SH7216 Group

Using the A/D Converter, Data Transfer Controller and Serial Communication Interface

R01AN0070EJ0100
Rev. 1.00
Nov. 01, 2010

Summary

This application note provides an example to use the SH7216 A/D Converter, Data Transfer Controller, and Serial Communication Interface.

Target Device

SH7216 MCU

Contents

1. Introduction

1.1 Specifications

Use the SH7216 Data Transfer Controller to transmit the A/D conversion value in serial as the following steps:

- The compare match interrupt activates the Data Transfer Controller, transfers the data to start the A/D conversion from the internal RAM to the A/D control register, and activates the A/D Converter.
- The A/D conversion end interrupt activates the Data Transfer Controller, and the Data Transfer Controller transfers the A/D conversion value to the internal RAM in block transfer mode. Also, the Data Transfer Controller transfers the data to enable the transmit data empty interrupt to the Serial control register using the chain transfer, and generates the transmit data empty interrupt.
- The transmit data empty interrupt activates the Data Transfer Controller to transfer the A/D conversion value from the internal RAM to the Serial Communication Interface, and transfer the data in asynchronous mode.

1.2 Modules Used

- Compare Match Timer (CMT)
- A/D Converter (ADC)
- Data Transfer Controller (DTC)
- Serial Communication Interface (SCI)

1.3 Applicable Conditions

MCU	SH7216
Operating Frequencies	Internal clock: 200 MHz Bus clock: 50 MHz Peripheral clock: 50 MHz AD clock: 50 MHz
Integrated Development Environment	Renesas Electronics Corporation High-performance Embedded Workshop Ver.4.07.00
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ Compiler Package Ver.9.03 Release 00
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2afpu -fpu=single -debug -gbr=auto -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1)

1.4 Related Application Note

For more information, refer to the following application note:

- SH7216 Group Example of Initialization

2. Applications

2.1 Overview of Modules

2.1.1 A/D Converter

The A/D Converter includes two A/D modules (A/D_0 and A/D_1) to input four channels with 12-bit resolution. Data converted by the A/D Converter is stored in the A/D data register (ADDR).

A/D Converter operates in single-cycle scan mode, and continuous scan mode. In single-cycle scan mode, the A/D Converter converts the input analog voltage to digital on one or more channels specified, and enters the A/D conversion wait state. In continuous scan mode, the A/D Converter converts the input analog voltage to digital repeatedly on one or more channels specified. When converting analog to digital is completed, the A/D conversion end interrupt can be generated to the CPU. The Direct Memory Access Controller and Data Transfer Controller can be activated when the A/D conversion end interrupt occurs (note).

Table 1 lists the A/D Converter specifications. Figure 1 shows its block diagram. For more information, refer to the A/D Converter (ADC) chapter in the SH7214 Group, SH7216 Group Hardware User's Manual.

Note: When the Direct Memory Access Controller is activated, the CPU interrupt is not generated. The Direct Memory Access Controller can only be activated by A/D module_0 (A/D_0).

Table 1 A/D Converter Specifications

Item	Description
Resolution	12-bit
Conversion speed	Minimum conversion timer per channel: 1.0 µs (Aφ is operating at 50 MHz)
Number of modules	2 (A/D_0, A/D_1)
Number of input channels	8 (AN0 to AN7)
Operating mode	• Single-cycle scan mode • Continuous scan mode
Sample-and-hold function	• Channels 0 to 3 share one circuit, channels 4 to 7 share one circuit • Channels 0 to 2 have dedicated circuits for each channel (3 circuits in total)
A/D conversion trigger	• Software: ADST bit setting • Timer: — TRGAN, TRG0N, TRG4AN, and TRG4BN from the Multi-function Timer Pulse Unit 2 — TRGAN, TRG4AN, and TRG4BN from the Multi-function Timer Pulse Unit 2 • External trigger: $\overline{\text{ADTRG}}$

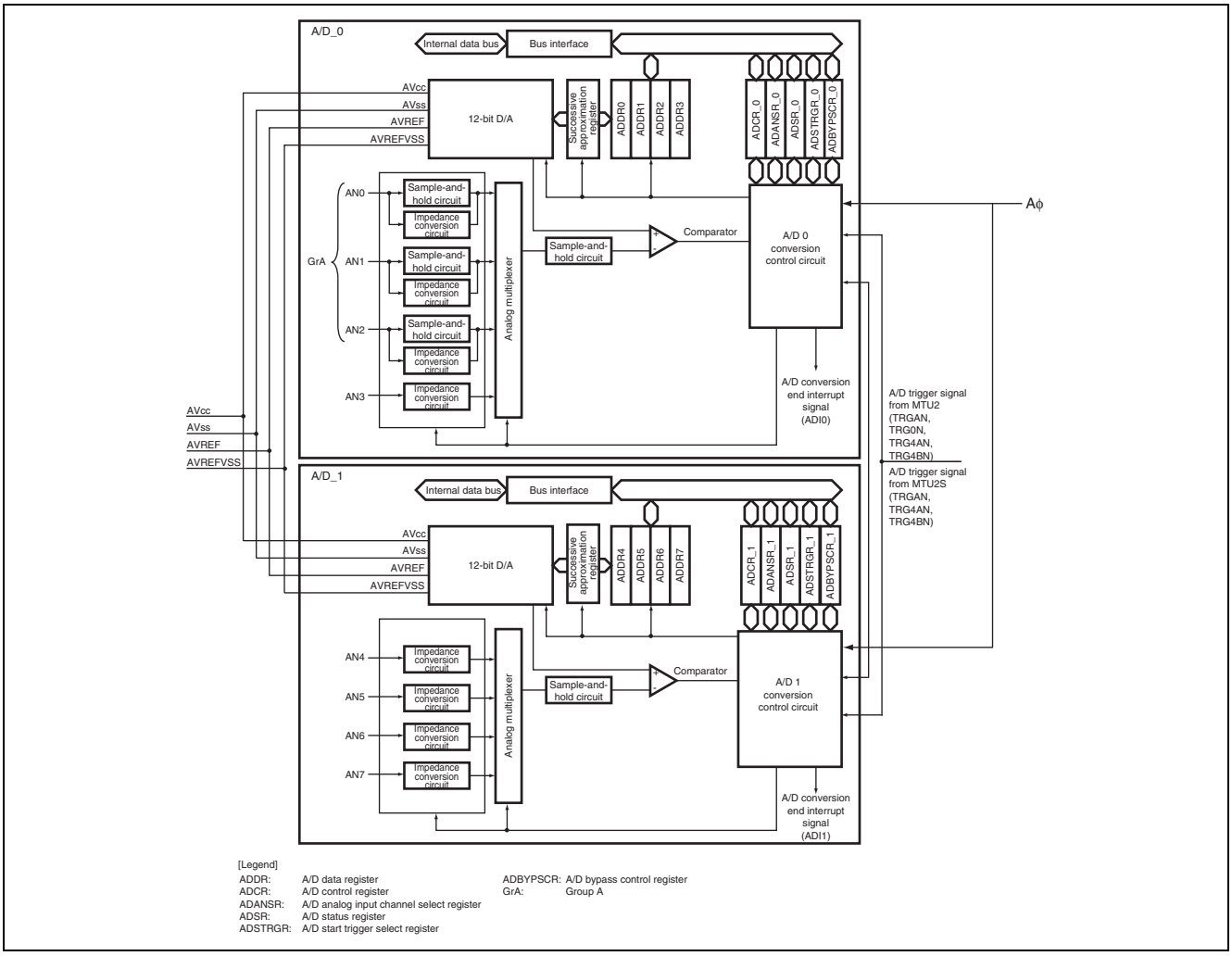


Figure 1 A/D Converter Block Diagram

2.1.2 Data Transfer Controller

The Data Transfer Controller is a module to transfer data activated by an interrupt request. It has three operation modes; normal transfer mode, repeat transfer mode, and block transfer mode. It stores the transfer information in the data area to transfer data on the specified number of channels. When it is activated, it reads the transfer information from the data area to transfer data, and writes back the transfer information after transferring the data.

For more information, refer to the Data Transfer Controller (DTC) chapter in the SH7214 Group, SH7216 Group Hardware User's Manual.

Table 2 lists the features of the Data Transfer Controller. Figure 2 shows its block diagram.

Table 2 Data Transfer Controller Specifications

Item	Description
Number of channels	Any number of channels can be set
Transfer mode	• Normal mode • Repeat mode • Block transfer mode
Number of transfers	• Normal mode: 1 to 65536 • Repeat mode: 1 to 256 • Block transfer mode: 1 to 65536
Transfer data size	Bytes, words (2 bytes), and longwords (4 bytes)
CPU interrupt request	• A CPU interrupt using the DTC can be requested • A CPU interrupt can be requested after transferring the specified data is completed
Activate source	• External pins (8 sources) • On-chip peripheral modules (55 sources)
Other	• Multiple data can be transferred to single activation source (chain transfer) • Reading the transfer information can be skipped • Module stop mode can be specified • Short address mode can be specified • Timing to release buses can be selected among three options • Priority to activate the DTC can be selected from two options

Note: Either the transfer source or transfer destination must be set as the on-chip peripheral module. Data cannot be transferred among external memory, memory-mapped external device and on-chip memory.

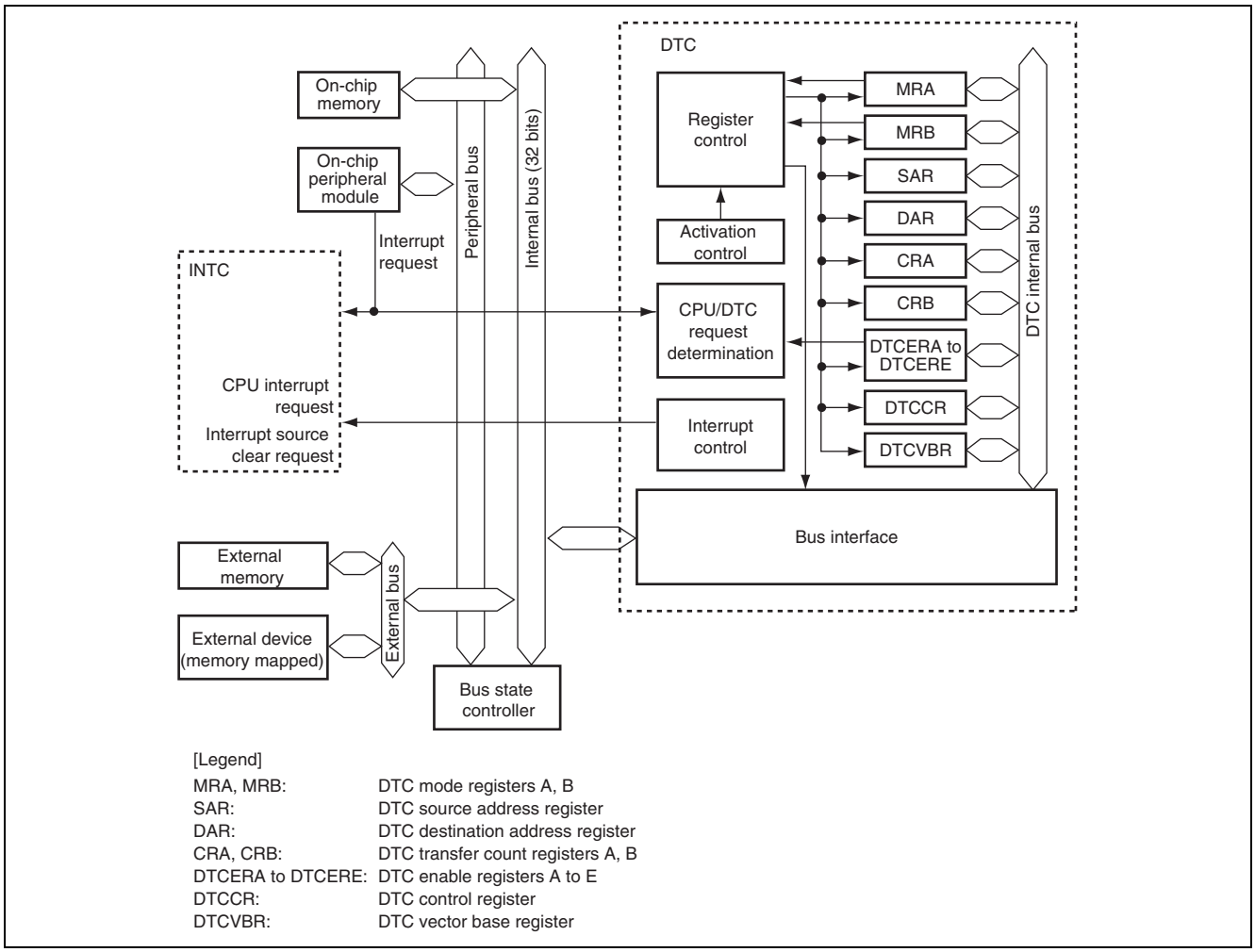


Figure 2 Data Transfer Controller Block Diagram

2.1.3 Serial Communication Interface (Asynchronous Mode)

The SH7216 Serial Communication Interface transmits or receives a "character", appending a start bit which indicates the initiation of the communication, and a stop bit which indicates the end of the communication to data. Then, the SH7216 SCIF handles communication in sync per character. Each channel has an independent transmitter and receiver to transmit and receive data at the same time. As the transmitter and receiver are double-buffered, the serial data can be transmitted or received in series at high speed.

The transmission line is normally held in mark (high) state. The SCI monitors the line and starts serial communication when the line goes to space (low) state, indicating a start bit.

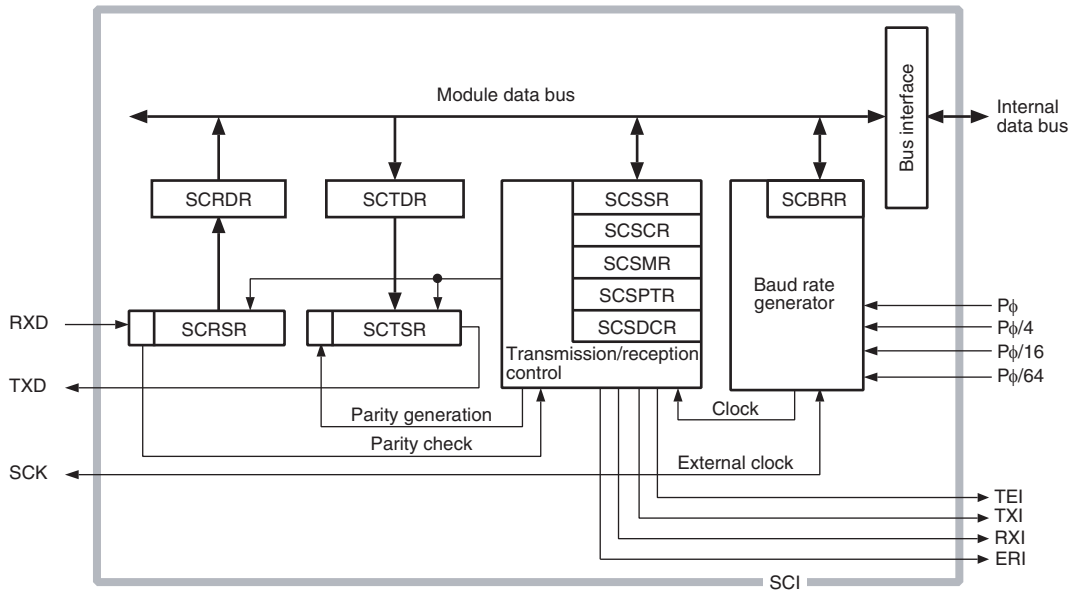
A serial character consists of a start bit (low), data (LSB first), parity bit (high or low), and stop bit (high).

For more information, refer to the Serial Communication Interface (SCI) chapter in the SH7214 Group, SH7216 Group Hardware User's Manual.

Table 3 lists the specifications of the asynchronous serial communication. Figure 3 shows the block diagram of the Serial Communication Interface.

Table 3 Asynchronous Serial Communication Specifications

Item	Description
Number of channels	4 (SCI0, SCI1, SCI2, and SCI4)
Clock source	Either internal clock or external clock can be specified: • Internal clock: Pφ, Pφ/4, Pφ/16, and Pφ/64 — SCI operates at the baud rate generator clock, and can output • External clock: SCK pin input clock — Repeat mode
Data format	• Transfer data length: 7-bit or 8-bit can be specified • Transfer order: LSB first or MSB first can be specified (When specifying 7-bit, only LSB first can be selected)
Baud rate	• Internal clock: 100 bps to 1.5625 Mbs (Pφ is operating at 50 MHz) • External clock: 781250 bps (max.) (Pφ is operating at 50 MHz, and external clock input is at 15 MHz)
Error detection	Framing error, parity error, overrun error, and break can be detected
Interrupt request	• Transmit data empty interrupt (TXI) • Receive data full interrupt (RXI) • Receive error interrupt (ERI) • Transmit end interrupt (TEI)



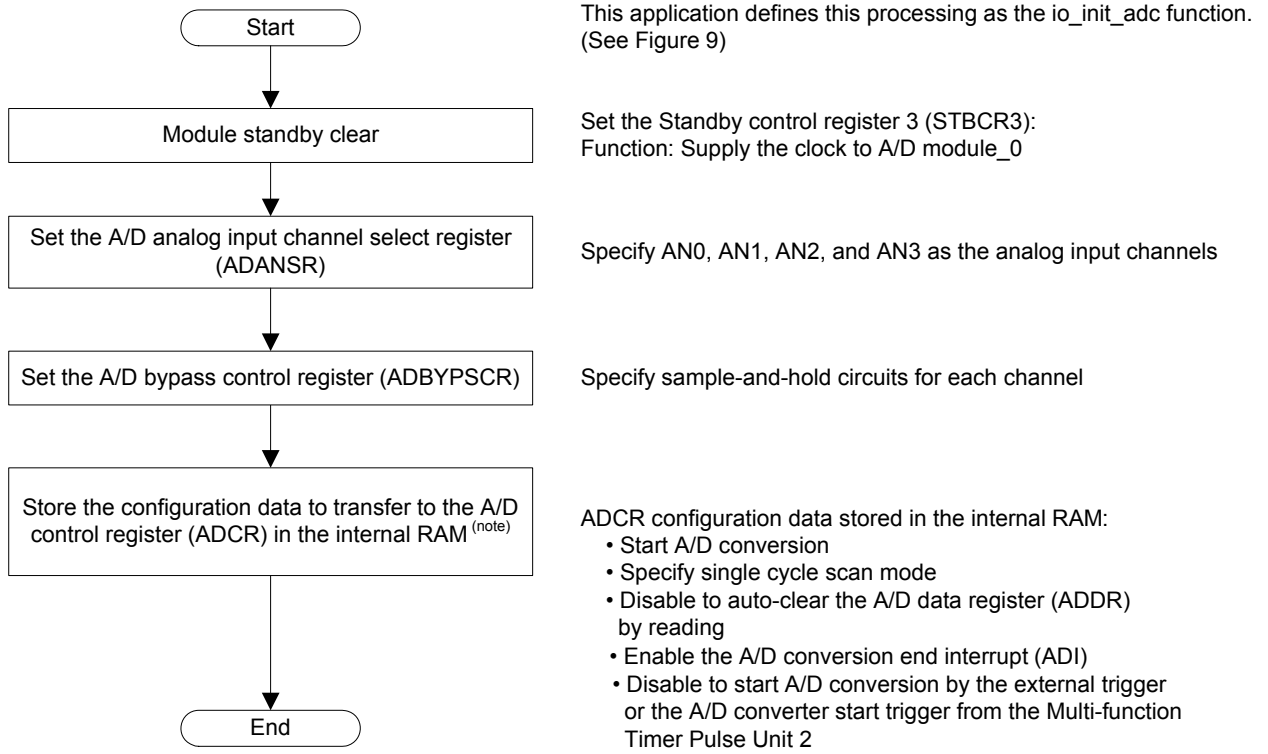
[Legend]
SCRSR: Receive shift register
SCRDR: Receive data register
SCTSR: Transmit shift register
SCTDR: Transmit data register
SCSMR: Serial mode register
SCSCR: Serial control register
SCSSR: Serial status register
SCBRR: Bit rate register
SCSPTR: Serial port register
SCSDCR: Serial direction control register

Figure 3 Serial Communication Interface Block Diagram

2.2 Configuration Procedure

2.2.1 Configuring the A/D Converter

Figure 4 shows the flow chart for configuring the A/D Converter used in this application. For more information on register settings, refer to the SH7214 Group, SH7216 Group Hardware User's Manual.



Note: The Data Transfer Controller transfers the A/D conversion value to set the ADCR in this application.

Figure 4 Flow Chart for Configuring the A/D Converter

2.2.2 Configuring the Data Transfer Controller

Figure 5 shows the flow chart for configuring the Data Transfer Controller used in this application. For more information on register settings, refer to the SH7214 Group, SH7216 Group Hardware User's Manual.

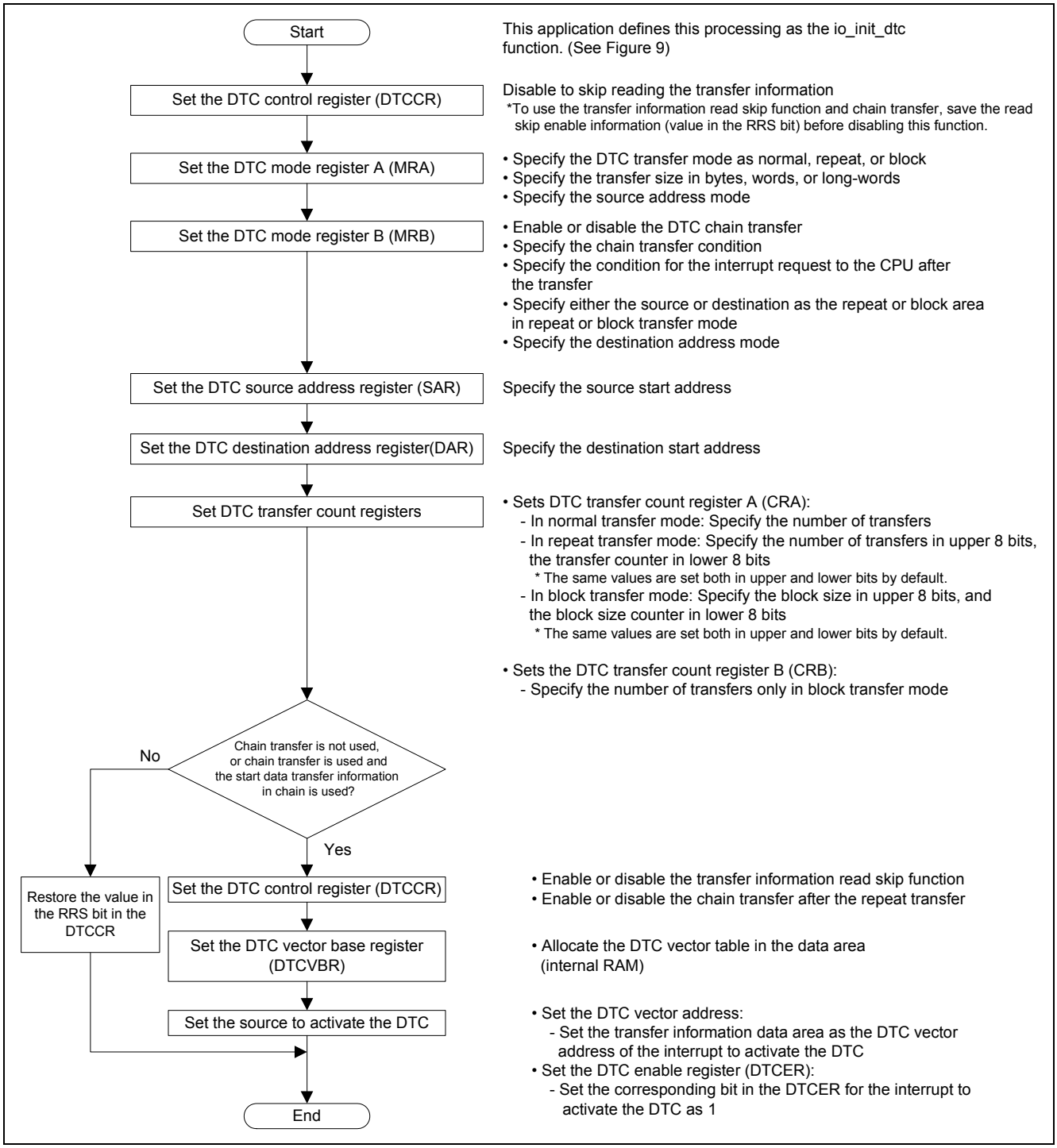
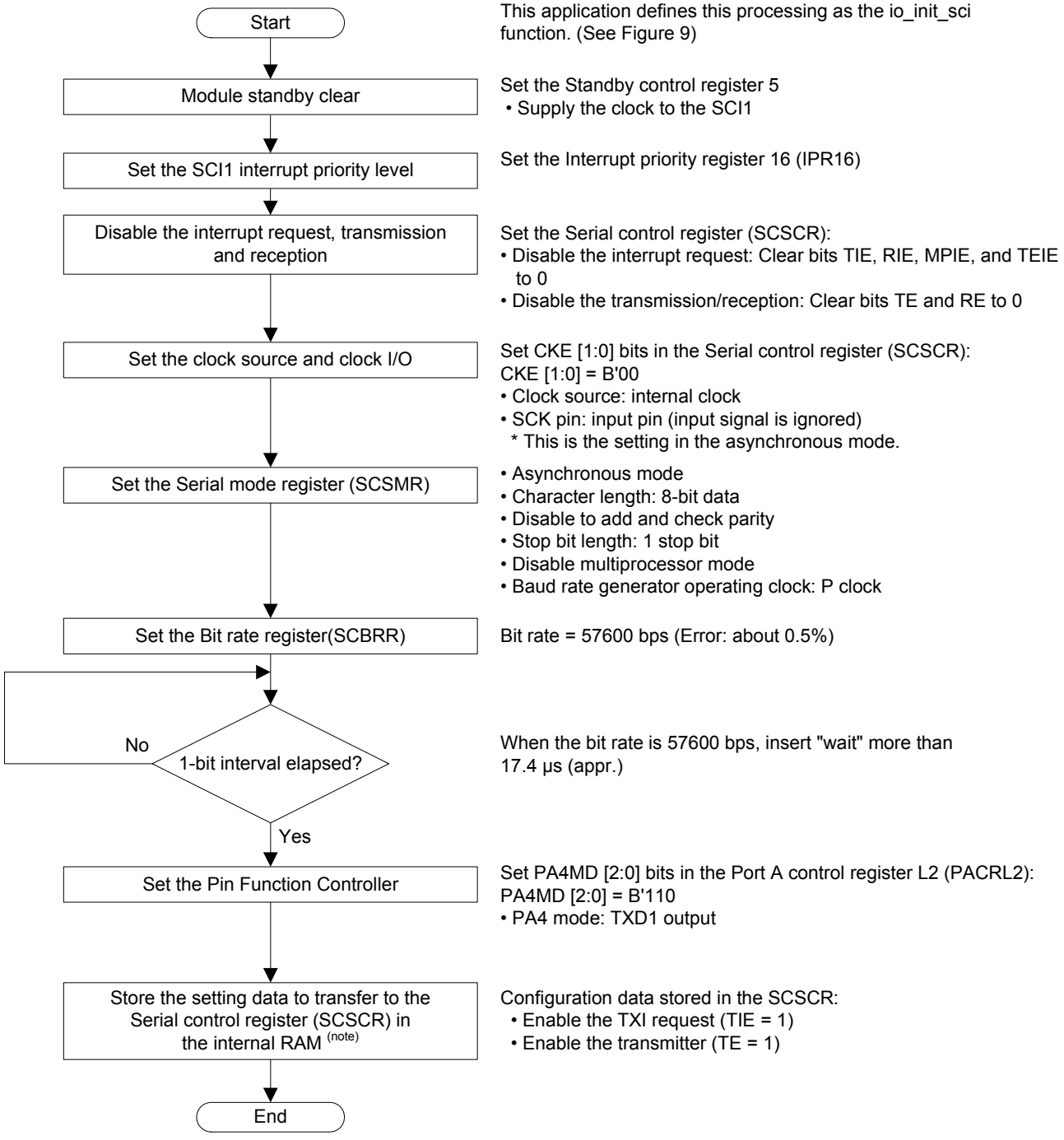


Figure 5 Flow Chart for Configuring the Data Transfer Controller

2.2.3 Configuring the Serial Communication Interface

Figure 6 shows the flow chart for configuring the Serial Communication Interface used in this application. For more information on register settings, refer to the SH7214 Group, SH7216 Group Hardware User's Manual.



Note: The Data Transfer Controller transfers the setting data to the SCSCR in this application.

Figure 6 Flow Chart for Configuring the Serial Communication Interface

2.3 Sample Program Procedure

2.3.1 Sample Program Operation

The sample program activates the Data Transfer Controller multiple times in a row by three interrupt sources, writes the A/D conversion results in four channels in the internal RAM. Then, it transfers the conversion results to the Serial Communication Interface, and transmits data continuously. The Data Transfer Controller repeats a series of processing for 10 times in every 5-ms period.

This section describes three interrupt sources to activate the Data Transfer Controller, and transferred data. Table 4 lists the Data Transfer Controller operation specifications.

1. Compare match interrupt (CMI0)
   When the Data Transfer Controller is activated by the CMI0, it transfers the data to start the A/D conversion from the internal RAM to the A/D control register. When the transfer is completed, the A/D Converter is activated, and starts A/D conversion for four channels (AN0 to AN3). The A/D conversion end interrupt (ADI0) occurs after the A/D conversion is completed.

2. A/D conversion end interrupt (ADI0)
   When the Data Transfer Controller is activated by the ADI0, it uses the chain transfer to transfer data as follows;
   (1) It transfers the data to enable the Transmit data empty interrupt (TXI1) which is pre-stored in the internal RAM to the Serial control register.
   (2) It uses block transfer mode to transfer the A/D conversion result for four channels (A/D data registers 0 to 3) to the internal RAM.
   TXI1 is generated when the transfer (1) is completed, however, the Data Transfer Controller which is activated by the TXI1 transfers data after both (1) and (2) are completed.

3. Transmit data empty interrupt (TXI1)
   When the Data Transfer Controller is activated by the TXI1, it transfers the A/D conversion result for four channels (total: 8 bytes) from the internal RAM to the Transmit data register, and transmits serial data.

Table 4 Data Transfer Controller Operation Specifications

Activation Source	Source	Destination	Number of transfers	Remarks
Compare match interrupt (CMI0)	Internal RAM	A/D control register (ADCR_0)	10	Stores the transfer source data in the internal RAM when configuring the compare match timer
A/D conversion end interrupt (ADI0)	Internal RAM	Serial control register (SCSCR_1)	10	• Stores the transfer source data in the internal RAM when configuring the ADC • Uses chain transfer
	A/D data registers (ADDR0 to 3)	Internal RAM	10	Uses block transfer mode (Block size: 4 words)
Transmit data empty interrupt (TXI1)	Internal RAM	Transmit data register (SCTDR_1)	8	Number of transfers are equal to the number of continuous serial transmission of A/D conversion results (8 bytes) for four channels

After the CPU is reset, the sample program configures the compare match timer, A/D Converter, Data Transfer Controller, and Serial Communication Interface in main processing.

1. In the A/D Converter configuration, the sample program stores the data to enable the A/D conversion start request and A/D conversion end interrupt (ADI0) to set to the A/D control register (ADCR_0) in the internal RAM. In the Data Transfer Controller configuration, it sets the transfer information to 3 activation sources. In the Serial Communication Interface configuration, the sample program stores the data to enable the transmit data empty interrupt (TXI1) and the transmission to set to the Serial control register (SCSCR_1) in the internal RAM.
2. After the above configuration is completed, the sample program starts the compare match timer. The compare match interrupt (CMI0) request is generated every 5-ms period after the compare match timer starts counting. These interrupt requests activate the Data Transfer Controller, and the Data Transfer Controller transfers the data to start the A/D conversion and the data to enable ADI0 from the internal RAM to the ADCR_0.
3. Then, the A/D Converter is activated to execute A/D conversion for four channels (AN0 to AN3). After the A/D conversion is completed, the ADI0 request is generated. This ADI0 request activates the Data Transfer Controller again, and transfers the TXI1 and the data to enable the transmitter from the internal RAM to the SCSCR_1. After that, it transfers the A/D conversion results for four channels to the internal RAM using the chain transfer in block transfer mode.
4. After the above transfer is completed, TXI1 requests (total: 8 times) activates the Data Transfer Controller to transfer A/D conversion results (total: 8 bytes) for four channels from the internal RAM to the Transmit data register (SCTDR_1), and outputs the data from the serial port.
5. When the Data Transfer Controller completes to output the A/D conversion results for four channels, the TXI1 request is generated to the CPU. As the sample program sets the Data Transfer Controller transfer information within the TXI1 interrupt, the sample program repeats a series of processing steps 1 to 5 as described above.

The sample program repeats a series of processing for the number of transfers (10 times) of the Data Transfer Controller used to activate both CMI0 and ADI0. When it completes to transfer data for 10 times, the sample program clears the Transfer end (TEND) flag in the TXI1 interrupt to 0, enables the transmit end interrupt (TEI1), and disables the transmitter in the TEI1 interrupt (note).

When the above processing is completed, the CMI0 request is generated to the CPU. The sample program stops the counting (compare match timer) in the CMI0 interrupt.

Figure 7 and Figure 8 show overview of the sample program.

Note: When the TXI activates the Data Transfer Controller to write data to the SCTDR, the status of the TEND flag is undefined. Therefore, the flag cannot be used as the transmit end flag. The sample program uses the TEI to determine the transmission is completed.

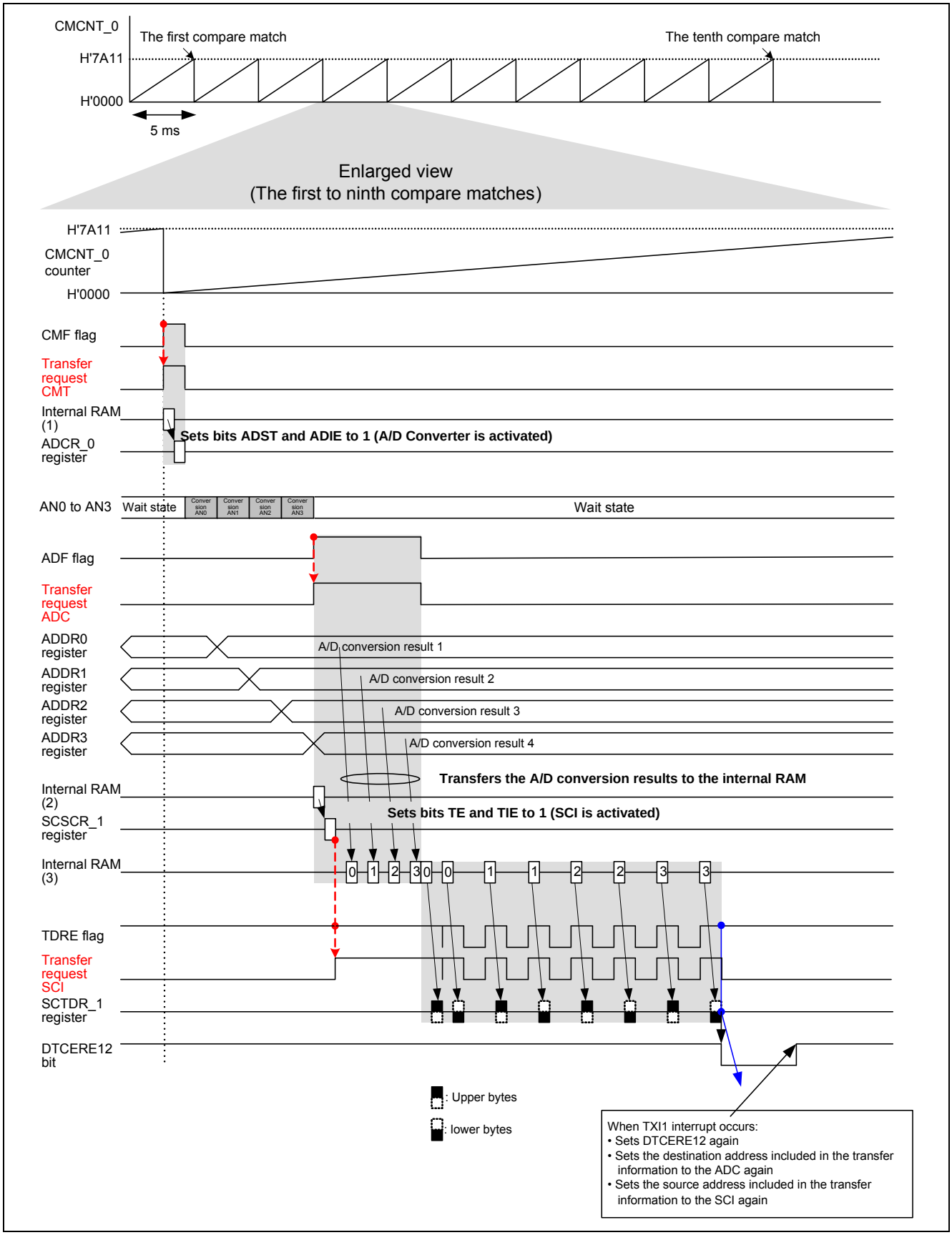


Figure 7 Sample Program Operation (Overview, 1/2)

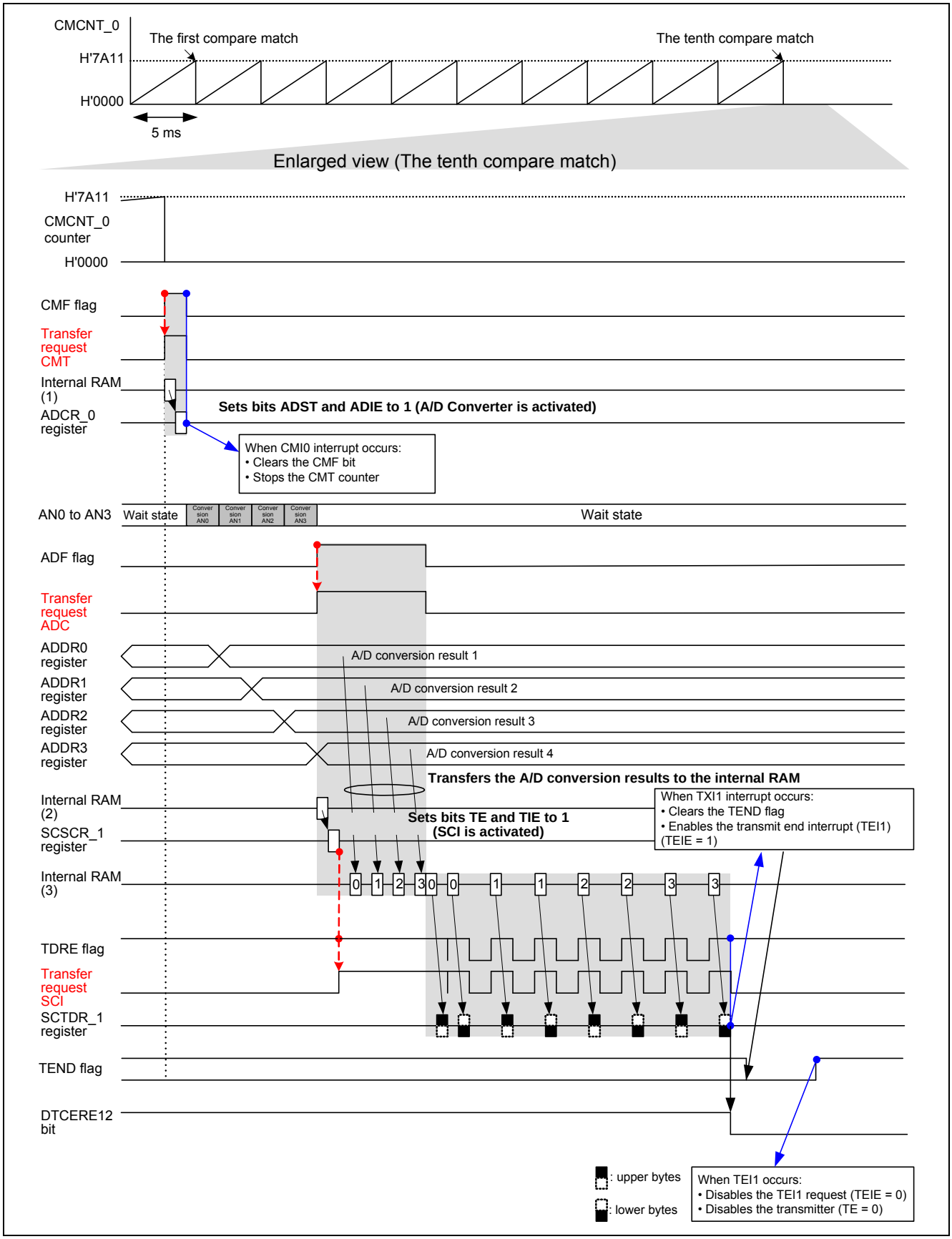


Figure 8 Sample Program Operation (Overview, 2/2)

2.3.2 A/D Converter Register Setting

Table 5 lists the register settings for the A/D Converter.

The following setting enables the A/D conversion end interrupt, however, the CPU interrupt is not generated because this application uses chain transfer when the Data Transfer Controller is activated.

Table 5 A/D Converter Register Setting

Register Name	Address	Setting	Description
Standby control register 3 (STBCR3)	H'FFFE 0408	H'7A	• MSTP32 = "0": Supplies the clock to A/D_0
A/D control register_0 (ADCR_0)	H'FFFF E800	H'90 (note)	• ADST = "1": Starts the A/D conversion • ADCS = "0": Specifies single-cycle scan mode • ACE = "0": Disables to clear the ADDR register automatically by reading the ADDR register • ADIE = "1": Enables to generate the A/D conversion end interrupt • TRGE = "0": Disables the A/D conversion by the external trigger input or the A/D converter start trigger from the Multi-function Timer Pulse Unit 2 • EXTRG = "0": Activates the A/D Converter by the A/D converter start trigger from the Multi-function Timer Pulse Unit 2
A/D analog input channel select register_0 (ADANSR_0)	H'FFFF E820	H'0F	• ANS3 = "1": • ANS2 = "1": • ANS1 = "1": • ANS0 = "1": Specifies analog input channels AN0 to AN3
A/D bypass control register_0 (ADBYPSCR_0)	H'FFFF E830	H'01	• SH = "1": Specifies the exclusive sample-and-hold circuit for each channel

Note: When the A/D Converter is configured, this application stores the setting value in the internal RAM, and the Data Transfer Controller which is activated by CMI0 sets the register.

2.3.3 Data Transfer Controller Register Setting

Table 6 to Table 10 show the register settings for the Data Transfer Controller.

Table 6 Data Transfer Controller Register Setting (Common)

Register Name	Address	Setting	Description
DTC enable register A (DTCERA)	H'FFFE 6000	H'0088	• DTCE7 = "1": Sets the interrupt source as ADI0 • DTCE3 = "1": Sets the interrupt source as CMI0
DTC enable register E (DTCERE)	H'FFFE 6008	H'1000	• DTCE12 = "1": Sets the interrupt source as TXI1
DTC control register (DTCCR)	H'FFFE 6010	H'00	• RRS = "0": Disables the DTC transfer information read skip flag • RCHNE = "0": Disables the chain transfer after repeat transfer
DTC vector base register (DTCVBR)	H'FFFE 6014	H'FFF8 5000	Vector address when calculating the DTC vector table address

Table 7 Data Transfer Controller Register Setting (Activation Source: CMI0)

Register Name	Address (note)	Setting	Description
DTC mode register A (MRA)	H'FFF8 4000	H'00	• MD [1:0] = "B'00": Normal transfer mode • Sz [1:0] = "B'00": Byte-size transfer • SM [1:0] = "B'00": SAR is fixed after a transfer is completed
DTC mode register B (MRB)	H'FFF8 4001	H'00	• CHNE = "0": Disables the chain transfer • CHNS = "0": Repeats the chain transfer • DISEL = "0": Interrupt occurs when the specified number of transfers is completed • DTS = "0" Specifies the destination as repeat or block area • DM [1:0] = "B'00": DAR is fixed after the transfer
DTC source address register (SAR)	H'FFF8 4004	H'FFF8 4108	Transfer source address: Address on the internal RAM storing the data to start converting A/D_0 (H'90)
DTC destination address register (DAR)	H'FFF8 4008	H'FFFF E800	Transfer destination address: A/D control register_0 (ADCR_0)
DTC transfer count register A (CRA)	H'FFF8 400C	H'000A	Number of transfers: 10

Note: Address refers to the area on memory to store the transfer information in this application, and it depends on the user setting.

Table 8 Data Transfer Controller Register Setting (Activation Source: ADI0) (1/2)

Register Name	Address (note)	Setting	Description
DTC mode register A (MRA)	H'FFF8 4010	H'00	• MD [1:0] = "B'00": Normal transfer mode • Sz [1:0] = "B'00": Byte-size transfer • SM [1:0] = "B'00": SAR is fixed after a transfer is completed
DTC mode register B (MRB)	H'FFF8 4011	H'80	• CHNE = "1": Enables the chain transfer • CHNS = "0": Repeats the chain transfer • DISEL = "0": Interrupt occurs when the specified number of transfers is completed • DTS = "0" Specifies the destination as repeat or block area • DM [1:0] = "B'00": DAR is fixed after a transfer
DTC source address register (SAR)	H'FFF8 4014	H'FFF8 410C	Transfer source address: Address on the internal RAM storing the data to enable TXI1 (H'A0)
DTC destination address register (DAR)	H'FFF8 4018	H'FFFF 8804	Transfer destination address: Serial control register 1 (SCSCR_1)
DTC transfer count register A (CRA)	H'FFF8 401C	H'000A	Number of transfers: 10

Note: Address refers to the area on memory to store the transfer information in this application, and it depends on the user setting.

Table 9 Data Transfer Controller Register Setting (Activation Source: ADI0) (2/2)

Register Name	Address (note)	Setting	Description
DTC mode register A (MRA)	H'FFF8 4020	H'98	• MD [1:0] = "B'10": Block transfer mode • Sz [1:0] = "B'01": Word-size transfer • SM [1:0] = "B'10": Increments SAR after a transfer is completed
DTC mode register B (MRB)	H'FFF8 4021	H'18	• CHNE = "0": Disables the chain transfer • CHNS = "0": Repeats the chain transfer • DISEL = "0": Interrupt occurs when the specified number of transfers is completed • DTS = "1" Specifies the source as block area • DM [1:0] = "B'10": Increments DAR after a transfer is completed
DTC source address register (SAR)	H'FFF8 4024	H'FFFF E840	Transfer source address: A/D data register 0 (ADDR0)
DTC destination address register (DAR)	H'FFF8 4028	H'FFF8 4102	Transfer destination address: Start address in the internal RAM storing the A/D conversion results (AN0 to AN3)
DTC transfer count register A (CRA)	H'FFF8 402C	H'0404	• CRAH (Upper 8 bits): Block size = 4 words • CRAL (Lower 8 bits): Block size counter = 4 (Specified as the same as the block size by default)
DTC transfer count register B (CRB)	H'FFF8 402E	H'000A	Number of transfers: 10

Note: Address refers to the area on memory to store the transfer information in this application, and it depends on the user setting.

Table 10 Data Transfer Controller Register Setting (Activation Source: TXI1)

Register Name	Address (note)	Setting	Description
DTC mode register A (MRA)	H'FFF8 4030	H'08	• MD [1:0] = "B'00": Normal transfer mode • Sz [1:0] = "B'00": Byte-size transfer • SM [1:0] = "B'10": Increments SAR after a transfer is completed
DTC mode register B (MRB)	H'FFF8 4031	H'00	• CHNE = "0": Disables the chain transfer • CHNS = "0": Repeats the chain transfer • DISEL = "0": Interrupt occurs when the specified number of transfers is completed • DTS = "0" Specifies the destination as repeat or block area • DM [1:0] = "B'00": DAR is fixed after the transfer
DTC source address register (SAR)	H'FFF8 4034	H'FFF8 4102	Transfer source address: Start address in the internal RAM storing the A/D conversion results (AN0 to AN3)
DTC destination address register (DAR)	H'FFF8 4038	H'FFFF 8806	Transfer destination address: Transmit data register 1 (SCTDR_1)
DTC transfer count register A (CRA)	H'FFF8 403C	H'0008	Number of transfers: 8

Note: Address refers to the area on memory to store the transfer information in this application, and it depends on the user setting.

2.3.4 Serial Communication Interface Register Setting

Table 11 lists the register settings for the Serial Communication Interface.

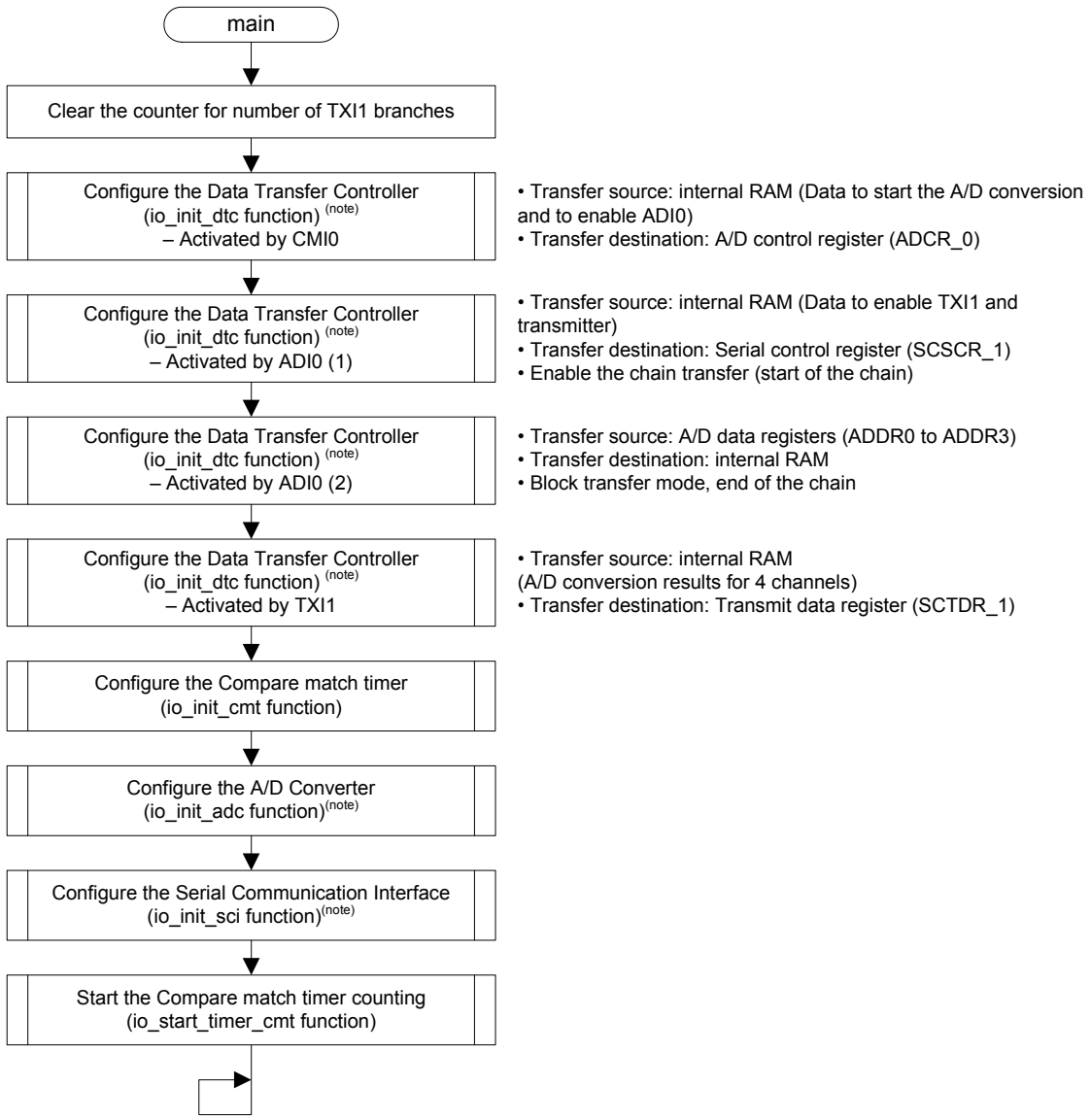
Table 11 Serial Communication Interface Register Setting

Register Name	Address	Setting	Description
Standby control register 5 (STBCR5)	H'FFFE 0418	H'BF	• MSTP56 = "0": Supply the clock to SCI1
Interrupt priority register (IPR16)	H'FFFE 0C14	H'0F00	SCI1 interrupt priority level: 15
Port A control register L2 (PACRL2)	H'FFFE 3814	H'0006	• PA4MD [2:0] = "B'110": Specify the PA4 to TXD1 output
Serial mode register_1 (SCSMR_1)	H'FFFF 8800	H'00	• C/$\overline{A}$ = "0": Clock asynchronous mode • CHR = "0": 8-bit data • PE = "0": Disables to add and check the parity bit • STOP = "0": 1 stop bit • MP ="0" Disables the multiprocessor mode • CKS [1:0] ="B'00: Peripheral clock
Bit rate register_1 (SCBRR_1)	H'FFFF 8802	H'1A	Bit rate setting : 57870.37 bps (Assumed bit rate is 57600 bps)
Serial control register_1 (SCSCR_1) (note)	H'FFFF 8804	H'A0 (note)	• TIE = "1": Enables the transmit data empty interrupt (TXI) • TE = "1": Enables the transmitter • CKE [1:0] = "B'00": Internal clock/SCK pin is input (Input signal is ignored)

Note: When the A/D Converter is configured, this application stores the setting value in the internal RAM, and the Data Transfer Controller which is activated by the ADI0 sets the register.

2.3.5 Sample Program Flow Chart

Figure 9 to Figure 14 show the flow charts of the sample program.



Note: Refer to 2.2 Configuration Procedure for flow charts of the io_init_adc, io_init_dtc, and io_init_sci.

Figure 9 Sample Program Flow Chart

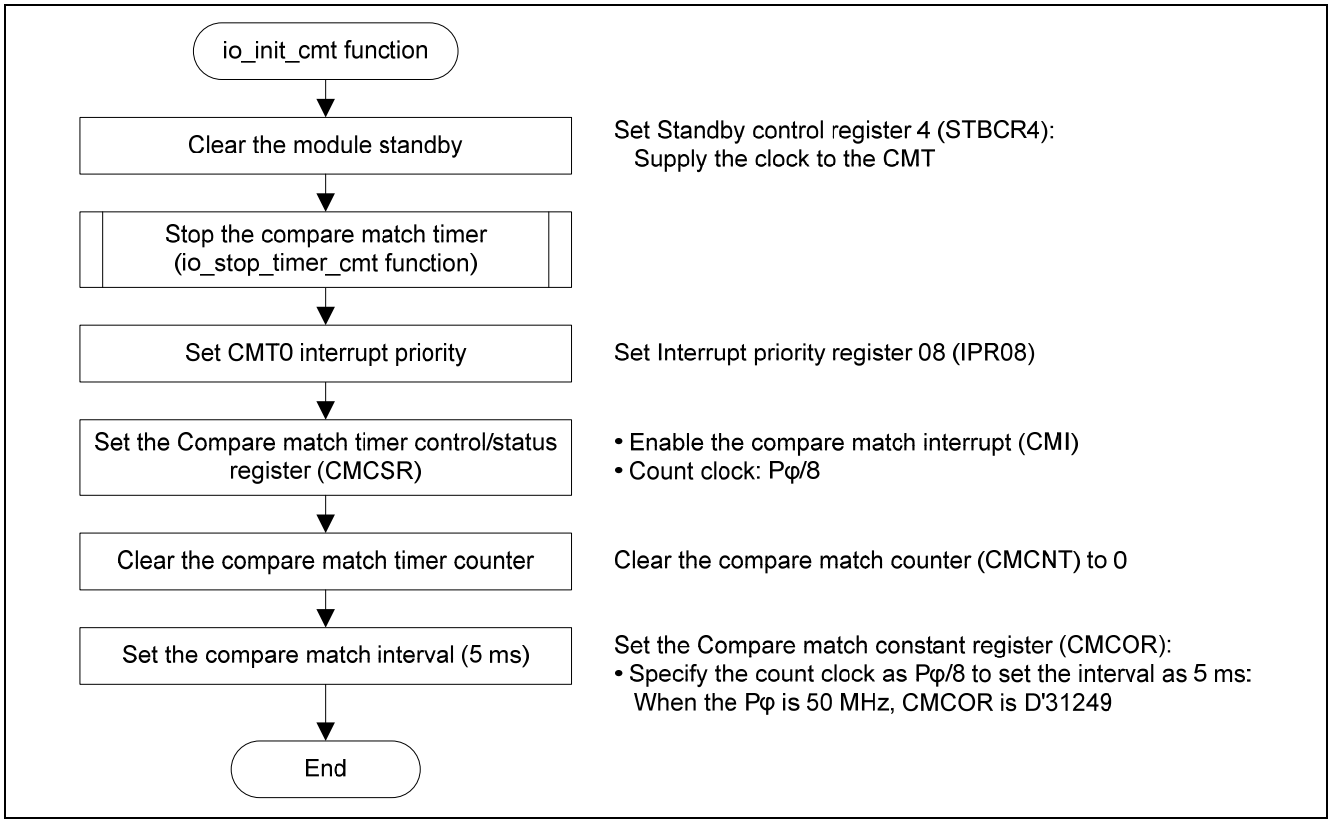


Figure 10 Flow Chart for Configuring the Compare Match Timer

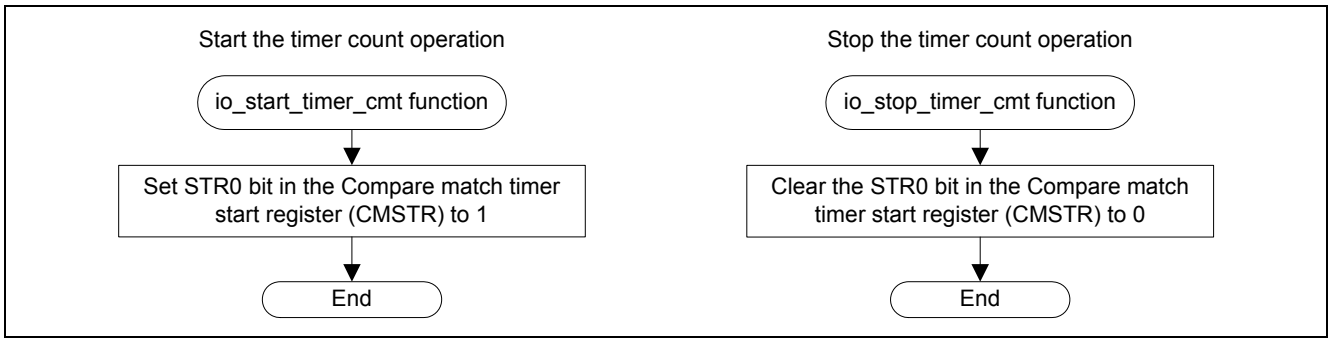


Figure 11 Flow Chart for Controlling the Compare Match Timer Count Operation

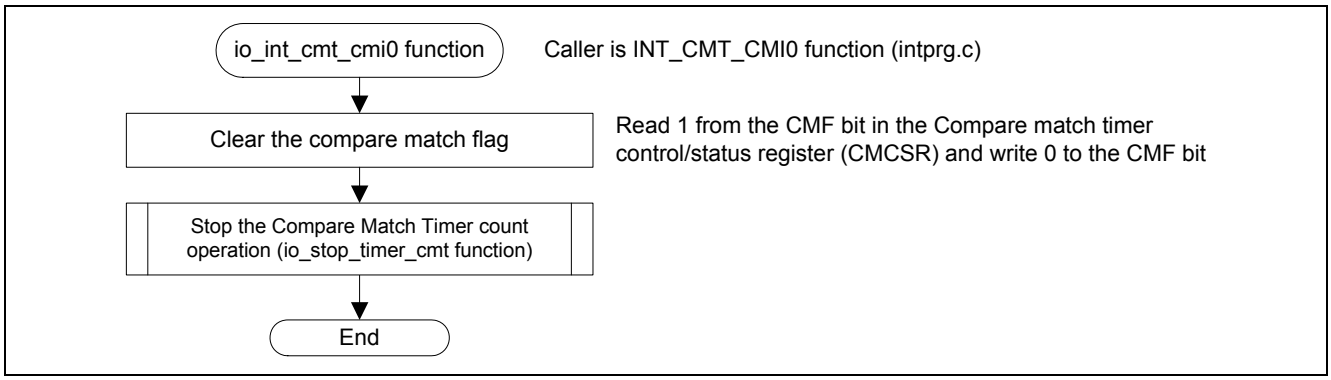


Figure 12 Compare Match Interrupt (CMI0) Flow Chart

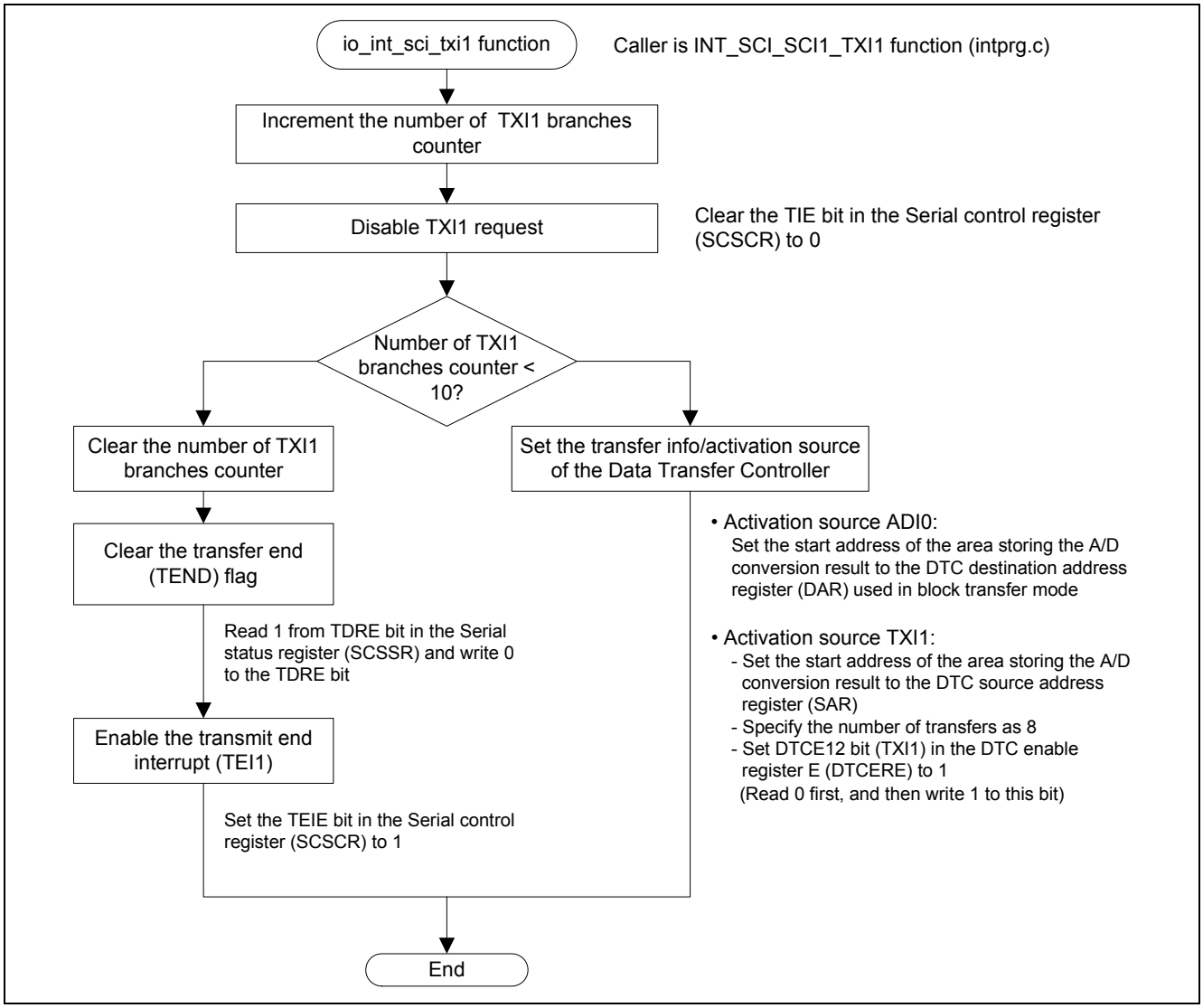


Figure 13 Transmit Data Empty Interrupt (TXI1) Flow Chart

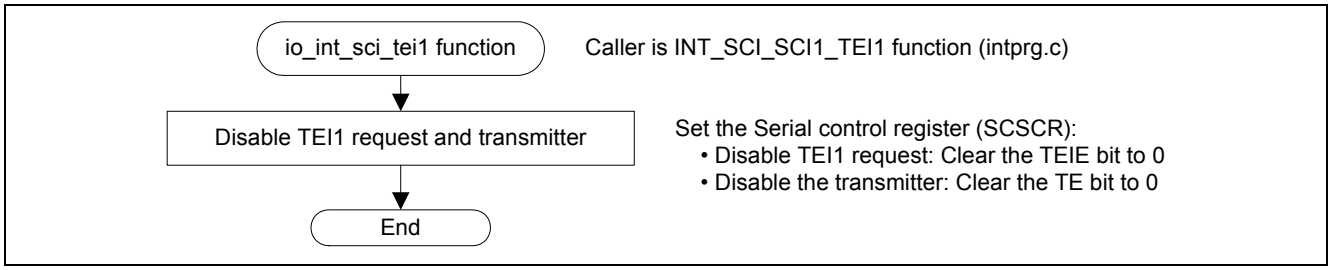


Figure 14 Transmit End Interrupt (TEI1) Flow Chart

3. Sample Program Listing

3.1 Sample Program Listing "main.c" (1/14)

```
1     /***********************************************************************
2     *   DISCLAIMER
3     *
4     *   This software is supplied by Renesas Electronics Corporation and is only
5     *   intended for use with Renesas products.  No other uses are authorized.
6     *
7     *   This software is owned by Renesas Electronics Corporation and is protected under
8     *   all applicable laws, including copyright laws.
9     *
10    *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    *   DISCLAIMED.
15    *
16    *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21    *
22    *   Renesas reserves the right, without notice, to make changes to this
23    *   software and to discontinue the availability of this software.
24    *   By using this software, you agree to the additional terms and
25    *   conditions found by accessing the following link:
26    *   http://www.renesas.com/disclaimer
27    ************************************************************************
28    *   Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
29    ************************************************************************/
30    /*""FILE COMMENT""************ Technical reference data ************************
31    *   System Name : SH7216 Sample Program
32    *   File Name   : main.c
33    *   Abstract    : ADC+DTC+SCI Module Application
34    *   Version     : 1.00.00
35    *   Device      : SH7216
36    *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37    *               : C/C++ compiler package for the SuperH RISC engine family
38    *               :                             (Ver.9.03 Release00).
39    *   OS          : None
40    *   H/W Platform: R0K572167 (CPU board)
41    *   Description :
42    ************************************************************************
43    *   History     : Nov.01,2010 Ver.1.00.00
44    *""FILE COMMENT END""*********************************************************/
45    #include <machine.h>
46    #include "iodefine.h"
47    #include "dtc.h"
48
```

3.2 Sample Program Listing "main.c" (2/14)

```
49      /* ==== Prototype declaration ==== */
50      void main(void);
51      void io_init_adc(void);
52      void io_init_dtc(DTC_TRANS_INFO *dtc_info, unsigned long src,
53                  unsigned long dst, unsigned long count, unsigned long mode);
54      void io_init_sci(void);
55      void io_init_cmt(void);
56      void io_start_timer_cmt(void);
57      void io_stop_timer_cmt(void);
58      void io_int_cmt_cmi0(void);
59      void io_int_sci_txi1(void);
60      void io_int_sci_tei1(void);
61
62      /* ==== Global variable ==== */
63      int gCnt_TXI;  /* Number of TXI1 branches counter */
64      unsigned long *gData_DTCvect = (unsigned long *)DTC_VECT_BASE;  /* DTC vector table */
65      /* ---- Area to store the data transferred by DTC (URAM) ---- */
66      #pragma section DTCDATA
67      unsigned char gData_ADstart;  /* Data to start the A/D conversion */
68      unsigned char gData_TXIenable;/* Data to enable the TXI */
69      unsigned short gData_ADconv[4];   /* A/D conversion data (At even-numbered address) */
70      #pragma section
71
72      /*""FUNC COMMENT""*****************************************************************
73       * ID           :
74       * Outline      : Sample program main
75       *-----------------------------------------------------------------------------
76       * Include      : "dtc.h"
77       *-----------------------------------------------------------------------------
78       * Declaration  : void main(void);
79       *-----------------------------------------------------------------------------
80       * Description  : Sample program main
81       *-----------------------------------------------------------------------------
82       * Argument     : void
83       *-----------------------------------------------------------------------------
84       * Return Value : void
85       *-----------------------------------------------------------------------------
86       * Note         : None
87       *""FUNC COMMENT END""*************************************************************/
88      void main(void)
89      {
90        /* ==== Clears the number of TXI1 branches counter ==== */
91        gCnt_TXI = 0;
92
```

3.3 Sample Program Listing "main.c" (3/14)

```
93      /* ==== Sets the DTC ==== */
94      /* ---- Activated by CMI0 ---- */
95      io_init_dtc(&DTC_CMI0,      /* Address of the area storing the transfer info */
96                (unsigned long)&gData_ADstart,    /* Source: URAM */
97                (unsigned long)&ADC0.ADCR,        /* Destination: ADCR */
98                DTC_COUNT_LINK,        /* Number of times to activate DTC */
99                (DTC_TRG_CMI0 | DTC_MODE_NORMAL | DTC_SIZE_BYTE)
100               /* Activated by CMI0, normal transfer mode, byte-size transfer */
101               );
102     /* ---- Activated by ADI0 (chain transfer) ---- */
103     io_init_dtc(&DTC_ADI0_CHN1,     /* Address of the area storing the transfer info */
104               (unsigned long)&gData_TXIenable,  /* Source: URAM */
105               (unsigned long)&SCI1.SCSCR,       /* Destination: SCSCR */
106               DTC_COUNT_LINK,        /* Number of times to activate DTC */
107               (DTC_TRG_ADI0 | DTC_MODE_NORMAL | DTC_SIZE_BYTE | DTC_CHN_INITIAL)
108               /* Activated by ADI0, normal transfer mode, byte-size transfer, */
109               /* chain transfer is used (start of the chain)  */
110               );
111     io_init_dtc(&DTC_ADI0_CHN2,     /* Address of the area storing the transfer info */
112               (unsigned long)&ADC0.ADDR0,       /* Source: ADDR0 to ADDR3 */
113               (unsigned long)&gData_ADconv[0],  /* Destination: URAM */
114               (DTC_COUNT_LINK | DTC_COUNT_BLKSIZE),     /* Number of times to activate DTC, */
115                                                 /* specifies the block size */
116               (DTC_TRG_ADI0 | DTC_MODE_BLOCK | DTC_SIZE_WORD | DTC_SRC_UP | DTC_DST_UP |
117                DTC_TARGET_SRC | DTC_CHN_FINAL)
118               /* Activated by ADI0, block transfer mode, word-size transfer */
119               /* Increments both SAR and DAR after a transfer is completed, */
120               /* specifies the source as block area, chain transfer is used */
121               /* (end of the chain) */
122               );
123     /* ---- Activated by TXI1 ---- */
124     io_init_dtc(&DTC_TXI1,      /* Address of the area storing the transfer info */
125               (unsigned long)&gData_ADconv[0],  /* Source: URAM */
126               (unsigned long)&SCI1.SCTDR,       /* Destination: SCTDR */
127               DTC_COUNT_SCI,     /* Number of transfers of serial data */
128               (DTC_TRG_TXI1 | DTC_MODE_NORMAL | DTC_SIZE_BYTE | DTC_SRC_UP)
129               /* Activated by TXI1, normal transfer mode, byte-size transfer, */
130               /* Increments SAR after a transfer is completed */
131               );
```

3.4 Sample Program Listing "main.c" (4/14)

```
132
133       /* ==== Sets the CMT ==== */
134       io_init_cmt();
135
136       /* ==== Sets the ADC ==== */
137       io_init_adc();
138
139       /* ==== Sets the SCI ==== */
140       io_init_sci();
141
142       /* ==== Starts the CMT counting ==== */
143       io_start_timer_cmt();
144
145       while(1){
146         /* Dead loop */
147       }
148     }
149
```

3.5 Sample Program Listing "main.c" (5/14)

```
150    /*""FUNC COMMENT""*****************************************************************
151     * ID           :
152     * Outline      : ADC configuration
153     *-------------------------------------------------------------------------------
154     * Include      : "iodefine.h"
155     *-------------------------------------------------------------------------------
156     * Declaration  : void io_init_adc(void);
157     *-------------------------------------------------------------------------------
158     * Description  : Configures ADC0.
159     *              : - Operating mode: Single-cycle scan mode
160     *              : - Analog input channels: AN0, AN1, AN2, AN3
161     *              : - Conversion circuit: Sample-and-hold circuit
162     *              :
163     *              : As the Data Transfer Controller transfers the setting data to
164     *              : the ADCR (to start A/D conversion and enable the ADI request),
165     *              : this function stores the setting data at source to URAM.
166     *-------------------------------------------------------------------------------
167     * Argument     : void
168     *-------------------------------------------------------------------------------
169     * Return Value : void
170     *-------------------------------------------------------------------------------
171     * Note         : None
172     *""FUNC COMMENT END""*************************************************************/
173    void io_init_adc(void)
174    {
175      /* ==== Clears the module standby ==== */
176      STB.CR3.BIT._ADC0 = 0;          /* Supply the clock to ADC0 */
177
178      /* ==== Sets the A/D analog input channel select register (ADANSR) ==== */
179      ADC0.ADANSR.BYTE = 0x0f;    /* Selects analog input channels AN0 to AN3 */
180
181      /* ==== Sets the A/D bypass control register (ADBYPSCR) ==== */
182      ADC0.ADBYPSCR.BIT.SH = 1;       /* Selects sample-and-hold circuit */
183
184      /* ==== Stores the setting data in URAM to transfer to the A/D control register */
185      /* (ADCR) ==== */
186      gData_ADstart = 0x90;
187        /*
188            bit 7: ADST = 1 ---- Starts A/D conversion
189            bit 6: ADCS = 0 ---- Single-cycle scan
190            bit 5: ACE = 0 ----- Disables to auto-clear the ADDR by reading the ADDR
191            bit 4: ADIE = 1 ---- Enables the A/D conversion end interrupt (ADI)
192            bits 3, 2: Reserved (0)
193            bit 1: TRGE = 0 ---- Disables the A/D conversion by the external trigger or
194                 :               the A/D converter start trigger from the MTU2/MTU2S
195            bit 0: EXTRG = 0 --- Activates the A/D Converter by the A/D converter
196                 :               start trigger from the MTU2/MTU2S
197        */
198    }
199
```

3.6 Sample Program Listing "main.c" (6/14)

```
200    /*""FUNC COMMENT""*****************************************************************
201     * ID           :
202     * Outline      : DTC configuration
203     *-------------------------------------------------------------------------------
204     * Include      : "iodefine.h" and "dtc.h"
205     *-------------------------------------------------------------------------------
206     * Declaration  : void io_init_dtc(DTC_TRANS_INFO *dtc_info,
207     *              :                  unsigned long src,
208     *              :                  unsigned long dst,
209     *              :                  unsigned long count,
210     *              :                  unsigned long mode);
211     *-------------------------------------------------------------------------------
212     * Description  : Configures the DTC.
213     *              : Transfers the transfer information specified in arguments src,
214     *              : dst, count, and mode to the DTC transfer information registers
215     *              : (MRA, MRB, SAR, DAR, CRA, and CRB) specified in the argument
216     *              : *dtc_info.  When not using the chain transfer, or chain
217     *              : transfer is used and the start data transfer information in
218     *              : the chain is used, register the transfer information area
219     *              : address specified in the *dtc_info to the DTC vector table of
220     *              : the interrupt to activate the DTC.
221     *-------------------------------------------------------------------------------
222     * Argument     : - DTC_TRANS_INFO *dtc_info ; I/O : Address to the DTC transfer
223     *              :                                  : information register area
224     *              : - unsigned long src        ; I   : Address to set to the SAR
225     *              : - unsigned long dst        ; I   : Address to set to the DAR
226     *              : - unsigned long count      ; I   : Number of transfers
227     *              : - unsigned long mode       ; I   :
228     *              :    Specify the information to set the DTC (such as transfer
229     *              :    mode, transfer size, and activation source trigger) by a
230     *              :    parameter which is macro-defined in the sample program
231     *              :    header file "dtc.h". When specifying multiple parameters,
232     *              :    delimit the parameters by OR operators "|".
233     *-------------------------------------------------------------------------------
234     * Return Value : void
235     *-------------------------------------------------------------------------------
236     * Note         : - When executing this function, disable the interrupt from the
237     *              :   module to activate the DTC.
238     *              : - Align the transfer information register address specified in
239     *              :   the *dtc_info on 4-byte boundary.
240     *              : - To use block transfer mode, specify the block size in bits
241     *              :   23 to 16 in the argument count, and number of transfers in
242     *              :   lower 16 bits (bits 15 to 0) in count.
243     *""FUNC COMMENT END""*************************************************************/
244    void io_init_dtc(DTC_TRANS_INFO *dtc_info, unsigned long src,
245                unsigned long dst, unsigned long count, unsigned long mode)
246    {
247      unsigned long chain_info;
248      unsigned char set_mra, set_mrb, set_dtccr, f_rrs;
249
```

3.7 Sample Program Listing "main.c" (7/14)

```
250     /* ---- Retrieves the information to set the DTC from the fifth argument ---- */
251     set_mra = (unsigned char)((mode >> 8) & 0x000000fc);   /* Info to set the MRA */
252     set_mrb = (unsigned char)(mode & 0x000000fc);  /* Info to set the MRB */
253     set_dtccr = (unsigned char)((mode >> 24) & 0x00000018);   /* Info to set the DTCCR */
254     chain_info = mode & 0x00ffc0c3;     /* Chain transfer info */
255
256     /* **** Sets the transfer information **** */
257     /* ==== Sets the DTC control register (DTCCR) ==== */
258     /* ---- Saves the transfer information read skip enable information */
259     /*      (to use the chain transfer) ---- */
260     f_rrs = DTC.DTCCR.BIT.RRS;
261     /* ---- Disables to skip reading the transfer information ---- */
262     DTC.DTCCR.BIT.RRS = 0;
263     /* ==== Sets DTC mode register A (MRA) ==== */
264     dtc_info->MRA = set_mra;
265     /* ==== Sets DTC mode register B (MRB) ==== */
266     dtc_info->MRB = set_mrb;
267     /* ==== Sets the DTC source address register (SAR) ==== */
268     dtc_info->SAR = src;
269     /* ==== Sets the DTC destination address register (DAR) ==== */
270     dtc_info->DAR = dst;
271     /* ==== Sets DTC transfer count registers A, B (CRA, CRB) ==== */
272     if((chain_info & DTC_MODE_REPEAT) != 0){
273       /* ---- In repeat transfer mode ---- */
274       dtc_info->CRA.BYTE.H = (unsigned char)(count & 0x000000ff); /* Number of transfers */
275       dtc_info->CRA.BYTE.L = dtc_info->CRA.BYTE.H; /* Transfer counter */
276     }
277     else if((chain_info & DTC_MODE_BLOCK) != 0){
278       /* ---- In block transfer mode ---- */
279       dtc_info->CRA.BYTE.H = (unsigned char)((count & 0x00ff0000) >> 16); /* Block size */
280       dtc_info->CRA.BYTE.L = dtc_info->CRA.BYTE.H; /* Block size counter */
281       dtc_info->CRB = (unsigned short)(count & 0x0000ffff);    /* Number of transfers */
282     }
283     else{
284       /* ---- In normal transfer mode ---- */
285       dtc_info->CRA.WORD = (unsigned short)(count & 0x0000ffff);   /* Number of transfers */
286     }
287
288     if(((chain_info & 0x00000083) == DTC_CHN_DISABLE) ||
289       ((chain_info & 0x00000083) == DTC_CHN_INITIAL)){
290       /* ---- When NOT using the chain transfer or using the chain transfer and
291               setting the start data transfer information in the chain ---- */
292
```

3.8 Sample Program Listing "main.c" (8/14)

```
293         /* **** Sets the DTC control register (DTCCR) **** */
294         DTC.DTCCR.BYTE = set_dtccr;
295             /*
296                 bits 7 to 5: Reserved (0).
297                 bit 4: RRS = x ----- Enables/disables to skip reading the transfer information
298                      :               (Disabled by default)
299                 bit 3: RCHNE = x --- Enables/disables the chain transfer after the
300                       :               repeat transfer (Disabled by default).
301                 bits 2, 1: Reserved (0)
302                 bit 0: ERR = 0 ----- Transfer stop flag
303             */
304
305         /* **** Sets the DTC vector base register (DTCVBR) **** */
306         DTC.DTCVBR = DTC_VECT_BASE;
307
308         /* **** Sets the DTC vector address/source to activate the DTC **** */
309         switch(chain_info & 0x00ff0000){
310         case 0x00010000:   /* Sets the DTC vector address: CMI0 */
311             gData_DTCvect[DTC_VECT_CMI0/sizeof(unsigned long)] = (unsigned long)dtc_info;
312             DTC.DTCERA.BIT.CMI0 = 1;   /* Enables to activate the DTC: CMI0 */
313             break;
314         case 0x00020000:   /* Sets the DTC vector address: ADI0 */
315             gData_DTCvect[DTC_VECT_ADI0/sizeof(unsigned long)] = (unsigned long)dtc_info;
316             DTC.DTCERA.BIT.ADI0 = 1;   /* Enables to activate the DTC: ADI0 */
317             break;
318         case 0x00030000:   /* Sets the DTC vector address: TXI1 */
319             gData_DTCvect[DTC_VECT_TXI1/sizeof(unsigned long)] = (unsigned long)dtc_info;
320             DTC.DTCERE.BIT.TXI1 = 1;   /* Enables to activate the DTC: TXI1 */
321             break;
322         default:
323             break;
324         }
325       }
326       else{
327         /* ---- When using the chain transfer and setting the middle or end data transfer
328                 information in the chain ---- */
329         DTC.DTCCR.BIT.RRS = f_rrs; /* Returns the transfer information read skip information */
330       }
331     }
332
```

3.9 Sample Program Listing "main.c" (9/14)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : SCI configuration
 *-----------------------------------------------------------------------------
 * Include      : "iodefine.h"
 *-----------------------------------------------------------------------------
 * Declaration  : void io_init_sci(void);
 *-----------------------------------------------------------------------------
 * Description  : Configures SCI1
 *              :  - Asynchronous mode
 *              :  - Character length: 8-bit data
 *              :  - No parity
 *              :  - Stop bit length: 1 stop bit
 *              :  - Bit rate: 57600 bps
 *              :
 *              : PA4 pin function is set to TXD1 (output).
 *              :
 *              : As the Data Transfer Controller transfers the setting data to
 *              : the SCSCR (to enable the TXI request and transmitter), this
 *              : function stores the setting data at source to URAM.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void io_init_sci(void)
{
  volatile int CntWait_1bit = 600;    /* Counter to wait for 1-bit interval @ 200 MHz */

  /* ==== Clears the module standby ==== */
  STB.CR5.BIT._SCI1 = 0;          /* Supplies the clock to SCI1 */

  /* ==== Sets SCI1 interrupt priority level ==== */
  INTC.IPR16.BIT._SCI1 = 0xf;         /* Level = 15 */

  /* ==== Sets the Serial control register (SCSCR) ==== */
  /* ---- Disables the interrupt request, transmission and reception ---- */
  SCI1.SCSCR.BYTE = 0x00;
  /* ---- Sets the clock source/clock I/O ---- */
  SCI1.SCSCR.BIT.CKE = 0;     /* Sets the internal clock/SCK pin as input */
                           /* (input signal is ignored) */

```

3.10 Sample Program Listing "main.c" (10/14)

```
377     /* ==== Sets the Serial mode register (SCSMR) ==== */
378     SCI1.SCSMR.BYTE = 0x00;
379       /*
380           bit 7: C/A# = 0 ---------- Asynchronous mode
381           bit 6: CHR = 0 ----------- Character length: 8-bit data
382           bit 5: PE = 0 ------------ Disables to add and check parity
383           bit 4: O/E# = 0 ---------- Not used
384           bit 3: STOP = 0 ---------- Stop bit length: 1 stop bit
385           bit 2: MP = 0 ------------ Disables multiprocessor mode
386           bits 1, 0: CKS[1:0] = B'00 --- Baud rate generator operating clock: P clock
387       */
388
389     /* ==== Sets the Bit rate register (SCBRR) ==== */
390     SCI1.SCBRR = 0x1a;              /* Bit rate setting is 57870.37 bps @ P clock is */
391                                /* 50 MHz (Assumed bit rate is 57600 bps) */
392
393     /* ==== Wait for 1-bit interval ==== */
394     while(CntWait_1bit-- > 0){
395     }
396
397     /* ==== Sets the pin function ==== */
398     PFC.PACRL2.BIT.PA4MD = 6;       /* Sets PA4 pin function to TXD1 (output) */
399
400     /* ==== Transfers the data value in URAM to set to the SCSCR ==== */
401     gData_TXIenable = 0xa0;
402       /*
403           bit 7: TIE = 1 ----------- Enables the TXI request
404           bit 6: RIE = 0 ----------- Disables RXI, ERI requests
405           bit 5: TE = 1 ------------ Enables the transmission
406           bit 4: RE = 0 ------------ Disables the reception
407           bit 3: MPIE = 0 ---------- Disables multiprocessor mode
408           bit 2: TEIE = 0 ---------- Disables the TEI request
409           bits 1, 0: CKE[1:0] = B'00 --- Sets internal clock/SCK pin as input
410                 :                     (Input signal is ignored)
411       */
412   }
413
```

3.11 Sample Program Listing "main.c" (11/14)

```
414    /*""FUNC COMMENT""*****************************************************************
415     * ID           :
416     * Outline      : CMT setting
417     *-------------------------------------------------------------------------------
418     * Include      : "iodefine.h"
419     *-------------------------------------------------------------------------------
420     * Declaration  : void io_init_cmt(void);
421     *-------------------------------------------------------------------------------
422     * Description  : Sets CMT0 as a 5-msec fixed-cycle timer.
423     *-------------------------------------------------------------------------------
424     * Argument     : void
425     *-------------------------------------------------------------------------------
426     * Return Value : void
427     *-------------------------------------------------------------------------------
428     * Note         : None
429     *""FUNC COMMENT END""*************************************************************/
430    void io_init_cmt(void)
431    {
432      /* ==== Clears the module standby ==== */
433      STB.CR4.BIT._CMT = 0;       /* Supplies the clock to the CMT */
434      /* ==== Stops the CMT counting ==== */
435      io_stop_timer_cmt();
436      /* ==== Sets CMT0 interrupt priority level ==== */
437      INTC.IPR08.BIT._CMT0 = 0xf;     /* Level = 15 */
438      /* ==== Sets the CMT control/status register (CMCSR) ==== */
439      CMT0.CMCSR.WORD = 0x0040;   /* Enables the CMI request count clock: P clock/8 */
440      /* ==== Clears the CMT counter ==== */
441      CMT0.CMCNT = 0x0000;
442      /* ==== Sets the compare match interval ==== */
443      CMT0.CMCOR = 31250 - 1;     /* 5-msec @ P clock/8 (P clock = 50 MHz) */
444    }
445
446    /*""FUNC COMMENT""*****************************************************************
447     * ID           :
448     * Outline      : CMT counting start
449     *-------------------------------------------------------------------------------
450     * Include      : "iodefine.h"
451     *-------------------------------------------------------------------------------
452     * Declaration  : void io_start_timer_cmt(void);
453     *-------------------------------------------------------------------------------
454     * Description  : Starts the CMT counting.
455     *-------------------------------------------------------------------------------
456     * Argument     : void
457     *-------------------------------------------------------------------------------
458     * Return Value : void
459     *-------------------------------------------------------------------------------
460     * Note         : None
461     *""FUNC COMMENT END""*************************************************************/
462    void io_start_timer_cmt(void)
463    {
464      CMT.CMSTR.BIT.STR0 = 1;     /* Starts CMCNT_0 counting */
465    }
```

3.12 Sample Program Listing "main.c" (12/14)

```
466
467    /*""FUNC COMMENT""*****************************************************************
468     * ID           :
469     * Outline      : CMT counting stop
470     *-------------------------------------------------------------------------------
471     * Include      : "iodefine.h"
472     *-------------------------------------------------------------------------------
473     * Declaration  : void io_stop_timer_cmt(void);
474     *-------------------------------------------------------------------------------
475     * Description  : Stops the CMT counting.
476     *-------------------------------------------------------------------------------
477     * Argument     : void
478     *-------------------------------------------------------------------------------
479     * Return Value : void
480     *-------------------------------------------------------------------------------
481     * Note         : None
482     *""FUNC COMMENT END""*************************************************************/
483    void io_stop_timer_cmt(void)
484    {
485      CMT.CMSTR.BIT.STR0 = 0;     /* Stops CMCNT_0 counting */
486    }
487
488    /*""FUNC COMMENT""*****************************************************************
489     * ID           :
490     * Outline      : CMT compare match interrupt (CMI0).
491     *-------------------------------------------------------------------------------
492     * Include      : "iodefine.h"
493     *-------------------------------------------------------------------------------
494     * Declaration  : void io_int_cmt_cmi0(void);
495     *-------------------------------------------------------------------------------
496     * Description  : Clears the compare match flag, and stops the CMT counting.
497     *-------------------------------------------------------------------------------
498     * Argument     : void
499     *-------------------------------------------------------------------------------
500     * Return Value : void
501     *-------------------------------------------------------------------------------
502     * Note         : None
503     *""FUNC COMMENT END""*************************************************************/
504    void io_int_cmt_cmi0(void)
505    {
506      unsigned short dummy;
507
508      /* ==== Clears the compare match flag ==== */
509      CMT0.CMCSR.BIT.CMF &= 0;
510      /* ==== Stops the CMT counting ==== */
511      io_stop_timer_cmt();
512
513      dummy = CMT0.CMCSR.WORD;/* dummy read */
514    }
515
```

3.13 Sample Program Listing "main.c" (13/14)

```
516    /*""FUNC COMMENT""*****************************************************************
517     * ID           :
518     * Outline      : SCI transmit data empty interrupt (TXI1).
519     *-----------------------------------------------------------------------------
520     * Include      : "iodefine.h"
521     *-----------------------------------------------------------------------------
522     * Declaration  : void io_int_sci_txi1(void);
523     *-----------------------------------------------------------------------------
524     * Description  : After disabling TXI1 request, clearing the TDRE flag, executes
525     *              : the following processing depending on the number of branches
526     *              : to TXI1:
527     *              :
528     *              : When the number of branches to TXI1 is less than the number of
529     *              : times of activating the DTC when it is configured, this
530     *              : function initializes the following transfer information and
531     *              : enables TXI1 as the activation source.
532     *              :   - Transfer destination address (DAR) for block transfer on ADI0
533     *              :   - Transfer source address (SAR) and number of transfers (CRA)
534     *              :
535     *              : When the number of branches to TXI1 reaches the number of
536     *              : times of activating the DTC when it is configured, this
537     *              : function clears the TEND flag (read 1 from the TDRE bit and
538     *              : write 0), and enables the SCI transmit end interrupt (TEI1).
539     *-----------------------------------------------------------------------------
540     * Argument     : void
541     *-----------------------------------------------------------------------------
542     * Return Value : void
543     *-----------------------------------------------------------------------------
544     * Note         : None
545     *""FUNC COMMENT END""*************************************************************/
546    void io_int_sci_txi1(void)
547    {
548      /* ==== Increments the number of TXI1 branches counter ==== */
549      gCnt_TXI++;
550
551      /* ==== Disables TXI1 request ==== */
552      SCI1.SCSCR.BIT.TIE = 0;
553
```

3.14 Sample Program Listing "main.c" (14/14)

```
554       /* ==== Clears the TDRE flag ==== */
555       SCI1.SCSSR.BIT.TDRE = 0;
556
557       if(gCnt_TXI < DTC_COUNT_LINK){
558         /* ==== Sets the DTC transfer information/activation source again ==== */
559         /* ---- Activation source ADI0 ---- */
560         DTC_ADI0_CHN2.DAR = (unsigned long)&gData_ADconv[0]; /* DAR for block transfer */
561         /* ---- Activation source TXI1 ---- */
562         DTC_TXI1.SAR = (unsigned long)&gData_ADconv[0];      /* SAR */
563         DTC_TXI1.CRA.WORD = DTC_COUNT_SCI;    /* Number of serial data transfers */
564         DTC.DTCERE.BIT.TXI1 = 1;   /* Enables to activate the DTC: TXI1 */
565       }
566       else{
567         /* ==== Clears the number of TXI1 branches counter ==== */
568         gCnt_TXI = 0;
569         /* ==== Clears the TEND flag ==== */
570         SCI1.SCSSR.BIT.TDRE = 0;   /* Reads 1 from the TDRE bit and writes 0 */
571         /* ==== Enables TEI1 request ==== */
572         SCI1.SCSCR.BIT.TEIE = 1;
573       }
574     }
575
576     /*""FUNC COMMENT""*****************************************************************
577      * ID           :
578      * Outline      : SCI transmit end interrupt (TEI1).
579      *-----------------------------------------------------------------------------
580      * Include      : "iodefine.h"
581      *-----------------------------------------------------------------------------
582      * Declaration  : void io_int_sci_tei1(void);
583      *-----------------------------------------------------------------------------
584      * Description  : Disables TEI1 request and transmission.
585      *-----------------------------------------------------------------------------
586      * Argument     : void
587      *-----------------------------------------------------------------------------
588      * Return Value : void
589      *-----------------------------------------------------------------------------
590      * Note         : None
591      *""FUNC COMMENT END""*************************************************************/
592     void io_int_sci_tei1(void)
593     {
594       unsigned char dummy;
595
596       /* ==== Disables TEI1 request ==== */
597       SCI1.SCSCR.BIT.TEIE = 0;
598       /* ==== Disables the transmission ==== */
599       SCI1.SCSCR.BIT.TE = 0;
600
601       dummy = SCI1.SCSCR.BYTE;/* dummy read */
602     }
603
604     /* End of File */
```

3.15 Sample Program Listing "intprg.c" (1/2)

```
1   /***********************************************************************
2   *   DISCLAIMER
3   *
4   *   This software is supplied by Renesas Electronics Corporation and is only
5   *   intended for use with Renesas products.  No other uses are authorized.
6   *
7   *   This software is owned by Renesas Electronics Corporation and is protected under
8   *   all applicable laws, including copyright laws.
9   *
10  *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11  *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12  *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13  *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14  *   DISCLAIMED.
15  *
16  *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17  *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18  *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19  *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20  *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21  *
22  *   Renesas reserves the right, without notice, to make changes to this
23  *   software and to discontinue the availability of this software.
24  *   By using this software, you agree to the additional terms and
25  *   conditions found by accessing the following link:
26  *   http://www.renesas.com/disclaimer
27  ***********************************************************************
28  *   Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
29  ***********************************************************************/
30  /*""FILE COMMENT""************ Technical reference data ************************
31  *   System Name : SH7216 Sample Program
32  *   File Name   : intprg.c
33  *   Abstract    : Interrupt Functions
34  *   Version     : 1.00.00
35  *   Device      : SH7216
36  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37  *               : C/C++ compiler package for the SuperH RISC engine family
38  *               :                             (Ver.9.03 Release00).
39  *   OS          : None
40  *   H/W Platform: R0K572167 (CPU board)
41  *   Description :
42  ***********************************************************************
43  *   History     : Nov.01,2010 Ver.1.00.00
44  *""FILE COMMENT END""*********************************************************/
```

3.16 Sample Program Listing "intprg.c" (2/2)

```
45      #include <machine.h>
46      #include "vect.h"
47
48      extern void io_int_cmt_cmi0(void);
49      extern void io_int_txi1(void);
50      extern void io_int_tei1(void);
51
52      #pragma section IntPRG
53
54      // 4 Illegal code
55      void INT_Illegal_code(void){/* sleep(); */}
56      // 5 Reserved
57
58      // 6 Illegal slot
59      void INT_Illegal_slot(void){/* sleep(); */}

…       (omitted)

326     // 140 CMT CMI0
327     void INT_CMT_CMI0(void)
328     {
329       /* ==== CMT compare match interrupt (CMI0) ==== */
330       io_int_cmt_cmi0();
331     }

…       (omitted)

542     // 246 SCI SCI1 TXI1
543     void INT_SCI_SCI1_TXI1(void)
544     {
545       /* ==== SCI transmit data empty interrupt (TXI1) ==== */
546       io_int_sci_txi1();
547     }
548     // 247 SCI SCI1 TEI1
549     void INT_SCI_SCI1_TEI1(void)
550     {
551       /* ==== SCI transmit end interrupt (TEI1) ==== */
552       io_int_sci_tei1();
553     }

…       (omitted)

566     // 254 SCIF SCIF3 RXI3
567     void INT_SCIF_SCIF3_RXI3(void){/* sleep(); */}
568     // 255 SCIF SCIF3 TXI3
569     void INT_SCIF_SCIF3_TXI3(void){/* sleep(); */}
570     // Dummy
571     void Dummy(void){/* sleep(); */}
572
573     /* End of File */
```

3.17 Sample Program Listing "dtc.h" (1/3)

```
1      /***********************************************************************
2      *   DISCLAIMER
3      *
4      *   This software is supplied by Renesas Electronics Corporation and is only
5      *   intended for use with Renesas products.  No other uses are authorized.
6      *
7      *   This software is owned by Renesas Electronics Corporation and is protected under
8      *   all applicable laws, including copyright laws.
9      *
10     *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11     *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12     *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13     *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14     *   DISCLAIMED.
15     *
16     *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17     *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18     *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19     *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20     *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21     *
22     *   Renesas reserves the right, without notice, to make changes to this
23     *   software and to discontinue the availability of this software.
24     *   By using this software, you agree to the additional terms and
25     *   conditions found by accessing the following link:
26     *   http://www.renesas.com/disclaimer
27     ***********************************************************************
28     *   Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
29     ***********************************************************************/
30     /*""FILE COMMENT""************ Technical reference data ************************
31     *   System Name : SH7216 Sample Program
32     *   File Name   : dtc.h
33     *   Abstract    : Macro definition for DTC setting
34     *   Version     : 1.00.00
35     *   Device      : SH7216
36     *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37     *               : C/C++ compiler package for the SuperH RISC engine family
38     *               :                             (Ver.9.03 Release00).
39     *   OS          : None
40     *   H/W Platform: R0K572167 (CPU board)
41     *   Description :
42     ***********************************************************************
43     *   History     : Nov.01,2010 Ver.1.00.00
44     *""FILE COMMENT END""*********************************************************/
45     #ifndef _DTC_H_
46     #define _DTC_H_
47
```

3.18 Sample Program Listing "dtc.h" (2/3)

```
48     /* **** Structure definition **** */
49     /* ==== Register definition for DTC transfer information ==== */
50     typedef struct {
51       unsigned char MRA;              /* DTC mode register A                  */
52       unsigned char MRB;              /* DTC mode register B                  */
53       unsigned char dummy1;           /* Reserved                             */
54       unsigned char dummy2;           /* Reserved                             */
55       unsigned long SAR;              /* DTC source address register          */
56       unsigned long DAR;              /* DTC destination address register     */
57       union{
58         unsigned short WORD;
59         struct {
60             unsigned char H;
61             unsigned char L;
62         } BYTE;
63       } CRA;                          /* DTC transfer count register A             */
64       unsigned short CRB;             /* DTC transfer count register B             */
65     } DTC_TRANS_INFO;
66
67
68     /* **** Macro definition **** */
69     /* ==== Transfer information storage area ==== */
70     #define DTC_CMI0   (*(volatile DTC_TRANS_INFO *)0xfff84000)  /* CMI0 activation */
71     #define DTC_ADI0_CHN1  (*(volatile DTC_TRANS_INFO *)0xfff84010)  /* ADI0 activation 1 */
72     #define DTC_ADI0_CHN2  (*(volatile DTC_TRANS_INFO *)0xfff84020)  /* ADI0 activation 2 */
73     #define DTC_TXI1   (*(volatile DTC_TRANS_INFO *)0xfff84030)  /* TXI1 activation */
74
75     /* ==== Parameters to set the DTC vector ==== */
76     /* ---- Base address for the DTC vector table ---- */
77     #define DTC_VECT_BASE  0xfff85000
78     /* ---- DTC activation source vector address offset ---- */
79     #define DTC_VECT_ADI0  0x0570  /* ADI0 */
80     #define DTC_VECT_CMI0  0x0630  /* CMI0 */
81     #define DTC_VECT_TXI1  0x07d8  /* TXI1 */
82
83     /* ==== Parameters to set the fourth argument (number of transfers) of */
84     /*      the io_init_dtc function ==== */
85     #define DTC_COUNT_LINK 10      /* Number of times to activate DTC */
86     #define DTC_COUNT_BLKSIZE  0x00040000 /* Block size in block transfer mode */
87                                       /* (bits23 to 16) */
88     #define DTC_COUNT_SCI  8       /* Number of transfers of serial data */
89
```

3.19 Sample Program Listing "dtc.h" (3/3)

```
90    /* ==== Parameters to set the fifth argument (mode) of the io_init_dtc function ==== */
91    /* ---- Activation source trigger ---- */
92    #define DTC_TRG_CMI0   0x00010000 /* CMI0 */
93    #define DTC_TRG_ADI0   0x00020000 /* ADI0 */
94    #define DTC_TRG_TXI1   0x00030000 /* TXI1 */
95    /* ---- Transfer mode ---- */
96    #define DTC_MODE_NORMAL    0x00000000 /* Normal transfer mode */
97    #define DTC_MODE_REPEAT    0x00004000 /* Repeat transfer mode */
98    #define DTC_MODE_BLOCK 0x00008000 /* Block transfer mode */
99    /* ---- Transfer size ---- */
100   #define DTC_SIZE_BYTE  0x00000000 /* Byte-size transfer */
101   #define DTC_SIZE_WORD  0x00001000 /* Word-size transfer */
102   #define DTC_SIZE_LONG  0x00002000 /* Longword-size transfer */
103   /* ---- Source/destination address mode ---- */
104   #define DTC_SRC_FIX        0x00000000 /* SAR is fixed after a transfer is completed */
105   #define DTC_SRC_UP     0x00000800 /* Increments SAR after a transfer is completed */
106   #define DTC_SRC_DOWN   0x00000c00 /* Decrements SAR after a transfer is completed */
107   #define DTC_DST_FIX        0x00000000 /* DAR is fixed after a transfer is completed */
108   #define DTC_DST_UP     0x00000008 /* Increments DAR after a transfer is completed */
109   #define DTC_DST_DOWN   0x0000000c /* Decrements DAR after a transfer is completed */
110   /* ---- Chain transfers ---- */
111   #define DTC_CHN_DISABLE    0x00000000 /* Disables the chain transfer */
112   #define DTC_CHN_INITIAL    0x00000080 /* Chain transfer is used (start of the chain) */
113   #define DTC_CHN_MIDDLE 0x00000081 /* Chain transfer is used (middle of the chain) */
114   #define DTC_CHN_FINAL  0x00000002 /* Chain transfer is used (end of the chain) */
115   #define DTC_CHN_CONTINUE   0x00000000 /* Repeats chain transfers */
116   #define DTC_CHN_COUNTER0   0x00000040 /* Chain transfer is used when the */
117                                     /* transfer counter is 0 */
118   /* ---- Timing  to generate the CPU interrupt ---- */
119   #define DTC_INT_TRANSTIME  0x00000000 /* The CPU interrupt occurs when the specified */
120                                     /* number of transfers is completed */
121   #define DTC_INT_EVERYTIME  0x00000020 /* The CPU interrupt occurs every time a transfer */
122                                     /* is completed */
123   /* ---- Target area in repeat/block transfer mode ---- */
124   #define DTC_TARGET_DST 0x00000000 /* Specifies the destination as repeat or block area */
125   #define DTC_TARGET_SRC 0x00000010 /* Specifies the source as repeat or block area */
126   /* ---- Set the DTCCR ---- */
127   #define DTC_CR_RRS_DISABLE 0x00000000 /* Disables to skip reading the transfer information */
128   #define DTC_CR_RRS_ENABLE  0x10000000 /* Enables to skip reading the transfer information */
129   #define DTC_CR_RCHNE_DISABLE   0x00000000 /* Disables the chain transfer after */
130                                         /* the repeat transfer */
131   #define DTC_CR_RCHNE_ENABLE    0x08000000 /* Enables the chain transfer after the */
132                                     /* repeat transfer */
133
134
135   #endif  /* _DTC_H_ */
136
137   /* End of File */
```

4. References

- Software Manual
  SH-2A/SH2A-FPU Software Manual Rev.3.00
  The latest version of the software manual can be downloaded from the Renesas Electronics website.

- Hardware Manual
  SH7214 Group, SH7216 Group Hardware Manual Rev.2.00
  The latest version of the hardware manual can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Nov.01.10	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

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.

RENESAS

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

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

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

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

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

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

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

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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

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

Colophon 1.0