To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

SH7263 Group

Sampling Rate Converter (SRC) Setting Example

Introduction

This application note describes a setting example of sampling rate converter (SRC) for the SH7263.

Target Device

SH7263

Contents

1. Preface

1.1 Specifications

- By using the sampling rate converter (SRC), the sampling rate for PCM data is converted from 22.05 kHz to 44.1 kHz and the PCM data is transmitted via the serial sound interface (SSI).
- Data is input to or output from the SRC and input to the SSI using the direct memory access controller (DMAC).

1.2 Modules Used

- Sampling rate converter (SRC)
- Direct memory access controller (DMAC)
- Serial sound interface (SSI)

1.3 Applicable Conditions

- MCU: SH7263
- Operating frequency: Internal clock 200 MHz
  Bus clock 66.67 MHz
  Peripheral clock 33.33 MHz
- C compiler: SuperH RISC engine Family C/C++ Compiler Package Ver.9.01 Release01
  from Renesas Technology
- Compiler options: -cpu = sh2afpu -fpu = single -include = "$(WORKSPDIR)\inc"
  -object = "$(CONFIGDIR)\$(FILELEAF).obj" -debug -gbr = auto -chgincpath
  -errorpath -global_volatile = 0 -opt_range = all -infinite_loop = 0 -del_vacant_loop = 0
  -struct_alloc = 1 -nologo

1.4 Related Application Note

- None

2. Description of the Sample Application

In this sample application, the SRC converts the PCM data sampling rate from 22.05 kHz to 44.1 kHz.

2.1 Operational Overview of Module Used

The SRC converts the sampling rate for data produced by decoders such as WMA, MP3, or AAC.

Table 1 shows a module summary of the SRC used in this sample application. Figure 1 shows a block diagram of the SRC. For details of the SRC, see descriptions of the Sampling Rate Converter (SRC) in the SH7263 Group Hardware Manual (REJ09B0290).

Table 1 Features of SRC Used in This Sample Application

Item	Description
Data size	16 bits (stereo/monaural)
Sampling rates	Input: Either 8 kHz, 11.025 kHz, 12 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, or 48 kHz is selectable. Output: Either 44.1 kHz or 48 kHz is selectable.
Processing capacity	A maximum of 8 μs sample output interval (Pϕ = 33 MHz)
SNR	80 dB or higher
Interrupt Sources	Three interrupt sources: Input data FIFO empty, output data FIFO full, and output data FIFO overwrite
DMA Transfer Sources	Two DMA transfer sources: Input data FIFO empty and output data FIFO full

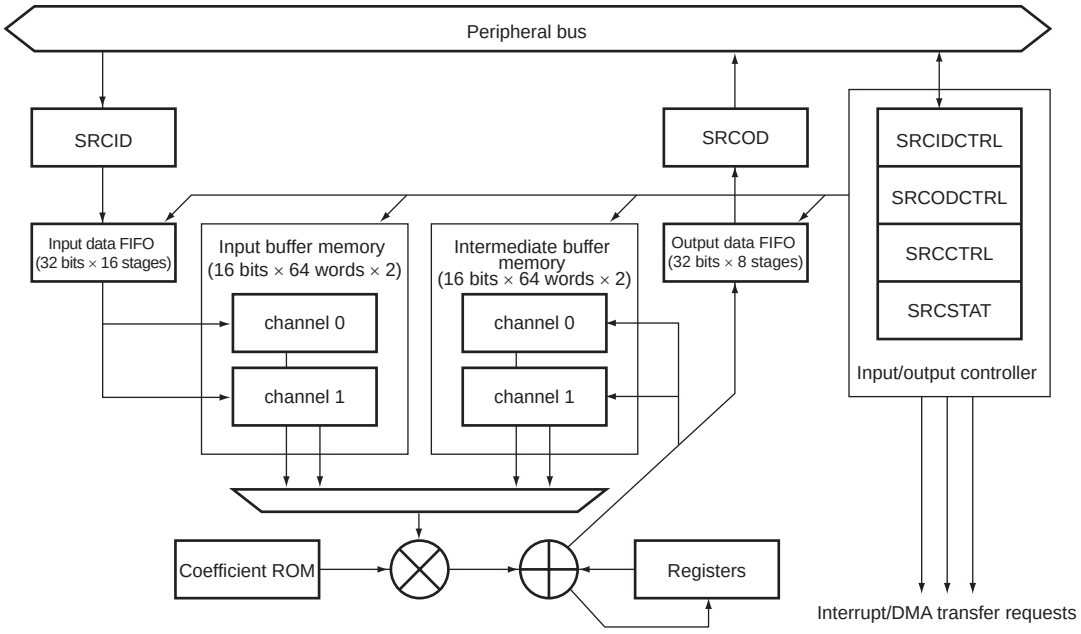


Figure 1 Block Diagram of SRC

2.2 Procedure for Setting the Module Used

Figure 2 shows an example of the initial SRC setting sequence. For details of the register settings, see the SH7263 Group Hardware Manual (REJ09B0290).

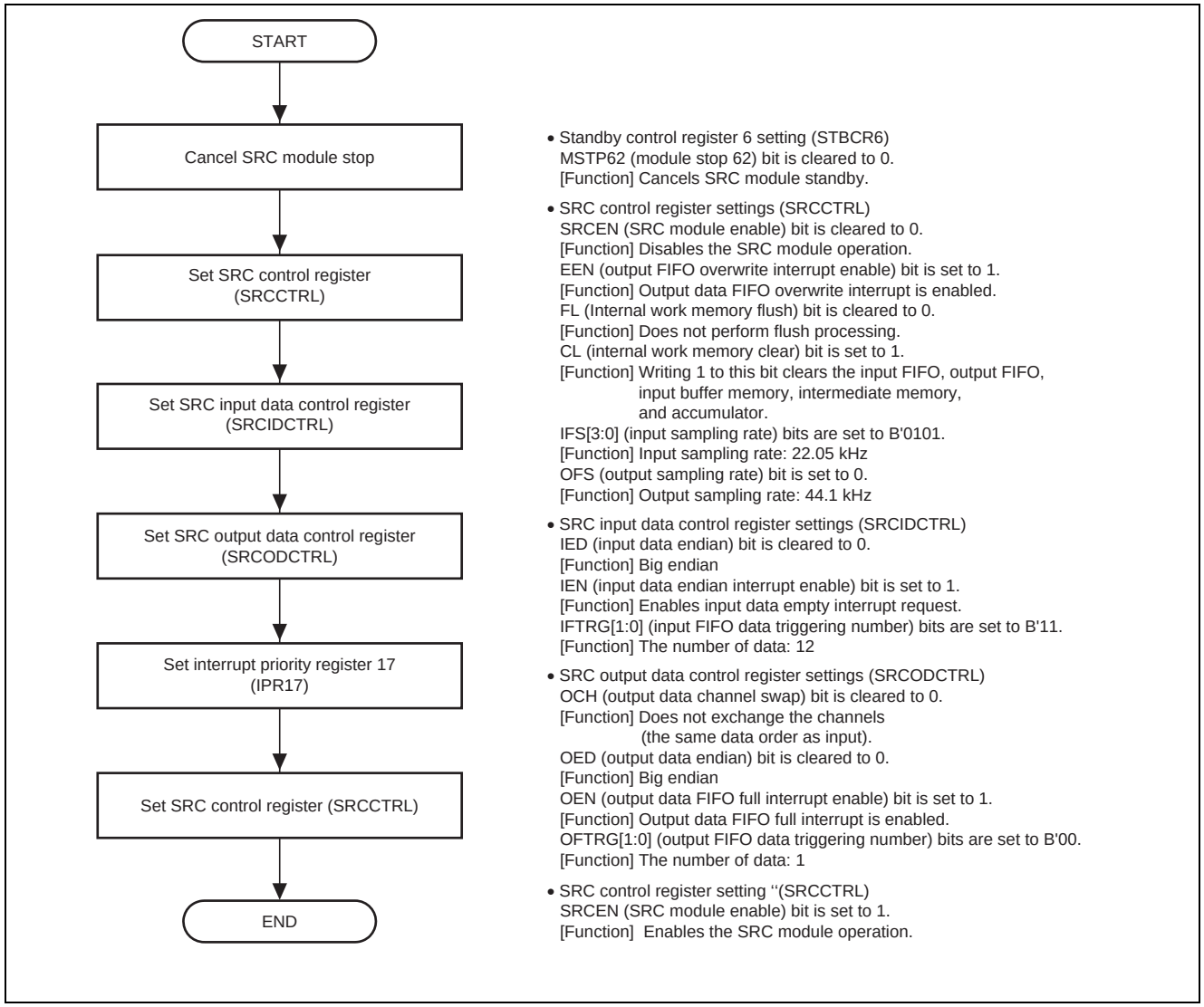


Figure 2 Example of SRC Initial Setting Sequence

2.3 Operation of the Sample Program

In this sample program, the PCM data sampling rate is converted from 22.05 to 44.1 kHz. The converted PCM data is transferred to the SSI transmit data register (SSITDR) by the DMAC. Figure 3 shows the PCM data flow and figure 4 shows the sample program sequence flow.

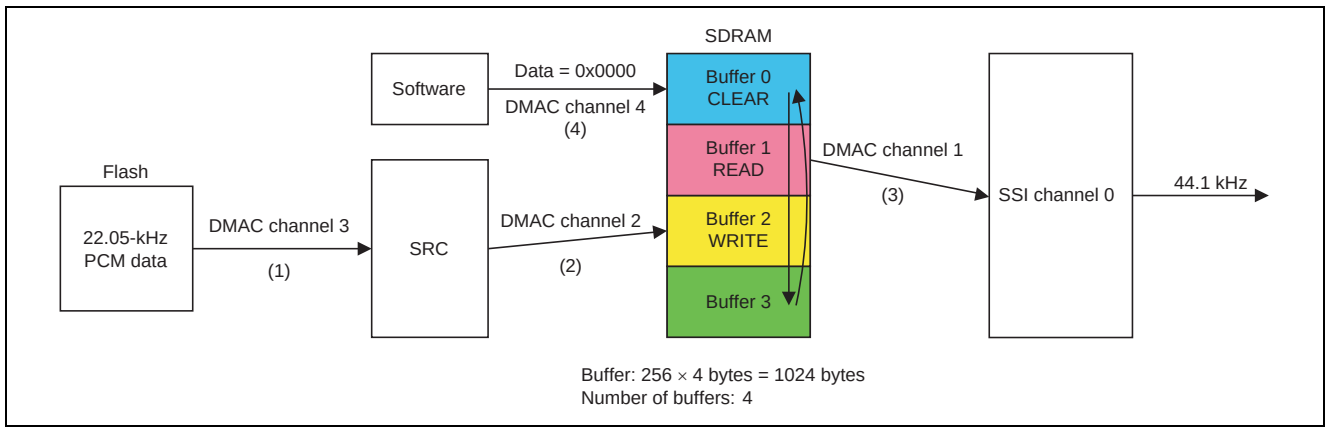


Figure 3 PCM Data Flow

Description of Operation

1. The IDE interrupt request from the SRC is issued to activate channel 3 of the DMAC. Data are written to the SRCID register of the SRC by the DMA transfer.

2. The ODF interrupt request from the SRC is issued to activate channel 2 of the DMAC. Data are written to the SRCOD register of the SRC by the DMA transfer.
   As long as space is available in the buffer, PCM data are transferred to the buffer.
   When the buffer has no available space, transfer on the DMAC channel 2 is suspended.
   When space in the buffer becomes available again, transfer on DMAC channel 2 is re-enabled.

3. SSI channel 0 issues an interrupt request to activate DMAC channel 1. Data are written to the SSITDR register of the SRC by the DMA transfer.
   PCM data stored in the buffer is transferred to the SSI channel 0. When all of the sampled PCM data has been transferred, the SSI is set to mute.

4. When data from the buffer has been transferred to the SSI channel 0, the buffer is cleared for checking of operation.

If transfer on DMAC channel 2 is suspended because the buffer is full, the output FIFO may become full of data since reading it is not possible. If a further conversion is completed while the output FIFO is full, the OVF bit in the SRC status register (SRCSTAT) is set to 1. Conversion processing is stopped until the OVF bit is cleared. Request for transfer on DMAC channel 3 are not generated if conversion processing has been stopped. The OVF bit is cleared when transfer on DMAC channel 2 is again possible.

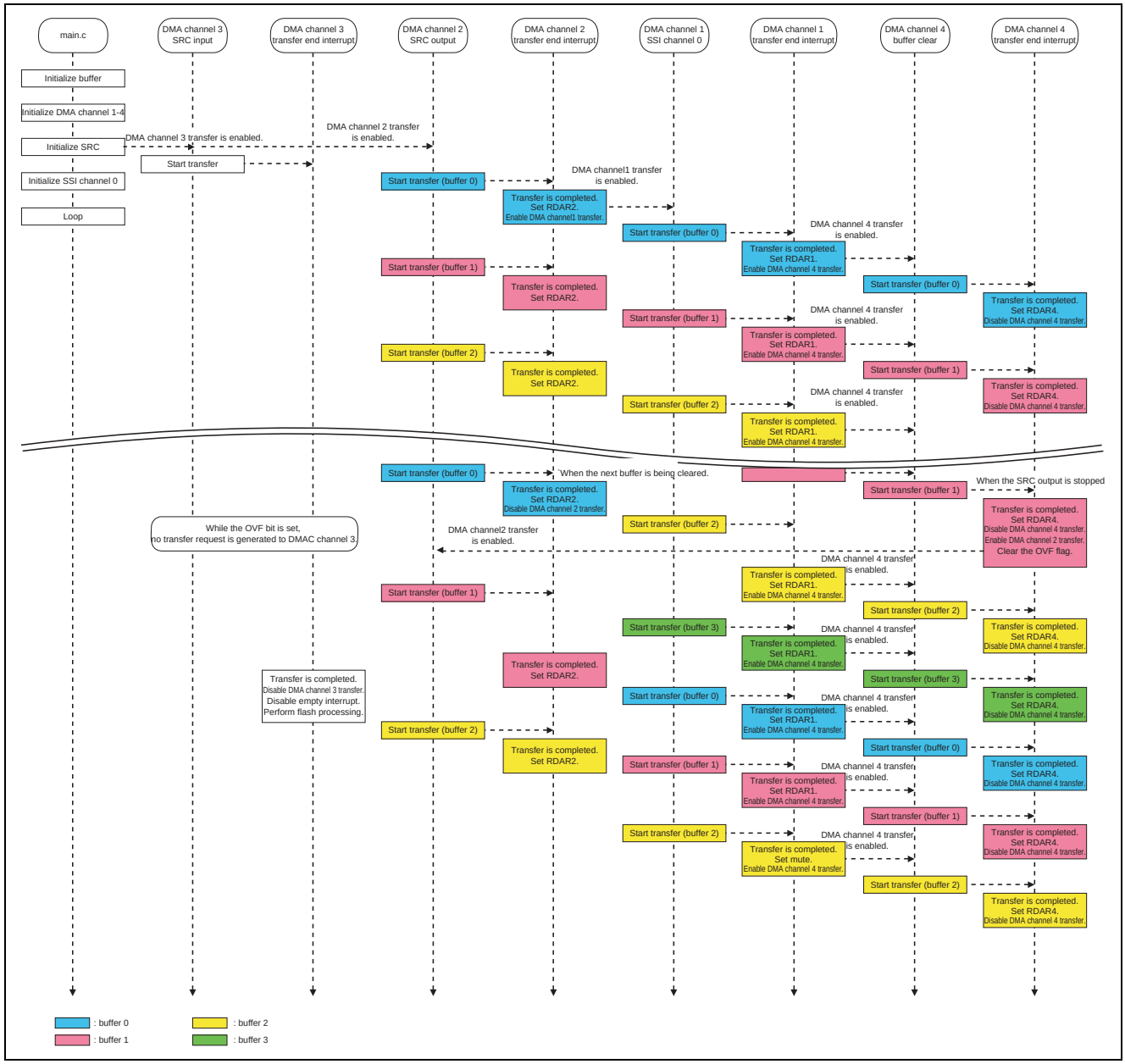


Figure 4 Sample Program Sequence Flow

Figures 5 to 10 show flowchanrts of the sample program.

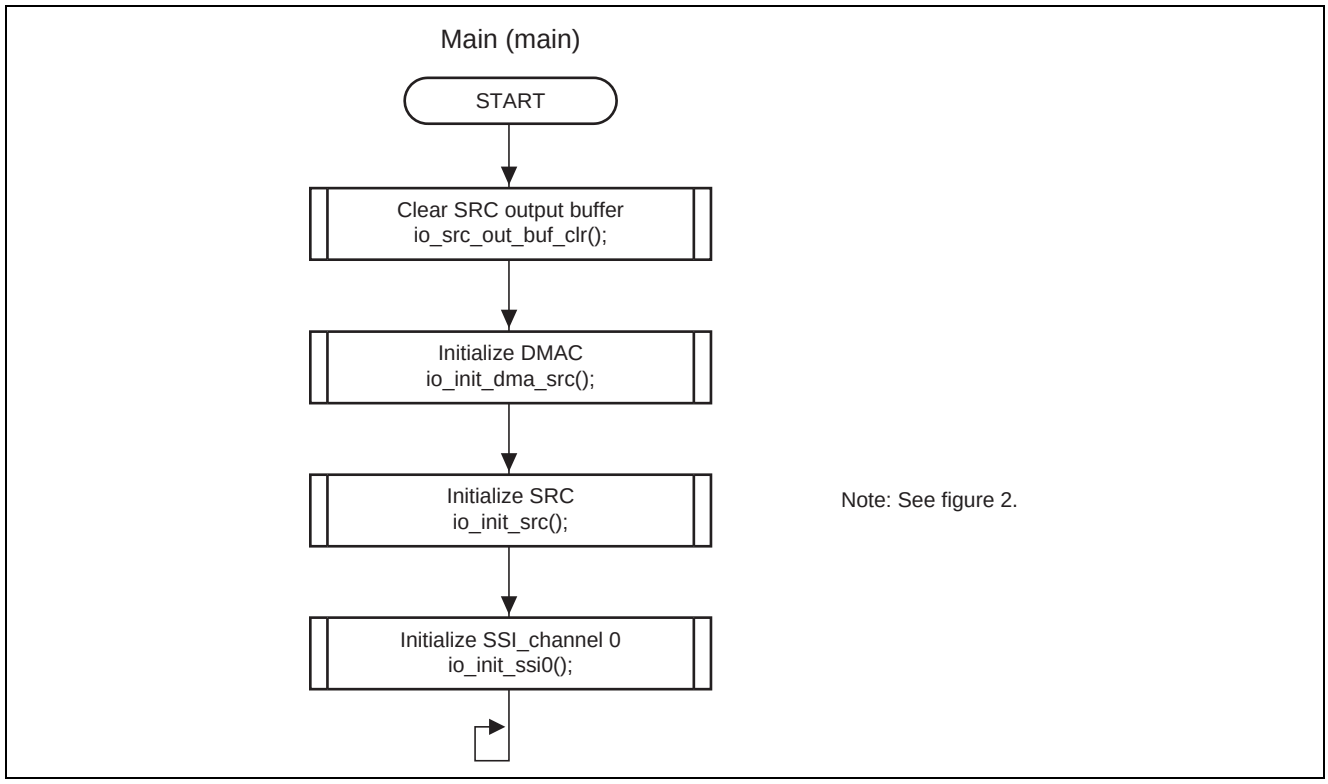
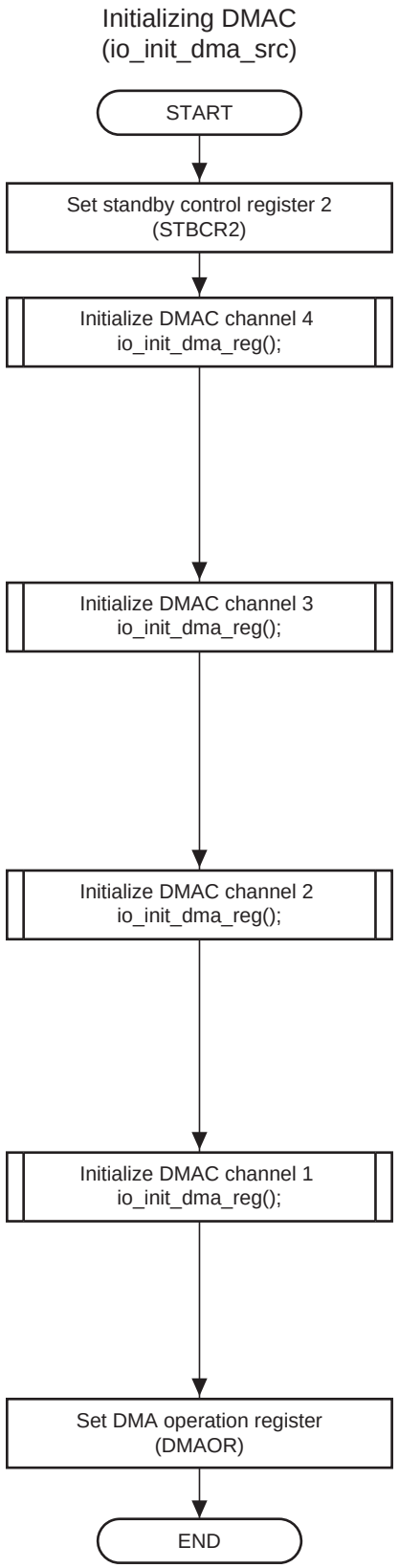


Figure 5 Sample Program Flowchart (1)



Supplies clock to DMAC.

- For clearing buffer
  Disables the SAR reload function.
  Enables the DAR reload function.
  DMA transfer is disabled when the TE bit is set.
  Destination address is incremented.
  Fixed source address
  Auto request
  Cycle steal mode
  Longword transfer
  Enables an interrupt request.
  DMA transfer disabled

- For SRC input
  Disables the SAR reload function.
  Disables the DAR reload function.
  DMA transfer is disabled when the TE bit is set.
  Fixed destination address
  Source address is incremented.
  DMA extension resource selector
  Cycle steal mode
  Longword transfer
  Enables an interrupt request.
  DMA transfer enabled

- For SRC output
  Disables the SAR reload function.
  Enables the DAR reload function.
  DMA transfer is disabled when the TE bit is set.
  Destination address is incremented.
  Fixed source address
  DMA extension resource selector
  Cycle steal mode
  Longword transfer
  Enables an interrupt request.
  DMA transfer enabled

- For SSI transmission
  Enables the SAR reload function.
  Disables the DAR reload function.
  DMA transfer does not stop even when the TE bit is set.
  Fixed destination address
  Source address is incremented.
  DMA extension resource selector
  Cycle steal mode
  Longword transfer
  Enables an interrupt request.
  DMA transfer disabled

Clears the address error flag and the NMI flag.
Enables DMA transfer on all channels.

Figure 6 Sample Program Flowchart (2)

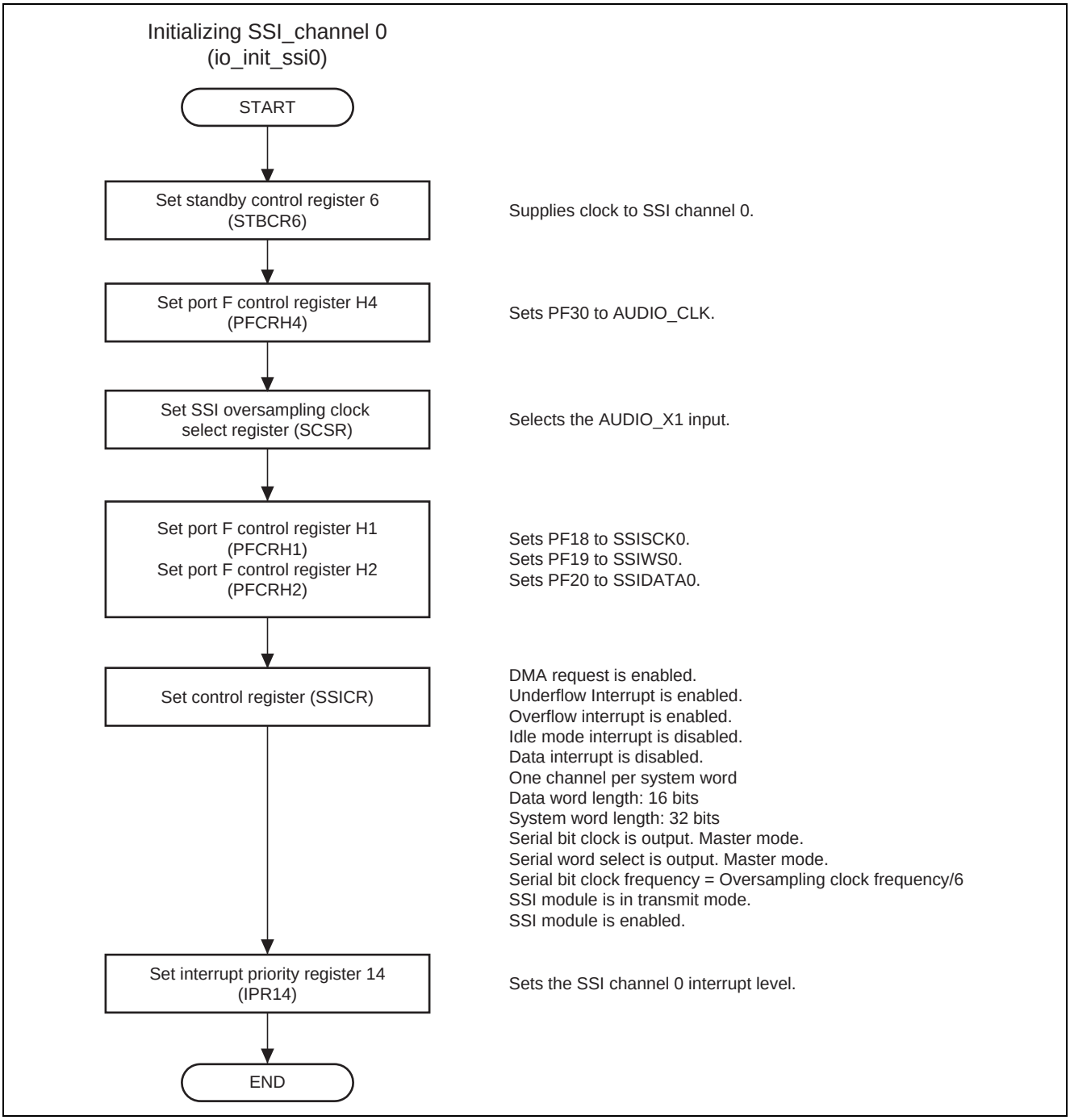


Figure 7 Sample Program Flowchart (3)

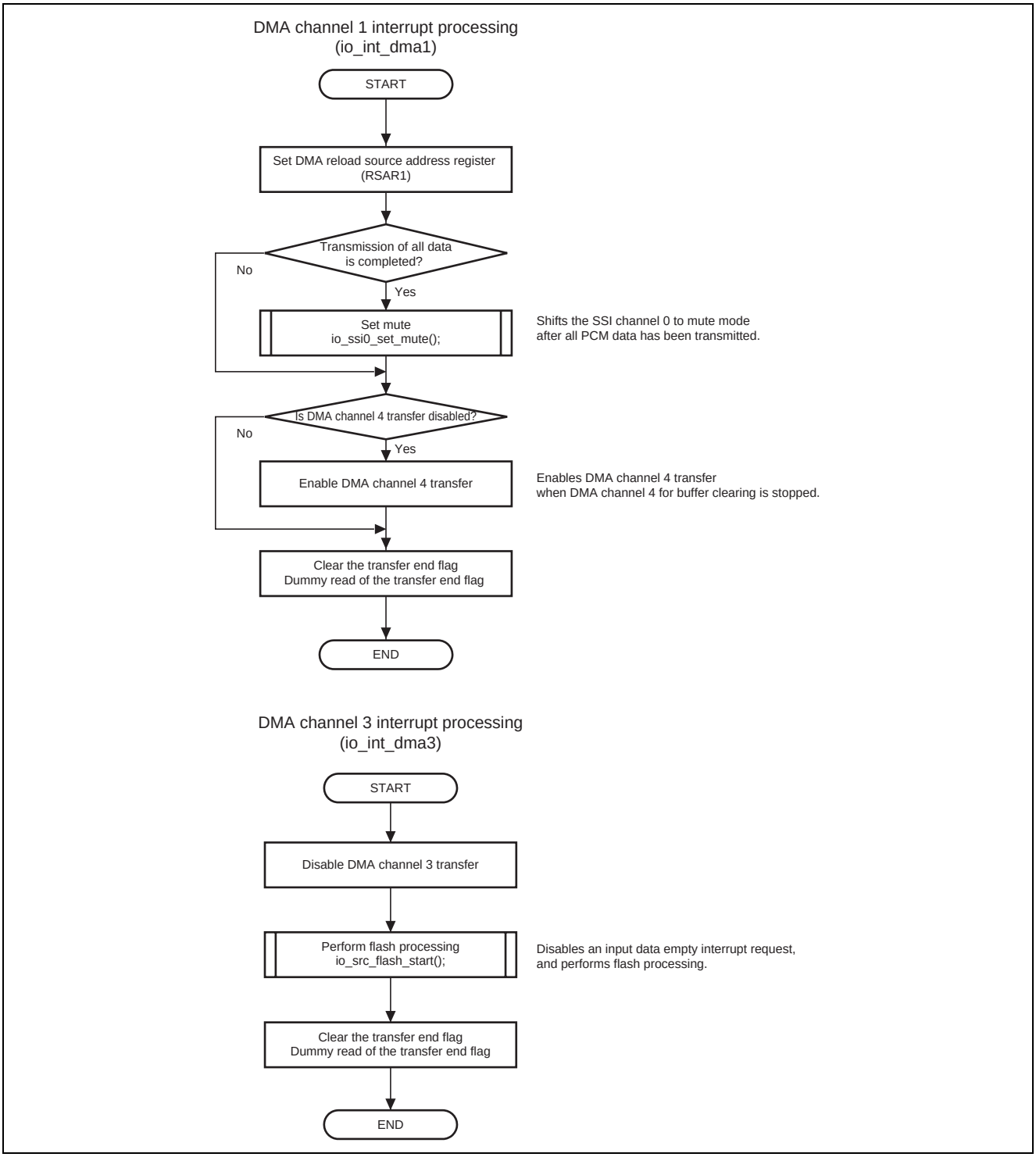


Figure 8 Sample Program Flowchart (4)

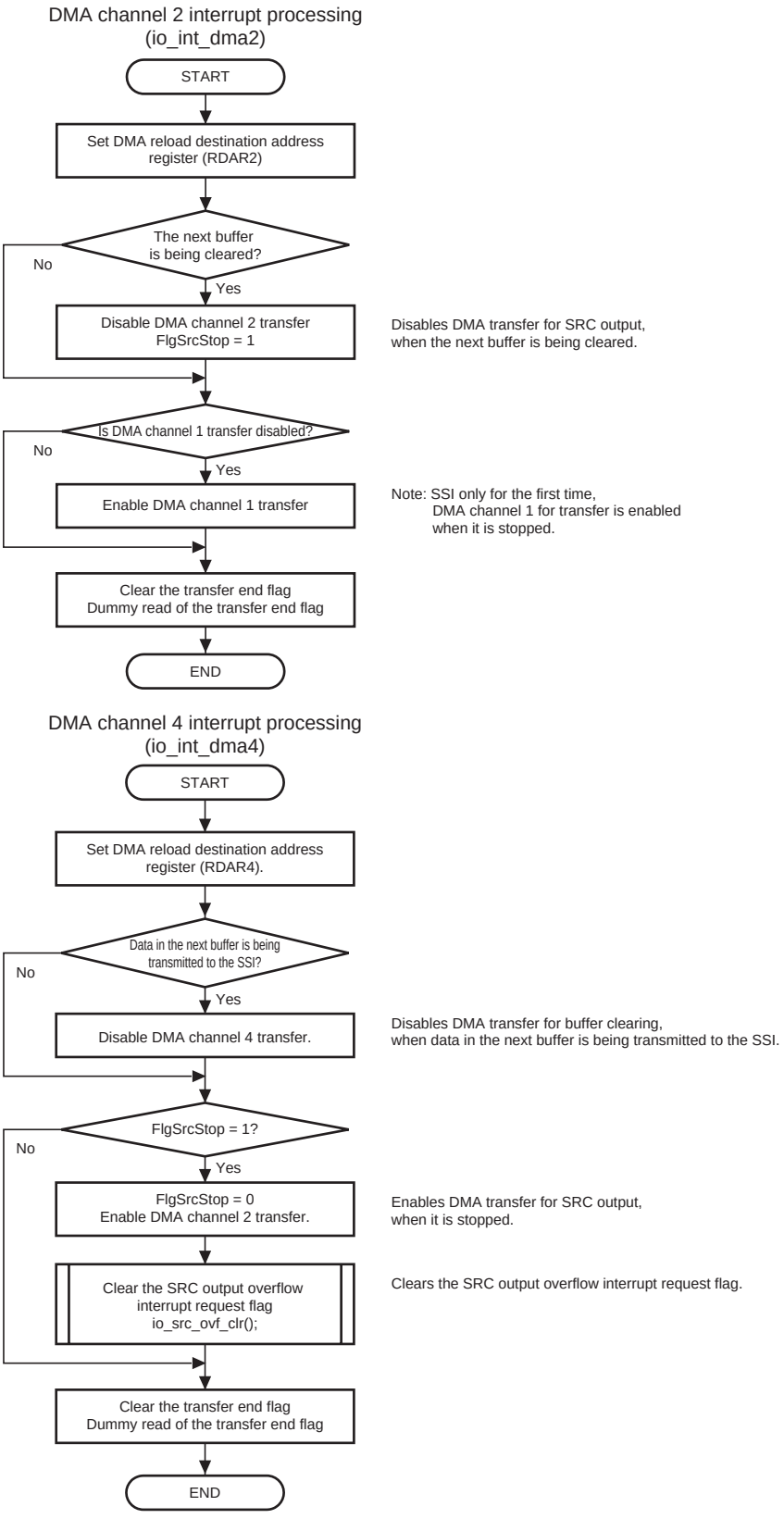


Figure 9 Sample Program Flowchart (5)

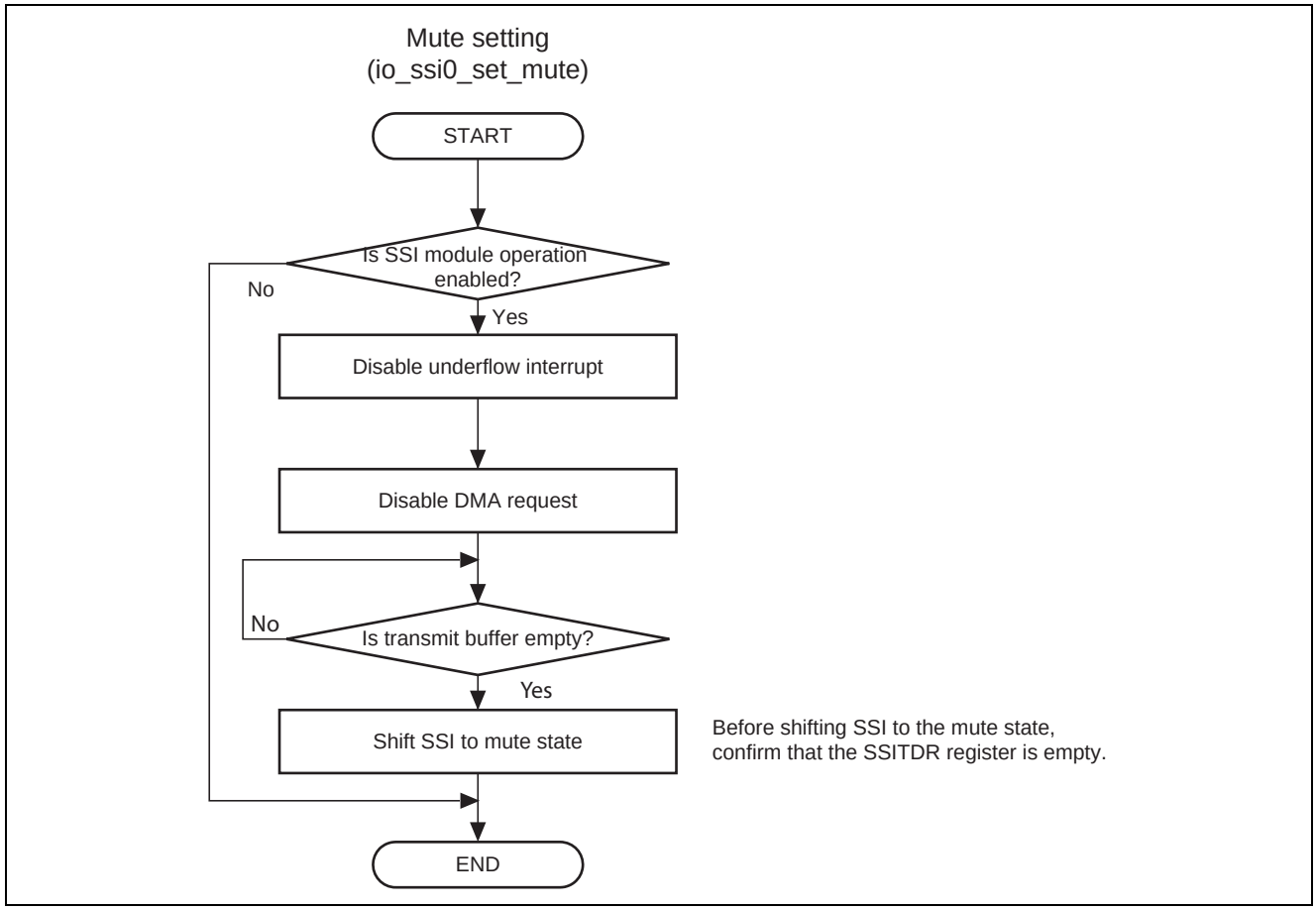


Figure 10 Sample Program Flowchart (6)

2.4 Sequence of Processing by the Sample Program

Tables 2 to 7 give the register settings in the sample program.

Table 2 SRC Register Settings Used in Sample Program

Register Name	Address	Setting Value	Description
SRC input data control register (SRCIDCTRL)	H'FFFF 4008	H'0103	IED = 0: Big endian IEN = 1: Input data FIFO empty interrupt is enabled. IFTRG[1:0] = B'11: 12 (Number of data in the FIFO)
SRC output data control register (SRCODCTRL)	H'FFFF 400A	H'0100	OCH = 0: Does not exchange the channels. OED = 0: Big endian OEN = 1: Output data FIFO full interrupt is enabled. OFTRG[1:0] = B'00: 1 (Number of data in the FIFO)
SRC control register (SRCCTRL)	H'FFFF 400C	H'0550	SRCEN = 0: Disables the SRC module operation. EEN = 1: Output data FIFO overwrite interrupt is enabled. FL = "0": Does not perform flush processing. CL = "1": Writing 1 to this bit clears the input FIFO, output FIFO, input buffer memory, intermediate memory, and accumulator. IFS[3:0] = "B'0101": 22.05 kHz OFS = "0": 44.1 kHz
		H'1450	SRCEN = 1: Enables the SRC module operation.

Table 3 SSI Register Settings Used in Sample Program

Register Name	Address	Setting Value	Description
Control register 0 (SSICR_0)	H'FFFF C000	H'1C0B D553	DMEN = "1": DMA request is enabled. UIEN = "1": Underflow Interrupt is enabled. OIEN = "1": Overflow interrupt is enabled. IIEN = "0": Idle mode interrupt is disabled. CHNL = "B'00": One channel per system word DWL = "B'001": Data word length 16 bits SWL = "B'011": System Word Length 32 bits SCKD = "1": Serial bit clock is output. Master mode. SWSD = "1": Serial word is set as output. Master mode. SCKP = "0": Serial bit clock polarity. SSIWS and SSIDATA change on the SSISCK falling edge. SWSP = "1": Serial WS polarity. High for 1st channel, low for 2nd channel. SPDP = "0": Padding bits are low. SDTA = "1": Transmitting and receiving in the order of padding bits and serial data PDTA = "0": When a data word length is 16 bits, the PDTA setting is ignored. The first data word is held by bits 15 to 0 and the second data word is held by bits 31 to 16. DEL = "1": No delay between SSIWS and SSIDATA CKDV = "B'101": Oversampling clock frequency/6 MUEN = "0": Module is not muted. TRMD = "1": Module is in transmit mode. EN = "1": Module is enabled.

Table 4 DMAC Register Settings Used in Sample Program (SSI)

Register Name	Address	Setting Value	Description
DMA channel control register_1 (CHCR1)	H'FFFE 101C	H'0000 0000	DE = "0": DMA transfer disabled
		H'2010 1814	TC = "0": Transmits data once. RLDSAR = "1": Enables the SAR reload function. RLDDAR = "0": Disables the DAR reload function. TEMASK = "1"*[1]: DMA transfer does not stop, even when the TE bit is set to 1. DM = "B'00": Fixed destination address SM = "B'01": Source address is incremented. RS = "B'1000": DMA extension resource selector TB = "0": Cycle steal mode TS = "B'10": Longword transfer IE = "1": Enables an interrupt request.
		H'2010 1815	DE = "1": DMA transfer enabled
DMA source address register_1 (SAR1)	H'FFFE 1010	H'2C00 0000	Transfer source start address: Sets as an area in SDRAM.
DMA reload source address register_1 (RSAR1)	H'FFFE 1110	H'2C00 0400	Transfer source start address: Sets as an area in SDRAM.
DMA destination address register_1 (DAR1)	H'FFFE 1014	H'FFFF C008	Transfer destination start address: SSI transmit data register (SSITDR_0)
DMA reload destination address register_1 (RDAR1)	H'FFFE 1114	H'FFFF C008	Transfer destination start address: SSI transmit data register (SSITDR_0)
DMA transfer count register_1 (DMATCR1)	H'FFFE 1018	H'0000 0100	Transfer count: 256
DMA reload transfer count register_1 (RDMATCR1)	H'FFFE 1118	H'0000 0100	Transfer count: 256
DMA extension resource selector 0 (DMARS0)	H'FFFE 1300	H'0023	SSI channel 0 selected

Note: 1. PCM data must be output from the SSI at a constant timing.
If TEMASK = 0, the DMA is stopped on completion of DMA transfer. Thus, the SSI might have an underflow if interrupt processing on completion of DMA transfer is delayed due to the period over which interrupts are disabled in the main routine and so on. To prevent this, we recommend the setting TEMASK = 1 so that DMA transfer can continue immediately after a previous round of DMA transfer is completed.

Table 5 DMAC Register Settings Used in Sample Program (SRC Input)

Register Name	Address	Setting Value	Description
DMA channel control register_3 (CHCR3)	H'FFFE 103C	H'0000 0000	DE = "0": DMA transfer disabled
		H'0000 1814	TC = "0": Transmits data once. RLDSAR = "0": Disables the SAR reload function. RLDDAR = "0": Disables the DAR reload function. TEMASK = "0": DMA transfer is disabled, when the TE bit is set to 1. DM = "B'00": Fixed destination address SM = "B'01": Source address is incremented. RS = "B'1000": DMA extension resource selector TB = "0": Cycle steal mode TS = "B'10": Longword transfer IE = "1": Enables an interrupt request.
		H'0000 1815	DE = "1": DMA transfer enabled
DMA source address register_3 (SAR3)	H'FFFE 1030	H'0000 1900	Transfer source start address: Sets as an area in flash memory.
DMA reload source address register_3 (RSAR3)	H'FFFE 1130	H'0000 1900	Transfer source start address: Sets as an area in flash memory.
DMA destination address register_3 (DAR3)	H'FFFE 1034	H'FFFF 4000	Transfer destination start address: SRC input data register (SRCID)
DMA reload destination address register_3 (RDAR3)	H'FFFE 1134	H'FFFF 4000	Transfer destination start address: SRC input data register (SRCID)
DMA transfer count register_3 (DMATCR3)	H'FFFE 1038	H'0000 4EDF	Transfer count: 20191
DMA reload transfer count register_3 (RDMATCR3)	H'FFFE 1138	H'0000 4EDF	Transfer count: 20191
DMA extension resource selector 1 (DMARS1)	H'FFFE 1304	H'4100	SRC input data empty selected

Table 6 DMAC Register Settings Used in Sample Program (SRC Output)

Register Name	Address	Setting Value	Description
DMA channel control register_2 (CHCR2)	H'FFFE 102C	H'0000 0000	DE = "0": DMA transfer disabled
		H'1000 4814	TC = "0": Transmits data once. RLDSAR = "0": Disables the SAR reload function. RLDDAR = "1": Enables the DAR reload function. TEMASK = "0": DMA transfer is disabled, when the TE bit is set to 1. DM = "B'01": Destination address is incremented. SM = "B'00": Fixed source address RS = "B'1000": DMA extension resource selector TB = "0": Cycle steal mode TS = "B'10": Longword transfer IE = "1": Enables an interrupt request.
		H'1000 4815	DE = "1": DMA transfer enabled
DMA source address register_2 (SAR2)	H'FFFE 1020	H'FFFF 4004	Transfer source start address: SRC output data register (SRCOD)
DMA reload source address register_2 (RSAR2)	H'FFFE 1120	H'FFFF 4004	Transfer source start address: SRC output data register (SRCOD)
DMA destination address register_2 (DAR2)	H'FFFE 1024	H'2C00 0000	Transfer destination start address: Sets as an area in SDRAM.
DMA reload destination address register_2 (RDAR2)	H'FFFE 1124	H'2C00 0000	Transfer destination start address: Sets as an area in SDRAM.
DMA transfer count register_2 (DMATCR2)	H'FFFE 1028	H'0000 0100	Transfer count: 256
DMA reload transfer count register_2 (RDMATCR2)	H'FFFE 1128	H'0000 0100	Transfer count: 256
DMA extension resource selector 1 (DMARS1)	H'FFFE 1304	H'0042	SRC output data full selected

Table 7 DMAC Register Settings Used in Sample Program (Buffer Clear)

Register Name	Address	Setting Value	Description
DMA channel control register_4 (CHCR4)	H'FFFE 104C	H'0000 0000	DE = "0": DMA transfer disabled
		H'1000 4414	TC = "0": Transmits data once. Note: Enabled only for on-chip peripheral module requests. RLDSAR = "0": Disables the SAR reload function. RLDDAR = "1": Enables the DAR reload function. TEMASK = "0": DMA transfer is disabled, when the TE bit is set to 1. DM = "B'01": Destination address is incremented. SM = "B'00": Fixed source address RS = "B'0100": Auto request TB = "0": Cycle steal mode TS = "B'10": Longword transfer IE = "1": Enables an interrupt request.
		H'1000 4415	DE = "1": DMA transfer enabled
DMA source address register_4 (SAR4)	H'FFFE 1040	H'FFF807E0	Transfer source start address: Sets as an area in RAM.
DMA reload source address register_4 (RSAR4)	H'FFFE 1140	H'FFF807E0	Transfer source start address: Sets as an area in RAM.
DMA destination address register_4 (DAR4)	H'FFFE 1044	H'2C00 0000	Transfer destination start address: Sets as an area in SDRAM.
DMA reload destination address register_4 (RDAR4)	H'FFFE 1144	H'2C00 0000	Transfer destination start address: Sets as an area in SDRAM.
DMA transfer count register_4 (DMATCR4)	H'FFFE 1048	H'0000 0100	Transfer count: 256
DMA reload transfer count register_4 (RDMATCR4)	H'FFFE 1148	H'0000 0100	Transfer count: 256
DMA extension resource selector2 (DMARS2)	H'FFFE 1308	H'0000	Not used for auto requests.

Tables 8 and 9 show variables and macro definitions used in this sample program, respectively.

Table 8 Variables Used in Sample Program

Variable Name	Description	Area	Used Module Name
unsigned long rom_ding22_05_bin[ ]	Sampling rate 22.05 kHz PCM data	Flash	void io_init_dma3 (void *src, void *dst, size_t size, unsigned int mode)
unsigned long size_ding22_05_bin	Sampling rate 22.05 kHz Size of PCM data	Flash	void io_init_dma3 (void *src, void *dst, size_t size, unsigned int mode)

Table 9 Macro Definitions Used in Sample Program

Macro Definition	Setting Value	Description
IN_SAMPLING_8_KHZ	H'0000	SRC input sampling rate 8 kHz
IN_SAMPLING_11_025_KHZ	H'0001	SRC input sampling rate 11.025 kHz
IN_SAMPLING_12_KHZ	H'0002	SRC input sampling rate 12 kHz
IN_SAMPLING_16_KHZ	H'0003	SRC input sampling rate 16 kHz
IN_SAMPLING_22_05_KHZ	H'0004	SRC input sampling rate 22.05 kHz
IN_SAMPLING_24_KHZ	H'0005	SRC input sampling rate 24 kHz
IN_SAMPLING_32_KHZ	H'0006	SRC input sampling rate 32 kHz
IN_SAMPLING_44_1_KHZ	H'0008	SRC input sampling rate 44.1 kHz
IN_SAMPLING_48_KHZ	H'0009	SRC input sampling rate 48 kHz
OUT_SAMPLING_44_1_KHZ	H'0000	SRC output sampling rate 44.1 kHz
OUT_SAMPLING_48_KHZ	H'0001	SRC output sampling rate 48 kHz
SRC_OUT_BUF_SIZE	256	Buffer size on the SRC output side
SRC_OUT_BUF_NUM	4	Number of buffers on the SRC output side
BUF_CLR	0	Buffer clear operation
BUF_SRC	1	SRC output operation
BUF_SSI	2	SSI transmit operation
DMA_RLDSAR_OFF	H'0000 0000	DMA SAR reload function is disabled.
DMA_RLDSAR_ON	H'2000 0000	DMA SAR reload function is enabled.
DMA_RLDDAR_OFF	H'0000 0000	DMA DAR reload function is disabled.
DMA_RLDDAR_ON	H'1000 0000	DMA DAR reload function is enabled.
DMA_TE_STOP	H'0000 0000	DMA transfer is stopped when the DMA TE bit is set.
DMA_TE_CONT	H'0010 0000	DMA transfer is continued even when the DMA TE bit is set.
DMA_DAR_FIX	H'0000 0000	DMA destination address is fixed.
DMA_DAR_INC	H'0000 4000	DMA destination address is incremented.
DMA_DAR_DEC	H'0000 8000	DMA destination address is decremented.
DMA_SAR_FIX	H'0000 0000	DMA source address is fixed.
DMA_SAR_INC	H'0000 1000	DMA source address is incremented.
DMA_SAR_DEC	H'0000 2000	DMA source address is decremented.
DMA_AUTO_REQ	H'0000 0400	DMA resource select: Auto request
DMA_ENH_RES	H'0000 0800	DMA resource select: Extension resource selector
DMA_SIZE_BYTE	H'0000 0000	DMA transfer size: Byte unit
DMA_SIZE_WORD	H'0000 0008	DMA transfer size: Word unit
DMA_SIZE_LONG	H'0000 0010	DMA transfer size: Longword unit
DMA_SIZE_LONGx4	H'0000 0018	DMA transfer size: 16-byte (four longword) unit
DMA_INT_DISABLE	H'0000 0000	DMA transfer completion interrupt is not used.
DMA_INT_ENABLE	H'0000 0004	DMA transfer completion interrupt is used.
DMA_DISABLE	H'0000 0000	DMA transfer is disabled.
DMA_ENABLE	H'0000 0001	DMA transfer is enabled.

3. Listing of the Sample Program

```
/*""FILE COMMENT""*****************************************************************
*
*       System Name     : SH7263 Sample Program
*       File Name       : main.c
*       Contents        : SRC Sample Program
*       Version         : 1.00.00
*       Model           : R0K572630D001BR
*       CPU             : SH7263
*       Compiler        : SHC9.1.1.0
*       note            :
*
*                           Note
*                           This sample program is for reference
*                           and its operation is not guaranteed.
*                           Customers should use this sample program for technical reference
*                           in software development
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       history         : 2008.05.09 ver.1.00.00
*""FILE COMMENT END""*************************************************************/
#include <string.h>
#include "iodefine.h"             /* iodefine.h is a file automatically created by HEW
#include "src.h"

/* ==== Prototype declaration ==== */
void main(void);

```

Figure 11 Sample Program Listing: main.c (1)

```
/*""FUNC COMMENT""*************************************************
* Outline      : Sample program main
*-----------------------------------------------------------------
* Include      : #include <string.h>
*-----------------------------------------------------------------
* Declaration  : void main(void);
*-----------------------------------------------------------------
* Function     : Clears the PCM data storage buffer after modification of
*              : sampling rate, and then sets DMA ch1 to ch4, SRC, and SSI ch0.
*              : For SRC setting, the input sampling rate is set to 22.05 kHz,
*              : and the output sampling rate to 44.1 kHz.
*-----------------------------------------------------------------
* Argument     : void
*-----------------------------------------------------------------
* Return Value : void
*-----------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*************************************************/
void main(void)
{
    /* ---- Initialize SRC buffer ---- */
    src_out_buf_clr();                          /* Clear the SRC output buffer */

    /* ---- Initialize DMA channels 1 to 4 ---- */
    io_init_dma_src();
           /* ch1 : SSI transmission */
           /* ch2 : SRC output */
           /* ch3 : SRC input */
           /* ch4 : Buffer clear */
    /* ---- Initialize SRC ---- */
    io_init_src( IN_SAMPLING_22_05_KHZ,         /* Input sampling rate */
                 OUT_SAMPLING_44_1_KHZ);        /* Output sampling rate */

    /* ---- Initialize SSI ch0 ---- */
    io_init_ssi0();

    while(1){
           /* Program end */
    }
}

/* End of File */
```

Figure 12 Sample Program Listing: main.c (2)

```
/*""FILE COMMENT""*****************************************************************
*
*       System Name     : SH7263 Sample Program
*       File Name       : dmac.c
*       Contents        : DMAC
*       Version         : 1.00.00
*       Model           : R0K572630D001BR
*       CPU             : SH7263
*       Compiler        : SHC9.1.1.0
*       note            :
*
*                           Note
*                           This sample program is for reference
*                           and its operation is not guaranteed.
*                           Customers should use this sample program for technical reference
*                           in software development
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       history         : 2008.05.09 ver.1.00.00
*""FILE COMMENT END""*************************************************************/
#include <string.h>
#include "iodefine.h"             /* iodefine.h is a file automatically created by HEW */
#include "src.h"
#include "dmac.h"

extern unsigned long size_ding22_05_bin;
extern unsigned long rom_ding22_05_bin[];

const ST_DMA DMA[8]={
    { /* ch0 */
            &DMAC.SAR0.LONG,        /* DMA source address register (SAR) */
            &DMAC.DAR0.LONG,        /* DMA destination address register (DAR) */
            &DMAC.DMATCR0.LONG,     /* DMA transfer count register (DMATCR) */
            &DMAC.CHCR0.LONG,       /* DMA channel control registers (CHCR) */
            &DMAC.RSAR0.LONG,       /* DMA reload source address register (RSAR) */
            &DMAC.RDAR0.LONG,       /* DMA reload destination address register (RDAR) */
            &DMAC.RDMATCR0.LONG,    /* DMA reload transfer count register (RDMATCR) */
    },
```

Figure 13 Sample Program Listing: dmac.c (1)

```
 46        { /* ch1 */
 47            &DMAC.SAR1.LONG,
 48            &DMAC.DAR1.LONG,
 49            &DMAC.DMATCR1.LONG,
 50            &DMAC.CHCR1.LONG,
 51            &DMAC.RSAR1.LONG,
 52            &DMAC.RDAR1.LONG,
 53            &DMAC.RDMATCR1.LONG,
 54        },
 55        { /* ch2 */
 56            &DMAC.SAR2.LONG,
 57            &DMAC.DAR2.LONG,
 58            &DMAC.DMATCR2.LONG,
 59            &DMAC.CHCR2.LONG,
 60            &DMAC.RSAR2.LONG,
 61            &DMAC.RDAR2.LONG,
 62            &DMAC.RDMATCR2.LONG,
 63        },
 64        { /* ch3 */
 65            &DMAC.SAR3.LONG,
 66            &DMAC.DAR3.LONG,
 67            &DMAC.DMATCR3.LONG,
 68            &DMAC.CHCR3.LONG,
 69            &DMAC.RSAR3.LONG,
 70            &DMAC.RDAR3.LONG,
 71            &DMAC.RDMATCR3.LONG,
 72        },
 73        { /* ch4 */
 74            &DMAC.SAR4.LONG,
 75            &DMAC.DAR4.LONG,
 76            &DMAC.DMATCR4.LONG,
 77            &DMAC.CHCR4.LONG,
 78            &DMAC.RSAR4.LONG,
 79            &DMAC.RDAR4.LONG,
 80            &DMAC.RDMATCR4.LONG,
 81        },
 82        { /* ch5 */
 83            &DMAC.SAR5.LONG,
 84            &DMAC.DAR5.LONG,
 85            &DMAC.DMATCR5.LONG,
 86            &DMAC.CHCR5.LONG,
 87            &DMAC.RSAR5.LONG,
 88            &DMAC.RDAR5.LONG,
 89            &DMAC.RDMATCR5.LONG,
 90        },
 91        { /* ch6 */
 92            &DMAC.SAR6.LONG,
 93            &DMAC.DAR6.LONG,
 94            &DMAC.DMATCR6.LONG,
 95            &DMAC.CHCR6.LONG,
 96            &DMAC.RSAR6.LONG,
 97            &DMAC.RDAR6.LONG,
 98            &DMAC.RDMATCR6.LONG,
 99        },
100        { /* ch7 */
101            &DMAC.SAR7.LONG,
102            &DMAC.DAR7.LONG,
103            &DMAC.DMATCR7.LONG,
104            &DMAC.CHCR7.LONG,
105            &DMAC.RSAR7.LONG,
106            &DMAC.RDAR7.LONG,
107            &DMAC.RDMATCR7.LONG,
108        }};
109
```

Figure 14 Sample Program Listing: dmac.c (2)

```
110    const ST_DMA_REG DmaReg[8] = {
111        { /* ch0 */
112            0ul,            /* DMA channel control registers (CHCR) */
113            0u,             /* DMA extension resource selectors (DMARS) */
114            0u              /* Interrupt priority register (IPR) */
115        },
116        { /* ch1 */
117            DMA_RLDSAR_ON | DMA_RLDDAR_OFF | DMA_TE_CONT | DMA_DAR_FIX | DMA_SAR_INC | DMA_ENH_RES
118            | DMA_SIZE_LONG | DMA_INT_ENABLE | DMA_DISABLE,
119            DMARS_SSI_0,
120            0x0fu
121        },
122        { /* ch2 */
123            DMA_RLDSAR_OFF | DMA_RLDDAR_ON | DMA_TE_STOP | DMA_DAR_INC | DMA_SAR_FIX | DMA_ENH_RES
124            | DMA_SIZE_LONG | DMA_INT_ENABLE | DMA_ENABLE,
125            DMARS_SRC_FULL,
126            0x0au
127        },
128        { /* ch3 */
129            DMA_RLDSAR_OFF | DMA_RLDDAR_OFF | DMA_TE_STOP | DMA_DAR_FIX | DMA_SAR_INC | DMA_ENH_RES
130            | DMA_SIZE_LONG | DMA_INT_ENABLE | DMA_ENABLE,
131            DMARS_SRC_EMP,
132            0x01u
133        },
134        { /* ch4 */
135            DMA_RLDSAR_OFF | DMA_RLDDAR_ON | DMA_TE_STOP | DMA_DAR_INC | DMA_SAR_FIX | DMA_AUTO_REQ
136            | DMA_SIZE_LONG | DMA_INT_ENABLE | DMA_DISABLE,
137            0u,
138            0x0au
139        },
140        { /* ch5 */
141            0ul,
142            0u,
143            0u
144        },
145        { /* ch6 */
146            0ul,
147            0u,
148            0u
149        },
150        { /* ch7 */
151            0ul,
152            0u,
153            0u
154        }};
155
156    /* ==== Prototype declaration ==== */
157    void io_init_dma_reg(unsigned int ch,void *src, void *r_src,void *dst, void *r_dst, size_t size);
```

Figure 15 Sample Program Listing: dmac.c (3)

```
158    /*""FUNC COMMENT""*************************************************
159    * Outline      : DMA initialization
160    *-----------------------------------------------------------------------
161    * Include      : #include <iodefine.h>
162    *-----------------------------------------------------------------------
163    * Declaration  : void io_init_dma_src(void);
164    *-----------------------------------------------------------------------
165    * Function     : Initializes channels 1 to 4.
166    *-----------------------------------------------------------------------
167    * Argument     : void
168    *-----------------------------------------------------------------------
169    * Return Value : void
170    *-----------------------------------------------------------------------
171    * Notice       :
172    *""FUNC COMMENT END""*************************************************/
173    void io_init_dma_src(void)
174    {
175        static unsigned int src_clr = 0x00000000ul;
176
177        /* ====Set standby control register 2(STBCR2) ==== */
178        CPG.STBCR2.BIT.MSTP8 = 0u;                  /* Cancel DMAC module stop */
179
180        /* ==== Initialize DMAC and enable transfer ==== */
181        /* ---- Buffer clear ---- */
182        io_init_dma_reg( 4,                         /* channel */
183                        &src_clr,                   /* source address */
184                        &src_clr,                   /* reload source address */
185                        get_buf_adr_proc(BUF_CLR),  /* destination address */
186                        get_buf_adr_proc(BUF_CLR),  /* reload destination address */
187                        src_out_buf_size());        /* Number of bytes */
188        /* ---- SRC input ---- */
189        io_init_dma_reg( 3,                         /* channel */
190                        rom_ding22_05_bin,          /* source address */
191                        rom_ding22_05_bin,          /* reload source address */
192                        (void *)&SRC.SRCID,         /* destination address */
193                        (void *)&SRC.SRCID,         /* reload destination address */
194                        size_ding22_05_bin);        /* Number of bytes */
195        /* ---- SRC output ---- */
196        io_init_dma_reg( 2,                         /* channel */
197                        (void *)&SRC.SRCOD,         /* source address */
198                        (void *)&SRC.SRCOD,         /* reload source address */
199                        get_buf_adr_proc(BUF_SRC),  /* destination address */
200                        get_buf_adr_proc(BUF_SRC),  /* reload destination address */
201                        src_out_buf_size());        /* Number of bytes */
202        /* ---- SSI transmission ---- */
203        io_init_dma_reg( 1,                         /* channel */
204                        get_buf_adr_proc(BUF_SSI),  /* source address */
205                        get_buf_adr_proc(BUF_SSI),  /* reload source address */
206                        (void *)&SSI0.SSITDR,       /* destination address */
207                        (void *)&SSI0.SSITDR,       /* reload destination address */
208                        src_out_buf_size());        /* Number of bytes */
209
210        /* ---- Set DMA operation register ---- */
211        DMAC.DMAOR.WORD &= 0xfff9u;                 /* Clear the AE and NMIF bits */
212
213        if(DMAC.DMAOR.BIT.DME == 0u){               /* Enable DMA transfer on all channels */
214                DMAC.DMAOR.BIT.DME = 1u;
215        }
216    }
```

Figure 16 Sample Program Listing: dmac.c (4)

```
217    /*""FUNC COMMENT""*************************************************
218    * Outline      : DMA initial setting
219    *-----------------------------------------------------------------------
220    * Include      : #include <iodefine.h>
221    *              : #include "dmac.h"
222    *-----------------------------------------------------------------------
223    * Declaration  : void io_init_dma_reg(unsigned int ch,void *src, void *r_src,
224    *              : void *dst, void *r_dst, size_t size);
225    *-----------------------------------------------------------------------
226    * Function     : Transfers data for the number of bytes specified by “size”
227    *              : from source address src to destination address dst using the DMAC.
228    *-----------------------------------------------------------------------
229    * Argument     : unsigned int: channel
230    *              : void *src : source address
231    *              : void *r_src : reload source address
232    *              : void *dst : destination address
233    *              : void *r_dst : reload destination address
234    *              : size_t size : Transfer size (byte)
235    *-----------------------------------------------------------------------
236    * Return Value : void
237    *-----------------------------------------------------------------------
238    * Notice       : If the transfer size and source/destination address alignment
239    *              : do not match, correct operation is not guaranteed.
240    *              : To use an interrupt, the interrupt routine should be registered.
241    *""FUNC COMMENT END""*************************************************/
242    void io_init_dma_reg(unsigned int ch,void *src, void *r_src,void *dst, void *r_dst, size_t size)
243    {
244        unsigned long ts = DmaReg[ch].chcr & 0x00000018ul;
245        unsigned long ie = DmaReg[ch].chcr & 0x00000004ul;
246        unsigned long de = DmaReg[ch].chcr & 0x00000001ul;
247        unsigned long rs = DmaReg[ch].chcr & 0x00000f00ul;
248
249        /* ---- Set DMA channel control registers ---- */
250        *DMA[ch].CHCR = 0ul; /* Disable DMA transfer */
251        /* ---- Set DMA source address register ---- */
252        *DMA[ch].SAR = (unsigned long)src;
253        /* ---- Set DMA reload source address register ---- */
254        *DMA[ch].RSAR = (unsigned long)r_src;
255        /* ---- Set DMA destination address register ---- */
256        *DMA[ch].DAR = (unsigned long)dst;
257        /* ---- Set DMA reload destination address register ---- */
258        *DMA[ch].RDAR = (unsigned long)r_dst;
259        /* ---- Set DMA transfer count register ---- */
260        /* ---- Set DMA reload transfer count register ---- */
261        switch(ts){
262        case DMA_SIZE_BYTE:
263            *DMA[ch].DMATCR = size;              /* Set transfer count (1/1) */
264            *DMA[ch].RDMATCR = size;
265            break;
266        case DMA_SIZE_WORD:
267            *DMA[ch].DMATCR = size >> 1u;        /* Set transfer count (1/2) */
268            *DMA[ch].RDMATCR = size >> 1u;
269            break;
270        case DMA_SIZE_LONG:
271            *DMA[ch].DMATCR = size >> 2u;        /* Set transfer count (1/4) */
272            *DMA[ch].RDMATCR = size >> 2u;
273            break;
274        case DMA_SIZE_LONGx4:
275            *DMA[ch].DMATCR = size >> 4u;        /* Set transfer count (1/16) */
276            *DMA[ch].RDMATCR = size >> 4u;
277            break;
278        default:
279            break;
280        }
```

Figure 17 Sample Program Listing: dmac.c (5)

```
281         /* ---- Set DMA channel control registers ---- */
282         *DMA[ch].CHCR = DmaReg[ch].chcr & 0xfffffffeul;
283         /*
284             bit31       : TC :0--------------------       1 Transfers data once
285             bit30       : reserve 0
286             bit29       : RLDSAR : x-----------------     Enables the function to reload SAR
287             bit28       : RLDDAR : x-----------------     Disables the function to reload DAR
288             bit27-24    : reserve 0
289             bit23       : DO over run0 : 0-----------     Not used
290             bit22       : TL TEND low active : 0-----     Not used
291             bit21       : reserve 0
292             bit20       : TEMASK : x-----------------     Stops DMA transfer or continues DMA transfer
293             bit19       : HE :0----------------------     Not used
294             bit18       : HIE :0---------------------     Not used
295             bit17       : AM :0----------------------     Not used
296             bit16       : AL :0----------------------     Not used
297             bit15-14    : DM1:x DM0:x----------------     Destination address is fixed/increased/decreased
298             bit13-12    : SM1:x SM0:x----------------     Source address is fixed/increased/decreased
299             bit11-8     : RS : B'xxxx----------------     Resource selector
300             bit7        : DL : DREQ level : 0 -------     Not used
301             bit6        : DS : DREQ select :0 Low level   Not used
302             bit5        : TB : cycle :0---------------    Cycle steal mode
303             bit4-3      : TS : transfer size:B'10---      Longword unit
304             bit2        : IE : interrupt enable:1---      Enables an interrupt request
305             bit1        : TE : transfer end----------
306             bit0        : DE : DMA enable bit:0-----      DMA extension resource selector
307         */
308         /* ---- Set DMA extension resource selectors ---- */
309         if(rs == DMA_ENH_RES){
310             switch(ch){
311             case 0:
312                     DMAC.DMARS0.BYTE.CH0 = (unsigned char)DmaReg[ch].dmars;
313                     break;
314             case 1:
315                     DMAC.DMARS0.BYTE.CH1 = (unsigned char)DmaReg[ch].dmars;
316                     break;
317             case 2:
318                     DMAC.DMARS1.BYTE.CH2 = (unsigned char)DmaReg[ch].dmars;
319                     break;
320             case 3:
321                     DMAC.DMARS1.BYTE.CH3 = (unsigned char)DmaReg[ch].dmars;
322                     break;
323             case 4:
324                     DMAC.DMARS2.BYTE.CH4 = (unsigned char)DmaReg[ch].dmars;
325                     break;
326             case 5:
327                     DMAC.DMARS2.BYTE.CH5 = (unsigned char)DmaReg[ch].dmars;
328                     break;
329             case 6:
330                     DMAC.DMARS3.BYTE.CH6 = (unsigned char)DmaReg[ch].dmars;
331                     break;
332             case 7:
333                     DMAC.DMARS3.BYTE.CH7 = (unsigned char)DmaReg[ch].dmars;
334                     break;
335             default:
336                     break;
337             }
338     }
```

Figure 18 Sample Program Listing: dmac.c (6)

```
339         /* ---- Set interrupt priority level ---- */
340         if(ie == DMA_INT_ENABLE){
341             switch(ch){
342             case 0:
343                     INTC.IPR06.BIT._DMAC0 = (unsigned short)DmaReg[ch].ipr;
344                     break;
345             case 1:
346                     INTC.IPR06.BIT._DMAC1 = (unsigned short)DmaReg[ch].ipr;
347                     break;
348             case 2:
349                     INTC.IPR06.BIT._DMAC2 = (unsigned short)DmaReg[ch].ipr;
350                     break;
351             case 3:
352                     INTC.IPR06.BIT._DMAC3 = (unsigned short)DmaReg[ch].ipr;
353                     break;
354             case 4:
355                     INTC.IPR07.BIT._DMAC4 = (unsigned short)DmaReg[ch].ipr;
356                     break;
357             case 5:
358                     INTC.IPR07.BIT._DMAC5 = (unsigned short)DmaReg[ch].ipr;
359                     break;
360             case 6:
361                     INTC.IPR07.BIT._DMAC6 = (unsigned short)DmaReg[ch].ipr;
362                     break;
363             case 7:
364                     INTC.IPR07.BIT._DMAC7 = (unsigned short)DmaReg[ch].ipr;
365                     break;
366             default:
367                     break;
368             }
369         }
370
371         /* ---- Perform DMA transfer ---- */
372         *DMA[ch].CHCR |= de;                 /* Enable DMA transfer */
373
374     }
```

Figure 19 Sample Program Listing: dmac.c (7)

```
375    /*""FUNC COMMENT""************************************************
376    * Outline      : DMA transfer end interrupt [SSI transmission]
377    *-----------------------------------------------------------------
378    * Include      : #include "iodefine.h"
379    *-----------------------------------------------------------------
380    * Declaration  : void io_int_dma1(void);
381    *-----------------------------------------------------------------
382    * Function     : Updates the reload register, and processes interrupts.
383    *-----------------------------------------------------------------
384    * Argument     : void
385    *-----------------------------------------------------------------
386    * Return Value : void
387    *-----------------------------------------------------------------
388    * Notice       :
389    *""FUNC COMMENT END""*********************************************/
390    void io_int_dma1(void)
391    {
392        volatile unsigned long dummy;
393        /* ---- Set DMA reload source address registers ---- */
394        DMAC.RSAR1.LONG = (unsigned long)get_buf_adr_proc(BUF_SSI);
395
396        (*trans_cmplt_hdr[BUF_SSI])();           /* Interrupt processing */
397
398        DMAC.CHCR1.BIT.TE = 0ul;
399        dummy = DMAC.CHCR1.BIT.TE;
400    }
401    /*""FUNC COMMENT""************************************************
402    * Outline      : DMA transfer end interrupt [SRC output]
403    *-----------------------------------------------------------------
404    * Include      : #include <iodefine.h>
405    *-----------------------------------------------------------------
406    * Declaration  : void io_int_dma2(void);
407    *-----------------------------------------------------------------
408    * Function     : Updates the reload register, and processes interrupts.
409    *-----------------------------------------------------------------
410    * Argument     : void
411    *-----------------------------------------------------------------
412    * Return Value : void
413    *-----------------------------------------------------------------
414    * Notice       :
415    *""FUNC COMMENT END""*********************************************/
416    void io_int_dma2(void)
417    {
418        volatile unsigned long dummy;
419        /* ---- Set DMA reload destination address registers ---- */
420        DMAC.RDAR2.LONG = (unsigned long)get_buf_adr_proc(BUF_SRC);
421
422        (*trans_cmplt_hdr[BUF_SRC])();           /* Interrupt processing */
423
424        DMAC.CHCR2.BIT.TE = 0ul;
425        dummy = DMAC.CHCR2.BIT.TE;
426    }
```

Figure 20 Sample Program Listing: dmac.c (8)

```
427    /*""FUNC COMMENT""************************************************
428    * Outline      : DMA transfer end interrupt [SRC input]
429    *-----------------------------------------------------------------
430    * Include      : #include <iodefine.h>
431    *-----------------------------------------------------------------
432    * Declaration  : void io_int_dma3(void);
433    *-----------------------------------------------------------------
434    * Function     : Disables DMA transfer, and performs SRC flash processing.
435    *-----------------------------------------------------------------
436    * Argument     : void
437    *-----------------------------------------------------------------
438    * Return Value : void
439    *-----------------------------------------------------------------
440    * Notice       :
441    *""FUNC COMMENT END""*********************************************/
442    void io_int_dma3(void)
443    {
444        volatile unsigned long dummy;
445
446        DMAC.CHCR3.BIT.DE = 0ul;          /* Stop the DMA ch3 (interrupt source: SRC input)*/
447
448        io_src_flash_start();             /* Perform flash processing */
449
450        DMAC.CHCR3.BIT.TE = 0ul;
451        dummy = DMAC.CHCR3.BIT.TE;
452    }
453    /*""FUNC COMMENT""************************************************
454    * Outline      : DMA transfer end interrupt [buffer clear]
455    *-----------------------------------------------------------------
456    * Include      : #include <iodefine.h>
457    *-----------------------------------------------------------------
458    * Declaration  : void io_int_dma4(void);
459    *-----------------------------------------------------------------
460    * Function     : Updates the reload register, and processes interrupts.
461    *-----------------------------------------------------------------
462    * Argument     : void
463    *-----------------------------------------------------------------
464    * Return Value : void
465    *-----------------------------------------------------------------
466    * Notice       :
467    *""FUNC COMMENT END""*********************************************/
468    void io_int_dma4(void)
469    {
470        volatile unsigned long dummy;
471        /* ---- DMA reload destination address registers ---- */
472        DMAC.RDAR4.LONG = (unsigned long)get_buf_adr_proc(BUF_CLR);
473
474        (*trans_cmplt_hdr[BUF_CLR])();        /* Interrupt processing */
475
476        DMAC.CHCR4.BIT.TE = 0ul;
477        dummy = DMAC.CHCR4.BIT.TE;
478    }
479
480    /* End of File */
```

Figure 21 Sample Program Listing: dmac.c (9)

```
/*""FILE COMMENT""*****************************************************************
*
*       System Name     : SH7263 Sample Program
*       File Name       : main.c
*       Contents        : SRC Sample Program
*       Version         : 1.00.00
*       Model           : R0K572630D001BR
*       CPU             : SH7263
*       Compiler        : SHC9.1.1.0
*       note            :
*
*                           Note
*                           This sample program is for reference
*                           and its operation is not guaranteed.
*                           Customers should use this sample program for technical reference
*                           in software development
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       history         : 2008.05.12 ver.1.00.00
*""FILE COMMENT END""*************************************************************/
#include <string.h>
#include "iodefine.h"          /* iodefine.h is a file automatically created by HEW */
#include "src.h"

/* ==== Prototype declaration ==== */
void io_init_src(unsigned int in_sampling,unsigned int out_sampling);

```

Figure 22 Sample Program Listing: src.c (1)

```
35    /*""FUNC COMMENT""************************************************
36    * Outline      : SRC register initialization
37    *-----------------------------------------------------------------------
38    * Include      : #include <iodefine.h>
39    *-----------------------------------------------------------------------
40    * Declaration  : void io_init_src(unsigned int in_sampling,
41    *              : unsigned int out_sampling);
42    *-----------------------------------------------------------------------
43    * Function     : Initializes the SRC registers, and enables the SRC module operation.
44    *-----------------------------------------------------------------------
45    * Argument     : unsigned int in_sampling : Input sampling rate
46    *              : unsigned int out_sampling : Output sampling rate
47    *-----------------------------------------------------------------------
48    * Return Value : void
49    *-----------------------------------------------------------------------
50    * Notice       :
51    *""FUNC COMMENT END""*************************************************/
52    void io_init_src(unsigned int in_sampling,unsigned int out_sampling)
53    {
54        /* ==== Supply clock to SRC ==== */
55        CPG.STBCR6.BIT.MSTP62 = 0u;
56
57        /* ==== Set SRC control register ==== */
58        SRC.SRCCTRL.WORD = 0x0500u | (in_sampling << 4u) | out_sampling;
59                /*
60                    bit15-13 : reserve : 0
61                    bit12 : SRCEN : 0 -----------  Disable the SRC module operation
62                    bit11 : reserve : 0
63                    bit10 : EEN : 1 -------------  Enable the output FIFO overwrite interrupt.
64                    bit9 : FL : 0 --------------
65                    bit8 : CL : 1 --------------   Clear the internal work memory
66                    bit7-4 : IFS : B'0110 -------- Input sampling rate: 24.0 kHz
67                    bit3-1 : reserve : 0
68                    bit0 : OFS : 0 -------------   Output sampling rate: 44.1 kHz
69                */
70        /* ==== Set SRC input data control register ==== */
71        SRC.SRCIDCTRL.WORD = 0x0103u;
72                /*
73                    bit15-10 : reserve : 0
74                    bit9 : IED : 0 -------------   Input data: big endian
75                    bit8 : IEN : 1 -------------   Input data empty interrupt is enabled
76                    bit7-2 : reserve : 0
77                    bit1-0 : IFTRG : B'11 -------- Input FIFO data triggering number: 12
78                */
79        /* ==== Set SRC output data control register ==== */
80        SRC.SRCODCTRL.WORD = 0x0100u;
81                /*
82                    bit15-11 : reserve : 0
83                    bit10 : OCH : 0 -------------  Does not exchange the channels
84                    bit9 : OED : 0 -------------   Output data: big endian
85                    bit8 : OEN : 1 -------------   Enable the output FIFO data full interrupt
86                    bit7-2 : reserve : 0
87                    bit1-0 : OFTRG : B'00 -------- Output FIFO data triggering number: 1
88
89                */
90        /* ==== Set interrupt priority level ==== */
91        INTC.IPR17.BIT._SRC = 0x01u;
92
93        SRC.SRCCTRL.BIT.SRCEN = 1u;       /* Enable the SRC module operation */
94    }
```

Figure 23 Sample Program Listing: src.c (2)

```
  95    /*""FUNC COMMENT""*************************************************
  96    * Outline      : Output FIFO overflow interrupt processing
  97    *-----------------------------------------------------------------
  98    * Include      :
  99    *-----------------------------------------------------------------
 100    * Declaration  : void io_int_src_out_ovr(void);
 101    *-----------------------------------------------------------------
 102    * Function     :
 103    *-----------------------------------------------------------------
 104    * Argument     : void
 105    *-----------------------------------------------------------------
 106    * Return Value : void
 107    *-----------------------------------------------------------------
 108    * Notice       :
 109    *""FUNC COMMENT END""*********************************************/
 110    void io_int_src_out_ovr(void)
 111    {
 112    }
 113    /*""FUNC COMMENT""*************************************************
 114    * Outline      : Output FIFO full interrupt processing
 115    *-----------------------------------------------------------------
 116    * Include      : #include <iodefine.h>
 117    *-----------------------------------------------------------------
 118    * Declaration  : void io_int_src_out_full(void);
 119    *-----------------------------------------------------------------
 120    * Function     : Clears the output FIFO full interrupt request bit.
 121    *-----------------------------------------------------------------
 122    * Argument     : void
 123    *-----------------------------------------------------------------
 124    * Return Value : void
 125    *-----------------------------------------------------------------
 126    * Notice       :
 127    *""FUNC COMMENT END""*********************************************/
 128    void io_int_src_out_full(void)
 129    {
 130        volatile unsigned long dummy;
 131
 132        SRC.SRCSTAT.BIT.OINT = 0u;
 133        dummy = SRC.SRCSTAT.BIT.OINT;
 134    }
 135    /*""FUNC COMMENT""*************************************************
 136    * Outline      : Input FIFO empty interrupt processing
 137    *-----------------------------------------------------------------
 138    * Include      : #include <iodefine.h>
 139    *-----------------------------------------------------------------
 140    * Declaration  : void io_int_src_in_emp(void);
 141    *-----------------------------------------------------------------
 142    * Function     : Clears the input FIFO empty interrupt request bit.
 143    *-----------------------------------------------------------------
 144    * Argument     : void
 145    *-----------------------------------------------------------------
 146    * Return Value : void
 147    *-----------------------------------------------------------------
 148    * Notice       :
 149    *""FUNC COMMENT END""*********************************************/
 150    void io_int_src_in_emp(void)
 151    {
 152        volatile unsigned long dummy;
 153
 154        SRC.SRCSTAT.BIT.IINT = 0u;
 155        dummy = SRC.SRCSTAT.BIT.IINT;
 156    }
```

Figure 24 Sample Program Listing: src.c (3)

```
157    /*""FUNC COMMENT""*****************************************************
158    * Outline      : SRC flash execution
159    *-----------------------------------------------------------------------
160    * Include      : #include <iodefine.h>
161    *-----------------------------------------------------------------------
162    * Declaration  : void io_src_flash_start(void);
163    *-----------------------------------------------------------------------
164    * Function     : Disables an input data empty interrupt request,
165    *              : and then performs flash processing.
166    *-----------------------------------------------------------------------
167    * Argument     : void
168    *-----------------------------------------------------------------------
169    * Return Value : void
170    *-----------------------------------------------------------------------
171    * Notice       :
172    *""FUNC COMMENT END""*************************************************/
173    void io_src_flash_start(void)
174    {
175        SRC.SRCIDCTRL.BIT.IEN = 0u;       /* Disable an input data empty interrupt */
176        SRC.SRCCTRL.BIT.FL = 1u;          /* Perform flash processing */
177    }
178    /*""FUNC COMMENT""*****************************************************
179    * Outline      : Output FIFO overflow interrupt request bit clear
180    *-----------------------------------------------------------------------
181    * Include      : #include <iodefine.h>
182    *-----------------------------------------------------------------------
183    * Declaration  : void io_src_ovf_clr(void);
184    *-----------------------------------------------------------------------
185    * Function     : Clears the output FIFO overflow interrupt request bit.
186    *-----------------------------------------------------------------------
187    * Argument     : void
188    *-----------------------------------------------------------------------
189    * Return Value : void
190    *-----------------------------------------------------------------------
191    * Notice       :
192    *""FUNC COMMENT END""*************************************************/
193    void io_src_ovf_clr(void)
194    {
195        volatile unsigned long dummy;
196
197        SRC.SRCSTAT.BIT.OVF = 0u;
198        dummy = SRC.SRCSTAT.BIT.OVF;
199    }
200
201    /* End of File */
```

Figure 25 Sample Program Listing: src.c (4)

```
/*""FILE COMMENT""*****************************************************************
*
*       System Name     : SH7263 Sample Program
*       File Name       : ssi.c
*       Contents        : SSI data transfer
*       Version         : 1.00.00
*       Model           : R0K572630D001BR
*       CPU             : SH7263
*       Compiler        : SHC9.1.1.0
*       note            : A sample program of data transfer using SSI0
*
*
*                           Note
*                           This sample program is for reference
*                           and its operation is not guaranteed.
*                           Customers should use this sample program for technical reference
*                           in software development
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       history         : 2008.05.09 ver.1.00.00
*""FILE COMMENT END""*************************************************************/
#include <string.h>
#include "iodefine.h"          /* iodefine.h is a file automatically created by HEW*/
#include "src.h"

/* ==== Prototype declaration ==== */
void io_init_ssi0(void);
void io_ssi0_set_mute(void);
void io_int_ssi0(void);

```

Figure 26 Sample Program Listing: ssi.c (1)

```
/*""FUNC COMMENT""*************************************************
* Outline      : SSI module initialization
*-----------------------------------------------------------------
* Include      : #include <iodefine.h>
*-----------------------------------------------------------------
* Declaration  : void io_init_ssi0(void);
*-----------------------------------------------------------------
* Function     : Transfers data in master transmitter mode.
*              : Sampling frequency is 44.1 kHz
*-----------------------------------------------------------------
* Argument     : void
*-----------------------------------------------------------------
* Return Value : void
*-----------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*************************************************/
void io_init_ssi0(void)
{
    /* ==== Supply clock to SSI module ==== */
    CPG.STBCR6.BIT.MSTP67 = 0u;           /* SSI0 */

    /* ==== Select oversampling clock supply source ==== */
    PORT.PFCRH4.BIT.PF30MD = 1u;          /* select AUDIO_CLK */
    PORT.SCSR.BIT.SSI0CKS = 0u ;          /* AUDIO_X1 input 16.9344 MHz */

    /* ----SSI module pin enabled ---- */
    PORT.PFCRH1.BIT.PF18MD = 1u;          /* SSISCK0 */
    PORT.PFCRH1.BIT.PF19MD = 1u;          /* SSIWS0 */
    PORT.PFCRH2.BIT.PF20MD = 1u;          /* SSIDATA0 */

    /* ==== Control register (SSICR) ==== */
    SSI0.SSICR.LONG = 0x1c0bd553ul;
        /*
            bit31-29    : reserve 0
            bit28       : DMEN : 1------------  DMA request is enabled
            bit27       : UIEN : 1------------  Underflow interrupt is enabled
            bit26       : OIEN : 1------------  Overflow interrupt is enabled
            bit25       : IIEN : 0------------  Idle mode interrupt is disabled
            bit24       : DIEN : 0------------  Data interrupt is disabled
            bit23-22    : CHNL : 0------------  Having one channel per system word
            bit21-19    : DWL : B'001---------  Data word length 16 bits
            bit18-16    : SWL : B'011---------  System word length 32 bits
            bit15       : SCKD : 1------------  Serial bit clock is output; master mode
            bit14       : SWSD : 1------------  Serial word select is output; master mode
            bit13       : SCKP : 0------------  SSIWS and SSIDATA change at the SSISCK rising edge
            bit12       : SWSP : 1------------  SSIWS is high for first channel, and low for second channel
            bit11       : SPDP : 0------------  Padding bits are low
            bit10       : SDTA : 1------------  Transmitting and receiving padding bits and serial data in this order
            bit9        : PDTA : 0------------  Not used
            bit8        : DEL : 1-------------  No delay between SSIWS and SSIDATA
            bit7        : reserve 0
            bit6-4      : CKDV : B'101--------  Oversampling clock frequency (16.9344 MHz) / 6[44.1 kHz]
            bit3        : MUEN : 0------------  SSI module is not muted
            bit2        : reserve 0
            bit1        : TRMD : 1------------  SSI module is in transmit mode
            bit0        : EN : 1 -------------  SSI module is enabled
        */
    INTC.IPR14.BIT._SSI0 = 0x01u;            /* Set the interrupt priority */
}
```

Figure 27 Sample Program Listing: ssi.c (2)

```
 97    /*""FUNC COMMENT""*************************************************
 98    * Outline      : SSI interrupt
 99    *-----------------------------------------------------------------------
100    * Include      : #include <iodefine.h>
101    *-----------------------------------------------------------------------
102    * Declaration  : void io_int_ssi0(void);
103    *-----------------------------------------------------------------------
104    * Function     : Clears interrupt requests such as an underflow error.
105    *-----------------------------------------------------------------------
106    * Argument     : void
107    *-----------------------------------------------------------------------
108    * Return Value : void
109    *-----------------------------------------------------------------------
110    * Notice       :
111    *""FUNC COMMENT END""*************************************************/
112    void io_int_ssi0(void)
113    {
114        /* Underflow error */
115        if(SSI0.SSISR.BIT.UIRQ == 1ul){
116            SSI0.SSISR.BIT.UIRQ = 0ul;
117            while(1){
118                /* dead loop */
119            }
120        }
121        /* Overflow error */
122        if(SSI0.SSISR.BIT.OIRQ == 1ul){
123            SSI0.SSISR.BIT.OIRQ = 0ul;
124            while(1){
125                /* dead loop */
126            }
127        }
128        /* Idle mode */
129        if(SSI0.SSISR.BIT.IIRQ == 1ul){
130            SSI0.SSISR.BIT.IIRQ = 0ul;
131        }
132    }
```

Figure 28 Sample Program Listing: ssi.c (3)

```
/*""FUNC COMMENT""*******************************************************
* Outline      : SSI mute setting
*-----------------------------------------------------------------------
* Include      : #include <iodefine.h>
*-----------------------------------------------------------------------
* Declaration  : void io_ssi0_set_mute(void);
*-----------------------------------------------------------------------
* Function     : Shifts SSI to the mute state.
*-----------------------------------------------------------------------
* Argument     : void
*-----------------------------------------------------------------------
* Return Value : void
*-----------------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*************************************************/
void io_ssi0_set_mute(void)
{
    if(SSI0.SSICR.BIT.EN == 1ul){
        /* ---- disable SSI interrupt ---- */
        SSI0.SSICR.BIT.UIEN = 0ul;

        /* ---- disable dreq ---- */
        SSI0.SSICR.BIT.DMEN =0ul;

        while(SSI0.SSISR.BIT.DIRQ == 0ul){
            /* ---- wait data req ---- */
        }
        SSI0.SSICR.BIT.MUEN = 1ul; /* mute start */
    }
}
/* End of File */
```

Figure 29 Sample Program Listing: ssi.c (4)

```
/*""FILE COMMENT""*****************************************************************
*
*       System Name     : SH7263 Sample Program
*       File Name       : buf.c
*       Contents        : Ring buffer
*       Version         : 1.00.00
*       Model           : R0K572630D001BR
*       CPU             : SH7263
*       Compiler        : SHC9.1.1.0
*       note            :
*
*                           Note
*                           This sample program is for reference
*                           and its operation is not guaranteed.
*                           Customers should use this sample program for technical reference
*                           in software development
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       history         : 2008.05.09 ver.1.00.00
*""FILE COMMENT END""*************************************************************/
#include <string.h>
#include "iodefine.h"
#include "src.h"
#pragma section SDRAM

/* ==== Buffer for storing data after modification of sampling rate ==== */
static unsigned long SrcOutBuffer[SRC_OUT_BUF_NUM][SRC_OUT_BUF_SIZE];

#pragma section
static int BufArea[3]={0,0,0};
static int FlgSrcStop = 0;

static void src_out_cnt_up(int* cnt,int max);
static void src_trans_cmplt_proc(void);
static void ssi_trans_cmplt_proc(void);
static void clr_trans_cmplt_proc(void);

unsigned long* get_buf_adr_proc(int module);
void (*trans_cmplt_hdr[])(void) = {
    clr_trans_cmplt_proc,
    src_trans_cmplt_proc,
    ssi_trans_cmplt_proc,
};

```

Figure 30 Sample Program Listing: buf.c (1)

```
/*""FUNC COMMENT""*************************************************
* Outline      : Buffer clear
*-----------------------------------------------------------------
* Include      : #include <string.h>
*-----------------------------------------------------------------
* Declaration  : void src_out_buf_clr(void);
*-----------------------------------------------------------------
* Function     : Clears all buffer areas.
*-----------------------------------------------------------------
* Argument     : void
*-----------------------------------------------------------------
* Return Value : void
*-----------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*********************************************/
void src_out_buf_clr(void)
{
    memset(SrcOutBuffer,0x00,sizeof(SrcOutBuffer));      /* Clear the buffer */
}
/*""FUNC COMMENT""*************************************************
* Outline      : Buffer address acquisition
*-----------------------------------------------------------------
* Include      :
*-----------------------------------------------------------------
* Declaration  : unsigned long* get_buf_adr_proc(int module);
*-----------------------------------------------------------------
* Function     : Returns a buffer address.
*-----------------------------------------------------------------
* Argument     : int module   : BUF_SRC SRC output
*              :              : BUF_SSI SSI transmission
*              :              : BUF_CLR Buffer clear
*-----------------------------------------------------------------
* Return Value : adr : Buffer address
*-----------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*********************************************/
unsigned long* get_buf_adr_proc(int module)
{
    unsigned long *adr;

    adr = SrcOutBuffer[BufArea[module]];                 /* Acquire an address */
    src_out_cnt_up(&BufArea[module],SRC_OUT_BUF_NUM);    /* Count up BufArea[module] */

    return(adr);
}
```

Figure 31 Sample Program Listing: buf.c (2)

```
 97    /*""FUNC COMMENT""*************************************************
 98    * Outline      : SRC output interrupt processing
 99    *-----------------------------------------------------------------------
100    * Include      : #include <iodefine.h>
101    *-----------------------------------------------------------------------
102    * Declaration  : static void src_trans_cmplt_proc(void);
103    *-----------------------------------------------------------------------
104    * Function     : When the next buffer is being cleared, disables DMA transfer
105    *              : [SRC output], and sets 1 to variable “FlgSrcStop”.
106    *              : Enables DMA transfer [SSI transmission] after storing PCM data in the
107    *              : first buffer.
108    *-----------------------------------------------------------------------
109    * Argument     : void
110    *-----------------------------------------------------------------------
111    * Return Value : void
112    *-----------------------------------------------------------------------
113    * Notice       :
114    *""FUNC COMMENT END""*************************************************/
115    static void src_trans_cmplt_proc(void)
116    {
117        if(BufArea[BUF_SRC] == BufArea[BUF_CLR]){ /* When the next buffer is being cleared
118                                                     is cleared, */
119            DMAC.CHCR2.BIT.DE = 0ul;              /* Stop DMA ch2 (interrupt source: SRC
120                                                     output) */
121            FlgSrcStop = 1;
122        }
123        if(DMAC.CHCR1.BIT.DE == 0ul){             /* Note: Only for the first time, */
124            DMAC.CHCR1.BIT.DE = 1ul;              /* after SRC output to the first buffer has
125                                                     completed, */
126                                                  /* enable DMA transfer [SSI transmission] */
127        }
128
129    }
130    /*""FUNC COMMENT""*************************************************
131    * Outline      : SSI transmission interrupt processing
132    *-----------------------------------------------------------------------
133    * Include      : #include <iodefine.h>
134    *-----------------------------------------------------------------------
135    * Declaration  : static void ssi_trans_cmplt_proc(void);
136    *-----------------------------------------------------------------------
137    * Function     : When transmission of all data is completed (all data
138    *              : has been transmitted), shifts the SSI to the mute state.
139    *-----------------------------------------------------------------------
140    * Argument     : void
141    *-----------------------------------------------------------------------
142    * Return Value : void
143    *-----------------------------------------------------------------------
144    * Notice       :
145    *""FUNC COMMENT END""*************************************************/
146    static void ssi_trans_cmplt_proc(void)
147    {
148        if(BufArea[BUF_SRC] == BufArea[BUF_SSI]){    /* End determination */
149            io_ssi0_set_mute();                      /* Set mute */
150        }
151        if(DMAC.CHCR4.BIT.DE == 0ul){
            DMAC.CHCR4.BIT.DE = 1ul;                 /* Enable DMA transfer [buffer clear] */
        }

    }
```

Figure 32 Sample Program Listing: buf.c (3)

```
/*""FUNC COMMENT""************************************************
* Outline      : Buffer clear interrupt processing
*-----------------------------------------------------------------
* Include      : #include <iodefine.h>
*-----------------------------------------------------------------
* Declaration  : static void clr_trans_cmplt_proc(void);
*-----------------------------------------------------------------
* Function     : When data in the next buffer is being tranmitted to the SSI,
*              : disables DMA transfer [buffer clear].
*              : If SRC output is stopped, resumes SRC output,
*              : and clears the overflow interrupt request bit.
*-----------------------------------------------------------------
* Argument     : void
*-----------------------------------------------------------------
* Return Value : void
*-----------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*********************************************/
static void clr_trans_cmplt_proc(void)
{
    if(BufArea[BUF_SSI] == BufArea[BUF_CLR]){    /* When data in the next buffer is being
tranmitted to the SSI                                          transmission,*/
        DMAC.CHCR4.BIT.DE = 0ul;                 /* Stop DMA ch4 (interrupt source:
                                                    software)*/

    }
    if(FlgSrcStop == 1){                         /* If SRC output is stopped */
        FlgSrcStop = 0;
        DMAC.CHCR2.BIT.DE = 1ul;                 /* Start DMA ch2 (interrupt source: SRC
                                                    output) */

        io_src_ovf_clr();                        /* Clear the SRC output overflow interrupt
                                                    request flag */

    }
}
/*""FUNC COMMENT""************************************************
* Outline      : Count up
*-----------------------------------------------------------------
* Include      :
*-----------------------------------------------------------------
* Declaration  : static void src_out_cnt_up(int* cnt,int max);
*-----------------------------------------------------------------
* Function     : Counts up from 0 to variable “max”.
*-----------------------------------------------------------------
* Argument     : int* cnt  : RAM for storing counts
*              : int max   : Maximum value
*-----------------------------------------------------------------
* Return Value : void
*-----------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*********************************************/
static void src_out_cnt_up(int* cnt,int max)
{
    if(*cnt < max - 1){
        *cnt += 1;
    }
    else{
        *cnt = 0;
    }
}
```

Figure 33 Sample Program Listing: buf.c (4)

```
207    /*""FUNC COMMENT""*****************************************************
208    * Outline      : SRC output buffer block size return
209    *-------------------------------------------------------------------------
210    * Include      :
211    *-------------------------------------------------------------------------
212    * Declaration  : size_t src_out_buf_size(void);
213    *-------------------------------------------------------------------------
214    * Function     : Returns block size of the SRC output buffer.
215    *-------------------------------------------------------------------------
216    * Argument     : void
217    *-------------------------------------------------------------------------
218    * Return Value : size_t size : SRC output buffer block size
219    *-------------------------------------------------------------------------
220    * Notice       :
221    *""FUNC COMMENT END""*************************************************/
222    size_t src_out_buf_size(void)
223    {
224        return(sizeof(SrcOutBuffer[0]));
225    }
226    /* End of File */
```

Figure 34 Sample Program Listing: buf.c (5)

```
    1    /************************************************************************/
    2    /*                                                                      */
    3    /* FILE :intprg.c                                                       */
    4    /* DATE :Wed, Nov 21, 2007                                              */
    5    /* DESCRIPTION :Interrupt Program                                       */
    6    /* CPU TYPE :SH7263                                                     */
    7    /*                                                                      */
    8    /* This file is generated by Renesas Project Generator (Ver.4.5).      */
    9    /*                                                                      */
   10    /************************************************************************/
   11
   12
   13
   14    #include <machine.h>
   15    #include "vect.h"
   16    #pragma section IntPRG
   17
         (Snip)
  427    // 112 DMAC1 TEI1
  428    void INT_DMAC1_TEI1(void)
  429    {
  430        extern void io_int_dma1(void);
  431        io_int_dma1();
  432    }
  433
         (Snip)
  443    // 116 DMAC2 TEI2
  444    void INT_DMAC2_TEI2(void)
  445    {
  446        extern void io_int_dma2(void);
  447        io_int_dma2();
  448    }
  449
         (Snip)
  461    // 120 DMAC3 TEI3
  462    void INT_DMAC3_TEI3(void)
  463    {
  464        extern void io_int_dma3(void);
  465        io_int_dma3();
  466    }
  467
         (Snip)
  478    // 124 DMAC4 TEI4
  479    void INT_DMAC4_TEI4(void)
  480    {
  481        extern void io_int_dma4(void);
  482        io_int_dma4();
  483    }
  484
         (Snip)
  987    // 214 SSI0
  988    void INT_SSI0(void)
  989    {
  990        extern void io_int_ssi0(void);
  991        io_int_ssi0();
  992    }
  993
```

Figure 35 Sample Program Listing: intprg.c (1)

```
        (Snip)
1167    // 244 SRC OVF
1168    void INT_SRC_OVF(void)
1169    {
1170        extern void io_int_src_out_ovr(void);
1171        io_int_src_out_ovr();
1172    }
1173
1174    // 245 SRC IDFI
1175    void INT_SRC_IDFI(void)
1176    {
1177        extern void io_int_src_out_full(void);
1178        io_int_src_out_full();
1179    }
1180
1181    // 246 SRC IDEI
1182    void INT_SRC_IDEI(void)
1183    {
1184        extern void io_int_src_in_emp(void);
1185        io_int_src_in_emp();
1186    }
1187    // 247 IEB IEBI
1188    void INT_IEB_IEBI(void)
1189    {
1190        /* sleep(); */
1191    }
1192    // 248 Dummy
1193    void Dummy(void)
1194    {
1195        /* sleep(); */
1196    }
1197
1198    /* End of File */
```

Figure 36 Sample Program Listing: intprg.c (2)

4. Documents for Reference

- Software Manual
  SH-2A, SH2A-FPU Software Manual (REJ09B0051)
  The most up-to-date version of this document is available on the Renesas Technology Website.
- Hardware Manual
  SH7263 Group Hardware Manual (REJ09B0290)
  The most up-to-date version of this document is available on the Renesas Technology Website.

Website and Support

Renesas Technology Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jan.19.09	—	First edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.