The following document contains information on Cypress products. Although the document is marked with the name "Spansion" and "Fujitsu", the company that originally developed the specification, Cypress will continue to offer these products to new and existing customers.

Continuity of Specifications
There is no change to this document as a result of offering the device as a Cypress product. Any changes that have been made are the result of normal document improvements and are noted in the document history page, where supported. Future revisions will occur when appropriate, and changes will be noted in a document history page.

Continuity of Ordering Part Numbers
Cypress continues to support existing part numbers. To order these products, please use only the Ordering Part Numbers listed in this document.

For More Information
Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress
Cypress (NASDAQ: CY) delivers high-performance, high-quality solutions at the heart of today's most advanced embedded systems, from automotive, industrial and networking platforms to highly interactive consumer and mobile devices. With a broad, differentiated product portfolio that includes NOR flash memories, F-RAM™ and SRAM, Traveo™ microcontrollers, the industry's only PSoC® programmable system-on-chip solutions, analog and PMIC Power Management ICs, CapSense® capacitive touch-sensing controllers, and Wireless BLE Bluetooth® Low-Energy and USB connectivity solutions, Cypress is committed to providing its customers worldwide with consistent innovation, best-in-class support and exceptional system value.

FUJITSU

AN706-00041-2v0-E

32-BIT MICROCONTROLLER
FM3 family Application Note

Simple AV System Solution

(JPEG, I^2S, MP3, AAC, USB)

Application Note

The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.

The information, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of Fujitsu semiconductor device; Fujitsu does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. Fujitsu assumes no liability for any damages whatsoever arising out of the use of the information.

Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of Fujitsu or any third party or does Fujitsu warrant non-infringement of any third-party's intellectual property right or other right by using such information. Fujitsu assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite).

Please note that Fujitsu will not be liable against you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products.

Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.

If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the prior authorization by Japanese government will be required for export of those products from Japan.

The company names and brand names herein are the trademarks or registered trademarks of their respective owners.

Revision History

Rev	Date	Remark
1.0	Aug.23,2011	First Edition
2.0	Feb.06,2012	Correction format Correction lineup of FM3 Correction by RoHS c compliant for board, parts change, and software change

Table of Contents

Target products

This application note is described about below products;

(TYPE0)

Series	Product Number (not included Package suffix)
MB9B500B	MB9BF504NB,MB9BF505NB,MB9BF506NB MB9BF504RB,MB9BF505RB,MB9BF506RB
MB9B300B	MB9BF304NB,MB9BF305NB,MB9BF306NB MB9BF304RB,MB9BF305RB,MB9BF306RB

1 INTRODUCTION

This application note is for those planning to design an image output processing system or process audio data using a microcontroller of the FM3 family made by Fujitsu Semiconductor. This document takes specific examples of a system or audio output control, LCD control and decoding of audio or image data contained in a USB memory, and gives the ROM and RAM size required by the FM3 family, and actual measurement results of CPU occupancy and system processing speed.

2 SIMPLE AV SYSTEM BOARD

The simple AV system described in the application notes conducts the following operations. For details, see the User Manual for the simple AV system board.

① Reads image/audio files in USB memory (JPEG, MP3, AAC)
② Switch detection
③ Image/audio file decoding (JPEG, MP3, AAC)
④ Image output (LCD display)
⑤ Touch panel control
⑥ Audio output

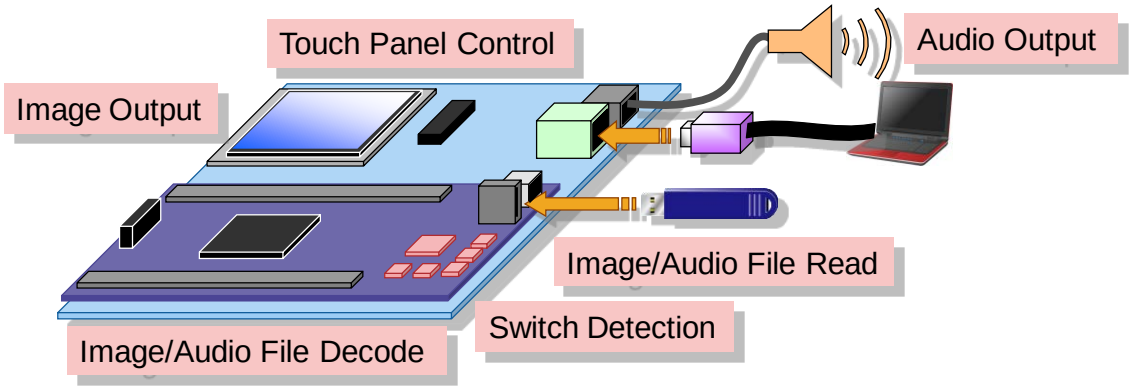


Figure 1 Simple AV System Board Schematic View

2.1 System Operation

◆Image data processing

(1) Reads JPEG files in USB memory connected to the USB interface.

(2) Decodes JPEG files.

(3) Outputs decoded JPEG files on the LCD.

* JPEG file processing is not implemented if audio data is AAC.
(See "3.3.1 Amount of ROM/RAM Used")

◆Audio data processing

(1) Reads audio data in USB memory connected to the USB interface.

(2) Decodes read audio files.

(3) Outputs decoded audio data to DA converter(DAC).

* Format of output audio data can be modified by middleware assembled. The application notes contain performance measurement results for MP3 and AAC file processing.

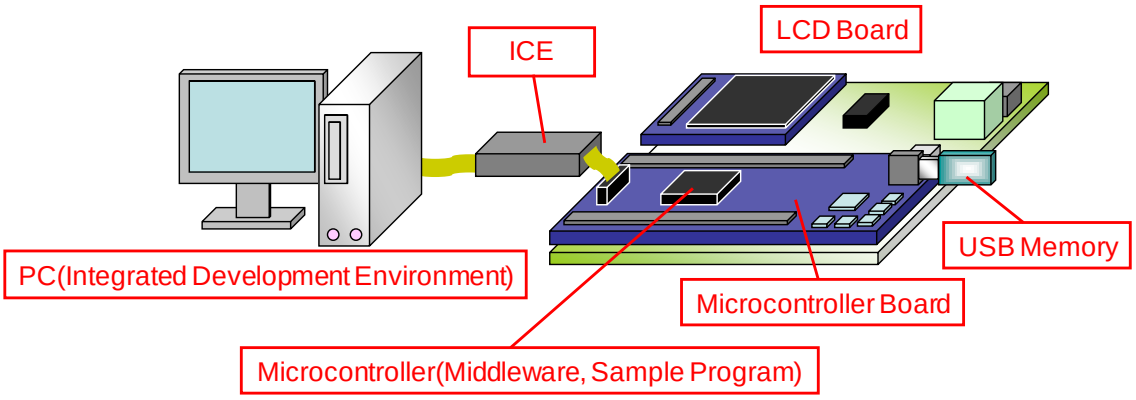


Figure 2 System Configuration Diagram

2.2 Hardware

2.2.1 External Appearance of Simple AV System Board

A photograph of the outer external appearance of the simple AV system board is shown in Figure 3.

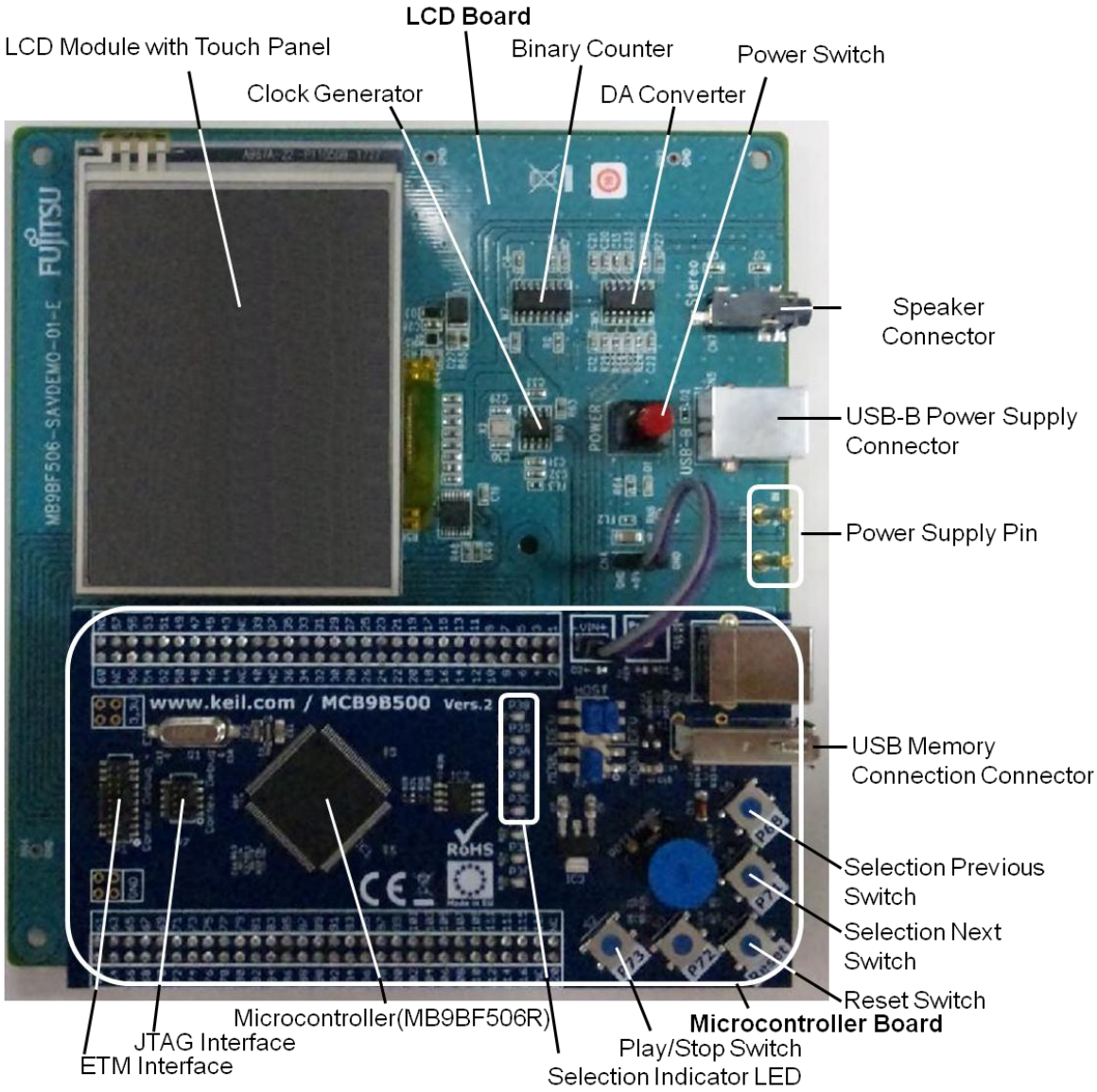


Figure 3 Photograph of External Appearance of Simple AV System Board

2.2.2 Hardware Block Diagram

The hardware block diagram of the system is shown in Figure 4.

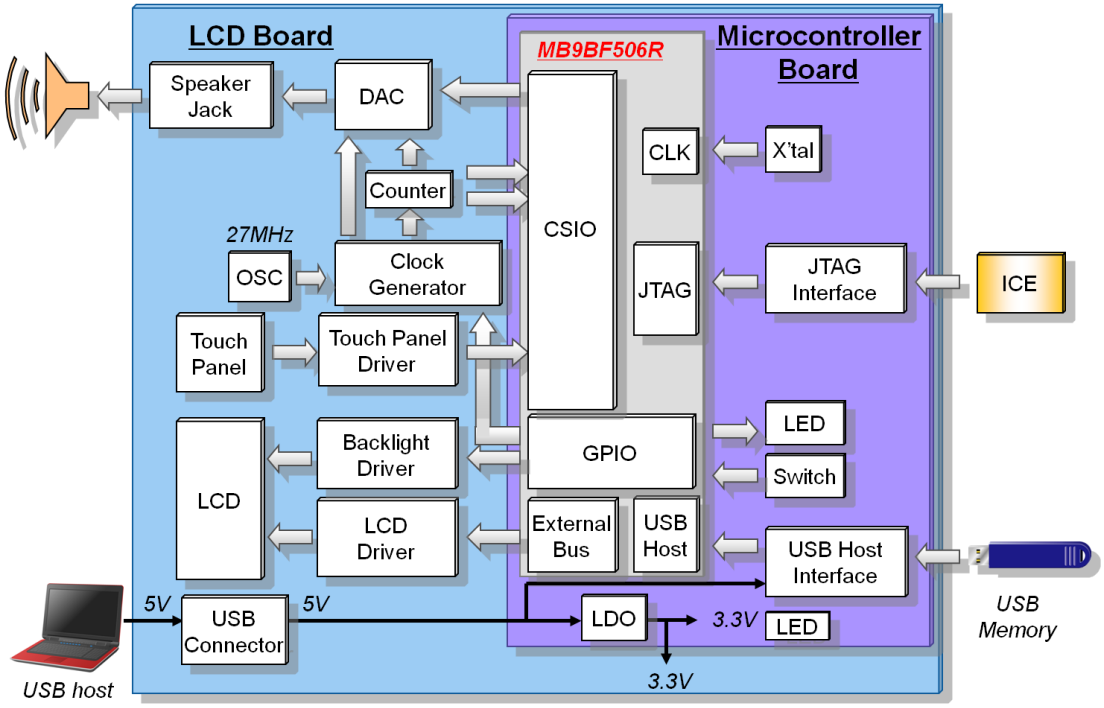


Figure 4 Hardware Block Diagram

2.3 Software

2.3.1 Software Block Diagram

The software block diagram of the system is shown in Figure 5.

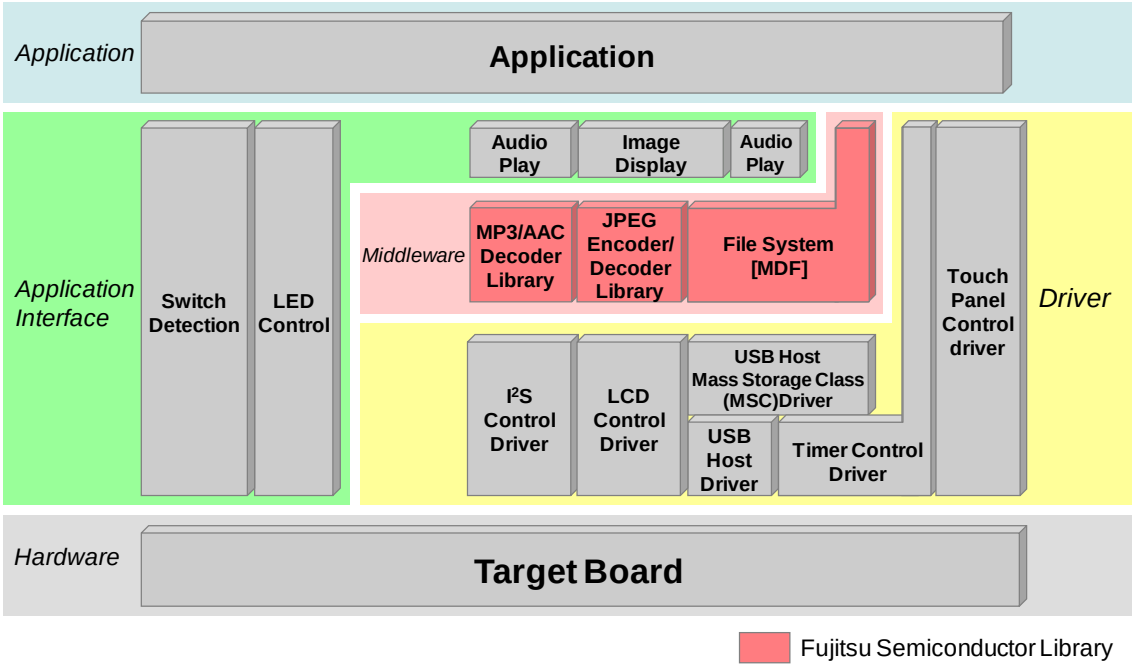


Figure 5 Software Block Diagram

2.3.2 Operation Flow of Entire Application

2.3.2.1 MP3

(1) The application operation flow with audio data playback stopped is as follows.

① USB MSC device connection/disconnection judgment is executed in the main loop.

② If a USB memory is connected, after reading the JPEG files from the USB memory and displaying the images for selection, switch pressing detection and touch panel detection are conducted.

③ If the play/stop switch is detected to be pressed down, or if not detected, but an area of the touch panel is detected to have been touched, the JPEG files corresponding to the selected MP3 file are read from the USB memory and displayed for playback. If the selection previous switch or selection next switch are detected to have been pressed down, MP3 file selection is shifted and LED control is executed.

④ The MP3 selected from the USB memory is then opened.

⑤ The MP3 file header is read, MP3 file header analysis processing is conducted and operation shifts to audio data playback in progress status.

This operation is shown in Figure 6.

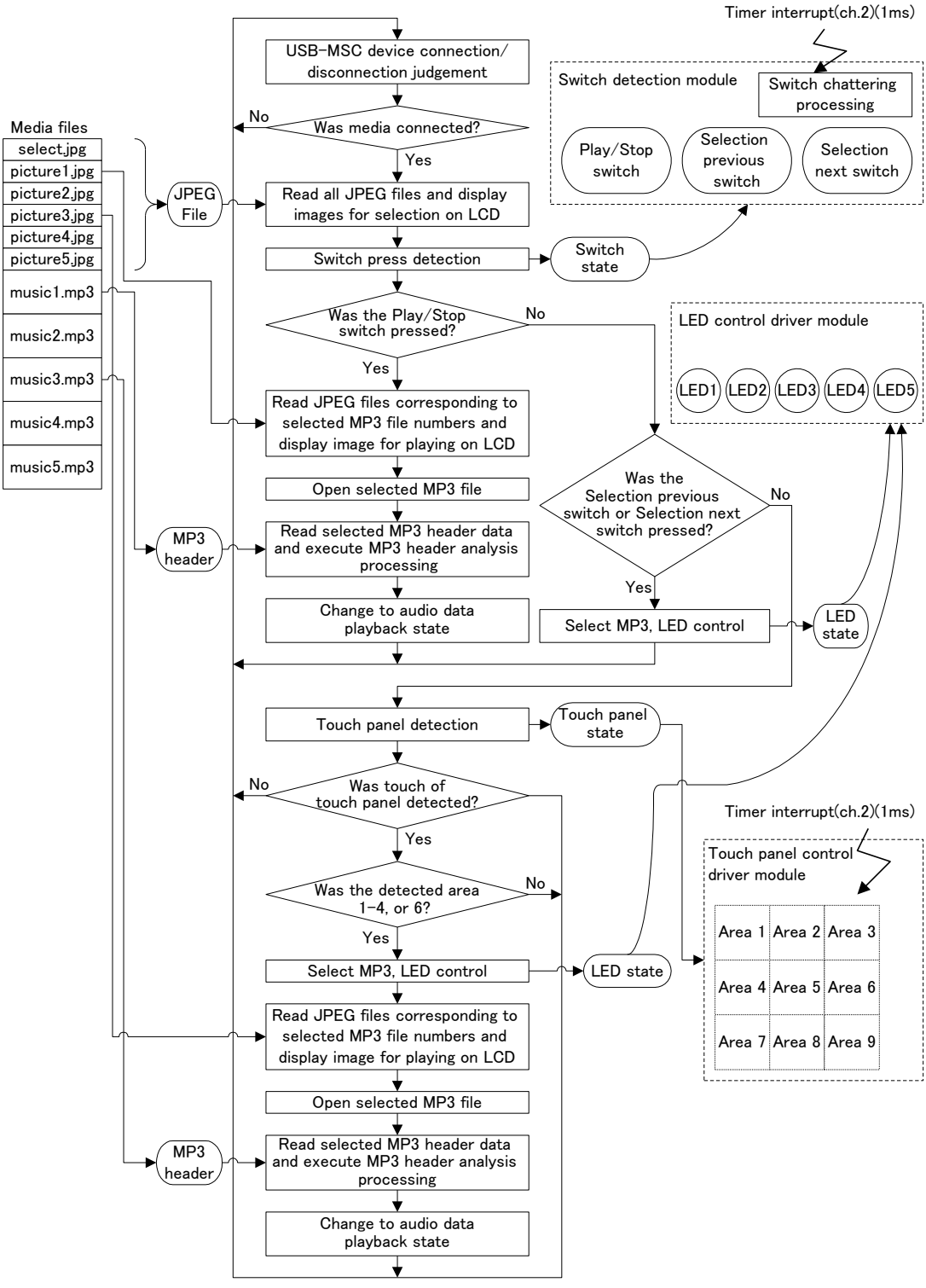


Figure 6 Application Operation Flow (Audio Playback Stopped Status, Case of MP3)

(2) The application operation flow with audio data playback in progress is as follows.

① USB MSC device connection/disconnection judgment is executed in the main loop.

② If the USB memory has been removed, stop playback, close the opened MP3 file, quit the file system and operation shifts to initialization status.

③ If the USB memory is connected, play/stop switch press down detection and touch panel touch detection are executed.

④ If the play/stop switch is detected to have been pressed down, or if not detected, but an area of the touch panel is detected to have been touched, playback is stopped, the MPA file is closed, all JPEF filed are read from the USB memory, that images for selection are displayed and operation shifts to audio data playback stopped status.

⑤ It is confirmed whether there is enough empty area in an input buffer.

⑥ If there is enough empty area, the MP3 file is read from the USB memory and copied in the input buffer.

⑦ One frame of the input buffer is decoded and stored in the RAW buffer.

⑧ When 1 frame had been decoded, the RAW buffer is up-sampled and buried in the output buffer.

⑨ With DMA ch2 interrupt, data is sent from the output buffer to I^2S in sequence.

This operation is shown in Figure 7.

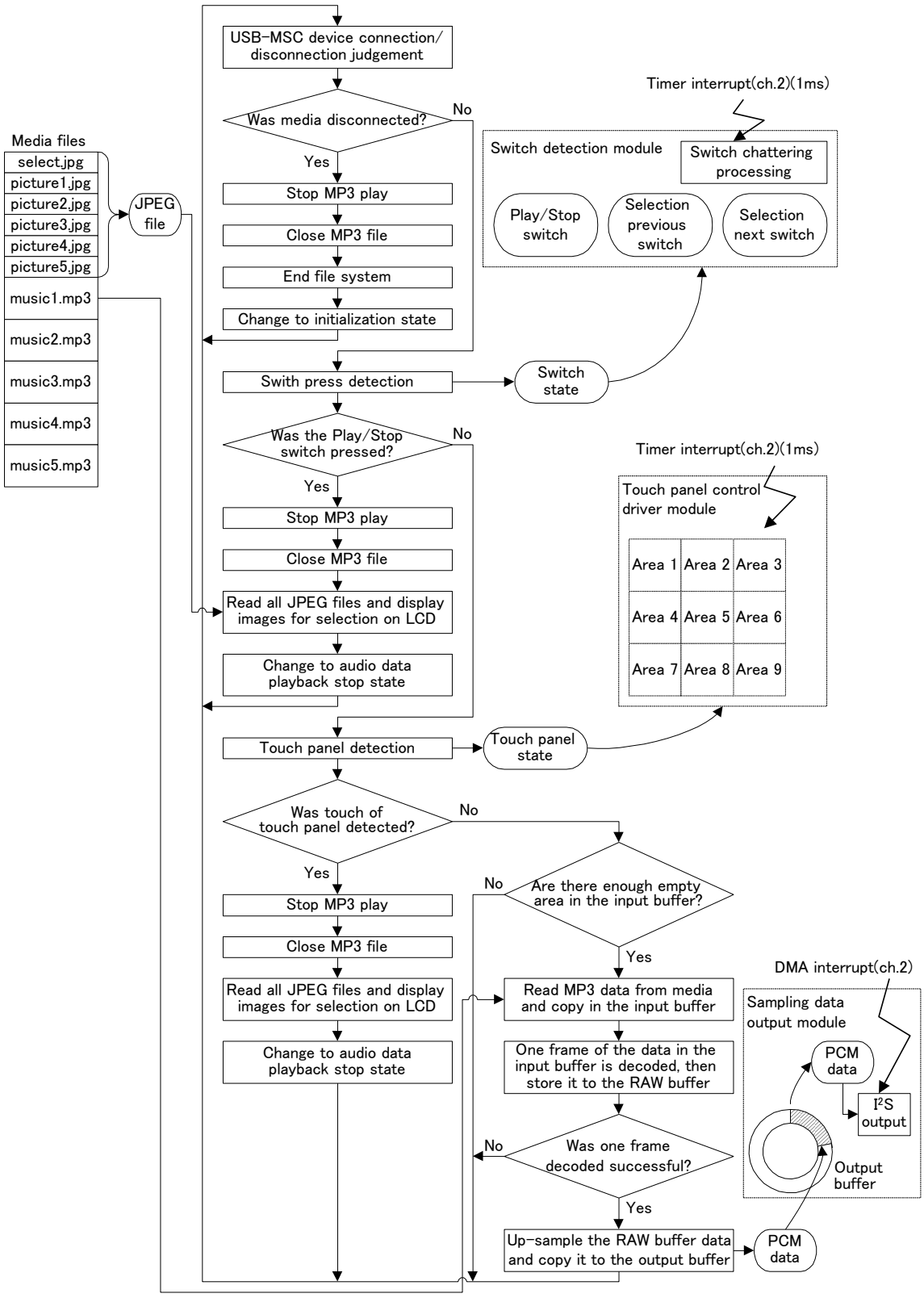


Figure 7 Application Operation Flow (Audio Playback Status, Case of MP3)

2.3.2.2 AAC

(1) The application operation flow with audio data playback stopped is as follows.

① USB MSC device connection/disconnection judgment is executed in the main loop.

② If the USB memory is connected, switch press-down detection is executed.

③ If the play/stop switch is detected to have been pressed down, the AAC file selected from the USB memory is opened. If the selection previous switch or selection next switch are detected to have been pressed down, AAC selection is shifted and LED control is executed.

④ The AAC file header is read, AAC file header analysis processing is conducted and operation shifts to audio data playback status.

This operation is shown in Figure 8.

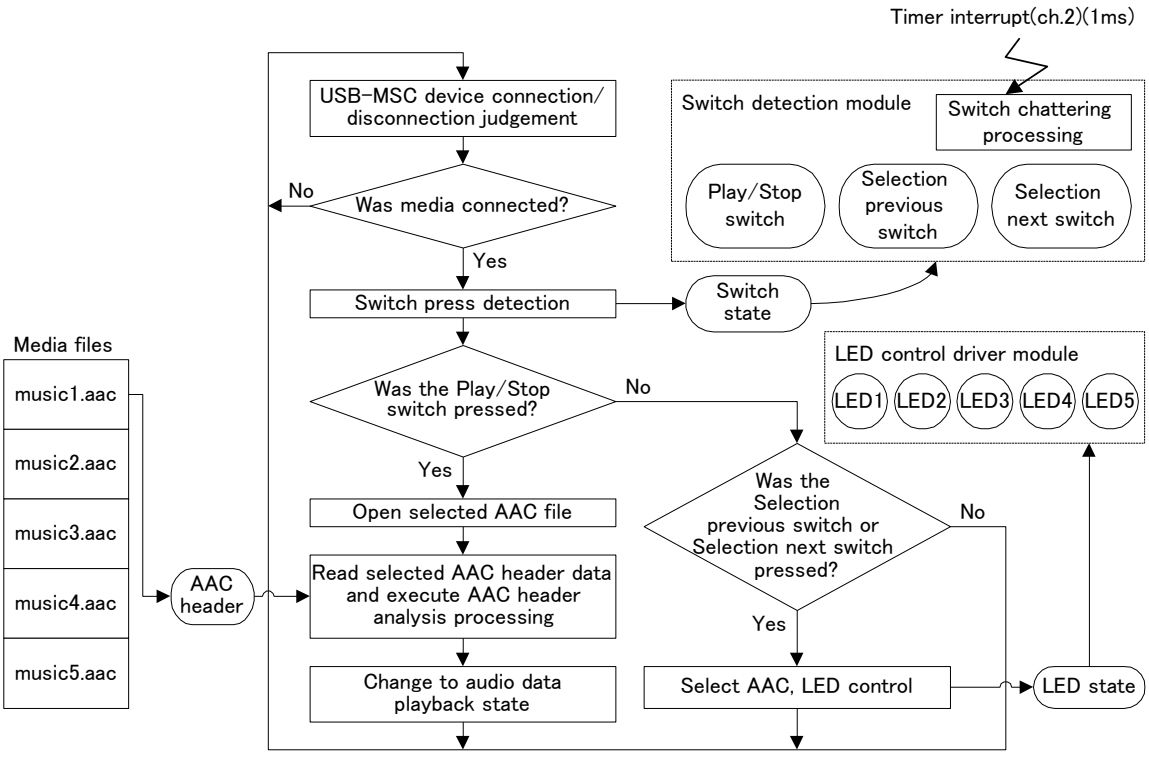


Figure 8 Application Operation Flow (Audio Playback Stopped Status, Case of AAC)

(2) The application operation flow with audio data playback in progress is as follows.

① USB MSC device connection/disconnection judgment is executed in the main loop.

② If the USB memory has been removed, stop playback, close the opened AAC file, quit the file system and operation shifts to initialization status.

③ If the USB memory is connected, play/stop button press-down detection is executed.

④ If the play/stop button is detected to have been pressed down, playback stops, the AAC file is closed, and operation shifts to audio data playback stop status.

⑤ It is confirmed whether there is enough empty area in an input buffer.

⑥ If there is enough empty area, the AAC file is read from the USB memory and copied in the input buffer.

⑦ One elementary stream of the input buffer is decoded and stored in the RAW buffer.

⑧ When 1 elementary stream had been decoded, the RAW buffer is up-sampled and buried in the output buffer.

⑨ With DMA ch2 interrupt, data is sent from the output buffer to I^2S in sequence.

This operation is shown in Figure 9.

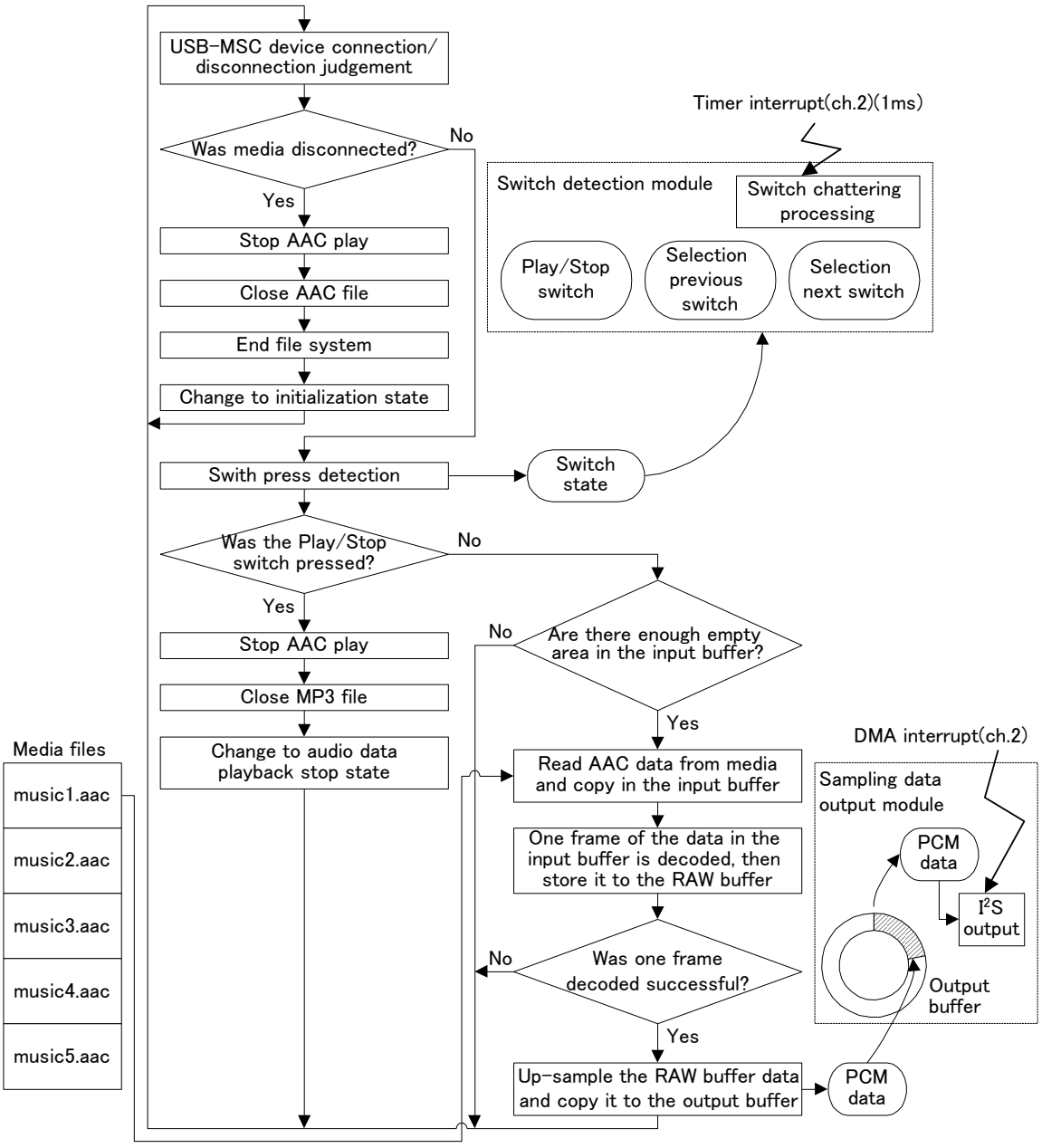


Figure 9 Application Operation Flow (Audio Playback Status, Case of AAC)

2.3.3 I²S Operation

With the simple AV solution, in order to transmit data by 44.1kHz or 48kHz sampling to DAC, MFS channels 4 and 5 are used as CSIO in the slave mode and the clock inputs 8 divisions of 11.2896MHz or 12.288MHz.

As for data, PCM data for which results of decoded MP3 or AAC file are output by up-sampling to 44.1kHz or 48kHz. (For up-sampling, see "3.2 Performance Measurement Items")

PCM data output to I²S is shown in Figure 10.

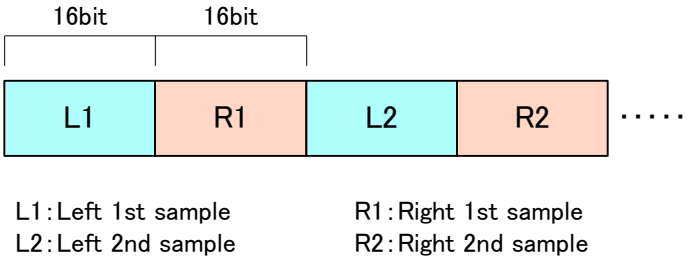


Figure 10 PCM Data Output to I²S

PCM data is synchronized and output to DAC from MFS channel 5, and left/right data is synchronized and output to DAC from channel 4. PCM data is transmitted to MFS using DMAC channel 2 and left/right data using DMAC channel 3.And left/right data is set 1bit earlier than PCM data.I/O format of I²S is shown in Figure 11.

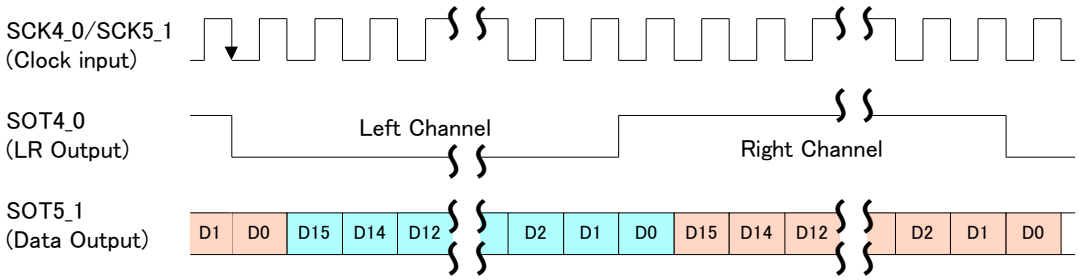


Figure 11 I²S I/O Format

Because PCM data of 1 sample output to DAC is 16-bit Stereos (2ch), it is

(11.2896[MHz]/8) / 16[bit] / 2[ch] = 44100[Hz]

(12.288 [MHz] /8) / 16 [bit] / 2 [ch] = 48000 [Hz],

and is therefore played at the sampling rate of 44.1kHz or 48kHz.

For details concerning the MFS and DMAC usage method, see the "FM3 32 bit Microcontroller MB9Axxx/MB9Bxxx Series Peripheral Manual".

2.3.3.1 I²S Driver API

This section provides a brief description of the I²S driver's API.

Function	void I2S_Init (void)	
Overview	Initializes I²S driver. Set MFS initial settings (CSIO settings) and DMAC initial settings.	
Parameter	None	
Return value	None	

Function	STATUS I2S_Start (uint8_t audio_sample_rate)	
Overview	Starts I²S operation. Set data for DMAC and activates CSIO. Be sure to initialize by I2S_Init function before function is invoked.	
Parameter	audio sample_rate	Sampling rate AUDIO_SAMPLE_44100 44.1kHz AUDIO_SAMPLE_48000 48kHz
Return value	I2S_RET_OK I2S_RET_ILLEGAL_ERROR	Successful Not yet initialized, operating or stopped

Function	STATUS I2S_Stop (void)	
Overview	Completes I²S operation. Stop CSIO.	
Parameter	None	
Return value	I2S_RET_OK I2S_RET_ILLEGAL_ERROR	Successful Not yet initialized or stopped

Function	uint8_t I2S_Get_Status(void)	
Overview	Gets I²S operating status.	
Parameter	None	
Return value	I2S_STATUS_INIT I2S_STATUS_START I2S_STATUS_STOP	Initialization status (I²S can be started) Start (I²S operating) status Stop (I²S cannot be started) status

2.3.3.2 Usage Example I²S Driver API

The source codes for places using the I²S driver API are shown in Figure 12, Figure 13 and Figure 14. The figure gives the AUDIO_PlayTask for audio.c, AUDIO_DecodeStop function used in the AUDIO_PlayTask function and AUDIO_Init function used in AUDIO_DecodeStop. See the sample program for audio.c.

```
void AUDIO_PlayTask(void)
{
        (An omission)
    i2s_status = I2S_Get_Status();

    /* check I2S status,if I2S stopped,change audio task to stop */
    if (i2s_status == I2S_STATUS_STOP)
    {
        if (( s_AudioPlayStage != AUDIO_STAGE_STOP) &&
            (s_AudioPlayStage != AUDIO_STAGE_INIT))
        {
            AUDIO_SetAudioStage(AUDIO_STAGE_STOP);
        }
    }

    /* stop loop until decode one frame or stop(some error happened) */
    while (decode_loop == TRUE)
    {
        switch (s_AudioPlayStage)
        {
            (An omission)
            case AUDIO_STAGE_UPSAMPLE:
            (An omission)
                    /* the audio playing not been started */
                    if (s_AudioStartFlg == FALSE)
                    {
                        /* start I2S */
                        if ((s_AudioSampleRate == SAMPLING_RATE_8000HZ)  ||
                            (s_AudioSampleRate == SAMPLING_RATE_12000HZ) ||
                            (s_AudioSampleRate == SAMPLING_RATE_16000HZ) ||
                            (s_AudioSampleRate == SAMPLING_RATE_24000HZ) ||
                            (s_AudioSampleRate == SAMPLING_RATE_32000HZ) ||
                            (s_AudioSampleRate == SAMPLING_RATE_48000HZ)) {
                            I2S_Start(AUDIO_SAMPLE_48000);
                        } else {
                            I2S_Start(AUDIO_SAMPLE_44100);
                        }
                        s_AudioStartFlg = TRUE;
                    }
            (An omission)
                break;

            case AUDIO_STAGE_STOP:
                /* stop playing audio */
                AUDIO_DecodeStop();
            (An omission)
        }
    }

    return;
}
```

Gets I²S operating status

Starts I²S operation(48kHz)

Starts I²S operation(44.1kHz)

Please refer to AUDIO_DecodeStop function

Figure 12 AUDIO_PlayTask Function Source Code (Excerpt)

```
void AUDIO_DecodeStop(void)
{
    /* stop I2S operation */
    (void)I2S_Stop();
    /* free the memory for library context */
    if (s_LibContext != NULL)
    {
        free(s_LibContext);
        s_LibContext = NULL;
    }

    /* close file */
    FS_close(s_AudioPlayFileNo);

    /* AUDIO module initialize */
    AUDIO_Init();

    return;
}
```

Stops I²S operation

Please refer to AUDIO_Init function

Figure 13 AUDIO_DecodeStop Function Source Code

```
void AUDIO_Init(void)
{
   (An omission)
    /* initialize I2S */
    I2S_Init();
    /* up-sampling initialize */
    UPSAMPLE_Init();

    return;
}
```

Initializes I²S driver

Figure 14 AUDIO_Init Function Source Code (Excerpt)

3 PERFORMANCE

3.1 Performance Measurement Environment

The performance measurement environment is given in Table 1.

Table 1 Performance Measurement Environment

No.	Part Number	Description	Manufacturer	Remarks (Ver. No., etc.)
1	ICE	ULINK2	KEIL	
2	Integrated development environment	µVision V4.20.03.0	KEIL	
3	Microcontroller board	MCB9B500 Vers.2	KEIL	
4	LCD board	—	Fujitsu Semiconductor	
5	Middleware	Multi Device File Access Library V03L01(object for small MCU)	Fujitsu Semiconductor	V03L01 (*1)
6	Middleware	MP3 Decoder Library for FM3 V01 Evaluation	Fujitsu Semiconductor	V01L01 (*1)
7	Middleware	MPEG-4/2 AAC LC Decoder Library (2ch) for FM3 V01 Evaluation	Fujitsu Semiconductor	V01L01 (*1)
8	Middleware	JPEG Baseline Process Encoder/Decoder Library for FM3 V01 Evaluation	Fujitsu Semiconductor	V01L01 (*1)
9	Sample program	—	Fujitsu Semiconductor	
10	USB memory	— (equipment that supports USB Full Speed)	—	

(*1) Middleware is for evaluation.
A separate contract is required for usage.

3.2 Performance Measurement Items

Performance measurement items are given in Table 2.

Table 2 Performance Measurement Items

No.	Performance Measurement Items	Condition 1	Condition 2	Remarks
1	Amount of ROM/RAM used	MP3 decoder assembly (*2)	JPEG decoder assembly, screen display assembly	Compile conditions: No optimization
2		AAC decoder assembly (*2)	No JPEG decoder assembly, no screen display assembly	Compile conditions: No optimization
3	Processing time (*1)	MP3 decoder assembly (*2)	MP3 Decode	1 frame decode time
4			Up-sampling (*3)	Set to 1 unit (*4) I^2S output buffer
5			JPEG decode	1 scan decode time
6			LCD display	1 JPEG data display time
7			USB data read	JPEG/MP3 file
8		AAC decoder assembly (*2)	AAC Decode	1ES (*5) decode time
9			Up-sampling (*3)	Set to 1 unit (*4) I^2S output buffer
10			USB data read	AAC file
11		Common	MSC (*6) connection detection	Time from USB connection to MSC (*6) connection detection
12	CPU occupancy	MP3 play in progress	Sampling rate: 24kHz Bit rate: 64kbps	Channel mode: Joint Stereo
13			Sampling rate: 32kHz Bit rate: 96kbps	Channel mode: Joint Stereo
14			Sampling rate: 48kHz Bit rate: 128kbps	Channel mode: Joint Stereo
15		AAC play in progress	Sampling rate: 24kHz Bit rate: 64kbps	Channel mode: Stereo
16			Sampling rate: 32kHz	Channel mode: Stereo

No.	Performance Measurement Items	Condition 1	Condition 2	Remarks
			Bit rate: 96kbps	
17			Sampling rate: 48kHz Bit rate: 128kbps	Channel mode: Stereo

(*1) Main operating frequency conditions: 80MHz, APB1/APB2/APB3:40MHz

(*2) MP3 decoder and AAC decoder cannot be assembled simultaneously.

(*3) Because sampling frequency for audio data input to DAC is fixed at 44.1kHz or 48kHz, the following up-sampling processing is used.

8/12/16/24/32KHz => 48kHz

11.025/22.05kHz => 44.1kHz

(*4) 1 unit is as follows according to sampling rate.

8,16,32(kHz)：3072byte

11.25,12,22.05,24,44.1,48(kHz)：2048byte

(*5) ES: Elementary Stream

(*6) MSC: Mass Storage Class

3.3 Performance Measurement Results

3.3.1 Amount of ROM/RAM Used

The amount of ROM and RAM used when the decoding file is MP3 and when it is AAC are given in Table 3 and Table 4 respectively.

Table 3 Amount of ROM/RAM Used for MP3 Decoder Assembly

Unit: Bytes

	ROM	RAM		
		Variable	Stack	Heap
MP3 Decoder	34403	0	364	19148
JPEG Encoder/Decoder	7914	252	124	0
MDF (*1)	21362	6588	2344	204
USB host driver	11804	268	376	2048
USB host MSC driver	4070	92	920	0
Application	2191	29	6360	0
Audio playback	19660	16979	6360	0
Screen display	6904	8301	2368	2048
Others	7380	259	64	0
Total	***115688***	***32768***	***(Secured area) 8192***	***(Secured area) 24576***
		65536		

Amount of ROM used : 113.0 Kbytes (*2)
Amount of RAM used : 64.0 Kbytes (*2)
Stack : Area size secured on system given in total
Max. usage size given for each item
Heap : Area size secured on system given in total

(*1) Multi Device File Access Library
(*2) 1Kbyte = 1024byte

Table 4 Amount of ROM/RAM Used for AAC Decoder Assembly

Unit: Bytes

	ROM	RAM		
		Variable	Stack	Heap
AAC Decoder	90364	168	1104	29388
JPEG Encoder/Decoder (*1)	—	—	—	—
MDF (*2)	21362	6588	1536	204
USB host driver	11804	268	376	2048
USB host MSC driver	4070	92	920	0
Application	1603	21	6360	0
Audio playback	19652	17511	6360	0
Screen display (*1)	—	—	—	—
Others	7413	256	64	0
Total	***156268***	***24904***	***(Secured area) 8192***	***(Secured area) 32256***
		65360		

Amount of ROM used : 152.6 Kbytes (*3)

Amount of RAM used : 63.8 Kbytes (*3)

Stack : Area size secured on system given in total
Max. usage size given for each item

Heap : Area size secured on system given in total

(*1) Image display impossible because screen display requires 8K of RAM.

(*2) Multi Device File Access Library

(*3) 1Kbyte = 1024byte

3.3.2 Processing Time

Processing time measurement results are given in Table 5.

Table 5 Processing time

No.	Condition 1	Condition 2	Condition 3	Processing time
1	MP3 decoder assembly	MP3 Decode	Sampling rate: 24kHz Bit rate: 64kbps Channel mode: Joint Stereo	8.7ms
2			Sampling rate: 32kHz Bit rate: 96kbps Channel mode: Joint Stereo	18.2ms
3			Sampling rate: 48kHz Bit rate: 128kbps Channel mode: Joint Stereo	16.5ms
4		Up-sampling	Sampling rate: 24kHz Bit rate: 64kbps Channel mode: Joint Stereo	3.10ms
5			Sampling rate: 32kHz Bit rate: 96kbps Channel mode: Joint Stereo	1.75ms
6			Sampling rate: 48kHz Bit rate: 128kbps Channel mode: Joint Stereo	1.90ms
7		JPEG decode	Capacity 26813 bytes, YUV 4:2:0	348ms
8			Capacity 93265 bytes, YUV 4:4:4	860ms
9		LCD display	Capacity 26813 bytes, YUV 4:2:0	396ms
10			Capacity 93265 bytes, YUV 4:4:4	920ms
11		USB data read	JPEG file (*1)	4.04ms
12			MP3 file (*2)	2.02ms

No.	Condition 1	Condition 2	Condition 3	Processing time
13	AAC decoder assembly	AAC Decode	Sampling rate: 24kHz Bit rate: 64kbps Channel mode: Stereo	17.2ms
14			Sampling rate: 32kHz Bit rate: 96kbps Channel mode: Stereo	16.4ms
15			Sampling rate: 48kHz Bit rate: 128kbps Channel mode: Stereo	14.5ms
16		Up-sampling	Sampling rate: 24kHz Bit rate: 64kbps Channel mode: Stereo	1.35ms
17			Sampling rate: 32kHz Bit rate: 96kbps Channel mode: Stereo	2.3ms
18			Sampling rate: 48kHz Bit rate: 128kbps Channel mode: Stereo	1.72ms
19		USB data read	AAC file (*2)	2.02ms
20	Common	MSC connection detection	—	526ms

(*1) 2048 bytes read

(*2) 1024 bytes read

Measurement location for processing time are as follows.

① MP3 decode, AAC decode

From before to after place where audio_decode_decoding called

② Up-sampling

From before to after place where audio_decode_upsample called

③ JPEG decode

From before to after place where JPEGDecodeScan function and JPEGGetScanHeader in image_scan_decode function called

④ LCD display

From before to after place where IMAGE_Show function called

⑤ USB data read

JPEG file: From before to after place where image_code_input function called

MP3/AAC file: From before to after place where audio_decode_read function called

⑥ MSC connection detection

From place immediately preceding USB connection detection status is set in av_demoapp_usb_event_callback function to place following completion of internal processing in av_demoapp_msc_connect_callback function.

See sample program for location of audio_decode_decoding function, audio_decode_upsample function, JPEGGetScanHeader function, JPEGDecodeScan function, IMAGE_Show function, image_code_input function, audio_decode_read function invocation, av_demoapp_usb_event_callback function, and av_demoapp_msc_connect_callback function.

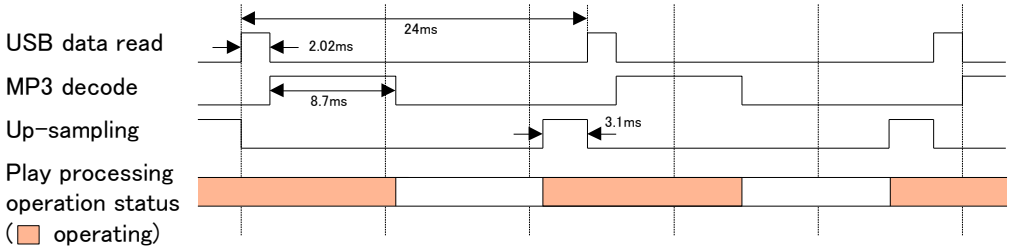
3.3.3 CPU Occupancy

3.3.3.1 MP3 Play in Progress

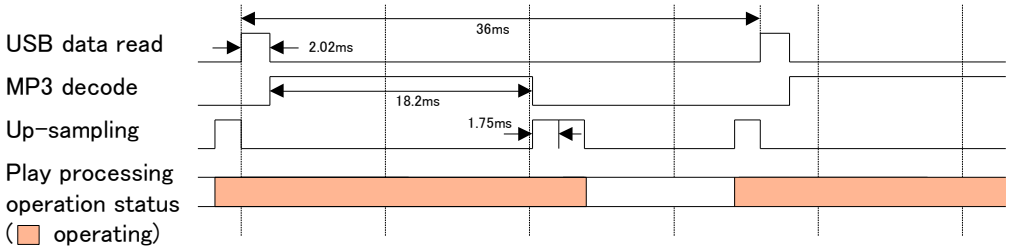
CPU processing status when MP3 is played is shown in Figure 15.

(Vertical line is 10 ms unit.)

Sampling Rate:24kHz, Bit Rate:64kbps, Channel Mode:Joint Stereo



Sampling Rate:32kHz, Bit Rate:96kbps, Channel Mode:Joint Stereo



Sampling Rate:48kHz, Bit Rate:128kbps, Channel Mode:Joint Stereo

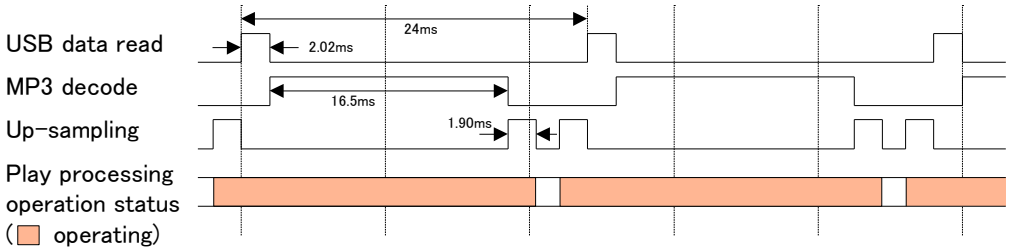


Figure 15 Processing Status While MP3 Plays

CPU occupancy is as given in Table 6 while MP3 plays.

Table 6 CPU Occupancy While MP3 Plays.

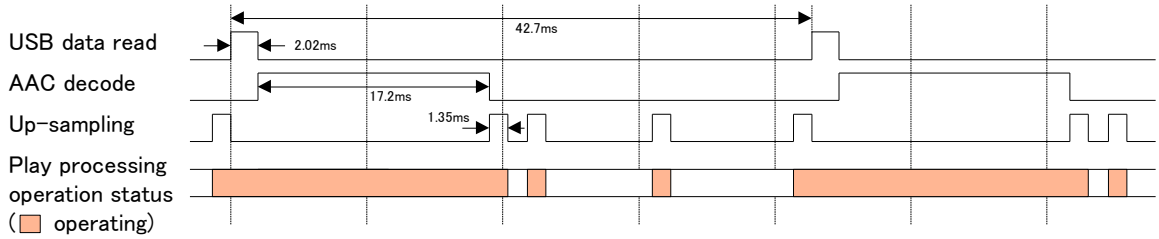
Conditions			CPU
Sampling Rate	Bit Rate	Channel Mode	occupancy (%)
24kHz	64kbps	Joint Stereo	57.6
32kHz	96kbps	Joint Stereo	70.8
48kHz	128kbps	Joint Stereo	93.0

3.3.3.2 AAC Play in Progress

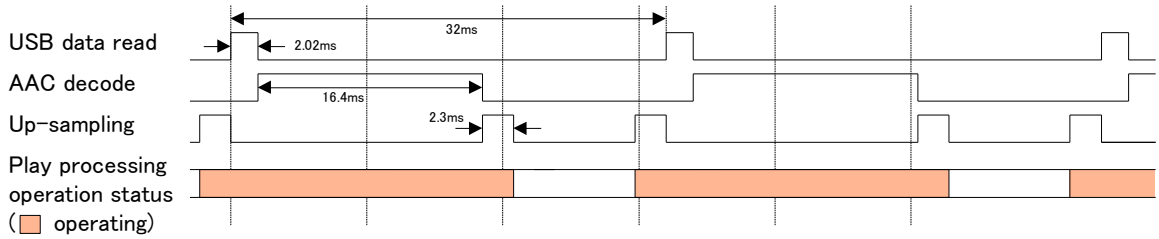
CPU processing status when AAC is played is shown in Figure 16.

(Vertical line is 10 ms unit.)

Sampling Rate:24kHz, Bit Rate:64kbps, Channel Mode:Stereo



Sampling Rate:32kHz, Bit Rate:96kbps, Channel Mode:Stereo



Sampling Rate:48kHz, Bit Rate:128kbps, Channel Mode:Stereo

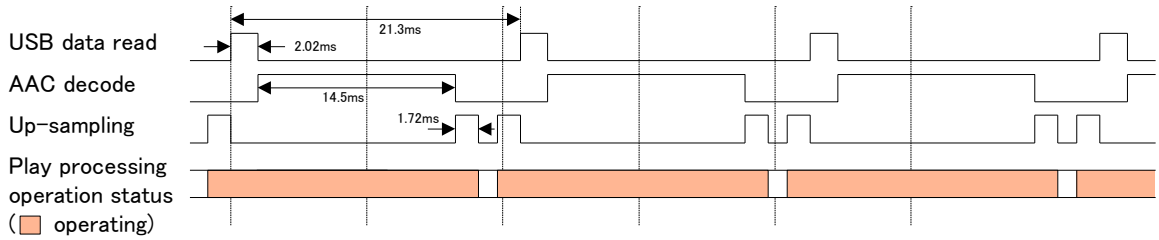


Figure 16 Processing Status While AAC Plays

CPU occupancy is as given in Table 7 while AAC plays.

Table 7 CPU Occupancy While AAC Plays.

Conditions			CPU Occupancy (%)
Sampling Rate	Bit Rate	Channel Mode	
24kHz	64kbps	Stereo	58.4
32kHz	96kbps	Stereo	71.9
48kHz	128kbps	Stereo	93.6

-End-