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.

M32C/80 Series

Digital Audio I/F driver using the peripheral functions

REJ05B0146-0110Z
Rev.1.10
Jul 25, 2003

1. Abstract

This application note introduces and shows a sample program of Digital Audio I/F driver using the peripheral functions on the M32C/80 series.

2. Introduction

The explanation of this issue is applied to the following condition:

Applicable MCU: M32C/80 series

Table of contents

1. Abstract 1
2. Introduction 1
3. Overview of Digital Audio I/F driver 3
3.1 Outline of H/W connection 3
3.2 Digital Audio Signal 4
3.3 Structure of the driver 5
3.3.1 DMAC 5
3.3.2 CSIO interrupt cause selection 6
3.3.3 Endian conversion in the case of using 8bits CSIO 6
4. Setting of peripheral functions 7
4.1 The peripheral functions which use and setting 7
4.2 Flow chart of the peripheral function setting 11
4.2.1 Flow chart of starting the audio signal (Using 8 bits CSIO) 11
4.2.2 Flow chart of stopping the audio signal (Using 8 bits CSIO) 12
4.2.3 Flow chart of starting the audio signal (Using 16 bits CSIO) 13
4.2.4 Flow chart of stopping the audio signal (Using 16 bits CSIO) 13
5. Driver function details 14
_dai_start 15
_dai_stop 16
_cnv_endian 17
Init_SCLK 17
Start_LRCK 17
Init_CSIO8 18
Start_DMA0 18

Start_CSIO8 18
Init_CSIO16 19
Start_DMA1 19
Start_CSIO16 19
6. Sample programs of Digital Audio I/F driver 20
dai_drv.h 20
C_dai_drv.c 21
A_endian.a30 36
7. Reference 37

3. Overview of Digital Audio I/F driver

Generally, the digital audio data which music CD (CD-DA) is treating is the following formats. The sampling frequency of data is 44.1kHz and the quantization bit is 16bits PCM (Pulse Code Modulation). By inputting this digital audio data to DAC (Digital Analog Converter) at the timing which DAC demands, digital audio data will be music.

This sample shows an example of Digital Audio I/F driver which can play music by using peripheral functions on MCU.

3.1 Outline of H/W connection

In this sample, assumes play music by inputting digital audio data into the outside DAC using CSIO (Clock Synchronous serial I/O). Figure 1 shows an example of H/W connection of audio interface parts which is assumed in this sample.

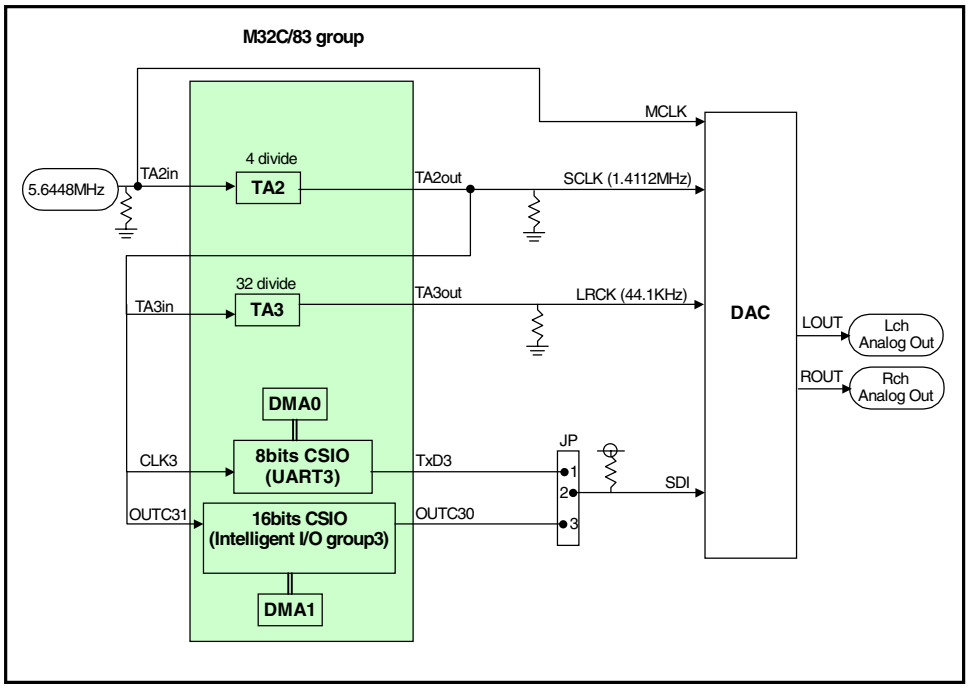


Figure 1 Example of H/W connection

[Note] 16bits CSIO can use only on the M32C/83 group.
16bits CSIO uses Intelligent I/O group 3.

- MCLK: The master clock for DAC
- SCLK: The serial clock for DAC (is used for clock synchronous serial transmission)
- LRCK: Lch, Rch Clock for DAC (is used for synchronization of Lch and Rch)
- SDI : Serial data (The digital audio data which input to DAC)

3.2 Digital Audio Signal

The specification of the digital audio signal format and timings of each signal assumed in this example are shown below.

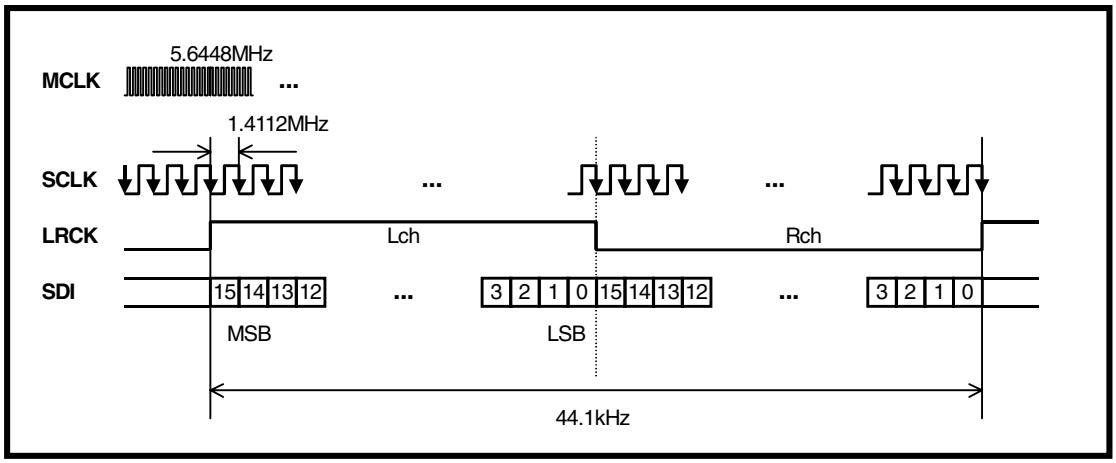


Figure 2 The format of digital audio signal

- Sampling frequency: 44.1 kHz
- Quantization bits: 16 bits
- Stereo PCM data

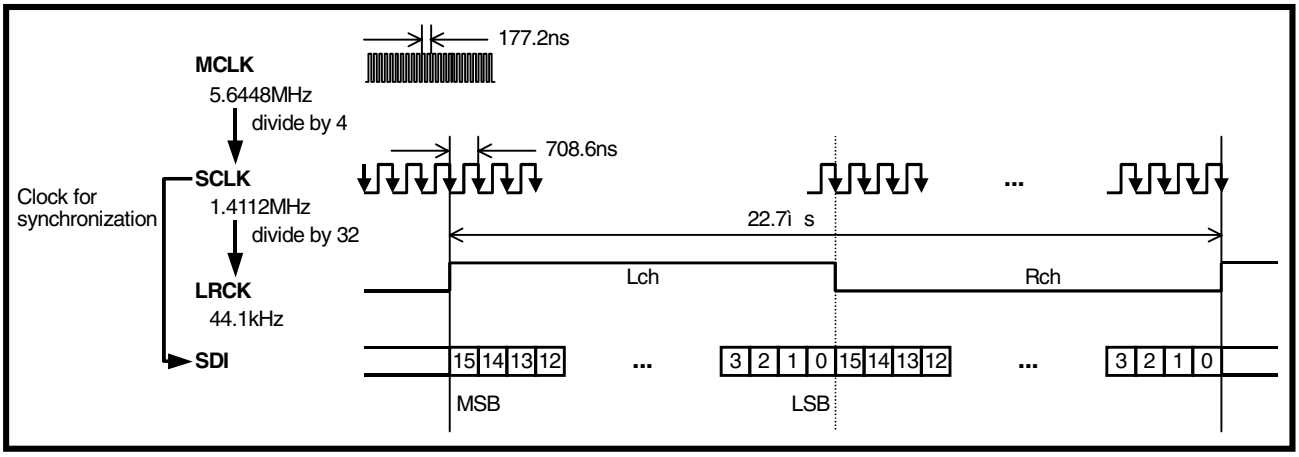


Figure 3 The timing of digital audio signal

- Polarity of LRCK changes at falling edge of SCLK.
- Synchronize data at falling edge of SCLK.

3.3 Structure of the driver

This section describes the structure of the digital audio I/F driver of this example.

3.3.1 DMAC

In this example, processing which transmits digital audio data to the DAC is realized combining DMAC and CSIO which the MCU contains. And, digital audio data to play shall be stored in storage devices, such as HDD. An example of the system from the viewpoint of data flow is shown below.

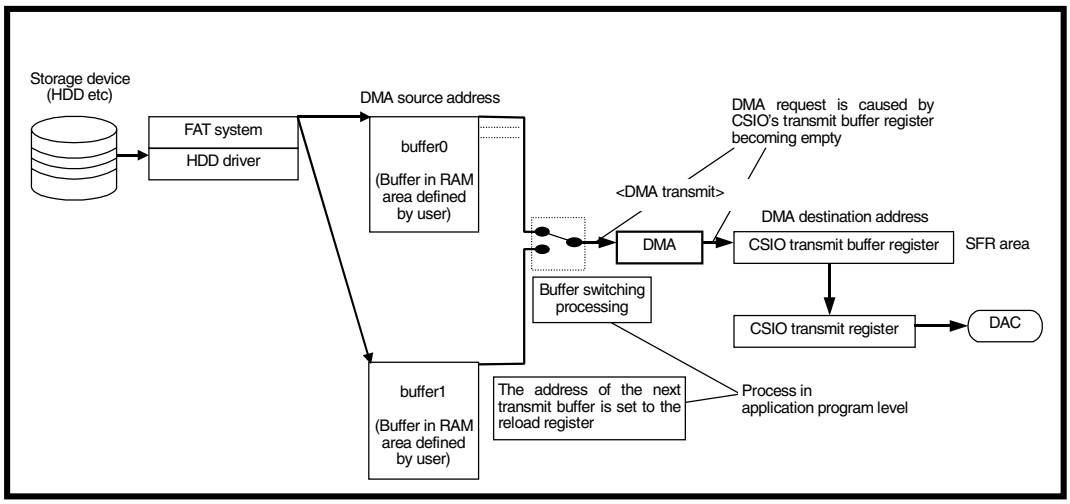


Figure 4 Example of data flow system

(1) As shown in Figure 4, digital audio data read from the storage device are stored in the buffer (RAM). The data which were stored in the buffer are transmitted to the CSIO transmit buffer register by DMAC.

(2) DMA source, destination address and DMA request factor are set up as follows.
DMA source address: buffer (RAM)
DMA destination address: CSIO transmission buffer register
DMA request factor: CSIO transmission interrupt request

(3) In order to maintain the continuity of a music signal, the two page buffers are prepared and using switching them in this example.
While transmitting data which stored in one buffer, store data from storage device into the other buffer.

(4) DMA transfer mode is set as the repeat transfer mode. And the reload-register of “transfer count" and “address" which DMAC of M32C/80 series has are used.
When outputting of the data stored in the buffer is completed, the DMA interrupt generates. The DMA transmission discontinuation at the time of switch processing of buffers is avoided by setting up reload-registers by the time DMA interruption occurs.

[Supplement]

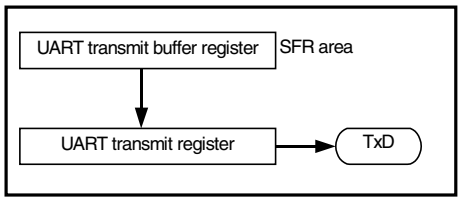
This example treats the buffer of two pages structure. Also it is possible to consider the system structure which has two or more pages buffers.

3.3.2 CSIO interrupt cause selection

The interrupt cause of clock synchronous serial I/O mode (CSIO) chooses “transmit buffer empty.”

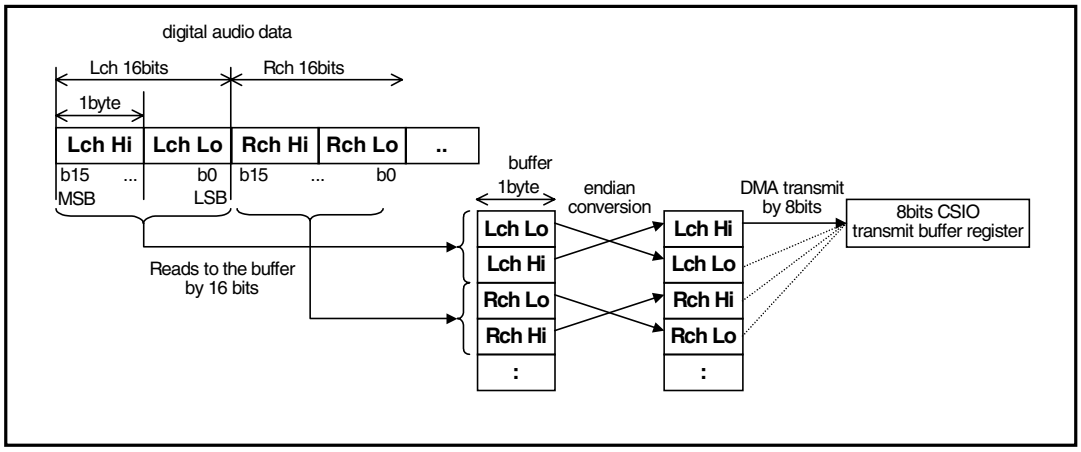
The registers of serial I/O are configured as two step composition of the “transmit buffer register” and the “transmit register”, as shown below.

In this example, if the “transmit buffer register" becomes empty, interruption of CSIO will occur and DMAC will transmit data to the “transmit buffer register”.



3.3.3 Endian conversion in the case of using 8bits CSIO

M32C/80 series microcomputer is little endian type. The digital audio data which treats in this example are 16 bits data per one channel and it’s structure shown below. So, endian conversion is necessity when data transmission is executed per 8 bits using 8bits CSIO.



4. Setting of peripheral functions

This example shows how to play music from digital audio data by using internal peripheral functions of microcomputer, shown in “Figure 1 Example of H/W connection.” This chapter explains setting of peripheral functions and setting flow chart which are used by the digital audio I/F driver that described system in “3 Overview of Digital Audio I/F driver.”

4.1 The peripheral functions which use and setting

The list of peripheral functions used in this example is shown in Table 1, and these settings are shown in Table 2 to Table 4.

Table 1 The list of peripheral functions which use in digital audio I/F driver

CSIO	Peripheral	Outline of the setting
CSIO8, CSIO16 common	TA2	Generate SCLK, Event counter mode
CSIO8, CSIO16 common	TA3	Generate LRCK, Event counter mode
CSIO8	UART3	Transfer data length: 8 bits Clock synchronous serial I/O mode, uses for transmitting SDI data. Transfer clock: external clock (SCLK)
CSIO8	DMA0	Transmit 8 bits data to the UART3 transmit buffer register Transfer mode: Repeat transfer Transfer direction: Memory (forward direction) → fixed address (UART3 transmit buffer register)
CSIO16	Intelligent I/O group 3	Transfer data length: 16 bits Clock synchronous serial I/O mode, uses for transmitting SDI data. Transfer clock: external clock (SCLK)
CSIO16	DMA1	Transmit 16 bits data to the transmit buffer register of intelligent I/O group 3. Transfer mode: Repeat transfer Transfer direction: Memory (forward direction) → fixed address (the transmit buffer register of intelligent I/O group 3) When using the Intelligent I/O Group 3, usable DMAC is restricted to DMA1 or DMA3.

Table 2 The setting of peripheral functions (8 bits CSIO, 16 bits CSIO common)

Peripheral	Uses	Item	Setting
TA2	Generate the SCLK by dividing the MCLK by 4.	Mode	Event counter mode
		Count source	Counts the rising edge of the external signal which inputted into TA2IN pin (MCLK→TA2IN pin) (Set the TA2IN pin as input port)
		Count operation	Up/down flag's content (Down count) Reload type
		Divide ratio	Generates the SCLK by dividing external clock (MCLK) by 4 and outputting pulse to TA2OUT pin. (By the pulse output function, at the time of underflow occurring, the polarity of TAiOUT pin is reversed. So, set 2 to the divide ratio.)
		TAiIN pin function	Sets TA2IN pin as input port.
		TAiOUTpin function	Pulse output (Sets TA2OUT output function by corresponding function select registers)
		Interrupt level	0
TA3	Generate the LRCK by dividing the SCLK by 32.	Mode	Event counter mode
		Count source	Counts the falling edge of the external signal (SCLK) which inputted into TA3IN pin. (Sets the TA3IN pin as input port)
		Count operation	Up/down flag's content (Down count) Reload type
		Divide ratio	Generates the LRCK by dividing external clock (SCLK) by 32 and outputting pulse to TA3OUT pin. (By the pulse output function, at the time of underflow occurring, the polarity of TAiOUT pin is reversed. So, set 16 to the divide ratio. But, only at starting point set 17 to the divide ratio to take synchronization of the SCLK, the LRCK and the SDI. (See Figure 5)
		TAiIN pin function	Sets the TA3IN pin as input port.
		TAiOUT pin function	Pulse output (Sets TA3OUT output function by corresponding function select registers)
		Interrupt level	0

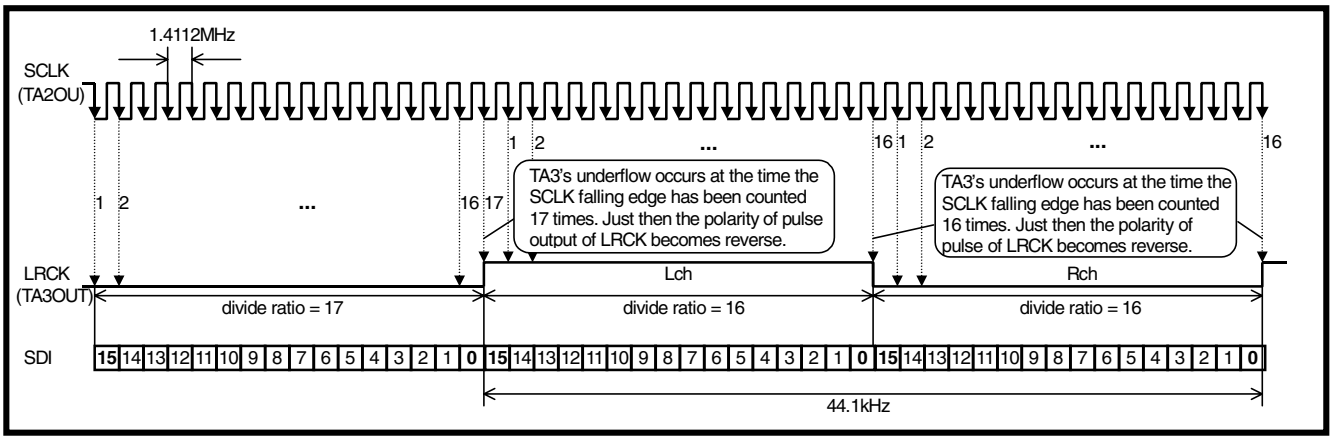


Figure 5 Setting the LRCK (TA3)

Table 3 The setting of peripheral functions (8 bits CSIO)

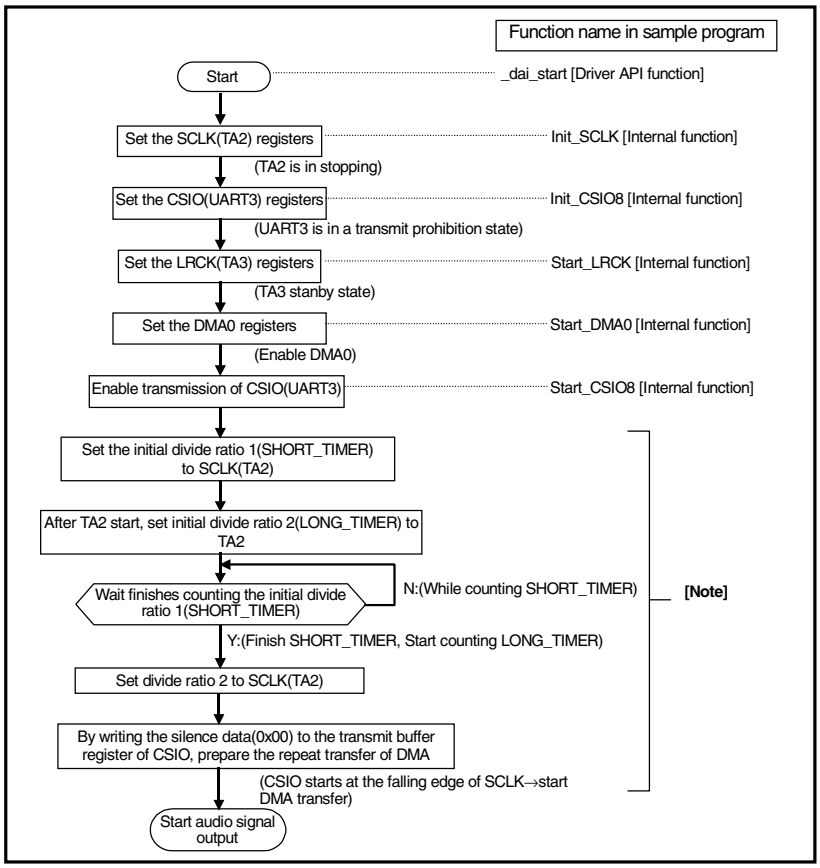
Peripheral	Uses	Item	Setting
UART3	Transmit digital audio data by clock synchronous transfer.	Mode	Clock synchronous serial I/O mode
		Transfer data format	Transfer data length 8 bits MSB first
		Transfer clock	External clock (input from CLK3 pin) (Sets the corresponding function select register A to I/O port)
		BRG count source	f1 (No divided)
		CLKi polarity select	Transmit data is output at falling edge of transfer clock
		TxD output	TxD output is selected by the corresponding function select register A, B and C.
		Interrupt cause select	Transmit buffer empty
		Other	CTS/RTS function disabled. Data logic select: No reverse
		Interrupt level	0
DMA0	Transmit data stored in the buffer to the transmit buffer register of UART3	Mode	Repeat transfer
		DMA request factor	UART3 transmission interrupt requests
		Transfer unit	8 bits
		Transfer direction	Memory → fixed address (UART3 transmit buffer register)
		Interrupt level	7

Table 4 The setting of peripheral functions (16 bits CSIO)

Peripheral	Uses	Item	Setting
Intelligent I/O group 3	Transmit digital audio data by clock synchronous transfer.	Mode	Clock synchronous serial I/O mode
		Transfer data format	Transfer data length: 16 bits (This function is usable only in the Intelligent I/O group 3) MSB first
		Transfer clock	External clock (input from ISCLK3 pin) (Set the corresponding function select register A to I/O port)
		BRG count source	f1 (No divided)
		CLKi polarity select	Unselectable (Transmit data is output only at falling edge of transfer clock)
		TxD output	II/O group3 output is selected by the corresponding function select register A, B and C.
		Interrupt cause select	Transmit buffer empty
		Other	Group 3 waveform generation control: Assigns communication output to a port. Group 3 function enable register: Enables function on ch0 and ch1.
		Interrupt level	0
DMA1	Transmit data stored in the buffer to the transmit buffer register of the Intelligent I/O group 3 SI/O	Mode	Repeat transfer
		DMA request factor	Intelligent I/O interrupt control register 10
		Transfer unit	16 bits
		Transfer direction	Memory→fixed address (Group 3 SI/O transmit buffer register)
		Interrupt level	7

4.2 Flow chart of the peripheral function setting

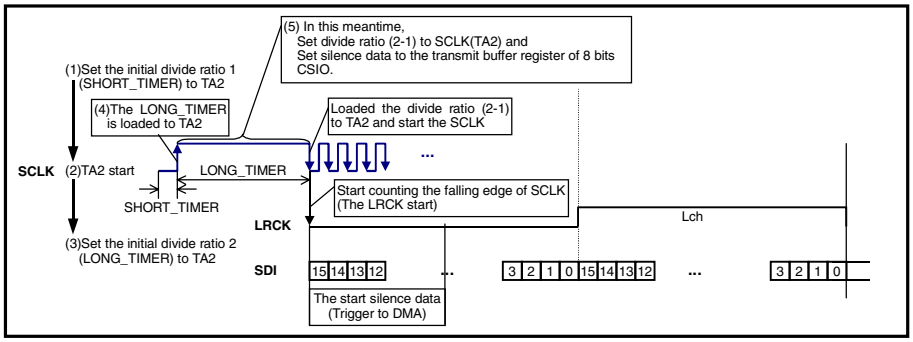
4.2.1 Flow chart of starting the audio signal (Using 8 bits CSIO)



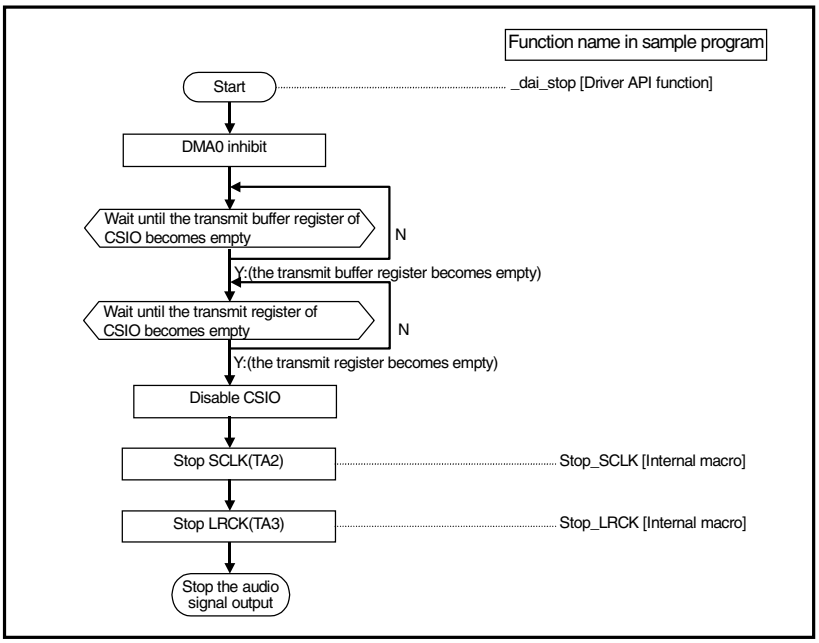
[Note] Precaution for using 8 bits CSIO

As transmission start condition, when selecting external clock and selecting falling edge to the CLKi polarity, it is necessity that CLKi pin level is H. So, the following methods are taken in this example:

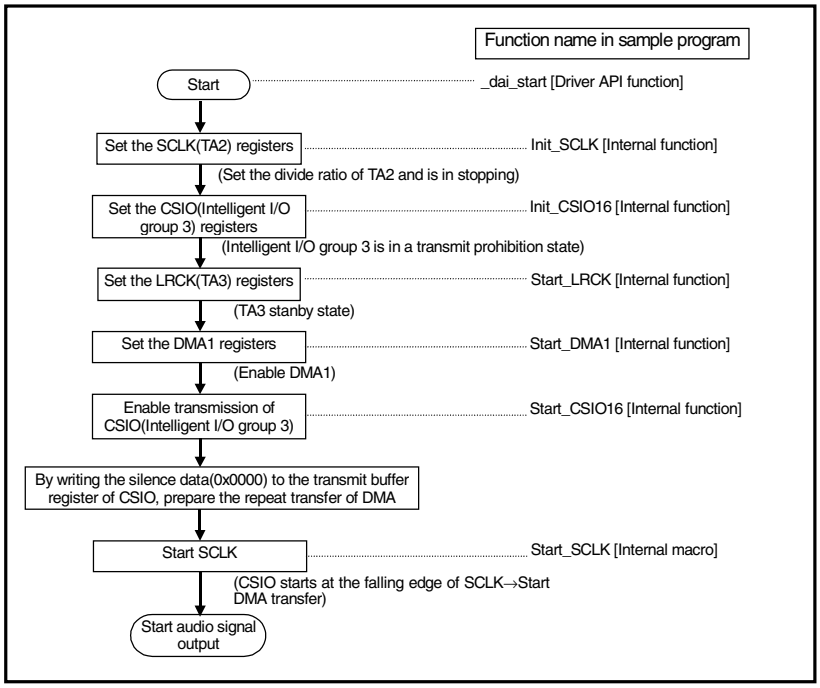
- In initializing the SCLK (TA2), once the level of CLKi (=SCLK=the output pulse of TA2) is lifted to H and in this meantime writes the start silence data as DMA trigger to the transmit buffer register of CSIO. Following figure shows the initialization parts of SCLK (pulse output of TA2).



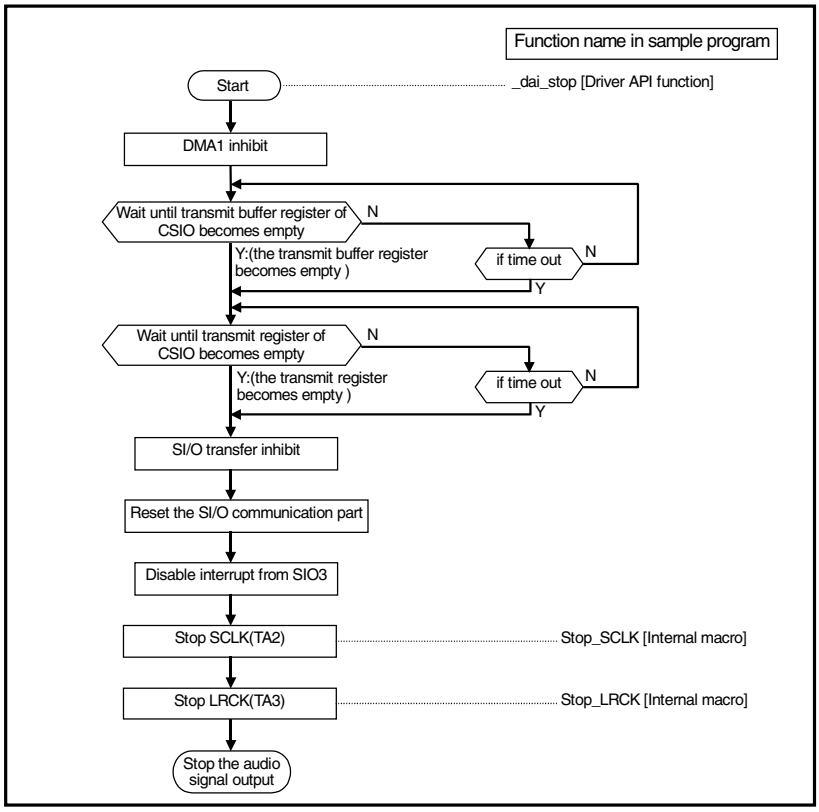
4.2.2 Flow chart of stopping the audio signal (Using 8 bits CSIO)



4.2.3 Flow chart of starting the audio signal (Using 16 bits CSIO)



4.2.4 Flow chart of stopping the audio signal (Using 16 bits CSIO)



5. Driver function details

In this chapter, each function of the sample digital audio I/F driver is detailed. The sample driver is classified as follows.

(1) API (Application Program Interface) function: The function which is called from application program

(2) Internal function and internal macro: The function (macro) which is called from within driver

Function name	Classification	Functional outline
_dai_start	API	●Starts outputting digital audio data streaming to the DAC.
_dai_stop	API	●Stops outputting digital audio data play. ●Disables DMA transfer and CSIO transfer. ●Stops SCLK and LRCK.
_cnv_endian	API (8 bits CSIO)	●Converts endian, when using 8 bits CSIO.
Init_SCLK	Internal function	●Sets the SCLK registers.
Start_SCLK	Internal function	●Sets the LRCK registers. ●The LRCK is in the start state.
Init_CSIO8	Internal function (8 bits CSIO)	●Sets the UART3 as 8 bits CSIO. ●CSIO8 uses the external clock SCLK as synchronous clock.
Start_DMA0	Internal function (8 bits CSIO)	●Sets the DMA when using 8 bits CSIO. ●DMA is in standby state as the repeat transfer mode. And transmission starts by the request by the interrupt of emptying the transmit buffer of CSIO.
Start_CSIO8	Internal function (8 bits CSIO)	●Enables transmission of CSIO that set by the function Init_CSIO8.
Init_CSIO16	Internal function (16 bits CSIO)	●Sets the Intelligent I/O group 3 as 16 bits CSIO. ●CSIO16 uses the external clock SCLK as synchronous clock.
Start_DMA1	Internal function (16 bits CSIO)	●Sets the DMA when using 16 bits CSIO. ●DMA is in standby state as the repeat transfer mode. And transmission starts by the request by the interrupt of emptying the transmit buffer of CSIO.
Start_CSIO16	Internal function (16 bits CSIO)	●Enables transmission of CSIO that set by the Init_CSIO16.

Function	**_dai_start**		
Classification	API		
Description	Start outputting digital audio data streaming to the DAC. This function needs to be called after storing following data in the user-defined buffers. *buf0 ← silence data (0 clearance) *buf1 ← digital audio data Starts outputting from data which stored in *buf0.		
Format	void _dai_start(unsigned short far *buf0, unsigned short far *buf1, unsigned short size, unsigned short zero_cnt)		
Parameter	*buf0	I	Top address of the buffer0 defined by user
	*buf1	I	Top address of the buffer1 defined by user
	size	I	Size of the buffer (= Size of data which stored in *buf1) (Specify in WORD unit)
	zero_cnt	I	Sets the number of outputting silence data (< Size of buffer) When using 8 bits CSIO: To the zero_cnt, set the value (1+4n; n=1,2,...) When using 16 bits CSIO: To the zero_cnt, set the value (2n; n=1,2,...) The relationship of zero_cnt and L/R channel output is shown in Figure 6.
Return	None		
Subroutines	Init_SCLK; Internal function Start_LRCK; Internal function Start_CSIO8/Start_CSIO16; Internal function Start_DMA0/Start_DMA1; Internal function		
Note	Do not set 0 to the parameter zero_cnt. The *buf0 is specified by the pointer of the buffer which stored silence data (=0h). The *buf1 is specified by the pointer of the buffer which stored digital audio data.		

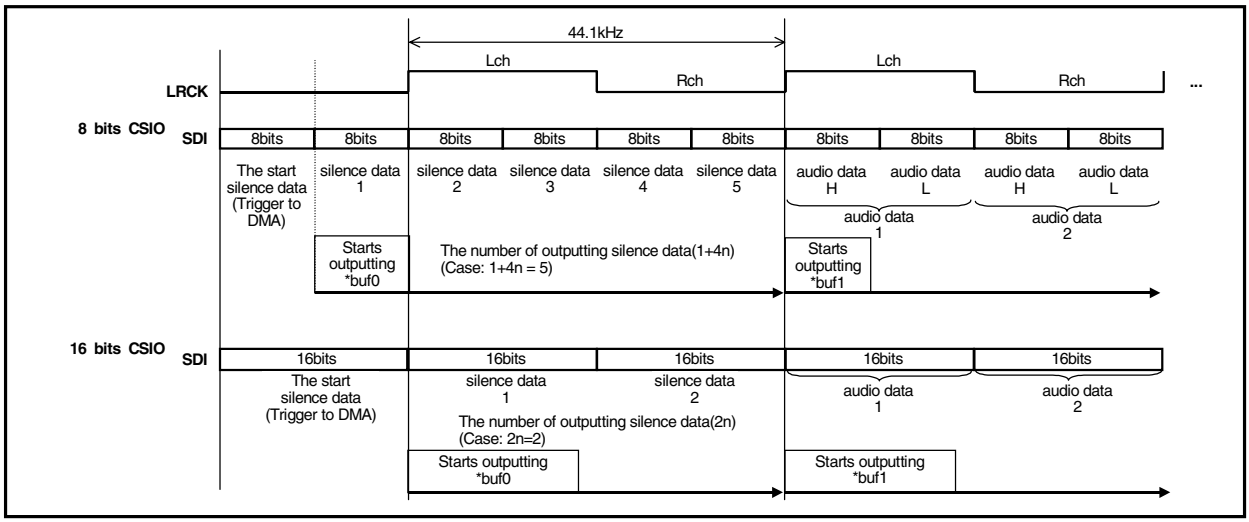


Figure 6. zero_cnt(silence data counter) and L/R channel output

[Explanation]

(1) Parameters of this function are set to following registers.

Parameter	Setup register
*buf0	DMA memory address register (source memory address)
*buf1	DMA memory address reload register (source memory address)
size	DMA transfer count reload register
zero_cnt	DMA transfer count register

(2) The start silence data (Trigger to DMA) shown in Figure 6

Makes the trigger which starts the repeat mode DMA transfer by writing silence data (0h) to the transmit buffer of CSIO.

(3) When this function is called from application program,

(i) Starts outputting the number of silence data specified by zero_cnt from the buffer *buf0 which stored silence data.

(ii) DMA interrupt request generates at the time of completing silence data output.

And the address of *buf1 and the value of size are reloaded to the DMA memory address register and DMA transfer count register, respectively. Furthermore starts outputting audio data which stored in *buf1.

Function	**_dai_stop**		
Classification	API		
Description	Stops outputting digital audio data streaming. Disables DMA transfer and CSIO transfer. Stops the SCLK and the LRCK.		
Format	void _dai_stop(void)		
Parameter	None		
Return	None		
Subroutines	None		
Supplement	DMA transfer count register and DMA transfer count reload register holds values at the time of DMA transfer stopping. Waits until transmit buffer register and transmit register of CSIO become empty, in this function.		

Function	**_cnv_endian**		
Classification	API		
Description	Converts endian of 2 bytes (WORD) data. This function is necessity when using 8 bits CSIO.		
Format	void _cnv_endian(unsigned char far *ram_buff, unsigned short Bsize) #pragma PARAMETER _cnv_endian(A0, R1)		
Parameter	*ram_buff	I/O	The address of the buffer which stored data to convert.
	Bsize	I	The size of conversion bytes (Sets the even number to Bsize)
Return	None		
Subroutines	None		
Note	Specify the even number bytes to the Bsize. Bsize can be specified from 2 to (FFFFh-1). Do not set 0 to the Bsize.		
Supplement	M32C/80 series microcomputer is little endian type. Digital audio data length treated in this example is 16 bits per 1 channel. When using 8 bits CSIO, endian conversion of data which stored in the buffer is necessity. We recommend size of the buffer which can be divided by 4. Because one unit of digital audio data treated in this example consists of 4 bytes (=32 bits, viz. Lch and Rch).		

Function	**Init_SCLK**		
Classification	Internal function		
Description	Sets the SCLK (TA2) registers. TA2 is stopping in this function.		
Format	void Init_SCLK(void)		
Parameter	None		
Return	None		
Subroutines	None		
Supplement	In this function, the interrupt priority level of TA2 is set to 0. (Disable interrupt of TA2)		

Function	**Start_LRCK**		
Classification	Internal function		
Description	Sets the LRCK (TA3) registers. After setting registers of TA3, TA3 is started in this function. The LRCK counts event (falling edge) of the SCLK, so the LRCK is stopping until the SCLK starts.		
Format	void Start_LRCK(void)		
Parameter	None		
Return	None		
Subroutines	None		
Supplement	In this function, the interrupt priority level of TA3 is set to 0. (Disable interrupt of TA3)		

Function	**Init_CSIO8**		
Classification	Internal function (When using 8 bits CSIO)		
Description	Sets the UART3 registers as 8 bits clock synchronous serial I/O mode. Uses the external clock SCLK as synchronous clock. Transmit data is output at falling edge of transfer clock.		
Format	void Init_CSIO8(void)		
Parameter	None		
Return	None		
Subroutines	None		
Supplement	As transmission start condition, when selecting external clock and selecting falling edge to the CLKi polarity, it is necessity that CLKi pin level is H.		

Function	**Start_DMA0**		
Classification	Internal function (When using 8 bits CSIO)		
Description	Sets DMA0 as DMAC, when using 8 bits CSIO. By this function calling, the DMA will be in the standby state which starts transmission by the interruption request that occurs when the 8 bits CSIO transmit buffer register becomes empty.		
Format	void Start_DMA0(unsigned short far *buf0, unsigned short far *buf1, unsigned short size, unsigned short zero_cnt)		
Parameter	*buf0	I	Top address of the buffer0 defined by user
	*buf1	I	Top address of the buffer1 defined by user
	size	I	Size of the buffer. (Specify in WORD unit)
	zero_cnt	I	Sets the number of outputting silence data.
Return	None		
Subroutines	None		
Supplement	In this function, the interrupt priority level of DMA0 is set to 7. All parameters of this function are passed from the _dai_start function which calls this function. DMA mode register 0 contains setting bits of both DMA0 and DMA1. This function sets DMA0 only. (Setting of DMA1 is held)		

Function	**Start_CSIO8**		
Classification	Internal function (When using 8 bits CSIO)		
Description	Enables CSIO transmission which is set by the Init_CSIO8 function.		
Format	void Start_CSIO8(void)		
Parameter	None		
Return	None		
Subroutines	None		
Supplement			

Function	**Init_CSIO16**		
Classification	Internal function (When using 16 bits CSIO)		
Description	Sets the intelligent I/O group 3 registers as 16 bits clock synchronous serial I/O mode. Uses the external clock SCLK as synchronous clock. Transmit data is output at falling edge of transfer clock.		
Format	void Init_CSIO16(void)		
Parameter	None		
Return	None		
Subroutines	None		
Supplement	When using the intelligent I/O group 3 as clock synchronous serial mode, transmission data is output at falling edge of transfer clock. (The clock polarity at data transmission is restricted to the falling edge.)		

Function	**Start_DMA1**		
Classification	Internal function (When using 16 bits CSIO)		
Description	Sets DMA1 as DMAC, when using 16 bits CSIO. By this function calling, the DMA will be in the standby state which starts transmission by the interruption request that occurs when the 16 bits CSIO transmit buffer register becomes empty.		
Format	void Start_DMA1(unsigned short far *buf0, unsigned short far *buf1, unsigned short size, unsigned short zero_cnt)		
Parameter	*buf0	I	Top address of the buffer0 defined by user
	*buf1	I	Top address of the buffer1 defined by user
	size	I	Size of the buffer. (Specify in WORD unit)
	zero_cnt	I	Sets the number of outputting silence data.
Return	None		
Subroutines	None		
Supplement	In this function, the interrupt priority level of DMA1 is set to 7. All parameters of this function are passed from the _dai_start function which calls this function. DMA mode register 0 contains setting bits of both DMA0 and DMA1. This function sets DMA1 only. (Setting of DMA0 is held)		

Function	**Start_CSIO16**		
Classification	Internal function (When using 16 bits CSIO)		
Description	Enables CSIO transmission which is set by the Init_CSIO16 function.		
Format	void Start_CSIO16(void)		
Parameter	None		
Return	None		
Subroutines	None		

6. Sample programs of Digital Audio I/F driver

This chapter shows sample program of digital audio I/F driver which are described in “5 Driver function details.” This program is sample, so changes and adjustment which tuned to the user application program are required. And this sample does not guarantee operation in any system/application program.

- List of files

File name	Contents
dai_drv.h	Header file for application program. This file needs to be included from user program, in order to use the driver API functions in user application program.
C_dai_drv.c	Sample program of the Digital Audio I/F driver.
A_endian.a30	Sample program which converts endian of 2 bytes (WORD) data.

- Compile option
  When using 16 bits CSIO, define “CSIO16.”
  If “CSIO16” is not defined, this example generates the driver using 8 bits CSIO in this sample code.
  The following examples show how to define “CSIO16.”

  (Example 1) By typing “#define CSIO16” in sample program, define the “CSIO16”.
  (Example 2) By using the compiling option -D of the NC308, define the “CSIO16” at the time of compiling.

dai_drv.h

```
/*""FILE COMMENT""*****************************************************************
 * System Name : Digital Audio I/F Driver sample
 * File Name   : dai_drv.h
 * Version     : 0.01
 * Contents    : Digital Audio I/F Driver sample include file
 * CPU         : M32C/80 series
 * Compiler    : NC308WA (V.3.10 Release 3)
 * OS          :
 * Note        : (1) The 16 bits CSIO can use only on the M32C/83 group.
 *               (2) When using 16 bits CSIO, define "CSIO16".
 *                   This sample default generates the driver using 8 bits CSIO.
 *
 ***********************************************************************************
 * Copyright(C)2003, Renesas Technology Corp.
 * Copyright(C)2003, Renesas Solutions Corp.
 * All rights reserved.
 ***********************************************************************************
 * History     : 2002.10.01 Ver 0.01
 *             :
 *""FILE COMMENT END""*************************************************************/


/*****************************************/
/*  Defines                              */
/*****************************************/
```

```
/*****************************************/
/*    Prototypes                         */
/*****************************************/
extern void _dai_start( unsigned short far *buf0,
                        unsigned short far *buf1,
                        unsigned short size,
                        unsigned short zero_cnt );

extern void _dai_stop( void );
extern void _dai_pause( void );


#if !defined (CSIO16)
extern void _cnv_endian( unsigned char far *ram_buff, unsigned short Bsize );
#pragma PARAMETER _cnv_endian( A0, R1 )
#endif


/*****************************************/
/*    Global variables                   */
/*****************************************/
```

C_dai_drv.c

```
/*""FILE COMMENT""*******************************************************************
  * System Name : Digital Audio I/F Driver sample
  * File Name      : C_dai_drv.c
  * Version        : 0.02
  * Contents       : Digital Audio I/F Driver sample programs
  * CPU              : M32C/80 series
  * Compiler       : NC308WA (V.3.10 Release 3)
  * OS               :
  * Note             : (1) The 16 bits CSIO can use only on the M32C/83 group.
  *                        (2) When using 16 bits CSIO, define "CSIO16".
  *                              This sample default generates the driver using 8 bits CSIO.
  *
  ************************************************************************************
  * Copyright(C)2003, Renesas Technology Corp.
  * Copyright(C)2003, Renesas Solutions Corp.
  * All rights reserved.
  ************************************************************************************
  * History         : 2002.10.01 Ver 0.01
  *                     : 2002.10.25 Ver 0.02
  *                     :
  *""FILE COMMENT END""*************************************************************/


/*****************************************/
/* Include headers                       */
/*****************************************/
#include "sfr32c83.h"      /* File contains SFR (Special Function Register) definitions for M32C/83 */
```

```
/*****************************************/
/* Defines                               */
/*****************************************/
#define Start_SCLK          ta2s = 1
#define Stop_SCLK           ta2s = 0
#define Stop_LRCK           ta3s = 0


/*****************************************/
/* Global variables                      */
/*****************************************/


/*****************************************/
/* Prototypes                            */
/*****************************************/
void _dai_start( unsigned short far *buf0,
                 unsigned short far *buf1,
                 unsigned short size,
                 unsigned short zero_cnt );
void _dai_stop( void );


void Init_SCLK( void );
void Start_LRCK( void );

#if defined (CSIO16)
/******* 16bits CSIO ***********************************************************/
    void Init_CSIO16( void );
    void Start_CSIO16(void);
    void Start_DMA1( unsigned short far *buf0,
                     unsigned short far *buf1,
                     unsigned short size,
                     unsigned short init_cnt );

#else
/******* 8bits CSIO ************************************************************/
    void Init_CSIO8( void );
    void Start_CSIO8( void );
    void Start_DMA0( unsigned short far *buf0,
                     unsigned short far *buf1,
                     unsigned short size,
                     unsigned short zero_cnt );
#endif
/*******************************************************************************/



/*""FUNC COMMENT""*****************************************************************
 * ID                :
 * Abstract          : Start of digital audio data play
 *-------------------------------------------------------------------------------
 * Include           : "sfr32c83.h"
 *-------------------------------------------------------------------------------
 * Declaration       : void _dai_start( unsigned short far *buf0,
 *                   :                  unsigned short far *buf1,
 *                   :                  unsigned short size,
 *                   :                  unsigned short zero_cnt )
 *-------------------------------------------------------------------------------
 * Function          : Start outputting digital audio data streaming to the DAC.
 *                   : This function needs to be called after storing following data in buffers.
 *                   :       buf0 <-- silence data (0 clearance)
 *                   :       buf1 <-- digital audio data
 *-------------------------------------------------------------------------------
```

```
 * Argument         : unsigned short far *buf0    ; Top address of the buffer 0 defined by user
 *                  : unsigned short far *buf0    ; Top address of the buffer 1 defined by user
 *                  : unsigned short size         ; Size of buffer(Specify in WORD unit)
 *                  : unsigned short zero_cnt     ; Sets the number of outputting silence data
 *-------------------------------------------------------------------------
 * Return value : None
 *-------------------------------------------------------------------------
 * Inputs           : None
 * Outputs          : None
 *-------------------------------------------------------------------------
 * Subroutines    : Init_SCLK
 *                  : Init_CSIO8 / Init_CSIO16
 *                  : Start_LRCK
 *                  : Start_DMA0 / Start_DMA1
 *                  : Start_CSIO8 / Start_CSIO16
 *-------------------------------------------------------------------------
 * Note             : Case of 8bits CSIO
 *                  :       To the zero_cnt, set the value 1+4n(n=1,2,...)
 *                  : Case of 16bits CSIO
 *                  :       To the zero_cnt, set the value 2n(n=1,2,...)
 *                  :
 *                  : Before calling this function, initialize the DAC.
 *-------------------------------------------------------------------------
 * History          :
 *""FUNC COMMENT END""*****************************************************/
#define SHORT_TIMER 5                   /* <-- Changes and adjustment which tuned to the user system are necessity */
#define LONG_TIMER    10000             /* <-- Changes and adjustment which tuned to the user system are necessity */
void _dai_start( unsigned short far *buf0,    /* Top address of the buffer 0 defined by user */
                 unsigned short far *buf1,    /* Top address of the buffer 0 defined by user */
                 unsigned short size,         /* Size of the buffer (Specify in WORD unit) */
                 unsigned short zero_cnt )    /* Sets the number of outputting silence data */
{
    /* Set the SCLK(TA2) registers */
    Init_SCLK();


#if defined (CSIO16)
/******* 16bits CSIO *********************************************************/
    /* Set the CSIO (Intelligent I/O group 3) registers */
    Init_CSIO16();
#else
/******* 8bits CSIO **********************************************************/
    /* Set the CSIO (UART3) registers */
    Init_CSIO8();
#endif
/*****************************************************************************/


    /* Set the LRCK (TA3) registers, The LRCK is stopping until the SCLK starts */
    Start_LRCK();


#if defined (CSIO16)
/******* 16bits CSIO *********************************************************/
    /* Set the DMA1 registers */
    Start_DMA1( buf0, buf1, size, zero_cnt );

    /* Enable transmission of CSIO */
    Start_CSIO16();

    /* By writing the silence data, start repeating DMA transfer */
    g3tb = 0x0000;

    /* Start the SCLK */
    Start_SCLK;
```

```
#else
/******* 8bits CSIO ********************************************************/
        /* Set the DMA0 registers */
        Start_DMA0( buf0, buf1, size, zero_cnt );

        /* Enable transmission of CSIO */
        Start_CSIO8();

        /* Starts processing of the SCLK */
        ta2 = (SHORT_TIMER -1 );        /* Set the initial divide ratio SHORT_TIMER */
        Start_SCLK;                      /* Start the SCLK with SHORT_TIMER and output pulse is lifted to H */
        ta2 = (LONG_TIMER -1 );         /* Set the initial divide ratio LONG_TIMER to the reload-register and reload at next time */
        while( ir_ta2ic != 1 );         /* Wait finishes counting the SHORT_TIMER */
                                          /* Start counting LONG_TIMER in H level */
        ta2 = (2-1);                    /* In this meantime, set divide ratio 2 to SCLK (TA2) and set transmit-buffer-register */

        /* By writing the silence data, start repeating DMA transfer */
        u3tbl = 0x00;
        ir_ta2ic = 0;
#endif
/***************************************************************************/

}



/*""FUNC COMMENT""*************************************************************
  * ID                :
  * Abstract          : Initialize the SCLK (TA2)
  *-----------------------------------------------------------------------
  * Include           : "sfr32c83.h"
  *-----------------------------------------------------------------------
  * Declaration       : void Init_SCLK( void )
  *-----------------------------------------------------------------------
  * Function          : Sets the SCLK (TA2) registers.
  *                         SCLK (TA2) is stopping in this function.
  *
  *                           Mode; Event counter mode
  *                           Count source; Counts the rising edge of external signal which
  *                                         inputted into TA2IN pin (MCLK --> TA2IN pin)
  *                           Divide ratio; dividing external clock (MCLK) by 4
  *                           TAiIN pin function; Set TA2IN pin as input port
  *                           TAiOUT pin function; Pulse output
  *-----------------------------------------------------------------------
  * Argument          : None
  *-----------------------------------------------------------------------
  * Return value : None
  *-----------------------------------------------------------------------
  * Inputs            : None
  * Outputs           : None
  *-----------------------------------------------------------------------
  * Subroutines       : None
  *-----------------------------------------------------------------------
  * Note              :
  *-----------------------------------------------------------------------
  * History           :
  *""FUNC COMMENT END""********************************************************/
```

```
void Init_SCLK( void )
{
      /* Stop counting */
      ta2s = 0;

      /* Disable interruption */
      ta2ic = 0;

      /* Set mode */
      ta2mr = (0x09); /* Timer Ai mode register */
            /* b[1:0]; Operation mode                          [01;Event counter mode] */
            /* b[2];     invalid                               [X; invalid in M32C/80 series] */
            /* b[3];     Count polarity select                 [1; Counts external signal's rising edges] */
            /* b[4];     Up/down switching case select    [0; Up/down flag's content] */
            /* b[5];     Set to 0 in Event counter mode [0; (Set to 0)] */
            /* b[6];     Count operation type select       [0; Reload type] */
            /* b[7];     Two-phase pulse signal processing operation select [0; fixed on TA2] */

      /* Trigger select */
      ta2tgl = 0;         /* Input on TA2IN is selected */
      ta2tgh = 0;

      /* TA2IN pin function */
      p7_5 = 0;            /* Set TA2IN pin as input port */
      pd7_5 = 0;


      /* TA2OUT pin function */
      psc_4 = 0;          /* function select register C     : Timer output (TA2OUT) */
      psl1_4 = 0;         /* function select register B1    : Function that was selected in PSC_4 */
      ps1_4 = 1;          /* function select register A1    : Function that was selected in PSL1_4 */


#if defined (CSIO16)
      ta2 = (2-1);       /* Generate the SCLK by dividing MCLK by 4 and outputting pulse to TA2OUT pin */
#endif

}



/*""FUNC COMMENT""***************************************************************
  * ID                :
  * Abstract        : Start LRCK(TA3)
  *-------------------------------------------------------------------------
  * Include         : "sfr32c83.h"
  *-------------------------------------------------------------------------
  * Declaration   : void Start_LRCK( void )
  *-------------------------------------------------------------------------
  * Function        : Sets the LRCK (TA3) registers.
  *                       After setting registers of TA3, TA3 is started in this function.
  *                       The LRCK counts event (falling edge) of the SCLK, so the LRCK is
  *                       stopping until the SCLK starts.
  *
  *                          Mode; Event counter mode
  *                          Count source; Counts the falling edge of the external signal which
  *                                              inputted into TA3IN pin.
  *                          Divide ratio; dividing external clock(SCLK) by 32
  *                          TAiIN pin function; Set TA3IN pin as input port
  *                          TAiOUT pin function; Pulse output
  *-------------------------------------------------------------------------
  * Argument        : None
  *-------------------------------------------------------------------------
  * Return value : None
  *-------------------------------------------------------------------------
```

```
 * Inputs          : None
 * Outputs         : None
 *-----------------------------------------------------------------------
 * Subroutines    : None
 *-----------------------------------------------------------------------
 * Note            :
 *-----------------------------------------------------------------------
 * History         :
 *""FUNC COMMENT END""*****************************************************/
void Start_LRCK( void )
{
    /* Stop counting */
    ta3s = 0;

    /* Disable interruption */
    ta3ic = 0;

    /* Set mode */
    ta3mr = (0x01); /* Timer Ai mode register */
        /* b[1:0]; Operation mode                    [01;Event counter mode] */
        /* b[2];   invalid                           [X; invalid in M32C/80 series] */
        /* b[3];   Count polarity select             [0; Counts external signal's falling edges] */
        /* b[4];   Up/down switching case select     [0; Up/down flag's content] */
        /* b[5];   Set to 0 in Event counter mode [0; (Set to 0)] */
        /* b[6];   Count operation type select       [0; Reload type] */
        /* b[7];   Two-phase pulse signal processing operation select      [0; not using two-phase pulse signal] */

    /* Trigger select */
    ta3tgl = 0;       /* Input on TA3IN is selected */
    ta3tgh = 0;

    /* TA3IN pin function */
    p7_7 = 0;         /* Set TA3IN pin as input port */
    pd7_7 = 0;

    /* TA3OUT pin function */
    psl1_6 = 1;       /* function select register B1    : Timer output (TA3OUT) */
    ps1_6 = 1;        /* function select register A1    : Function that was selected in PSL1_6 */

    /* Set the initial divide ratio */
    ta3 = 17-1;

    /* Start timer */
    ta3s = 1;
    ta3 = 16-1;           /* divide ratio = 32 (Set the re-load register) */
}



#if defined (CSIO16)
/*""FUNC COMMENT""*************************************************************
 * ID              :
 * Abstract        : Initialize the Intelligent I/O group 3 as 16 bits CSIO
 *-----------------------------------------------------------------------
 * Include         : "sfr32c83.h"
 *-----------------------------------------------------------------------
 * Declaration     : void Init_CSIO16( void )
 *-----------------------------------------------------------------------
 * Function        : Set the Intelligent I/O group 3 registers as 16 bits CSIO.
 *                   : Uses the external clock SCLK as synchronous clock.
 *                   : Transmit data is output at falling edge of transfer clock.
 *-----------------------------------------------------------------------
 * Argument        : None
 *-----------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------
```

```
 * Inputs             : None
 * Outputs            : None
 *------------------------------------------------------------------------
 * Subroutines    : None
 *------------------------------------------------------------------------
 * Note                 :
 *------------------------------------------------------------------------
 * History            :
 *""FUNC COMMENT END""***************************************************/
void Init_CSIO16( void )
{

    /* Set Base timer */
    g3bcr0 = 0x7F;    /* Group 3 base timer control register 0 */
        /* b[1:0]; Count source select                          [11;       f1] */
        /* b[6:2]; Count source division ratio select    [11111; No division] */
        /* b[7];      Base timer interrupt select               [0;         Bit 15 overflow] */

    g3bcr1 = 0x00;    /* Group 3 base timer control register 1 */
        /* b[0]; Base timer reset cause select               [0; Synchronize the base timer 2 reset without resetting the timer] */
        /* b[1]; Base timer reset cause select               [0; Does not reset the base timer when it matches WG register ch0] */
        /* b[2]; Reserved bit                                      [0; (Set to 0)] */
        /* b[3]; -                                                   [0; (Set to 0)] */
        /* b[4]; Base timer start                                 [0; Base timer reset] */
        /* b[6:5]; -                                                 [00;(Set to 0)] */
        /* b[7]; Parallel real-time port function select [0; Not use] */

    /* Group 3 waveform generation control register 0,1 */
    g3pocr0 = 0x07; /* Group 3 WG control register 0 */
        /* b[2:0]; Operation mode select                      [111; Assigns communication output to a port] */
        /* b[3];      Parallel RTP output trigger select    [0;      Match of WG register j isn't trigger] */
        /* b[4];      Output initial value select               [0;      Outputs 0 as the initial value] */
        /* b[5];      Reload timing select                          [0;      Reload a new count when CPU writes the count] */
        /* b[6];      RTP function select                           [0;      Not use] */
        /* b[7];      Inverted output function select         [0;      Output is not inverted] */

    g3pocr1 = 0x07; /* Group 3 WG control register 1 */
        /* b[2:0]; Operation mode select                      [111; Assigns communication output to a port] */
        /* b[3];      Parallel RTP output trigger select    [0;      Match of WG register j isn't trigger] */
        /* b[4];      Output initial value select               [0;      Outputs 0 as the initial value] */
        /* b[5];      Reload timing select                          [0;      Reload a new count when CPU writes the count] */
        /* b[6];      RTP function select                           [0;      Not use] */
        /* b[7];      Inverted output function select         [0;      Output is not inverted] */


    /* Group3 channel i operation */
    g3fe = 0x03;       /* Group 3 function enable register */
        /* b[7:0]; Ch i function enable bit                  [0000 0011; Enables function on ch0 and ch1] */


    /* Communication mode */
    g3mr = 0x45;       /* Group 3 SI/O communication mode register */
        /* b[1:0]; Communication mode select                  [01; Serial I/O mode] */
        /* b[2];      Internal/external clock select          [1;    External clock] */
        /* b[3];      Transmission data length select         [0;    16 bits] */
        /* b[5:4]; -                                                 [00; (Set to 0)] */
        /* b[6];      Transfer direction select                [1;    MSB first] */
        /* b[7];      Transmit interrupt cause select         [0;    Transmit buffer is empty] */
```

```
    /* Communication control */
    g3cr = 0x00;       /* Group 3 SI/O communication control register */
          /* b[0]; Transmit enable bit                    [0; Transmission disabled] */
          /* b[1]; Transmit register empty flag           [0; Data present in transmit register (R-Only)] */
          /* b[2]; Transmit buffer empty flag             [0; Data present in transmit buffer register (R-Only)] */
          /* b[3]; -                                          [0; (Set to 0)] */
          /* b[4]; Receive enable bit                     [0; Reception disabled] */
          /* b[5]; Receive complete flag                  [0; No data present in receive buffer register (R-Only)] */
          /* b[6]; TxD output polarity reverse select     [0; No reverse] */
          /* b[7]; RxD input polarity reverse select      [0; No reverse] */


    /* Set the input pin which inputs external clock (ISCLK3; P12_1) as the input port */
    ps6_1 = 0;                  /* P12_1 ; Input port */

    /* Set the TxD pin (ICTxD3; P12_0) as the II/O group 3 output */
    ps6_0 = 1;                  /* P12_0 ; II/O group 3 output (OUTC30) */


    /* Set the interrupt request bit of the II/O SIO3t */
    iio10ie = 0x10;             /* Enable the interrupt request from communication function of II/O group 3 */
}


/*""FUNC COMMENT""************************************************************
 * ID                :
 * Abstract          : Start the Intelligent I/O group 3 as 16bits CSIO
 *-----------------------------------------------------------------------------
 * Include           : "sfr32c83.h"
 *-----------------------------------------------------------------------------
 * Declaration       : void Start_CSIO16( void )
 *-----------------------------------------------------------------------------
 * Function          : Enables CSIO transmission which is set by the Init_CSIO16 function
 *-----------------------------------------------------------------------------
 * Argument          : None
 *-----------------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------------
 * Inputs            : None
 * Outputs           : None
 *-----------------------------------------------------------------------------
 * Subroutines       : None
 *-----------------------------------------------------------------------------
 * Note              :
 *-----------------------------------------------------------------------------
 * History           :
 *""FUNC COMMENT END""*******************************************************/
void Start_CSIO16( void )
{
    /* Communication control */
    g3cr = 0x01;       /* Group 3 SI/O communication control register */
          /* b[0]; Transmit enable bit                    [1; Transmission enabled] */
          /* b[1]; Transmit register empty flag           [0; Data present in transmit register (R-Only)] */
          /* b[2]; Transmit buffer empty flag             [0; Data present in transmit buffer register (R-Only)] */
          /* b[3]; -                                          [0; (Set to 0)] */
          /* b[4]; Receive enable bit                     [0; Reception disabled] */
          /* b[5]; Receive complete flag                  [0; No data present in receive buffer register (R-Only)] */
          /* b[6]; TxD output polarity reverse select     [0; No reverse] */
          /* b[7]; RxD input polarity reverse select      [0; No reverse] */

}
```

```
#else
/*""FUNC COMMENT""*****************************************************************
 * ID                :
 * Abstract          : Initialize the UART3 as 8bits CSIO
 *-----------------------------------------------------------------------
 * Include           : "sfr32c83.h"
 *-----------------------------------------------------------------------
 * Declaration       : void Init_CSIO8( void )
 *-----------------------------------------------------------------------
 * Function          : Set the UART3 registers as 8 bits clock synchronous serial I/O mode.
 *                   : Uses the external clock SCLK as synchronous clock.
 *                   : Transmit data is output at falling edge of transfer clock
 *-----------------------------------------------------------------------
 * Argument          : None
 *-----------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------
 * Inputs            : None
 * Outputs           : None
 *-----------------------------------------------------------------------
 * Subroutines       : None
 *-----------------------------------------------------------------------
 * Note              :
 *-----------------------------------------------------------------------
 * History           :
 *""FUNC COMMENT END""*************************************************************/
void Init_CSIO8( void )
{
    /* Disable transmission */
    u3c1 = 0x02;      /* UARTi transmit/receive control register 1 */
        /* b[0]; Transmit enable bit                          [0; Transmission disabled] */
        /* b[1]; Transmit buffer empty flag                   [1; No data present in transmit buffer register (R only)] */
        /* b[2]; Receive enable bit                           [0; Reception disabled] */
        /* b[3]; Receive complete flag                        [0; Data present in receive buffer register (R only)] */
        /* b[4]; UARTi transmit interrupt cause select        [0; Transmit buffer empty] */
        /* b[5]; UARTi continuous receive mode enable bit [0; Continuous receive mode disabled] */
        /* b[6]; Data logic select bit                        [0; No reverse] */
        /* b[7]; Clock divide synchronizing stop bit          [0; Synchronizing stop] */

    /* Set transmit/receive mode */
    u3mr = 0x09;      /* UARTi transmit/receive mode register */
        /* b[2:0]; Serial I/O mode select                     [001; Serial I/O mode (Clock synchronous)] */
        /* b[3];     Internal/external clock select           [1;     External clock] */
        /* b[4];     Stop bit length select                   [X;     Invalid] */
        /* b[5];     Odd/even parity select                   [X;     Invalid] */
        /* b[6];     Parity enable bit                        [X;     Invalid] */
        /* b[7];     TxD, RxD input/output polarity switch bit [0; No reversed] */

    u3c0 = 0x98;      /* UARTi transmit/receive control register 0 */
        /* b[1:0]; BRG count source select                    [00; f1 is selected] */
        /* b[2];     CTS/RTS function select                  [0;   See bit 4, valid when bit 4 = 0] */
        /* b[3];     Transmit register empty flag             [1;   No data present in transmit register (R only)] */
        /* b[4];     CTS/RTS disable bit                      [1;   CTS/RTS function disabled] */
        /* b[5];     Data output select bit                   [0;   TxDi pin is CMOS output] */
        /* b[6];     CLK polarity select bit                  [0;   Transmit data is output at falling edge of transfer clock] */
        /* b[7];     Transfer format select bit               [1;   MSB first] */


    /* Set bit rate */
    u3brg = 0x00;     /* UARTi bit rate generator */
        /* b[7:0]; Assuming that set value = n, BRGi divides the count source by n+1 [0; f1] */


    /* Set the CLK3 pin (P9_0) as the I/O port */
    prc2 = 1;           /* Release of the protection (Enable writing) */
    ps3_0 = 0;          /* Set port P9_0 as I/O port */
                        /* (Input direction when reset) */
```

```
    /* Set the TxD3 pin (P9_2) as the UART3 output */
    psl3_2 = 0;         /* UART3 output */
    prc2 = 1;           /* Release of the protection (Enable writing) */
    ps3_2 = 1;          /* Function that was selected in PSL3_2 */
}


/*""FUNC COMMENT""*****************************************************************
 * ID                :
 * Abstract          : Start the UART3 as 8bits CSIO
 *-------------------------------------------------------------------------------
 * Include           : "sfr32c83.h"
 *-------------------------------------------------------------------------------
 * Declaration    : void Start_CSIO8( void )
 *-------------------------------------------------------------------------------
 * Function          : Enables CSIO transmission which is set by the Init_CSIO8 function
 *-------------------------------------------------------------------------------
 * Argument          : None
 *-------------------------------------------------------------------------------
 * Return value : None
 *-------------------------------------------------------------------------------
 * Inputs            : None
 * Outputs           : None
 *-------------------------------------------------------------------------------
 * Subroutines    : None
 *-------------------------------------------------------------------------------
 * Note              :
 *-------------------------------------------------------------------------------
 * History           :
 *""FUNC COMMENT END""*************************************************************/
void Start_CSIO8( void )
{
    /* Enable transmission */
    u3c1 = 0x03;        /* UARTi transmit/receive control register 1 */
        /* b[0]; Transmit enable bit                       [1; Transmission enabled] */
        /* b[1]; Transmit buffer empty flag                [1; No data present in transmit buffer register (R only)] */
        /* b[2]; Receive enable bit                        [0; Reception disabled] */
        /* b[3]; Receive complete flag                     [0; Data present in receive buffer register (R only)] */
        /* b[4]; UARTi transmit interrupt cause select     [0; Transmit buffer empty] */
        /* b[5]; UARTi continuous receive mode enable bit [0; Continuous receive mode disabled] */
        /* b[6]; Data logic select bit                     [0; No reverse] */
        /* b[7]; Clock divide synchronizing stop bit       [0; Synchronizing stop] */
}
#endif


#if defined (CSIO16)
/*""FUNC COMMENT""*****************************************************************
 * ID                :
 * Abstract          : Set DMA1 and enable DMA1
 *-------------------------------------------------------------------------------
 * Include           : "sfr32c83.h"
 *-------------------------------------------------------------------------------
 * Declaration    : void Start_DMA1( unsigned short far *buf0,
 *                   :                unsigned short far *buf1,
 *                   :                unsigned short size,
 *                   :                unsigned short zero_cnt )
 *-------------------------------------------------------------------------------
 * Function          : Set DMA1 as DMAC when using 16 bits CSIO.
 *                   : By calling this function, the DMA will be in a standby state
 *                     which starts transmission by the interruption request that occurs
 *                     when the 16 bits CSIO transmit buffer register becomes empty.
 *-------------------------------------------------------------------------------
```

```
 * Argument         : unsigned short far *buf0    ; Top address of the buffer 0 defined by user
 *                  : unsigned short far *buf0    ; Top address of the buffer 1 defined by user
 *                  : unsigned short size         ; Size of buffer (Specify in WORD unit)
 *                  : unsigned short zero_cnt     ; Set the number of outputting silence data
 *-------------------------------------------------------------------------
 * Return value : None
 *-------------------------------------------------------------------------
 * Inputs           : None
 * Outputs          : None
 *-------------------------------------------------------------------------
 * Subroutines    : None
 *-------------------------------------------------------------------------
 * Note             :
 *-------------------------------------------------------------------------
 * History          :
 *""FUNC COMMENT END""***************************************************/
void Start_DMA1( unsigned short far *buf0,    /* Top address of the buffer 0 defined by user */
                 unsigned short far *buf1,    /* Top address of the buffer 1 defined by user */
                 unsigned short size,         /* Size of buffer (Specify in WORD unit) */
                 unsigned short zero_cnt )    /* Set the number of outputting silence data */
{
    unsigned short    dma_cnt;
    unsigned char tmp;

    dma_cnt     = size;     /* DMA transfer count (Transfer count per 16 bits) */

    /* DMA inhibit */
    asm("     stc       DMD0,     $$[FB]", tmp);
    asm("     and.b     #00Fh,    $$[FB]", tmp);   /* Holds setting of the DMA0 */
    asm("     ldc       $$[FB], DMD0"    , tmp);

    /* Select the DMA request cause at the DMA inhibit state */
    dm1sl = 0x9C;     /* DMAi request cause select register */
        /* b[4:0]; DMA request cause select        [11100; Intelligent I/O interrupt control register 10] */
        /* b[5];     Software DMA request bit      [0;       ignore] */
        /* b[6];     -                             [0;       (set to 0)] */
        /* b[7];     DMA request bit (DRQ)         [1;       (set to 1, See Note1)] */
        /* Note1:    Set DMA inhibit before changing the DMA request cause, Set DRQ bit to 1 simultaneously */

    /* Set the DMA transfer count */
    asm("     ldc $$[FB], DCT1", zero_cnt);     /* DMAi transfer count register */
    asm("     ldc $$[FB], DRC1", dma_cnt);      /* DMAi transfer count reload register */

    /* Set the DMA source address (When the transfer direction select bits is 1, this register is source address) */
    asm("     ldc $$[FB], DMA1", buf0);         /* DMAi memory address register */
    asm("     ldc $$[FB], DRA1", buf1);         /* DMAi memory address reload register */

    /* Set the DMA destination address */
    asm("     ldc #017Ch, DSA1");               /* DMAi SFR address register [Group 3 SI/O transmit buffer register] */

    /*
        Insert dummy cycles if needed
        (Adjust cycles depending on the number of DMA using in the application)
    */

    /* Set DMA mode and enable DMA transfer */
    asm("     or.b      #0F0h,    $$[FB]", tmp);
    asm("     ldc       $$[FB], DMD0"    , tmp);   /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select          [XX; Hold setting of DMA0] */
        /* b[2];     Channel 0 transfer unit select        [X;    Hold setting of DMA0] */
        /* b[3];     Channel 0 transfer direction select   [X;    Hold setting of DMA0] */
        /* b[4:5]; Channel 1 transfer mode select          [11; Repeat transfer] */
        /* b[6];     Channel 1 transfer unit select        [1;    16 bits] */
        /* b[7];     Channel 1 transfer direction select   [1;    Memory to fixed address] */

    /* Interrupt priority level */
    dm1ic = 0x07;
}
```

```
#else


/*""FUNC COMMENT""*****************************************************************
 * ID                :
 * Abstract          : Setting DMA0 and enable DMA0
 *-----------------------------------------------------------------------
 * Include           : "sfr32c83.h"
 *-----------------------------------------------------------------------
 * Declaration       : void Start_DMA0( unsigned short far *buf0,
 *                   :                  unsigned short far *buf1,
 *                   :                  unsigned short size,
 *                   :                  unsigned short zero_cnt )
 *-----------------------------------------------------------------------
 * Function          : Set DMA0 as DMAC when using 8 bits CSIO.
 *                   : By calling this function, the DMA will be in a standby state
 *                     which starts transmission by the interruption request that occurs
 *                     when the 8 bits CSIO transmit buffer register becomes empty.
 *-----------------------------------------------------------------------
 * Argument          : unsigned short far *buf0    ; Top address of the buffer 0 defined by user
 *                   : unsigned short far *buf0    ; Top address of the buffer 1 defined by user
 *                   : unsigned short size         ; Size of buffer (Specify in WORD unit)
 *                   : unsigned short zero_cnt     ; Set the number of outputting silence data
 *-----------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------
 * Inputs            : None
 * Outputs           : None
 *-----------------------------------------------------------------------
 * Subroutines       : None
 *-----------------------------------------------------------------------
 * Note              :
 *-----------------------------------------------------------------------
 * History           :
 *""FUNC COMMENT END""*************************************************************/
void Start_DMA0( unsigned short far *buf0,    /* Top address of the buffer 0 defined by user */
                 unsigned short far *buf1,    /* Top address of the buffer 1 defined by user */
                 unsigned short size,         /* Size of buffer (Specify in WORD unit) */
                 unsigned short zero_cnt )    /* Set the number of outputting silence data */
{
    unsigned short    dma_cnt;
    unsigned char tmp;

    dma_cnt     = size * 2;     /* DMA transfer count (Transfer count per 16 bits) */

    /* DMA inhibit */
    asm("     stc       DMD0,     $$[FB]", tmp);
    asm("     and.b     #0F0h,    $$[FB]", tmp);   /* Holds setting of the DMA1 */
    asm("     ldc       $$[FB], DMD0"    , tmp);

    /* Select the DMA request cause at the DMA inhibit state */
    dm0sl = 0x94;     /* DMAi request cause select register */
        /* b[4:0]; DMA request cause select        [10100; UART3 transmit] */
        /* b[5];   Software DMA request bit        [0;         ignore] */
        /* b[6];   -                               [0;         (set to 0)] */
        /* b[7];   DMA request bit (DRQ)           [1;         (set to 1, See Note1] */
        /* Note1:   Set DMA inhibit before changing the DMA request cause, Set DRQ bit to 1 simultaneously */

    /* Set the DMA transfer count */
    asm("     ldc $$[FB], DCT0", zero_cnt);     /* DMAi transfer count register */
    asm("     ldc $$[FB], DRC0", dma_cnt);      /* DMAi transfer count reload register */

    /* Set the DMA source address (When the transfer direction select bits is 1, this register is source address) */
    asm("     ldc $$[FB], DMA0", buf0);         /* DMAi memory address register */
    asm("     ldc $$[FB], DRA0", buf1);         /* DMAi memory address reload register */
```

```
    /* Set the DMA destination address */
    asm("    ldc #032Ah, DSA0");                    /* DMAi SFR address register [UART3 transmit buffer register] */

    /*
        Insert dummy cycles if needed
        (Adjust cycles depending on the number of DMA using in the application)
    */

    /* Set DMA mode and enable DMA transfer */
    asm("    or.b      #00Bh,   $$[FB]", tmp);
    asm("    ldc       $$[FB], DMD0"   , tmp);   /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select          [11; Repeat transfer] */
        /* b[2];   Channel 0 transfer unit select          [0;   8 bits] */
        /* b[3];   Channel 0 transfer direction select     [1;   Memory to fixed address] */
        /* b[4:5]; Channel 1 transfer mode select          [XX; Hold setting of DMA1] */
        /* b[6];   Channel 1 transfer unit select          [X;   Hold setting of DMA1] */
        /* b[7];   Channel 1 transfer direction select     [X;   Hold setting of DMA1] */

    /* Interrupt priority level */
    dm0ic = 0x07;
}
#endif



/*""FUNC COMMENT""*****************************************************************
 * ID                :
 * Abstract          : Stop outputting digital audio data play
 *-------------------------------------------------------------------------------
 * Include           : "sfr32c83.h"
 *-------------------------------------------------------------------------------
 * Declaration       : void _dai_stop( void )
 *-------------------------------------------------------------------------------
 * Function          : Stop outputting digital audio data streaming.
 *                   : Stop the SCLK and the LRCK
 *-------------------------------------------------------------------------------
 * Argument          : None
 *-------------------------------------------------------------------------------
 * Return value : None
 *-------------------------------------------------------------------------------
 * Inputs            : None
 * Outputs           : None
 *-------------------------------------------------------------------------------
 * Subroutines       : None
 *-------------------------------------------------------------------------------
 * Note              :
 *-------------------------------------------------------------------------------
 * History           :
 *""FUNC COMMENT END""*************************************************************/
void _dai_stop( void )
{
    unsigned char tmp;



#if defined (CSIO16)
/******* 16bits CSIO *********************************************************/
#define TIME_OUT (5000) /* approx 1 msec : M32C/83 @20MHz */
    unsigned short time_out;
```

```
    /* DMA1 inhibit */
    dm1ic = 0;                                          /* DMA1 interrupt disable */
    asm("    stc      DMD0,    $$[FB]", tmp);    /* Store DMA mode register to tmp */
    asm("    and.b    #00Fh,   $$[FB]", tmp);    /* Holds setting of the DMA0 */
    asm("    or.b     #0C0h,   $$[FB]", tmp);
    asm("    ldc      $$[FB], DMD0"    , tmp);    /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select          [XX; Hold setting of DMA0] */
        /* b[2];   Channel 0 transfer unit select          [X;   Hold setting of DMA0] */
        /* b[3];   Channel 0 transfer direction select     [X;   Hold setting of DMA0] */
        /* b[4:5]; Channel 1 transfer mode select          [00; DMA1 inhibit] */
        /* b[6];   Channel 1 transfer unit select          [1;   16 bits] */
        /* b[7];   Channel 1 transfer direction select     [1;   Memory to fixed address] */

    /* Loops until the transmit buffer register of CSIO becomes empty */
    time_out = TIME_OUT;
    while( ti_g3cr == 0 ){
        if( time_out -- == 0 ) break;
    }
    /* Loops until the transmit register of CSIO becomes empty */
    time_out = TIME_OUT;
    while( txept_g3cr == 0 ){
        if( time_out -- == 0 ) break;
    }

    /* Disable CSIO transmission */

    /* Group 3 SI/O communication control register */
    te_g3cr = 0;        /* Transmission disabled */

    /* Group 3 SI/O communication mode register */
    g3mr = 0;           /* Communication part is reset */

    /* II/O SIO3 interrupt request bit */
    sio3te = 0;         /* Disable the interrupt request from communication function of II/O group 3 */


#else
/******* 8bits CSIO *********************************************************/
    /* DMA0 inhibit */
    dm0ic = 0;                                          /* DMA0 interrupt disable */
    asm("    stc      DMD0,    $$[FB]", tmp);    /* Store DMA mode register to tmp */
    asm("    and.b    #0F0h,   $$[FB]", tmp);    /* Holds setting of the DMA1 */
    asm("    or.b     #008h,   $$[FB]", tmp);
    asm("    ldc      $$[FB], DMD0"    , tmp);    /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select          [00; DMA0 inhibit] */
        /* b[2];   Channel 0 transfer unit select          [0;   8 bits] */
        /* b[3];   Channel 0 transfer direction select     [1;   Memory to fixed address] */
        /* b[4:5]; Channel 1 transfer mode select          [XX; Hold setting of DMA1] */
        /* b[6];   Channel 1 transfer unit select          [X;   Hold setting of DMA1] */
        /* b[7];   Channel 1 transfer direction select     [X;   Hold setting of DMA1] */


    /* Loops until the transmit buffer register of CSIO becomes empty */
    while( ti_u3c1 == 0 );
    /* Loops until the transmit register of CSIO becomes empty */
    while( txept_u3c0 == 0 );


    /* Disable CSIO transmission */
    u3c1    = 0x02;     /* UARTi transmit/receive control register 1 */
        /* b[0]; Transmit enable bit                       [0; Transmission disabled] */
        /* b[1]; Transmit buffer empty flag                [1; No data present in transmit buffer register (R only)] */
        /* b[2]; Receive enable bit                        [0; Reception disabled] */
        /* b[3]; Receive complete flag                     [0; Data present in receive buffer register (R only)] */
        /* b[4]; UARTi transmit interrupt cause select     [0; Transmit buffer empty] */
        /* b[5]; UARTi continuous receive mode enable bit  [0; Continuous receive mode disabled] */
        /* b[6]; Data logic select bit                     [0; No reverse] */
        /* b[7]; Clock divide synchronizing stop bit       [0; Synchronizing stop] */
```

```
#endif
/***********************************************************************/


    /* Stop the SCLK and the LRCK */
    Stop_SCLK;
    Stop_LRCK;

}
```

A_endian.a30

```
;""FILE COMMENT""*****************************************************************
; System Name      : Digital Audio I/F Driver sample
; File Name        : A_endian.a30
; Version          : 0.01
; Contents         : Digital Audio I/F Driver subroutine
;                          Convert endian
;
; CPU                : M32C/80 series
; Compiler         :
; Assembler        : AS308(Version 3.10 Release2)
; Note               : 8bits CSIO needs this module
;*****************************************************************************
;      Copyright(C)2003, Renesas Technology Corp.
;      Copyright(C)2003, Renesas Solutions Corp.
;      All rights reserved.
;*****************************************************************************
; History            : 2002.10.01 Ver 0.01
;                          :
;""FILE COMMENT END""*************************************************************


.section        program, align
.glb        __cnv_endian
;""SUBR COMMENT""*****************************************************************
; Module             : _cnv_endian
;                          void _cnv_endian ( unsigned char far *ram_buff,
;                                                  unsigned short Bsize )
;                          #pragma PARAMETER _cnv_endian( A0, R1 )
;-----------------------------------------------------------------------------
; Abstract           : Converts endian
; Function           : Converts endian of 2 bytes (WORD) data
;-----------------------------------------------------------------------------
; Inputs             : *ram_bur [A0] ; The address to the buffer which stored data to convert.
;                          : Bsize       [R1] ; The size of conversion bytes. (Set the even number to Bsize)
;
; Outputs            : *ram_buf    ; Converted data are stored to the buffer indicated by this pointer.
;
; Returns            : None
;-----------------------------------------------------------------------------
; Subroutines      : None
;-----------------------------------------------------------------------------
; Note                 : (1) Specify the even number bytes to the Bsize.
;                          : (2) Bsize can be specified from 2 to (FFFFh-1).
;                          : (3) Do not set 0 to the Bsize.
; History            :
;""SUBR COMMENT END""*************************************************************
__cnv_endian:
        pushm      A0, R1

        shl.w      #-2,        R1                ; cnt = cnt /4 (LONG)
        jnc          CONV_4BYTE                      ; if(cannot divide by 4)
                                                  ;       Previously, 2bytes are converted
        mov.w      [A0],      R0
        mov.b      R0H,        [A0]              ; Store Hi -->    [A0]
        mov.b      R0L,        1[A0]             ; Store Lo --> 1[A0]
        add.L:Q #2,          A0                ;      ram_buff+=2
        cmp.w      #0,         R1                ; if( cnt == 0 ) goto END_CNV
        jeq          END_CNV
```

```
;===============================================================================
CONV_4BYTE:
      mov.w     [A0],     R0              ;
      mov.b     R0H,      [A0]            ; Store Hi -->    [A0]
      mov.b     R0L,      1[A0]           ; Store Lo --> 1[A0]


      mov.w     2[A0],    R0              ;
      mov.b     R0H,      2[A0]           ; Store Hi --> 2[A0]
      mov.b     R0L,      3[A0]           ; Store Lo --> 3[A0]


      add.L:Q #4,         A0              ;     ram_buff+=4
      adjnz.w #-1, R1, CONV_4BYTE ;     cnt--
                                          ;     if ( cnt !=0 ) goto CONV_4BYTE
;===============================================================================
END_CNV:
      popm      A0, R1

RTS

.END
```

7. Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com/

Technical Support

E-mail: support_apl@renesas.com

Data Sheet

M32C/83 Group Preliminary REV. B3

(Use the latest version on the Homepage:http://www.renesas.com/)

REVISION HISTORY	Digital Audio I/F driver using the peripheral functions Application Note

Rev.	Date	Description	
		Page	Summary
1.10	Jul 25, 2003	-	First edition issued

Keep safety first in your circuit designs!

- Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
  Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
- Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
  The information described here may contain technical inaccuracies or typographical errors.
  Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
  Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
  Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.