APPLICATION NOTE

RX62N Group, RX621 Group

I^{2}S Communication Using RSPI, DTCa, and MTU2

R01AN0631EJ0100
Rev.1.00
Sep 27, 2011

Introduction

This application note describes a method for transferring audio data by I^{2}S communication, using the serial peripheral interface (RSPI), data transfer controller (DTCa), and multi-function timer pulse unit 2 (MTU2) of a RX62N Group or RX621 Group microcontroller (MCU).

Target Devices

RX62N Group and RX621 Group MCUs

In order to use the sample code with another MCU it must be modified to match the specifications of the target device and its operation tested and evaluated thoroughly.

Contents

1. Specifications

The sample program uses the RSPI, DTCa, and MTU2 to transmit and receive audio data by I^{2}S communication. The RSPI performs serial transfer of audio data according to the clock signal generated by the MTU2.

Table 1.1 lists the peripheral functions used by the sample program and their applications, and figure 1.1 shows a block diagram.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
RSPI channel 0	Audio data (left channel) input/output
RSPI channel 1	Audio data (right channel) input/output
MTU2 channel 2	Generation of serial transfer clock (SCK)
MTU2 channel 3	Generation of word select signal (WS)
MTU2 channel 4	Generation of slave select signal (SSL) to RSPI
DTCa	Transfer of audio data to and from on-chip RAM

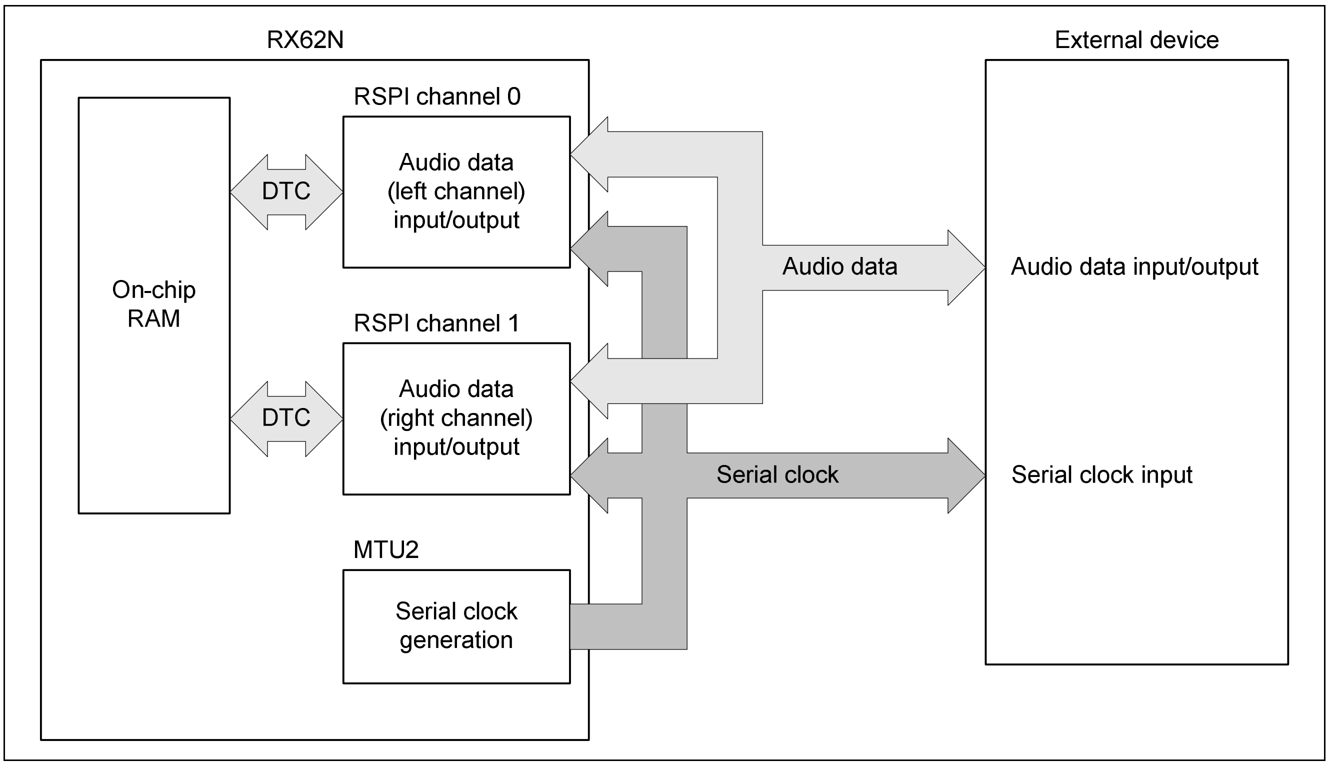


Figure 1.1 Block Diagram

2. Operation Confirmation Conditions

The conditions used for confirming the operation of the sample code included in the application example are listed below.

Table 2.1 Operation Confirmation Conditions

Item	Description
MCU	RX62N Group
Device	HD74LV32A (Renesas Electronics)
Operating frequencies	• Input clock: 12 MHz • CPU clock: 96 MHz • Peripheral module clock: 48 MHz
Power supply voltage	3.3 V
Integrated development environment	Renesas Electronics High-performance Embedded Workshop Version 4.08.00.011
C compiler	Renesas Electronics C/C++ compiler package for RX family V.1.00 Release 01
	Compile option -cpu=rx600 -lang=c -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" debug –nologo (The default settings of the integrated development environment are used.)
Operating mode	Single-chip mode
Sample code version	Version 1.00
Board	Renesas Starter Kit (R0K5562N0S000BE)

3. Hardware

3.1 List of Pins

Table 3.1 lists the pins used by the sample program.

Table 3.1 List of Pins

Pin	Input/Output	Description
P24_MTCLKA-A	Input	External clock (12.288 MHz) input
P26_MTIOC2A	Output	Serial transfer clock (SCK) output
P17_MTIOC3A	Output	Word select signal (WS) output
P82_MTIOC4A-B	Output	Slave select signal (SSL) output
PC4_SSLA0-A	Input	RSPI0 slave select signal input
PC5_RSPCKA-A	Input	RSPI0 serial transfer clock input
PC6_MOSIA-A	Input	Audio data (left channel) input
PC7_MISOA-A	Output	Audio data (left channel) output*[1]
PE4_SSLB0-B	Input	RSPI1 slave select signal input
PE5_RSPCKB-B	Input	RSPI1 serial transfer clock input
PE6_MOSIB-B	Input	Audio data (right channel) input
PE7_MISOB-B	Output	Audio data (right channel) output*[1]

Note: 1. This pin is not used during receive-only operation and is set to high-impedance.

3.2 Connections to Peripheral Devices

Figure 3.1 shows a connection diagram. Table 3.2 lists the clock signals generated by the MTU2.

The MTU2 uses the external clock (12.288 MHz) as the count clock and generates the clock signals listed in table 3.2.

RSPI channel 0 (RSPI0) is used to transmit and receive the left channel (L-ch) portion of the audio data and RSPI channel 1 (RSPI1) the right channel (R-ch) portion. The RSPI operates in slave mode and switches to the active channel according to the SSL signal.

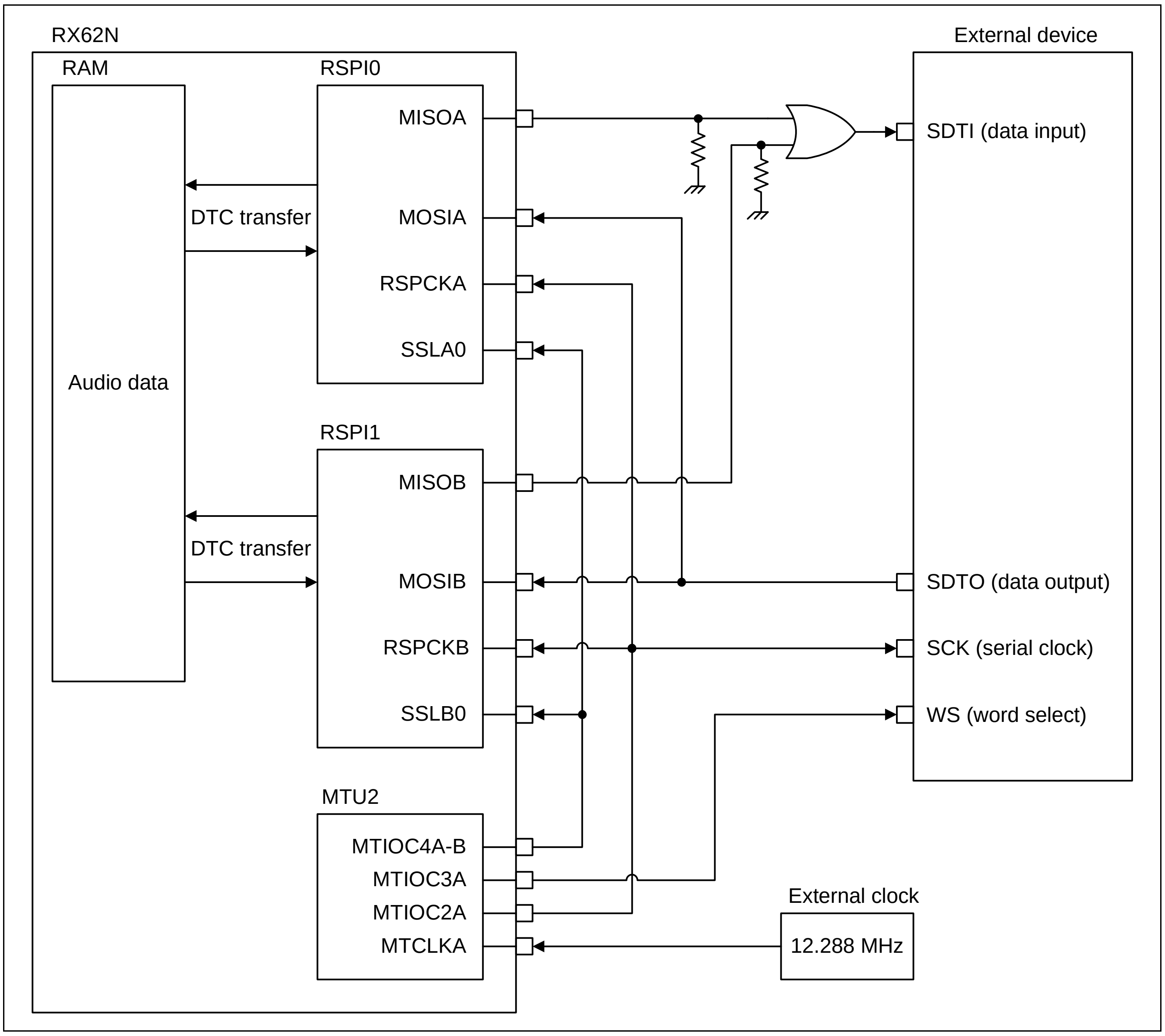


Figure 3.1 Connection Diagram

Table 3.2 Clock Signals Generated by MTU2

Channel	Clock Signal	Symbol	Output Pin	Frequency
Channel 2	Serial transfer clock	SCK	MTIOC2A	3.072 MHz
Channel 3	Word select signal	WS	MTIOC3A	48 kHz
Channel 4	RSPI slave select signal	SSL	MTIOC4A-B	48 kHz

4. Software

4.1 Operation Overview

4.1.1 Implementation of I^2S Communication by Using RSPI, DTCa, and MTU2

The method of implementing I^2S communication by using the RSPI, DTCa, and MTU2 is described below.

(1) Audio Data Format

The sample program handles audio data in 32-bit units, each comprising 24 bits of data plus 8 bits of padding.

(a) Data Format in On-chip RAM

Figure 4.1 shows the format of audio data in the on-chip RAM.

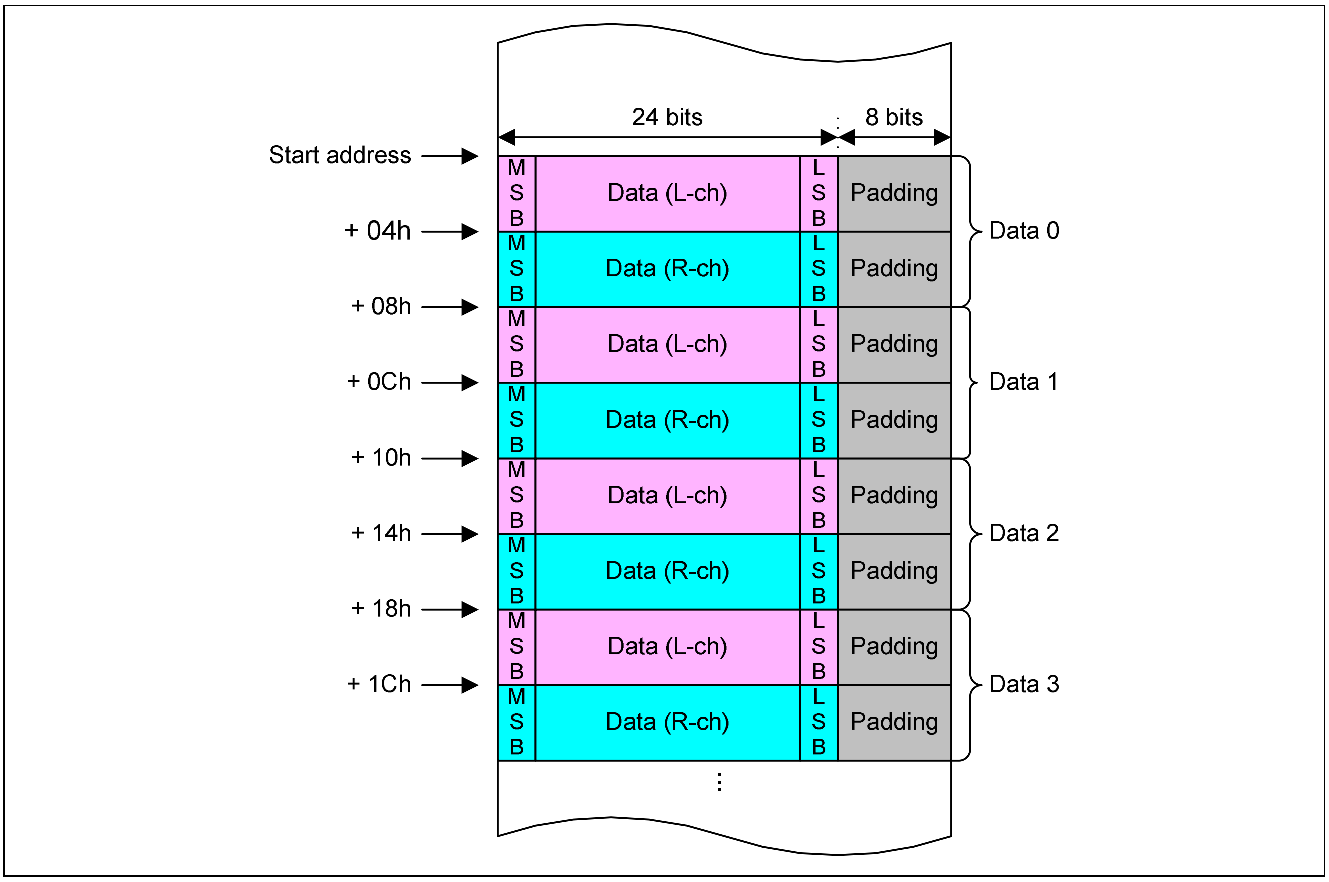


Figure 4.1 Audio Data Format in On-Chip RAM

(b) Transmit/Receive Data Format

The sample program performs serial transfers using a data length of 32 bits and MSB-first bit ordering. SCK is used as the serial transfer clock and WS as the channel select signal. The L-ch portion of the audio data is transferred when WS is low-level, and the R-ch portion is transferred when WS is high-level.

The position where padding is added to the transmit/receive data is selected according to the audio interface format of the external device. The sample program allows selection from among the following three formats.

- Standard format
- Backward-padding format
- Forward-padding format

Figure 4.2 shows the transmit/receive data formats.

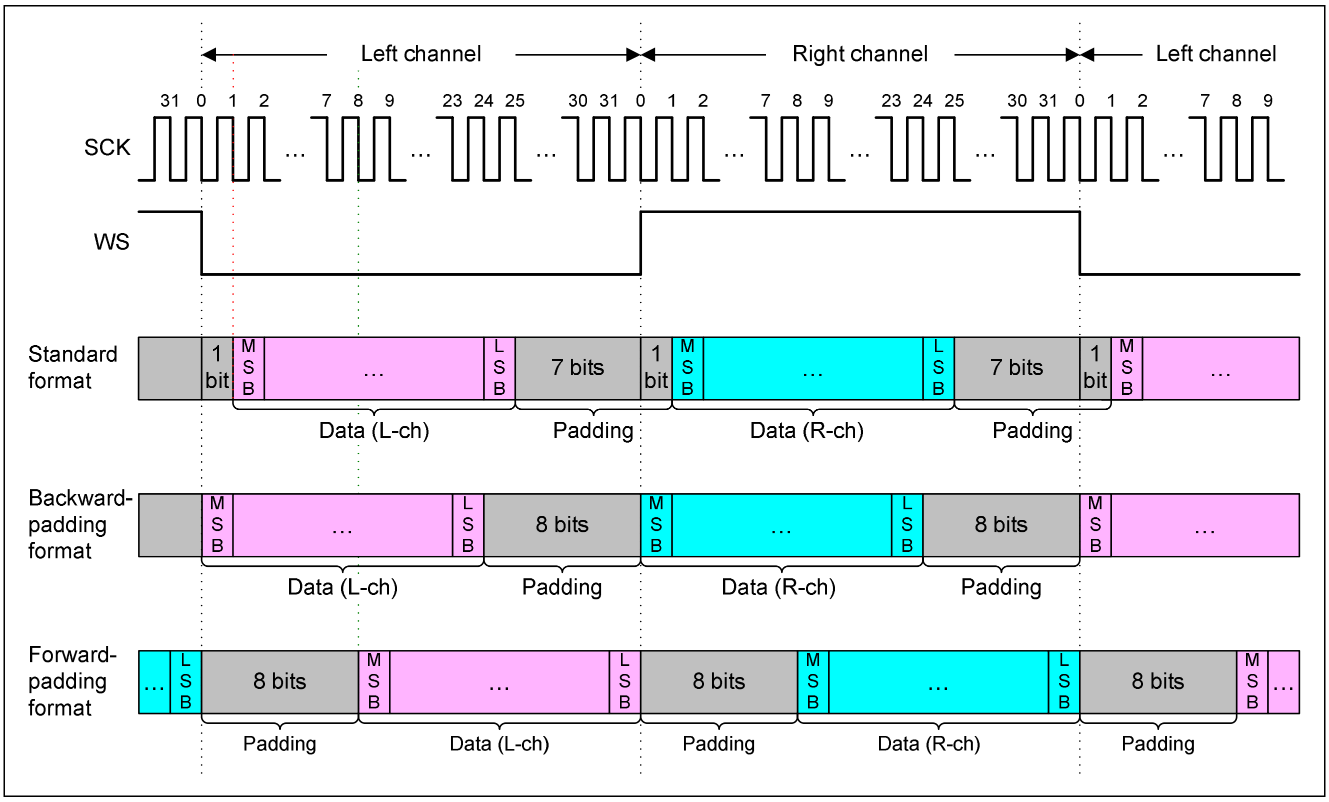


Figure 4.2 Transmit/Receive Data Formats

(2) Transmitting/Receiving Audio Data

In I^{2}S transmission, audio data located in the on-chip RAM is transferred to the RSPI by the DTC and then output by the RSPI. In I^{2}S reception, audio data received by the RSPI is transferred to the on-chip RAM by the DTC.

The RSPI operates in slave mode (SPI operation) using a data length of 32 bits and MSB-first bit ordering.

Figure 4.3 illustrates the flow of audio data.

The audio data is separated into L-ch and R-ch portions, with the L-ch portion transmitted/received by RSPI0 and the R-ch portion by RSPI1. The slave-select polarity of RSPI0 is set to low-active and that of RSPI1 to high-active. The SSL signal output on MTU2 channel 4 is used for channel selection between RSPI0 and RSPI1.

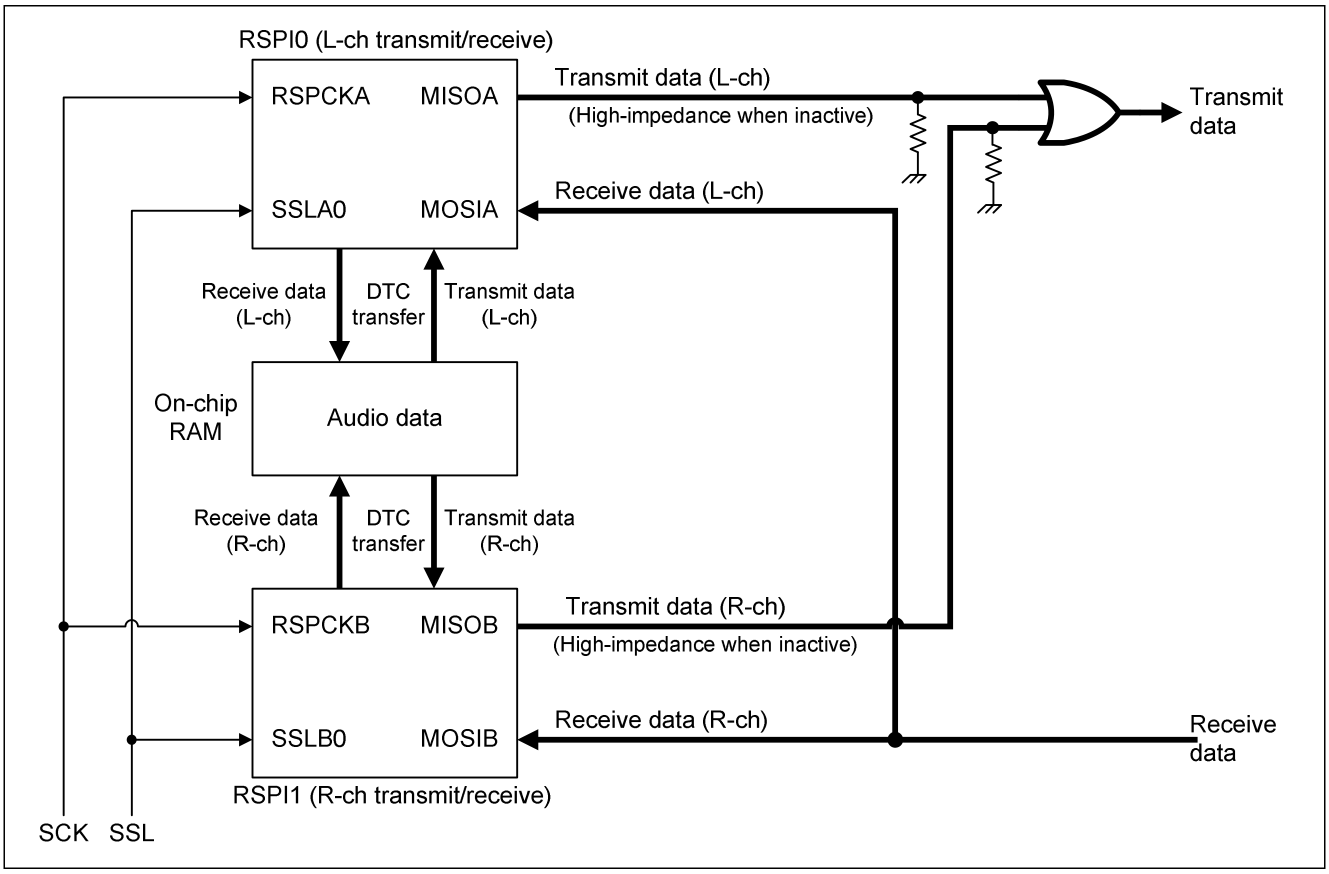


Figure 4.3 Flow of Audio Data

(a) Transmit Operation

Figure 4.4 shows the audio data transmit timing of the RSPI.

For transmission of audio data to the external device, the transmit data (L-ch) output from RSPI0 and the transmit data (R-ch) output from RSPI1 are generated through synthesis by an external OR circuit. At this time, the RSPI output channel that is not outputting transmit data is high-impedance, and it should be pulled down to prevent it from affecting the data synthesized by the OR circuit.

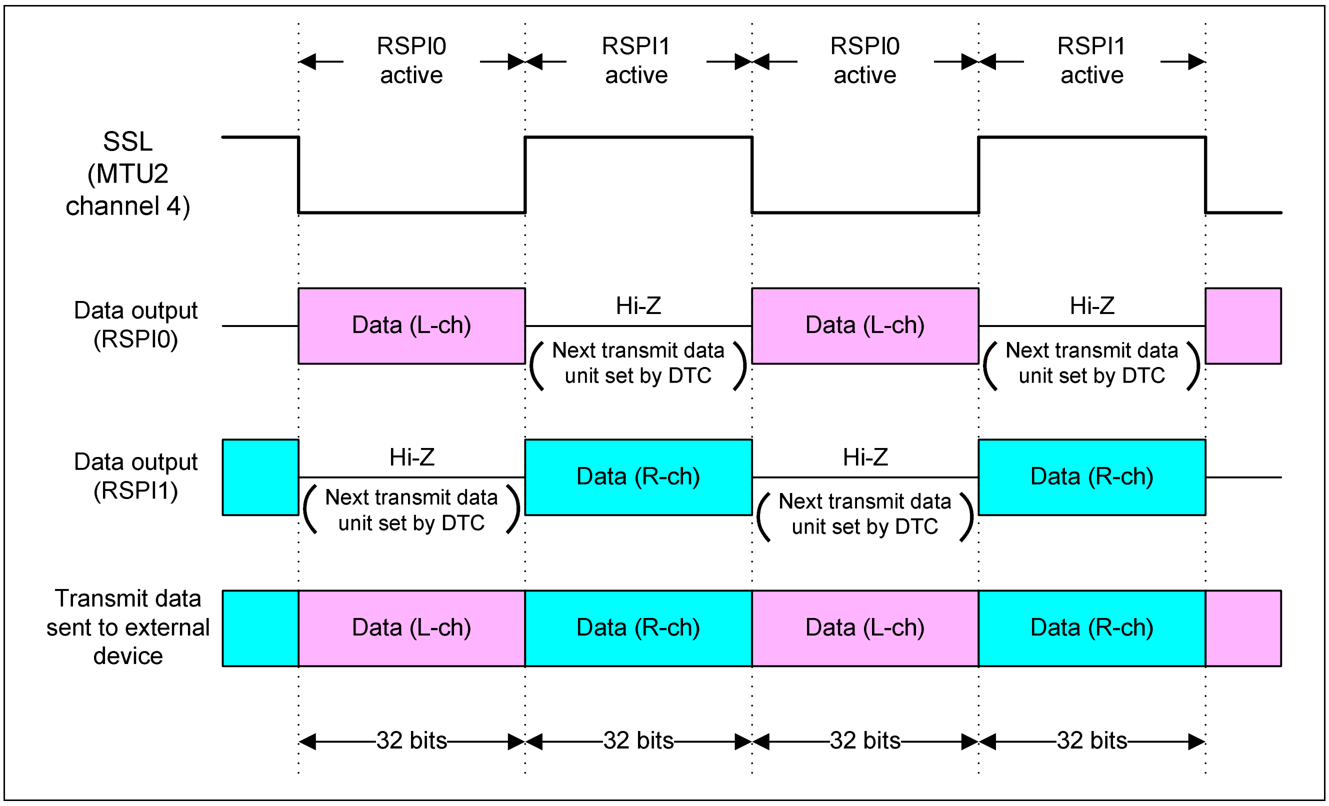


Figure 4.4 RSPI Audio Data Transmit Timing

(b) Receive Operation

Figure 4.5 shows the audio data receive timing of the RSPI.

For reception of audio data from the external device, data is input to both channels, RSPI0 and RSPI1. At this time, the active RSPI channel is selected according to the SSL signal, and the input audio data is divided into L-ch and R-ch portions and received.

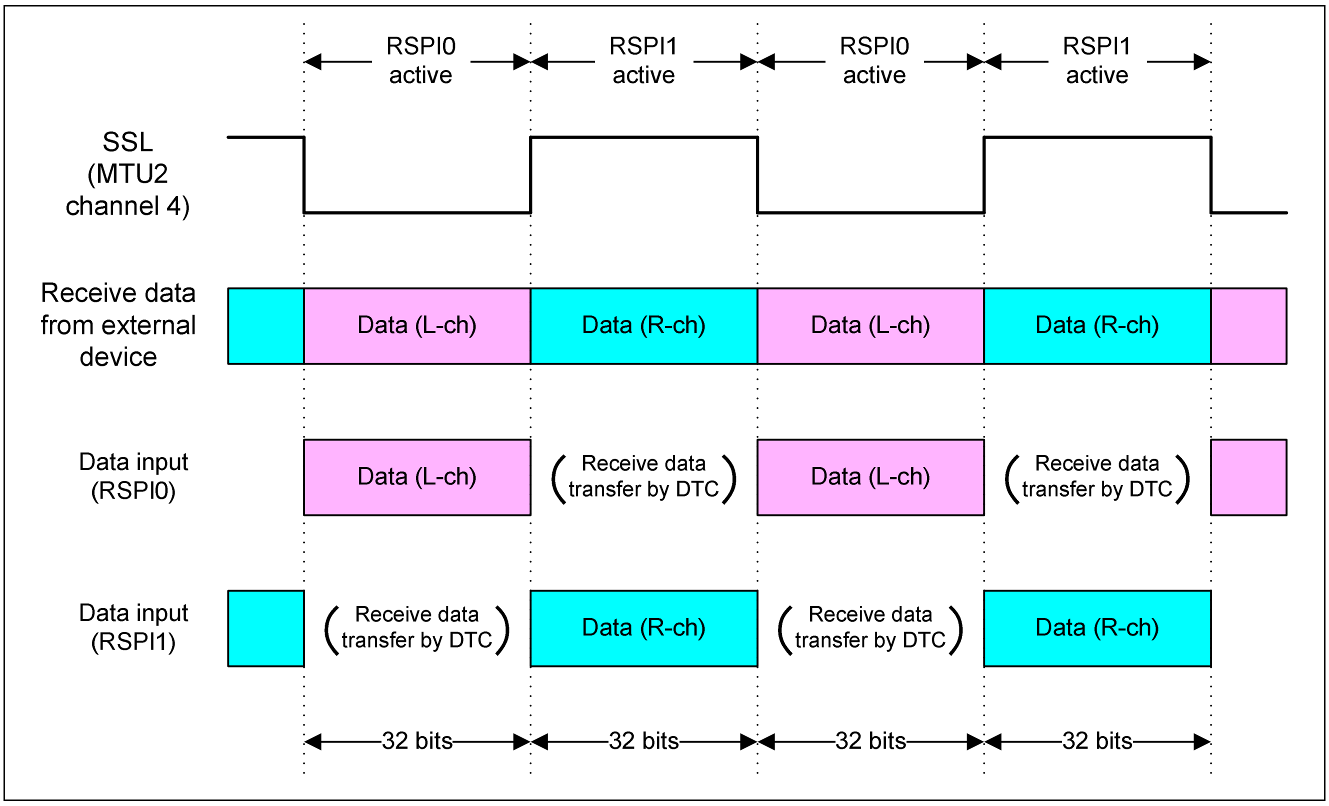


Figure 4.5 RSPI Audio Data Receive Timing

(3) I^{2}S Communication Synchronization Recovery and RSPI Initialization

I^{2}S communication synchronization recovery is performed separately for L-ch and R-ch by initializing the RSPI. RSPI initialization is performed by the DTC.

Figure 4.6 shows the RSPI initialization timing.

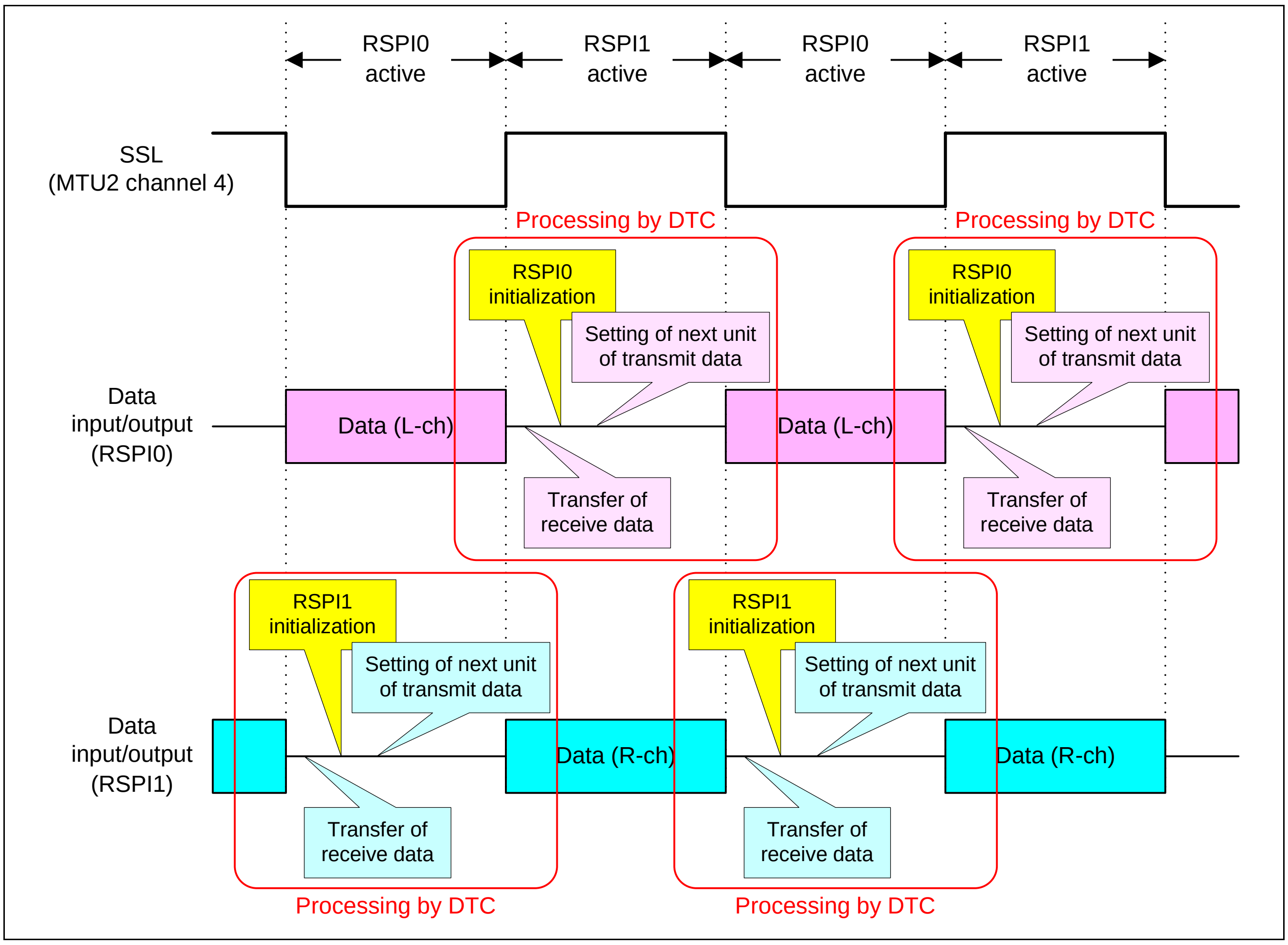


Figure 4.6 RSPI Initialization Timing

(4) Generation of I^2S Communication Clock and Channel Synchronization Signals

The external clock (12.288 MHz) is used as the count clock, and SCK and WS are generated by the MTU2 by using compare-match. In like manner, the RSPI channel select signal (SSL) is generated by the MTU2 by using compare-match.

(a) Generation of SCK

MTU2 channel 2 is set to PWM mode 1 and SCK is output with a frequency of 3.072 MHz (12.288 MHz / 4), a duty ratio of 50%, and high-level initial output.

Figure 4.7 illustrates the generation of SCK on MTU2 channel 2.

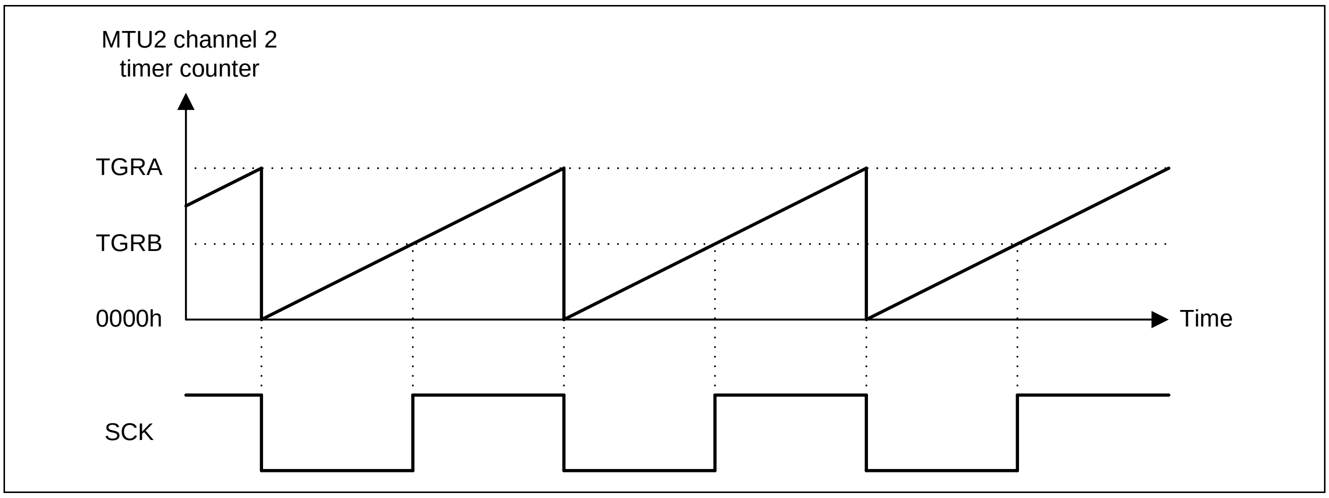


Figure 4.7 Generation of SCK on MTU2 Channel 2

(b) Generation of WS

MTU2 channel 3 is set to PWM mode 1 and WS is output with a frequency of 48 kHz (12.288 MHz / 256), a duty ratio of 50%, and high-level initial output.

Figure 4.8 illustrates the generation of WS on MTU2 channel 3.

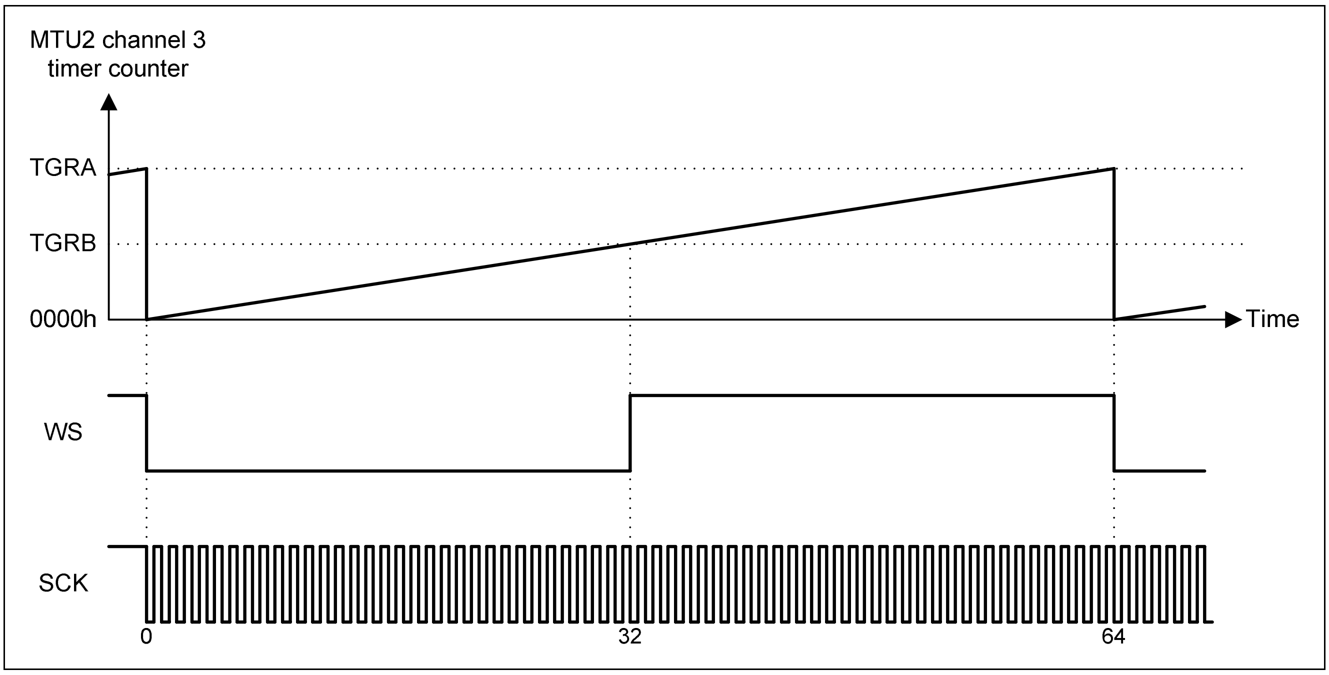


Figure 4.8 Generation of WS on MTU2 Channel 3

(c) Generation of SSL

MTU2 channel 4 is set to PWM mode 1 and SSL is output with a frequency of 48 kHz (12.288 MHz / 256), a duty ratio of 50%, and high-level initial output.

A phase difference is generated between WS and SSL by setting a timer counter initial value for MTU2 channel 4 that is different from the timer counter initial value of MTU2 channel 3, which generates WS. Different phase differences between WS and SSL correspond to the transmit/receive data formats shown in 4.1.1 (1) (b), Transmit/receive data format. For details of the phase difference between WS and SSL, see 4.1.1 (5), Phase Difference between Word Select Signal and Slave Select Signal.

Figure 4.9 illustrates the generation of SSL on MTU2 channel 4.

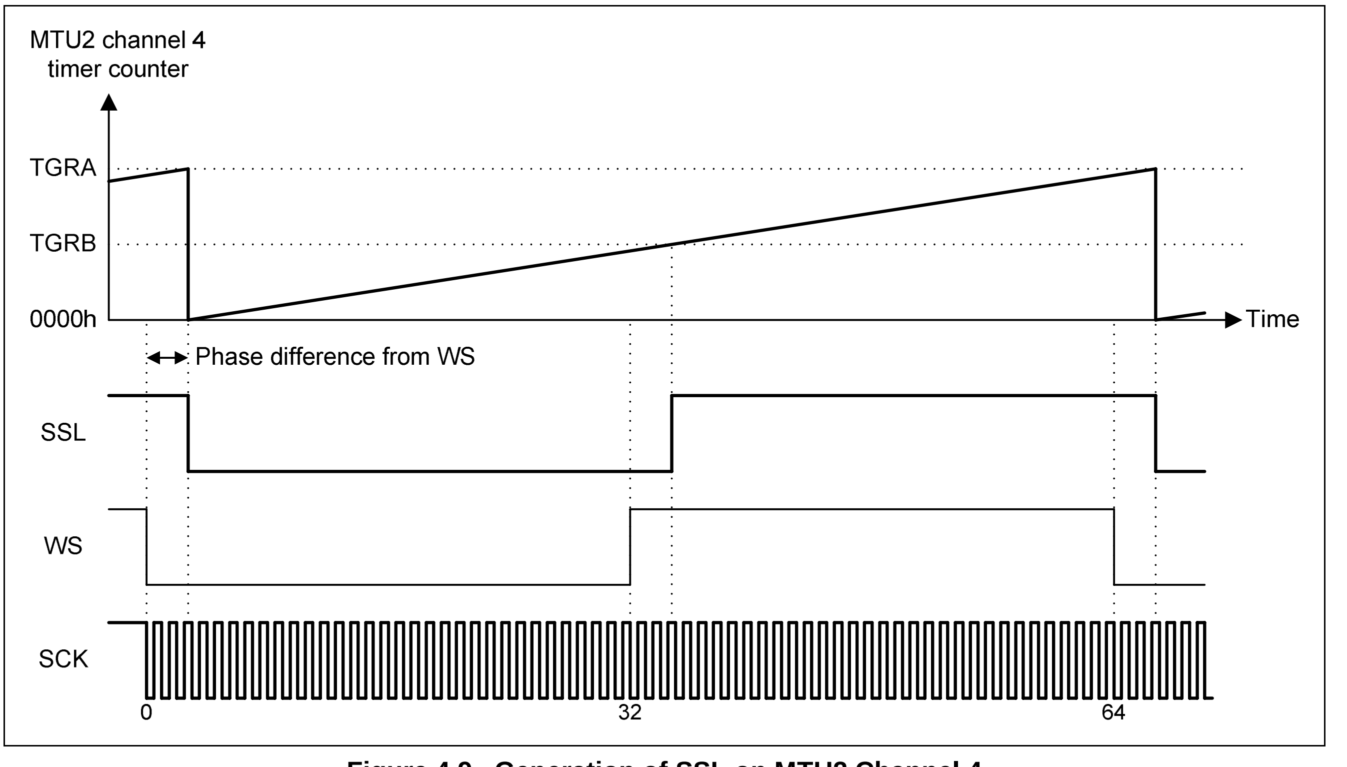


Figure 4.9 Generation of SSL on MTU2 Channel 4

(5) Phase Difference between Word Select Signal and Slave Select Signal

The sample program supports the transmit/receive data formats shown in 4.1.1 (1) (b), Transmit/receive data format, by changing the initial value of MTU2 channel 4 as described in 4.1.1 (4) (c), Generation of SSL.

Figure 4.10 shows the phase difference between WS and SSL in each transmit/receive data format.

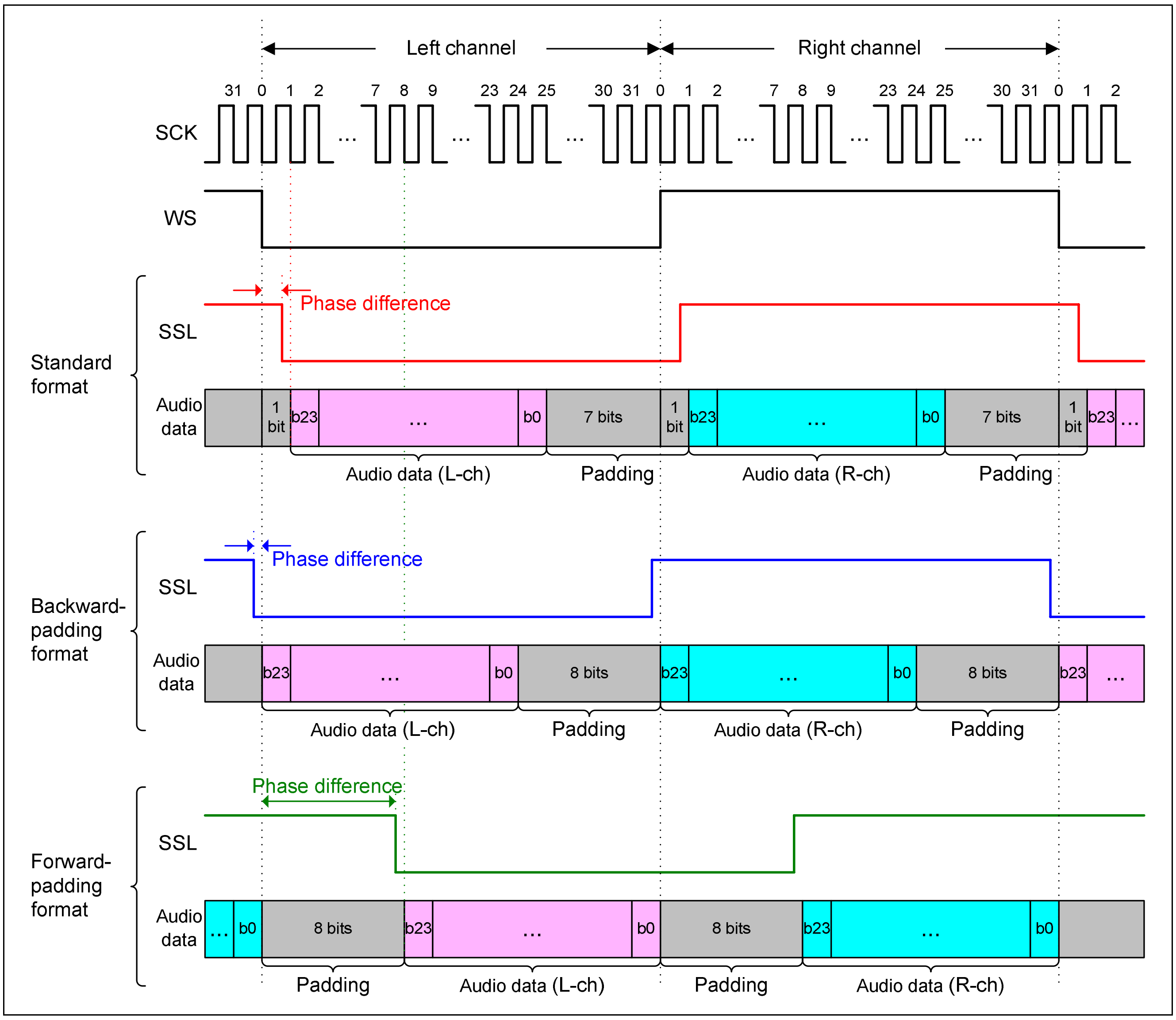


Figure 4.10 Phase Difference between WS and SSL in Each Transmit/Receive Data Format

4.1.2 Transmit Operation

(1) Transmit Operation Timing

Figure 4.11 shows the transmit operation timing (standard format).

The RSPI transmits audio data located in the on-chip RAM. The RSPI operates in slave mode (SPI operation) using a data length of 32 bits and MSB-first bit ordering.

The audio data to be transmitted is separated into L-ch and R-ch portions, with the L-ch portion transmitted by RSPI0 and the R-ch portion by RSPI1. The slave-select polarity of RSPI0 is set to low-active and that of RSPI1 to high-active. Channel selection between RSPI0 and RSPI1 takes place according to the SSL signal.

Data output on the active RSPI channel starts at the falling edge of the first SCK signal generated after channel switching, and thereafter data output is synchronized with SCK.

The MTU2 outputs the SSL signal with a phase delay relative to WS of approximately one bit according to SCK. For details of the phase difference between WS and SSL, see 4.1.1 (5), Phase Difference between Word Select Signal and Slave Select Signal.

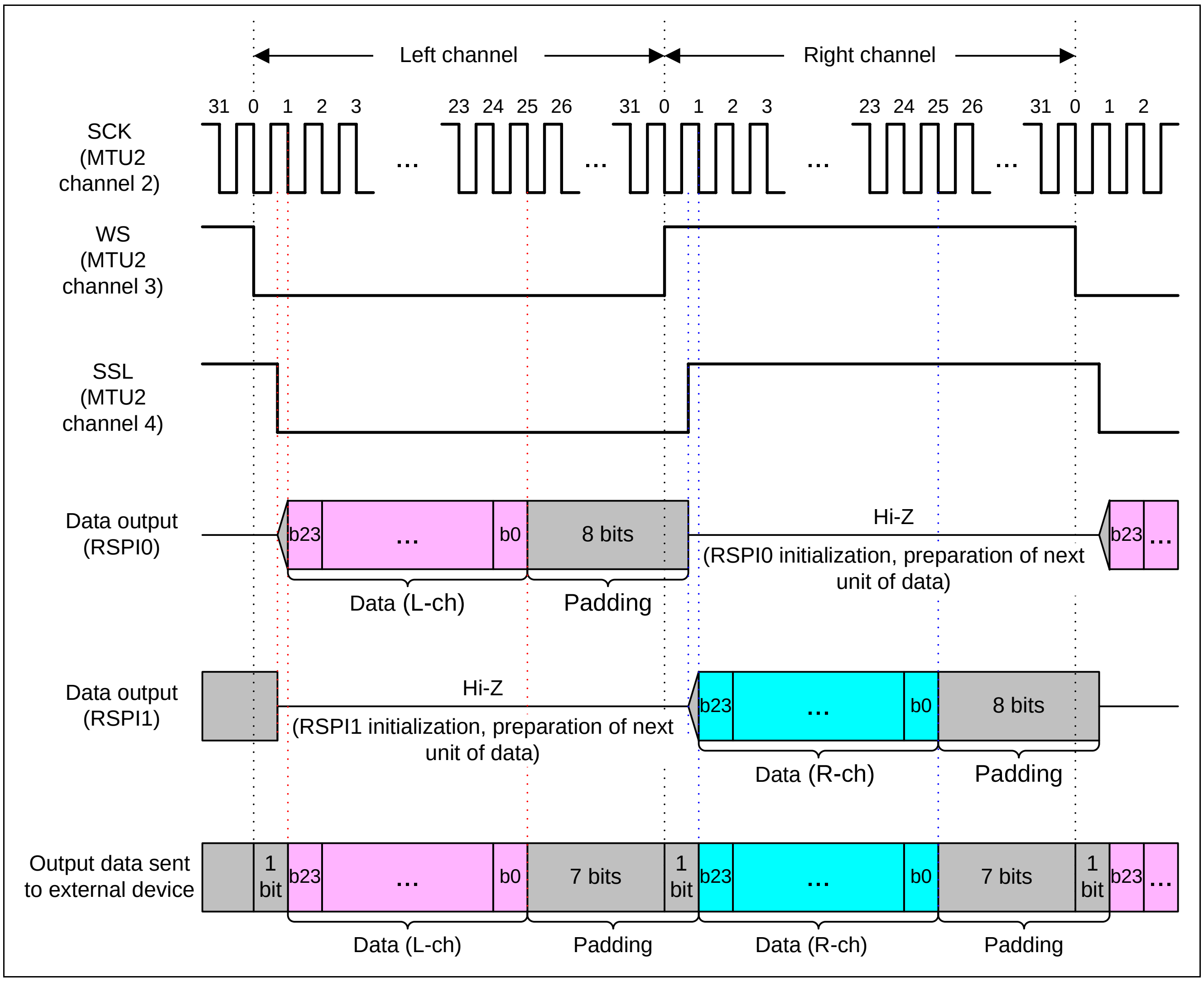


Figure 4.11 Transmit Operation Timing (Standard Format)

(2) DTC Operation during Transmission

Figure 4.12 illustrates the operation of the DTC during transmission.

The DTC transfers transmit data from the on-chip RAM to the RSPI. DTC transfers take place in pairs, one for L-ch transmit and one for R-ch transmit. The DTC generates a compare-match interrupt at each SSL edge. The DTC transfer for L-ch transmit starts at the SSL rising edge, and the DTC transfer for R-ch transmit starts at the SSL falling edge.

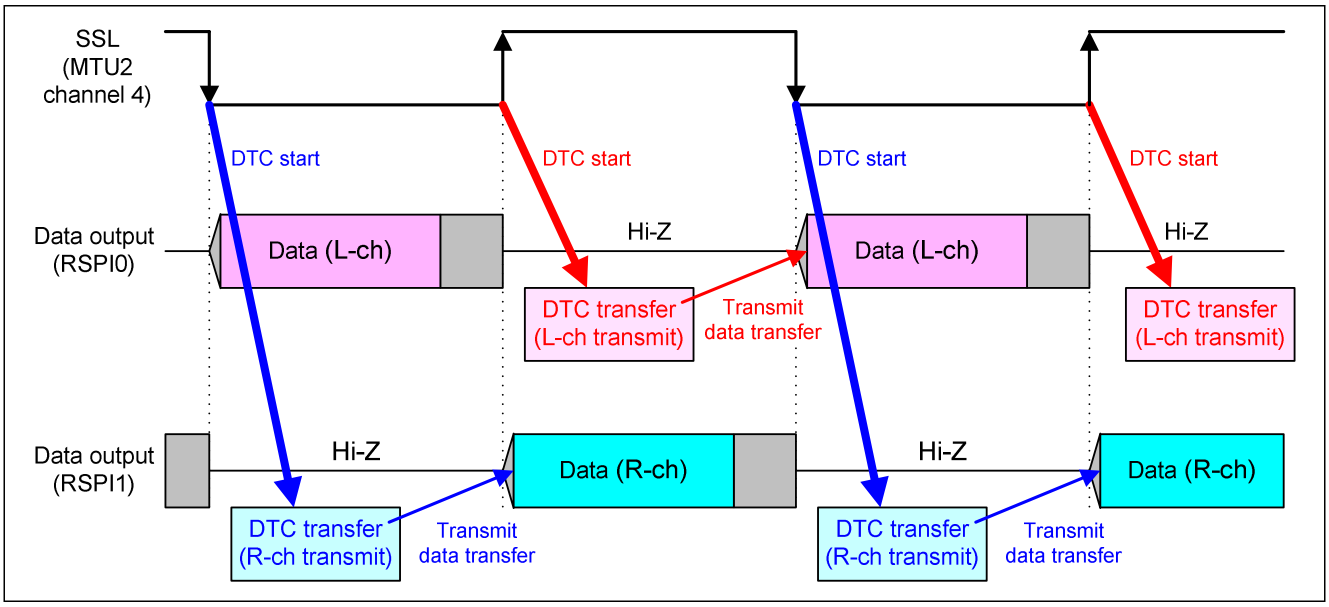


Figure 4.12 DTC Operation during Transmission

Figure 4.13 illustrates DTC operation during L-ch transmit, and figure 4.14 illustrates DTC operation during R-ch transmit.

The DTC uses chained transfer to initialize the RSPI, transfer transmit data, and transfer the transmit data address. Note that the RSPI is initialized by writing 0 to the RSPI function enable bit (SPE) in the RSPI control register (SPCR) to disable the RSPI and then writing 1 to the same bit to re-enable the RSPI.

Note: See the Hardware Manual for information on using the RSPI function enable bit to initialize the RSPI.

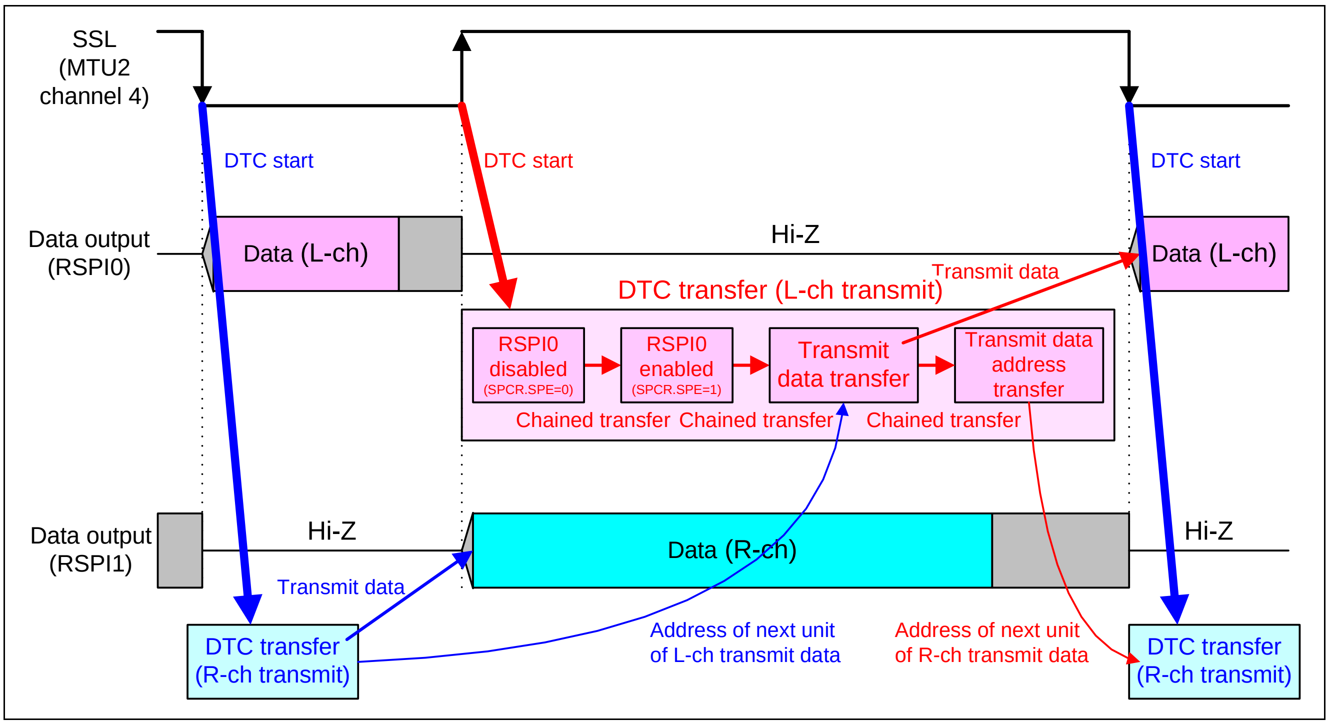


Figure 4.13 DTC Operation during L-ch Transmit

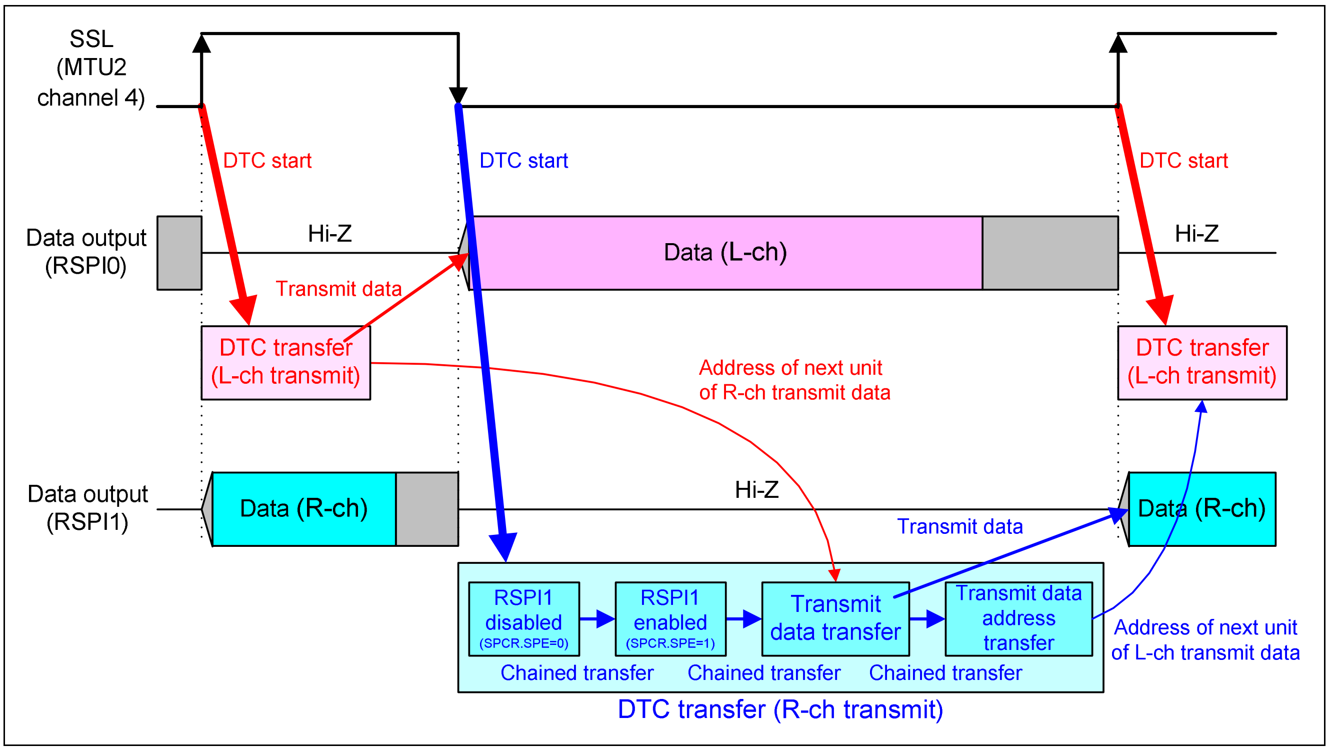


Figure 4.14 DTC Operation during R-ch Transmit

(3) Transmit Data Address Transfer by DTC

Figure 4.15 illustrates transfer of transmit data addresses by the DTC.

The sample program arranges the audio data to be transmitted in alternating L-ch and R-ch portions, each 4 bytes in length, in the on-chip RAM as shown in figure 4.1, Audio Data Format in On-Chip RAM. The sample program transfers the transmit data in pairs of DTC transfers, each of which consists of an L-ch transmit DTC transfer and an R-ch transmit DTC transfer.

As shown in figure 4.15, in the L-ch transmit DTC transfer, data 0 (L-ch) of the transmit data is transferred, after which chained transfer is used to transfer the transfer source address after the L-ch transmit DTC transfer to the transfer source address of the R-ch transmit DTC transfer. In the R-ch transmit DTC transfer, the transfer source address after the L-ch transmit DTC transfer is used as the start address for transferring data 0 (R-ch), then chained transfer is used to transfer the transfer source address after the R-ch transmit DTC transfer to the transfer source address of the L-ch transmit DTC transfer.

In the L-ch transmit DTC transfer, the transfer source address after the R-ch transmit DTC transfer, transferred by the R-ch transmit chained transfer, is used as the start address for transferring data 1 (L-ch), then chained transfer is used to transfer the transfer source address after the L-ch transmit DTC transfer to the transfer source address of the R-ch transmit DTC transfer, which is required as the start address for the next R-ch transmit DTC transfer.

In this manner, each L-ch transmit DTC transfer transfers the transmit data address required by the following R-ch transmit DTC transfer, and each R-ch transmit DTC transfer transfers the transmit data address required by the following L-ch transmit DTC transfer.

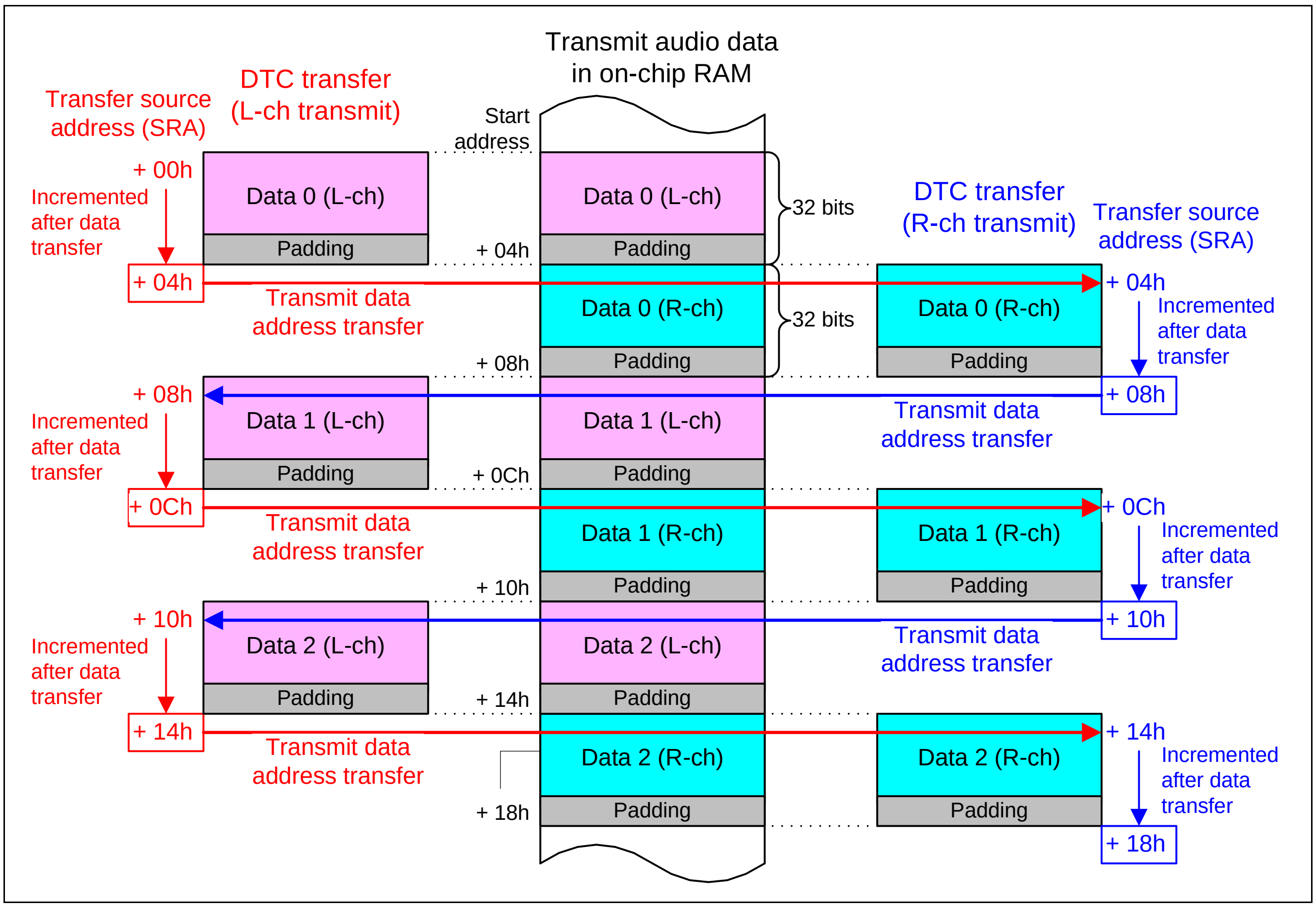

Transmit audio data in on-chip RAM
Transfer source address (SRA)
DTC transfer (L-ch transmit)
Start address
+ 00h
Incremented after data transfer
Data 0 (L-ch)
Padding
+ 04h
32 bits
DTC transfer (R-ch transmit)
Transfer source address (SRA)
Transmit data address transfer
Data 0 (R-ch)
+ 08h
Data 1 (L-ch)
+ 0Ch
Data 1 (R-ch)
+ 10h
Data 2 (L-ch)
+ 14h
Data 2 (R-ch)
+ 18h

Figure 4.15 Transmit Data Address Transfer by DTC

(4) Audio Data Transmit End

The sample program ends transmission of audio data by disabling the RSPI, thereby halting its operation. Disabling the RSPI is performed for L-ch transmit and R-ch transmit DTC transfers.

(a) Transmit end processing for L-ch transmit

Figure 4.16 shows an outline of transmit end operation for L-ch transmit.

When the n–1'th L-ch transmit DTC transfer starts, chained transfer is used to perform transmit end processing. By means of transmit end processing, channel RSPI0 only is disabled when the n'th transfer starts, thereby ending RSPI0 operation.

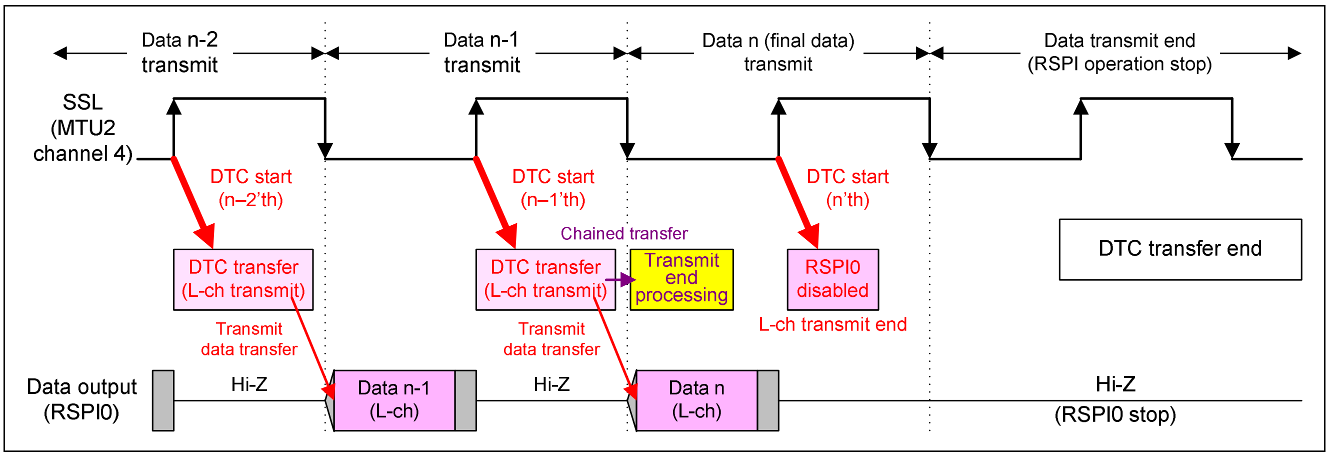


Figure 4.16 Transmit End Operation for L-ch Transmit

Figure 4.17 illustrates in detail the use of chained transfer to perform transmit end processing for L-ch transmit.

As shown in figure 4.13, DTC Operation during L-ch Transmit, for L-ch transmit DTC transfer, chained transfer is used to disable RSPI0, to enable RSPI0, to transfer the transmit data, and to transfer the transmit data address.

By setting to n–1 the transfer counter, which is part of the transfer information of the transmit data address transfer, and setting the DTC chain transfer select bit to 1 (chain transfer is performed only when the transfer counter is 0), transmit end processing is triggered by chained transfer after the transmit data address transfer at the n–1'th start.

Transmit end processing disables chained transfer after the disabling of RSPI0 at the n'th start by overwriting the value of the DTC chain transfer enable bit in the RSPI0 disable transfer information. In this way, RSPI0 only is disabled at the n'th start, stopping RSPI0 operation and ending transmit operation.

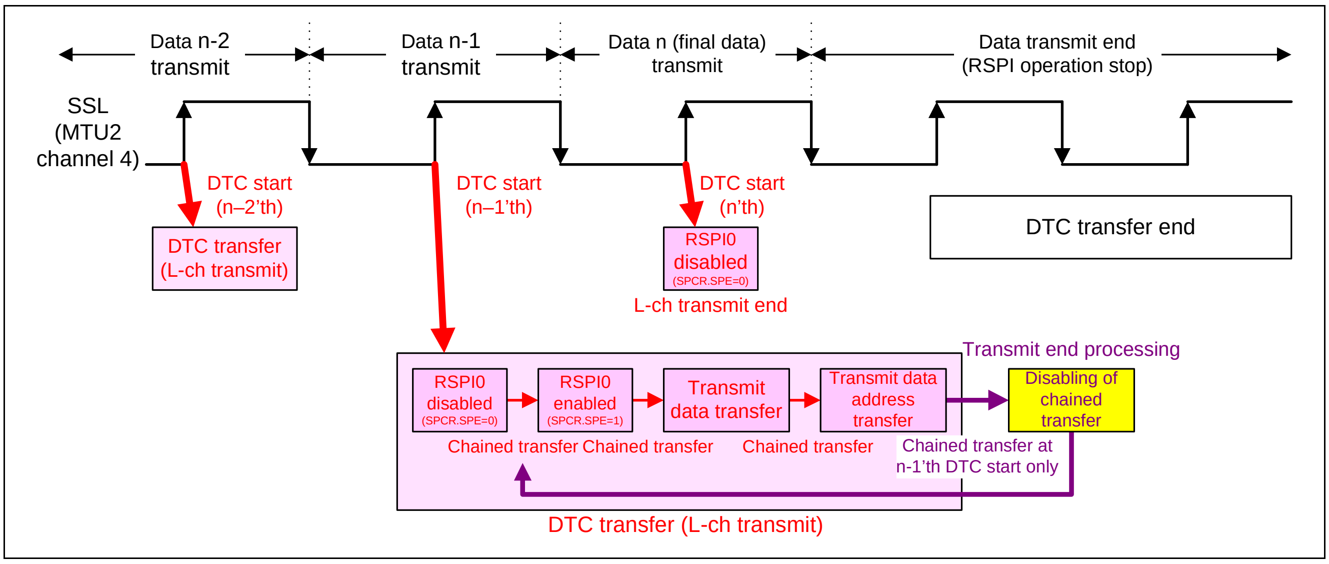

Data n-2 transmit
Data n-1 transmit
Data n (final data) transmit
Data transmit end (RSPI operation stop)
SSL (MTU2 channel 4)
DTC start (n–2'th)
DTC start (n–1'th)
DTC start (n'th)
DTC transfer (L-ch transmit)
RSPI0 disabled (SPCR.SPE=0)
L-ch transmit end
DTC transfer end
Transmit end processing
RSPI0 disabled (SPCR.SPE=0)
RSPI0 enabled (SPCR.SPE=1)
Transmit data transfer
Transmit data address transfer
Disabling of chained transfer
Chained transfer
Chained transfer
Chained transfer
Chained transfer at n-1'th DTC start only
DTC transfer (L-ch transmit)

Figure 4.17 Transmit End Processing by L-ch Transmit DTC Transfer

(b) Transmit end processing for R-ch transmit

Figure 4.18 shows an outline of transmit end operation for R-ch transmit.

When the n'th R-ch transmit DTC transfer starts, chained transfer is used to perform transmit end processing. By means of transmit end processing, channel RSPI1 only is disabled when the n+1'th transfer starts, thereby ending RSPI1 operation.

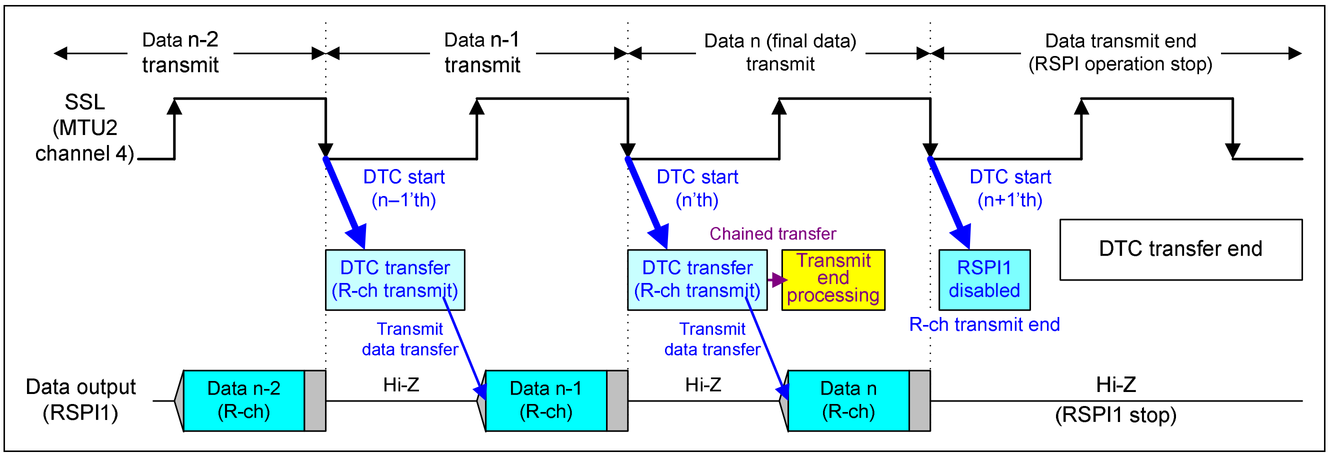


Figure 4.18 Transmit End Operation for R-ch Transmit

Figure 4.19 illustrates in detail the use of chained transfer to perform transmit end processing for R-ch transmit.

As shown in figure 4.14, DTC Operation during R-ch Transmit, for R-ch transmit DTC transfer, chained transfer is used to disable RSPI1, to enable RSPI1, to transfer the transmit data, and to transfer the transmit data address.

By setting to n the transfer counter, which is part of the transfer information of the transmit data address transfer, and setting the DTC chain transfer select bit to 1 (chain transfer is performed only when the transfer counter is 0), transmit end processing is triggered by chained transfer after the transmit data address transfer at the n'th start.

Transmit end processing disables chained transfer after the disabling of RSPI0 at the n+1'th start by overwriting the value of the DTC chain transfer enable bit in the RSPI1 disable transfer information. In this way, RSPI1 only is disabled at the n+1'th start, stopping RSPI1 operation and ending transmit operation.

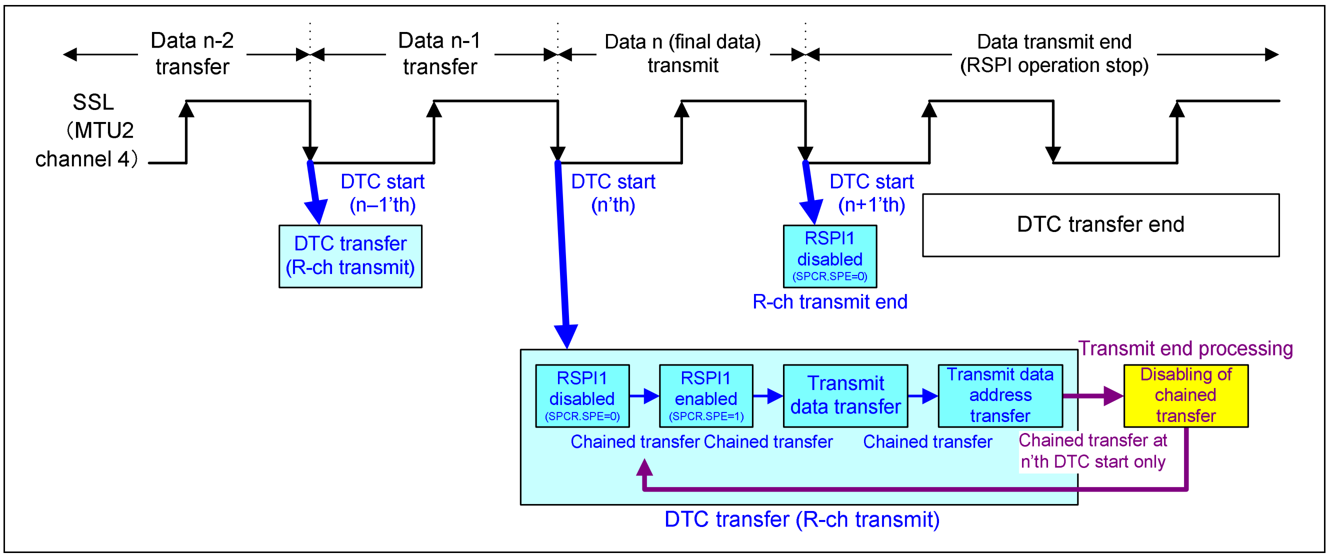

Data n-2 transfer
Data n-1 transfer
Data n (final data) transmit
Data transmit end (RSPI operation stop)
SSL (MTU2 channel 4)
DTC start (n–1'th)
DTC start (n'th)
DTC start (n+1'th)
DTC transfer (R-ch transmit)
RSPI1 disabled (SPCR.SPE=0)
R-ch transmit end
DTC transfer end
Transmit end processing
RSPI1 disabled (SPCR.SPE=0)
RSPI1 enabled (SPCR.SPE=1)
Transmit data transfer
Transmit data address transfer
Disabling of chained transfer
Chained transfer
Chained transfer
Chained transfer
Chained transfer at n'th DTC start only
DTC transfer (R-ch transmit)

Figure 4.19 Transmit End Processing by R-ch Transmit DTC Transfer

4.1.3 Receive Operation

(1) Receive Operation Timing

Figure 4.20 shows the timing of receive operation (standard format).

The RSPI receives audio data from the external device. The RSPI operates in slave mode (SPI operation) using a data length of 32 bits and MSB-first bit ordering.

The audio data from the external device is input to both channels, RSPI0 and RSPI1, and separated into L-ch and R-ch portions for reception. The slave-select polarity of RSPI0 is set to low-active and that of RSPI1 to high-active. Channel selection between RSPI0 and RSPI1 takes place according to the SSL signal. Data is imported on the active RSPI channel at the rising edge of the first SCK signal generated after channel switching, and thereafter importing of input data is synchronized with SCK.

The input data imported by the RSPI is transferred to the on-chip RAM by the DTC and stored as shown in figure 4.1.

The MTU2 outputs the SSL signal with a phase delay relative to WS of approximately one bit according to SCK. For details of the phase difference between WS and SSL, see 4.1.1 (5), Phase Difference between Word Select Signal and Slave Select Signal.

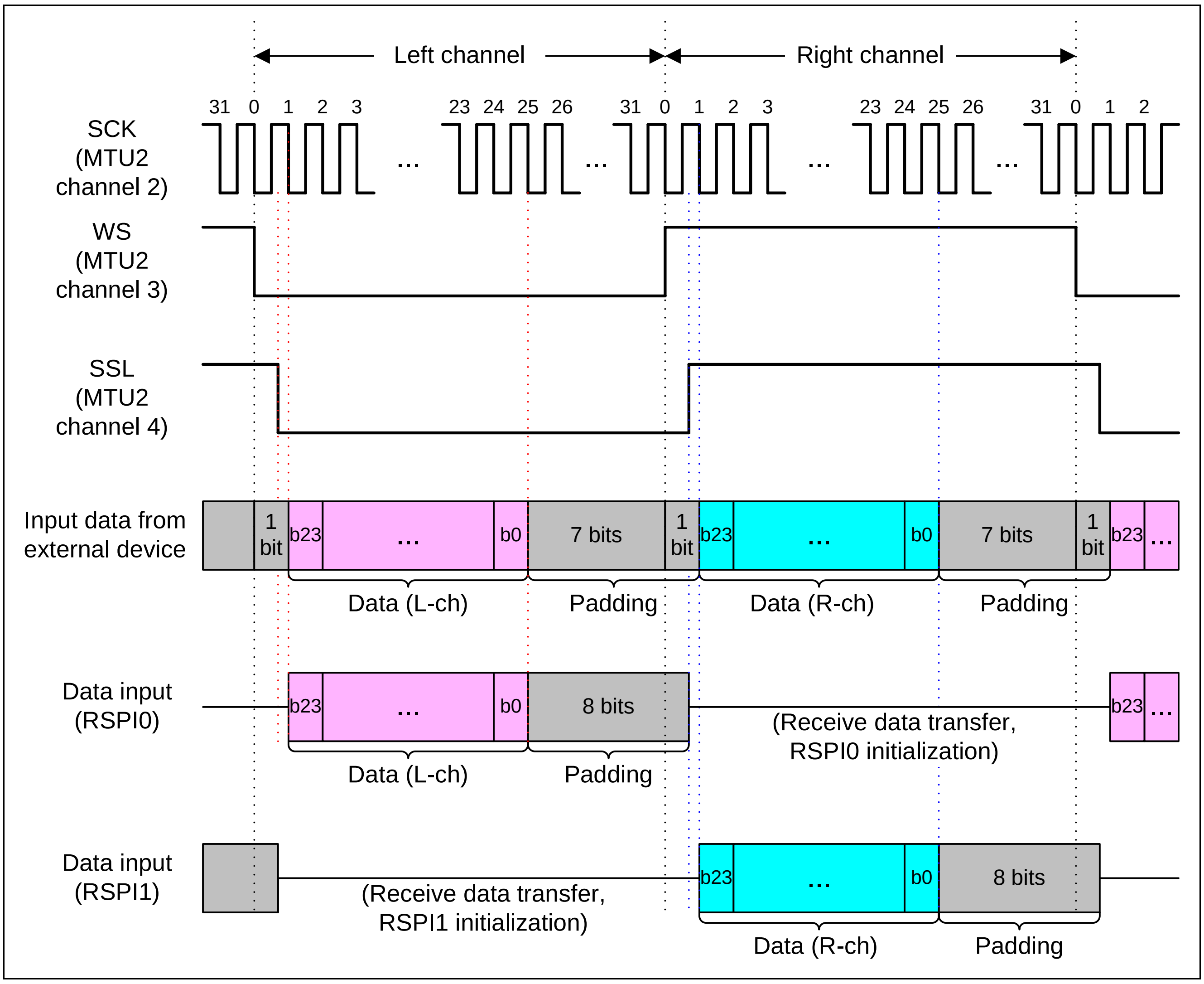


Figure 4.20 Receive Operation Timing (Standard Format)

(2) DTC Operation during Reception

Figure 4.21 illustrates the DTC operation during reception.

The DTC transfers receive data from the RSPI to the on-chip RAM. DTC transfers take place in sets of four: L-ch receive and R-ch receive, and RSPI0 initialization and RSPI1 initialization. For data reception, the DTC is started by a receive buffer full interrupt. The DTC activation for L-ch receive occurs at the RSPI0 receive buffer full interrupt, and the DTC activation for R-ch receive occurs at the RSPI1 receive buffer full interrupt. For RSPI initialization, the DTC is activated by a compare-match interrupt at each SSL edge. The DTC activation for RSPI0 initialization occurs at the SSL rising edge, and the DTC activation for RSPI1 initialization occurs at the SSL falling edge.

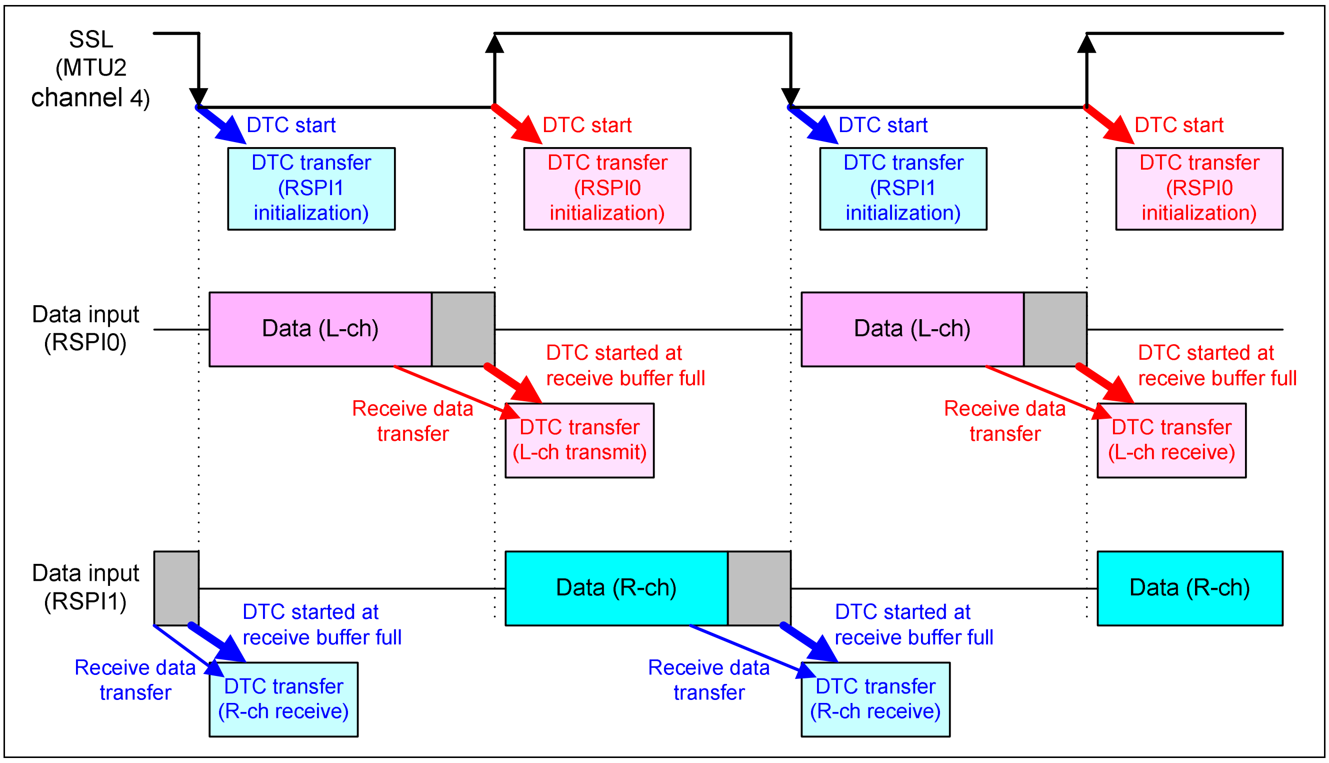

SSL (MTU2 channel 4)
DTC start
DTC transfer (RSPI1 initialization)
DTC start
DTC transfer (RSPI0 initialization)
DTC start
DTC transfer (RSPI1 initialization)
DTC start
DTC transfer (RSPI0 initialization)
Data input (RSPI0)
Data (L-ch)
DTC started at receive buffer full
Receive data transfer
DTC transfer (L-ch transmit)
Data (L-ch)
DTC started at receive buffer full
Receive data transfer
DTC transfer (L-ch receive)
Data input (RSPI1)
DTC started at receive buffer full
Receive data transfer
DTC transfer (R-ch receive)
Data (R-ch)
DTC started at receive buffer full
Receive data transfer
DTC transfer (R-ch receive)
Data (R-ch)

Figure 4.21 DTC Operation during Reception

Figure 4.22 illustrates DTC operation during L-ch receive, and figure 4.23 illustrates DTC operation during R-ch receive.

For data reception the DTC uses chained transfer to transfer receive data and transfer the receive data address. In addition, the DTC initializes the RSPI by using chained transfer. Note that the RSPI is initialized by writing 0 to the RSPI function enable bit (SPE) in the RSPI control register (SPCR) to disable the RSPI and then writing 1 to the same bit to re-enable the RSPI.

Note: See the Hardware Manual for information on using the RSPI function enable bit to initialize the RSPI.

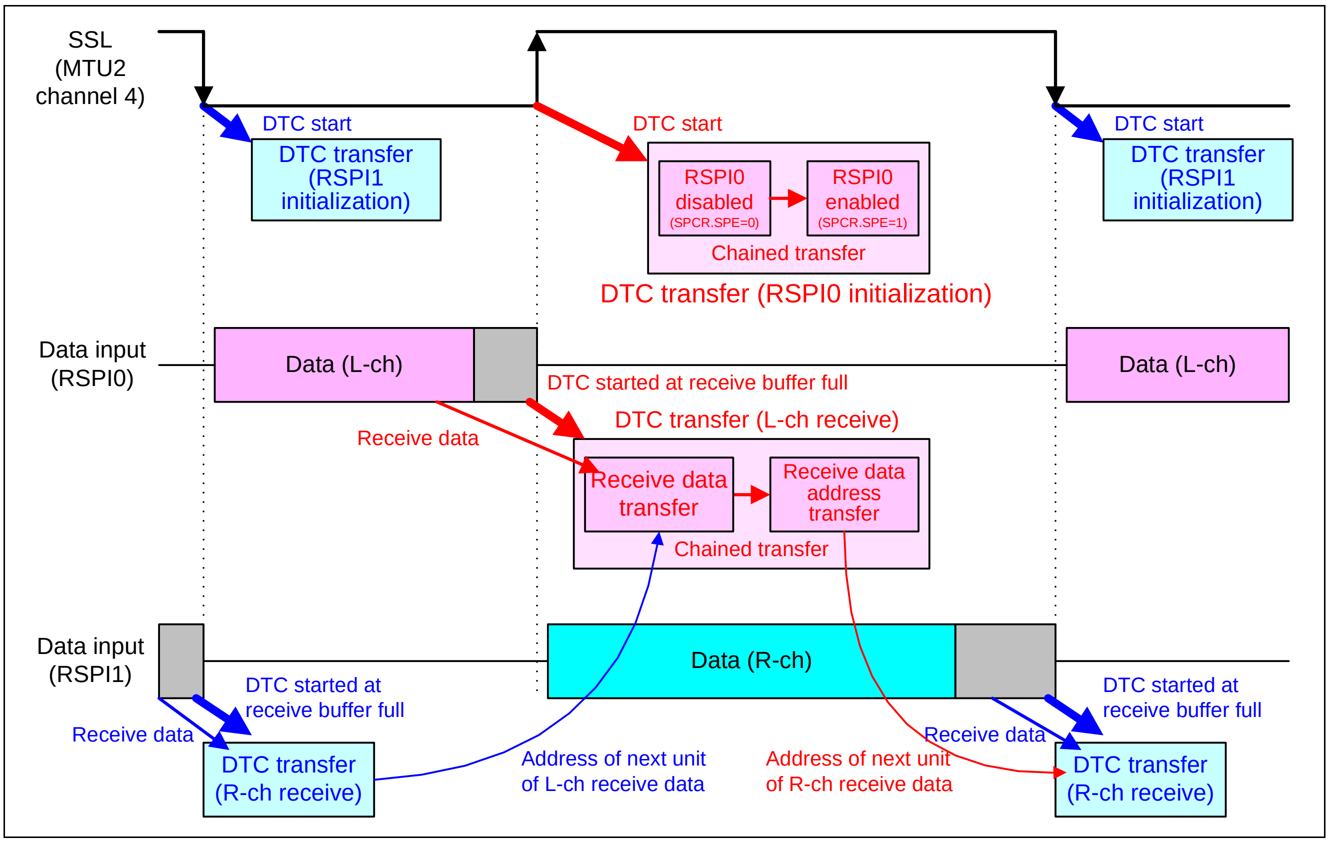

SSL (MTU2 channel 4)
DTC start
DTC transfer (RSPI1 initialization)
DTC start
RSPI0 disabled (SPCR.SPE=0)
RSPI0 enabled (SPCR.SPE=1)
Chained transfer
DTC transfer (RSPI0 initialization)
DTC start
DTC transfer (RSPI1 initialization)
Data input (RSPI0)
Data (L-ch)
DTC started at receive buffer full
Data (L-ch)
Receive data
DTC transfer (L-ch receive)
Receive data transfer
Receive data address transfer
Chained transfer
Data input (RSPI1)
Data (R-ch)
DTC started at receive buffer full
Receive data
DTC transfer (R-ch receive)
Address of next unit of L-ch receive data
Address of next unit of R-ch receive data
Receive data
DTC started at receive buffer full
DTC transfer (R-ch receive)

Figure 4.22 DTC Operation during L-ch Receive

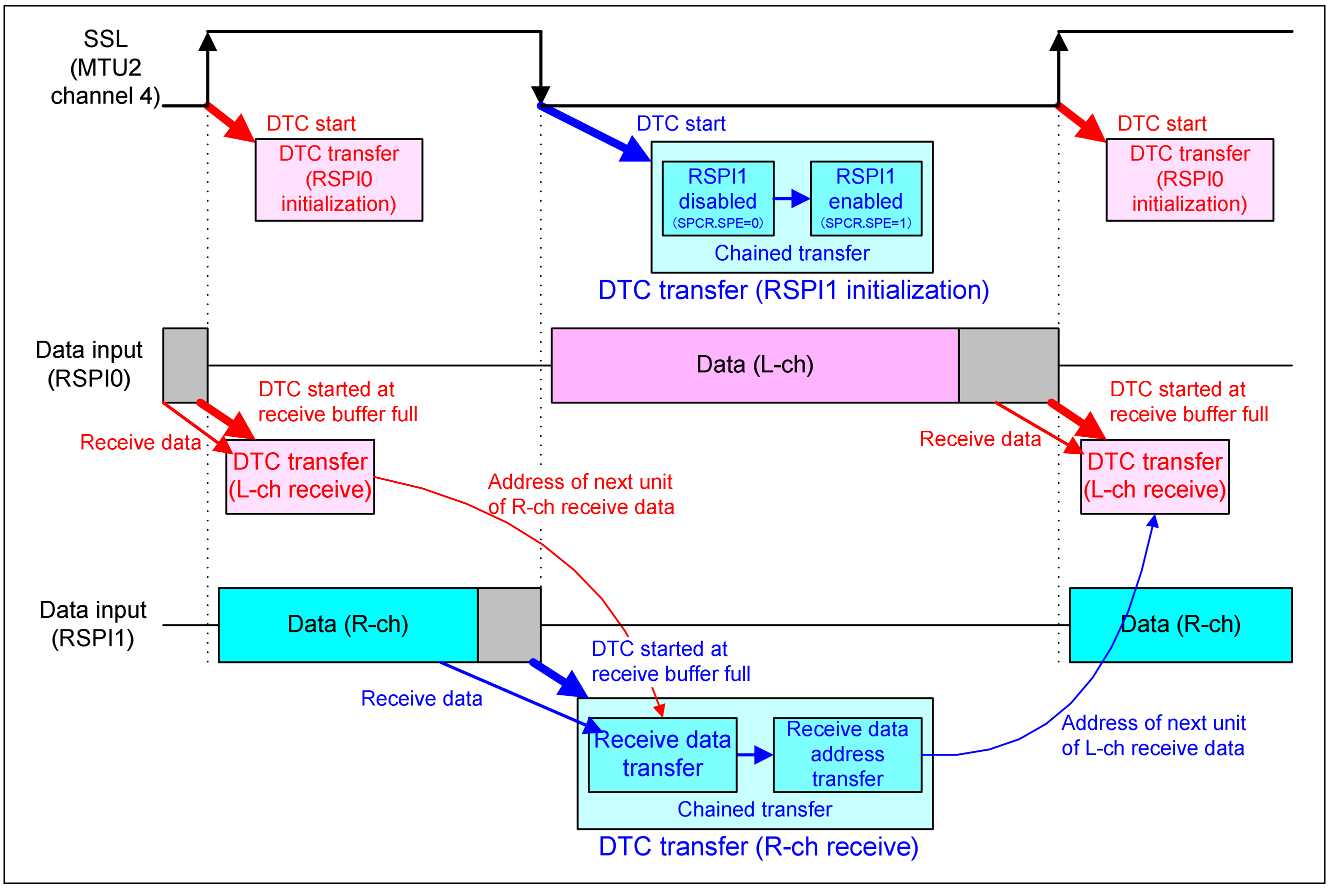

SSL (MTU2 channel 4)
DTC start
DTC transfer (RSPI0 initialization)
DTC start
RSPI1 disabled (SPCR.SPE=0)
RSPI1 enabled (SPCR.SPE=1)
Chained transfer
DTC transfer (RSPI1 initialization)
DTC start
DTC transfer (RSPI0 initialization)
Data input (RSPI0)
Data (L-ch)
DTC started at receive buffer full
Receive data
DTC transfer (L-ch receive)
Address of next unit of R-ch receive data
Receive data
DTC started at receive buffer full
DTC transfer (L-ch receive)
Data input (RSPI1)
Data (R-ch)
Data (R-ch)
DTC started at receive buffer full
Receive data
Receive data transfer
Receive data address transfer
Chained transfer
DTC transfer (R-ch receive)
Address of next unit of L-ch receive data

Figure 4.23 DTC Operation during R-ch Receive

(3) Receive Data Address Transfer by DTC

Figure 4.24 illustrates transfer of receive data addresses by the DTC.

The sample program arranges the received audio data in alternating L-ch and R-ch portions, each 4 bytes in length, in the on-chip RAM as shown in figure 4.1, Audio Data Format in On-Chip RAM. The sample program transfers the receive data in pairs of DTC transfers, each of which consists of an L-ch receive DTC transfer and an R-ch receive DTC transfer.

As shown in figure 4.24, in the L-ch receive DTC transfer, data 0 (L-ch) of the receive data is transferred, after which chained transfer is used to transfer the transfer destination address after the L-ch receive DTC transfer to the transfer destination address of the R-ch receive DTC transfer. In the R-ch receive DTC transfer, the transfer destination address after the L-ch receive DTC transfer is used as the start address for transferring data 0 (R-ch), then chained transfer is used to transfer the transfer destination address after the R-ch receive DTC transfer to the transfer destination address of the L-ch receive DTC transfer.

In the L-ch receive DTC transfer, the transfer destination address after the R-ch receive DTC transfer, transferred by the R-ch receive chained transfer, is used as the start address for transferring data 1 (L-ch), then chained transfer is used to transfer the transfer destination address after the L-ch receive DTC transfer to the transfer destination address of the R-ch receive DTC transfer, which is required as the start address for the next R-ch receive DTC transfer.

In this manner, each L-ch receive DTC transfer transfers the receive data address required by the following R-ch receive DTC transfer, and each R-ch receive DTC transfer transfers the receive data address required by the following L-ch receive DTC transfer.

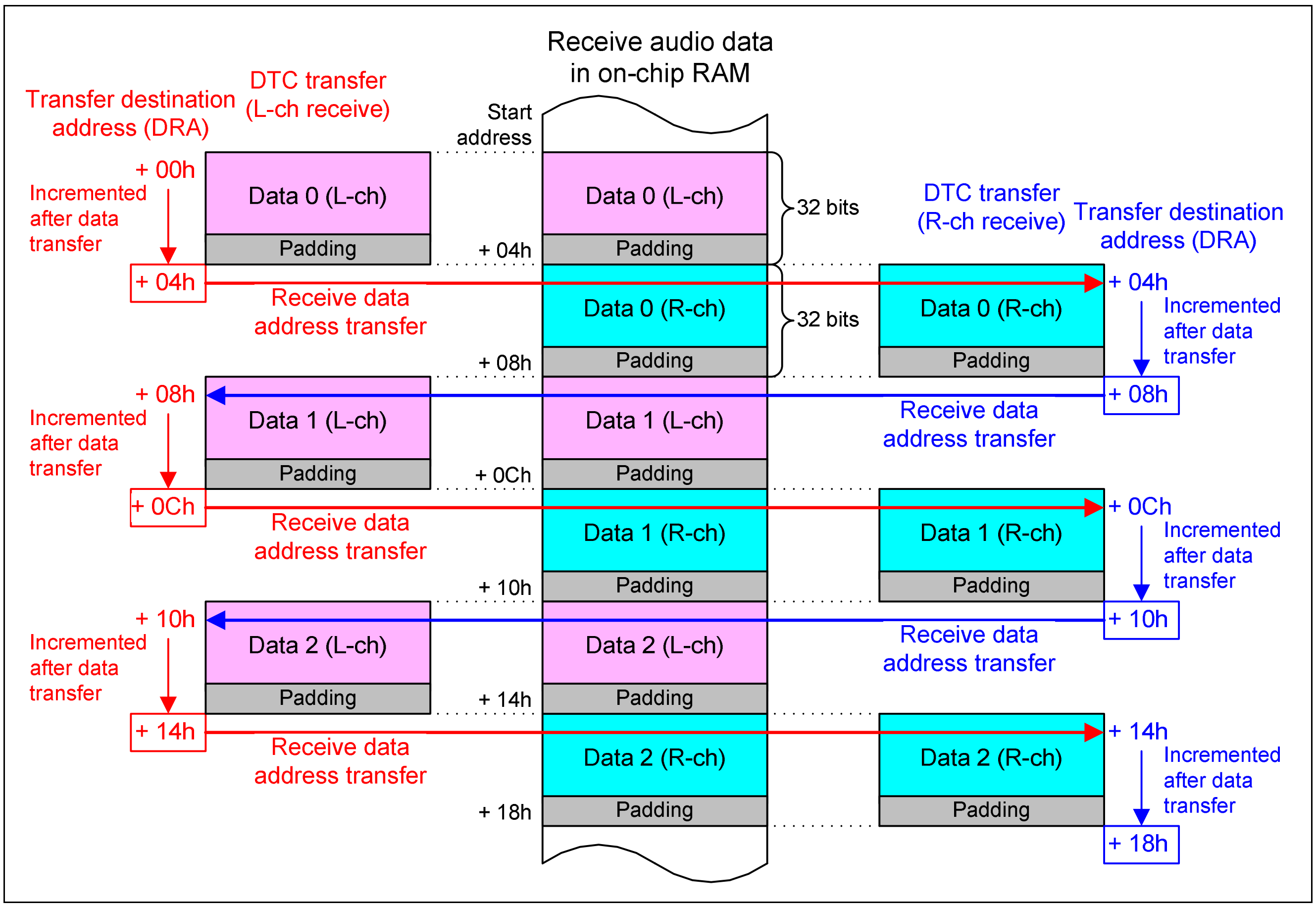

Receive audio data in on-chip RAM
Transfer destination address (DRA)
DTC transfer (L-ch receive)
Start address
+ 00h
Incremented after data transfer
Data 0 (L-ch)
Padding
+ 04h
32 bits
DTC transfer (R-ch receive)
Transfer destination address (DRA)
Receive data address transfer
Data 0 (R-ch)
+ 08h
Data 1 (L-ch)
+ 0Ch
Data 1 (R-ch)
+ 10h
Data 2 (L-ch)
+ 14h
Data 2 (R-ch)
+ 18h

Figure 4.24 Receive Data Address Transfer by DTC

(4) Audio Data Receive End

The sample program ends reception of audio data by disabling the RSPI, thereby halting its operation. Disabling the RSPI is performed for RSPI0 initialization and RSPI1 initialization DTC transfers.

(a) Receive end processing for L-ch receive

Figure 4.25 shows an outline of receive end operation for L-ch receive.

When the n–1'th RSPI0 initialization DTC transfer starts, chained transfer is used to perform receive end processing. By means of receive end processing, channel RSPI0 is disabled when the n'th transfer starts, thereby ending RSPI0 operation.

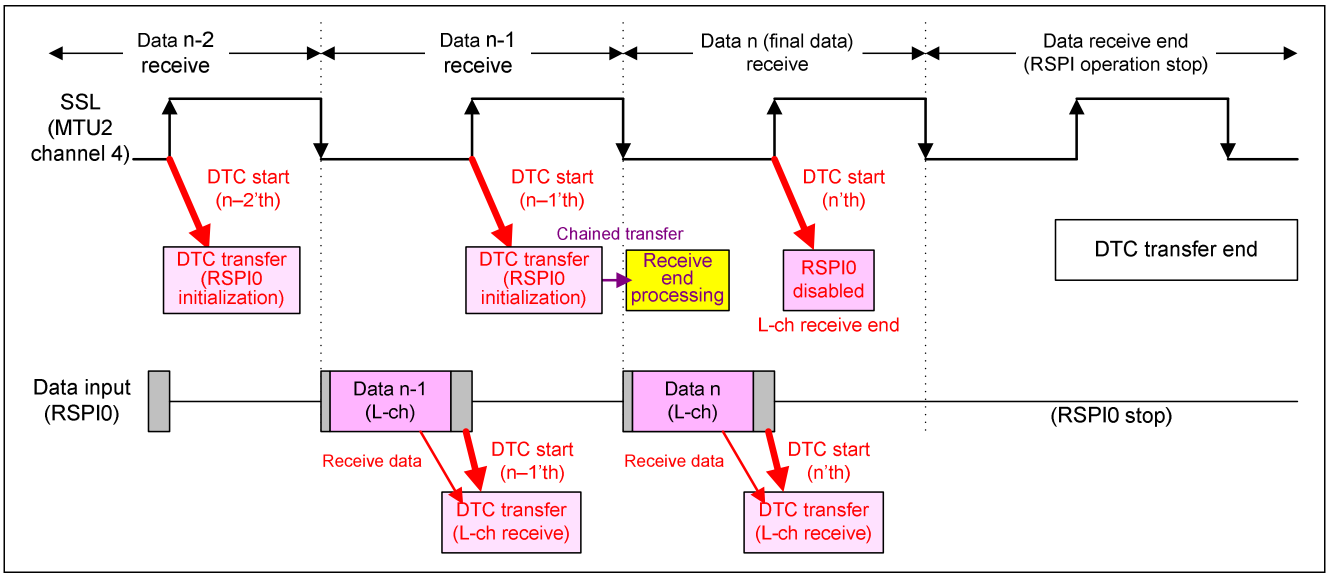


Figure 4.25 Receive End Operation for L-ch Receive

Figure 4.26 illustrates in detail the use of chained transfer to perform receive end processing for L-ch receive.

As shown in figure 4.22, DTC Operation during L-ch Receive, for RSPI0 initialization DTC transfer, chained transfer is used to disable RSPI0 and to enable RSPI0.

By setting to n–1 the transfer counter, which is part of the transfer information of the RSPI0 enable transfer, and setting the DTC chain transfer select bit to 1 (chain transfer is performed only when the transfer counter is 0), receive end processing is triggered by chained transfer after the RSPI0 enable transfer at the n–1'th start.

Receive end processing disables chained transfer after the disabling of RSPI0 at the n'th start by overwriting the value of the DTC chain transfer enable bit in the RSPI0 disable transfer information. In this way, RSPI0 is disabled at the n'th start, stopping RSPI0 operation and ending receive operation.

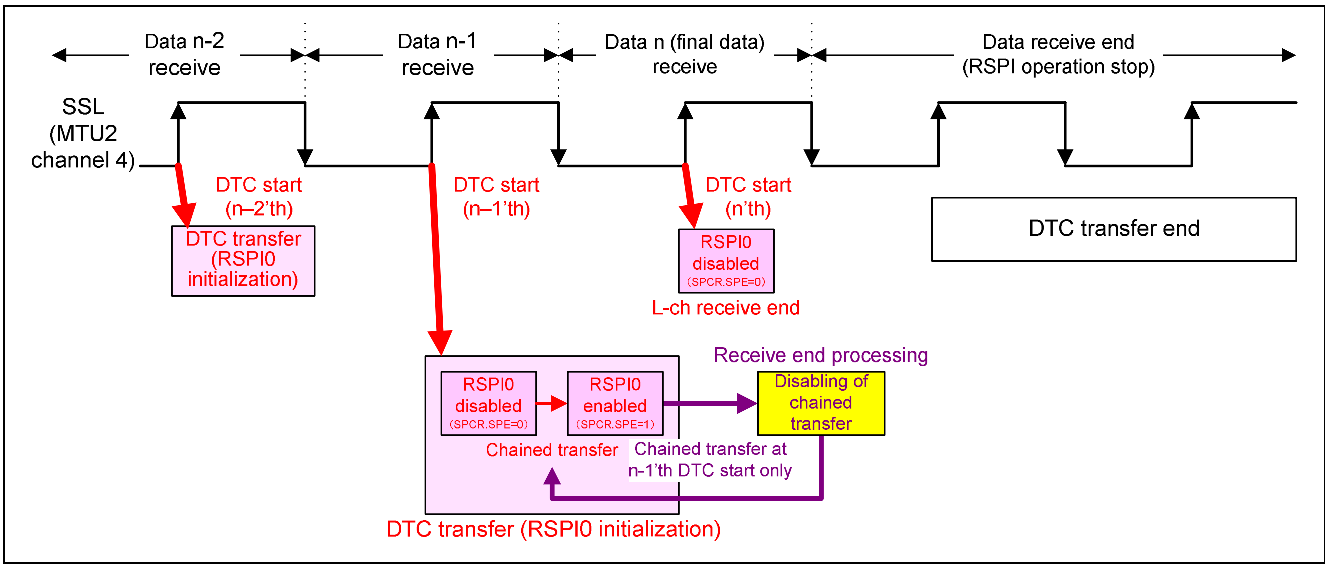

Data n-2 receive
Data n-1 receive
Data n (final data) receive
Data receive end (RSPI operation stop)
SSL (MTU2 channel 4)
DTC start (n–2'th)
DTC start (n–1'th)
DTC start (n'th)
DTC transfer (RSPI0 initialization)
RSPI0 disabled (SPCR.SPE=0)
L-ch receive end
DTC transfer end
Receive end processing
RSPI0 disabled (SPCR.SPE=0)
RSPI0 enabled (SPCR.SPE=1)
Disabling of chained transfer
Chained transfer
Chained transfer at n-1'th DTC start only
DTC transfer (RSPI0 initialization)

Figure 4.26 Receive End Processing by RSPI0 Initialization DTC Transfer

(b) Receive end processing for R-ch receive

Figure 4.27 shows an outline of receive end operation for R-ch receive.

When the n'th RSPI1 initialization DTC transfer starts, chained transfer is used to perform receive end processing. By means of receive end processing, channel RSPI1 is disabled when the n+1'th transfer starts, thereby ending RSPI1 operation.

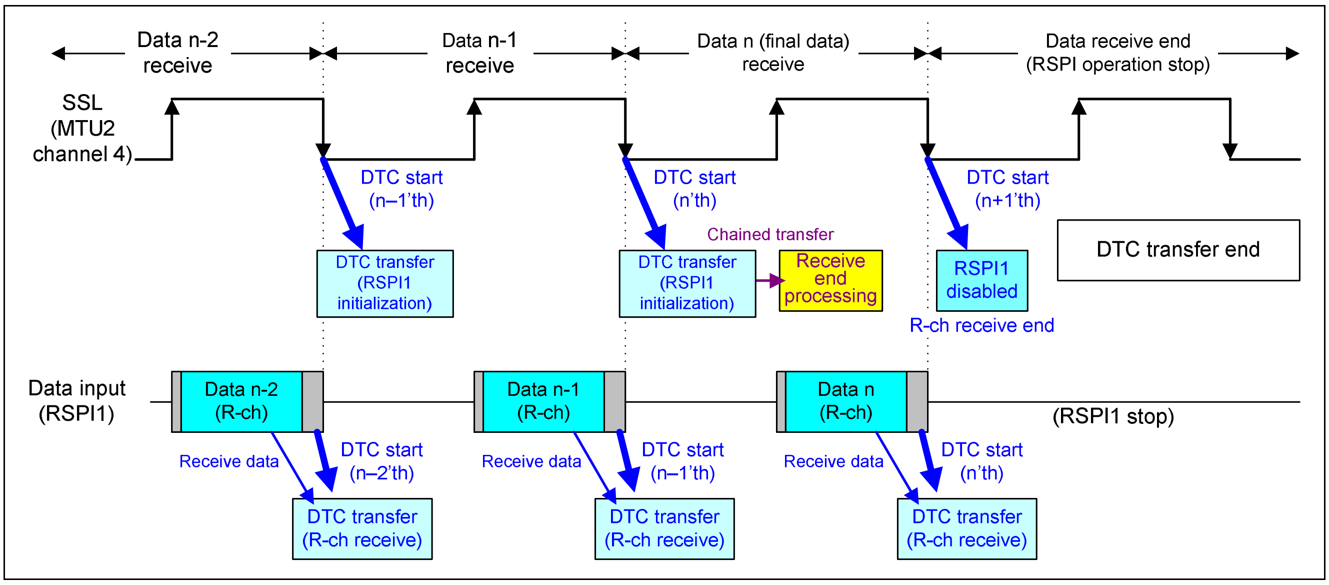


Figure 4.27 Receive End Operation for R-ch Receive

Figure 4.28 illustrates in detail the use of chained transfer to perform receive end processing for R-ch receive.

As shown in figure 4.23, DTC Operation during R-ch Receive, for RSPI1 initialization DTC transfer, chained transfer is used to disable RSPI1 and to enable RSPI1.

By setting to n the transfer counter, which is part of the transfer information of the RSPI1 enable transfer, and setting the DTC chain transfer select bit to 1 (chain transfer is performed only when the transfer counter is 0), receive end processing is triggered by chained transfer after the RSPI1 enable transfer at the n'th start.

Receive end processing disables chained transfer after the disabling of RSPI1 at the n+1'th start by overwriting the value of the DTC chain transfer enable bit in the RSPI1 disable transfer information. In this way, RSPI1 is disabled at the n+1'th start, stopping RSPI1 operation and ending receive operation.

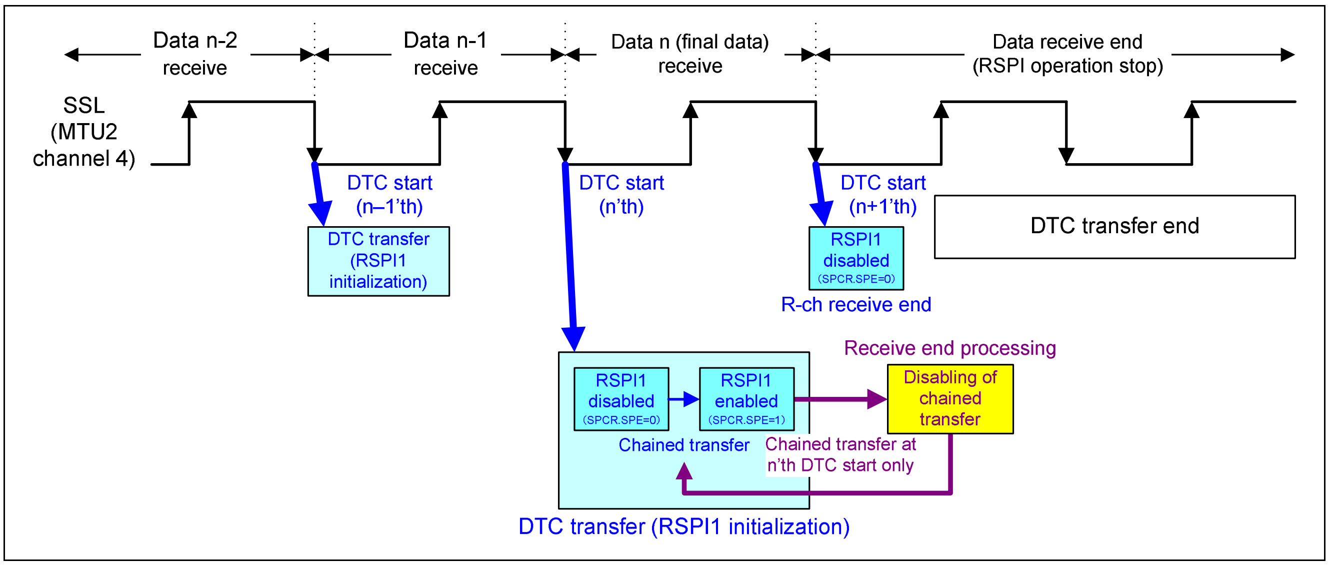

Data n-2 receive
Data n-1 receive
Data n (final data) receive
Data receive end (RSPI operation stop)
SSL (MTU2 channel 4)
DTC start (n–1'th)
DTC start (n'th)
DTC start (n+1'th)
DTC transfer (RSPI1 initialization)
RSPI1 disabled (SPCR.SPE=0)
R-ch receive end
DTC transfer end
Receive end processing
RSPI1 disabled (SPCR.SPE=0)
RSPI1 enabled (SPCR.SPE=1)
Disabling of chained transfer
Chained transfer
Chained transfer at n'th DTC start only
DTC transfer (RSPI1 initialization)

Figure 4.28 Receive End Processing by RSPI1 Initialization DTC Transfer

4.2 File Structure

Table 4.1 lists the files used by the sample program. Note that files that are created automatically by the integrated development environment are omitted.

Table 4.1 File Structure

File Name	Description	Remarks
i2s_main.c	I^2S communication main process	
i2s_dtc.c	DTCa operation settings	
i2s_mtu2.c	MTU2 operation settings	
i2s_rspi.c	RSPI operation settings	
i2s.h	I^2S communication macro definitions	Transmit/receive data format settings
iodefine_RX62N.h	RX62N Series I/O header file	

4.3 List of Constants

Table 4.2 lists the constants used by the sample program.

Table 4.2 Constants Used by Sample Program

Constant	Setting Value	Description
L_CH	0	L-ch index No. of audio data array in on-chip RAM
R_CH	1	R-ch index No. of audio data array in on-chip RAM
NULL_DATA	0x00000000	Initialization data for receive audio data area
TRANSMIT_FORMAT	STANDARD_PADDING or BACKWARD_PADDING or FORWARD_PADDING	Transmit/receive data format selection (selection among standard, backward-padding, and forward-padding)
STANDARD_PADDING	0	Standard format
BACKWARD_PADDING	1	Backward-padding format
FORWARD_PADDING	2	Forward-padding format
TRANSMIT_MODE	0	Transmit operation mode
RECEIVE_MODE	1	Receive operation mode
TRANSCEIVE_MODE	2	Transceive operation mode
AUD_SIZE	0x100	Audio data total byte count
AUD_NUM	(AUD_SIZE / 4) / 2	Audio data (L-ch/R-ch) data count
DTC_TX_CHAIN_SIZE	5	Transfer information count for DTCa chained transfers during transmit operation
DTC_RX_CHAIN_SIZE	2	Transfer information count for DTCa chained transfers during receive operation
DTC_TX_L_COUNT	AUD_NUM – 1	DTC transfer (L-ch transmit, RSPI0 initialization) count
DTC_TX_R_COUNT	AUD_NUM	DTC transfer (R-ch transmit, RSPI1 initialization) count
DTC_RX_L_COUNT	AUD_NUM	DTC transfer (L-ch transmit) count
DTC_RX_R_COUNT	AUD_NUM	DTC transfer (R-ch transmit) count
SCK_CYCLE_VALUE	0x0003	SCK cycle
WS_SSL_CYCLE_VALUE	0x00FF	WS and SSL cycle
SSL_DELAY_VALUE_S	0x0003	WS and SSL phase value (standard format)
SSL_DELAY_VALUE_B	0x0001	WS and SSL phase value (Backward-padding format)
SSL_DELAY_VALUE_F	0x001F	WS and SSL phase value (Forward-padding format)

4.4 List of Structures and Unions

Figure 4.29 lists the structures and unions used by the sample program.

```
#ifdef __LIT
struct st_dtc_data{                               /* Little-endian */
            unsigned char       wk[2];      /* Reserved area */
            unsigned char       MRB;        /* DTC mode register B */
            unsigned char       MRA;        /* DTC mode register A */
            unsigned long       SAR;        /* DTC transfer source address register */
            unsigned long       DAR;        /* DTC transfer destination address register  */
            unsigned short      CRB;        /* DTC transfer count register B */
            unsigned short      CRA;        /* DTC transfer count register A */
};
#endif
#ifdef __BIG
struct st_dtc_data{                               /* Big-endian */
            unsigned char       MRA;        /* DTC mode register A */
            unsigned char       MRB;        /* DTC mode register B */
            unsigned char       wk[2];      /* Reserved area */
            unsigned long       SAR;        /* DTC transfer source address register */
            unsigned long       DAR;        /* DTC transfer destination address register  */
            unsigned short      CRA;        /* DTC transfer count register A */
            unsigned short      CRB;        /* DTC transfer count register B */
};
#endif

struct st_dtc_data DTC_TX_L[DTC_TX_CHAIN_SIZE];       /* DTC transfer information (L-ch transmit) */
struct st_dtc_data DTC_TX_R[DTC_TX_CHAIN_SIZE];       /* DTC transfer information (R-ch transmit) */
struct st_dtc_data DTC_RX_L[DTC_RX_CHAIN_SIZE];       /* DTC transfer information (L-ch receive)*/
struct st_dtc_data DTC_RX_R[DTC_RX_CHAIN_SIZE];       /* DTC transfer information (R-ch receive) */
```

Figure 4.29 Structures and Unions Used by Sample Program

4.5 List of Variables

Table 4.3 lists the global variables and table 4.4 the **const** variables.

Table 4.3 Global Variables

Type	Variable Name	Description	Functions Used By
unsigned long	tx_au_data[AUD_NUM][2]	Transmit audio data	i2s_au_data_init i2s_start i2s_dtc_tx_l_init i2s_dtc_tx_r_init
unsigned long	rx_au_data[AUD_NUM][2]	Receive audio data	i2s_au_data_init i2s_dtc_rx_l_init i2s_dtc_rx_r_init
unsigned long	DTC_VECT_TABLE[256]	DTC vector table	i2s_dtc_init

Table 4.4 *const* Variables

Type	Variable Name	Description	Functions Used By
const unsigned char	RSPI_TX_DISABLE	Register setting values for RSPI initialization (transmit operation)	i2s_dtc_tx_l_init i2s_dtc_tx_r_init
const unsigned char	RSPI_TX_ENABLE	Register setting values for RSPI initialization cancellation (transmit operation)	i2s_dtc_tx_l_init i2s_dtc_tx_r_init
const unsigned char	RSPI_RX_DISABLE	Register setting values for RSPI initialization (receive operation)	i2s_dtc_tx_l_init i2s_dtc_tx_r_init
const unsigned char	RSPI_RX_ENABLE	Register setting values for RSPI initialization cancellation (receive operation)	i2s_dtc_tx_l_init i2s_dtc_tx_r_init
const unsigned char	RSPI_TRX_DISABLE	Register setting values for RSPI initialization (transceive operation)	i2s_dtc_tx_l_init i2s_dtc_tx_r_init
const unsigned char	RSPI_TRX_ENABLE	Register setting values for RSPI initialization cancellation (transceive operation)	i2s_dtc_tx_l_init i2s_dtc_tx_r_init
const unsigned char	DTC_CHAIN_DISABLE	Register setting values for disabling chained transfers	i2s_dtc_tx_l_init i2s_dtc_tx_r_init

4.6 List of Functions

Table 4.5 lists the functions.

Table 4.5 Functions

Function	Description
main	I^2S communication main function
i2s_mcu_init	CPU initial settings
i2s_au_data_init	Audio data creation in on-chip RAM
i2s_start	I^2S communication start process
i2s_dtc_init	DTC setting function
i2s_dtc_tx_l_init	DTC transfer information creation (L-ch transmit, RSPI0 initialization)
i2s_dtc_tx_r_init	DTC transfer information creation (R-ch transmit, RSPI1 initialization)
i2s_dtc_rx_l_init	DTC transfer information creation (L-ch receive)
i2s_dtc_rx_r_init	DTC transfer information creation (R-ch receive)
i2s_mtu2_init	MTU2 setting function
i2s_mtu2_ch2_init	MTU2 channel 2 setting function (SCK generation)
i2s_mtu2_ch3_init	MTU2 channel 3 setting function (WS generation)
i2s_mtu2_ch4_init	MTU2 channel 4 setting function (SSL generation)
i2s_rspi_init	RSPI setting function
i2s_rspi0_init	RSPI0 setting function (L-ch transceive)
i2s_rspi1_init	RSPI1 setting function (R-ch transceive)

4.7 Function Specifications

The specifications of the functions used by the sample program are listed below.

main	
Overview	This is the main function for I^{2}S communication operations.
Header	i2s.h
Declaration	void main(void)
Description	
Arguments	None
Return value	None
Notes	

i2s_mcu_init	
Overview	This function makes initial settings to the CPU.
Header	iodefine_RX62N.h
Declaration	void i2s_mcul_init(void)
Description	• Sets the operating frequencies. • Cancels module stop for the DTC, MTU2, and RSPI.
Arguments	None
Return value	None
Notes	

i2s_au_data_init	
Overview	This function creates audio data in the on-chip RAM.
Header	i2s.h
Declaration	void i2s_data_init(void)
Description	• Creates transmit audio data (tx_au_data). • Clears to 0 the receive audio data area (rx_au_data).
Arguments	None
Return value	None
Notes	

i2s_start	
Overview	This function starts I^{2}S communication.
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_start(char i2s_mode)
Description	• Enables the RSPI function. • Starts the DTC module. • Starts MTU2 count operation
Arguments	1st argument: i2s_mode: Operation mode selection
Return value	None
Notes	

i2s_dtc_init	
Overview	This function makes initial settings to the DTC.
Header	iodefine_RX62N.h
Declaration	void i2s_dtc_init(char i2s_mode)
Description	• Creates the DTC vector table. • Enables DTC activation by interrupt.
Arguments	1st argument: i2s_mode: Operation mode selection
Return value	None
Notes	

i2s_dtc_tx_l_init	
Overview	This function creates DTC transfer information (L-ch transmit, RSPI0 initialization).
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_dtc_tx_l_init(char i2s_mode)
Description	Transfers transmit data from on-chip RAM to RSPI0.
Arguments	1st argument: i2s_mode: Operation mode selection
Return value	None
Notes	

i2s_dtc_tx_r_init	
Overview	This function creates DTC transfer information (R-ch transmit, RSPI1 initialization)
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_dtc_tx_r_init(char i2s_mode)
Description	Transfers transmit data from on-chip RAM to RSPI1.
Arguments	1st argument: i2s_mode: Operation mode selection
Return value	None
Notes	

i2s_dtc_rx_l_init	
Overview	This function creates DTC transfer information (L-ch receive).
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_dtc_rx_l_init(void)
Description	Transfers receive data from RSPI0 to on-chip RAM.
Arguments	None
Return value	None
Notes	

i2s_dtc_rx_r_init	
Overview	This function creates DTC transfer information (R-ch receive).
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_dtc_rx_r_init(void)
Description	Transfers receive data from RSPI1 to on-chip RAM.
Arguments	None
Return value	None
Notes	

i2s_mtu2_init	
Overview	This function makes initial settings to the MTU2.
Header	iodefine_RX62N.h
Declaration	void i2s_mtu2_init(void)
Description	Makes MTU2 port settings.
Arguments	None
Return value	None
Notes	

i2s_mtu2_ch2_init	
Overview	This function makes initial settings to MTU2 channel 2 (SCK generation).
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_mtu2_ch2_init(void)
Description	Generates SCK from the external clock and outputs it.
Arguments	None
Return value	None
Notes	

i2s_mtu2_ch3_init	
Overview	This function makes initial settings to MTU2 channel 3 (WS generation).
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_mtu2_ch3_init(void)
Description	Generates WS from the external clock and outputs it.
Arguments	None
Return value	None
Notes	

i2s_mtu2_ch4_init	
Overview	This function makes initial settings to MTU2 channel 4 (SSL generation).
Header	i2s.h, iodefine_RX62N.h
Declaration	void i2s_mtu2_ch4_init(void)
Description	• Generates SSL from the external clock and outputs it. • Enables compare-match interrupts.
Arguments	None
Return value	None
Notes	

i2s_rspi_init	
Overview	This function makes initial settings to the RSPI.
Header	None
Declaration	void i2s_rspi_init(char i2s_mode)
Description	
Arguments	1st argument: i2s_mode: Operation mode selection
Return value	None
Notes	

i2s_rspi0_init	
Overview	This function makes initial settings to RSPI0 (L-ch transmit/receive).
Header	iodefine_RX62N.h
Declaration	void i2s_rspi0_init(char i2s_mode)
Description	Makes settings to RSPI0.
Arguments	1st argument: i2s_mode: Operation mode selection
Return value	None
Notes	

i2s_rspi1_init	
Overview	This function makes initial settings to RSPI1 (R-ch transmit/receive).
Header	iodefine_RX62N.h
Declaration	void i2s_rspi1_init(char i2s_mode)
Description	Makes settings to RSPI1.
Arguments	1st argument: i2s_mode: Operation mode selection
Return value	None
Notes	

4.8 Flowcharts

4.8.1 main Function

Figure 4.30 is a flowchart of the main function.

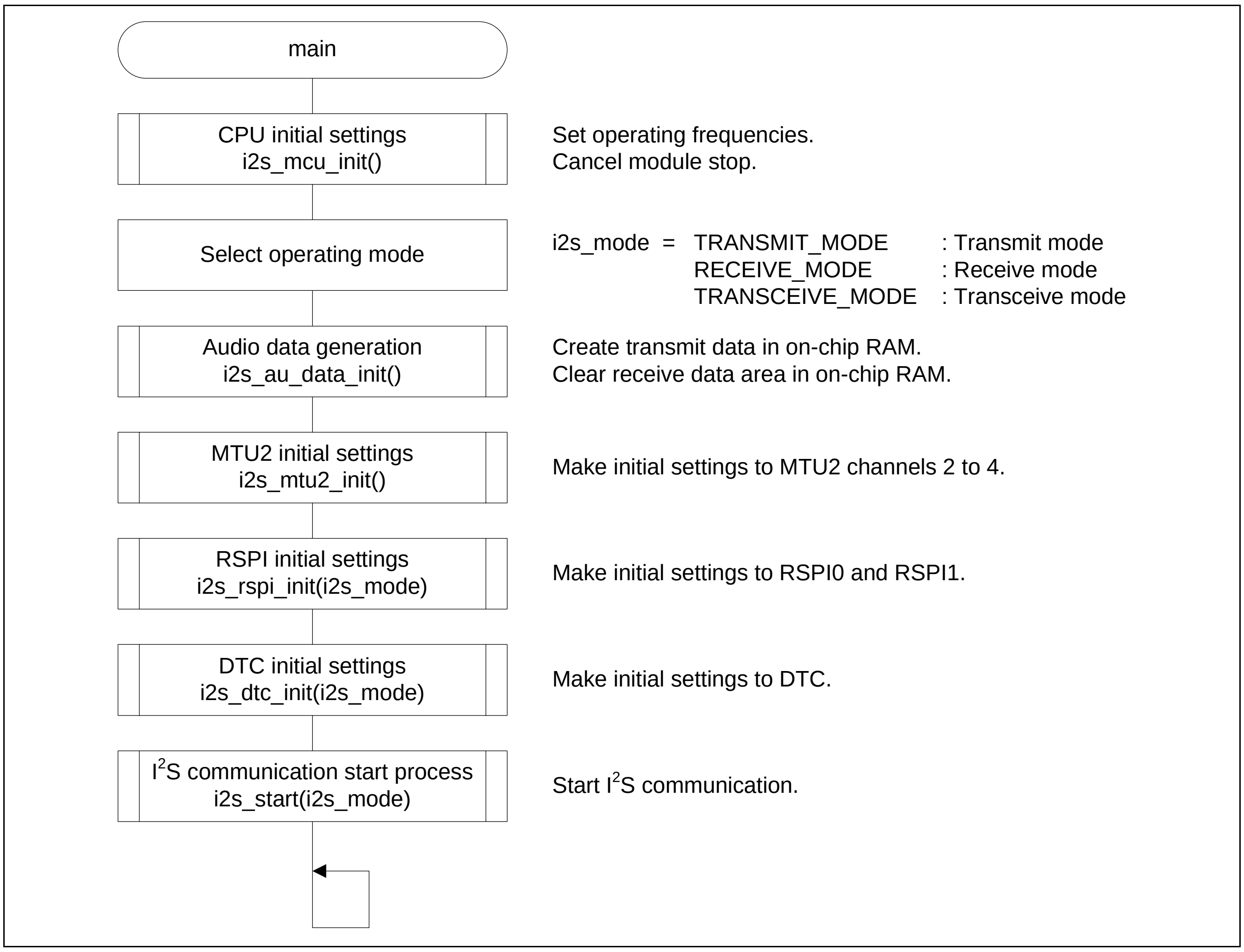

main
CPU initial settings
i2s_mcu_init()
Set operating frequencies.
Cancel module stop.
Select operating mode
i2s_mode = TRANSMIT_MODE : Transmit mode
RECEIVE_MODE : Receive mode
TRANSCEIVE_MODE : Transceive mode
Audio data generation
i2s_au_data_init()
Create transmit data in on-chip RAM.
Clear receive data area in on-chip RAM.
MTU2 initial settings
i2s_mtu2_init()
Make initial settings to MTU2 channels 2 to 4.
RSPI initial settings
i2s_rspi_init(i2s_mode)
Make initial settings to RSPI0 and RSPI1.
DTC initial settings
i2s_dtc_init(i2s_mode)
Make initial settings to DTC.
I2S communication start process
i2s_start(i2s_mode)
Start I2S communication.

Figure 4.30 main Function

4.8.2 i2s_mcu_init Function

Figure 4.31 is a flowchart of the i2s_mcu_init function.

The i2s_mcu_init function makes initial settings to the CPU.

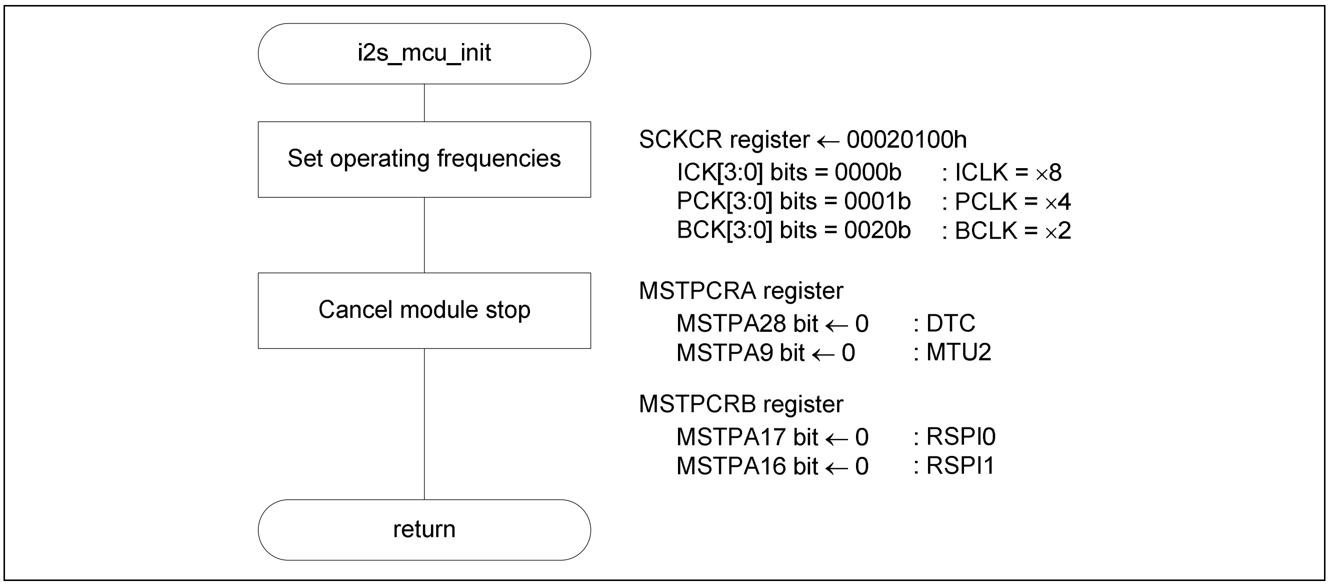


Figure 4.31 i2s_mcu_init Function

4.8.3 i2s_au_data_init Function

Figure 4.32 is a flowchart of the i2s_au_data_init function.

The i2s_au_data_init function creates audio data in the on-chip RAM.

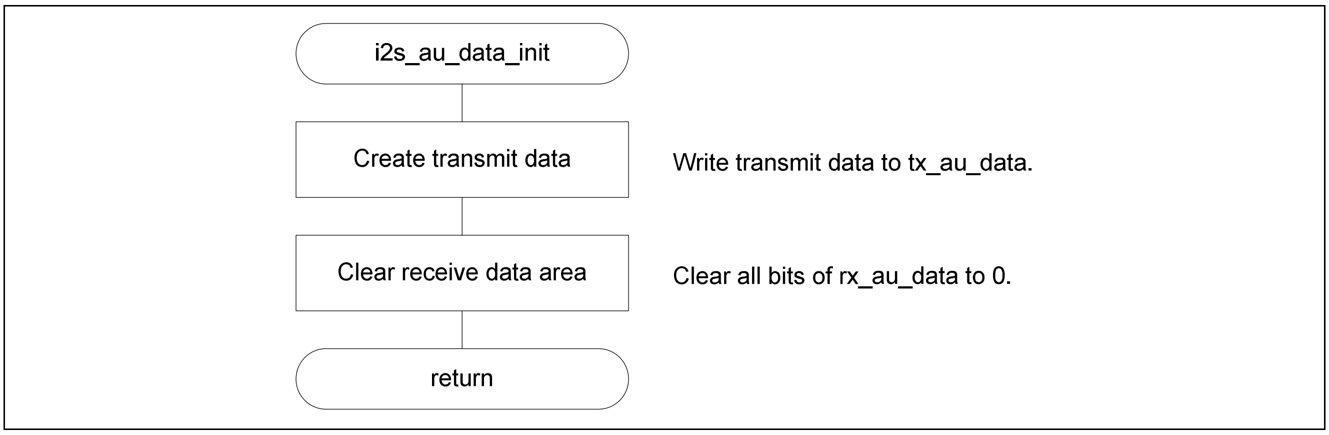
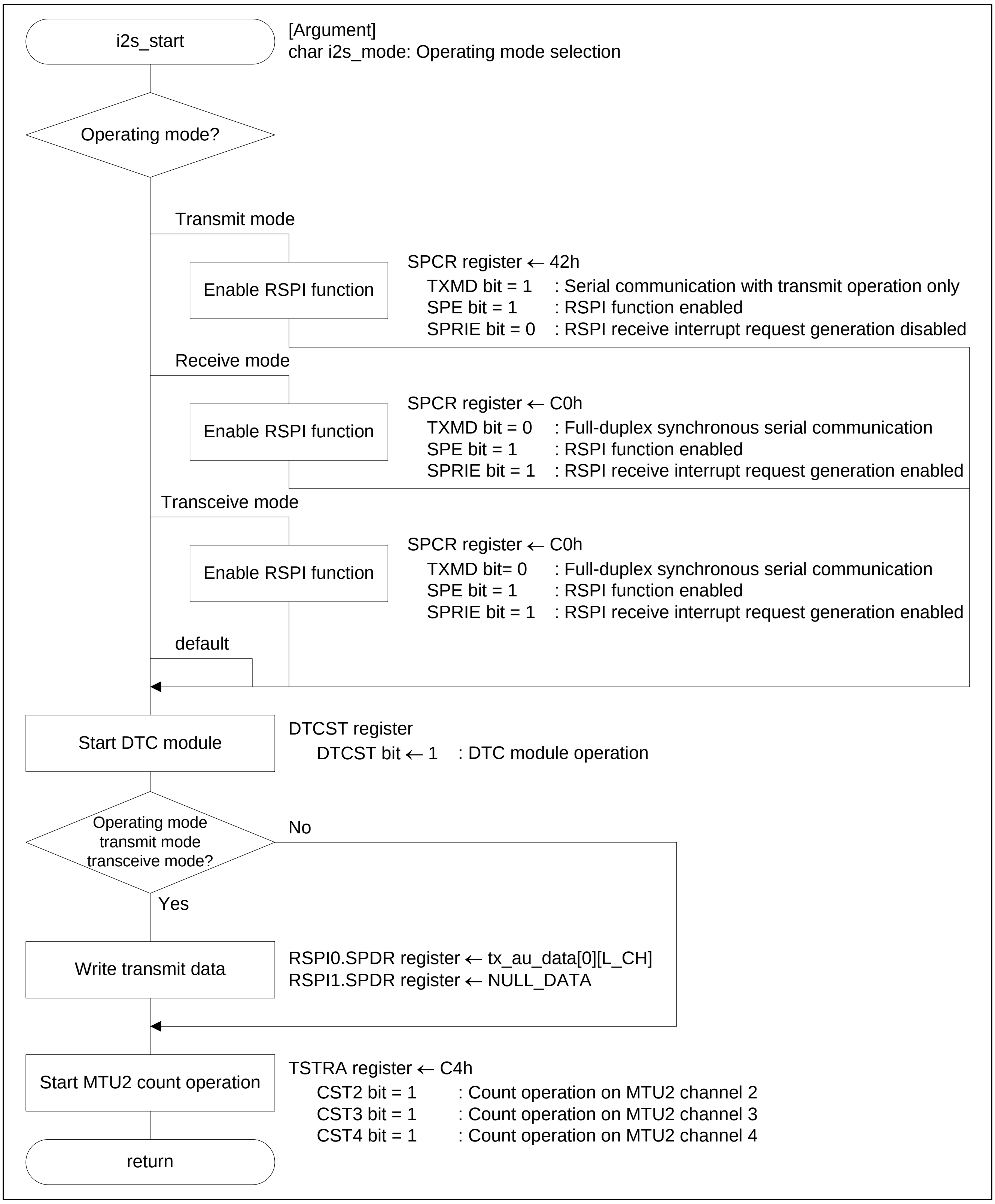

i2s_au_data_init
Create transmit data
Write transmit data to tx_au_data.
Clear receive data area
Clear all bits of rx_au_data to 0.
return

Figure 4.32 i2s_au_data_init Function

4.8.4 i2s_start Function

Figure 4.33 is a flowchart of the i2s_start function.

The i2s_start function enables peripheral functions and starts I^2S communication.



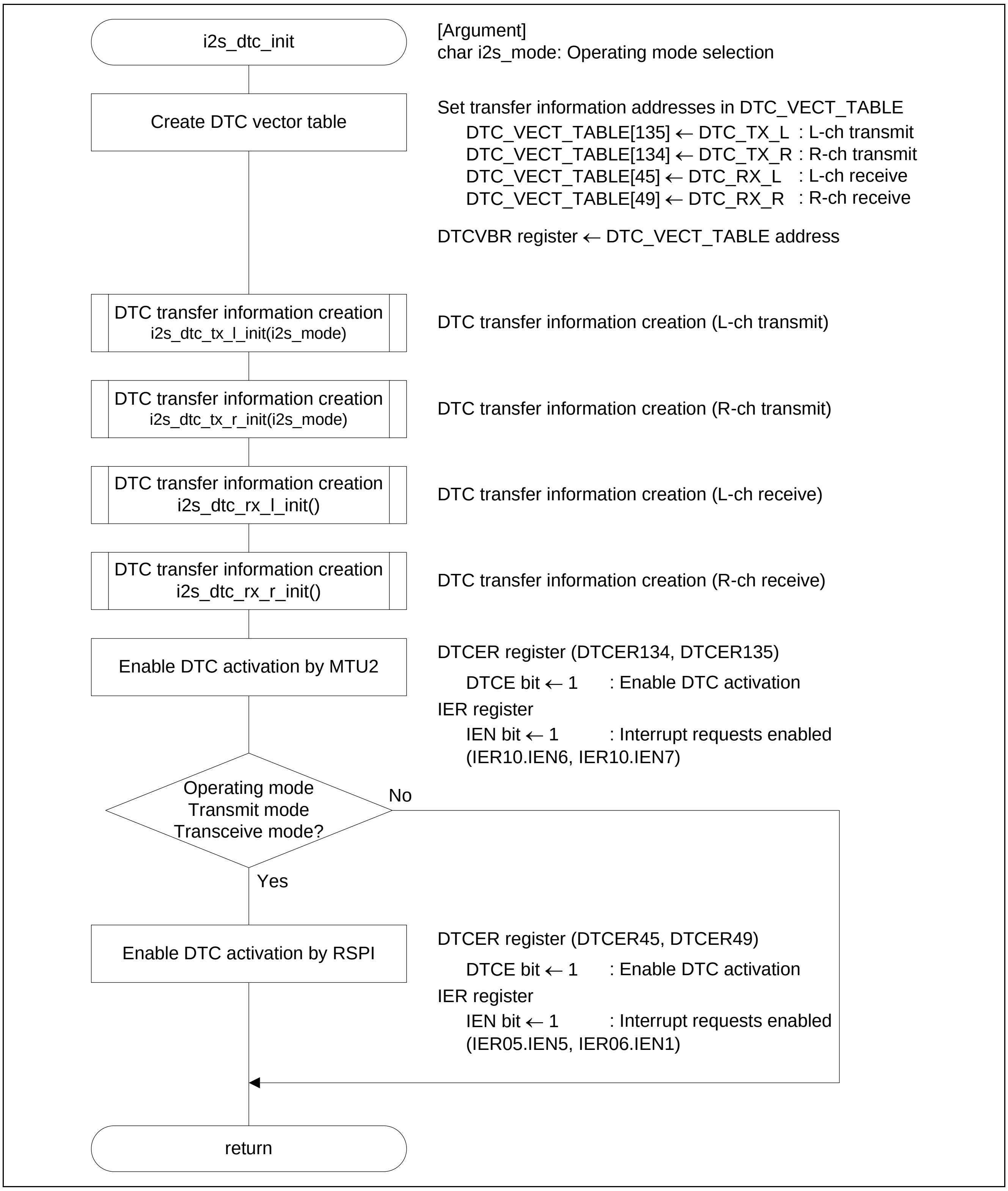
i2s_start
[Argument]
char i2s_mode: Operating mode selection
Operating mode?
Transmit mode
Enable RSPI function
SPCR register ← 42h
TXMD bit = 1 : Serial communication with transmit operation only
SPE bit = 1 : RSPI function enabled
SPRIE bit = 0 : RSPI receive interrupt request generation disabled
Receive mode
Enable RSPI function
SPCR register ← C0h
TXMD bit = 0 : Full-duplex synchronous serial communication
SPE bit = 1 : RSPI function enabled
SPRIE bit = 1 : RSPI receive interrupt request generation enabled
Transceive mode
Enable RSPI function
SPCR register ← C0h
TXMD bit= 0 : Full-duplex synchronous serial communication
SPE bit = 1 : RSPI function enabled
SPRIE bit = 1 : RSPI receive interrupt request generation enabled
default
Start DTC module
DTCST register
DTCST bit ← 1 : DTC module operation
Operating mode transmit mode transceive mode?
No
Yes
Write transmit data
RSPI0.SPDR register ← tx_au_data[0][L_CH]
RSPI1.SPDR register ← NULL_DATA
Start MTU2 count operation
TSTRA register ← C4h
CST2 bit = 1 : Count operation on MTU2 channel 2
CST3 bit = 1 : Count operation on MTU2 channel 3
CST4 bit = 1 : Count operation on MTU2 channel 4
return

Figure 4.33 i2s_start Function

4.8.5 i2s_dtc_init Function

Figure 4.34 is a flowchart of the i2s_dtc_init function.

The i2s_dtc_init function makes initial settings to the DTCa.



i2s_dtc_init
[Argument]
char i2s_mode: Operating mode selection
Create DTC vector table
Set transfer information addresses in DTC_VECT_TABLE
DTC_VECT_TABLE[135] ← DTC_TX_L : L-ch transmit
DTC_VECT_TABLE[134] ← DTC_TX_R : R-ch transmit
DTC_VECT_TABLE[45] ← DTC_RX_L : L-ch receive
DTC_VECT_TABLE[49] ← DTC_RX_R : R-ch receive
DTCVBR register ← DTC_VECT_TABLE address
DTC transfer information creation i2s_dtc_tx_l_init(i2s_mode)
DTC transfer information creation (L-ch transmit)
DTC transfer information creation i2s_dtc_tx_r_init(i2s_mode)
DTC transfer information creation (R-ch transmit)
DTC transfer information creation i2s_dtc_rx_l_init()
DTC transfer information creation (L-ch receive)
DTC transfer information creation i2s_dtc_rx_r_init()
DTC transfer information creation (R-ch receive)
Enable DTC activation by MTU2
DTCER register (DTCER134, DTCER135)
DTCE bit ← 1 : Enable DTC activation
IER register
IEN bit ← 1 : Interrupt requests enabled
(IER10.IEN6, IER10.IEN7)
Operating mode Transmit mode Transceive mode?
No
Yes
Enable DTC activation by RSPI
DTCER register (DTCER45, DTCER49)
DTCE bit ← 1 : Enable DTC activation
IER register
IEN bit ← 1 : Interrupt requests enabled
(IER05.IEN5, IER06.IEN1)
return

Figure 4.34 i2s_dtc_init Function

4.8.6 i2s_dtc_tx_l_init Function

Figure 4.35 is a flowchart of the i2s_dtc_tx_l_init function.

The i2s_dtc_tx_l_init function creates DTC transfer information for L-ch transmit (transmit mode or transceive mode) or RSPI0 initialization (receive mode).

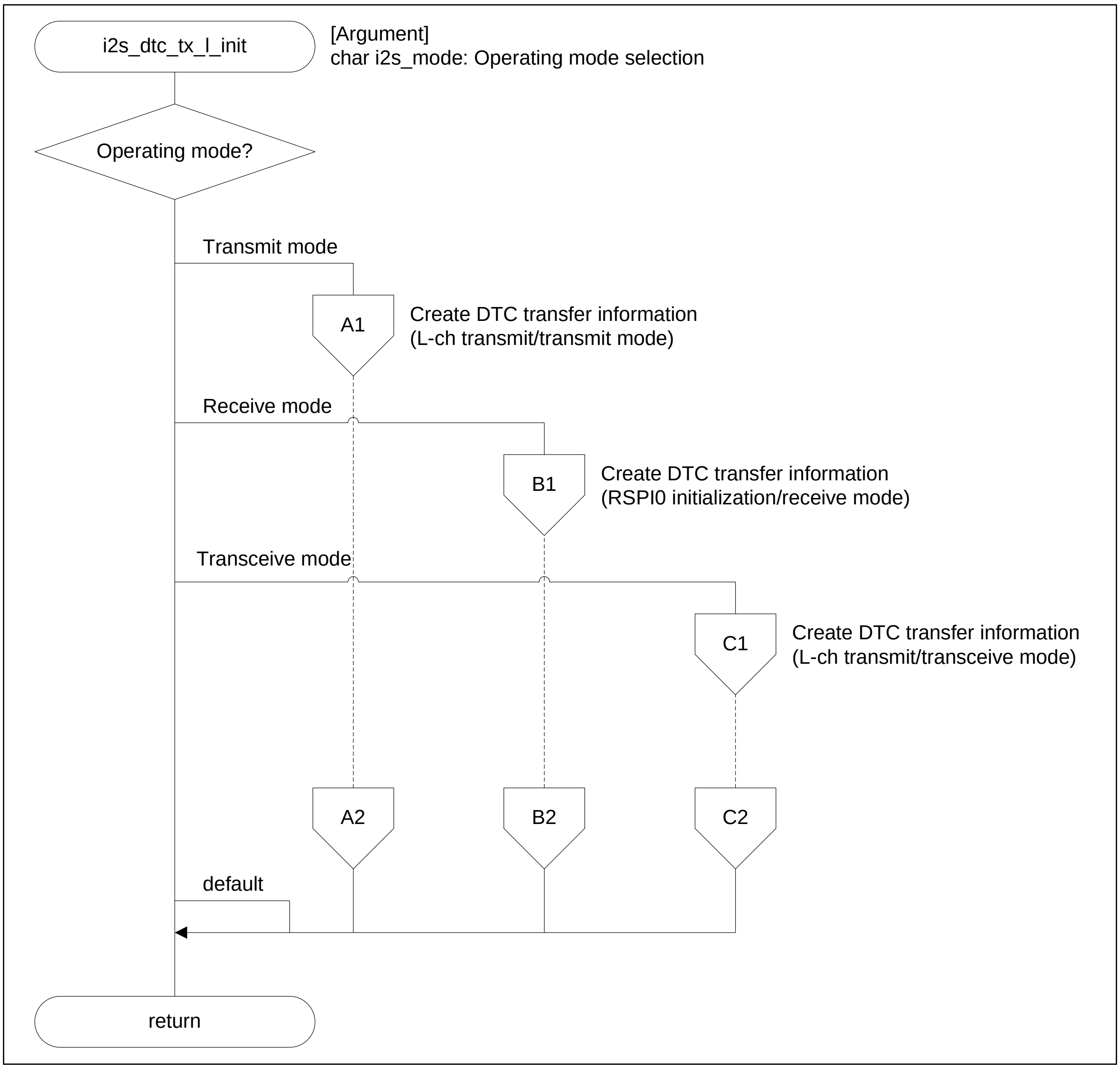
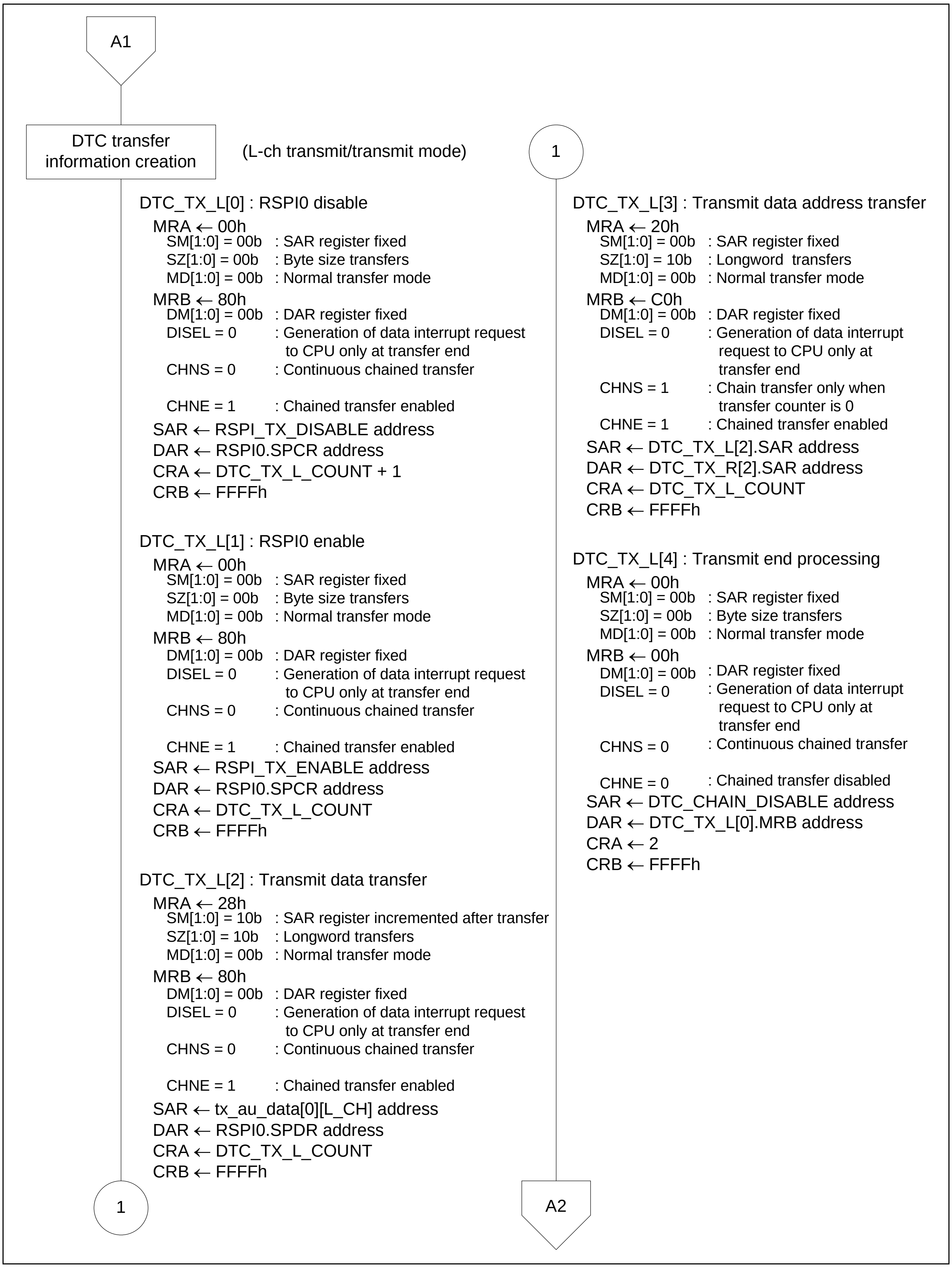

i2s_dtc_tx_l_init
[Argument]
char i2s_mode: Operating mode selection
Operating mode?
Transmit mode
A1
Create DTC transfer information
(L-ch transmit/transmit mode)
Receive mode
B1
Create DTC transfer information
(RSPI0 initialization/receive mode)
Transceive mode
C1
Create DTC transfer information
(L-ch transmit/transceive mode)
A2
B2
C2
default
return

Figure 4.35 i2s_dtc_tx_l_init Function (1/4)


A1
DTC transfer information creation
(L-ch transmit/transmit mode)
1
1
A2

DTC_TX_L[0] : RSPI0 disable
MRA ← 00h
SM[1:0] = 00b : SAR register fixed
SZ[1:0] = 00b : Byte size transfers
MD[1:0] = 00b : Normal transfer mode
MRB ← 80h
DM[1:0] = 00b : DAR register fixed
DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
CHNS = 0 : Continuous chained transfer
CHNE = 1 : Chained transfer enabled
SAR ← RSPI_TX_DISABLE address
DAR ← RSPI0.SPCR address
CRA ← DTC_TX_L_COUNT + 1
CRB ← FFFFh

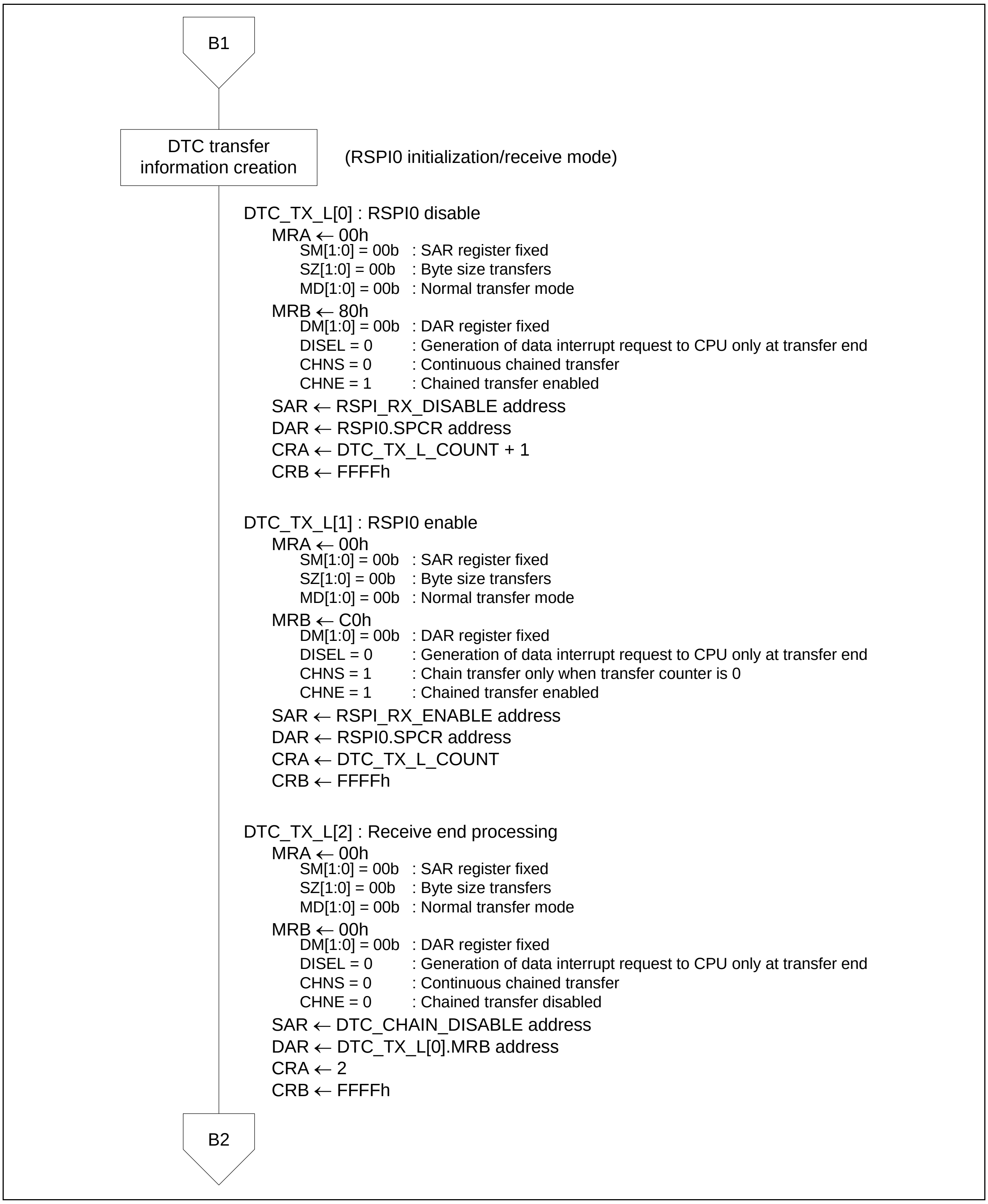
DTC_TX_L[1] : RSPI0 enable
MRA ← 00h
SM[1:0] = 00b : SAR register fixed
SZ[1:0] = 00b : Byte size transfers
MD[1:0] = 00b : Normal transfer mode
MRB ← 80h
DM[1:0] = 00b : DAR register fixed
DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
CHNS = 0 : Continuous chained transfer
CHNE = 1 : Chained transfer enabled
SAR ← RSPI_TX_ENABLE address
DAR ← RSPI0.SPCR address
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[2] : Transmit data transfer
MRA ← 28h
SM[1:0] = 10b : SAR register incremented after transfer
SZ[1:0] = 10b : Longword transfers
MD[1:0] = 00b : Normal transfer mode
MRB ← 80h
DM[1:0] = 00b : DAR register fixed
DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
CHNS = 0 : Continuous chained transfer
CHNE = 1 : Chained transfer enabled
SAR ← tx_au_data[0][L_CH] address
DAR ← RSPI0.SPDR address
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[3] : Transmit data address transfer
MRA ← 20h
SM[1:0] = 00b : SAR register fixed
SZ[1:0] = 10b : Longword transfers
MD[1:0] = 00b : Normal transfer mode
MRB ← C0h
DM[1:0] = 00b : DAR register fixed
DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
CHNS = 1 : Chain transfer only when transfer counter is 0
CHNE = 1 : Chained transfer enabled
SAR ← DTC_TX_L[2].SAR address
DAR ← DTC_TX_R[2].SAR address
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[4] : Transmit end processing
MRA ← 00h
SM[1:0] = 00b : SAR register fixed
SZ[1:0] = 00b : Byte size transfers
MD[1:0] = 00b : Normal transfer mode
MRB ← 00h
DM[1:0] = 00b : DAR register fixed
DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
CHNS = 0 : Continuous chained transfer
CHNE = 0 : Chained transfer disabled
SAR ← DTC_CHAIN_DISABLE address
DAR ← DTC_TX_L[0].MRB address
CRA ← 2
CRB ← FFFFh

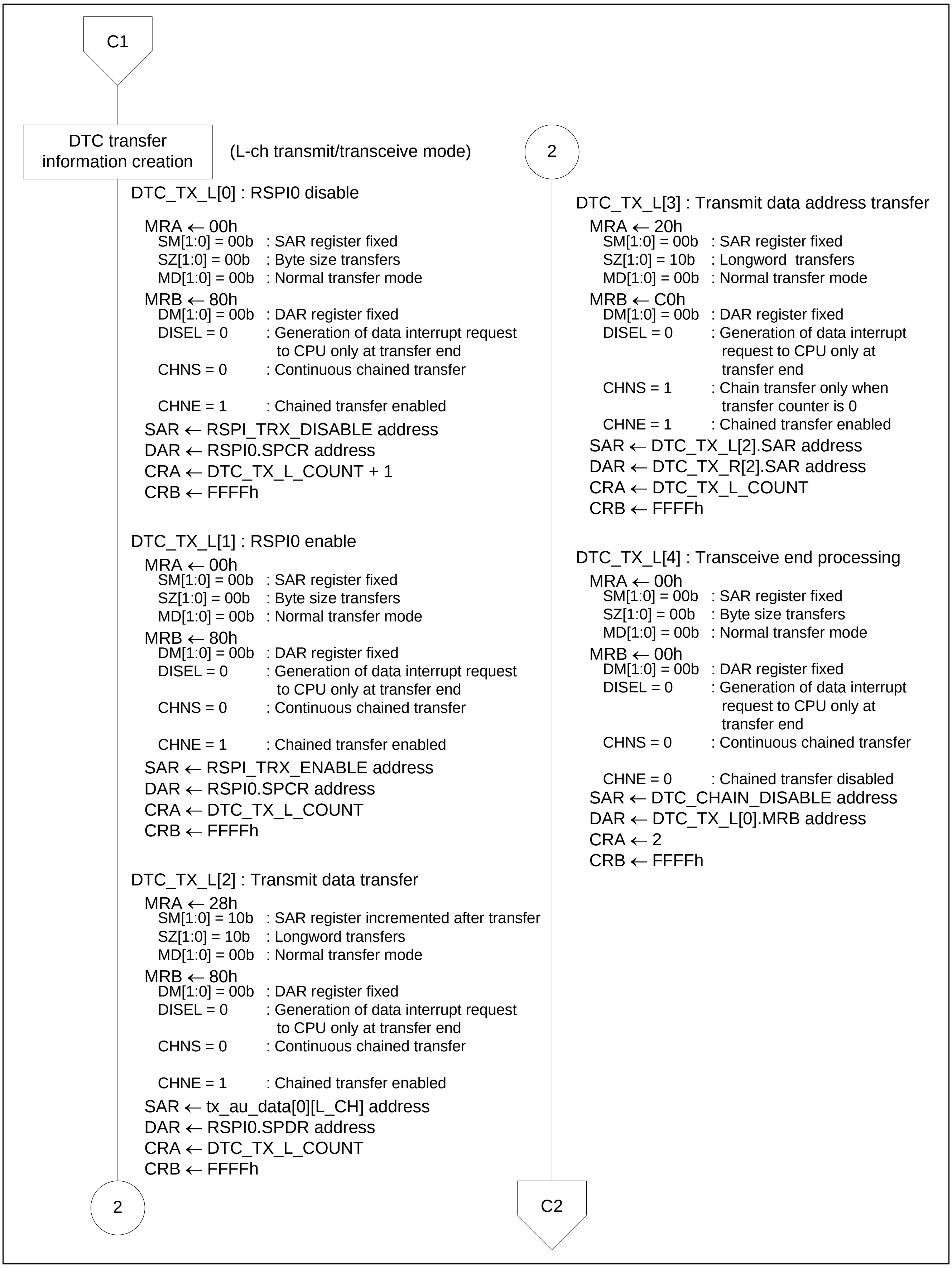
Figure 4.35 i2s_dtc_tx_l_init Function (2/4)



(RSPI0 initialization/receive mode)

```
DTC_TX_L[0] : RSPI0 disable
    MRA ← 00h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 00b   : Byte size transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← 80h
        DM[1:0] = 00b   : DAR register fixed
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 0        : Continuous chained transfer
        CHNE = 1        : Chained transfer enabled
    SAR ← RSPI_RX_DISABLE address
    DAR ← RSPI0.SPCR address
    CRA ← DTC_TX_L_COUNT + 1
    CRB ← FFFFh

DTC_TX_L[1] : RSPI0 enable
    MRA ← 00h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 00b   : Byte size transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← C0h
        DM[1:0] = 00b   : DAR register fixed
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 1        : Chain transfer only when transfer counter is 0
        CHNE = 1        : Chained transfer enabled
    SAR ← RSPI_RX_ENABLE address
    DAR ← RSPI0.SPCR address
    CRA ← DTC_TX_L_COUNT
    CRB ← FFFFh

DTC_TX_L[2] : Receive end processing
    MRA ← 00h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 00b   : Byte size transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← 00h
        DM[1:0] = 00b   : DAR register fixed
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 0        : Continuous chained transfer
        CHNE = 0        : Chained transfer disabled
    SAR ← DTC_CHAIN_DISABLE address
    DAR ← DTC_TX_L[0].MRB address
    CRA ← 2
    CRB ← FFFFh
```

Figure 4.35 i2s_dtc_tx_l_init Function (3/4)


C1
DTC transfer information creation
(L-ch transmit/transceive mode)
2
2
C2

DTC_TX_L[0] : RSPI0 disable

MRA ← 00h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 00b : Byte size transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 80h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 1 : Chained transfer enabled

SAR ← RSPI_TRX_DISABLE address
DAR ← RSPI0.SPCR address
CRA ← DTC_TX_L_COUNT + 1
CRB ← FFFFh

DTC_TX_L[1] : RSPI0 enable

MRA ← 00h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 00b : Byte size transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 80h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 1 : Chained transfer enabled

SAR ← RSPI_TRX_ENABLE address
DAR ← RSPI0.SPCR address
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[2] : Transmit data transfer

MRA ← 28h
- SM[1:0] = 10b : SAR register incremented after transfer
- SZ[1:0] = 10b : Longword transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 80h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 1 : Chained transfer enabled

SAR ← tx_au_data[0][L_CH] address
DAR ← RSPI0.SPDR address
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[3] : Transmit data address transfer

MRA ← 20h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 10b : Longword transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← C0h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 1 : Chain transfer only when transfer counter is 0
- CHNE = 1 : Chained transfer enabled

SAR ← DTC_TX_L[2].SAR address
DAR ← DTC_TX_R[2].SAR address
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[4] : Transceive end processing

MRA ← 00h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 00b : Byte size transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 00h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 0 : Chained transfer disabled

SAR ← DTC_CHAIN_DISABLE address
DAR ← DTC_TX_L[0].MRB address
CRA ← 2
CRB ← FFFFh

Figure 4.35 i2s_dtc_tx_l_init Function (4/4)

4.8.7 i2s_dtc_tx_r_init Function

Figure 4.36 is a flowchart of the i2s_dtc_tx_r_init function.

The i2s_dtc_tx_r_init function creates DTC transfer information for R-ch transmit (transmit mode or transceive mode) or RSPI1 initialization (receive mode).

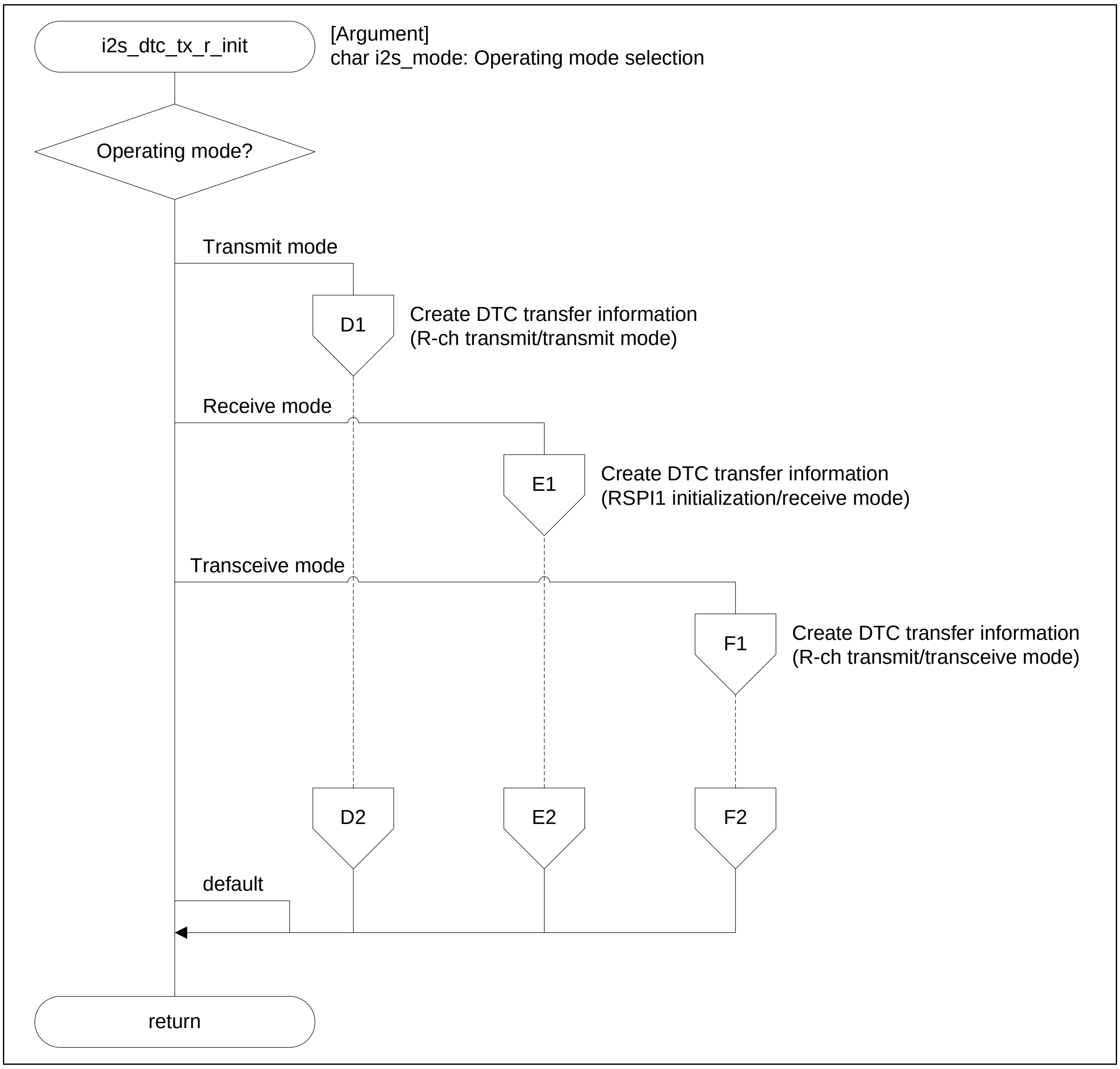
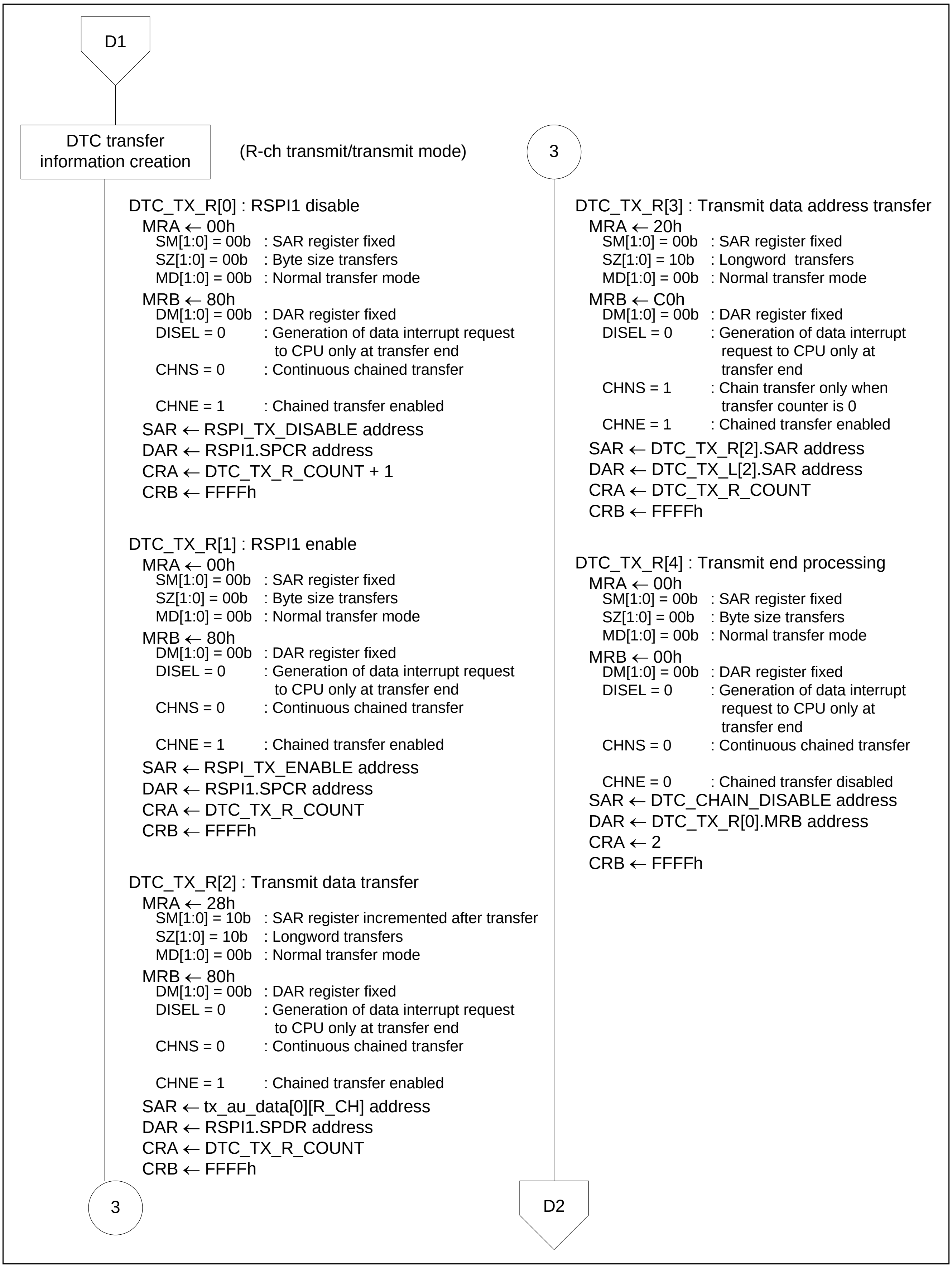
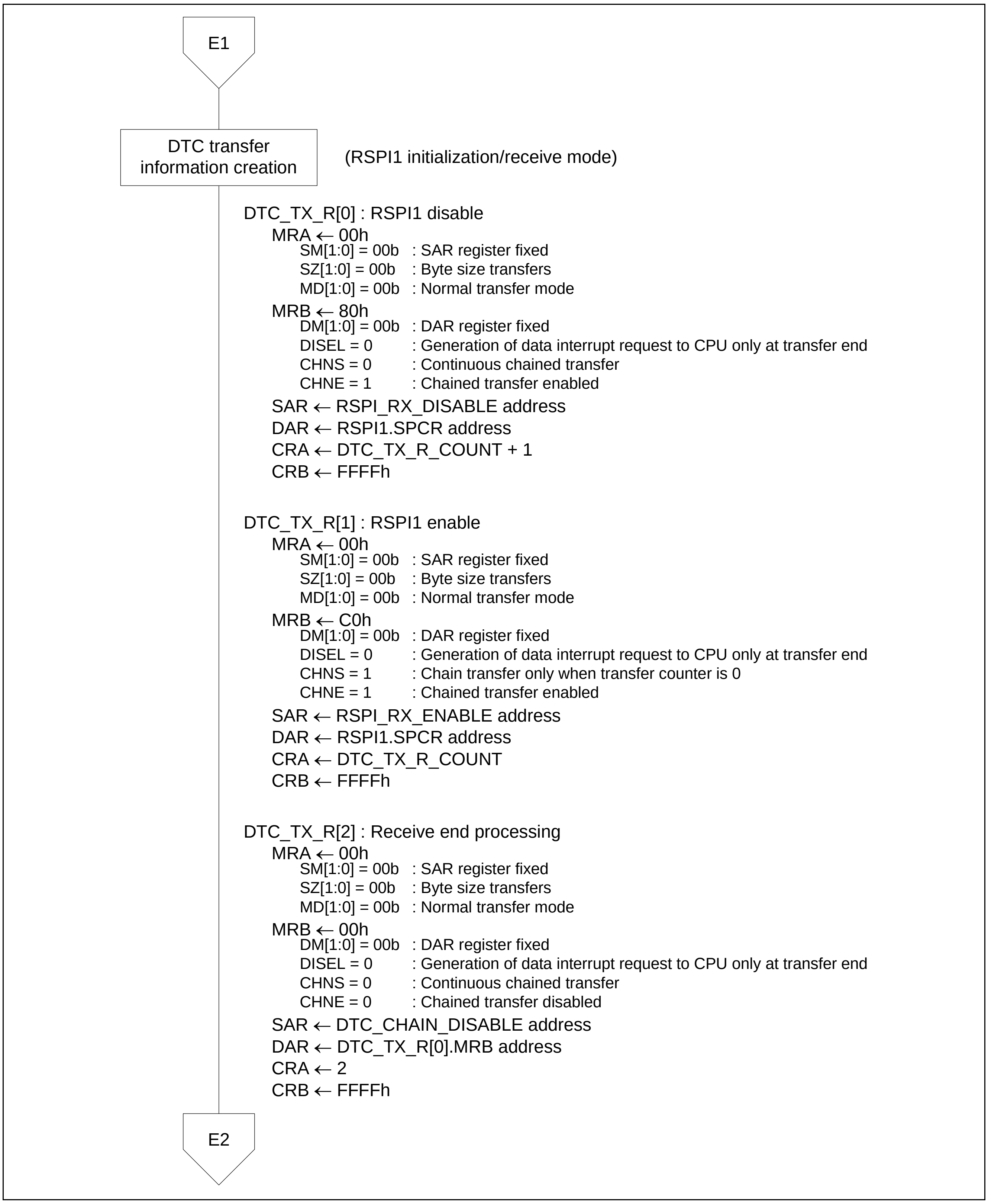

i2s_dtc_tx_r_init
[Argument]
char i2s_mode: Operating mode selection
Operating mode?
Transmit mode
D1
Create DTC transfer information
(R-ch transmit/transmit mode)
Receive mode
E1
Create DTC transfer information
(RSPI1 initialization/receive mode)
Transceive mode
F1
Create DTC transfer information
(R-ch transmit/transceive mode)
D2
E2
F2
default
return

Figure 4.36 i2s_dtc_tx_r_init Function (1/4)


D1
DTC transfer information creation
(R-ch transmit/transmit mode)
3
3
D2

```
DTC_TX_R[0] : RSPI1 disable
  MRA ← 00h
    SM[1:0] = 00b  : SAR register fixed
    SZ[1:0] = 00b  : Byte size transfers
    MD[1:0] = 00b  : Normal transfer mode
  MRB ← 80h
    DM[1:0] = 00b  : DAR register fixed
    DISEL = 0      : Generation of data interrupt request
                     to CPU only at transfer end
    CHNS = 0       : Continuous chained transfer

    CHNE = 1       : Chained transfer enabled
  SAR ← RSPI_TX_DISABLE address
  DAR ← RSPI1.SPCR address
  CRA ← DTC_TX_R_COUNT + 1
  CRB ← FFFFh

DTC_TX_R[1] : RSPI1 enable
  MRA ← 00h
    SM[1:0] = 00b  : SAR register fixed
    SZ[1:0] = 00b  : Byte size transfers
    MD[1:0] = 00b  : Normal transfer mode
  MRB ← 80h
    DM[1:0] = 00b  : DAR register fixed
    DISEL = 0      : Generation of data interrupt request
                     to CPU only at transfer end
    CHNS = 0       : Continuous chained transfer

    CHNE = 1       : Chained transfer enabled
  SAR ← RSPI_TX_ENABLE address
  DAR ← RSPI1.SPCR address
  CRA ← DTC_TX_R_COUNT
  CRB ← FFFFh

DTC_TX_R[2] : Transmit data transfer
  MRA ← 28h
    SM[1:0] = 10b  : SAR register incremented after transfer
    SZ[1:0] = 10b  : Longword transfers
    MD[1:0] = 00b  : Normal transfer mode
  MRB ← 80h
    DM[1:0] = 00b  : DAR register fixed
    DISEL = 0      : Generation of data interrupt request
                     to CPU only at transfer end
    CHNS = 0       : Continuous chained transfer

    CHNE = 1       : Chained transfer enabled
  SAR ← tx_au_data[0][R_CH] address
  DAR ← RSPI1.SPDR address
  CRA ← DTC_TX_R_COUNT
  CRB ← FFFFh

DTC_TX_R[3] : Transmit data address transfer
  MRA ← 20h
    SM[1:0] = 00b  : SAR register fixed
    SZ[1:0] = 10b  : Longword  transfers
    MD[1:0] = 00b  : Normal transfer mode
  MRB ← C0h
    DM[1:0] = 00b  : DAR register fixed
    DISEL = 0      : Generation of data interrupt
                     request to CPU only at
                     transfer end
    CHNS = 1       : Chain transfer only when
                     transfer counter is 0
    CHNE = 1       : Chained transfer enabled
  SAR ← DTC_TX_R[2].SAR address
  DAR ← DTC_TX_L[2].SAR address
  CRA ← DTC_TX_R_COUNT
  CRB ← FFFFh

DTC_TX_R[4] : Transmit end processing
  MRA ← 00h
    SM[1:0] = 00b  : SAR register fixed
    SZ[1:0] = 00b  : Byte size transfers
    MD[1:0] = 00b  : Normal transfer mode
  MRB ← 00h
    DM[1:0] = 00b  : DAR register fixed
    DISEL = 0      : Generation of data interrupt
                     request to CPU only at
                     transfer end
    CHNS = 0       : Continuous chained transfer

    CHNE = 0       : Chained transfer disabled
  SAR ← DTC_CHAIN_DISABLE address
  DAR ← DTC_TX_R[0].MRB address
  CRA ← 2
  CRB ← FFFFh
```

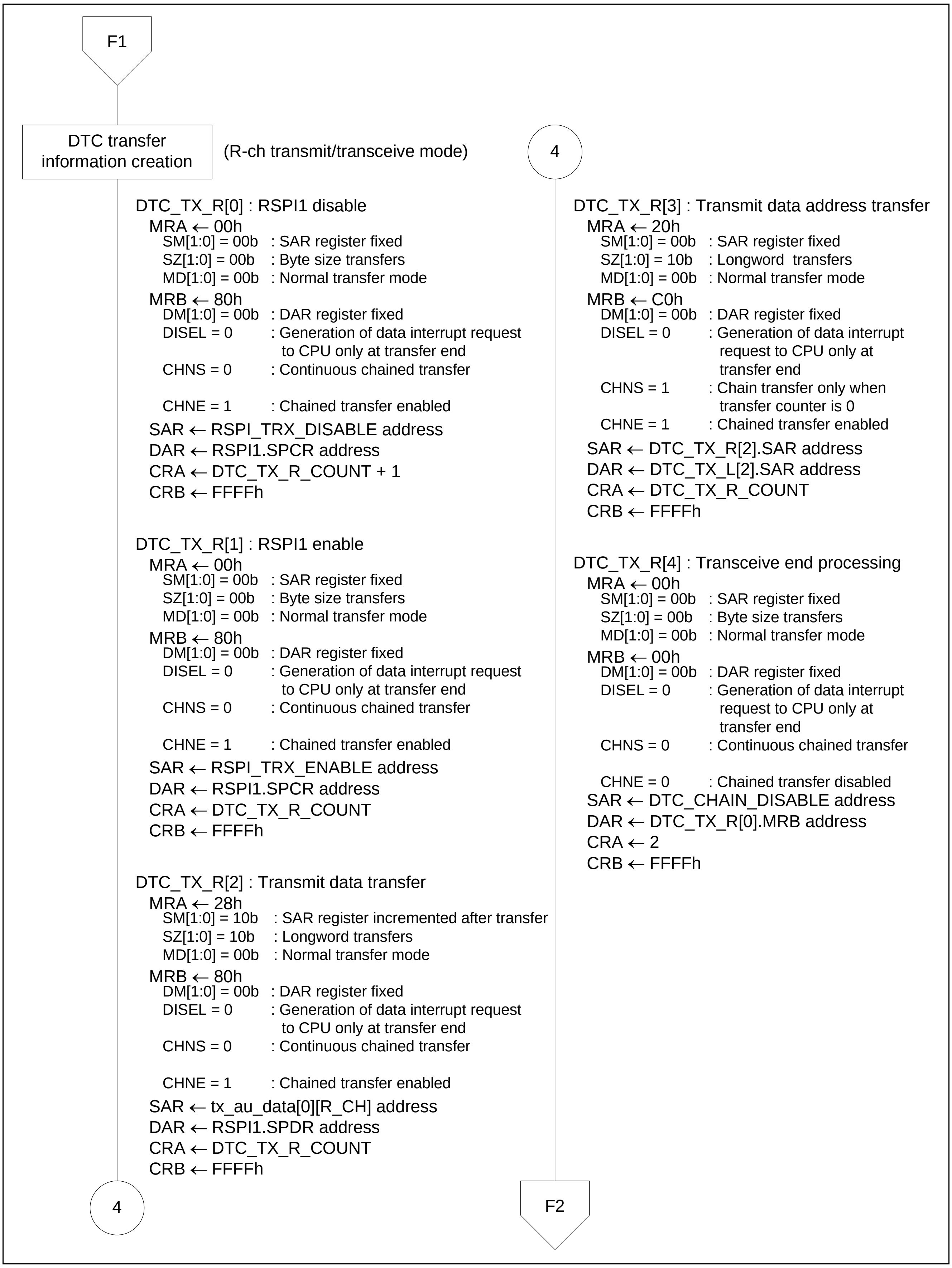
Figure 4.36 i2s_dtc_tx_r_init Function (2/4)



(RSPI1 initialization/receive mode)

```
DTC_TX_R[0] : RSPI1 disable
    MRA ← 00h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 00b   : Byte size transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← 80h
        DM[1:0] = 00b   : DAR register fixed
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 0        : Continuous chained transfer
        CHNE = 1        : Chained transfer enabled
    SAR ← RSPI_RX_DISABLE address
    DAR ← RSPI1.SPCR address
    CRA ← DTC_TX_R_COUNT + 1
    CRB ← FFFFh

DTC_TX_R[1] : RSPI1 enable
    MRA ← 00h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 00b   : Byte size transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← C0h
        DM[1:0] = 00b   : DAR register fixed
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 1        : Chain transfer only when transfer counter is 0
        CHNE = 1        : Chained transfer enabled
    SAR ← RSPI_RX_ENABLE address
    DAR ← RSPI1.SPCR address
    CRA ← DTC_TX_R_COUNT
    CRB ← FFFFh

DTC_TX_R[2] : Receive end processing
    MRA ← 00h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 00b   : Byte size transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← 00h
        DM[1:0] = 00b   : DAR register fixed
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 0        : Continuous chained transfer
        CHNE = 0        : Chained transfer disabled
    SAR ← DTC_CHAIN_DISABLE address
    DAR ← DTC_TX_R[0].MRB address
    CRA ← 2
    CRB ← FFFFh
```

Figure 4.36 i2s_dtc_tx_r_init Function (3/4)


F1
DTC transfer information creation
(R-ch transmit/transceive mode)
4
4
F2

DTC_TX_R[0] : RSPI1 disable
MRA ← 00h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 00b : Byte size transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 80h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 1 : Chained transfer enabled

SAR ← RSPI_TRX_DISABLE address
DAR ← RSPI1.SPCR address
CRA ← DTC_TX_R_COUNT + 1
CRB ← FFFFh

DTC_TX_R[1] : RSPI1 enable
MRA ← 00h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 00b : Byte size transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 80h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 1 : Chained transfer enabled

SAR ← RSPI_TRX_ENABLE address
DAR ← RSPI1.SPCR address
CRA ← DTC_TX_R_COUNT
CRB ← FFFFh

DTC_TX_R[2] : Transmit data transfer
MRA ← 28h
- SM[1:0] = 10b : SAR register incremented after transfer
- SZ[1:0] = 10b : Longword transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 80h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 1 : Chained transfer enabled

SAR ← tx_au_data[0][R_CH] address
DAR ← RSPI1.SPDR address
CRA ← DTC_TX_R_COUNT
CRB ← FFFFh

DTC_TX_R[3] : Transmit data address transfer
MRA ← 20h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 10b : Longword transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← C0h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 1 : Chain transfer only when transfer counter is 0
- CHNE = 1 : Chained transfer enabled

SAR ← DTC_TX_R[2].SAR address
DAR ← DTC_TX_L[2].SAR address
CRA ← DTC_TX_R_COUNT
CRB ← FFFFh

DTC_TX_R[4] : Transceive end processing
MRA ← 00h
- SM[1:0] = 00b : SAR register fixed
- SZ[1:0] = 00b : Byte size transfers
- MD[1:0] = 00b : Normal transfer mode

MRB ← 00h
- DM[1:0] = 00b : DAR register fixed
- DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
- CHNS = 0 : Continuous chained transfer
- CHNE = 0 : Chained transfer disabled

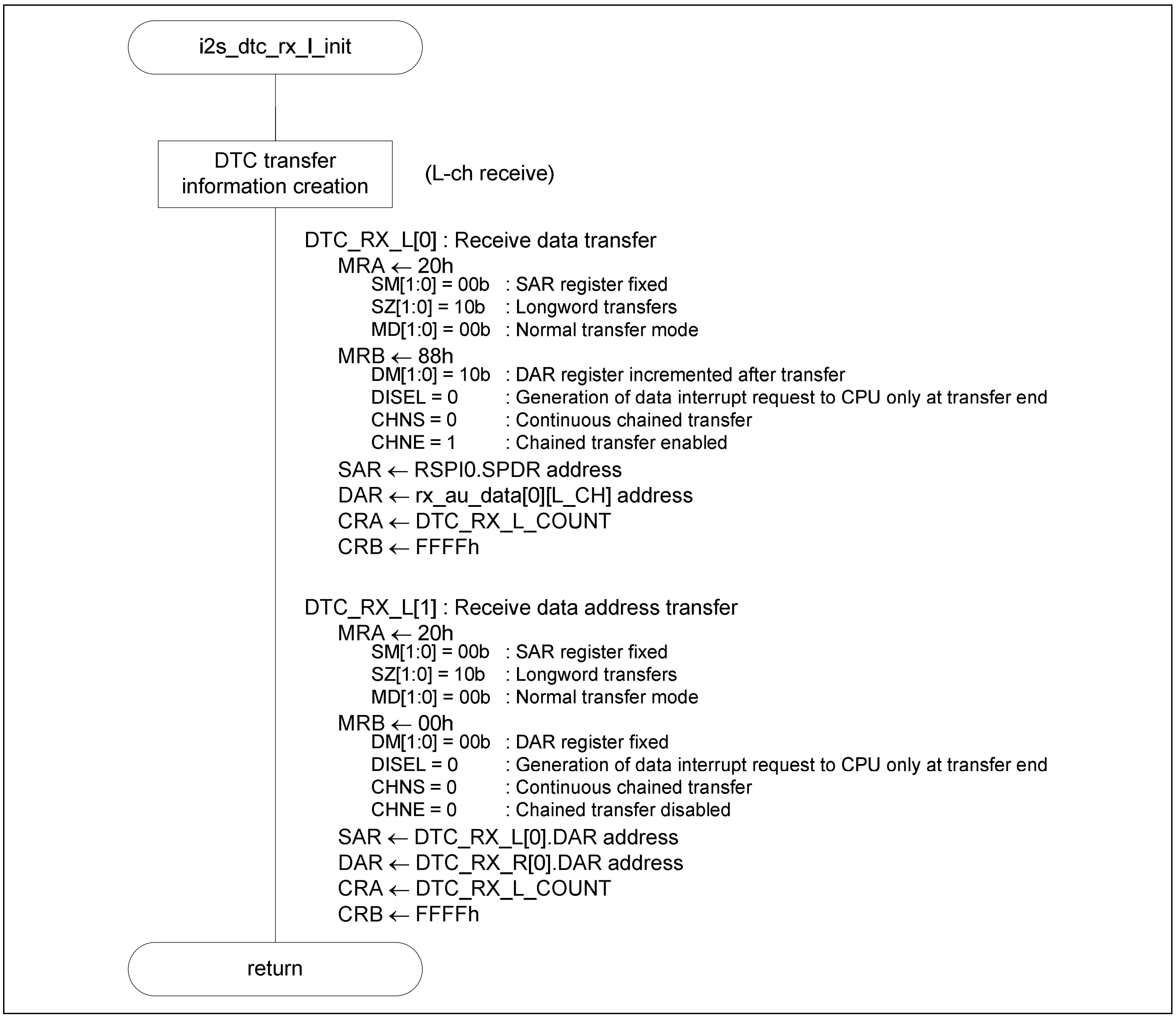
SAR ← DTC_CHAIN_DISABLE address
DAR ← DTC_TX_R[0].MRB address
CRA ← 2
CRB ← FFFFh

Figure 4.36 i2s_dtc_tx_r_init Function (4/4)

4.8.8 i2s_dtc_rx_l_init Function

Figure 4.37 is a flowchart of the i2s_dtc_rx_l_init function.

The i2s_dtc_rx_l_init function creates DTC transfer information for L-ch receive.



```
DTC_RX_L[0] : Receive data transfer
    MRA ← 20h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 10b   : Longword transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← 88h
        DM[1:0] = 10b   : DAR register incremented after transfer
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 0        : Continuous chained transfer
        CHNE = 1        : Chained transfer enabled
    SAR ← RSPI0.SPDR address
    DAR ← rx_au_data[0][L_CH] address
    CRA ← DTC_RX_L_COUNT
    CRB ← FFFFh

DTC_RX_L[1] : Receive data address transfer
    MRA ← 20h
        SM[1:0] = 00b   : SAR register fixed
        SZ[1:0] = 10b   : Longword transfers
        MD[1:0] = 00b   : Normal transfer mode
    MRB ← 00h
        DM[1:0] = 00b   : DAR register fixed
        DISEL = 0       : Generation of data interrupt request to CPU only at transfer end
        CHNS = 0        : Continuous chained transfer
        CHNE = 0        : Chained transfer disabled
    SAR ← DTC_RX_L[0].DAR address
    DAR ← DTC_RX_R[0].DAR address
    CRA ← DTC_RX_L_COUNT
    CRB ← FFFFh
```

Figure 4.37 i2s_dtc_rx_l_init Function

4.8.9 i2s_dtc_rx_r_init Function

Figure 4.38 is a flowchart of the i2s_dtc_rx_r_init function.

The i2s_dtc_rx_r_init function creates DTC transfer information for R-ch receive.

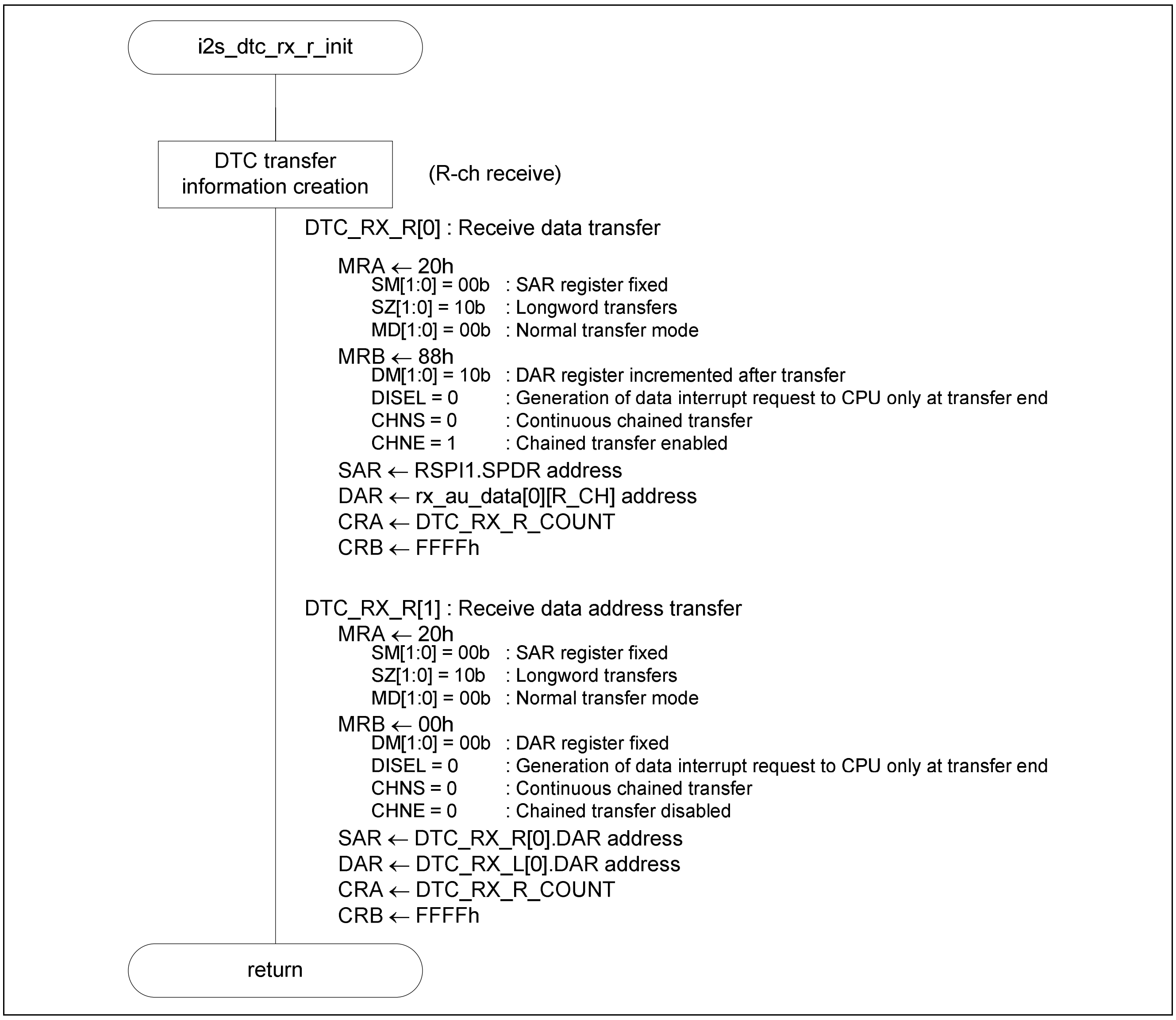
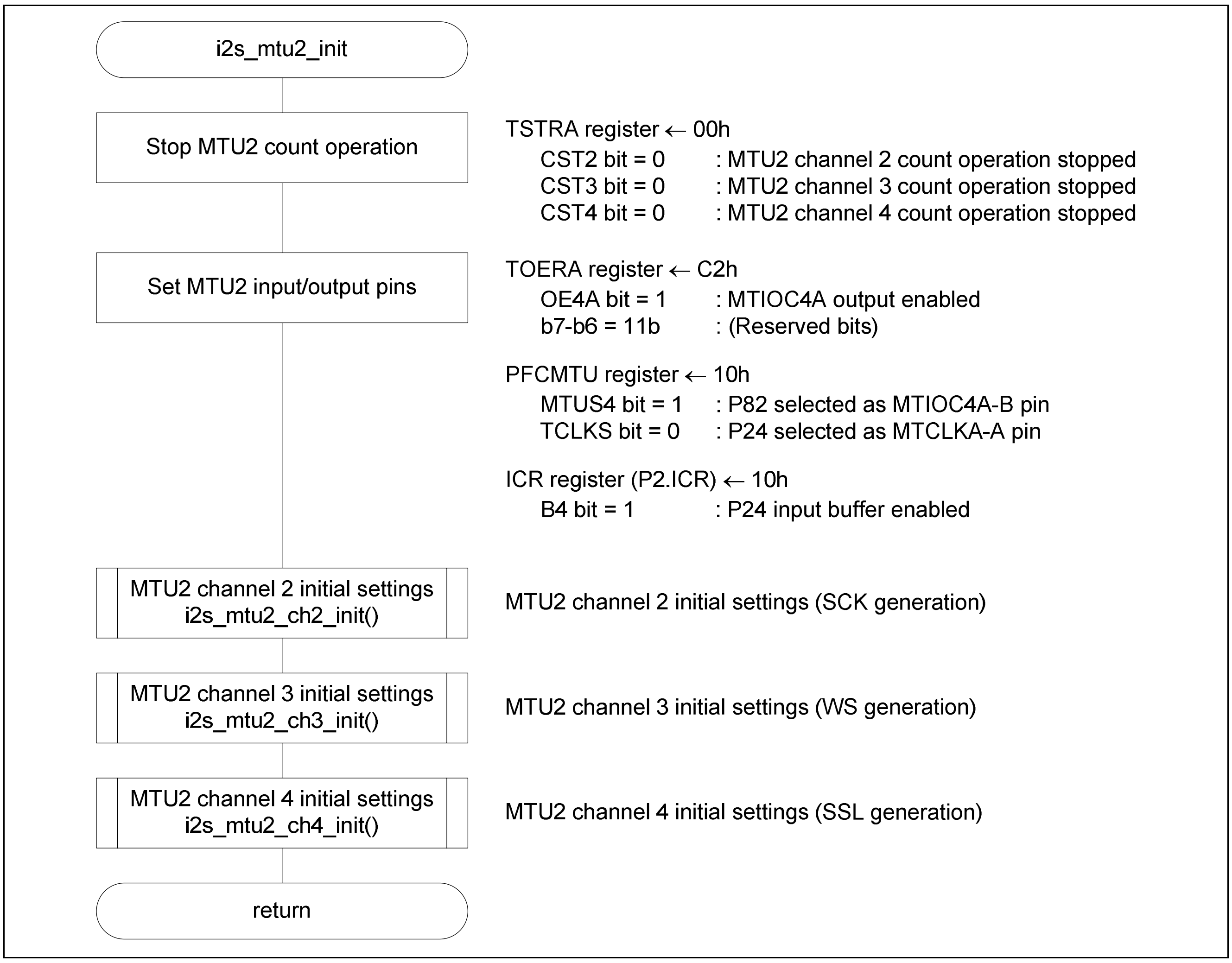

i2s_dtc_rx_r_init
DTC transfer information creation
(R-ch receive)
DTC_RX_R[0] : Receive data transfer
MRA ← 20h
SM[1:0] = 00b : SAR register fixed
SZ[1:0] = 10b : Longword transfers
MD[1:0] = 00b : Normal transfer mode
MRB ← 88h
DM[1:0] = 10b : DAR register incremented after transfer
DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
CHNS = 0 : Continuous chained transfer
CHNE = 1 : Chained transfer enabled
SAR ← RSPI1.SPDR address
DAR ← rx_au_data[0][R_CH] address
CRA ← DTC_RX_R_COUNT
CRB ← FFFFh
DTC_RX_R[1] : Receive data address transfer
MRA ← 20h
SM[1:0] = 00b : SAR register fixed
SZ[1:0] = 10b : Longword transfers
MD[1:0] = 00b : Normal transfer mode
MRB ← 00h
DM[1:0] = 00b : DAR register fixed
DISEL = 0 : Generation of data interrupt request to CPU only at transfer end
CHNS = 0 : Continuous chained transfer
CHNE = 0 : Chained transfer disabled
SAR ← DTC_RX_R[0].DAR address
DAR ← DTC_RX_L[0].DAR address
CRA ← DTC_RX_R_COUNT
CRB ← FFFFh
return

Figure 4.38 i2s_dtc_rx_r_init Function

4.8.10 i2s_mtu2_init Function

Figure 4.39 is a flowchart of the i2s_mtu2_init function.

The i2s_mtu2_init function makes initial settings to the MTU2.



i2s_mtu2_init
Stop MTU2 count operation
TSTRA register ← 00h
CST2 bit = 0 : MTU2 channel 2 count operation stopped
CST3 bit = 0 : MTU2 channel 3 count operation stopped
CST4 bit = 0 : MTU2 channel 4 count operation stopped
Set MTU2 input/output pins
TOERA register ← C2h
OE4A bit = 1 : MTIOC4A output enabled
b7-b6 = 11b : (Reserved bits)
PFCMTU register ← 10h
MTUS4 bit = 1 : P82 selected as MTIOC4A-B pin
TCLKS bit = 0 : P24 selected as MTCLKA-A pin
ICR register (P2.ICR) ← 10h
B4 bit = 1 : P24 input buffer enabled
MTU2 channel 2 initial settings i2s_mtu2_ch2_init()
MTU2 channel 2 initial settings (SCK generation)
MTU2 channel 3 initial settings i2s_mtu2_ch3_init()
MTU2 channel 3 initial settings (WS generation)
MTU2 channel 4 initial settings i2s_mtu2_ch4_init()
MTU2 channel 4 initial settings (SSL generation)
return

Figure 4.39 i2s_mtu2_init Function

4.8.11 i2s_mtu2_ch2_init Function

Figure 4.40 is a flowchart of the i2s_mtu2_ch2_init function.

The i2s_mtu2_ch2_init function makes settings to enable generation of SCK by using MTU2 channel 2.

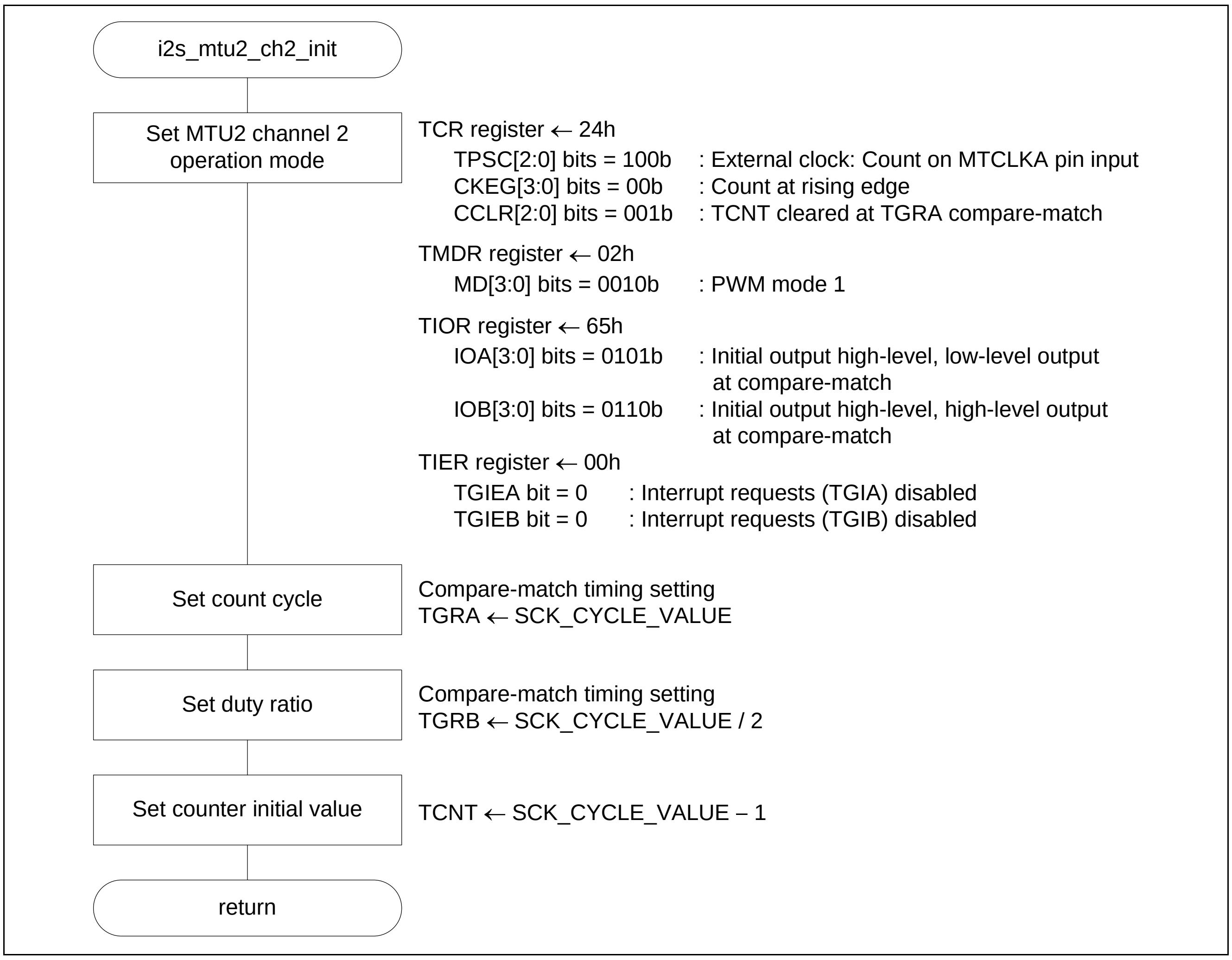

i2s_mtu2_ch2_init
Set MTU2 channel 2 operation mode
TCR register ← 24h
TPSC[2:0] bits = 100b : External clock: Count on MTCLKA pin input
CKEG[3:0] bits = 00b : Count at rising edge
CCLR[2:0] bits = 001b : TCNT cleared at TGRA compare-match
TMDR register ← 02h
MD[3:0] bits = 0010b : PWM mode 1
TIOR register ← 65h
IOA[3:0] bits = 0101b : Initial output high-level, low-level output at compare-match
IOB[3:0] bits = 0110b : Initial output high-level, high-level output at compare-match
TIER register ← 00h
TGIEA bit = 0 : Interrupt requests (TGIA) disabled
TGIEB bit = 0 : Interrupt requests (TGIB) disabled
Set count cycle
Compare-match timing setting
TGRA ← SCK_CYCLE_VALUE
Set duty ratio
Compare-match timing setting
TGRB ← SCK_CYCLE_VALUE / 2
Set counter initial value
TCNT ← SCK_CYCLE_VALUE – 1
return

Figure 4.40 i2s_mtu2_ch2_init Function

4.8.12 i2s_mtu2_ch3_init Function

Figure 4.41 is a flowchart of the i2s_mtu2_ch3_init function.

The i2s_mtu2_ch3_init function makes settings to enable generation of WS by using MTU2 channel 3.

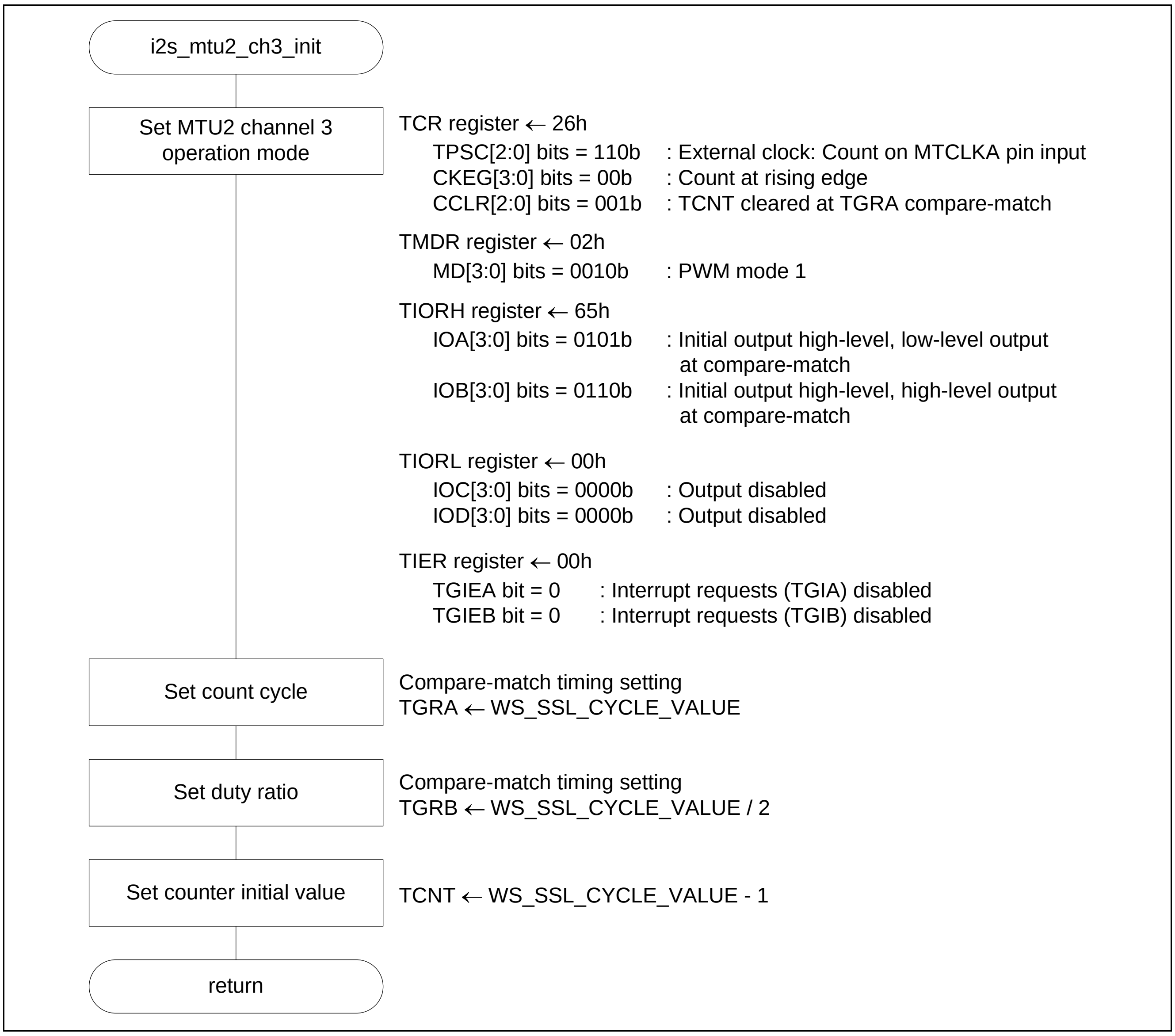
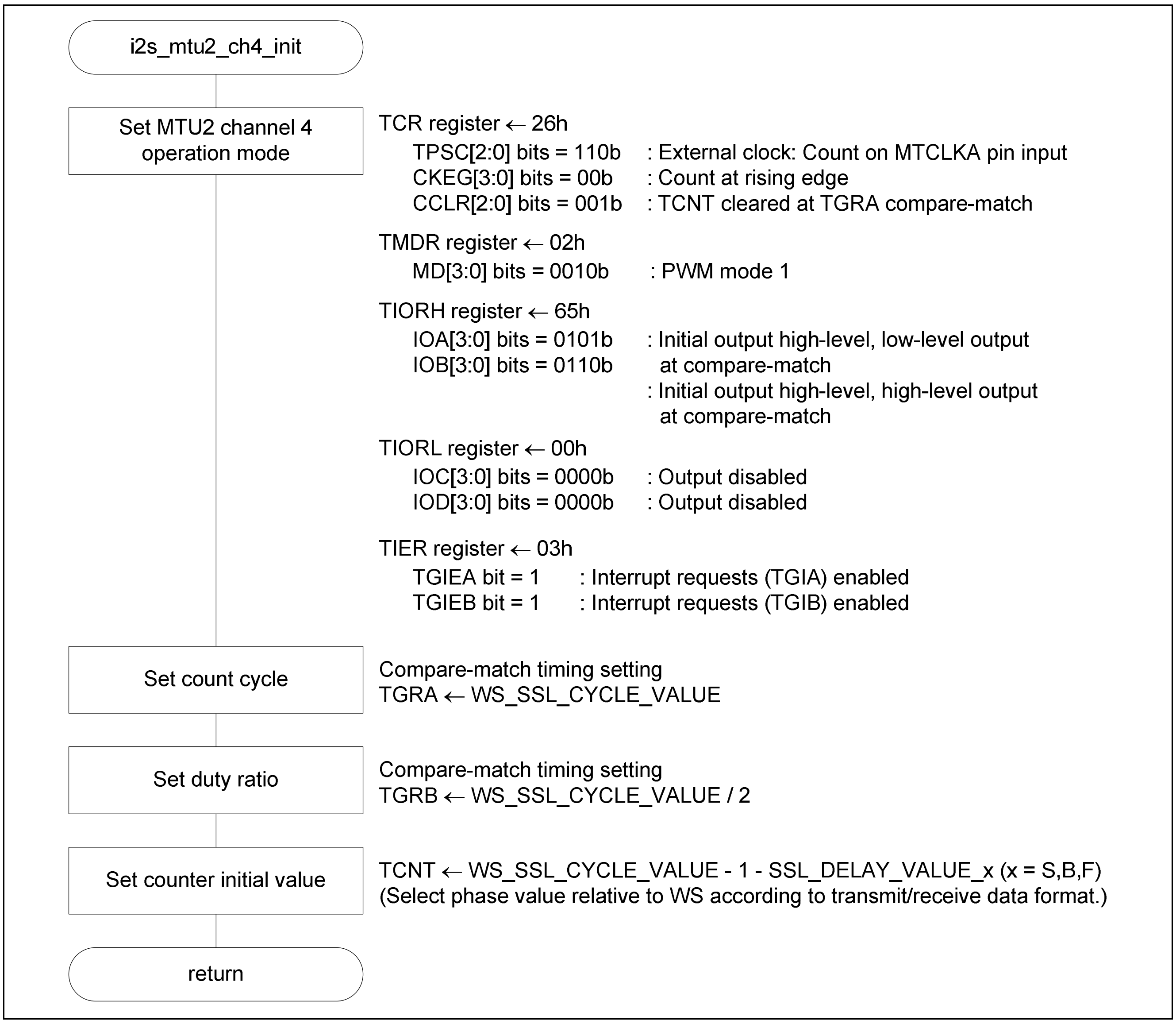

i2s_mtu2_ch3_init
Set MTU2 channel 3 operation mode
TCR register ← 26h
TPSC[2:0] bits = 110b : External clock: Count on MTCLKA pin input
CKEG[3:0] bits = 00b : Count at rising edge
CCLR[2:0] bits = 001b : TCNT cleared at TGRA compare-match
TMDR register ← 02h
MD[3:0] bits = 0010b : PWM mode 1
TIORH register ← 65h
IOA[3:0] bits = 0101b : Initial output high-level, low-level output at compare-match
IOB[3:0] bits = 0110b : Initial output high-level, high-level output at compare-match
TIORL register ← 00h
IOC[3:0] bits = 0000b : Output disabled
IOD[3:0] bits = 0000b : Output disabled
TIER register ← 00h
TGIEA bit = 0 : Interrupt requests (TGIA) disabled
TGIEB bit = 0 : Interrupt requests (TGIB) disabled
Set count cycle
Compare-match timing setting
TGRA ← WS_SSL_CYCLE_VALUE
Set duty ratio
Compare-match timing setting
TGRB ← WS_SSL_CYCLE_VALUE / 2
Set counter initial value
TCNT ← WS_SSL_CYCLE_VALUE - 1
return

Figure 4.41 i2s_mtu2_ch3_init Function

4.8.13 i2s_mtu2_ch4_init Function

Figure 4.42 is a flowchart of the i2s_mtu2_ch4_init function.

The i2s_mtu2_ch4_init function makes settings to enable generation of SSL by using MTU2 channel 4.



TCR register ← 26h
- TPSC[2:0] bits = 110b : External clock: Count on MTCLKA pin input
- CKEG[3:0] bits = 00b : Count at rising edge
- CCLR[2:0] bits = 001b : TCNT cleared at TGRA compare-match

TMDR register ← 02h
- MD[3:0] bits = 0010b : PWM mode 1

TIORH register ← 65h
- IOA[3:0] bits = 0101b : Initial output high-level, low-level output at compare-match
- IOB[3:0] bits = 0110b : Initial output high-level, high-level output at compare-match

TIORL register ← 00h
- IOC[3:0] bits = 0000b : Output disabled
- IOD[3:0] bits = 0000b : Output disabled

TIER register ← 03h
- TGIEA bit = 1 : Interrupt requests (TGIA) enabled
- TGIEB bit = 1 : Interrupt requests (TGIB) enabled

Compare-match timing setting
TGRA ← WS_SSL_CYCLE_VALUE

Compare-match timing setting
TGRB ← WS_SSL_CYCLE_VALUE / 2

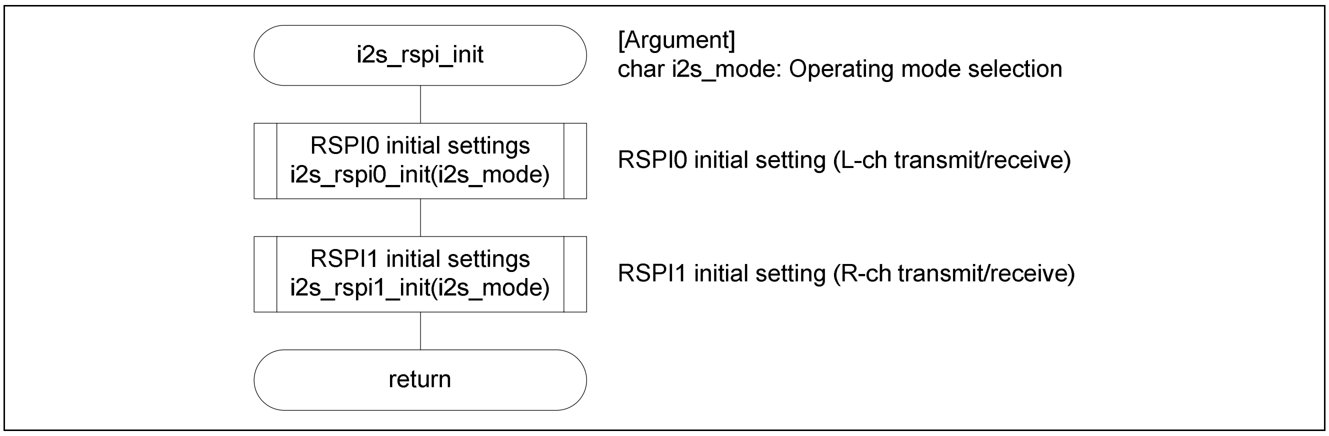
TCNT ← WS_SSL_CYCLE_VALUE - 1 - SSL_DELAY_VALUE_x (x = S,B,F)
(Select phase value relative to WS according to transmit/receive data format.)

Figure 4.42 i2s_mtu2_ch4_init Function

4.8.14 i2s_rspi_init Function

Figure 4.43 is a flowchart of the i2s_rspi_init function.

The i2s_rspi_init function makes the initial settings to the RSPI.



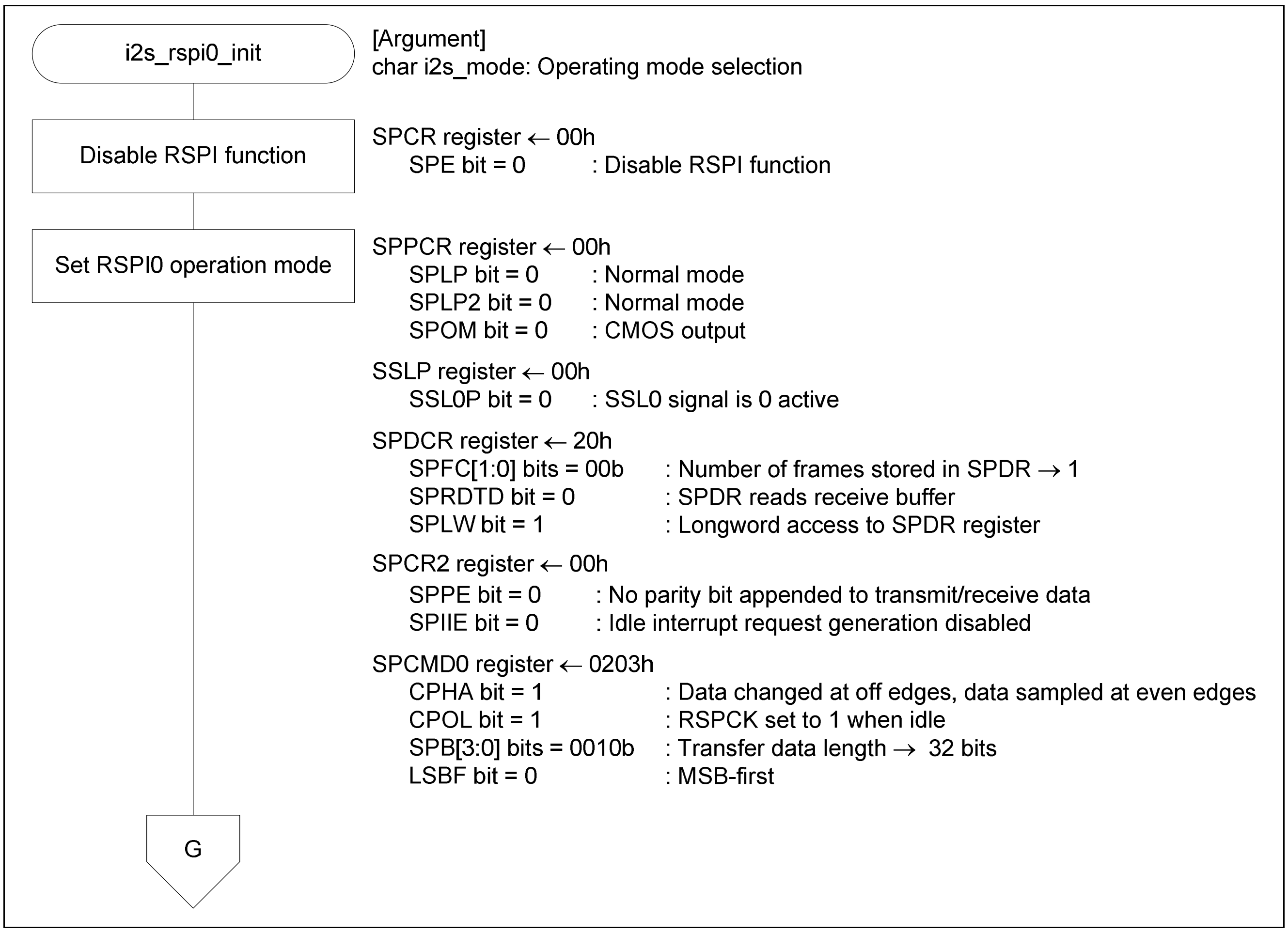
i2s_rspi_init
[Argument]
char i2s_mode: Operating mode selection
RSPI0 initial settings
i2s_rspi0_init(i2s_mode)
RSPI0 initial setting (L-ch transmit/receive)
RSPI1 initial settings
i2s_rspi1_init(i2s_mode)
RSPI1 initial setting (R-ch transmit/receive)
return

Figure 4.43 i2s_rspi_init Function

4.8.15 i2s_rspi0_init Function

Figure 4.44 is a flowchart of the i2s_rspi0_init function.

The i2s_rspi0_init function makes the initial settings to the RSPI0.



i2s_rspi0_init
[Argument]
char i2s_mode: Operating mode selection
Disable RSPI function
SPCR register ← 00h
SPE bit = 0 : Disable RSPI function
Set RSPI0 operation mode
SPPCR register ← 00h
SPLP bit = 0 : Normal mode
SPLP2 bit = 0 : Normal mode
SPOM bit = 0 : CMOS output
SSLP register ← 00h
SSLOP bit = 0 : SSL0 signal is 0 active
SPDCR register ← 20h
SPFC[1:0] bits = 00b : Number of frames stored in SPDR → 1
SPRDTD bit = 0 : SPDR reads receive buffer
SPLW bit = 1 : Longword access to SPDR register
SPCR2 register ← 00h
SPPE bit = 0 : No parity bit appended to transmit/receive data
SPIIE bit = 0 : Idle interrupt request generation disabled
SPCMD0 register ← 0203h
CPHA bit = 1 : Data changed at off edges, data sampled at even edges
CPOL bit = 1 : RSPCK set to 1 when idle
SPB[3:0] bits = 0010b : Transfer data length → 32 bits
LSBF bit = 0 : MSB-first
G

Figure 4.44 i2s_rspi0_init Function (1/2)

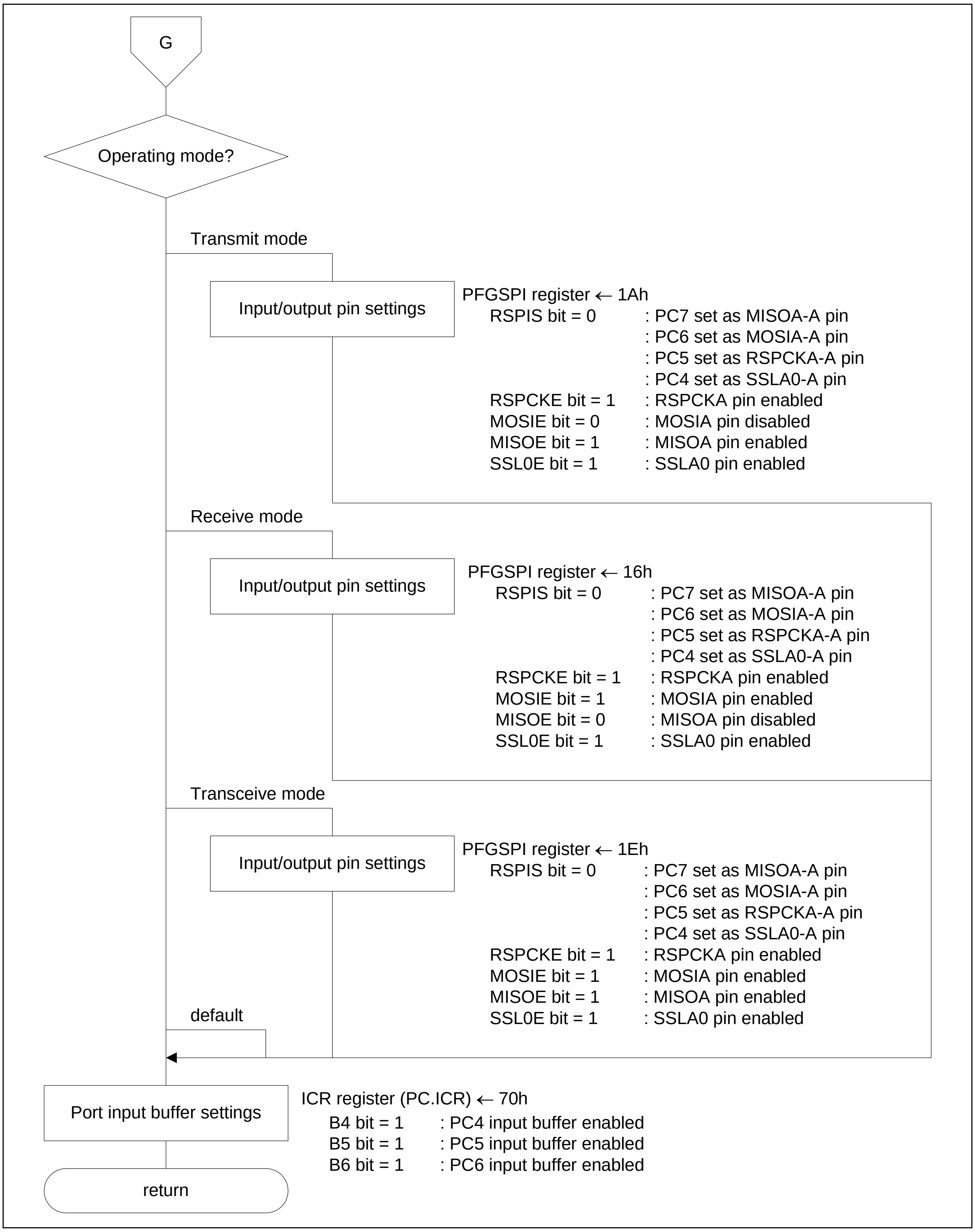
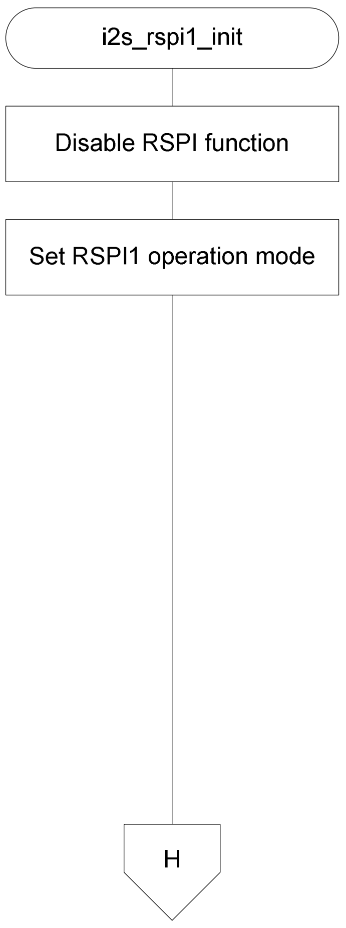

G
Operating mode?
Transmit mode
Input/output pin settings
PFGSPI register ← 1Ah
RSPIS bit = 0 : PC7 set as MISOA-A pin
: PC6 set as MOSIA-A pin
: PC5 set as RSPCKA-A pin
: PC4 set as SSLA0-A pin
RSPCKE bit = 1 : RSPCKA pin enabled
MOSIE bit = 0 : MOSIA pin disabled
MISOE bit = 1 : MISOA pin enabled
SSL0E bit = 1 : SSLA0 pin enabled
Receive mode
Input/output pin settings
PFGSPI register ← 16h
RSPIS bit = 0 : PC7 set as MISOA-A pin
: PC6 set as MOSIA-A pin
: PC5 set as RSPCKA-A pin
: PC4 set as SSLA0-A pin
RSPCKE bit = 1 : RSPCKA pin enabled
MOSIE bit = 1 : MOSIA pin enabled
MISOE bit = 0 : MISOA pin disabled
SSL0E bit = 1 : SSLA0 pin enabled
Transceive mode
Input/output pin settings
PFGSPI register ← 1Eh
RSPIS bit = 0 : PC7 set as MISOA-A pin
: PC6 set as MOSIA-A pin
: PC5 set as RSPCKA-A pin
: PC4 set as SSLA0-A pin
RSPCKE bit = 1 : RSPCKA pin enabled
MOSIE bit = 1 : MOSIA pin enabled
MISOE bit = 1 : MISOA pin enabled
SSL0E bit = 1 : SSLA0 pin enabled
default
Port input buffer settings
ICR register (PC.ICR) ← 70h
B4 bit = 1 : PC4 input buffer enabled
B5 bit = 1 : PC5 input buffer enabled
B6 bit = 1 : PC6 input buffer enabled
return

Figure 4.44 i2s_rspi0_init Function (2/2)

4.8.16 i2s_rspi1_init Function

Figure 4.45 is a flowchart of the i2s_rspi1_init function.

The i2s_rspi1_init function makes the initial settings to the RSPI1.



[Argument]
char i2s_mode: Operating mode selection

SPCR register ← 00h
SPE bit = 0 : Disable RSPI function

SPPCR register ← 00h
SPLP bit = 0 : Normal mode
SPLP2 bit = 0 : Normal mode
SPOM bit = 0 : CMOS output

SSLP register ← 00h
SSL0P bit = 0 : SSL0 signal is 0 active

SPDCR register ← 20h
SPFC[1:0] bits = 00b : Number of frames stored in SPDR → 1
SPRDTD bit = 0 : SPDR reads receive buffer
SPLW bit = 1 : Longword access to SPDR register

SPCR2 register ← 00h
SPPE bit = 0 : No parity bit appended to transmit/receive data
SPIIE bit = 0 : Idle interrupt request generation disabled

SPCMD0 register ← 0203h
CPHA bit = 1 : Data changed at off edges, data sampled at even edges
CPOL bit = 1 : RSPCK set to 1 when idle
SPB[3:0] bits = 0010b : Transfer data length → 32 bits
LSBF bit = 0 : MSB-first

Figure 4.45 i2s_rspi1_init Function (1/2)

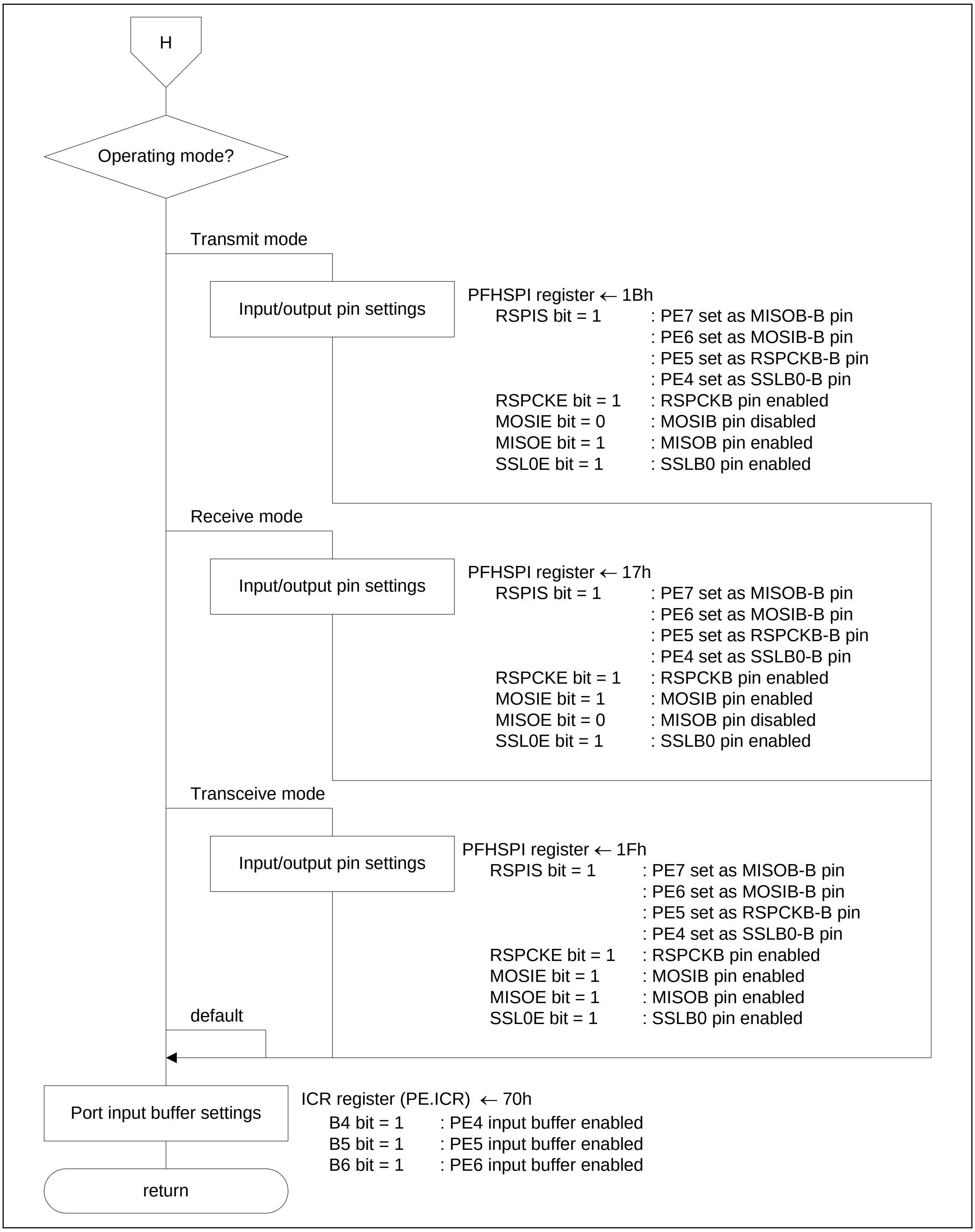

H
Operating mode?
Transmit mode
Input/output pin settings
PFHSPI register ← 1Bh
RSPIS bit = 1 : PE7 set as MISOB-B pin
: PE6 set as MOSIB-B pin
: PE5 set as RSPCKB-B pin
: PE4 set as SSLB0-B pin
RSPCKE bit = 1 : RSPCKB pin enabled
MOSIE bit = 0 : MOSIB pin disabled
MISOE bit = 1 : MISOB pin enabled
SSL0E bit = 1 : SSLB0 pin enabled
Receive mode
Input/output pin settings
PFHSPI register ← 17h
RSPIS bit = 1 : PE7 set as MISOB-B pin
: PE6 set as MOSIB-B pin
: PE5 set as RSPCKB-B pin
: PE4 set as SSLB0-B pin
RSPCKE bit = 1 : RSPCKB pin enabled
MOSIE bit = 1 : MOSIB pin enabled
MISOE bit = 0 : MISOB pin disabled
SSL0E bit = 1 : SSLB0 pin enabled
Transceive mode
Input/output pin settings
PFHSPI register ← 1Fh
RSPIS bit = 1 : PE7 set as MISOB-B pin
: PE6 set as MOSIB-B pin
: PE5 set as RSPCKB-B pin
: PE4 set as SSLB0-B pin
RSPCKE bit = 1 : RSPCKB pin enabled
MOSIE bit = 1 : MOSIB pin enabled
MISOE bit = 1 : MISOB pin enabled
SSL0E bit = 1 : SSLB0 pin enabled
default
Port input buffer settings
ICR register (PE.ICR) ← 70h
B4 bit = 1 : PE4 input buffer enabled
B5 bit = 1 : PE5 input buffer enabled
B6 bit = 1 : PE6 input buffer enabled
return

Figure 4.45 i2s_rspi1_init Function (2/2)

5. Sample Code

The sample code can be downloaded from the Renesas Electronics Corporation web site.

6. Reference Documents

- RX62N Group, RX621 Group User’s Manual: Hardware, Revision 1.11
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- Technical Updates and Technical Manuals
  (The latest information can be accessed at the Renesas Electronics Web site.)
- RX Family C Compiler Package, Version.1.0.1.0
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- C Compiler User’s Manual, Revision 1.00
  (The latest version can be downloaded from the Renesas Electronics Web site.)

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	Sep.27.11	—	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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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: +49-211-65030, 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.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
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 1.1