RX Family

SCIFA Clock Synchronous Single Master Control Module Using Firmware Integration Technology

R01AN2280EJ0109
Rev.1.09
Sep 30, 2016

Introduction

This application note describes a clock synchronous single master control module, which uses clock synchronous serial communication over the serial communication interface with FIFO (SCIFA) on RX Family microcontrollers, and explains its use. The module is a clock synchronous single master control module using Firmware Integration Technology (FIT). It is referred to below as the SCIFA FIT module. Other similar function control modules using FIT are referred to as FIT modules or as "function name" FIT modules.

SPI mode single master control can be enabled by adding slave device selection control by means of port control.

The SCIFA FIT module implements single master basic control. Use the SCIFA FIT module to create software for controlling slave devices.

Target Device

Supported microcontroller
RX64M Group
RX71M Group

Device on which operation has been confirmed
Renesas Electronics R1EX25xxx Series Serial EEPROM, 16 Kbit
Macronix International MX25/66L family serial NOR flash memory, 32 Mbit

When applying the information in this application note to a microcontroller other than the above, modifications should be made as appropriate to match the specification of the microcontroller and careful evaluation performed.

Note that the expression "RX Family microcontroller" is used in the discussion that follows for convenience as the target devices span multiple product groups.

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833EU)
- Board Support Package Module Using Firmware Integration Technology (R01AN1685EU)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723EU)
- Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826EJ)

Contents

1. Overview

The SCIFA built into the RX Family microcontroller is used to implement clock synchronous control. SPI mode single master control can be enabled by adding slave device selection control by means of port control.

Table 1.1 lists the peripheral devices used and their applications, and figure 1.1 shows a usage example.

The functions of the module are described briefly below.

- Block type device driver for clock synchronous single master using the SCIFA, with the RX Family microcontroller as the master device
- The SCIFA operates in clock synchronous serial communication mode. It can control one or more channels specified by the user.
- Reentrancy from a different channel is possible.
- Slave device selection control is unsupported.
  Slave device selection control must be implemented separately by means of port control.
- Operation with both big-endian and little-endian data order is supported.
- Data is transferred in MSB-first format.
- Only software transfers are supported.
  A separate DMAC or DTC transfer program is required to perform DMAC transfer or DTC transfer.

Table 1.1 Peripheral Devices Used and Their Uses

Peripheral Device	Use
SCIFA	Clock synchronous (three-line) serial: Single or multiple channels (required)
Port	For slave device selection control signals: A number of ports equal to the number of devices used are necessary (required). Not used by SCIFA FIT module.

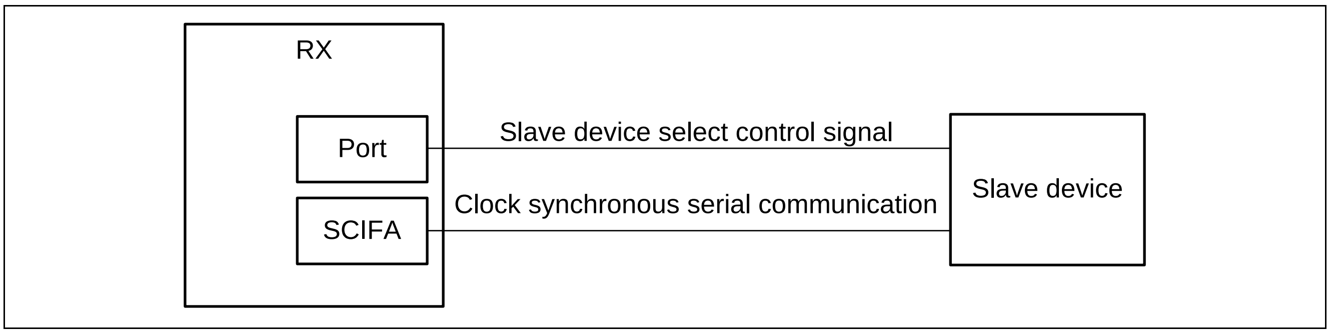

RX
Port
SCIFA
Slave device select control signal
Clock synchronous serial communication
Slave device

Figure 1.1 Sample Configuration

1.1 SCIFA FIT Module

The SCIFA FIT module can be combined with other FIT modules for easy integration into the target system.

The functions of the SCIFA FIT module can be incorporated into software programs by means of APIs. For information on incorporating the SCIFA FIT module into projects, see 2.9, "Adding Driver to Your Project", and 2.10, "Peripheral Functions and Modules Other than SCIFA".

1.2 Overview and Memory Size of APIs

1.2.1 Overview of APIs

Table 1.2 lists the API functions of the SCIFA FIT module.

Table 1.2 API Functions

Function Name	Description
R_SCIFA_SMstr_Open()	Driver initialization processing
R_SCIFA_SMstr_Close()	Driver end processing
R_SCIFA_SMstr_Control()	Driver control (bit rate) setting processing
R_SCIFA_SMstr_Write()*1	Single master transmit processing
R_SCIFA_SMstr_Read()*1	Single master receive processing
R_SCIFA_SMstr_WriteRead()*1	Single master transmit/receive (full duplex communication) processing
R_SCIFA_SMstr_Get_BuffRegAddress()	FTDR register and FRDR register address acquisition processing
R_SCIFA_SMstr_Int_Txif_Ier_Clear()	TXIF transmit interrupt request disable processing
R_SCIFA_SMstr_Int_Rxif_Ier_Clear()	RXIF receive interrupt request disable processing
R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set()	DMAC/DTC transmit-end flag setting processing
R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set()	DMAC/DTC receive-end flag setting processing
R_SCIFA_SMstr_GetVersion()	Driver version information acquisition processing
R_SCIFA_SMstr_Set_LogHdlAddress()	LONGQ FIT module handler address setting processing
R_SCIFA_SMstr_Log()	Error log acquisition processing using LONGQ FIT module
R_SCIFA_SMstr_1ms_Interval()*2	Interval timer count processing

Notes: 1. To speed up SCIFA control, 32-bit access is used for the FTDR and FRDR registers. Align the start address with a 4-byte boundary when specifying transmit and receive data storage buffer pointers.

2. This function must be called at 1 ms intervals, using a hardware or software timer, in order to implement timeout detection when using DMAC transfer or DTC transfer.

1.2.2 Operating Environment and Memory Sizes

(1) RX64M

Table 1.3 lists the conditions under which operation has been confirmed, and table 1.4 lists the required memory sizes for the SCIFA FIT module.

The memory sizes listed apply when the default settings listed in 2.6, Compile Settings, are used. The memory sizes differ according to the definitions selected.

Table 1.3 Operation Confirmation Conditions

Item	Contents
MCU used	RX64M Group (program ROM: 4 MB, RAM: 512 KB)
Operating frequency	ICLK: 120 MHz, PCLKA: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e^2 studio V3.1.0.24
C compiler	Renesas Electronics C/C++ compiler for RX Family V.2.01.00
	Compiler options: The integrated development environment default settings are used, with the following option added. -lang = c99
Endian order	Big-endian/Little-endian
Module version	Ver. 1.08
Board used	R0K50564MSxxxBE (Renesas Starter Kit for RX64M)

Table 1.4 Required Memory Sizes

Memory	Size	Remarks
ROM	3,288 bytes (Little-endian)	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c Under confirmation conditions listed above
RAM	22 bytes (Little-endian)	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c Under confirmation conditions listed above
Max. user stack	76 bytes	
Max. interrupt stack	4 bytes	Only when DMAC transfer or DTC transfer is specified

The required memory sizes differ according to the C compiler version and the compile conditions. The above memory sizes also differ according to endian mode.

(2) RX71M

Table 1.5 lists the conditions under which operation has been confirmed, and table 1.6 lists the required memory sizes for the SCIFA FIT module.

The memory sizes listed apply when the default settings listed in 2.6, Compile Settings, are used. The memory sizes differ according to the definitions selected.

Table 1.5 Operation Confirmation Conditions

Item	Contents
MCU used	RX71M Group (program ROM: 4 MB, RAM: 512 KB)
Operating frequency	ICLK: 240 MHz, PCLKA: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e^2 studio V3.1.2.09
C compiler	Renesas Electronics C/C++ compiler for RX Family V.2.01.00
	Compiler options: The integrated development environment default settings are used, with the following option added. -lang = c99
Endian order	Big-endian/Little-endian
Module version	Ver. 1.08
Board used	R0K50571MSxxxBE (Renesas Starter Kit for RX71M)

Table 1.6 Required Memory Sizes

Memory	Size	Remarks
ROM	3,288 bytes (Little-endian)	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c Under confirmation conditions listed above
RAM	22 bytes (Little-endian)	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c Under confirmation conditions listed above
Max. user stack	76 bytes	
Max. interrupt stack	4 bytes	Only when DMAC transfer or DTC transfer is specified

The required memory sizes differ according to the C compiler version and the compile conditions. The above memory sizes also differ according to endian mode.

1.3 Related Application Notes

The applications notes that are related to this application note are listed below. Reference should also be made to those application notes.

- RX Family LONGQ Module Using Firmware Integration Technology (R01AN1880EU)
- RX Family DMA Controller DMACA Control Module Using Firmware Integration Technology (R01AN2063EJ)
- RX Family DTC Module Using Firmware Integration Technology (R01AN1819EJ)
- RX Family Compare Match Timer Module Using Firmware Integration Technology (R01AN1856EU)
- RX Family EEPROM Access Clock Synchronous control module Using Firmware Integration Technology (R01AN2325EJ)
- RX Family General Purpose Input/Output Driver Module Using Firmware Integration Technology (R01AN1721EU)
- RX Family Multi-Function Pin Controller Module Using Firmware Integration Technology (R01AN1724EU)
- RX Family Serial Flash memory Access Clock Synchronous control module Firmware Integration Technology (R01AN2662EJ)
-

1.4 Hardware Settings

1.4.1 Hardware Configuration Example

Figure 1.2 is a connection diagram. To achieve high-speed operation, consider adding damping resistors or capacitors to improve the circuit matching of the various signal lines.

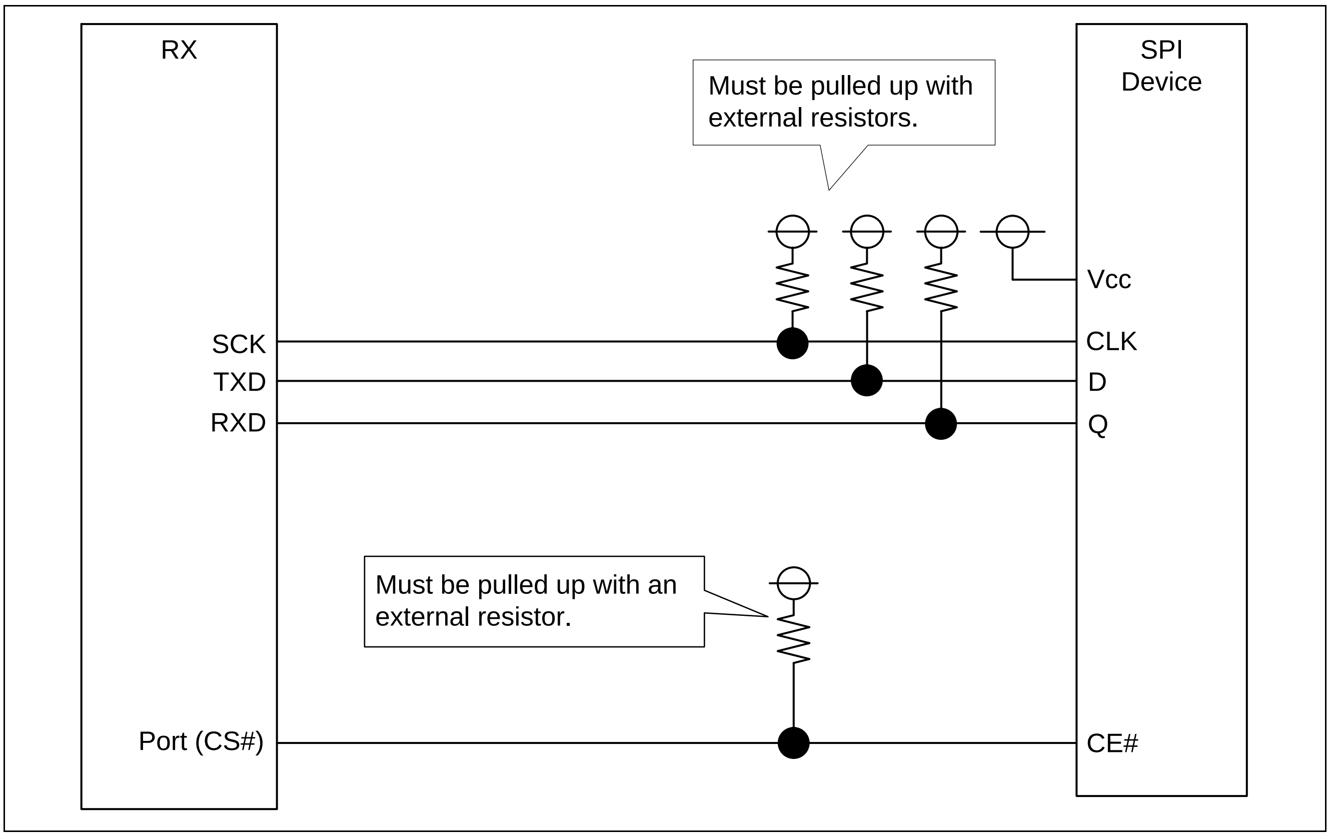

RX
SPI Device
Must be pulled up with external resistors.
Vcc
SCK
TXD
RXD
CLK
D
Q
Must be pulled up with an external resistor.
Port (CS#)
CE#

Figure 1.2 Sample Wiring Diagram for a RX Family MCU SCIFA and a SPI Slave Device

1.4.2 List of Pins

Table 1.7 lists the pins that are used and their uses.

Table 1.7 List of Pins Used

Pin Name	I/O	Description
SCK	Output	Clock output
TXD	Output	Master data output
RXD	Intput	Master data input
Port (Port(CS#) of figure 1.2)	Output	Slave device select output Not used by SCIFA FIT module.

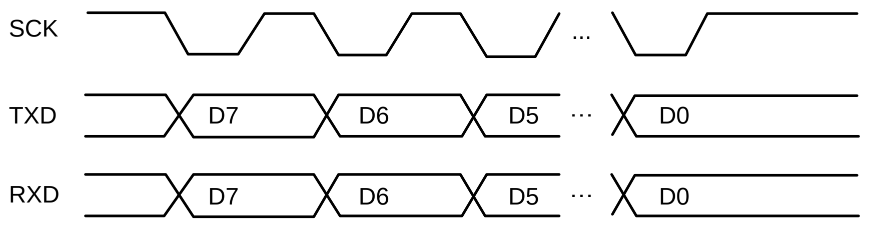
1.5 Software

1.5.1 Operation Overview

Utilizing the clock synchronous serial communication functionality of the SCIFA, clock synchronous single master control (single master transmit, single master receive, or single master transmit/receive) is implemented using the internal clock.

1.5.2 Controllable Slave Devices

Slave devices that support SPI mode 3 (CPOL = 1, CPHA = 1), illustrated in figure 1.3, can be controlled by the module.



- MCU → Slave device transmission: Transmission of transmit data is started on the falling edge of the transfer clock.
- Slave device → MCU reception: The receive data is taken in on the rising edge of the transfer clock.
- MSB-first mode transfer

The level of the SCK pin is held high when no transfer processing is in progress.

Figure 1.3 Timing of Controllable Slave Devices

Refer to the User's Manual: Hardware of the microcontroller and the data sheet of the slave device to determine the usable serial clock frequencies.

1.5.3 Slave Device CE# Pin Control

The SCIFA FIT module does not control the CE# pin of the slave device. To control a slave device, functionality to control the CE# pin of the slave device must be added separately.

Control is implemented by establishing a connection to the ports of the microcontroller and using the microcontroller's general port output for control.

In addition, it is necessary to provide a sufficient time interval (slave device CE# setup time) from the falling edge of the slave device's CE# (microcontroller port (CS#)) signal to the falling edge of the slave device's CLK (microcontroller SCK) signal.

In like manner, it is necessary to provide a sufficient time interval (slave device CE# hold time) from the rising edge of the slave device's CLK (microcontroller SCK) signal to the rising edge of the slave device's CE# (microcontroller port (CS#)).

Check the data sheet of the slave device and set the software wait time to match the system characteristics.

1.5.4 Software Structure

Figure 1.4 shows the software structure.

Use the SCIFA FIT module to create software for controlling slave devices.

Note that sample software for controlling slave devices is available for download.

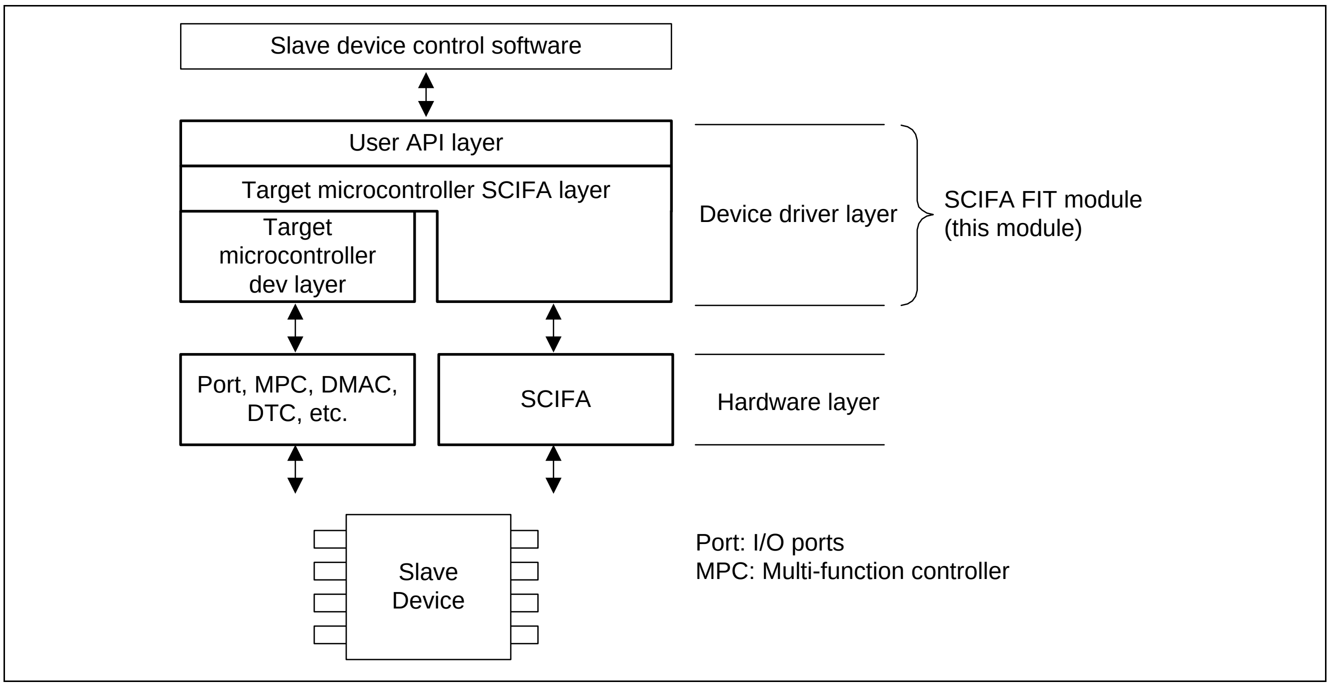

Slave device control software
User API layer
Target microcontroller SCIFA layer
Target microcontroller dev layer
Device driver layer
SCIFA FIT module (this module)
Port, MPC, DMAC, DTC, etc.
SCIFA
Hardware layer
Slave Device
Port: I/O ports
MPC: Multi-function controller

Figure 1.4 Software Structure

(a) User API layer (r_scifa_smstr.c)
This is the SCIFA clock synchronous single master control segment, which is not dependent on the specifications of the microcontroller or the SCIFA.
It also includes transfer start setting processing required for DMAC control or DTC control. It can be used in combination with the DMAC FIT module or DTC FIT module.

(b) Target microcontroller SCIFA layer (r_scifa_smstr_target.c)
This is the SCIFA resource control segment.
Separate versions are provided to accommodate different channel counts or SCIFA specifications.

(c) Target microcontroller dev layer (r_scifa_smstr_target_dev_port.c)
This segment controls functions such as I/O ports (GPIO) other than those of the SCIFA and the multi-function pin controller (MPC). The GPIO FIT module and MPC FIT module can be used. In addition, other FIT modules must be incorporated into the system as necessary.

(d) Control software for slave device
Sample code for controlling Renesas Electronics R1EX25xxx Series serial EEPROM (R01AN2325EJ) is provided as an example for reference. The SPI serial EEPROM Control Software has driver interface functions (r_eeprom_spi_drvif_devX.c: X=0 or 1) to incorporate the RSPI FIT module.

1.5.5 Data Buffers and Transmit/Receive Data

The SCIFA FIT module is a block type device driver that sets transmit and receive data pointers as arguments. The arrangement of data in the data buffer in RAM and the transmit and receive sequences are illustrated below. Regardless of the endian mode and the serial communication function, data is transmitted in the order in which it is arranged in the transmit data buffer, and it is written to the receive data buffer in the order in which it is received.

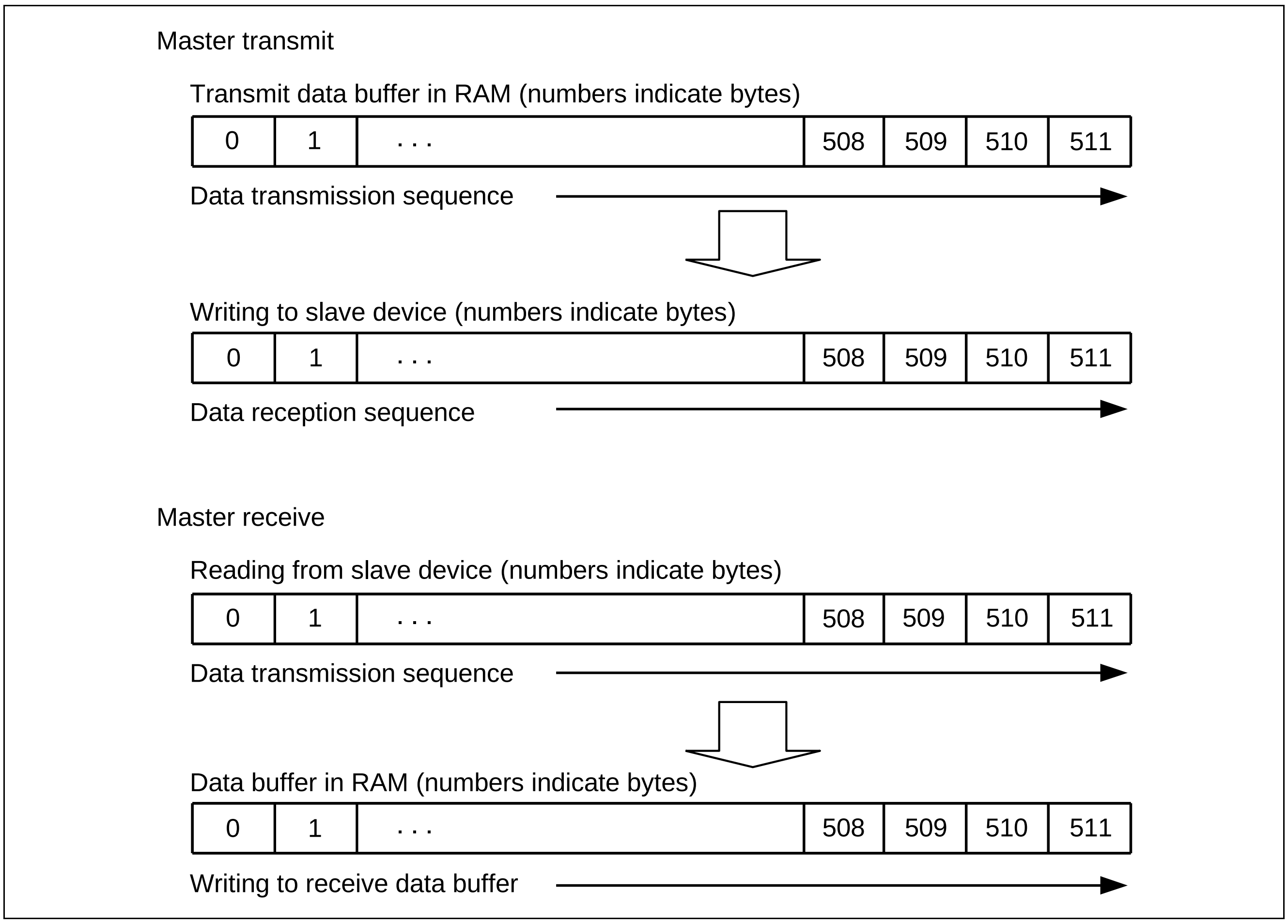

Master transmit
Transmit data buffer in RAM (numbers indicate bytes)
0
1
. . .
508
509
510
511
Data transmission sequence
Writing to slave device (numbers indicate bytes)
0
1
. . .
508
509
510
511
Data reception sequence
Master receive
Reading from slave device (numbers indicate bytes)
0
1
. . .
508
509
510
511
Data transmission sequence
Data buffer in RAM (numbers indicate bytes)
0
1
. . .
508
509
510
511
Writing to receive data buffer

Figure 1.5 Data Buffers and Transmit/Receive Data

1.5.6 State Transition Diagram

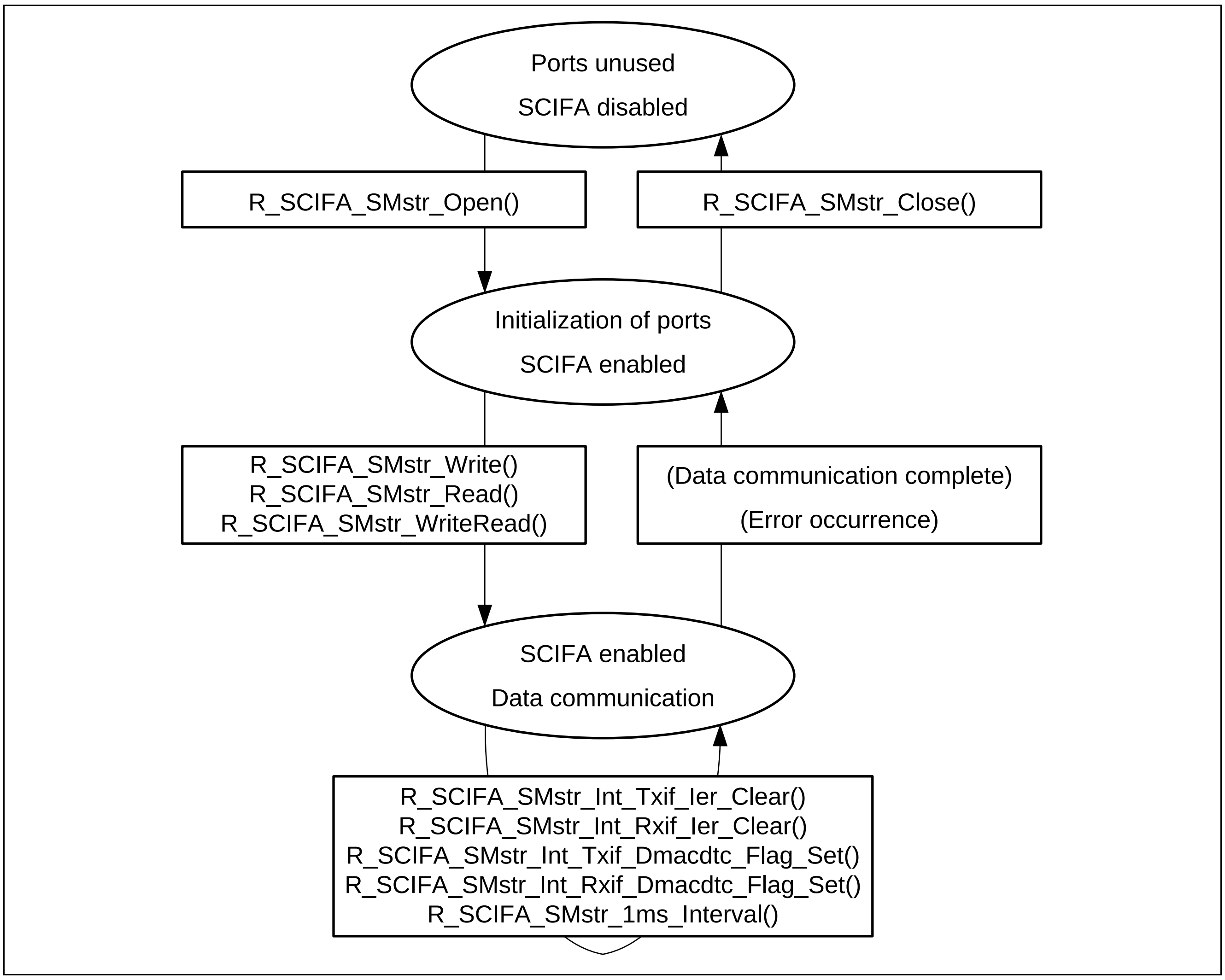

Ports unused
SCIFA disabled
R_SCIFA_SMstr_Open()
R_SCIFA_SMstr_Close()
Initialization of ports
SCIFA enabled
R_SCIFA_SMstr_Write()
R_SCIFA_SMstr_Read()
R_SCIFA_SMstr_WriteRead()
(Data communication complete)
(Error occurrence)
SCIFA enabled
Data communication
R_SCIFA_SMstr_Int_Txif_Ier_Clear()
R_SCIFA_SMstr_Int_Rxif_Ier_Clear()
R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set()
R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set()
R_SCIFA_SMstr_1ms_Interval()

Figure 1.6 State Transition Diagram

2. API Information

The names of the APIs of the SCIFA FIT module follow the Renesas API naming standard.

2.1 Hardware Requirements

The microcontroller used must support the following functionality.

- SCIFA

2.2 Software Requirements

This driver is dependent on the following packages.

- r_bsp
- r_cgc_rx (only on RX Family microcontrollers that require the clock generation circuit (CGC) FIT module)
- r_dmaca_rx (only when using the DMACA FIT module for DMAC transfers)
- r_dtc_rx (only when using the DTC FIT module for DTC transfers)
- r_cmt_rx (only when using DMAC transfer or DTC transfer and the compare match timer (CMT) FIT module) Another timer or a software timer can be used instead.
- r_gpio_rx (only when using the GPIO and MPC FIT modules to control the GPIO)
- r_mpc_rx (only when using the GPIO and MPC FIT modules to control the MPC)

2.3 Supported Toolchain

The operation of SCIFA FIT module has been confirmed with the toolchain listed in 1.2.2.

2.4 Header Files

All the API calls and interface definitions used are listed in r_scifa_smstr_rx_if.h.

Configuration options for individual builds are selected in r_scifa_smstr_rx_config.h and r_scifa_smstr_rx_pin_config.h. The include statements should be in the following order.

```
#include "r_scifa_smstr_rx_if.h"
```

2.5 Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.6 Compile Settings

The configuration option settings for the SCIFA FIT module are specified in r_scifa_smstr_rx_config.h and r_scifa_smstr_rx_pin_config.h.

The option names and setting values are described below.

Configuration options in *r_scifa_smstr_rx_config.h*	
#define SCIFA_SMSTR_CFG_USE_FIT Note: The default value is "enabled".	Selects whether or not the SCIFA FIT module is used in a BSP environment. When this option is set to "disabled", control of FIT modules such as r_bsp is disabled. Also, the equivalent processing must be incorporated separately. For details, see 2.11, "Using the Module in Other Than an FIT Module Environment". When this option is set to "enabled", control of FIT modules such as r_bsp is enabled.
#define SCIFA_SMSTR_CFG_CHx_INCLUDED Note: The default value for channel 8 is "enabled.". The channel number is represented by "x".	Selects whether or not the specified channel is used. When this option is set to "disabled", code for processing the specified channel is omitted. When this option is set to "enabled", code for processing the specified channel is included.
#define SCIFA_SMSTR_CFG_LONGQ_ENABLE Note: The default value is "disabled".	Selects whether or not debug error log acquisition processing is used. When this option is set to "disabled", code for the relevant processing is omitted. When this option is set to "enabled", code for the relevant processing is included. To use this functionality, the LONGQ FIT module is also required.
#define SCIFA_SMSTR_CFG_CHx_INT_TXIF_LEVEL Note: The default value for channel 8 is "10". The channel number is represented by "x".	Sets the interrupt level of the transmit FIFO data empty interrupt (TXIF) when using DMAC transfer or DTC transfer.
#define SCIFA_SMSTR_CFG_CHx_INT_RXIF_LEVEL Note: The default value for channel 8 is "10". The channel number is represented by "x".	Sets the interrupt level of the receive FIFO data full interrupt (RXIF) when using DMAC transfer or DTC transfer.

Configuration options in *r_scifa_smstr_rx_pin_config.h*	
#define R_SCIFA_SMSTR_CFG_SCIFAx_SCKx_PORT Note: The default value for channel 8 is "C". The channel number is represented by "x".	Sets the port number assigned to the SCIFA's SCK pin. Enclose the setting value in single quotation marks (' ').
#define R_SCIFA_SMSTR_CFG_SCIFAx_SCKx_BIT Note: The default value for channel 8 is "5". The channel number is represented by "x".	Sets the bit number assigned to the SCIFA's SCK pin. Enclose the setting value in single quotation marks (' ').
#define R_SCIFA_SMSTR_CFG_SCIFAx_TXDx_PORT Note: The default value for channel 8 is "C". The channel number is represented by "x".	Sets the port number assigned to the SCIFA's TXD pin. Enclose the setting value in single quotation marks (' ').
#define R_SCIFA_SMSTR_CFG_SCIFAx_TXDx_BIT Note: The default value for channel 8 is "7". The channel number is represented by "x".	Sets the bit number assigned to the SCIFA's TXD pin. Enclose the setting value in single quotation marks (' ').
#define R_SCIFA_SMSTR_CFG_SCIFAx_RXDx_PORT Note: The default value for channel 8 is "C". The channel number is represented by "x".	Sets the port number assigned to the SCIFA's RXD pin. Enclose the setting value in single quotation marks (' ').
#define R_SCIFA_SMSTR_CFG_SCIFAx_RXDx_BIT Note: The default value for channel 8 is "6". The channel number is represented by "x".	Sets the bit number assigned to the SCIFA's RXD pin. Enclose the setting value in single quotation marks (' ').

2.7 Arguments

The structure for the arguments of the API functions is shown below. This structure is listed in r_scifa_smstr_rx_if.h, along with the prototype declarations of the API functions.

```
typedef struct
{
   uint32_t  data_cnt;                  /* Number of data (byte unit)       */
   uint8_t * p_tx_data;                 /* Pointer to transmit data buffer  */
   uint8_t * p_rx_data;                 /* Pointer to receive data buffer   */
   scifa_smstr_tranmode_t tran_mode;    /* Data transfer mode               */
} scifa_smstr_info_t;
```

2.8 Return Values

The API function return values are shown below. This enumerated type is listed in r_scifa_smstr_rx_if.h, along with the prototype declarations of the API functions.

```
typedef enum e_scifa_smstr_status
{
   SCIFA_SMSTR_SUCCESS    = 0,          /* Successful operation             */
   SCIFA_SMSTR_ERR_PARAM  = -1,         /* Parameter error                  */
   SCIFA_SMSTR_ERR_HARD   = -2,         /* Hardware error                   */
   SCIFA_SMSTR_ERR_OTHER  = -7          /* Other error                      */
} scifa_smstr_status_t;
```

2.9 Adding Driver to Your Project

This module must be added to each project in the e^2 Studio.

There are two methods for adding to a project: using the FIT plug-in and adding manually.

When the FIT plug-in is used, FIT modules can be added to projects easily and the include file path will be updated automatically. Therefore we recommend using the FIT plug-in when adding FIT modules to a project.

There are the following methods to add FIT module using FIT plug in.

1. Use "FIT Configurator".
   This is the latest method that the plug-in function such as Lib file path automatic setting is enhanced, which we recommend the use.

   For the procedure, refer to “4.3.2 Install the FIT Modules with the FIT Plugin.” in “RX64M/RX71M Group RX Driver Package Ver.1.02 (R01AN2606EJ)” application note.

2. Use the existing ”FIT plug-in”.
   For the procedure, refer to “3. Adding FIT Modules to e2 studio Projects FIT Plug-In” in “Adding Firmware Integration Technology Modules to Projects (R01AN1723EU)” application note.

2.10 Peripheral Functions and Modules Other than SCIFA

In addition to the SCIFA, GPIO FIT module, and MPC FIT module, the SCIFA FIT module controls the following peripheral functions and software module.

- DMA controller (DMAC)
- Data transfer controller (DTC)
- Compare match timer (CMT) (only needed for DMAC transfer or DTC transfer)
- Long queue (LONGQ) software module

Other than LONGQ, FIT modules are not used for resource control. When using the module described in this document in an environment using FIT modules, it is recommended that the control processing of peripheral functions other than the SCIFA be replaced by equivalent FIT modules.

The target source code is contained in the file r_scifa_smstr_target_dev_port.c.

2.10.1 DMAC/DTC

The control method when using DMAC transfer or DTC transfer is described below.

The SCIFA FIT module sets the ICU.IERm.IENj bit to 1 to start a DMAC transfer or DTC transfer and then waits for the transfer to end. Other settings to DMAC registers or DTC registers can be performed by using the DMAC FIT module or DTC FIT module, or by using a custom processing routine created by the user.

Note that in the case of DMAC transfer settings, clearing of the ICU.IERm.IENj bit and clearing of the transfer-end flag must be performed by the user after the DMAC transfer has finished.

Use the control functions listed in table 2.1 to perform the various processing tasks.

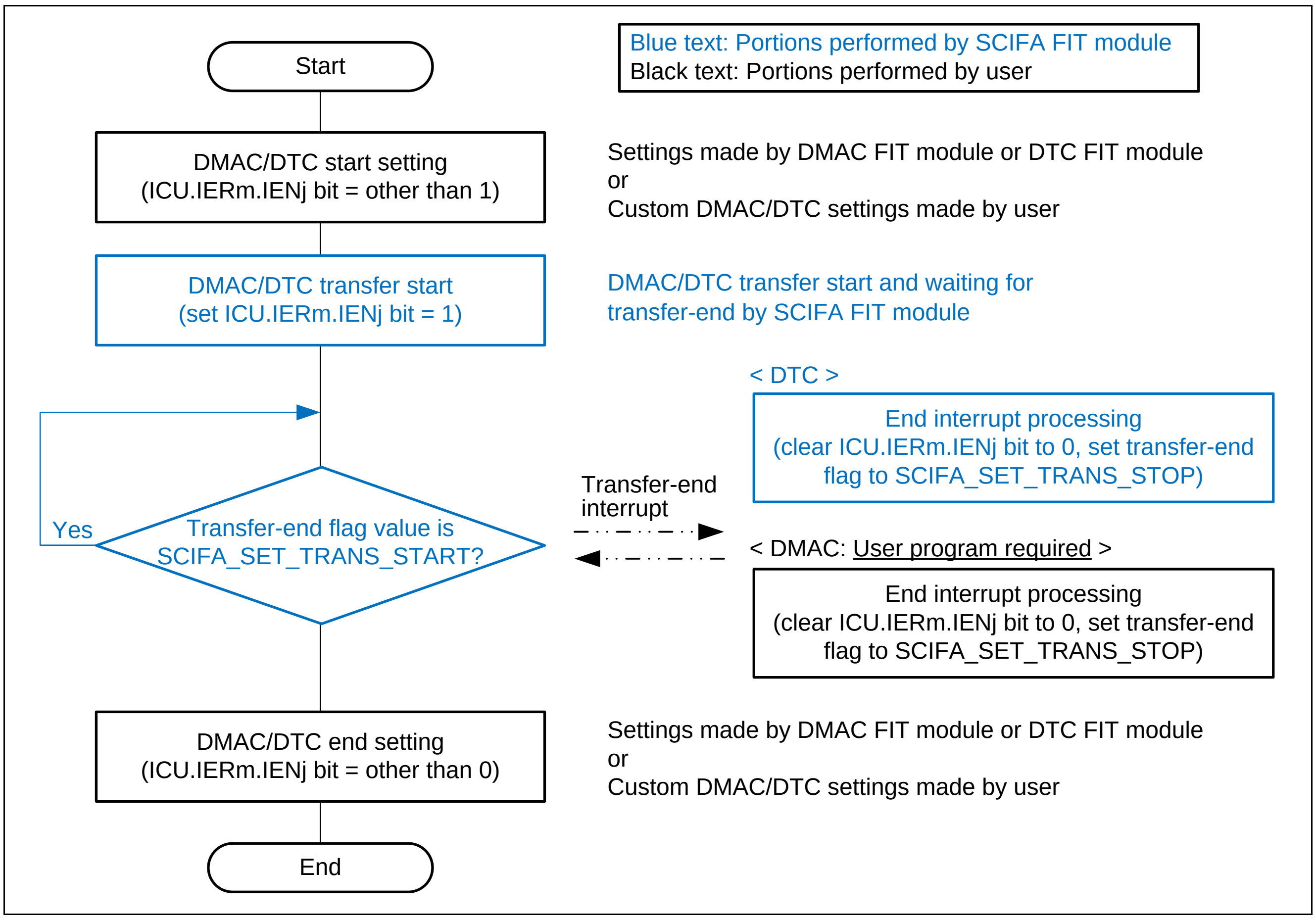

Blue text: Portions performed by SCIFA FIT module
Black text: Portions performed by user
Start
DMAC/DTC start setting (ICU.IERm.IENj bit = other than 1)
Settings made by DMAC FIT module or DTC FIT module or Custom DMAC/DTC settings made by user
DMAC/DTC transfer start (set ICU.IERm.IENj bit = 1)
DMAC/DTC transfer start and waiting for transfer-end by SCIFA FIT module
< DTC >
End interrupt processing (clear ICU.IERm.IENj bit to 0, set transfer-end flag to SCIFA_SET_TRANS_STOP)
Yes
Transfer-end flag value is SCIFA_SET_TRANS_START?
Transfer-end interrupt
< DMAC: User program required >
End interrupt processing (clear ICU.IERm.IENj bit to 0, set transfer-end flag to SCIFA_SET_TRANS_STOP)
DMAC/DTC end setting (ICU.IERm.IENj bit = other than 0)
Settings made by DMAC FIT module or DTC FIT module or Custom DMAC/DTC settings made by user
End

Figure 2.1 Processing for DMAC Transfer and DTC Transfer Settings

Table 2.1 lists the control functions and processing details related to DMAC/DTC control.

The data transmit-end wait processing function r_scifa_smstr_tx_dmacdtc_wait() and data receive-end wait processing function r_scifa_smstr_rx_dmacdtc_wait() wait for transmission or reception to end by running a 1 ms timer. It is therefore necessary to activate a 1 ms timer using the CMT, or the like, on the user system beforehand. Use a callback function, or the like, to call R_SCIFA_SMstr_1ms_Interval() at 1 ms intervals.

Table 2.1 Control Functions and Processing Details

Function Name	Processing Details
r_scifa_smstr_txif_isrX()	SCIFA channel “X” TXIF interrupt handler processing (X represents the channel number.)
r_scifa_smstr_rxif_isrX()	SCIFA channel “X” RXIF interrupt handler processing (X represents the channel number.)
r_scifa_smstr_tx_dmacdtc_wait()	DMAC/DTC transmit-end wait processing
r_scifa_smstr_rx_dmacdtc_wait()	DMAC/DTC receive-end wait processing
r_scifa_smstr_int_txif_init()	TXIF interrupt initialization processing
r_scifa_smstr_int_rxif_init()	RXIF interrupt initialization processing
r_scifa_smstr_int_txif_ier_set()	Sets the TXIF interrupt ICU.IERm.IENj bit to 1.
r_scifa_smstr_int_rxif_ier_set()	Sets the RXIF interrupt ICU.IERm.IENj bit to 1.
R_SCIFA_SMstr_Int_Txif_Ier_Clear()	Clears the TXIF interrupt ICU.IERm.IENj bit to 0.
R_SCIFA_SMstr_Int_Rxif_Ier_Clear()	Clears the RXIF interrupt ICU.IERm.IENj bit to 0.
R_SCIFA_SMstr_Int_Txif_Dmacdtc_flag_Set()	Sets the DMAC/DTC transfer-end flag for transmission operations.
R_SCIFA_SMstr_Int_Rxif_Dmacdtc_flag_Set()	Sets the DMAC/DTC transfer-end flag for reception operations.
R_SCIFA_SMstr_1ms_Interval()	Increments the internal timer counter of each channel.

2.10.2 CMT

Required when using DMAC transfer or DTC transfer. Used to detect transfer timeouts.

2.10.3 LONGQ

The LONGQ FIT module is used by the functionality that fetches the error log.

An example of control utilizing the LONGQ FIT module is included in the SCIFA FIT module. The default setting of the relevant configuration option of the SCIFA FIT module disables the error log fetching functionality. See 2.6, “Compile Settings”.

(1) R_LONGQ_Open() setting

Set to 1 ignore_overflow, the argument of the R_LONGQ_Open() function of LONGQ FIT module. This allows the error log buffer to be used as a ring buffer.

(2) Control procedure

Before calling R_SCIFA_SMstr_Open(), call the following functions in the order shown.

1. R_LONGQ_Open()
2. R_SCIFA_SMstr_Set_LogHdlAddress()

2.11 Using the Module in Other Than an FIT Module Environment

To operate the module in an environment in which FIT modules such as r_bsp are not used, perform the following.

Set #define SCIFA_SMSTR_CFG_USE_FIT in #r_scifa_smstr_rx_config.h to “disabled”.

Comment out the line #include “platform.h” in #r_scifa_smstr_rx_if.h.

Include the following header files in #r_scifa_smstr_rx_if.h.

```
#include “iodefine.h”
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <machine.h>
```

Add the definition #define BSP_MCU_RXxxx (replacing xxx with the microcontroller name using all capital letters) to #r_scifa_smstr_rx_if.h. For example, for the RX64M microcontroller use the string BSP_MCU_RX64M.

In #r_scifa_smstr_rx_if.h add the enum definitions shown below. Also add the #define definitions shown below. Set the system clock (ICLK) value in BSP_ICLK_HZ and the peripheral module clock (PCLKA) value in BSP_PCLKA_HZ. Note that it is possible that some of these definitions may duplicate other FIT module definitions. Insert the lines #ifndef SMSTR_WAIT and #define SMSTR_WAIT at the beginning of the definitions, and insert #endif as the last line.

```
#ifndef SMSTR_WAIT
#define SMSTR_WAIT
typedef enum
{
   BSP_DELAY_MICROSECS = 1000000,
   BSP_DELAY_MILLISECS = 1000,
   BSP_DELAY_SECS = 1
} bsp_delay_units_t;

#define BSP_ICLK_HZ  (120000000)    /* ICLK=120MHz        */
#define BSP_PCLKA_HZ (120000000)    /* PCLKA=120MHz       */
#endif                              /* #ifndef SMSTR_WAIT */
```

3. API Functions

3.1 R_SCIFA_SMstr_Open()

This function is run first when using the APIs of the SCIFA FIT module.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Open(
  uint8_t channel,
  uint8_t br_data
)
```

Parameters

channel

SCIFA channel number

br_data

SCIFA bit rate register (BRR) setting value

Return Values

```
SCIFA_SMSTR_SUCCESS      /* Successful operation */
SCIFA_SMSTR_ERR_PARAM    /* Parameter error */
SCIFA_SMSTR_ERR_OTHER    /* SCIFA resource has been acquired by other task. */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Initializes the SCIFA registers of the channel number specified by the argument channel.

Sets the value specified by the argument br_data in the bit rate register (BRR). Refer to the User's Manual: Hardware of the microcontroller and set br_data as appropriate for the operating environment.

When the function completes successfully, the SCIFA module stop state is canceled, the SCK and TXD pins are set as general output ports in the high-output state, and the RXD pin is set as a general input port.

Note that this function monopolizes the SCIFA resource for the channel number specified by the argument channel. To release this resource, call R_SCIFA_SMstr_Close().

Do not call this function when communication is in progress. Communication cannot be guaranteed if the function is called when communication is in progress.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;
uint8_t              br_data;

channel   = 8;
br_data   = 1;
ret = R_SCIFA_SMstr_Open(channel, br_data);
```

Special Notes

This function controls the GPIO and MPC to set each pin as a general I/O port. Confirm that no other peripheral function is using any of the affected pins before calling this function.

3.2 R_SCIFA_SMstr_Close()

This function is used to release the resources of the SCIFA FIT module currently in use.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Close(
  uint8_t channel
)
```

Parameters

channel

SCIFA channel number

Return Values

```
SCIFA_SMSTR_SUCCESS          /* Successful operation */
SCIFA_SMSTR_ERR_PARAM        /* Parameter error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Sets the SCIFA of the channel number specified by the argument channel to the module stop state.

When the function completes successfully, the SCK and TXD pins are set as general output ports in the high-output state, and the RXD pin is set as a general input port.

Note that this function releases the SCIFA resource for the channel number specified by the argument channel. To restart communication, call R_SCIFA_SMstr_Open().

Do not call this function when communication is in progress. Communication cannot be guaranteed if the function is called when communication is in progress.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;

channel = 8;
ret = R_SCIFA_SMstr_Close(channel);
```

Special Notes

This function controls the GPIO and MPC to set each pin as a general I/O port. Confirm that no other peripheral function is using any of the affected pins before calling this function.

After this function is called the states of the SCK and TXD pins differ from that after a reset (general input port). Review the pin settings if necessary.

3.3 R_SCIFA_SMstr_Control()

This function is used to change settings the bit rate.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Control(
  uint8_t channel,
  uint8_t br_data
)
```

Parameters

channel

SCIFA channel number

br_data

Bit rate register (BRR) setting value

Return Values

```
SCIFA_SMSTR_SUCCESS          /* Successful operation */
SCIFA_SMSTR_ERR_PARAM        /* Parameter error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Changes the SCIFA bit rate for the channel number specified by the argument channel.

Sets the value specified by the argument br_data in the bit rate register (BRR). Refer to the User's Manual: Hardware of the microcontroller and set br_data as appropriate for the operating environment.

Do not call this function when communication is in progress. Communication cannot be guaranteed if this function is called when communication is in progress.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t               channel;
uint8_t               br_data;

channel   = 8;
br_data   = 1;
ret = R_SCIFA_SMstr_Open(channel, br_data);

br_data   = 3;
ret = R_SCIFA_SMstr_Control(channel, br_data);
```

Special Notes

None

3.4 R_SCIFA_SMstr_Write()

This function is used to transmit data.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Write(
 uint8_t channel,
 scifa_smstr_info_t * p_scifa_smstr_info
)
```

Parameters

channel

SCIFA channel number

** p_scifa_smstr_info*

SCIFA information structure

data_cnt

The allowable setting range is 1 to 4,294,967,295. A setting of 0 causes an error to be returned. Also, use a setting value that is a multiple of 8 when specifying DMAC transfer or DTC transfer.

*p_tx_data

Specify the address of the transmit data storage buffer. Use a buffer address aligned with a 4-byte boundary when specifying DMAC transfer or DTC transfer.

*p_rx_data

Not used

tran_mode

Specify the transmit mode. Note that a separate DMAC or DTC transfer program is required in order to specify DMAC transfer or DTC transfer.

SCIFA_SMSTR_SW	: Software transfer
SCIFA_SMSTR_DMAC	: DMAC transfer
SCIFA_SMSTR_DTC	: DTC transfer

Return Values

```
SCIFA_SMSTR_SUCCESS         /* Successful operation */
SCIFA_SMSTR_ERR_PARAM       /* Parameter error */
SCIFA_SMSTR_ERR_HARD        /* Hardware error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Uses the SCIFA of the channel number specified by the argument channel to transmit data.

When DMAC transfer or DTC transfer is specified by the argument tran_mode, the transferrable byte counts are multiples of 8. If the value is not a multiple of 8, the function ends with an error and no transfer takes place.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
#define DATA_CNT (uint32_t)(4)

uint8_t                buf[DATA_CNT];
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t                channel;
scifa_smstr_info_t   tx_info;

channel = 8;
tx_info.data_cnt       = DATA_CNT;
tx_info.p_tx_data      = &buf[0];
tx_info.tran_mode      = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_Write(channel, &tx_info);
```

Special Notes

Take note of the following points when specifying DMAC transfer or DTC transfer.

- The DMAC FIT module, DTC FIT module, and timer module (CMT FIT module, for example) must be obtained separately.
- Use a buffer address aligned with a 4-byte boundary.
- Specify a transfer data count that is a multiple of 8 when calling this function. If the transfer data count results in a final transfer with a data count of 1 to 7, specify software transfer instead when calling this function.
- The data transmit-end wait processing function r_scifa_smstr_tx_dmacdtc_wait() uses a timer. Before calling this function, activate a 1 ms timer using the CMT, or the like. Then call R_SCIFA_SMstr_1ms_Interval() at 1 ms intervals.
- Before calling this function, ensure that the DMAC or DTC is ready to be activated.
- If this function is called by setting tran_mode to SCIFA_SMSTR_DMAC before the DMAC is ready to be activated, no DMAC transfer will take place. The return value in this case is SCIFA_SMSTR_ERR_HARD.
- If this function is called by setting tran_mode to SCIFA_SMSTR_DTC before the DTC is ready to be activated, no DTC transfer will take place. The return value in this case is SCIFA_SMSTR_ERR_HARD.

3.5 R_SCIFA_SMstr_Read()

This function is used to receive data.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Read(
  uint8_t channel,
  scifa_smstr_info_t * p_scifa_smstr_info
)
```

Parameters

channel

SCIFA channel number

** p_scifa_smstr_info*

SCIFA information structure

data_cnt

The allowable setting range is 1 to 4,294,967,295. A setting of 0 causes an error to be returned. Also, use a setting value that is a multiple of 8 when specifying DMAC transfer or DTC transfer.

*p_tx_data

Not used

*p_rx_data

Specify the address of the receive data storage buffer. Use a buffer address aligned with a 4-byte boundary when specifying DMAC transfer or DTC transfer.

tran_mode

Specify the transmit mode. Note that a separate DMAC or DTC transfer program is required in order to specify DMAC transfer or DTC transfer.

SCIFA_SMSTR_SW	: Software transfer
SCIFA_SMSTR_DMAC	: DMAC transfer
SCIFA_SMSTR_DTC	: DTC transfer

Return Values

```
SCIFA_SMSTR_SUCCESS          /* Successful operation */
SCIFA_SMSTR_ERR_PARAM        /* Parameter error */
SCIFA_SMSTR_ERR_HARD         /* Hardware error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Uses the SCIFA of the channel number specified by the argument channel to receive data.

When DMAC transfer or DTC transfer is specified by the argument tran_mode, the transferrable byte counts are multiples of 8. If the value is not a multiple of 8, the function ends with an error and no transfer takes place.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
#define DATA_CNT (uint32_t)(4)

uint8_t              buf[DATA_CNT];
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;
scifa_smstr_info_t   rx_info;

channel = 8;
rx_info.data_cnt     = DATA_CNT;
rx_info.p_rx_data    = &buf[0];
rx_info.tran_mode    = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_Read(channel, &rx_info);
```

Special Notes

Add the following processing when specifying DMAC transfer or DTC transfer.

- The DMAC FIT module, DTC FIT module, and timer module (CMT FIT module, for example) must be obtained separately.
- Use a buffer address aligned with a 4-byte boundary.
- Specify a transfer data count that is a multiple of 8 when calling this function. If the transfer data count results in a final transfer with a data count of 1 to 7, specify software transfer instead when calling this function.
- The data receive-end wait processing function r_scifa_smstr_rx_dmacdtc_wait() uses a timer. Before calling this function, activate a 1 ms timer using the CMT, or the like. Then call R_SCIFA_SMstr_1ms_Interval() at 1 ms intervals.
- Before calling this function, ensure that the DMAC or DTC is ready to be activated.
- If this function is called by setting tran_mode to SCIFA_SMSTR_DMAC before the DMAC is ready to be activated, no DMAC transfer will take place. The return value in this case is SCIFA_SMSTR_ERR_HARD.
- If this function is called by setting tran_mode to SCIFA_SMSTR_DTC before the DTC is ready to be activated, no DTC transfer will take place. The return value in this case is SCIFA_SMSTR_ERR_HARD.

3.6 R_SCIFA_SMstr_WriteRead()

This function is used to transmit and receive data (full duplex communication).

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_WriteRead(
 uint8_t channel,
 scifa_smstr_info_t * p_scifa_smstr_info
)
```

Parameters

channel

SCIFA channel number

** p_scifa_smstr_info*

SCIFA information structure

data_cnt

The allowable setting range is 1 to 4,294,967,295. A setting of 0 causes an error to be returned. Also, use a setting value that is a multiple of 8 when specifying DMAC transfer or DTC transfer.

*p_tx_data

Specify the address of the transmit data storage buffer. Use a buffer address aligned with a 4-byte boundary when specifying DMAC transfer or DTC transfer.

*p_rx_data

Specify the address of the receive data storage buffer. Use a buffer address aligned with a 4-byte boundary when specifying DMAC transfer or DTC transfer.

tran_mode

Specify the transmit mode. Here the transmit mode in which data is both transmitted and received is specified. Note that a separate DMAC or DTC transfer program is required in order to specify DMAC transfer or DTC transfer.

SCIFA_SMSTR_SW	: Software transfer
SCIFA_SMSTR_DMAC	: DMAC transfer
SCIFA_SMSTR_DTC	: DTC transfer

Return Values

```
SCIFA_SMSTR_SUCCESS          /* Successful operation */
SCIFA_SMSTR_ERR_PARAM        /* Parameter error */
SCIFA_SMSTR_ERR_HARD         /* Hardware error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Uses the SCIFA of the channel number specified by the argument channel to transmit and receive data in full duplex mode.

When DMAC transfer or DTC transfer is specified by the argument tran_mode, the transferrable byte counts are multiples of 8. If the value is not a multiple of 8, the function ends with an error and no transfer takes place.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
#define DATA_CNT (uint32_t)(4)

uint8_t              tx_buf[DATA_CNT];
uint8_t              rx_buf[DATA_CNT];
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;
scifa_smstr_info_t   trx_info;

channel = 8;
trx_info.data_cnt     = DATA_CNT;
trx_info.p_tx_data    = &tx_buf[0];
trx_info.p_rx_data    = &rx_buf[0];
trx_info.tran_mode    = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_WriteRead(channel, &trx_info);
```

Special Notes

Add the following processing when specifying DMAC transfer or DTC transfer.

- The DMAC FIT module, DTC FIT module, and timer module (CMT FIT module, for example) must be obtained separately.
- Use a buffer address aligned with a 4-byte boundary.
- Specify a transfer data count that is a multiple of 8 when calling this function. If the transfer data count results in a final transfer with a data count of 1 to 7, specify software transfer instead when calling this function.
- The data receive-end wait processing function r_scifa_smstr_rx_dmacdtc_wait() uses a timer. Before calling this function, activate a 1 ms timer using the CMT, or the like. Then call R_SCIFA_SMstr_1ms_Interval() at 1 ms intervals.
- If this function is called by setting tran_mode to SCIFA_SMSTR_DMAC before the DMAC is ready to be activated, no DMAC transfer will take place. The return value in this case is SCIFA_SMSTR_ERR_HARD.
- If this function is called by setting tran_mode to SCIFA_SMSTR_DTC before the DTC is ready to be activated, no DTC transfer will take place. The return value in this case is SCIFA_SMSTR_ERR_HARD.

3.7 R_SCIFA_SMstr_Get_BuffRegAddress()

This function is used to fetch the addresses of the transmit FIFO data register (FTDR) and receive FIFO data register (FRDR).

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Get_BuffRegAddress(
  uint8_t channel,
  uint32_t * p_ftdr_adr,
  uint32_t * p_frdr_adr
)
```

Parameters

channel

SCIFA channel number

** p_ftdr_adr*

The pointer for storing the address of FTDR. Set this to the address of the storage destination.

** p_frdr_adr*

The pointer for storing the address of FRDR. Set this to the address of the storage destination.

Return Values

```
SCIFA_SMSTR_SUCCESS        /* Successful operation */
SCIFA_SMSTR_ERR_PARAM      /* Parameter error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Use this function when setting the DMAC transfer or DTC transfer destination/transfer source address, etc.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
uint32_t             reg_buff_ftdr;
uint32_t             reg_buff_frdr;
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;

channel = 8;
ret = R_SCIFA_SMstr_Get_BuffRegAddress(channel, &reg_buff_ftdr,
&reg_buff_frdr);
```

Special Notes

None

3.8 R_SCIFA_SMstr_Int_Txif_Ier_Clear()

This function is used to clear the ICU.IERm.IENj bit of the transmit buffer-empty interrupt (TXIF).

Format

```
void R_SCIFA_SMstr_Int_Txif_Ier_Clear (
  uint8_t channel
)
```

Parameters

channel

SCIFA channel number

Return Values

None

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Use this function when disabling interrupts from within the handler of the TXIF interrupt generated at DMAC transfer-end.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
DMA_Handler_W()
{
    R_SCIFA_SMstr_Int_Txif_Ier_Clear(8);
    R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes

Do not use this function for software transfers or DTC transfers. Doing so could disrupt the transfer.

3.9 R_SCIFA_SMstr_Int_Rxif_Ier_Clear()

This function is used to clear the ICU.IERm.IENj bit of the receive buffer-full interrupt (RXIF).

Format

```
void R_SCIFA_SMstr_Int_Rxif_Ier_Clear (
  uint8_t channel
)
```

Parameters

channel

SCIFA channel number

Return Values

None

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Use this function when disabling interrupts from within the handler of the RXIF interrupt generated at DMAC transfer-end.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
DMA_Handler_R()
{
    R_SCIFA_SMstr_Int_Rxif_Ier_Clear(8);
    R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes

Do not use this function for software transfers or DTC transfers. Doing so could disrupt the transfer.

3.10 R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set()

This function is used to set the DMAC/DTC transfer-end flag for data transmission.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set(
  uint8_t channel,
  scifa_smstr_trans_flg_t flg
)
```

Parameters

channel

SCIFA channel number

flg

Flag. The settings are as follows.

SCIFA_SET_TRANS_STOP	: DMAC/DTC transfer-end
(SCIFA_SET_TRANS_START	: DMAC/DTC transfer-start: Setting by the user is prohibited.)

Return Values

```
SCIFA_SMSTR_SUCCESS        /* Successful operation */
SCIFA_SMSTR_ERR_PARAM      /* Parameter error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Set SCIFA_SET_TRANS_STOP from within the handler of the TXIF interrupt generated at DMAC transfer-end.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
DMA_Handler_W()
{
   R_SCIFA_SMstr_Int_Txif_Ier_Clear(8);
   R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes

Do not use this function for software transfers or DTC transfers. Doing so could disrupt the transfer.

3.11 R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set()

This function is used to set the DMAC/DTC transfer-end flag for data reception.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set(
  uint8_t channel,
  scifa_smstr_trans_flg_t flg
)
```

Parameters

channel

SCIFA channel number

flg

Flag. The settings are as follows.

SCIFA_SET_TRANS_STOP	: DMAC/DTC transfer-end
(SCIFA_SET_TRANS_START	: DMAC/DTC transfer-start: Setting by the user is prohibited.)

Return Values

```
SCIFA_SMSTR_SUCCESS         /* Successful operation */
SCIFA_SMSTR_ERR_PARAM       /* Parameter error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Set SCIFA_SET_TRANS_STOP from within the handler of the RXIF interrupt generated at DMAC transfer-end.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
DMA_Handler_R()
{
   R_SCIFA_SMstr_Int_Rxif_Ier_Clear(8);
   R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes

Do not use this function for software transfers or DTC transfers. Doing so could disrupt the transfer.

3.12 R_SCIFA_SMstr_GetVersion()

This function is used to fetch the driver version information.

Format

```
uint32_t R_SCIFA_SMstr_GetVersion(void)
```

Parameters

None

Return Values

Version number

Upper 2 bytes: major version, lower 2 bytes: minor version

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Returns the version information.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
uint32_t version;
version = R_SCIFA_SMstr_GetVersion();
```

Special Notes

None

3.13 R_SCIFA_SMstr_Set_LogHdlAddress()

This function specifies the handler address for the LONGQ FIT module. Call this function when using error log acquisition processing.

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Set_LogHdlAddress(
  uint32_t user_long_que
)
```

Parameters

user_long_que

Specify the handler address of the LONGQ FIT module.

Return Values

```
SCIFA_SMSTR_SUCCESS         /* Successful operation */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

The handler address of the LONGQ FIT module is set in the SCIFA FIT module.

Uses the LONGQ FIT module perform preparatory processing for fetching the error log.

Run this processing before calling R_SCIFA_SMstr_Open().

Reentrancy

Reentrancy from a different channel is possible.

Example

```
#define ERR_LOG_SIZE (16)
#define SCIFA_USER_LONGQ_IGN_OVERFLOW    (1)

scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint32_t             MtlLogTbl[ERR_LOG_SIZE];
longq_err_t          err;
longq_hdl_t          p_SCIFA_user_long_que;
uint32_t             long_que_hndl_address;


/* Open LONGQ module. */
err = R_LONGQ_Open(&MtlLogTbl[0],
                     ERR_LOG_SIZE,
                     SCIFA_USER_LONGQ_IGN_OVERFLOW,
                     &p_SCIFA_user_long_que
);

long_que_hndl_address = (uint32_t)p_SCIFA_user_long_que;
ret = R_SCIFA_SMstr_Set_LogHdlAddress(long_que_hndl_address);
```

Special Notes

Incorporate the LONGQ FIT module separately. Also, enable the line #define SCIFA_SMSTR_CFG_LONGQ_ENABLE in r_scifa_smstr_rx_config.h.

3.14 R_SCIFA_SMstr_Log()

This function fetches the error log. When an error occurs, call this function immediately before user processing ends.

Format

```
uint32_t R_SCIFA_SMstr_Log(
  uint32_t flg,
  uint32_t fid,
  uint32_t line
)
```

Parameters

flg

Set this to 0x00000001 (fixed value).

fid

Set this to 0x0000003f (fixed value).

line

Set this to 0x0001ffff (fixed value).

Return Values

```
0                           /* Successful operation */
1                           /* Error */
```

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

This function fetches the error log. When an error occurs, call this function immediately before user processing ends.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
#define  USER_DRIVER_ID     (0x00000001)
#define  USER_LOG_MAX       (0x0000003f)
#define  USER_LOG_ADR_MAX   (0x00001fff)

uint8_t              buf[DATA_CNT];
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;
scifa_smstr_info_t   tx_info;

channel = 8;
tx_info.data_cnt     = DATA_CNT;
tx_info.p_tx_data    = &buf[0];
tx_info.tran_mode    = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_Write(channel, &tx_info);

if (SCIFA_SMSTR_SUCCESS != ret)
{
    /* Set last error log to buffer. */
    R_SCIFA_SMstr_Log(
        USER_DRIVER_ID,
        USER_LOG_MAX,
        USER_LOG_ADR_MAX
    );

    R_SCIFA_SMstr_Close(channel);
}
```

Special Notes

Incorporate the LONGQ FIT module separately. Also, enable the line #define SCIFA_SMSTR_CFG_LONGQ_ENABLE in r_scifa_smstr_rx_config.h.

3.15 R_SCIFA_SMstr_1ms_Interval()

This function increments the internal timer counter each time it is called.

Format

```
void R_SCIFA_SMstr_1ms_Interval(void)
```

Parameters

None

Return Values

None

Properties

Prototype declarations are contained in r_scifa_smstr_rx_if.h.

Description

Increments the internal timer counter while waiting for the DMAC transfer or DTC transfer to finish.

Reentrancy

Reentrancy from a different channel is possible.

Example

```
void r_cmt_callback (void * pdata)
{
    uint32_t channel;

    channel = (uint32_t)pdata;
    if (channel == gs_cmt_channel)
    {
    R_SCIFA_SMstr_1ms_Interval();
    }
}
```

Special Notes

User a timer or the like to call this function at 1 ms intervals.

In the example above, this function is called by a callback function that runs at 1 ms intervals.

4. Pin Setting

Table 4.1 lists the pin states after a power on reset and after execution of various API functions.

As shown in 1.5.2, Controllable Slave Devices, this module supports SPI mode 3 (CPOL = 1, CPHA = 1). Regardless of the hardware configuration, **after a power on reset control the GPIO from the user side and put the SCK and TXD pins into the high-output state** to use this mode.

Also, the SCK and TXD pins are in the GPIO high-output state after R_SCIFA_SMstr_Close() runs. Review the pin settings if necessary.

Table 4.1 Pin States after Function Execution

Function Name	SCK Pin*[1]	TXD Pin	RXD Pin*[2]
(After power on reset)	GPIO input state	GPIO input state	GPIO input state
Before R_SCIFA_SMstr_Open()	GPIO high-output state Set on user side	GPIO high-output state Set on user side	GPIO input state
After R_SCIFA_SMstr_Open()	GPIO high-output state Set by this module	GPIO high-output state Set by this module	GPIO input state Set by this module
After R_SCIFA_SMstr_Close()	GPIO high-output state Set by this module	GPIO high-output state Set by this module	GPIO input state Set by this module

Notes: 1. Pulling up the SCK pin by means of an external resistor is not recommended when a memory card is connected. Therefore, this pin should be put in the GPIO high-output state after a power on reset.

2. Use an external resistor to pull up the RXD pin. See 1.4.1, Hardware Configuration Example.

5. Reference Documents

User's Manual: Hardware

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

[e^2 studio] RX Family Compiler CC-RX V2.01.00 User's Manual: RX Build (R20UT2747EJ0100)

[e^2 studio] RX Family Compiler CC-RX V2.01.00 User's Manual: RX Coding (R20UT2748EJ0100)

[e^2 studio] RX Family Compiler CC-RX V2.01.00 User's Manual: Message (R20UT2749EJ0100)

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/

Revision History

Rev.	Date	Description	
		Page	Summary
1.08	Dec 26, 2014	—	First edition issued
1.09	Sep 30, 2016	6	Added “RX Family Serial Flash memory Access Clock Synchronous control module Firmware Integration Technology (R01AN2662EJ)” in 1.3 Related Application Notes.
		13	Changed contents of r_scifa_smstr_rx_pin_config.h in 2.6 Compile Settings.
		15	Changed contents in 2.9 Adding Driver to Your Project.
		41	Moved 2.12 Pin States to 4. Pin Setting.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit 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 accordance 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 Microprocessing unit or Microcontroller unit products 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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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

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

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141