Abstract
This application note describes a method for transferring audio data using I2S communication with the serial peripheral interface (RSPI), data transfer controller (DTCa), and multi-function timer pulse unit 2 (MTU2a) in the RX63N Group and RX631 Group.

Products
- RX63N Group 177-pin and 176-pin packages with a ROM size between 768 KB and 2 MB
- RX63N Group 145-pin and 144-pin packages with a ROM size between 768 KB and 2 MB
- RX63N Group 100-pin package with a ROM size between 768 KB and 2 MB
- RX631 Group 177-pin and 176-pin packages with a ROM size between 256 KB and 2 MB
- RX631 Group 145-pin and 144-pin packages with a ROM size between 256 KB and 2 MB
- RX631 Group 100-pin package with a ROM size between 256 KB and 2 MB

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.
Contents

1 Specifications .................................................................................................................................... 4

2 Operation Confirmation Conditions ............................................................................................... 5

3 Reference Application Note ............................................................................................................ 5

4 Hardware ......................................................................................................................................... 6
   4.1 Pins Used ................................................................................................................................. 6
   4.2 Peripheral Device Connections ................................................................................................. 7

5 Software ......................................................................................................................................... 8
   5.1 Operation Overview .................................................................................................................. 8
      5.1.1 I2S Communication Using RSPI, DTCa, and MTU2a .................................................. 8
         5.1.1.1 Audio Data Format .................................................................................................... 8
         5.1.1.2 Transmitting and Receiving Audio Data ................................................................. 10
         5.1.1.3 I2S Communication Synchronization Recovery and RSPI Initialization ..... 13
         5.1.1.4 Generating I2S Communication Clocks and Channel Synchronization Signal .... 14
         5.1.1.5 Phase Difference between the Word Select Signal and Slave Select Signal ...... 16
      5.1.2 Transmit Operation ........................................................................................................... 17
         5.1.2.1 Timing of Transmit Operation .................................................................................. 17
         5.1.2.2 DTC Operation when Transmitting ......................................................................... 18
         5.1.2.3 Transferring the Transmit Data Address by the DTC ........................................... 20
         5.1.2.4 Completion of Audio Data Transmission ................................................................. 21
      5.1.3 Receive Operation .............................................................................................................. 23
         5.1.3.1 Timing of Receive Operation .................................................................................... 23
         5.1.3.2 DTC Operation when Receiving ............................................................................ 24
         5.1.3.3 Transferring Receive Data Address by the DTC .................................................... 26
         5.1.3.4 Completion of Audio Data Reception ...................................................................... 27

5.2 File Composition ......................................................................................................................... 29

5.3 Option-Setting Memory .............................................................................................................. 29

5.4 Constants ................................................................................................................................... 30

5.5 Structure/Union List ..................................................................................................................... 31

5.6 Variables .................................................................................................................................... 32

5.7 Functions ................................................................................................................................... 33

5.8 Function Specifications ................................................................................................................ 34

6 Flowcharts .................................................................................................................................... 38
   6.1 Main Processing ....................................................................................................................... 38
   6.2 Functions ................................................................................................................................. 39
      6.2.1 i2s_au_data_init ............................................................................................................... 39
      6.2.2 i2s_start .......................................................................................................................... 40
      6.2.3 i2s_dtc_init ....................................................................................................................... 41
      6.2.4 i2s_dtc_tx_l_init ............................................................................................................... 42
      6.2.5 i2s_dtc_tx_r_init ............................................................................................................... 46
      6.2.6 i2s_dtc_rx_l_init ............................................................................................................... 50
      6.2.7 i2s_dtc_rx_r_init ............................................................................................................... 51

5.1.2 Transmit Operation ........................................................................................................... 17

6.2.8  i2s_mtu2_init................................................................. 52
6.2.9  i2s_mtu2_ch2_init.......................................................... 53
6.2.10 i2s_mtu2_ch3_init.......................................................... 54
6.2.11 i2s_mtu2_ch4_init.......................................................... 55
6.2.12 i2s_rspi_init................................................................. 56
6.2.13 i2s_rspi0_init................................................................. 57
6.2.14 i2s_rspi1_init................................................................. 59

7  Appendix................................................................................. 61
8  Sample Code............................................................................ 62
9  Reference Documents............................................................. 62
1 Specifications

In this application note, the RSPI, DTCa, and MTU2a are used to transmit and receive audio data with I2S communication. The RSPI transfers audio data with serial communication according to the clock signal generated by the MTU2a.

Table 1.1 lists the Peripheral Functions and Their Applications, and Figure 1.1 shows the Block Diagram.

### Table 1.1 Peripheral Functions and Their Applications

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSPI channel 0</td>
<td>Input/output audio data for left channel</td>
</tr>
<tr>
<td>RSPI channel 1</td>
<td>Input/output audio data for right channel</td>
</tr>
<tr>
<td>MTU2a channel 2</td>
<td>Generate a serial transfer clock (SCK)</td>
</tr>
<tr>
<td>MTU2a channel 3</td>
<td>Generate a word select signal (WS)</td>
</tr>
<tr>
<td>MTU2a channel 4</td>
<td>Generate a slave select signal (SSL) to RSPI</td>
</tr>
<tr>
<td>DTCa</td>
<td>Transfer audio data to and from the on-chip RAM</td>
</tr>
</tbody>
</table>

![Figure 1.1 Block Diagram](image-url)
2 Operation Confirmation Conditions
The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>R5F563NBDDFC (RX63N Group)</td>
</tr>
</tbody>
</table>
| Operating frequencies     | - Main clock: 12 MHz
- PLL: 192 MHz (main clock divided by 1 and multiplied by 16)
- System clock (ICLK): 96 MHz (PLL divided by 2)
- Peripheral module clock A (PCLKA): 96 MHz (PLL divided by 2)
- Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4)
- External bus clock (BCLK): 48 MHz (PLL divided by 4)
- FlashIF clock (FCLK): 48 MHz (PLL divided by 4)
- IEBUS clock (IECLK): 48 MHz (PLL divided by 4) |
| Operating voltage         | 3.3 V                                                                     |
| Integrated development environment | Renesas Electronics Corporation                                      |
|                            | High-performance Embedded Workshop Version 4.09.01                        |
| C compiler                | Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01 |
|                           | Compile options -cpu=rx600 -output=obj="$\{(CONFIGDIR)$\$(FILELEAF).obj" -debug -nologo (The default setting is used in the integrated development environment.) |
| iodefine.h version        | Version 1.50                                                              |
| Endian                    | Little endian                                                             |
| Operating mode            | Single-chip mode                                                          |
| Processor mode            | Supervisor mode                                                           |
| Sample code version       | Version 1.00                                                              |
| Board used                | Renesas Starter Kit+ for RX63N (product part no.: R0K5063NC000BE)        |

3 Reference Application Note
For additional information associated with this document, refer to the following application note.

- RX63N Group, RX631 Group Initial Setting Rev. 1.00 (R01AN1245EJ0100)

The initial setting functions in the reference application note are used in the sample code in this application note. The revision number of the reference application note is the one when this application note was made. However the latest version is always recommended. Visit the Renesas Electronics Corporation website to check and download the latest version.
4 Hardware

4.1 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1_4/MTCLKA</td>
<td>Input</td>
<td>Input the external clock (12.288 MHz)</td>
</tr>
<tr>
<td>PB_5/MTIOC2A</td>
<td>Output</td>
<td>Output the serial transfer clock (SCK)</td>
</tr>
<tr>
<td>P1_7/MTIOC3A</td>
<td>Output</td>
<td>Output the word select signal (WS)</td>
</tr>
<tr>
<td>PA_0/MTIOC4A</td>
<td>Output</td>
<td>Output the slave select signal (SSL)</td>
</tr>
<tr>
<td>PA_4/SSLA0</td>
<td>Input</td>
<td>Input the RSPI0 slave select signal</td>
</tr>
<tr>
<td>PA_5/RSPCKA</td>
<td>Input</td>
<td>Input the RSPI0 serial transfer clock</td>
</tr>
<tr>
<td>PA_6/MOSIA</td>
<td>Input</td>
<td>Input audio data for left channel</td>
</tr>
<tr>
<td>PA_7/MISOA</td>
<td>Output</td>
<td>Output audio data for left channel (1)</td>
</tr>
<tr>
<td>PE_4/SSLB0</td>
<td>Input</td>
<td>Input the RSPI1 slave select signal</td>
</tr>
<tr>
<td>PE_5/RSPCKB</td>
<td>Input</td>
<td>Input the RSPI1 serial transfer clock</td>
</tr>
<tr>
<td>PE_2/MOSIB</td>
<td>Input</td>
<td>Input audio data for right channel</td>
</tr>
<tr>
<td>PE_3/MISOB</td>
<td>Output</td>
<td>Output audio data for right channel (1)</td>
</tr>
</tbody>
</table>

Note: When in reception mode, set the pin to high impedance so that data is not output.
4.2 Peripheral Device Connections

Figure 4.1 shows the Connection Diagram. Table 4.2 lists the Clock Signals Generated by MTU2a.
The MTU2a uses the external clock (12.288 MHz) as the count clock and generates the clock signals listed in Table 4.2.
RSPI channel 0 (RSPI0) is used to transmit and receive the left channel components (L-ch) of the audio data and RSPI channel 1 (RSPI1) is used for the right channel components (R-ch). The RSPI operates in slave mode and switches the active channel according to the SSL signal.

Table 4.2 Clock Signals Generated by MTU2a

<table>
<thead>
<tr>
<th>Channel</th>
<th>Clock Signal</th>
<th>Symbol</th>
<th>Output Pin</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel 2</td>
<td>Serial transfer clock</td>
<td>SCK</td>
<td>MTIOC2A</td>
<td>3.072 MHz</td>
</tr>
<tr>
<td>Channel 3</td>
<td>Word select signal</td>
<td>WS</td>
<td>MTIOC3A</td>
<td>48 kHz</td>
</tr>
<tr>
<td>Channel 4</td>
<td>RSPI slave select signal</td>
<td>SSL</td>
<td>MTIOC4A</td>
<td>48 kHz</td>
</tr>
</tbody>
</table>
5  Software

5.1  Operation Overview

5.1.1  I2S Communication Using RSPI, DTCa, and MTU2a

This section describes the method of I2S communication using RSPI, DTCa, and MTU2a.

5.1.1.1  Audio Data Format

Audio data that consists of 24-bit of data and 8-bit of padding is handled in 32-bit units.

(a) Data Format in the On-Chip RAM

Figure 5.1 shows the Audio Data in the On-Chip RAM.
(b) Transmit and Receive Data Format
Serial transfer is performed with 32-bit data length and MSB-first. SCK is used as the serial transfer clock and WS is used as the channel select signal. When WS is low, the L-ch components of the audio data are transferred. When WS is high, the R-ch components are transferred. The position where padding is added to the transmit and receive data is selected according to the audio interface format of the external device.

The data format can be selectable from the following three formats in this application note.
- Standard format
- Backward-padding format
- Forward-padding format

Figure 5.2 shows the Transmit and Receive Data Format.
5.1.1.2 Transmitting and Receiving Audio Data

I2S transmission is performed by transferring audio data placed in the on-chip RAM to the RSPI using the DTC, and outputting the audio data from the RSPI. I2S reception is performed by transferring audio data received from the RSPI to the on-chip RAM using the DTC.

The RSPI operates in slave mode (SPI operation) with 32-bit data length and MSB-first.

Figure 5.3 shows the Flow of Audio Data.

The audio data is separated into L-ch and R-ch components. RSPI0 transmits and receives the L-ch components, and RSPI1 transmits and receives the R-ch components. The slave-select polarity of RSPI0 is set to active low and that of RSPI1 is set to active high. RSPI0 or RSPI1 channel is selected according to the SSL output from MTU2a channel 4.

Figure 5.3 Flow of Audio Data
(a) Transmit Operation

Figure 5.4 shows the RSPI Audio Data Transmit Timing. Audio data for transmitting to the external device is generated by combining the transmit data (L-ch) output from RSPI0 and the transmit data (R-ch) output from RSPI1 with an external OR circuit. The RSPI output channel that is not outputting transmit data is placed in high-impedance, thus those channels need to be pulled down so that they do not affect the data combined with the OR circuit.

![RSPI Audio Data Transmit Timing Diagram]

---

**Figure 5.4 RSPI Audio Data Transmit Timing**
(b) Receive Operation

Figure 5.5 shows the RSPI Audio Data Receive Timing.

Audio data for receiving from the external device is input to both channels RSPI0 and RSPI1. The RSPI channel to be activated is selected according to the SSL signal. The input audio data is separated into L-ch and R-ch components for reception.

Figure 5.5   RSPI Audio Data Receive Timing
5.1.1.3 I2S Communication Synchronization Recovery and RSPI Initialization

To perform synchronization recovery in the I2S communication when a bit slip occurs due to noise insertion, the RSPI is initialized by the DTC each time L-ch or R-ch data is transferred.

Figure 5.6 shows the RSPI Initialization Timing.
5.1.1.4 Generating I2S Communication Clocks and Channel Synchronization Signal

SCK and WS are generated by the MTU2a compare match using an external clock (12.288 MHz) as a count clock, and the RSPI channel select signal (SSL) is also generated by the MTU2a compare match in the same way.

(a) Generating SCK

MTU2a channel 2 is set to PWM mode 1 to output SCK with a frequency of 3.072 MHz (12.288 MHz / 4), duty ratio of 50%, and high initial output.

Figure 5.7 shows the SCK Generation with MTU2a Channel 2.

(b) Generating WS

MTU2a channel 3 is set to PWM mode 1 to output WS with a frequency of 48 kHz (12.288 MHz / 256), duty ratio of 50%, and high initial output.

Figure 5.8 shows the WS Generation with MTU2a Channel 3.
(c) Generating SSL

MTU2a channel 4 is set to PWM mode 1 to output SSL with a frequency of 48 kHz (12.288 MHz / 256), duty ratio of 50%, and high initial output.

A phase difference is generated between WS and SSL by setting different initial values for the MTU2a channel 4 timer counter and the MTU2a channel 3 timer counter which generates WS. The phase difference between WS and SSL achieves the transmit and receive data formats described in (b) Transmit and Receive Data Format in 5.1.1.1. Refer to 5.1.1.5 Phase Difference between the Word Select Signal and Slave Select Signal for details on the phase difference.

Figure 5.9 shows the SSL Generation with MTU2a Channel 4.
5.1.1.5 Phase Difference between the Word Select Signal and Slave Select Signal

Each data format described in (b) Transmit and Receive Data Format in 5.1.1.1 can be achieved by changing the MTU2a channel 4 initial value described in (C) Generating SSL in 5.1.1.4.

Figure 5.10 shows the Phase Difference between WS and SSL in Each Transmit and Receive Data Format.

![Phase Difference Diagram](image-url)
5.1.2 Transmit Operation

5.1.2.1 Timing of Transmit Operation

Figure 5.11 shows the Timing of Transmit Operation for Standard Format.

The RSPI transmits audio data placed in the on-chip RAM. The RSPI operates in slave mode (SPI operation) with 32-bit data length and MSB-first.

The audio data to be transmitted is separated into L-ch and R-ch components. RSPI0 transmits the L-ch components and RSPI1 transmits the R-ch components. The slave-select polarity of RSPI0 is set to active low and that of RSPI1 is set to active high, and RSPI0 or RSPI1 channel is selected according to SSL. Data output on the active RSPI channel starts at the falling edge of the first SCK signal generated after channel switching, and thereafter data is output synchronizing with SCK.

The MTU2a outputs SSL which has approximately 1-bit phase delay of SCK relative to WS. Refer to 5.1.1.5 Phase Difference between the Word Select Signal and Slave Select Signal for details on the phase difference.

![Figure 5.11 Timing of Transmit Operation for Standard Format](image-url)

Figure 5.11 Timing of Transmit Operation for Standard Format
5.1.2.2 DTC Operation when Transmitting

Figure 5.12 shows the DTC Operation when Transmitting.

The DTC transfers transmit data from the on-chip RAM to the RSPI. The DTC performs two types of transfer operations; L-ch transmission and R-ch transmission. The DTC activation is triggered by a compare match interrupt generated when an edge occurs on SSL. The DTC is activated for L-ch transmission on a rising edge of SSL, and for R-ch transmission on a falling edge of SSL.

![Diagram showing DTC Operation when Transmitting]

Figure 5.12  DTC Operation when Transmitting
Figure 5.13 shows the DTC Operation when Transmitting L-ch data and Figure 5.14 shows the DTC Operation when Transmitting R-ch data.

The DTC uses chain transfer to initialize the RSPI, transfer the transmit data, and transfer the transmit data address. To initialize the RSPI, disable the RSPI by writing 0 to the RSPI function enable bit (SPE) in the RSPI control register (SPCR), and then enable the RSPI again by writing 1 to the SPE bit.

Note: Refer to the User’s Manual: Hardware for details on initialization with the RSPI function enable bit.
5.1.2.3 Transferring the Transmit Data Address by the DTC

Figure 5.15 shows the Transmit Data Address Transfer by the DTC.

L-ch components and R-ch components of the transmit audio data are alternately placed in the on-chip RAM in 4-byte units. Refer to Figure 5.1 Audio Data in the On-Chip RAM. In the sample code, transmit data is transferred by performing L-ch transmission and R-ch transmission alternately using DTC transfer.

As shown in Figure 5.15, the DTC transfers data 0 of the transmit data for L-ch transmission, then the transfer source address after completion of the L-ch transmission is transferred using chain transfer to be used as the transfer source address for data 0 of R-ch transmission. The DTC transfers data 0 of the transmit data for R-ch transmission, then the transfer source address after completion of the R-ch transmission is transferred using chain transfer to be used as the transfer source address for data 1 of L-ch transmission. The DTC transfers data 1 of the transmit data for L-ch transmission, then the transfer source address after completion of the L-ch transmission is transferred using chain transfer to be used as the transfer source address for next R-ch transmission.

In this manner, the DTC transfers the transmit data address as the transfer source address for R-ch transmission when transmitting L-ch data, and as the transfer source address for L-ch transmission when transmitting R-ch data.

![Figure 5.15 Transmit Data Address Transfer by the DTC](image-url)
5.1.2.4 Completion of Audio Data Transmission

Audio data transmission is completed by disabling the RSPI, thereby halting its operation. Disabling the RSPI is performed by DTC transfer for L-ch and R-ch transmission.

(a) End Processing of L-ch Transmission

Figure 5.16 shows the End Processing of L-ch Transmission. The DTC performs end processing for L-ch transmission using chain transfer at (n-1)th DTC activation. With end processing, processing at nth DTC activation is only disabling RSPI0, and then the RSPI0 operation stops.

Figure 5.16   End Processing of L-ch Transmission

Figure 5.17 shows the End Processing of L-ch Transmission by DTC Transfer. As shown in Figure 5.13 DTC Operation when Transmitting L-ch data, with the DTC transfer for L-ch transmission, the chain transfer is used for disabling and enabling RSPI0, transferring the transmit data, and transferring the transmit data address.

By setting the transfer counter, which is included in the transfer information of the transmit data address transfer, to n–1, 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 performed by chain transfer after the transmit data address transfer at (n –1)th DTC activation.

In transmit end processing, by rewriting the DTC chain transfer enable bit in the transfer information for disabling RSPI0, chain transfer is disabled after RSPI0 is disabled at nth DTC activation. Therefore only RSPI0 is disabled at nth DTC activation, then the RSPI0 operation is stopped and the transmit operation is completed.

Figure 5.17   End Processing of L-ch Transmission by DTC Transfer
(b) End Processing of R-ch Transmission

Figure 5.18 shows the End Processing of R-ch Transmission.
The DTC performs end processing for R-ch transmission using chain transfer at its nth activation. With end processing, processing at (n+1)th DTC activation is only disabling RSPI1, and then the RSPI1 operation stops.

Figure 5.18 End Processing of R-ch Transmission

Figure 5.19 shows the End Processing of R-ch Transmission by DTC Transfer.
As shown in Figure 5.14 DTC Operation when Transmitting R-ch data, with the DTC transfer for R-ch transmission, the chain transfer is used for disabling and enabling RSPI1, transferring the transmit data, and transferring the transmit data address.
By setting the transfer counter, which is included in the transfer information of the transmit data address transfer, to n, 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 performed by chain transfer after the transmit data address transfer at nth DTC activation.
In transmit end processing, by rewriting the DTC chain transfer enable bit in the transfer information for disabling RSPI1, chain transfer is disabled after RSPI1 is disabled at (n+1)th DTC activation. Therefore only RSPI1 is disabled at (n+1)th DTC activation, then the RSPI1 operation is stopped and the transmit operation is completed.

Figure 5.19 End Processing of R-ch Transmission by DTC Transfer
5.1.3 Receive Operation

5.1.3.1 Timing of Receive Operation

Figure 5.20 shows the Timing of Receive Operation for Standard Format.

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

The audio data from an external device is input to both channels RSPI0 and RSPI1, and is separated into L-ch and R-ch components to be received. The slave-select polarity of RSPI0 is set to active low and that of RSPI1 is set to active high. RSPI0 or RSPI1 channel is selected according to SSL. Data is input through the active RSPI channel at the rising edge of the first SCK signal generated after channel switching, and thereafter data is input synchronizing with SCK.

The data input by the RSPI is transferred to the on-chip RAM by the DTC and stored as shown in Figure 5.1. The MTU2a outputs SSL which has approximately 1-bit phase delay of SCK relative to WS. Refer to 5.1.1.5 Phase Difference between the Word Select Signal and Slave Select Signal for details on the phase difference.

![Figure 5.20 Timing of Receive Operation for Standard Format](image-url)
5.1.3.2 DTC Operation when Receiving

The DTC transfers receive data from the RSPI to the on-chip RAM. L-ch reception, R-ch reception, RSPI0 initialization, and RSPI1 initialization are performed using DTC transfer. The DTC is activated for data reception by the RSPI receive buffer full interrupt. The DTC is activated for L-ch reception by the RSPI0 receive buffer full interrupt, and for R-ch reception by the RSPI1 receive buffer full interrupt. The DTC is activated for RSPI initialization by a compare match interrupt when an edge occurs on SSL. The DTC is activated for RSPI0 initialization on the rising edge of SSL and for RSPI1 initialization on the falling edge of SSL.

![Diagram of DTC Operation when Receiving]

Figure 5.21 DTC Operation when Receiving

Figure 5.22 shows DTC Operation when Receiving L-ch data and Figure 5.23 shows DTC Operation when Receiving R-ch data.

The DTC uses chain transfer to transfer the receive data and receive data address, and initialize the RSPI. To initialize the RSPI, disable the RSPI by writing 0 to the RSPI function enable bit (SPE) in the RSPI control register (SPCR), and then enable the RSPI again by writing 1 to the SPE bit.

Note: Refer to the User’s Manual: Hardware for details on initialization with the RSPI function enable bit.
Figure 5.22 DTC Operation when Receiving L-ch data

Figure 5.23 DTC Operation when Receiving R-ch data
5.1.3.3 Transferring Receive Data Address by the DTC

Figure 5.24 shows Receive Data Address Transfer by the DTC.

L-ch components and R-ch components of the received audio data are alternately placed in the on-chip RAM in 4-byte units. Refer to Figure 5.1 Audio Data in the On-Chip RAM. In the sample code, the receive data is transferred by performing L-ch reception and R-ch reception alternately using DTC transfer.

As shown in Figure 5.24, the DTC transfers data 0 of the receive data for L-ch reception, then the transfer destination address after completion of the L-ch reception is transferred using chain transfer to be used as the transfer destination address for data 0 of R-ch reception. The DTC transfers data 0 of the receive data for R-ch reception, then the transfer destination address after completion of the R-ch reception is transferred using chain transfer to be used as the transfer destination address for data 1 of L-ch reception. The DTC transfers data 1 of the receive data for L-ch reception, then the transfer destination address after completion of the L-ch reception is transferred using chain transfer to be used as the transfer destination address for next R-ch reception.

In this manner, the DTC transfers the receive data address as the transfer destination address for R-ch reception when transferring L-ch data, and as the transfer destination address for L-ch reception when transferring R-ch data.

![Figure 5.24 Receive Data Address Transfer by the DTC](image-url)
5.1.3.4 Completion of Audio Data Reception

Audio data reception is completed by disabling the RSPI, thereby halting its operation. The RSPI is disabled by initializing RSPI0 and RSPI1 using DTC transfer.

(a) End Processing of L-ch Reception

Figure 5.25 shows End Processing of L-ch Reception. End processing is performed by DTC transfer for RSPI0 initialization using the chain transfer at (n-1)th DTC activation. With end processing, processing at nth DTC activation is only disabling RSPI0, and then the RSPI0 operation stops.

![End Processing of L-ch Reception](image)

Figure 5.25  End Processing of L-ch Reception

Figure 5.26 shows End Processing of L-ch Reception by DTC Transfer for RSPI0 Initialization. As shown in Figure 5.22 DTC Operation when Receiving L-ch data, with DTC transfer for RSPI0 initialization, the chain transfer is used for disabling and enabling RSPI0.

By setting the transfer counter, which is included in the transfer information for enabling RSPI0, to n–1, 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 performed by chain transfer after RSPI0 is enabled at (n–1)th DTC activation.

In receive end processing, by rewriting the DTC chain transfer enable bit in the transfer information for disabling RSPI0, chain transfer is disabled after RSPI0 is disabled at nth DTC activation. Therefore only RSPI0 is disabled at nth DTC activation, then the RSPI0 operation is stopped and the receive operation is completed.

![End Processing of L-ch Reception by DTC Transfer for RSPI0 Initialization](image)

Figure 5.26  End Processing of L-ch Reception by DTC Transfer for RSPI0 Initialization
(b) End Processing of R-ch Reception

Figure 5.27 shows End Processing of R-ch Reception.

End processing for R-ch reception is performed by DTC transfer for initializing RSPI1 using chain transfer at nth DTC activation. With end processing, processing at (n+1)th DTC activation is only disabling RSPI1, and then the RSPI1 operation stops.

---

Figure 5.27 End Processing of R-ch Reception

---

Figure 5.28 shows End Processing of R-ch Reception by DTC Transfer for RSPI1 Initialization.

As shown in Figure 5.23 DTC Operation when Receiving R-ch data, with the DTC transfer for RSPI1 initialization, the chain transfer is used for disabling and enabling RSPI1.

By setting the transfer counter, which is included in the transfer information for enabling RSPI1, to n, 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 performed by chain transfer after RSPI1 is enabled at nth DTC activation.

In receive end processing, by rewriting the DTC chain transfer enable bit in the transfer information for disabling RSPI1, chain transfer is disabled after RSPI1 is disabled at (n+1)th DTC activation. Therefore only RSPI1 is disabled at (n+1)th DTC activation, then the RSPI1 operation is stopped and the receive operation is completed.

---

Figure 5.28 End Processing of R-ch Reception by DTC Transfer for RSPI1 Initialization
5.2 File Composition

Table 5.1 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 5.1 Files Used in the Sample Code

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>Main Processing</td>
<td></td>
</tr>
<tr>
<td>r_init_stop_module.c</td>
<td>Stop processing for active peripheral</td>
<td></td>
</tr>
<tr>
<td></td>
<td>functions after a reset</td>
<td></td>
</tr>
<tr>
<td>r_init_stop_module.h</td>
<td>Header file for r_init_stop_module.c</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.c</td>
<td>Nonexistent port initialization</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.h</td>
<td>Header file for r_init_non_existent_port.c</td>
<td></td>
</tr>
<tr>
<td>r_init_clock.c</td>
<td>Clock initialization</td>
<td></td>
</tr>
<tr>
<td>r_init_clock.h</td>
<td>Header file for r_init_clock.c</td>
<td></td>
</tr>
<tr>
<td>i2s_dtc.c</td>
<td>Settings for the DTCa operation</td>
<td></td>
</tr>
<tr>
<td>i2s_mtu2.c</td>
<td>Settings for the MTU2a operation</td>
<td></td>
</tr>
<tr>
<td>i2s_respi.c</td>
<td>Settings for the RSPI operation</td>
<td>Settings for transmit and receive data format</td>
</tr>
<tr>
<td>i2s.h</td>
<td>Macro definitions for I2S communication</td>
<td></td>
</tr>
</tbody>
</table>

5.3 Option-Setting Memory

Table 5.2 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.2 Option-Setting Memory Configured in the Sample Code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS0</td>
<td>FFFF FF8Fh to FFFF FF8Ch</td>
<td>FFFF FFFFh</td>
<td>The IWDT is stopped after a reset. The WDT is stopped after a reset.</td>
</tr>
<tr>
<td>OFS1</td>
<td>FFFF FF8Bh to FFFF FF88h</td>
<td>FFFF FFFFh</td>
<td>The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.</td>
</tr>
<tr>
<td>MDES</td>
<td>FFFF FF83h to FFFF FF80h</td>
<td>FFFF FFFFh</td>
<td>Little endian</td>
</tr>
</tbody>
</table>
5.4 Constants

Table 5.3 lists the Constants Used in the Sample Code.

Table 5.3  Constants Used in the Sample Code

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_CH</td>
<td>0</td>
<td>L-ch index No. of the audio data array in the on-chip RAM</td>
</tr>
<tr>
<td>R_CH</td>
<td>1</td>
<td>R-ch index No. of the audio data array in the on-chip RAM</td>
</tr>
<tr>
<td>NULL_DATA</td>
<td>0x00000000</td>
<td>Initialization data for the receive audio data area</td>
</tr>
<tr>
<td>TRANSMIT_FORMAT</td>
<td>STANDARD_PADDING or BACKWARD_PADDING or FORWARD_PADDING</td>
<td>Transmit and receive data format selection (select from standard, backward-padding, and forward-padding)</td>
</tr>
<tr>
<td>STANDARD_PADDING</td>
<td>0</td>
<td>Standard format</td>
</tr>
<tr>
<td>BACKWARD_PADDING</td>
<td>1</td>
<td>Backward-padding format</td>
</tr>
<tr>
<td>FORWARD_PADDING</td>
<td>2</td>
<td>Forward-padding format</td>
</tr>
<tr>
<td>TRANSMIT_MODE</td>
<td>0</td>
<td>Transmit mode</td>
</tr>
<tr>
<td>RECEIVE_MODE</td>
<td>1</td>
<td>Receive mode</td>
</tr>
<tr>
<td>TRANSCIEVE_MODE</td>
<td>2</td>
<td>Transmit and receive mode</td>
</tr>
<tr>
<td>AUD_SIZE</td>
<td>0x100</td>
<td>Audio data total byte count</td>
</tr>
<tr>
<td>AUD_NUM</td>
<td>(AUD_SIZE / 4) / 2</td>
<td>The number of L-ch/R-ch audio data</td>
</tr>
<tr>
<td>DTC_TX_CHAIN_SIZE</td>
<td>5</td>
<td>The number of transfer information for DTCa chain transfer when transmitting</td>
</tr>
<tr>
<td>DTC_RX_CHAIN_SIZE</td>
<td>2</td>
<td>The number of transfer information for DTCa chain transfer when receiving</td>
</tr>
<tr>
<td>DTC_TX_L_COUNT</td>
<td>AUD_NUM - 1</td>
<td>The number of DTC transfers for L-ch transmission or RSPI0 initialization</td>
</tr>
<tr>
<td>DTC_TX_R_COUNT</td>
<td>AUD_NUM</td>
<td>The number of DTC transfers for R-ch transmission or RSPI1 initialization</td>
</tr>
<tr>
<td>DTC_RX_L_COUNT</td>
<td>AUD_NUM</td>
<td>The number of DTC transfers for L-ch reception</td>
</tr>
<tr>
<td>DTC_RX_R_COUNT</td>
<td>AUD_NUM</td>
<td>The number of DTC transfers for R-ch reception</td>
</tr>
<tr>
<td>SCK_CYCLE_VALUE</td>
<td>0x0003</td>
<td>SCK cycle</td>
</tr>
<tr>
<td>WS_SSL_CYCLE_VALUE</td>
<td>0x00FF</td>
<td>WS and SSL cycle</td>
</tr>
<tr>
<td>SSL_DELAY_VALUE_S</td>
<td>0x0003</td>
<td>Phase value of WS and SSL for standard format</td>
</tr>
<tr>
<td>SSL_DELAY_VALUE_B</td>
<td>0x0001</td>
<td>Phase value of WS and SSL for backward-padding format</td>
</tr>
<tr>
<td>SSL_DELAY_VALUE_F</td>
<td>0x001F</td>
<td>Phase value of WS and SSL for forward-padding format</td>
</tr>
</tbody>
</table>
5.5 Structure/Union List

Figure 5.29 shows the Structure/Union Used in the Sample Code.

```c
#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
#endif __BIG

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 5.29 Structure/Union Used in the Sample Code
5.6 Variables

Table 5.4 lists the Global Variables and Table 5.5 lists the const Variables.

Table 5.4 Global Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long</td>
<td>tx_au_data[AUD_NUM][2]</td>
<td>Transmit audio data</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
<tr>
<td></td>
<td>rx_au_data[AUD_NUM]</td>
<td>Receive audio data</td>
<td>i2s_dtc_rx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_rx_r_init</td>
</tr>
<tr>
<td>unsigned long</td>
<td>DTC_VECT_TABLE[256]</td>
<td>DTC vector table</td>
<td>i2s_dtc_init</td>
</tr>
</tbody>
</table>

Table 5.5 const Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>const uint8_t</td>
<td>RSPI_TX_DISABLE</td>
<td>Register setting value for RSPI initialization when transmitting</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
<tr>
<td>const uint8_t</td>
<td>RSPI_TX_ENABLE</td>
<td>Register setting value for canceling RSPI initialization when transmitting</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
<tr>
<td>const uint8_t</td>
<td>RSPI_RX_DISABLE</td>
<td>Register setting value for RSPI initialization when receiving</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
<tr>
<td>const uint8_t</td>
<td>RSPI_RX_ENABLE</td>
<td>Register setting value for canceling RSPI initialization when receiving</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
<tr>
<td>const uint8_t</td>
<td>RSPI_TRX_DISABLE</td>
<td>Register setting value for RSPI initialization when transmitting and receiving</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
<tr>
<td>const uint8_t</td>
<td>RSPI_TRX_ENABLE</td>
<td>Register setting value for canceling RSPI initialization when transmitting and receiving</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
<tr>
<td>const uint8_t</td>
<td>DTC_CHAIN_DISABLE</td>
<td>Register setting value for disabling chain transfer</td>
<td>i2s_dtc_tx_l_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2s_dtc_tx_r_init</td>
</tr>
</tbody>
</table>
5.7 Functions
Table 5.6 lists the Functions Used in the Sample Code.

Table 5.6 Functions Used in the Sample Code

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
</tr>
<tr>
<td>R_INIT_StopModule</td>
<td>Stop processing for active peripheral functions after a reset</td>
</tr>
<tr>
<td>R_INIT_NonExistentPort</td>
<td>Nonexistent port initialization</td>
</tr>
<tr>
<td>R_INIT_Clock</td>
<td>Clock initialization</td>
</tr>
<tr>
<td>i2s_au_data_init</td>
<td>Creating audio data in the on-chip RAM</td>
</tr>
<tr>
<td>i2s_start</td>
<td>I2S communication start processing</td>
</tr>
<tr>
<td>i2s_dtc_init</td>
<td>DTC initialization</td>
</tr>
<tr>
<td>i2s_dtc_tx_l_init</td>
<td>Creating DTC transfer information for L-ch transmission or RSPI0 initialization</td>
</tr>
<tr>
<td>i2s_dtc_tx_r_init</td>
<td>Creating DTC transfer information for R-ch transmission or RSPI1 initialization</td>
</tr>
<tr>
<td>i2s_dtc_rx_l_init</td>
<td>Creating DTC transfer information for L-ch reception</td>
</tr>
<tr>
<td>i2s_dtc_rx_r_init</td>
<td>Creating DTC transfer information for R-ch reception</td>
</tr>
<tr>
<td>i2s_mtu2_init</td>
<td>MTU2a initialization</td>
</tr>
<tr>
<td>i2s_mtu2_ch2_init</td>
<td>MTU2a channel 2 initialization for SCK generation</td>
</tr>
<tr>
<td>i2s_mtu2_ch3_init</td>
<td>MTU2a channel 3 initialization for WS generation</td>
</tr>
<tr>
<td>i2s_mtu2_ch4_init</td>
<td>MTU2a channel 4 initialization for SSL generation</td>
</tr>
<tr>
<td>i2s_rspi_init</td>
<td>RSPI initialization</td>
</tr>
<tr>
<td>i2s_rspi0_init</td>
<td>RSPI0 initialization for L-ch transmission and reception</td>
</tr>
<tr>
<td>i2s_rspi1_init</td>
<td>RSPI1 initialization for R-ch transmission and reception</td>
</tr>
</tbody>
</table>
### 5.8 Function Specifications

The following tables list the sample code function specifications.

| Function          | Outline                                      | Header                        | Declaration                                      | Description                                                                                     | Arguments | Return Value | Remarks                                                                                           |
|-------------------|----------------------------------------------|-------------------------------|-------------------------------------------------|------------------------------------------------------------------------------------------------|
| main              | Main processing                              | i2s.h                         | void main(void)                                 | Call following functions; port initialization, clock initialization, peripheral function initialization (MTU2a, RSPI, and DTC) | None      | None         | None                                                                                             |
| R_INIT_StopModule | Stop processing for active peripheral functions after a reset | r_init_stop_module.h         | void R_INIT_StopModule(void)                  | Configure the setting to enter the module-stop state.                                       | None      | None         | Transition to the module-stop state is not performed in the sample code. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details on this function. |
| R_INIT_NonExistentPort | Nonexistent port initialization             | r_init_non_existent_port.h   | void R_INIT_NonExistentPort(void)             | Initialize port direction registers for ports that do not exist in products with less than 176 pins. | None      | None         | The number of pins in the sample code is set for the 176-pin package (PIN_SIZE=176). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details on this function. |
| R_INIT_Clock      | Clock initialization                         | r_init_clock.h                | void R_INIT_Clock (void)                      | Initialize clocks.                                                                            | None      | None         | The sample code selects processing which uses PLL as the system clock without using the sub-clock. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details on this function. |
i2s_au_data_init
Outline  Creating audio data in the on-chip RAM
Header    i2s.h
Declaration void i2s_au_data_init (void)
Description - Create transmit audio data (tx_au_data).
               - Clear the receive audio data area to 0 (rx_au_data).
Arguments  None
Return Value None

i2s_start
Outline  I2S communication start processing
Header    i2s.h, iodefine.h
Declaration void i2s_start (char i2s_mode)
Description - Enable the RSPI function.
               - Activate the DTC module.
               - Start MTU2a count operation.
Arguments  i2s_mode : Operating mode selection
Return Value None

i2s_dtc_init
Outline  DTC initialization
Header    iodefine.h
Declaration void i2s_dtc_init (char i2s_mode)
Description - Create the DTC vector table.
               - Enable DTC activation by interrupts.
Arguments  i2s_mode : Operating mode selection
Return Value None

i2s_dtc_tx_l_init
Outline  Creating DTC transfer information for L-ch transmission or RSPI0 initialization
Header    i2s.h, iodefine.h
Declaration void i2s_dtc_tx_l_init (char i2s_mode)
Description Transfer transmit data from the on-chip RAM to RSPI0
Arguments  i2s_mode : Operating mode selection
Return Value None

i2s_dtc_tx_r_init
Outline  Creating DTC transfer information for R-ch transmission or RSPI1 initialization
Header    i2s.h, iodefine.h
Declaration void i2s_dtc_tx_r_init (char i2s_mode)
Description Transfer transmit data from the on-chip RAM to RSPI1
Arguments  i2s_mode : Operating mode selection
Return Value None
### i2s_dtc_rx_l_init

**Outline**
Creating DTC transfer information for L-ch reception

**Header**
i2s.h, iodefine.h

**Declaration**
void i2s_dtc_rx_l_init (void)

**Description**
Transfer receive data from RSPI0 to the on-chip RAM.

**Arguments**
None

**Return Value**
None

### i2s_dtc_rx_r_init

**Outline**
Creating DTC transfer information for R-ch reception

**Header**
i2s.h, iodefine.h

**Declaration**
void i2s_dtc_rx_r_init (void)

**Description**
Transfer receive data from RSPI1 to the on-chip RAM.

**Arguments**
None

**Return Value**
None

### i2s_mtu2_init

**Outline**
MTU2a initialization

**Header**
iodefine.h

**Declaration**
void i2s_mtu2_init (void)

**Description**
Configure MTU2a ports.

**Arguments**
None

**Return Value**
None

### i2s_mtu2_ch2_init

**Outline**
MTU2a channel 2 initialization for SCK generation

**Header**
i2s.h, iodefine.h

**Declaration**
void i2s_mtu2_ch2_init (void)

**Description**
Generate SCK from the external clock and output it.

**Arguments**
None

**Return Value**
None

### i2s_mtu2_ch3_init

**Outline**
MTU2a channel 3 initialization for WS generation

**Header**
i2s.h, iodefine.h

**Declaration**
void i2s_mtu2_ch3_init(void)

**Description**
Generate WS from the external clock and output it.

**Arguments**
None

**Return Value**
None

### i2s_mtu2_ch4_init

**Outline**
MTU2a channel 4 initialization for SSL generation

**Header**
i2s.h, iodefine.h

**Declaration**
void i2s_mtu2_ch4_init(void)

**Description**
- Generate SSL from the external clock and output it.
- Enable the compare match interrupt

**Arguments**
None

**Return Value**
None
### i2s_rspi_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>RSPI initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void i2s_rspi_init (char i2s_mode)</td>
</tr>
<tr>
<td>Description</td>
<td>Configure the RSPI.</td>
</tr>
<tr>
<td>Arguments</td>
<td>i2s_mode            : Operating mode selection</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### i2s_rspi0_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>RSPI0 initialization for L-ch transmission and reception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>iodefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void i2s_rspi0_init (char i2s_mode)</td>
</tr>
<tr>
<td>Description</td>
<td>Configure RSPI0.</td>
</tr>
<tr>
<td>Arguments</td>
<td>i2s_mode                                                 : Operating mode selection</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### i2s_rspi1_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>RSPI1 initialization for R-ch transmission and reception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>iodefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void i2s_rspi1_init (char i2s_mode)</td>
</tr>
<tr>
<td>Description</td>
<td>Configure RSPI1.</td>
</tr>
<tr>
<td>Arguments</td>
<td>i2s_mode                                                 : Operating mode selection</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
6 Flowcharts

6.1 Main Processing

Figure 6.1 shows the Main Processing.

```
main

Disable maskable interrupts

Stop processing for active peripheral functions after a reset
R_INIT_StopModule()

Nonexistent port initialization
R_INIT_NonExistPort()

Clock initialization
R_INIT_Clock()

Cancel the module-stop state

Select I2S communication mode

Audio data creation
i2s_au_data_init()

MTU2a initialization
i2s_mtu2_init()

RSPI initialization
i2s_rspi_init(i2s_mode)

DTC initialization
i2s_dtc_init(i2s_mode)

Enable maskable interrupts

I2S communication start processing
i2s_start(i2s_mode)

i flag ← 0

PRCR register ← A502h
PRC1 bit = 1: Enable writing to the registers related to the low power consumption function.

MSTPCRA register
MSTPA9 bit ← 0: The MTU module-stop state is canceled

MSTPCRBB register
MSTPB17 bit ← 0: The RSPI0 module-stop state is canceled
MSTPB16 bit ← 0: The RSPI1 module-stop state is canceled

PRCR register ← A500h
PRC1 bit = 0: Disable writing to the registers related to the low power consumption function.

i2s_mode = TRANSMIT_MODE: Transmit mode
RECEIVE_MODE: Receive mode
TRANSCEIVE_MODE: Transmit and receive mode

Create transmit data in the on-chip RAM.
Clear the receive data area in the on-chip RAM.

Initialize channels 2 to 4 of the MTU2a.

Initialize RSPI0 and RSPI1.

Initialize the DTC.
```

Figure 6.1 Main Processing
### 6.2 Functions

#### 6.2.1 i2s\_au\_data\_init

Figure 6.2 shows the Audio Data Creation.

The `i2s\_au\_data\_init` function creates audio data in the on-chip RAM.

![Audio Data Creation Diagram](image)

- **i2s\_au\_data\_init**
- Create transmit data
  - Write transmit data to `tx\_au\_data`.
- Clear the receive data area
  - Clear whole `rx\_au\_data` to 0.
- **return**

**Figure 6.2 Audio Data Creation**
6.2.2  i2s_start

Figure 6.3 shows the I2S Communication Start Processing.
The i2s_start function enables peripheral functions and starts the I2S communication.

Figure 6.3  I2S Communication Start Processing
6.2.3  i2s_dtc_init

Figure 6.4 shows the DTC Initialization.

The i2s_dtc_init function initializes the DTCa.

```
[Argument]
char i2s_mode: Operating mode selection

Set transfer information addresses in DTC_VECT_TABLE
DTC_VECT_TABLE[157] ← DTC_TX_L: L-ch transmission
DTC_VECT_TABLE[156] ← DTC_TX_R: R-ch transmission
DTC_VECT_TABLE[39] ← DTC_RX_L: L-ch reception
DTC_VECT_TABLE[42] ← DTC_RX_R: R-ch reception

DTCVBR register ← Address in DTC_VECT_TABLE

Create DTC vector table

Create DTC transfer information
i2s_dtc_tx_l_init(i2s_mode)
Create DTC transfer information
i2s_dtc_tx_r_init(i2s_mode)
Create DTC transfer information
i2s_dtc_rx_l_init()
Create DTC transfer information
i2s_dtc_rx_r_init()

Enable DTC activation by MTU2a

IER register
IEN bit ← 1: Interrupt requests enabled

DTCE bit ← 1: DTC activation is enabled

DTCE bit ← 1: DTC activation is enabled

Is operating mode receive mode or transmit and receive mode?

No

Yes

Enable DTC activation by RSPI

return
```

Figure 6.4  DTC Initialization
6.2.4  **i2s_dtc_tx_l_init**

Figure 6.5 to Figure 6.8 show the Creating DTC Transfer Information for L-ch Transmission or RSPI0 Initialization.

The `i2s_dtc_tx_l_init` function creates DTC transfer information for L-ch transmission for transmit mode, or transmit and receive mode, or for RSPI0 initialization for receive mode.

---

**Diagram:**

```
```

**Argument**

*char i2s_mode: Operating mode selection*

**Which operating mode?**

- **Transmit mode**
  - Create DTC transfer information (L-ch transmission/transmit mode)

- **Receive mode**
  - Create DTC transfer information (RSPI0 initialization/receive mode)

- **Transmit and receive mode**
  - Create DTC transfer information (L-ch transmission/transmit and receive mode)

**default**

**return**

**Figure 6.5 Creating DTC Transfer Information for L-ch Transmission or RSPI0 Initialization (1/4)**
Create DTC transfer information (L-ch transmission/transmit mode)

DTC_TX_L[0] : RSPI0 disabled
MRA ← 00h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 00b: 8-bit (byte) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 80h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 1: Chain transfer is enabled
SAR ← Address in RSPI_TX_DISABLE
DAR ← Address in RSPI0.SPCR
CRA ← DTC_TX_L_COUNT + 1
CRB ← FFFFh

DTC_TX_L[1] : RSPI0 enabled
MRA ← 00h
MRB ← 80h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 00b: 8-bit (byte) transfer
MD[1:0] = 00b: Normal transfer mode
SAR ← Address in RSPI_TX_ENABLE
DAR ← Address in RSPI0.SPCR
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[2] : Transmit data transfer
MRA ← 28h
SM[1:0] = 10b: SAR value is incremented after data transfer
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 80h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 1: Chain transfer is enabled
SAR ← Address in tx_au_data[0].L.CH
DAR ← Address in RSPI0.SPRD
CRA ← DTC_TX_L_COUNT
CRB ← FFFFh

DTC_TX_L[3] : Transmit data address transfer
MRA ← 20h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 00h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 0: Chain transfer is disabled
SAR ← Address in DTC_CHAIN_DISABLE
DAR ← Address in DTC_TX_L[0].MRB
CRA ← 2
CRB ← FFFFh

DTC_TX_L[4] : Transmit end processing
MRA ← 00h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 00b: 8-bit (byte) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 00h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 0: Chain transfer is disabled
SAR ← Address in DTC_CHAIN_DISABLE
DAR ← Address in DTC_TX_L[0].MRB
CRA ← 2
CRB ← FFFFh

Figure 6.6 Creating DTC Transfer Information for L-ch Transmission or RSPI0 Initialization (2/4)
Figure 6.7 Creating DTC Transfer Information for L-ch Transmission or RSPI0 Initialization (3/4)
Create DTC transfer information (L-ch transmission/transmit and receive mode)

**DTC_TX_L[0] : RSPI0 disabled**
- MRA ← 00h
- SM[1:0] = 00b: Address in the SAR register is fixed
- SZ[1:0] = 00b: 8-bit (byte) transfer
- MD[1:0] = 00b: Normal transfer mode
- MRB ← 00h
- DM[1:0] = 00b: Address in the DAR register is fixed
- DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
- CHNS = 0: Chain transfer is performed continuously
- CHNE = 0: Chain transfer is disabled
- SAR ← Address in RSPI_TRX_DISABLE
- DAR ← Address in RSPI0.SPCR
- CRA ← DTC_TX_L_COUNT + 1
- CRB ← FFFFh

**DTC_TX_L[1] : RSPI0 enabled**
- MRA ← 00h
- SM[1:0] = 00b: Address in the SAR register is fixed
- SZ[1:0] = 00b: 8-bit (byte) transfer
- MD[1:0] = 00b: Normal transfer mode
- MRB ← 00h
- DM[1:0] = 00b: Address in the DAR register is fixed
- DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
- CHNS = 0: Chain transfer is performed continuously
- CHNE = 0: Chain transfer is disabled
- SAR ← Address in RSPI_TRX_ENABLE
- DAR ← Address in RSPI0.SPCR
- CRA ← DTC_TX_L_COUNT
- CRB ← FFFFh

**DTC_TX_L[2] : Transmit data transfer**
- MRA ← 20h
- SM[1:0] = 10b: SAR value is incremented after data transfer
- SZ[1:0] = 10b: 32-bit (longword) transfer
- MD[1:0] = 00b: Normal transfer mode
- MRB ← 00h
- DM[1:0] = 00b: Address in the DAR register is fixed
- DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
- CHNS = 0: Chain transfer is performed continuously
- CHNE = 0: Chain transfer is disabled
- SAR ← Address in tx_au_data[0][L_Ch]
- DAR ← Address in RSPI0.SPDR
- CRA ← DTC_TX_L_COUNT
- CRB ← FFFFh

**DTC_TX_L[3] : Transmit data address transfer**
- MRA ← 20h
- SM[1:0] = 00b: Address in the SAR register is fixed
- SZ[1:0] = 10b: 32-bit (longword) transfer
- MD[1:0] = 00b: Normal transfer mode
- MRB ← 00h
- DM[1:0] = 00b: Address in the DAR register is fixed
- DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
- CHNS = 0: Chain transfer is performed continuously
- CHNE = 0: Chain transfer is disabled
- SAR ← Address in DTC_CHAIN_DISABLE
- DAR ← Address in DTC_TX_L[0].MRB
- CRA ← 2
- CRB ← FFFFh

**DTC_TX_L[4] : Transmit and receive end processing**
- MRA ← 00h
- SM[1:0] = 00b: Address in the SAR register is fixed
- SZ[1:0] = 00b: 8-bit (byte) transfer
- MD[1:0] = 00b: Normal transfer mode
- MRB ← 00h
- DM[1:0] = 00b: Address in the DAR register is fixed
- DISEL = 0: An interrupt request to the CPU is generated when specified data transfer is completed
- CHNS = 0: Chain transfer is performed continuously
- CHNE = 0: Chain transfer is disabled
- SAR ← Address in DTC_CHAIN_DISABLE
- DAR ← Address in DTC_TX_L[0].MRB
- CRA ← 2
- CRB ← FFFFh

---

Figure 6.8 Creating DTC Transfer Information for L-ch Transmission or RSPI0 Initialization (4/4)
6.2.5 i2s_dtc_tx_r_init

Figure 6.9 to Figure 6.12 show the Creating DTC Transfer Information for R-ch Transmission or RSPI1 Initialization.

The i2s_dtc_tx_r_init function creates DTC transfer information for R-ch transmission for transmit mode, or transmit and receive mode, or for RSPI1 initialization for receive mode.

![Diagram showing the i2s_dtc_tx_r_init function with arguments and decision paths for different operating modes.]

return

Figure 6.9 Creating DTC Transfer Information for R-ch Transmission or RSPI1 Initialization (1/4)
Create DTC transfer information

(D-ch transmission/transmit mode)

DTC_TX_R[0]: RSPI1 disabled
MRA ← 00h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 00b: 8-bit (byte) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 80h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 1: Chain transfer is enabled
SAR ← Address in RSPI1.SPDR
DAR ← Address in RSPI1.SPCR
CRA ← DTC_TX_R_COUNT + 1
CRB ← FFFFh

DTC_TX_R[1]: RSPI1 enabled
MRA ← 00h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 00b: 8-bit (byte) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 80h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 1: Chain transfer is enabled
SAR ← Address in RSPI1.SPDR
DAR ← Address in RSPI1.SPCR
CRA ← DTC_TX_R_COUNT
CRB ← FFFFh

DTC_TX_R[2]: Transmit data transfer
MRA ← 28h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 80h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 1: Chain transfer is enabled
SAR ← Address in tx_au_data[R_CH]
DAR ← Address in RSPI1.SPDR
CRA ← DTC_TX_R_COUNT
CRB ← FFFFh

DTC_TX_R[3]: Transmit data address transfer
MRA ← 20h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 00h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 1: Chain transfer is performed only when the transfer
counter is changed from 1 to 0
CHNE = 1: Chain transfer is enabled
SAR ← Address in DTC_TX_R[2].SAR
DAR ← Address in DTC_TX_R[2].SAR
CRA ← DTC_TX_R_COUNT
CRB ← FFFFh

DTC_TX_R[4]: Transmit end processing
MRA ← 00h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 00b: 8-bit (byte) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 00h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 0: Chain transfer is disabled
SAR ← Address in DTC_CHAIN_DISABLE
DAR ← Address in DTC_TX_R[0].MRB
CRA ← 2
CRB ← FFFFh

Figure 6.10 Creating DTC Transfer Information for R-ch Transmission or RSPI1 Initialization (2/4)
Create DTC transfer information

**DTC_TX_R[0]: RSPI1 disabled**

- **MRA**: 00h
- **SM[1:0]** = 00b: Address in the SAR register is fixed
- **SZ[1:0]** = 00b: 8-bit (byte) transfer
- **MD[1:0]** = 00b: Normal transfer mode

**MRB**: 80h

- **DM[1:0]** = 00b: Address in the DAR register is fixed
- **DISEL**: 0: An interrupt request to the CPU is generated when specified data transfer is completed
- **CHNS**: 0: Chain transfer is performed continuously
- **CHNE**: 0: Chain transfer is disabled

**MRA**: 00h

- **SM[1:0]** = 00b: Address in the SAR register is fixed
- **SZ[1:0]** = 00b: 8-bit (byte) transfer
- **MD[1:0]** = 00b: Normal transfer mode

**MRB**: C0h

- **DM[1:0]** = 00b: Address in the DAR register is fixed
- **DISEL**: 0: An interrupt request to the CPU is generated when specified data transfer is completed
- **CHNS**: 1: Chain transfer is performed only when the transfer counter is changed from 1 to 0
- **CHNE**: 1: Chain transfer is enabled

**SAR**: Address in RSPI_RX_DISABLE

**DAR**: Address in RSPI1.SPCR

**CRA**: DTC_TX_R_COUNT + 1

**CRB**: FFFFh

**DTC_TX_R[1]: RSPI1 enabled**

**MRA**: 00h

- **SM[1:0]** = 00b: Address in the SAR register is fixed
- **SZ[1:0]** = 00b: 8-bit (byte) transfer
- **MD[1:0]** = 00b: Normal transfer mode

**MRB**: C0h

- **DM[1:0]** = 00b: Address in the DAR register is fixed
- **DISEL**: 0: An interrupt request to the CPU is generated when specified data transfer is completed
- **CHNS**: 1: Chain transfer is performed only when the transfer counter is changed from 1 to 0
- **CHNE**: 1: Chain transfer is enabled

**SAR**: Address in RSPI_RX_ENABLE

**DAR**: Address in RSPI1.SPCR

**CRA**: DTC_TX_R_COUNT

**CRB**: FFFFh

**DTC_TX_R[2]: Receive end processing**

**MRA**: 00h

- **SM[1:0]** = 00b: Address in the SAR register is fixed
- **SZ[1:0]** = 00b: 8-bit (byte) transfer
- **MD[1:0]** = 00b: Normal transfer mode

**MRB**: 00h

- **DM[1:0]** = 00b: Address in the DAR register is fixed
- **DISEL**: 0: An interrupt request to the CPU is generated when specified data transfer is completed
- **CHNS**: 0: Chain transfer is performed continuously
- **CHNE**: 0: Chain transfer is disabled

**SAR**: Address in DTC_CHAIN_DISABLE

**DAR**: Address in DTC_TX_R[0].MRB

**CRA**: 2

**CRB**: FFFFh

---

**Figure 6.11 Creating DTC Transfer Information for R-ch Transmission or RSPI1 Initialization (3/4)**
Create DTC transfer information (R-ch transmission/transmit and receive mode)

**DTC_TX_R[0] :** RSPI1 disabled
- \(MRA \leftarrow 00h\)
- \(SM[1:0] = 00b\): Address in the SAR register is fixed
- \(SZ[1:0] = 00b\): 8-bit (byte) transfer
- \(MD[1:0] = 00b\): Normal transfer mode
- \(MRB \leftarrow 80h\)
- \(DM[1:0] = 00b\): Address in the DAR register is fixed
- \(DISEL = 0\): An interrupt request to the CPU is generated when specified data transfer is completed
- \(CHNS = 0\): Chain transfer is performed continuously
- \(CHNE = 1\): Chain transfer is enabled
- \(SAR \leftarrow Address\) in RSPI_TRX_DISABLE
- \(DAR \leftarrow Address\) in RSPI1.SPCR
- \(CRA \leftarrow DTC_TX_R_COUNT + 1\)
- \(CRB \leftarrow FFFFh\)

**DTC_TX_R[1] :** RSPI1 enabled
- \(MRA \leftarrow 00h\)
- \(SM[1:0] = 00b\): Address in the SAR register is fixed
- \(SZ[1:0] = 00b\): 8-bit (byte) transfer
- \(MD[1:0] = 00b\): Normal transfer mode
- \(MRB \leftarrow 80h\)
- \(DM[1:0] = 00b\): Address in the DAR register is fixed
- \(DISEL = 0\): An interrupt request to the CPU is generated when specified data transfer is completed
- \(CHNS = 0\): Chain transfer is performed continuously
- \(CHNE = 1\): Chain transfer is enabled
- \(SAR \leftarrow Address\) in RSPI_TRX_ENABLE
- \(DAR \leftarrow Address\) in RSPI1.SPCR
- \(CRA \leftarrow DTC_TX_R_COUNT\)
- \(CRB \leftarrow FFFFh\)

**DTC_TX_R[2] :** Transmit data transfer
- \(MRA \leftarrow 28h\)
- \(SM[1:0] = 10b\): SAR value is incremented after data transfer
- \(SZ[1:0] = 10b\): 32-bit (longword) transfer
- \(MD[1:0] = 00b\): Normal transfer mode
- \(MRB \leftarrow 80h\)
- \(DM[1:0] = 00b\): Address in the DAR register is fixed
- \(DISEL = 0\): An interrupt request to the CPU is generated when specified data transfer is completed
- \(CHNS = 0\): Chain transfer is performed continuously
- \(CHNE = 1\): Chain transfer is enabled
- \(SAR \leftarrow Address\) in tx_au_data[R_CH]
- \(DAR \leftarrow Address\) in DTC_TX_R[2].SAR
- \(CRA \leftarrow DTC_TX_R_COUNT\)
- \(CRB \leftarrow FFFFh\)

**DTC_TX_R[3] :** Transmit data address transfer
- \(MRA \leftarrow 20h\)
- \(SM[1:0] = 00b\): Address in the SAR register is fixed
- \(SZ[1:0] = 10b\): 32-bit (longword) transfer
- \(MD[1:0] = 00b\): Normal transfer mode
- \(MRB \leftarrow 00h\)
- \(DM[1:0] = 00b\): Address in the DAR register is fixed
- \(DISEL = 0\): An interrupt request to the CPU is generated when specified data transfer is completed
- \(CHNS = 1\): Chain transfer is performed only when the transfer counter is changed from 1 to 0
- \(CHNE = 1\): Chain transfer is enabled
- \(SAR \leftarrow Address\) in DTC_TX_R[2].SAR
- \(DAR \leftarrow Address\) in DTC_TX_L[2].SAR
- \(CRA \leftarrow DTC_TX_R_COUNT\)
- \(CRB \leftarrow FFFFh\)

**DTC_TX_R[4] :** Transmit and receive end processing
- \(MRA \leftarrow 00h\)
- \(SM[1:0] = 00b\): Address in the SAR register is fixed
- \(SZ[1:0] = 00b\): 8-bit (byte) transfer
- \(MD[1:0] = 00b\): Normal transfer mode
- \(MRB \leftarrow 00h\)
- \(DM[1:0] = 00b\): Address in the DAR register is fixed
- \(DISEL = 0\): An interrupt request to the CPU is generated when specified data transfer is completed
- \(CHNS = 0\): Chain transfer is performed continuously
- \(CHNE = 0\): Chain transfer is disabled
- \(SAR \leftarrow Address\) in DTC_CHAIN_DISABLE
- \(DAR \leftarrow Address\) in DTC_TX_R[0].MRB
- \(CRA \leftarrow 2\)
- \(CRB \leftarrow FFFFh\)

Figure 6.12 Creating DTC Transfer Information for R-ch Transmission or RSPI1 Initialization (4/4)
6.2.6 i2s_dtc_rx_l_init

Figure 6.13 shows the Creating DTC Transfer Information for L-ch Reception.

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

```
DTC_RX_L[0]: Receive data transfer
MRA ← 20h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 88h
DM[1:0] = 10b: DAR value is incremented after data transfer
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 1: Chain transfer is enabled
SAR ← Address in RSPI0.SPDR
DAR ← Address in rx_au_data[0][L_CH]
CRA ← DTC_RX_L_COUNT
CRB ← FFFFh

DTC_RX_L[1]: Receive data address transfer
MRA ← 20h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRB ← 00h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 0: Chain transfer is disabled
SAR ← Address in DTC_RX_L[0].DAR
DAR ← Address in DTC_RX_R[0].DAR
CRA ← DTC_RX_L_COUNT
CRB ← FFFFh
```

Figure 6.13 Creating DTC Transfer Information for L-ch Reception
6.2.7 i2s_dtc_rx_r_init

Figure 6.14 shows the Creating DTC Transfer Information for R-ch Reception.

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

```
Create DTC transfer information

i2s_dtc_rx_r_init

DTC_RX_R[0] : Receive data transfer
MRA ← 20h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRA ← 88h
DM[1:0] = 10b: DAR value is incremented after data transfer
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 1: Chain transfer is enabled
SAR ← Address in RSPI1.SPDR
DAR ← Address in rx.au_data[0][R_CH]
CRA ← DTC_RX_R_COUNT
CRB ← FFFFh

DTC_RX_R[1] : Receive data address transfer
MRA ← 20h
SM[1:0] = 00b: Address in the SAR register is fixed
SZ[1:0] = 10b: 32-bit (longword) transfer
MD[1:0] = 00b: Normal transfer mode
MRA ← 00h
DM[1:0] = 00b: Address in the DAR register is fixed
DISEL = 0: An interrupt request to the CPU is generated
when specified data transfer is completed
CHNS = 0: Chain transfer is performed continuously
CHNE = 0: Chain transfer is disabled
SAR ← Address in DTC_RX_R[0].DAR
DAR ← Address in DTC_RX_L[0].DAR
CRA ← DTC_RX_R_COUNT
CRB ← FFFFh

return
```

Figure 6.14 Creating DTC Transfer Information for R-ch Reception
### 6.2.8 i2s_mtu2_init

Figure 6.15 shows the MTU2a Initialization.

The i2s_mtu2_init function initializes the MTU2a.

```
Figure 6.15 MTU2a Initialization
```

```
6.2.8 i2s_mtu2_init

Figure 6.15 shows the MTU2a Initialization.

The i2s_mtu2_init function initializes the MTU2a.

```
```
6.2.9 i2s_mtu2_ch2_init

Figure 6.16 shows the MTU2a Channel 2 Initialization for SCK Generation.

The i2s_mtu2_ch2_init function configures the setting to generate SCK using MTU2a channel 2.

```
i2s_mtu2_ch2_init

Set MTU2a channel 2 operating mode
TCR register ← 24h
TPSC[2:0] bits = 100b: External clock: counts on MTCLKA pin input
CKE[1:0] bits = 00b: Count at rising edge
CCLR[2:0] bits = 001b: TCNT cleared by TGRA compare match

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

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

TIER register ← 00h
TGIEA bit = 0: Interrupt requests (TGIA) disabled
TGIEB bit = 0: Interrupt requests (TGIB) disabled

Set count cycle
TGRA ← SCK_CYCLE_VALUE

Set duty ratio
TGRB ← SCK_CYCLE_VALUE / 2

Set counter initial value
TCNT ← SCK_CYCLE_VALUE - 1

return
```

Figure 6.16 MTU2a Channel 2 Initialization for SCK Generation
6.2.10  i2s_mtu2_ch3_init

Figure 6.17 shows the MTU2a Channel 3 Initialization for WS Generation.

The i2s_mtu2_ch3_init function configures the setting to generate WS using MTU2a channel 3.

```
// Function to configure MTU2a Channel 3 for WS generation
void i2s_mtu2_ch3_init()
{
    // Set MTU2a channel 3 operating mode
    TCR register ← 26h
    TPSC[2:0] bits = 110b: External clock: Count on MTCLKA pin input
    CKEG[1:0] bits = 00b: Count at rising edge
    CCLR[2:0] bits = 001b: TCNT cleared at TGRA compare-match

    // Set count cycle
    TMDR register ← 02h
    MD[3:0] bits = 0010b: PWM mode 1

    // Set compare-match timing
    TGRA ← WS_SSL_CYCLE_VALUE

    // Set duty ratio
    TIORH register ← 65h
    IOA[3:0] bits = 0101b: Initial output is high output, low output at compare match
    IOB[3:0] bits = 0110b: Initial output is high output, high output at compare match

    // Set compare-match timing
    TIORL register ← 00h
    IOC[3:0] bits = 0000b: Output disabled
    IOD[3:0] bits = 0000b: Output disabled

    // Set counter initial value
    TCNT ← WS_SSL_CYCLE_VALUE - 1

    // Return
    return
}
```

Figure 6.17  MTU2a Channel 3 Initialization for WS Generation
6.2.11 i2s_mtu2_ch4_init

Figure 6.18 shows the MTU2a Channel 4 Initialization for SSL Generation.

The i2s_mtu2_ch4_init function configures the setting to generate SSL using MTU2a channel 4.

```
set i2s_mtu2_ch4_init

set MTU2a channel 4 operating mode
TCR register ← 26h
TPSC[2:0] bits = 110b: External clock: Count on MTCLKA pin input
CKEG[1:0] 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 is high output, low output at compare match
IOB[3:0] bits = 0110b: Initial output is high output, high 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

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

set count cycle

set duty ratio

set compare-match timing
TGRA ← WS_SSL_CYCLE_VALUE / 2

Figure 6.18 MTU2a Channel 4 Initialization for SSL Generation
```
6.2.12  **i2s_rspi_init**

Figure 6.19 shows the RSPI Initialization.

The `i2s_rspi_init` function initializes the RSPI.

```plaintext
i2s_rspi_init

RSPI0 initialization
i2s_rspi0_init(i2s_mode)

RSPI1 initialization
i2s_rspi1_init(i2s_mode)

return
```

[Argument]

char i2s_mode: Operating mode selection

Initialize RSPI0 for L-ch transmission and reception

Initialize RSPI1 for R-ch transmission and reception

Figure 6.19  RSPI Initialization
6.2.13 i2s_rspi0_init

Figure 6.20 and Figure 6.21 show the RSPI0 Initialization for L-ch Transmission and Reception.

The i2s_rspi0_init function initializes the RSPI0.

[Diagram]

i2s_rspi0_init

- [Argument]
  char i2s_mode: Operating mode selection

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

- Set RSPI0 operating mode
  SPPCR register ← 00h
  SPLP bit = 0: Normal mode
  SPLP2 bit = 0: Normal mode

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

- 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: SPDR is accessed in longwords

- SPCR2 register ← 00h
  SPPE bit = 0: Does not add the parity bit to transmit data and does not check the parity bit of receive data
  SPIIE bit = 0: Disable the generation of idle interrupt requests

- SPCMD0 register ← 0203h
  CPHA bit = 1: Data variation on odd edge, data sampling on even edge
  CPOL bit = 1: RSPCK is high when idle
  SPB[3:0] bits = 0010b: Transfer data length → 32 bits
  LSBF bit = 0: MSB first

- Set Input/Output pins
  PORTA.PMR register ← 00h
  B7 bit = 0: Use MISOA as a general I/O pin
  B6 bit = 0: Use MOISA as a general I/O pin
  B5 bit = 0: Use RSPCKA as a general I/O pin
  B4 bit = 0: Use SSLA0 as a general I/O pin

  PORTA.ODR1 register ← 00h
  B6 bit = 0: CMOS output

  PORTA.PDR register ← 80h
  B7 bit = 1: Function as an output pin
  B6 bit = 0: Function as an input pin
  B5 bit = 0: Function as an input pin
  B4 bit = 0: Function as an input pin

  PWPR register ← 40h
  BOWI bit = 0: Writing to the PFSWE bit is enabled
  PFSWE bit = 1: Writing to the PFS register is enabled

  PA7PFS register ← 00h
  PSEL[4:0] bits = 01101b: Use as MISOA

  PA6PFS register ← 00h
  PSEL[4:0] bits = 01101b: Use as MOSIA

  PA5PFS register ← 00h
  PSEL[4:0] bits = 01101b: Use as RSPCKA

  PA4PFS register ← 00h
  PSEL[4:0] bits = 01101b: Use as SSLA0

  PWPR register ← 80h
  BOWI bit = 1: Writing to the PFSWE bit is disabled
  PFSWE bit = 0: Writing to the PFS register is disabled

Figure 6.20 RSPI0 Initialization for L-ch Transmission and Reception (1/2)
Which operating mode?

Transmit mode

PORTA.PMR register ← B0h
B7 bit = 1: Use MISOA as an I/O port for peripheral functions
B6 bit = 0: Use MOISA as a general I/O pin
B5 bit = 1: Use RSPCKA as an I/O port for peripheral functions
B4 bit = 1: Use SSLA0 as an I/O port for peripheral functions

Receive mode

PORTA.PMR register ← 70h
B7 bit = 0: Use MISOA as a general I/O pin
B6 bit = 1: Use MOISA as an I/O port for peripheral functions
B5 bit = 1: Use RSPCKA as an I/O port for peripheral functions
B4 bit = 1: Use SSLA0 as an I/O port for peripheral functions

Transmit and receive mode

PORTA.PMR register ← F0h
B7 bit = 1: Use MISOA as an I/O port for peripheral functions
B6 bit = 1: Use MOISA as an I/O port for peripheral functions
B5 bit = 1: Use RSPCKA as an I/O port for peripheral functions
B4 bit = 1: Use SSLA0 as an I/O port for peripheral functions

default

return

Figure 6.21  RSPI0 Initialization for L-ch Transmission and Reception (2/2)
### 6.2.14 i2s_rspi1_init

Figure 6.22 to Figure 6.23 show the RSPI1 Initialization for R-ch Transmission and Reception.

The `i2s_rspi1_init` function initializes the RSPI1.

```plaintext
[i2s_rspi1_init]
char i2s_mode: Operating mode selection

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

SPPCR register ← 00h
SPLP bit = 0: Normal mode
SPLL2 bit = 0: Normal mode

Set RSPI1 operation mode

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

SPCR2 register ← 00h
SPPE bit = 0: Does not add the parity bit to transmit data and does not check the
parity bit of receive data
SPIIE bit = 0: Disable the generation of idle interrupt requests

SPCMD0 register ← 0203h
CPHA bit = 1: Data variation on odd edge, data sampling on even edge
CPOL bit = 1: RSPCK is high when idle
SPB[3:0] bits = 0010b: Transfer data length
LBF bit = 0: MSB first

Set Input/Output pins

PORTE.PMR register ← 00h
B5 bit = 0: Use RSPCKB as a general I/O pin
B4 bit = 0: Use SSLB0 as a general I/O pin
B3 bit = 0: Use MISOB as a general I/O pin
B2 bit = 0: Use MOSIB as a general I/O pin

PORTE.ODR0 register ← 00h
B6 bit = 0: CMOS output

PORTE.PDR register ← 04h
B5 bit = 0: Function as an input pin
B4 bit = 0: Function as an input pin
B3 bit = 1: Function as an output pin
B2 bit = 0: Function as an input pin

PWPR register ← 40h
BOWI bit = 0: Writing to the PFSWE bit is enabled
PFSWE bit = 1: Writing to the PFS register is enabled

PE5PFS register ← 0Dh
PSEL[4:0] bits = 01101b: Use as RSPCKB

PE4PFS register ← 0Dh
PSEL[4:0] bits = 01101b: Use as SSLB0

PE3PFS register ← 0Dh
PSEL[4:0] bits = 01101b: Use as MISOB

PE2PFS register ← 0Dh
PSEL[4:0] bits = 01110b: Use as MOSIB

PWPR register ← 80h
BOWI bit = 1: Writing to the PFSWE bit is disabled
PFSWE bit = 0: Writing to the PFS register is disabled
```

Figure 6.22 RSPI1 Initialization for R-ch Transmission and Reception (1/2)
Which Operating mode?

Transmit mode

Set port modes

PORTE.PMR register ← 38h
B5 bit = 1: Use RSPCKB as an I/O port for peripheral functions
B4 bit = 1: Use SSLB0 as an I/O port for peripheral functions
B3 bit = 1: Use MISOB as an I/O port for peripheral functions
B2 bit = 0: Use MOSIB as a general I/O pin

Receive mode

Set port modes

PORTE.PMR register ← 31h
B5 bit = 1: Use RSPCKB as an I/O port for peripheral functions
B4 bit = 1: Use SSLB0 as an I/O port for peripheral functions
B3 bit = 0: Use MISOB as a general I/O pin
B2 bit = 1: Use MOSIB as an I/O port for peripheral functions

Transmit and receive mode

Set port modes

PORTE.PMR register ← 3Ch
B5 bit = 1: Use RSPCKB as an I/O port for peripheral functions
B4 bit = 1: Use SSLB0 as an I/O port for peripheral functions
B3 bit = 1: Use MISOB as an I/O port for peripheral functions
B2 bit = 1: Use MOSIB as an I/O port for peripheral functions

default

return

Figure 6.23 RSPI Initialization for R-ch Transmission and Reception (2/2)
7 Appendix

When the Renesas Starter Kit for RX63N is used with the sample code, change the following two places on the board.

Figure 7.1 and Figure 7.2 show changes on the board.

**Figure 7.1 Change on the Board when Using P14 with MTCLKA**

**Figure 7.2 Change on the Board when Using P17 with MTIOC3A**
8 Sample Code

Sample code can be downloaded from the Renesas Electronics website.

9 Reference Documents

User’s Manual: Hardware
RX63N Group, RX631 Group User’s Manual: Hardware Rev.1.50 (R01UH0041EJ)
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools
RX Family C/C++ Compiler Package V.1.01 User’s Manual Rev.1.00 (R20UT0570EJ)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

Inquiries
http://www.renesas.com/contact/
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Aug. 1, 2013</td>
<td>First edition issued</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
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. 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.

2. 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.

3. 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, expressed or implied, in Renesas Electronics product(s) is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

6. 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.

7. 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 against the possibility of physical injury, and injury or damage caused by the 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 systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Use Renesas Electronics products in compliance with all applicable laws and regulations that require 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.

9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

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 confirm that 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.