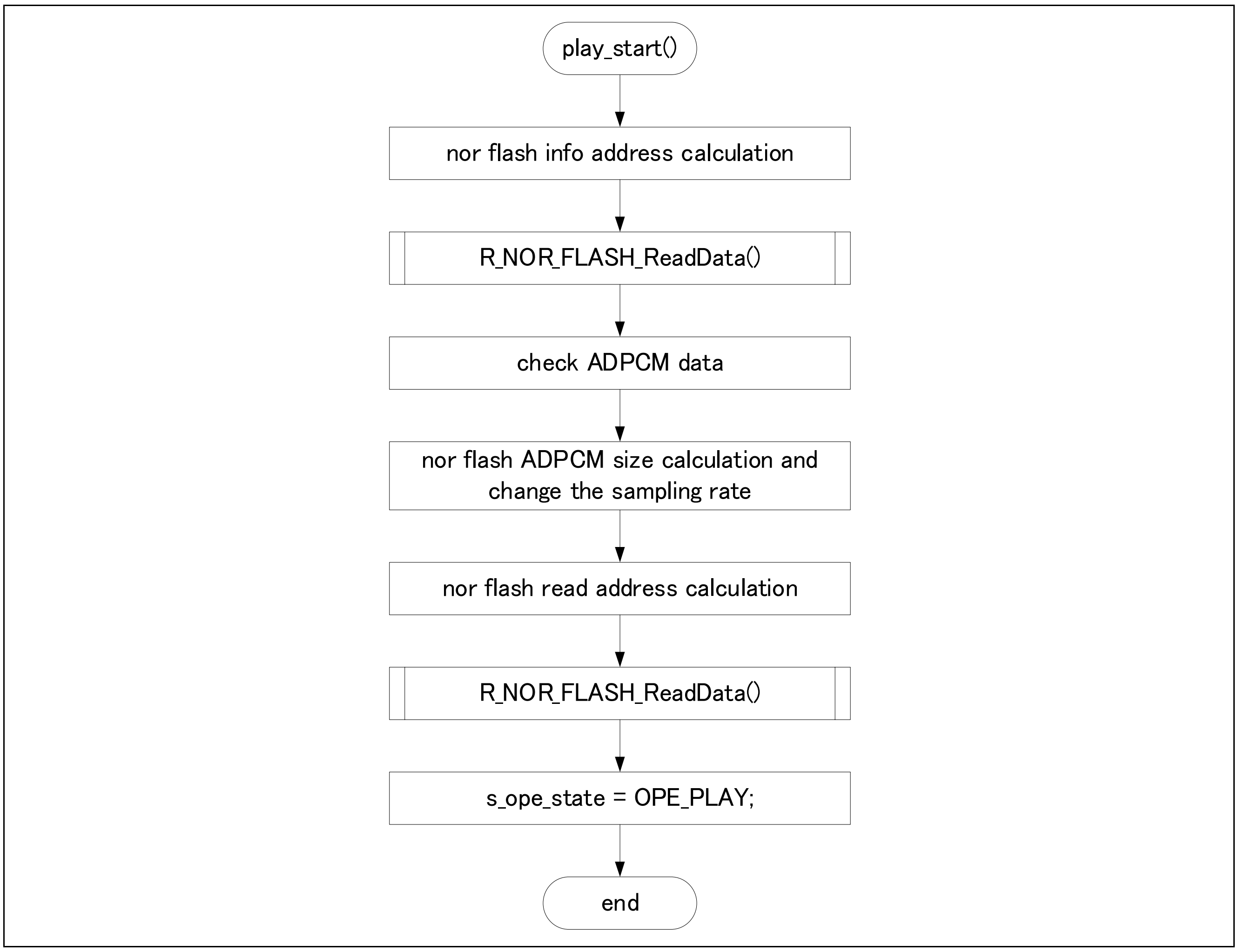
Figure 5-4 init ()



5.8.9.4 play_start ()

Figure 5-5 shows the flowchart of play_start ().

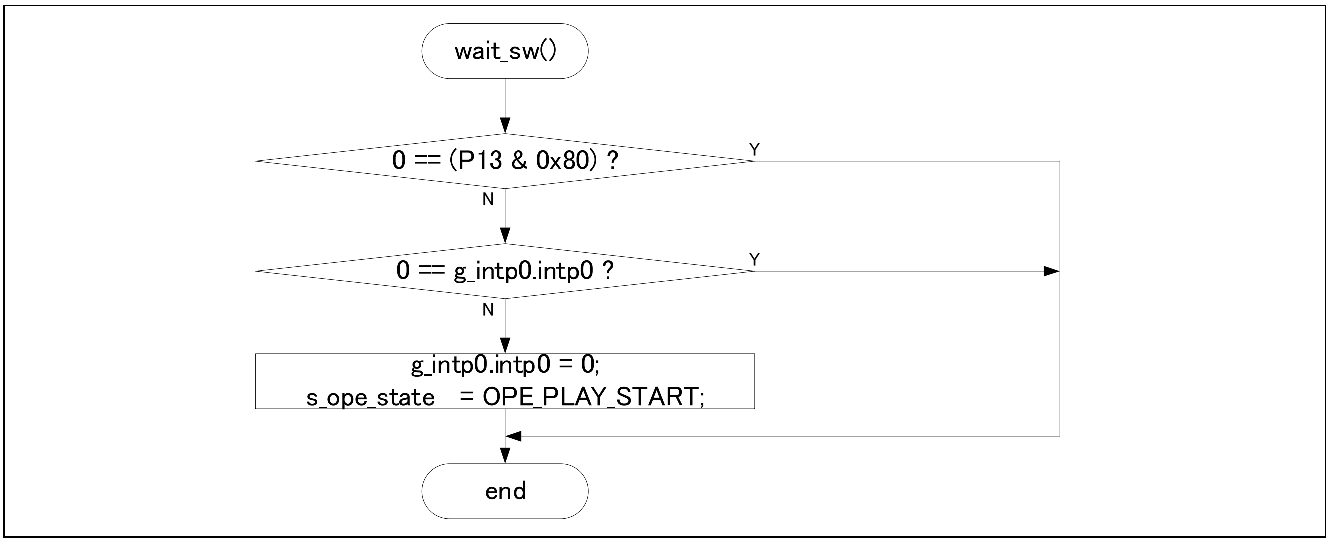
Figure 5-5 play_start ()


play_start()
nor flash info address calculation
R_NOR_FLASH_ReadData()
check ADPCM data
nor flash ADPCM size calculation and change the sampling rate
nor flash read address calculation
R_NOR_FLASH_ReadData()
s_ope_state = OPE_PLAY;
end

5.8.9.5 wait_sw ()

Figure 5-6 shows the flowchart of wait_sw ().

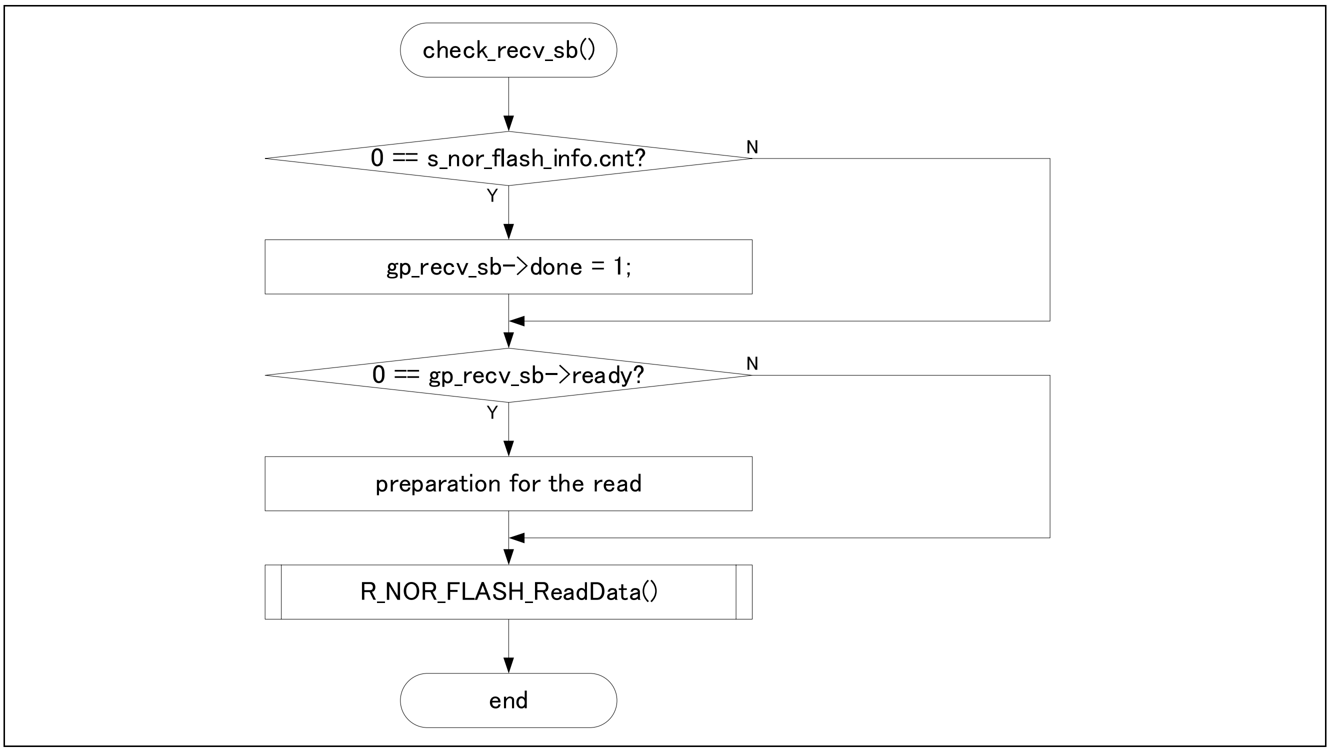
Figure 5-6 wait_sw ()



5.8.9.6 check_recv_sb ()

Figure 5-7 shows the flowchart of check_recv_sb ().

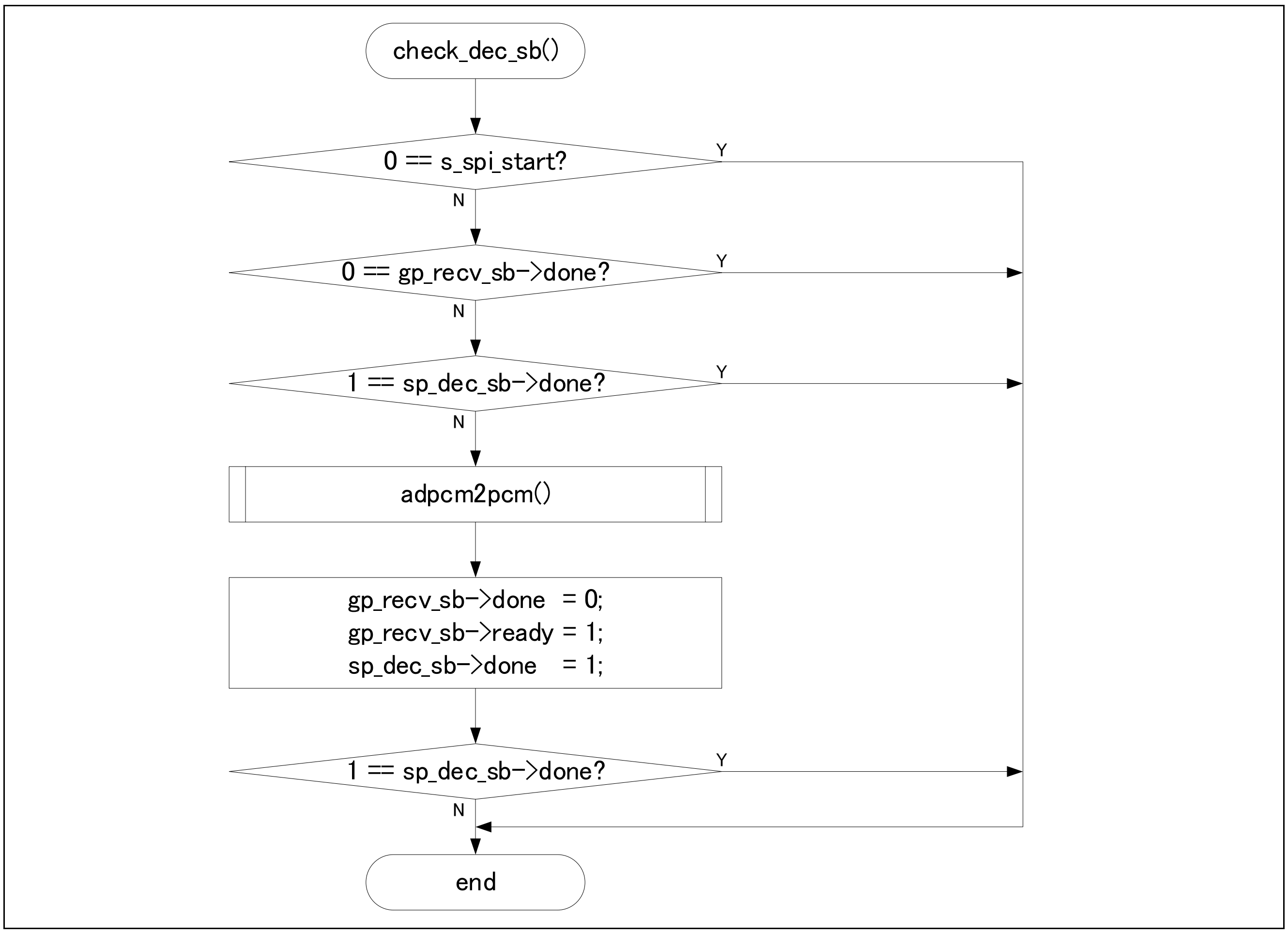
Figure 5-7 check_recv_sb ()



5.8.9.7 check_dec_sb ()

Figure 5-8 shows the flowchart of check_dec_sb ().

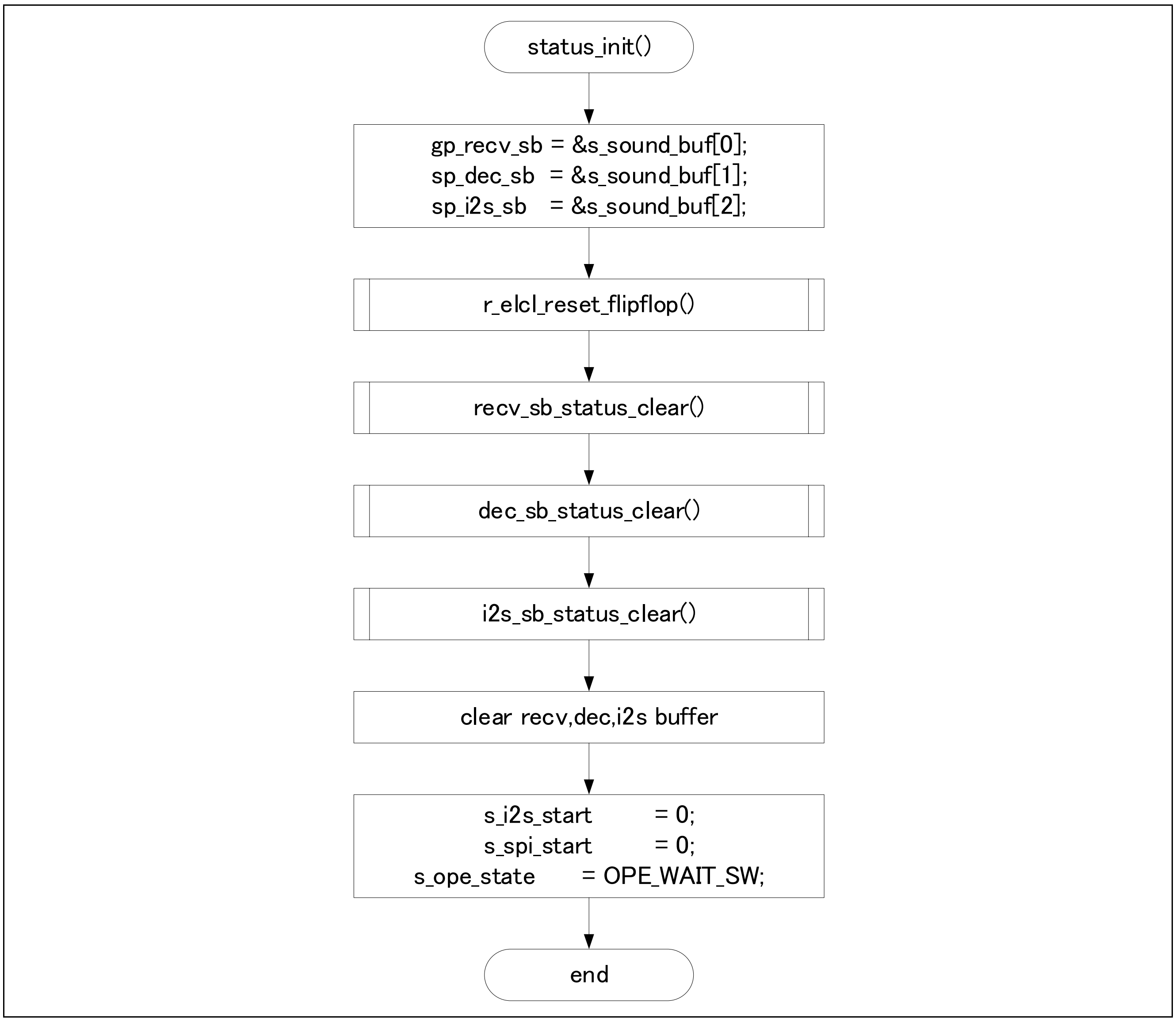
Figure 5-8 check_dec_sb ()



5.8.9.8 status_init ()

Figure 5-9 shows the flowchart of status_init ().

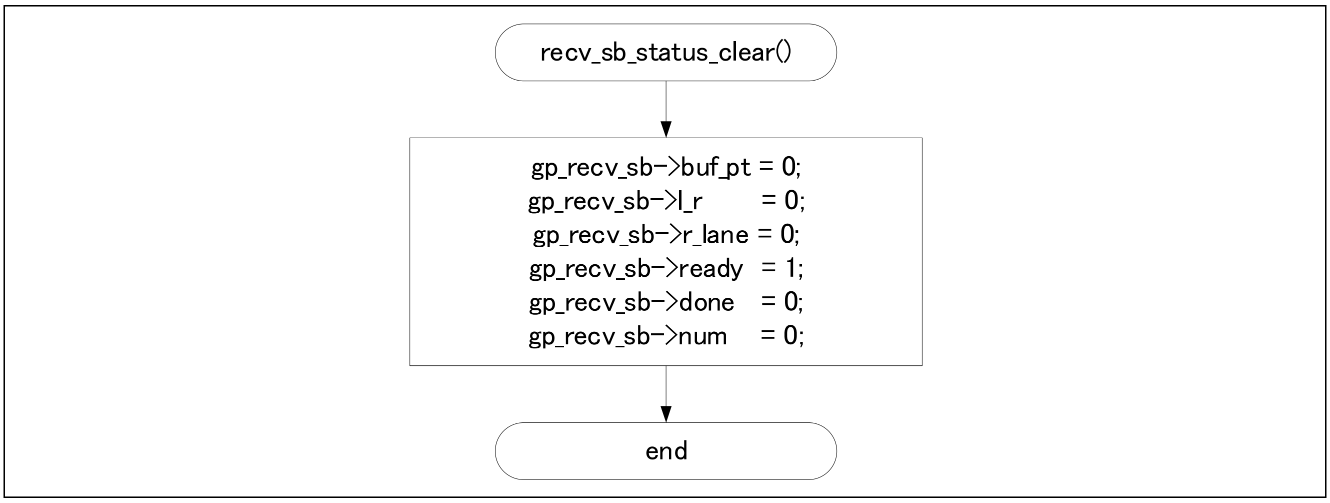
Figure 5-9 status_init ()


status_init()
gp_recv_sb = &s_sound_buf[0];
sp_dec_sb = &s_sound_buf[1];
sp_i2s_sb = &s_sound_buf[2];
r_elcl_reset_flipflop()
recv_sb_status_clear()
dec_sb_status_clear()
i2s_sb_status_clear()
clear recv,dec,i2s buffer
s_i2s_start = 0;
s_spi_start = 0;
s_ope_state = OPE_WAIT_SW;
end

5.8.9.9 recv_sb_status_clear ()

Figure 5-10 shows the flowchart of recv_sb_status_clear ().

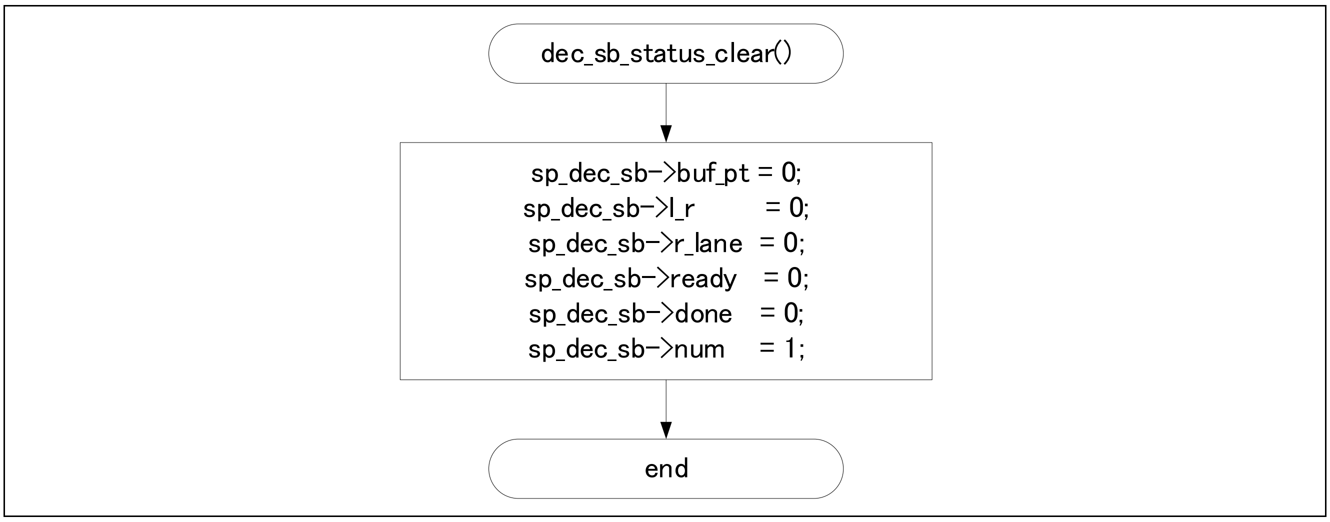
Figure 5-10 recv_sb_status_clear ()



5.8.9.10 dec_sb_status_clear ()

Figure 5-11 shows the flowchart of dec_sb_status_clear ().

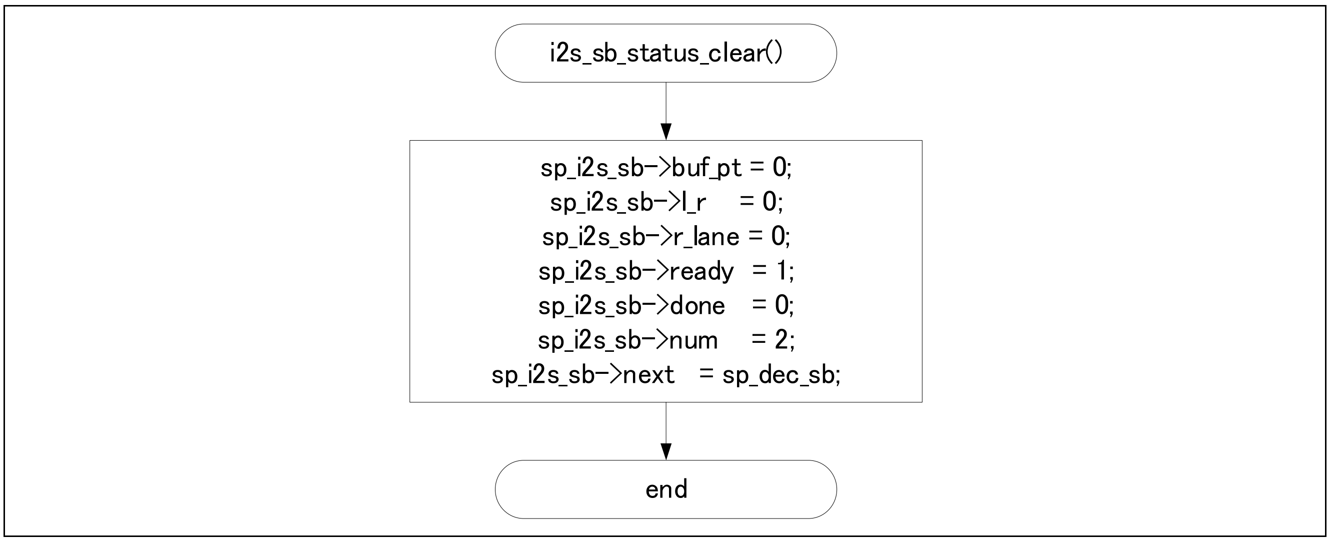
Figure 5-11 dec_sb_status_clear ()



5.8.9.11 i2s_sb_status_clear ()

Figure 5-12 shows the flowchart of i2s_sb_status_clear ().

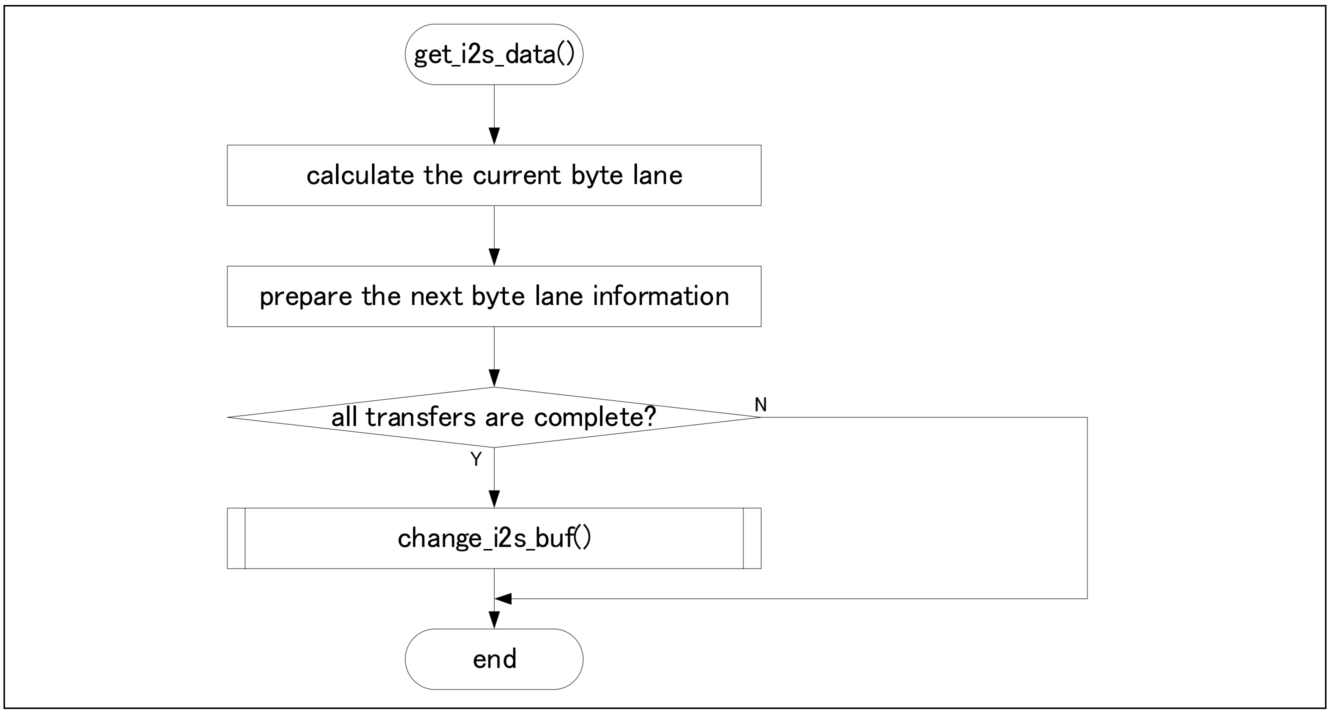
Figure 5-12 i2s_sb_status_clear ()



5.8.9.12 get_i2s_data ()

Figure 5-13 shows the flowchart of get_i2s_data ().

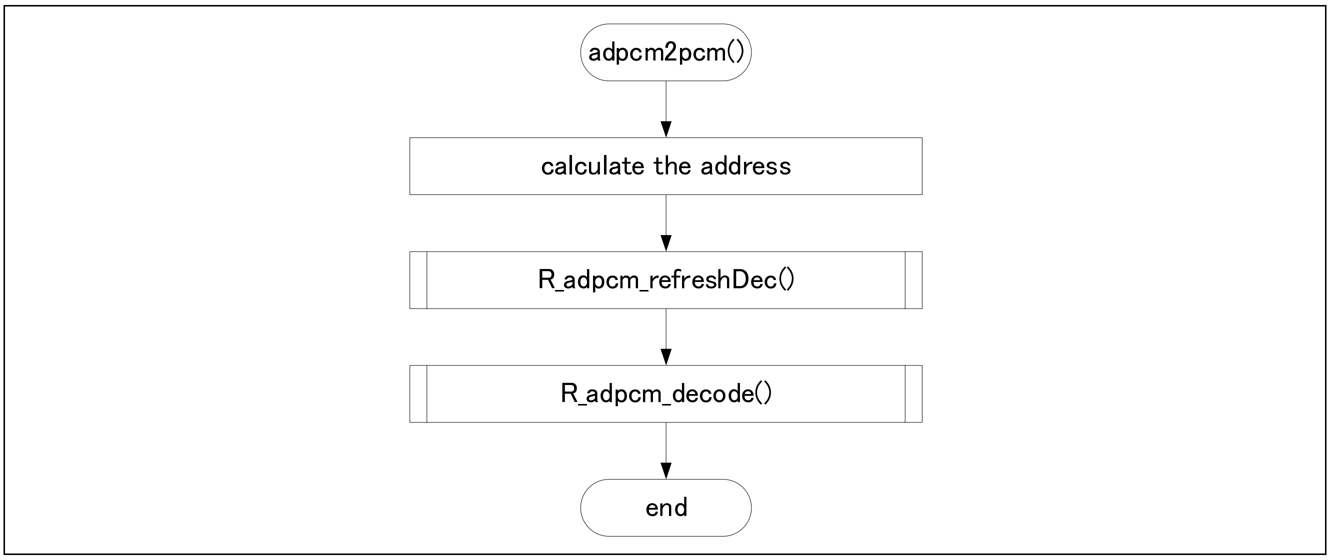
Figure 5-13 get_i2s_data ()


get_i2s_data()
calculate the current byte lane
prepare the next byte lane information
all transfers are complete?
N
Y
change_i2s_buf()
end

5.8.9.13 adpcm2pcm ()

Figure 5-14 shows the flowchart of adpcm2pcm ().

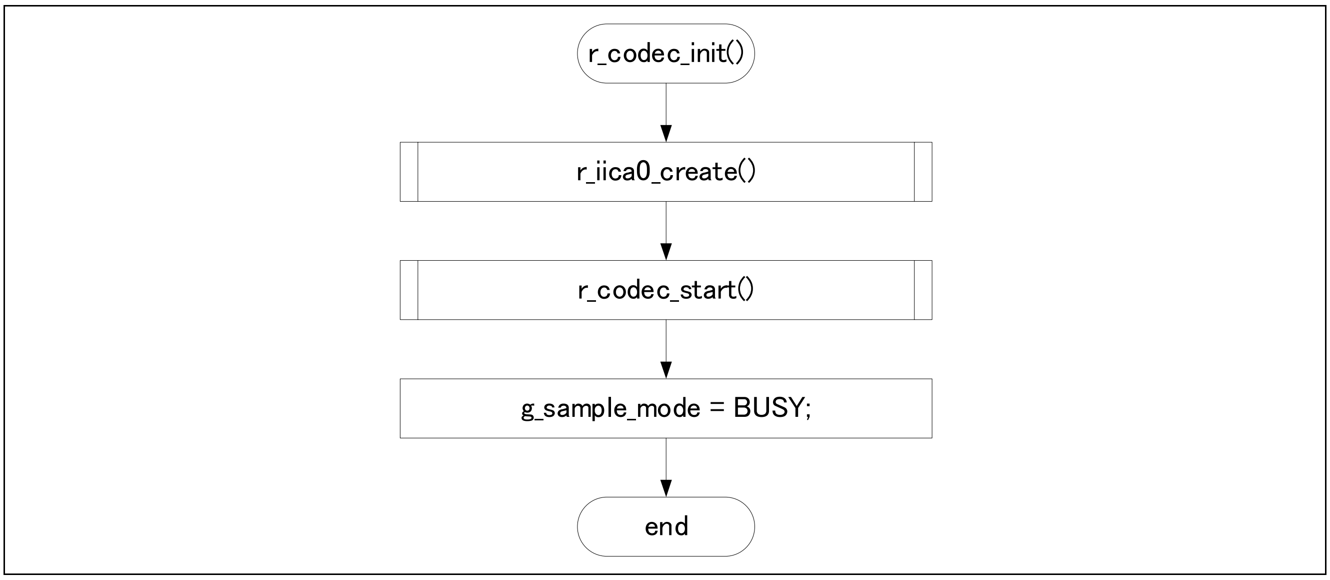
Figure 5-14 adpcm2pcm ()



5.8.9.14 r_codec_init ()

Figure 5-15 shows the flowchart of r_codec_init ().

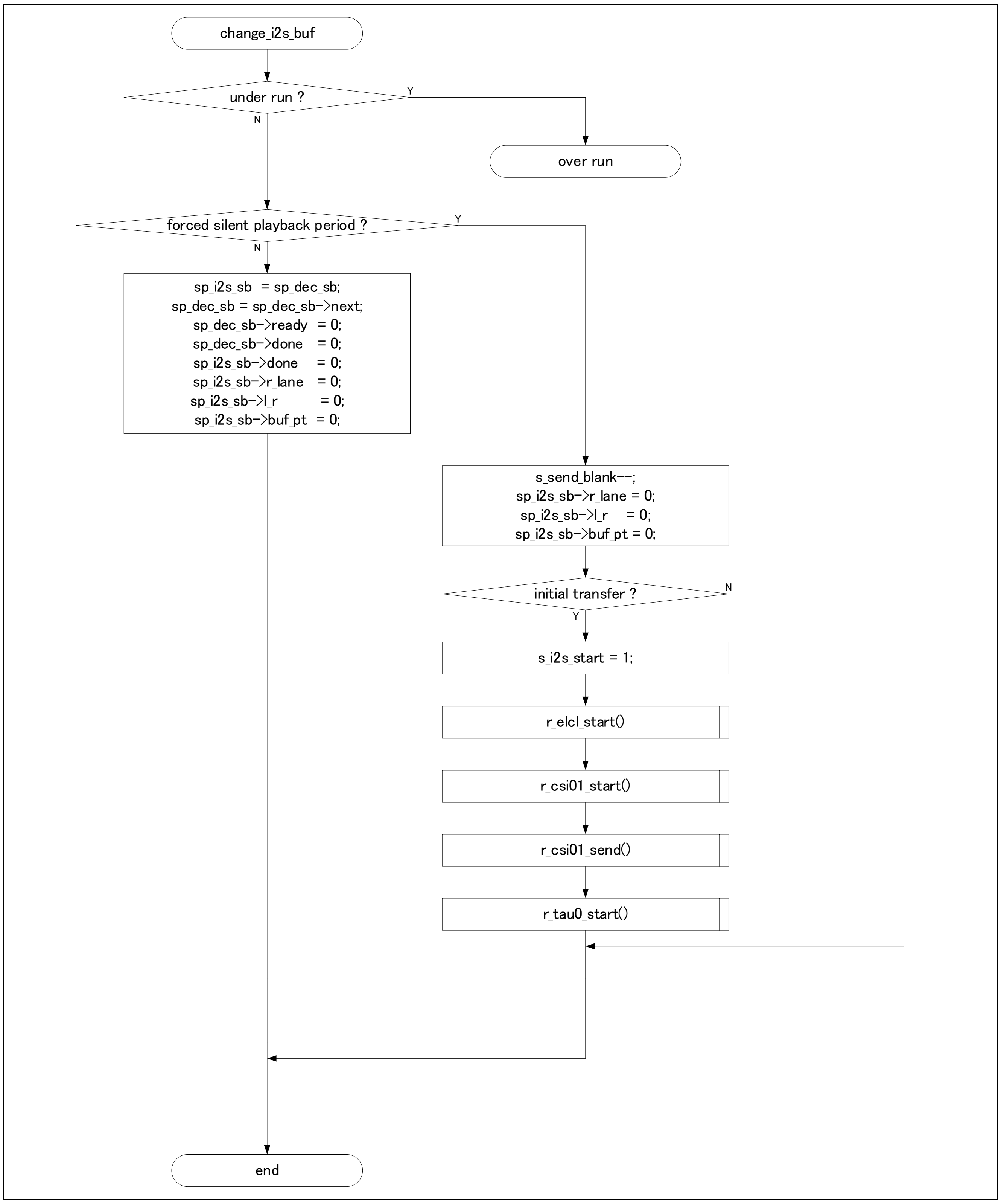
Figure 5-15 r_codec_init ()



5.8.9.15 change_i2s_buf ()

Figure 5-16 shows the flowchart of change_i2s_buf ().

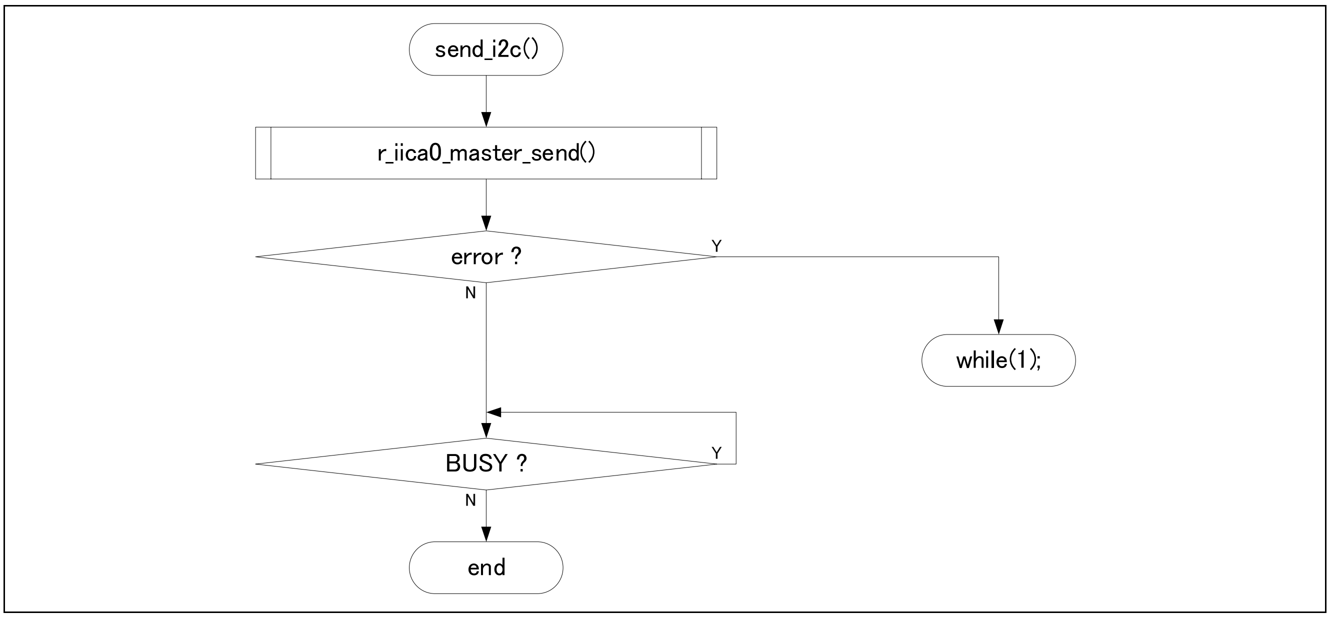
Figure 5-16 change_i2s_buf ()



5.8.9.16 send_i2c ()

Figure 5-17 shows the flowchart of send_i2c ().

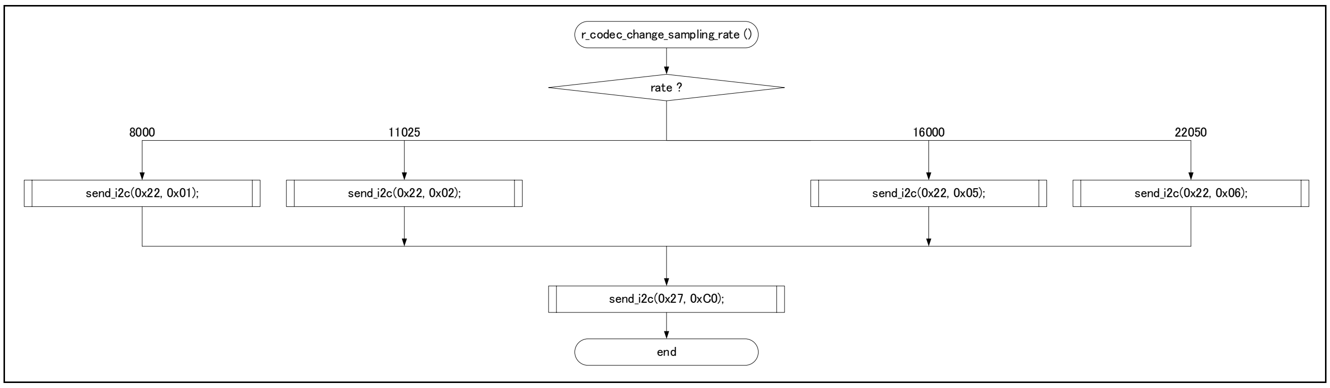
Figure 5-17 send_i2c ()



5.8.9.17 r_codec_change_sampling_rate ()

Figure 5-18 shows the flowchart of r_codec_change_sampling_rate ().

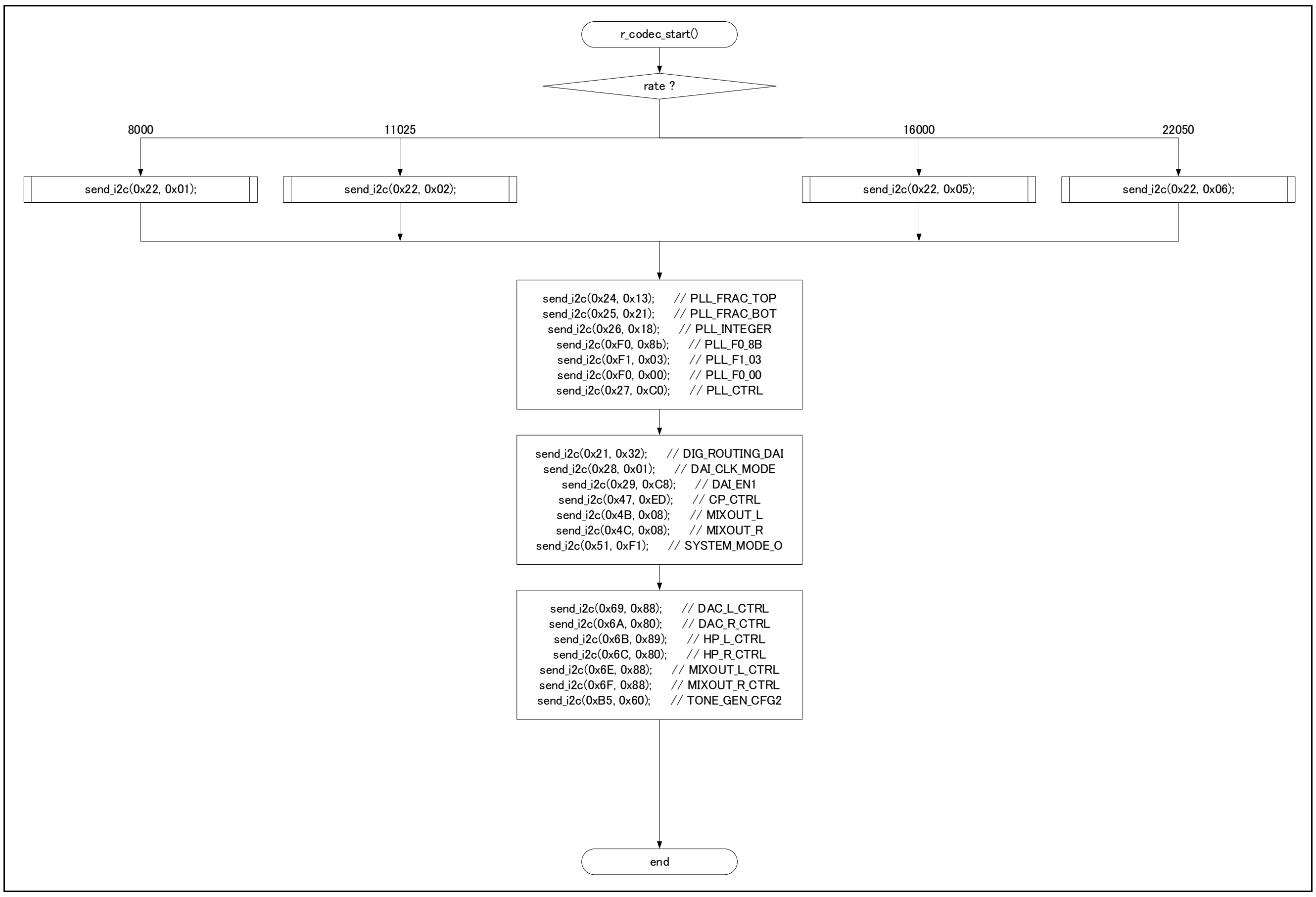
Figure 5-18 r_codec_change_sampling_rate ()



5.8.9.18 r_codec_start ()

Figure 5-19 shows the flowchart of r_codec_start ().

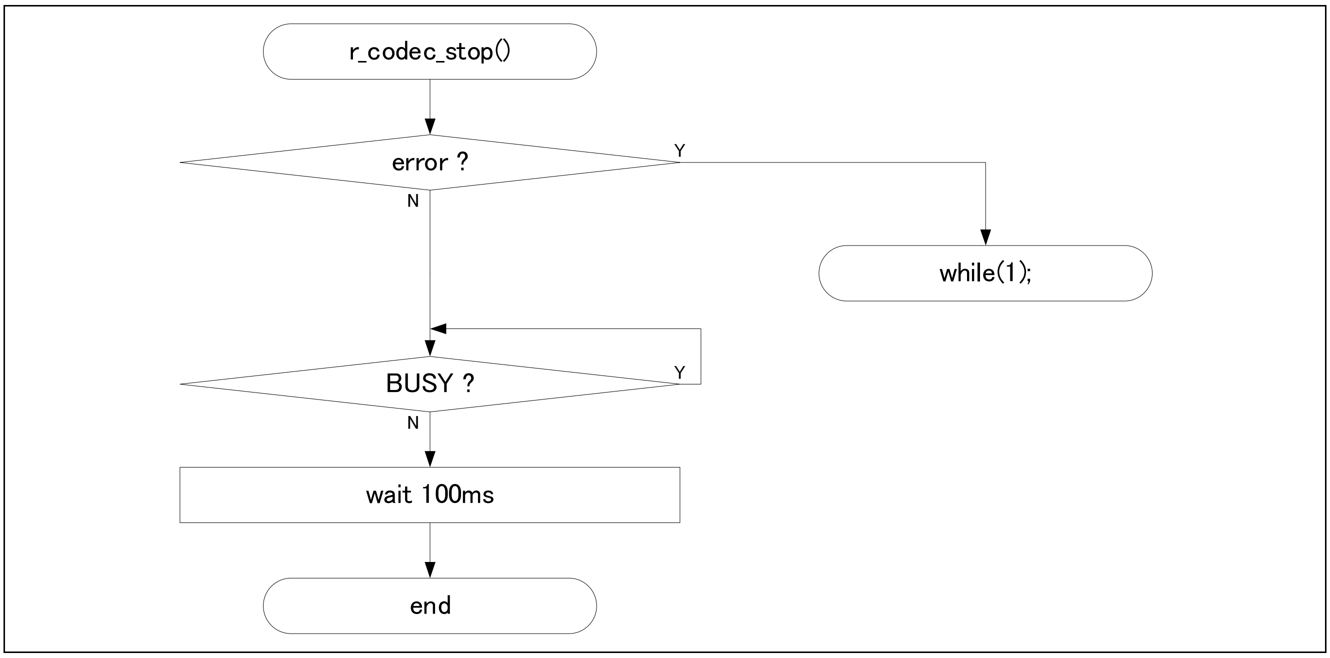
Figure 5-19 r_codec_start ()


r_codec_start()
rate ?
8000
11025
16000
22050
send_i2c(0x22, 0x01);
send_i2c(0x22, 0x02);
send_i2c(0x22, 0x05);
send_i2c(0x22, 0x06);
send_i2c(0x24, 0x13); // PLL_FRAC_TOP
send_i2c(0x25, 0x21); // PLL_FRAC_BOT
send_i2c(0x26, 0x18); // PLL_INTEGER
send_i2c(0xF0, 0x8b); // PLL_F0_8B
send_i2c(0xF1, 0x03); // PLL_F1_03
send_i2c(0xF0, 0x00); // PLL_F0_00
send_i2c(0x27, 0xC0); // PLL_CTRL
send_i2c(0x21, 0x32); // DIG_ROUTING_DAI
send_i2c(0x28, 0x01); // DAI_CLK_MODE
send_i2c(0x29, 0xC8); // DAI_EN1
send_i2c(0x47, 0xED); // CP_CTRL
send_i2c(0x4B, 0x08); // MIXOUT_L
send_i2c(0x4C, 0x08); // MIXOUT_R
send_i2c(0x51, 0xF1); // SYSTEM_MODE_O
send_i2c(0x69, 0x88); // DAC_L_CTRL
send_i2c(0x6A, 0x80); // DAC_R_CTRL
send_i2c(0x6B, 0x89); // HP_L_CTRL
send_i2c(0x6C, 0x80); // HP_R_CTRL
send_i2c(0x6E, 0x88); // MIXOUT_L_CTRL
send_i2c(0x6F, 0x88); // MIXOUT_R_CTRL
send_i2c(0xB5, 0x60); // TONE_GEN_CFG2
end

5.8.9.19 r_codec_stop ()

Figure 5-20 shows the flowchart of r_codec_stop ().

Figure 5-20 r_codec_stop ()



5.8.10 r01an7416_adpcm_player.scfg

This is the Smart Configurator configuration file used in the sample code. It contains all the features configured in the Smart Configurator. The sample code settings are as follows.

Table 5-8 Parameters of Smart Configurator

Tag name	Component	Contents
Clocks	-	Operation mod: High-speed main mode 2.4 (V) to 5.5 (V) EV_{DD} setting:1.8V≦EV_{DD0}＜5.5V High-speed on-chip oscillator: 32MHz f_{IHP}: 32MHz f_{CLK}: 32000kHz (High-speed on-chip oscillator) f_{SXP}: 32.768kHz (Low-speed on-chip oscillator)
System	-	On-chip debug operation setting: COM port Note 1 Pseudo-RRM/DMM function setting: Used Start/Stop function setting: Unused Trace function setting: Used Security ID setting: Use security ID Security ID: 0x00000000000000000000 Security ID authentication failure setting: Do not erase flash memory data
Components	r_bsp	Start up select: Enable (use BSP startup) Control of invalid memory access detection: Disable RAM guard space (GRAM0-1): Disabled Guard of control registers of port function (GPORT): Disabled Guard of registers of interrupt function (GINT): Disabled Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC): Disabled Data flash access control (DFLEN): Disables Initialization of peripheral functions by Code Generator/Smart Configurator: Enable API functions disable: Enable Parameter check enable: Enable Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode: High-speed Enable user warm start callback (PRE): Unused Enable user warm start callback (POST): Unused Watchdog Timer refresh enable: Unused
	Config_LVD0	Operation mode setting: Reset mode Voltage detection setting: Reset generation level (V_{LVD0}): 1.86 (V)
	Config_INTC	INTP0 Valid edge: Falling edge Priority: Level 3 (low)

Table 5-9 Parameters of Smart Configurator

Tag name	Component	Contents
Components	Config_TAU0_0	Components: Interval timer Operating mode: 16 bits count mode Resource: TAU0_0 Operation clock: CK00 Clock source: f_{CLK} Interval value:0.163us Generate INTTM00 interrupt at start of counting Interrupt setting: Unuse
	Config_TAU0_1	Components: External event count Resource: TAU0_1 Operation clock: CK00 Clock source: f_{CLK} Input source setting: ELCL Operation mode setting: 16-bit External event edge select (TI01): Rising edge Count value: 32 Interrupt setting: Unuse
	Config_TAU0_7	Components: Interval timer Operating mode: 16 bits count mode Resource: TAU0_7 Operation clock: CK01 Clock source: $f_{CLK}/2^8$ Interval value: 1ms Interrupt setting: use Priority: Level
	Config_Through	Components: ELCL Through Common setting: L3L0 Detail setting: L3L0 Input signal selector: ELISEL_5, TO00 Application: Through Output signal selector: P51

Table 5-10 Parameters of Smart Configurator

Tag name	Component	Contents
Components	Config_PCLBUZ0	Output clock: f_{MAIN}/2^3
	Config_PORT	Components: Port Port selection: PORT5 P53: Out (Output 1)
	Config_CSI20	Transfer clock mode: Internal clock (master) Operation clock:CK10 Clock source: f_{CLK} Transfer mode setting: Single transfer mode Data length setting: 8 bits Transfer direction setting: MSB Data transmission/reception timing setting: Type1 Baud rate: 1000000bps Transmit end interrupt priority (INTST0): Level 3(low) Callback function setting: Transmission end, reception end, Overrun error
	r_nor_flash	Parameter check: use system default ENABLE CHECKING OF THE WEL BIT.: Check WEL bit Select serial flash memory device: Renesas Electronics AT25QF Select serial flash memory capacity: 64M bit CS Port Number: PORT1 CS Bit Number: BIT7 Data transfer mode: CPU transfer (Software transfer) SPI(CSI) channel number: Channel 4 DTC Control Data Number: 0

5.8.10.1 Clocks

Set the clock used in the sample code. (For the settings, see Table 5-8.)

5.8.10.2 System

Set the on-chip debug of the sample code.

The settings of “On-chip debug operation setting” and “Security ID authentication failure setting” affect the “On-chip debugging is enabled” setting in Table 5-3 Option Byte Settings. Care must be taken when changing the settings.

5.8.10.3 r_bsp

Set the startup of the sample code.

5.8.10.4 Config_LVD0

Set the power management of the sample code.

This setting affects the LVD0 setting in Table 5-3Option Byte Settings. Care must be taken when changing the setting.

5.8.10.5 Config_INTC

Set INTP0 of the sample code.

In the sample code, INTP0 is used for determination on the user switch (SW1).

5.8.10.6 Config_TAU00

Set TAU00 of the sample code.

In the sample code, TAU00 is used as an interval timer to generate BCLK. The sampling frequency is automatically set in the sample program.

5.8.10.7 Config_TAU01

Set TAU01 of the sample code.

In the sample code, TAU01 is used as an external event count to generate LRCLK. The data size is fixed to 32 bits in the sample code.

5.8.10.8 Config_TAU07

Set TAU07 in the sample code.

The sample code uses it as a 1 ms interval timer.

5.8.10.9 Config_Through

Set to change the output terminal of BCLK in the sample code.

Set the output destination of TO00 used as BCLK to P51.

5.8.10.10 Config_PORT

Set the port of the sample code.

In the sample code, P53 is used to control LED1.

5.8.10.11 Config_CSI20

Used to control SPI for NOR FLASH.

5.8.10.12 r_nor_flash

This is middleware for NOR FLASH.

5.8.10.13 Config_PCLBUSZ0

Used to supply clocks to ARD-AUDIO-DA7212.

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference

RL78/G23 I^{2}S Communication with ELCL and SPI Application Note (R01AN6420）

Serial NOR Flash Memory Control Module Software Integration System (R01AN7243)

RL78 Family Sound Playback/Compression System (Original ADPCM Codec) M3S-S2-Tiny: Introduction Guide (R20AN0122)

RL78/G23 User’s Manual: Hardware (R01UH0896)

RL78 Family User’s Manual: Software (R01US0015)

RL78 Smart Configurator User’s Guide: CS+ (R20AN0580)

RL78 Smart Configurator User’s Guide: e2 studio (R20AN0579)

RL78 Smart Configurator User’s Guide: IAREW (R20AN0581)

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

Technical Update / Technical News

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.21.24	-	First edition

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Corporate Headquarters

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

Trademarks

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

Contact information

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