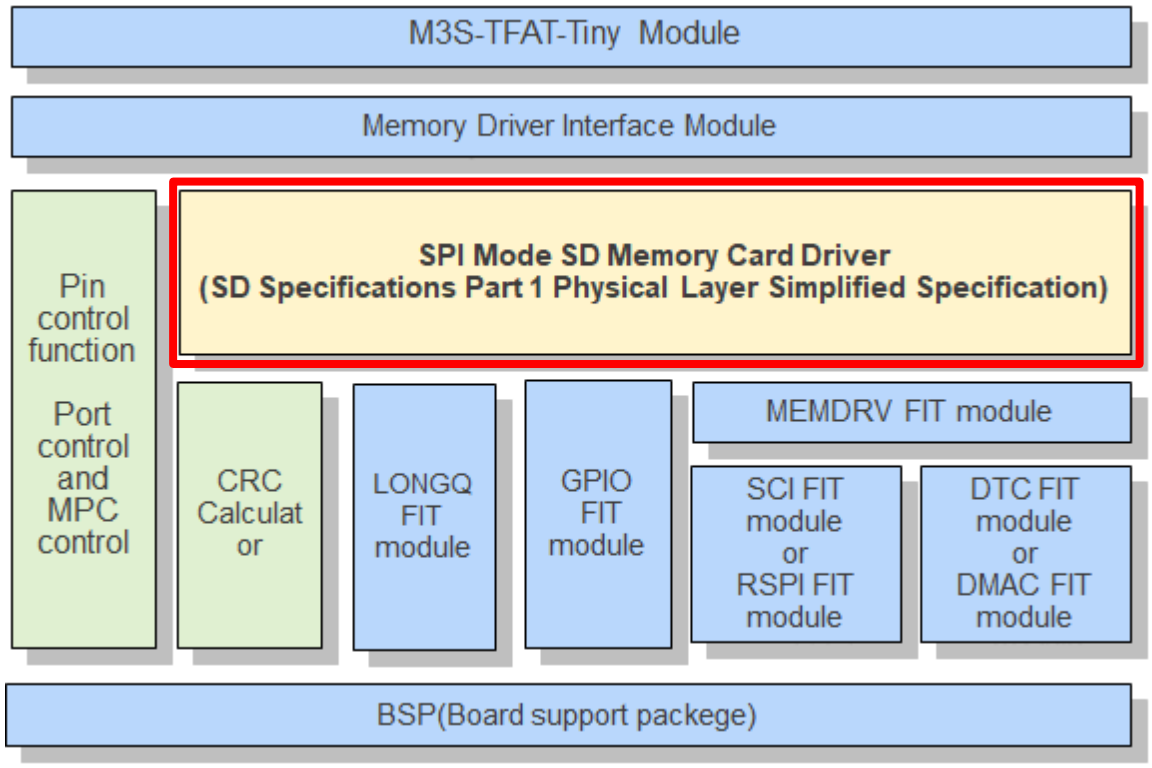
RX Family

SPI Mode SD Memory Card Driver Firmware Integration Technology

Introduction

This application note describes the SPI Mode SD Memory Card driver that uses Firmware Integration Technology (FIT). This module controls SD Memory Cards in SPI mode using the Serial Communication Interface (SCI) or Serial Peripheral Interface (RSPI) included in Renesas Electronics RX Family microcontrollers. In this document, this module is referred to as the SPI mode SD Memory Card driver.

The following shows the position of this module.



This module complies with the Simplified Specification.

When developing host devices that are compliant with the SD Specifications, the user must enter into the SD Card Association License Agreement (SDALA) and SD Card Association Membership Agreement (SDAMA).

For details, refer to the SD Association website.

https://www.sdcard.org/

Target Device

- RX Family

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

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details about the confirmed operational aspects of each complier, see section 6.1 Confirmed Operation Environment.

Related Documents

- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- RX Family DMAC Module Using Firmware Integration Technology (R01AN2063)
- RX Family DTC Module Using Firmware Integration Technology (R01AN1819)
- RX Family LONGQ Module Using Firmware Integration Technology (R01AN1889)
- RX Family SCI Module Using Firmware Integration Technology (R01AN1815)
- RX Family RSPI Module Using Firmware Integration Technology (R01AN1827)
- RX Family Memory Access Driver Interface Module Using Firmware Integration Technology (R01AN4548)
- RX Family Open Source FAT File System M3S-TFAT-Tiny Module FIT (R20AN0038)
- RX Family M3S-TFAT-Tiny Memory Driver Interface Module Using Firmware Integration Technology (R20AN0355)
- RX Family GPIO Module Using Firmware Integration Technology (R01AN1721)

Contents

1. Overview

1.1 SPI Mode SD Memory Card Driver

By using this module in conjunction with a clock synchronous serial interface or other software modules provided by Renesas free of charge (see Figure 1.1 and Table 1-1 on the next page), it is possible to control SD Memory Cards.

Using this module in conjunction with the following modules allows access to files on SD Memory Cards through a FAT file system: RX Family Open Source FAT File System M3S-TFAT-Tiny module (M3S-TFAT-Tiny module) and RX Family M3S-TFAT-Tiny Memory driver interface module (memory driver interface module). These modules are separately provided.

This module can be incorporated into a project as an API. For details on how to incorporate this module, see section 2.11 Adding the FIT Module to Your Project.

Figure 1.1 shows an application structure example when a FAT file system is constructed by using the SPI mode SD Memory Card driver.

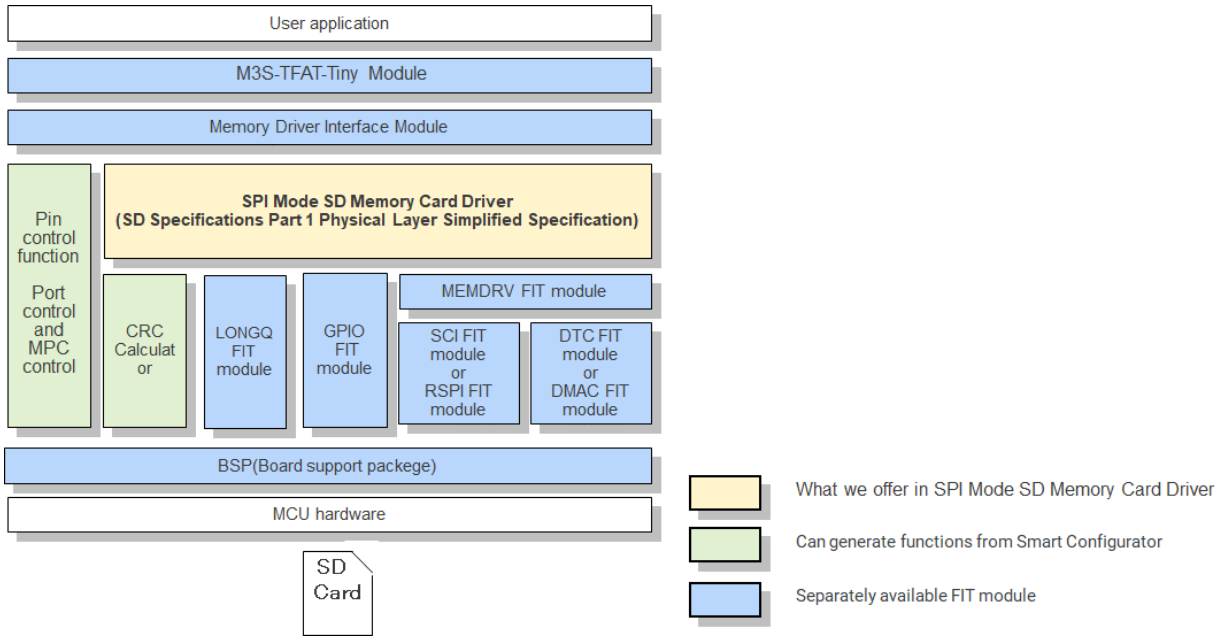


Figure 1.1 Application Structure Example

Table 1-1 lists the modules used in the application structure.

Table 1-1 Modules Used in Application Structure

Item	Document number	Remarks
M3S-TFAT-Tiny module	R20AN0038	
Memory driver interface module	R20AN0335	
SPI mode SD Memory Card driver module	R01AN6908	
MEMDRV FIT module	R01AN4548	
SCI FIT module	R01AN1815	
RSPI FIT module	R01AN1827	
DMAC FIT module	R01AN2063	
DTC FIT module	R01AN1819	
LONGQ FIT module	R01AN1889	
Config_CRC (CRC calculator)	-	Available by using the code generation feature of the Smart Configurator *1
GPIO FIT module	R01AN1721	
BSP (Board Support Package Module)	R01AN1685	
Pin setting function	-	Available by using the pin setting feature of the Smart Configurator *2

Notes: 1. On the [Components] page of the Smart Configurator, click the + (add component) icon, and then select the CRC calculator. Then, click the code generation button.

2. On the [Pins] page of the Smart Configurator, select the ports for CD, CS, and WP corresponding to

Figure 1.2 and the pins corresponding to SMOSI, SMISO, and SCK (or MOSI, MISO, and RSPCK for SPI). Then, click the code generation button.

(1) M3S-TFAT-Tiny module

This software is used for SD memory file management. Please obtain it from the following website as necessary.

RX Family Open Source FAT File System M3S-TFAT-Tiny: https://www.renesas.com/mw/tfat-rx

(2) Memory driver interface module

This software is used for connecting the Renesas Electronics M3S-TFAT-Tiny module and the SPI mode SD Memory Card driver API. Please obtain it from the M3S-TFAT-Tiny web page shown above as necessary.

RX Family M3S-TFAT-Tiny Memory Driver Interface Module Using Firmware Integration Technology

(3) SPI mode SD Memory Card driver module

This software implements SD memory protocol control conforming to the SD Specifications Part 1 Physical Layer Simplified Specification.

(4) MEMDRV FIT module

This software is used as an adapter between the SPI mode SD Memory Card driver and the SCI FIT module or RSPI FIT module.

(5) SCI FIT module

This software controls the SCI hardware. It also includes microcontroller-dependent target microcontroller interface functions and interrupt setting files.

(6) RSPI FIT module

This software controls the RSPI hardware. It also includes the microcontroller-dependent target microcontroller interface functions and interrupt setting files.

(7) DTC FIT module and DMAC FIT module

These software products implement DMAC control and DTC control.

(8) GPIO FIT module

This software controls general-purpose I/O ports.

(9) LONGQ FIT module

This software configures and manages longword (uint32_t) ring buffers used for acquiring error logs.

(10) Config_CRC (CRC calculator)

This software calculates the CRC used for sending or receiving commands.

(11) BSP (board support package module)

This software is the foundation of any project that uses FIT modules.

(12) Pin setting function

This function allocates pins. It specifies the port settings for the port used in SCI or RSPI and the pins used for the CD, WP, and CS pins of the SD Card.

The demo project attached with this module generates the pin setting function by allocating the pins with the Smart Configurator.

1.2 Overview of the SPI Mode SD Memory Card Driver

This module controls SD Memory Cards by using SPI.

Table 1-2 and Table 1-3 list the functions of this module.

Table 1-2 SPI Mode SD Memory Card Driver Functions

Item	Function
Conforming standard	Conforms to the SD Specifications Part 1 Physical Layer Simplified Specification Version 9.00
SD host interface control driver	Block type device driver with 512-byte/sector
Target SD Cards	SD Memory Card
SD Card operating voltage	2.7V to 3.6V This driver is only 3.3 V is supported.
SD Card interface voltage	Only 3.3 V is supported.
SD Card bus interface	SPI mode (1-bit) is supported.
SD Card speed mode	Default Speed mode is supported.
SD Card memory capacity	Standard-Capacity SD Memory Card (SDSC) and High-Capacity SD Memory Cards (SDHC and SDXC) are supported.
SD Card memory control objects	Only a user area is supported. Protected area control is not supported.
SD Card detection function	Detection using the CD pin is only supported.

Table 1-3 Microcontroller Functions

Item	Function
Target microcontroller	RX Family microcontrollers that include the SCI or RSPI
Microcontroller internal data transfer method	Software transfer, DMAC transfer, or DTC transfer can be selected by using the MEMDRV FIT module.
Endian order	Both big endian and little endian are supported.

1.3 API Overview

Table 1-4 shows the API functions for this module.

Table 1-4 API Functions

Function	Functional Overview
R_SDC_SPI_Open()	Driver open processing
R_SDC_SPI_Close()	Driver close processing
R_SDC_SPI_GetCardDetection()	Insertion verification processing
R_SDC_SPI_Initialize()	Initialization processing
R_SDC_SPI_End()	End processing
R_SDC_SPI_Read()	Read processing
R_SDC_SPI_Write()	Write processing
R_SDC_SPI_GetCardStatus()	Card status information acquisition processing
R_SDC_SPI_GetCardInfo()	Register information acquisition processing
R_SDC_SPI_SetLogHdlAddress()	LONGQ FIT module handler address registration processing
R_SDC_SPI_Log()	Error log acquisition processing *1
R_SDC_SPI_GetVersion()	Driver version information acquisition processing

Note: 1. The LONGQ FIT module is also required.

1.4 Hardware Settings

1.4.1 Hardware Configuration Example

Figure 1.2 shows the pin connection diagram. The pin names differ depending on the MCU and serial interface. Confirm the pins and functions used, and then assign the pin functions to the appropriate pins of the MCU.

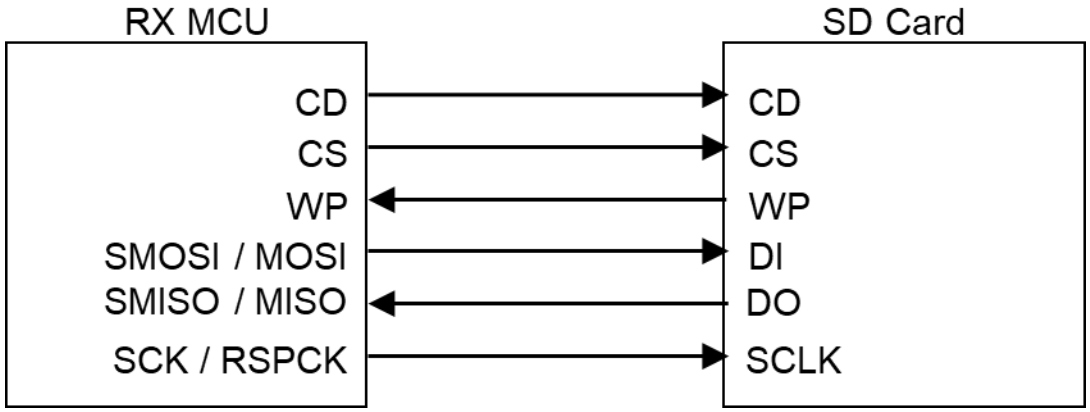
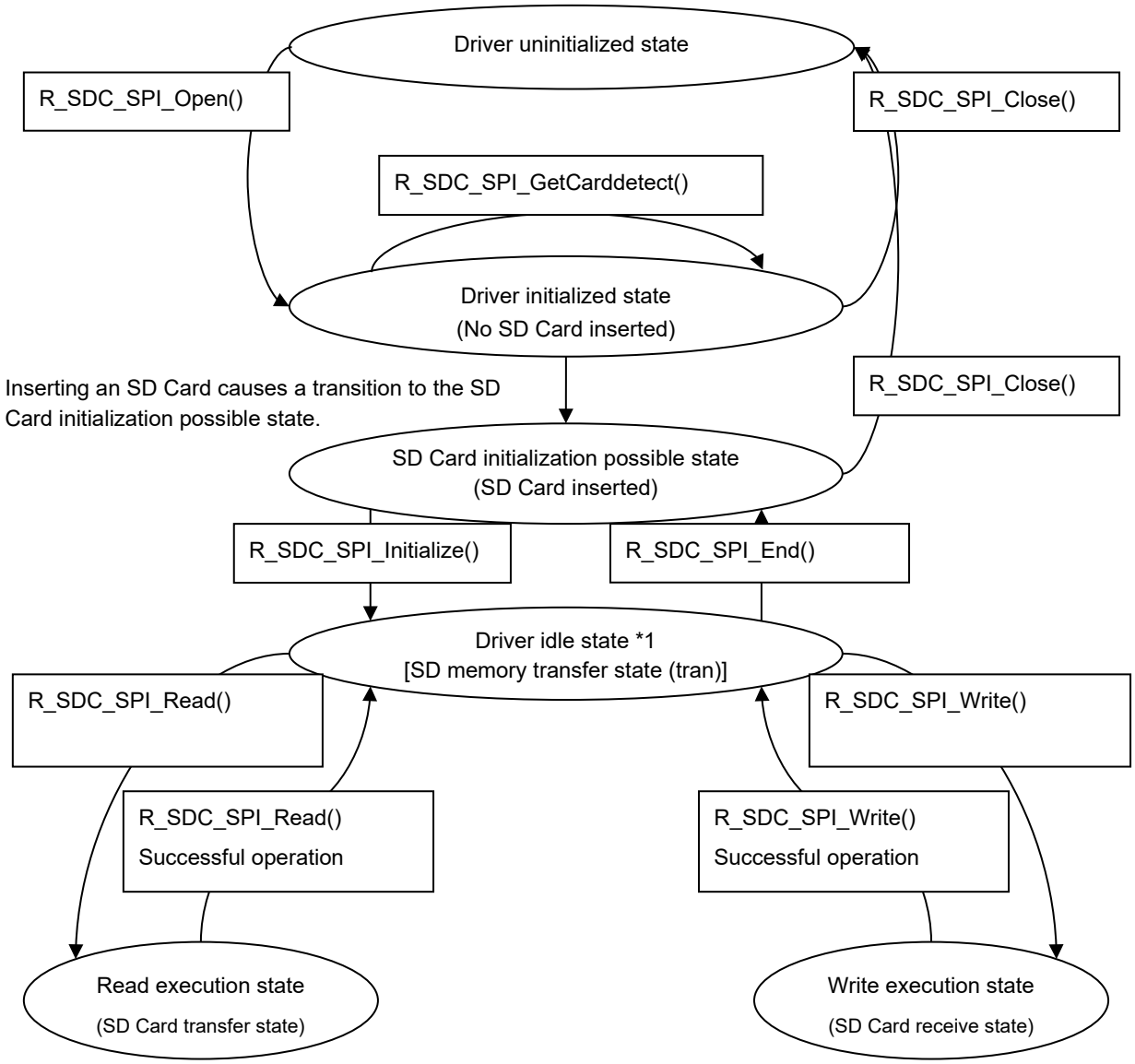


Figure 1.2 Pin Connection Diagram

1.5 State Transition Diagram

Figure 1.3 shows the state transition diagram for this driver.



Note: 1. [ ] indicates the state of the SD Memory Card as described in the SD Specifications Part 1 Physical Layer Simplified Specification.

Figure 1.3 State Transition Diagram for SD Memory Card Driver

1.6 Processing Examples

1.6.1 Basic Control

(1) Supported commands

The SPI mode SD Memory Card driver uses the following commands.

The table below lists the SD Card commands and the status of support in this SPI mode SD Memory Card driver.

Table 1-5 Supported Commands (✓: Supported, —: Not supported)

Command	This module	Remarks
CMD0	✓	Used in SD memory initialization
CMD1	—	
CMD5	—	
CMD6	✓	Used in SD memory initialization
CMD8	✓	Used in SD memory initialization
CMD9	✓	Used in SD memory initialization
CMD10	✓	Used in SD memory initialization
CMD12	✓	Used for SD memory read/write processing
CMD13	✓	Used in SD memory initialization
CMD16	—	
CMD17	✓	Used for SD memory read processing
CMD18	✓	Used for SD memory read processing
CMD24	✓	Used for SD memory write processing
CMD25	✓	Used for SD memory write processing
CMD27	—	
CMD28	—	
CMD29	—	
CMD30	—	
CMD32	—	
CMD33	—	
CMD38	—	
CMD42	—	
CMD55	✓	Used in SD memory initialization
CMD56	—	
CMD58	✓	Used in SD memory initialization
CMD59	✓	Used in SD memory initialization
ACMD13	✓	Used in SD memory initialization
ACMD18	—	
ACMD22	—	
ACMD23	✓	Used in SD memory initialization
ACMD25	—	
ACMD26	—	
ACMD38	—	
ACMD41	✓	Used in SD memory initialization
ACMD42	—	
ACMD51	✓	Used in SD memory initialization

(2) Relationship between data buffers and data in the SD Card

The SD Card driver is set up with the transmit/receive data pointers passed as arguments. Figure 1.4 shows the relationship between the transmit/receive order and the order of the data in the data buffers in RAM. The figure indicates that data in the transmit buffer is sent in the order it appears in the buffer and data is written to the receive buffer in the order received regardless of the endian order.

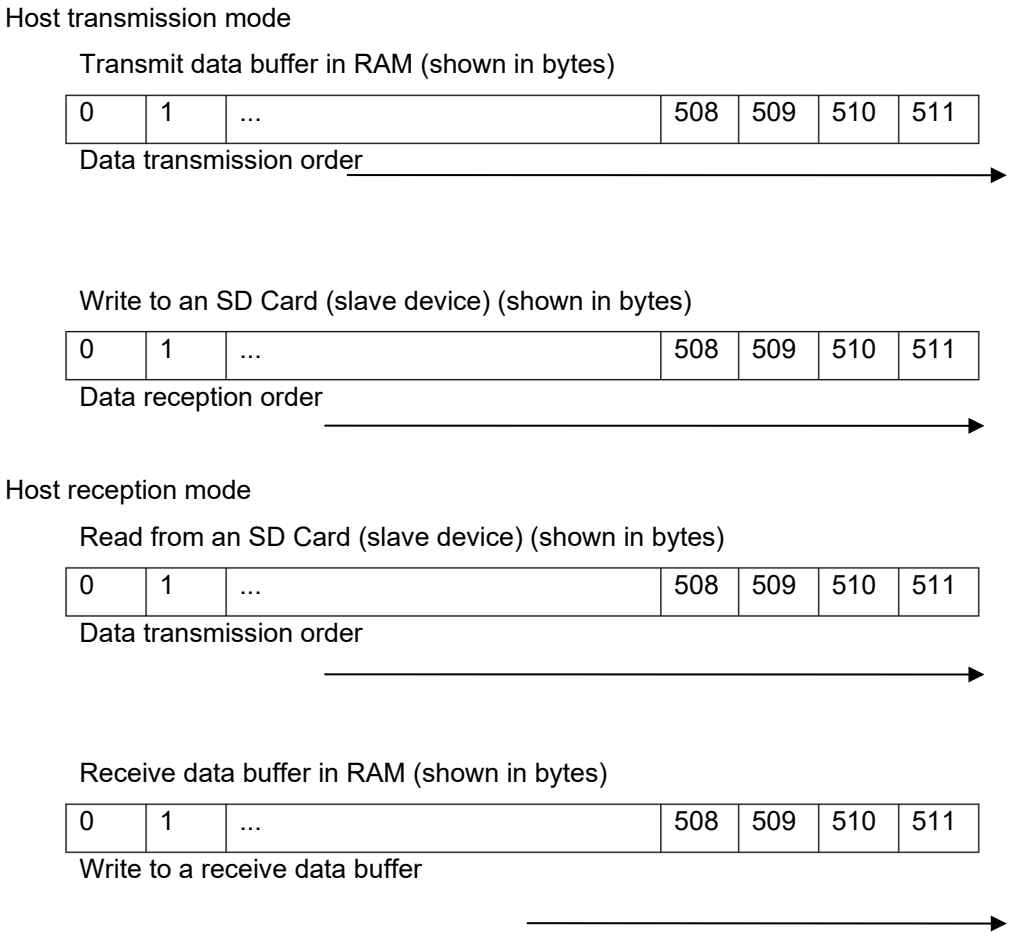


Figure 1.4 Transmission Data Storage

(3) Operating voltage settings

The operating voltage for the SD Card must be set as an argument of the R_SDC_SPI_Initialize() function.

When performing the second or onward initialization processing, place this module in the SD Card initialization possible state by calling the R_SDC_SPI_End() function. Then, remove and then reinsert the SD Card, set the operating voltage again, and then perform the initialization processing again.

Note that this module supports an operating voltage and interface voltage of 3.3 V. Therefore, set the operating voltage to 3.3 V using R_SDC_SPI_Initialize()..

(4) SD Card status verification

To use the SD Card, it is necessary to detect the SD Card insertion/removal state.

Whether the SD Card is inserted or removed can be detected by software polling with the function shown in Table 1-6.

Table 1-6 Status To Be Verified

Type	Status	Remarks
SD Card insertion/removal	SD Card inserted or removed	Detection is possible with the R_SDC_SPI_GetCardDetection() function.

1.6.2 Control After an Error

(1) Handling when an error occurs

We recommend retrying the processing when an error occurs in read, write, or other processing.

If an error occurs even after retrying the processing, remove and reinsert the SD Card, and then initialize it again. For details on the processing related to SD Card insertion and removal, see sections 4.1 SD Card Insertion and Power-On Sequence and 4.2 SD Card Removal and Power-Off Sequence.

Also, if the SD Card driver is used in conjunction with a FAT file system, before removing and reinserting the SD Card, perform any required processing, such as mounting and unmounting, by using the user application.

(2) Handling error termination after transition to the transfer state (tran)

If an error occurs after transition to the transfer state (tran), a CMD12 command is issued regardless of whether or not there was a data transfer. The purpose of issuing the CMD12 command is to transition to the transfer state (tran). Note, however, that if the CMD12 command is issued during write processing, the SD Card might transition to the busy state. This can cause the next R_SDC_SPI_Read function or R_SDC_SPI_Write function call to return an error.

1.7 Limitations

1.7.1 Notes on SD Card Power Supply

After an SD Card is inserted, the power supply stipulated by the SD Card specifications must be applied. See the Power Scheme section in the SD Specifications Part 1 Physical Layer Simplified Specification.

In particular, to control SD Card reinsertion after SD Card removal or power restoration after turning off power to the SD Card, establish the control timing for circuit and power cycle on the system side according to the regulations on the voltage value and voltage sustain period.

The time required to reach the operating voltage after supply of the power supply voltage is started must be adjusted.

Also, the application program must provide the wait time processing required to reach the voltage at which SD Card removal is allowed after supply of the power supply is stopped.

1.7.2 Software Write Protection

The SPI mode SD Card driver does not support software protection state control functions.

1.7.3 Compatible with SDUC card

Due to SD Specifications Part 1 Physical Layer Simplified Specification, SDUC card do not support SPI mode. For this reason, we cannot guarantee operation when the SDUC card is connected.

2. API Information

This FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

The MCU used must support one of the following functions:

- SCI
- RSPI

2.2 Software Requirements

This SD Card driver depends on the following FIT modules:

- r_bsp (Rev. 5.20 or later)
- r_gpio_rx
- r_memdrv_rx
- r_sci_rx
- r_rspi_rx
- r_dmaca_rx
- r_dtc_rx
- r_longq_rx

It also depends on the following code generation module.

- CRC calculator

2.3 Supported Toolchain

This module has been confirmed to work with the toolchain listed in 6.1 Operation Confirmation Environment.

2.4 Interrupt Vector

None

2.5 Header Files

The API calls and interface definitions used are defined in r_sdc_spi_rx_if.h.

The configuration options for each build are selected in r_sdc_spi_rx_config.h.

2.6 Integer Types

This SD Card driver is coded in ANSI C99. These types are defined in stdint.h.

2.7 Configuration Overview

All configurable options that can be set at build time are located in the file ″r_sdc_spi_rx_config.h″.

When using the Smart Configurator, the configuration options can be set on the software component configuration screen. The setting value is automatically reflected in r_sdc_spi_rx_config.h when modules are added to the user project. The option names and setting values are listed in the table below.

Configuration options in r_sdc_spi_rx_config.h	
SDC_SPI_CFG_PARAM_CHECKING_ENABLE Note: The default value is BSP_CFG_PARAM_CHECKING_ENABLE.	1: Parameter check processing is included in the code during build. 0: Parameter check processing is skipped in the code during build. If this option is set to BSP_CFG_PARAM_CHECKING_ENABLE, the system default setting is used.
SDC_SPI_CFG_ERROR_LOG_ACQUISITION Note: The default value is 0.	Define 1 to use the error log acquisition function using the LONGQ FIT module. To use this function, the LONGQ FIT module must be added to the project.
SDC_SPI_CFG_CH0_CD_ENABLE Note: The default value is 1.	Select whether to use the CD pin for channel 0. 0: Do not use the CD pin. 1: Use the CD pin.
SDC_SPI_CFG_CH0_CD_PORT Note: The default value is 0.	Specify the I/O port number to be used for the CD pin for channel 0. 0x00 to 0x19: I/O port number
SDC_SPI_CFG_CH0_CD_BIT Note: The default value is 0.	Specify the I/O port bit number to be used for the CD pin for channel 0. 0 to 7: I/O port bit number to be used
SDC_SPI_CFG_CH0_CS_ENABLE Note: The default value is 1.	Select whether to use the CS pin for channel 0. 0: Do not use the CS pin. 1: Use the CS pin.
SDC_SPI_CFG_CH0_CS_PORT Note: The default value is 0.	Specify the I/O port number to be used for the CS pin for channel 0. 0x00 to 0x19: I/O port number
SDC_SPI_CFG_CH0_CS_BIT Note: The default value is 0.	Specify the I/O port bit number to be used for the CS pin for channel 0. 0 to 7: I/O port bit number to be used
SDC_SPI_CFG_CH0_WP_ENABLE Note: The default value is 1.	Select whether to use the WP pin for channel 0. 0: Do not use the WP pin. 1: Use the WP pin.
SDC_SPI_CFG_CH0_WP_PORT Note: The default value is 0.	Specify the I/O port number to be used for the WP pin for channel 0. 0x00 to 0x19: I/O port number
SDC_SPI_CFG_CH0_WP_BIT Note: The default value is 0.	Specify the I/O port bit number to be used for the WP pin for channel 0. 0 to 7: I/O port bit number to be used
SDC_SPI_CFG_CH1_CD_ENABLE Note: The default value is 1.	Select whether to use the CD pin for channel 1. 0: Do not use the CD pin. 1: Use the CD pin.
SDC_SPI_CFG_CH1_CD_PORT Note: The default value is 0.	Specify the I/O port number to be used for the CD pin for channel 1. 0x00 to 0x19: I/O port number

Configuration options in r_sdc_spi_rx_config.h	
SDC_SPI_CFG_CH1_CD_BIT Note: The default value is 0.	Specify the I/O port bit number to be used for the CD pin for channel 1. 0 to 7: I/O port bit number to be used
SDC_SPI_CFG_CH1_CS_ENABLE Note: The default value is 1.	Select whether to use the CS pin for channel 1. 0: Do not use the CS pin. 1: Use the CS pin.
SDC_SPI_CFG_CH1_CS_PORT Note: The default value is 0.	Specify the I/O port number to be used for the CS pin for channel 1. 0x00 to 0x19: I/O port number
SDC_SPI_CFG_CH1_CS_BIT Note: The default value is 0.	Specify the I/O port bit number to be used for the CS pin for channel 1. 0 to 7: I/O port bit number to be used
SDC_SPI_CFG_CH1_WP_ENABLE Note: The default value is 1.	Select whether to use the WP pin for channel 1. 0: Do not use the WP pin. 1: Use the WP pin.
SDC_SPI_CFG_CH1_WP_PORT Note: The default value is 0.	Specify the I/O port number to be used for the WP pin for channel 1. 0x00 to 0x19: I/O port number
SDC_SPI_CFG_CH1_WP_BIT Note: The default value is 0.	Specify the I/O port bit number to be used for the WP pin for channel 1. 0 to 7: I/O port bit number to be used
SDC_SPI_CFG_SBLK_NUM Note: The default value is 1.	Set the maximum number of blocks for single block write command. 1 to 255: Maximum write block count
SDC_SPI_CFG_USE_SC_CRC Note: The default value is 0.	Define 1 to use the CRC calculation using the Smart Configurator CRC calculator. 0: Do not use the Smart Configurator CRC calculator. 1: Use the Smart Configurator CRC calculator. To use this function, Configure Config_CRC (CRC calculator) of the Smart Configurator as shown below: Generating Polynomial: CRC_CCITT Bit order: MSB Initial value: 0x0000

2.8 Code Size

The table below lists the sizes of ROM, RAM and maximum stack usage associated with this module.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.7 Configuration Overview.

The values in the table below are confirmed under the following conditions.

Module Revision: r_sdc_spi_rx rev.1.00
Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.05.00
(The option of "-lang = c99" is added to the default settings of the integrated development environment.)
GCC for Renesas RX 8.3.0.202311
(The option of "-lang = c99" is added to the default settings of the integrated development environment.)
IAR C/C++ Compiler for Renesas RX version 5.10.1
(Default settings of the integrated development environment)
Configuration Options: Default settings

ROM, RAM, and Stack Code Sizes

Device	Category	Memory Used					
		Renesas Compiler		GCC		IAR Compiler	
		With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking
RX140	ROM	8085byte	7507byte	9722byte	8944byte	10371byte	10371byte
	RAM	15byte		0byte		16byte	
	Stack size	176byte		-		108byte	

Measurement condition:

- Setting for SPI mode SD Memory Card Driver FIT
  - CH0 CD pin Enable: Enable
  - CH0 CS pin Enable: Enable
  - CH0 WP pin Enable: Enable
  - Use CRC smart configuration: Disable
- Setting for MEMDRV FIT
  - Device 0 data transfer mode: CPU transfer
  - Device 0 drive: SCI clock synchronous control FIT module

2.9 Parameters

This section presents the structures used as arguments to the API functions. These structures are included in the file r_sdc_spi_rx_if.h along with the API function prototype declarations.

(1) sdc_spi_cfg_t structure definition

```
typedef struct
{
    uint32_t    mode;
    uint32_t    voltage;
}sdc_spi_cfg_t;
```

(2) sdc_spi_access_t structure definition

```
typedef struct
{
    uint8_t     *p_buff;
    uint32_t    lbn;
    int32_t     cnt;
    uint32_t    write_mode;
}sdc_spi_access_t;
```

(3) sdc_spi_card_status_t structure definition

```
typedef struct
{
    uint32_t card_sector_size;
    uint32_t max_block_number;
    uint8_t  write_protect;
    uint8_t  csd_structure;
}sdc_spi_card_status_t;
```

(4) sdc_spi_card_reg_t structure definition

```
typedef struct
{
    uint32_t    ocr[1];
    uint32_t    cid[4];
    uint32_t    csd[4];
    uint32_t    scr[2];
    uint32_t    sdstatus[4];
}sdc_spi_card_reg_t;
```

2.10 Error Codes as Return Values

This section presents the return values from the API functions. This enumeration type is defined in the file r_sdc_spi_rx_if.h along with the API function prototype declarations.

If an error occurs during processing, these SD Card driver API functions return an error code in their return value.

Table 2-1 lists the error codes. Note that values not listed in the table are reserved for future expansion.

Table 2-1 Error Codes

Macro Definition	Value	Meaning	
SDC_SPI_SUCCESS	0	Successful operation	• Successful operation for functions other than the R_SDC_SPI_GetCardDetection() function • The return value of the R_SDC_SPI_GetCardDetection() function indicates either of the following: 1) The port level set for the CD pin is Low when the CD pin is used. 2) The CD pin is not used.
SDC_SPI_ERR	-1	General error	R_SDC_SPI_Open() function not yet executed, parameter error, and other errors
SDC_SPI_ERR_WP	-2	Write protect error	Write to an SD Card in the write protected state
SDC_SPI_ERR_CRC	-7	CRC error	CRC error detected
SDC_SPI_ERR_ILLEGALCMD	-83	Illegal command error	R1 response card status error (ILLEGAL_COMMAND)
SDC_SPI_ERR_ADDRESS_BOUNDARY	-89	Work address error	Argument buffer address error The address does not fall on a 4-byte boundary.
SDC_SPI_ERR_INTERNAL	-99	Internal error	Error in a module used within the driver

2.11 Adding the FIT Module to Your Project

This module must be added to each project in which it is to be used. Renesas recommends using the Smart Configurator described in (1) or (3) below. However, the Smart Configurator supports only some RX devices. For unsupported RX devices, use the methods outlined in (2) and (4).

(1) Adding the FIT module to your project by using the Smart Configurator in e² studio

By using the Smart Configurator in e² studio, the FIT module is automatically added to your project. Refer to “Renesas e² studio Smart Configurator User’s Guide (R20AN0451)” for details.

(2) Adding the FIT module to your project by using the FIT Configurator in e² studio

By using the FIT Configurator in e² studio, the FIT module is automatically added to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project by using the Smart Configurator in CS+

By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “Renesas e² studio Smart Configurator User’s Guide (R20AN0451)” for details.

(4) Adding the FIT module to your project in CS+

In CS+, manually add the FIT module to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

2.12 “for”, “while” and “do while” Statements

This module uses “for”, “while” and “do while” statements (loop processing) for processes such as waiting for information to be reflected to registers. Such loop processing includes comments with “WAIT_LOOP” as a keyword, which allows users to search for relevant processing when adding fail-safe processing to loop processing.

The following shows a description example.

```
while statement example:
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for statement example:
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while statement example:
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API Functions

3.1 R_SDC_SPI_Open()

This is the first function to be called when this SD Card driver API is used.

Format

```
sdc_spi_status_t R_SDC_SPI_Open(
uint32_t    card_no,
uint8_t     dev_no,
void        *p_sdc_spi_workarea
)
```

Parameters

card_no

SD Card number — The number of the SD Card used (numbering starts at 0)

dev_no

Channel number — The number of the MEMDRV FIT channel used (numbering starts at 0)

*p_sdc_spi_workarea

Pointer to a working area on a 4-byte boundary

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.
SDC_SPI_ERR_ADDRESS_BOUNDARY	There is a problem in the memory area address used by the driver.
SDC_SPI_ERR_INTERNAL	An error occurred in a module used within the driver.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function obtains resources for the general-purpose I/O port controlled by the argument card_no, and initializes the SPI mode SD Card driver and the MEMDRV FIT module specified by the argument dev_no.

The working area is retained until SPI mode SD Card driver close processing completes, and the application must not modify the working area content.

Example

```
uint32_t  g_sdc_spi_work[160/sizesof(uint32_t)];

/* ==== Please add the processing to set the pins. ==== */

if (R_SDC_SPI_Open(SDC_SPI_CARD0, MEMDRV_CH0, &g_sdc_spi_work) !=
SDC_SPI_SUCCESS)
{
    /* Error */
}
```

Special Notes

The pins must be set up before this function is called. For details, see section 4. Pin Setting. To use the SD Card CD pin, WP pin, and CS pin, modify the configuration option settings according to the pins to be used.

If this function does not complete successfully, library functions other than the R_SDC_SPI_GetVersion() and R_SDC_SPI_Log() functions cannot be used.

3.2 R_SDC_SPI_Close()

This function releases the resources being used by the SD Card driver.

Format

```
sdc_spi_status_t R_SDC_SPI_Close(
uint32_t    card_no
)
```

Parameters

card_no

SD Card number	The number of the SD Card used (numbering starts at 0)

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.
SDC_SPI_ERR_INTERNAL	An error occurred in a module used within the driver.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function terminates all SD Card driver processing and releases the resources for the channel specified by the argument card_no.

The working area specified with the R_SDC_SPI_Open() function is not used after this function has been executed. This area can be used for other purposes.

Example

```
/* ==== Please add the processing to set the pins. ==== */

if (R_SDC_SPI_Close(SDC_SPI_CARD0) != SDC_SPI_SUCCESS)
{
    /* Error */
}
```

Special Notes

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

3.3 R_SDC_SPI_GetCardDetection()

This function verifies the SD Card insertion state.

Format

```
sdc_spi_status_t R_SDC_SPI_GetCardDetection(
uint32_t    card_no
)
```

Parameters

card_no

SD Card number	The number of the SD Card used (numbering starts at 0)

Return Values

SDC_SPI_SUCCESS	The port level set for the CD pin is Low, or the CD pin is not used.
SDC_SPI_ERR	There is a problem with parameter settings, or the port level set for the CD pin is High.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function verifies the SD Card insertion state.

If the CD pin level specified by the compile option is Low or the CD pin is not used, this function returns SDC_SPI_SUCCESS.

If the CD pin level specified by the compile option is High or there is a problem with parameter settings, this function returns SDC_SPI_ERR.

Example

```
if (R_SDC_SPI_GetCardDetection(SDC_SPI_CARD0) != SDC_SPI_SUCCESS)
{
          /* Error */
}
```

Special Notes

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Before running this function, driver open processing must be performed by the R_SDC_SPI_Open() function.

The general-purpose I/O port connected to the CD pin of the SD Card socket is used as the SD Card insertion/removal detection pin.

After an SD Card has been detected, processing that provides the power supply to the SD Card must be performed.

3.4 R_SDC_SPI_Initialize()

This function initializes the SD Card, and then changes the SD Card initialization possible state to the driver idle state.

Format

```
sdc_spi_status_t R_SDC_SPI_Initialize(
uint32_t          card_no,
sdc_spi_cfg_t     *p_sdc_spi_config,
uint32_t          init_type
)
```

Parameters

card_no

SD Card number The number of the SD Card used (numbering starts at 0)

*p_sdc_spi_config

Structure that holds the operating settings

mode: The operating mode

0x00000000 (Fixed value. This value is equivalent to software transfer setting.)

voltage: Power supply voltage

0x00200000 (Fixed value. This value is equivalent to 3.3 V in the operating voltage settings.)

init_type: Initialization type

Specify the initialization target. Use the value of the macro definition for media support type in Table 3-1 SD Card Driver Operating Mode (mode).

Table 3-1 SD Card Driver Operating Mode (mode)

Type	Macro definition	Value (Bits)	Definition
Media support type	SDC_SPI_MODE_MEM	0x00000000	SD Memory Card/SD memory

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.
SDC_SPI_ERR_ILLEGALCMD	An illegal command error occurred.
SDC_SPI_ERR_CRC	A CRC error is detected.
SDC_SPI_ERR_INTERNAL	An error occurred in a module used within the driver.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function performs SD Card initialization processing. Call this function after detecting an SD Card.

When the return value is SDC_SPI_SUCCESS, the SD Card transitions to the transfer state (tran) in which read and write access to the SD Card is possible.

Example

```
sdc_spi_cfg_t    sdc_spi_config;

/* ==== Please add the processing to set the pins. ==== */

sdc_spi_config.mode = 0x00000000;
sdc_spi_config.voltage = 0x00200000;
if (R_SDC_SPI_Initialize(SDC_SPI_CARD0, &sdc_spi_config, SDC_SPI_MODE_MEM) !=
SDC_SPI_SUCCESS)
{
    /* Error */
}
```

Special Notes

The pins must be set up before running this function. See section 4 Pin Setting for details. Also, before running this function, driver open processing must be performed by the R_SDC_SPI_Open() function.

If this function returns an error, set the hardware to the SD Card initialization possible state by calling the R_SDC_SPI_End() function, and then perform the initialization processing again.

After initialization completes successfully and before performing the second or onward initialization processing, perform end processing by calling the R_SDC_SPI_End() function.

3.5 R_SDC_SPI_End()

This function clears the value of the working area and changes the driver idle state to the SD Card initialization possible state. Running this function does not change the state of the SD Card.

Format

```
sdc_spi_status_t R_SDC_SPI_End(
uint32_t    card_no,
uint32_t    end_type
)
```

Parameters

card_no

SD Card number	The number of the SD Card used (numbering starts at 0)

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.
SDC_SPI_ERR_INTERNAL	An error occurred in a module used within the driver.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function performs SD Card end processing to make the SD Card removable.

Example

```
if (R_SDC_SPI_End(SDC_SPI_CARD0) != SDC_SPI_SUCCESS)
{
    /* Error */
}

/* ==== Please add the processing to set the pins. ==== */
```

Special Notes

If the SD Card is removed after this function has run, the pins must be set up. See section 4 Pin Setting for details. Also, before running this function, driver open processing must be performed by the R_SDC_SPI_Open() function.

3.6 R_SDC_SPI_Read()

This function performs read processing.

Format

```
sdc_spi_status_t R_SDC_SPI_Read(
uint32_t              card_no,
sdc_spi_access_t    *p_sdc_spi_access
)
```

Parameters

card_no

SD Card number — The number of the SD Card used (numbering starts at 0)

*p_sdc_spi_access

Access information structure

*p_buff: Read buffer pointer

This must be set to an address on a 4-byte boundary.

lbn: Read start block number

cnt: Block count

The maximum value that can be set is 65,535.

write_mode: Write mode (Does not need to be set)

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.
SDC_SPI_ERR_ILLEGALCMD	An illegal command error occurred.
SDC_SPI_ERR_CRC	A CRC error is detected.
SDC_SPI_ERR_INTERNAL	An error occurred in a module used within the driver.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function reads data for the number of blocks specified by cnt in the argument p_sdc_spi_access starting at the block specified by lbn in the argument p_sdc_spi_access, and then stores the data in the buffer specified by p_buff in the argument p_sdc_spi_access.

Example

```
#define TEST_BLOCK_CNT (4)
#define BLOCK_NUM           (512)

sdc_spi_access_t       sdc_spi_access;
uint32_t  g_test_r_buff[(TEST_BLOCK_CNT*BLOCK_NUM)/sizeof(uint32_t)];

sdc_spi_access.p_buff = (uint8_t *)&g_test_r_buff[0];
sdc_spi_access.lbn    = 0x10000000;
sdc_spi_access.cnt    = TEST_BLOCK_CNT;

if(R_SDC_SPI_Read(SDC_SPI_CARD0, &sdc_spi_access) != SDC_SPI_SUCCESS)
{
    /* Error */
}
```

Special Notes

Before running this function, it is necessary to perform driver open processing by the R_SDC_SPI_Open() function and initialization by the R_SDC_SPI_Initialize() function.

We recommend repeating the read operation when this function terminates with a read error.

The size of a block is 512 bytes.

3.7 R_SDC_SPI_Write()

This function performs write processing.

Format

```
sdc_spi_status_t R_SDC_SPI_Write(
uint32_t              card_no,
sdc_spi_access_t    *p_sdc_spi_access
)
```

Parameters

card_no

SD Card number — The number of the SD Card used (numbering starts at 0)

*p_sdc_spi_access

Access information structure

*p_buff: Write buffer pointer

This must be set to an address on a 4-byte boundary.

lbn: Write start block number

cnt: Block count

The maximum value that can be set is 65,535.

write_mode: Write mode

Set this parameter to one of the macro definitions shown in Table 3-2 SD Card Driver Write Mode (write_mode).

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.
SDC_SPI_ERR_WP	An attempt was made to write to the write-protected SD Card.
SDC_SPI_ERR_ILLEGALCMD	An illegal command error occurred.
SDC_SPI_ERR_CRC	A CRC error is detected.
SDC_SPI_ERR_INTERNAL	An error occurred in a module used within the driver.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function writes data from the buffer specified by p_buff in the argument p_sdc_spi_access to an area with the number of blocks specified by cnt in the argument p_sdc_spi_access. That area starts at the blocks specified by lbn in the argument p_sdc_spi_access.

Table 3-2 SD Card Driver Write Mode (write_mode)

Type	Macro Definition	Value (Bits)
Write with pre-erase	SDC_SPI_WRITE_WITH_PREERASE	0x00000000
Normal write	SDC_SPI_WRITE_OVERWRITE	0x00000001

Example

```
#define TEST_BLOCK_CNT (4)
#define BLOCK_NUM           (512)

sdc_spi_access_t sdc_spi_access;
uint32_t         g_test_w_buff[(TEST_BLOCK_CNT*BLOCK_NUM)/sizeof(uint32_t)];

sdc_spi_access.p_buff  = (uint8_t *)&g_test_w_buff[0];
sdc_spi_access.lbn     = 0x10000000;
sdc_spi_access.cnt     = TEST_BLOCK_CNT;
sdc_spi_access.write_mode= SDC_SPI_WRITE_WITH_PREERASE;

if(R_SDC_SPI_Write(SDC_SPI_CARD0, &sdc_spi_access) != SDC_SPI_SUCCESS)
{
    /* Error */
}
```

Special Notes

Before running this function, it is necessary to perform driver open processing by the R_SDC_SPI_Open() function and initialization by the R_SDC_SPI_Initialize() function.

We recommend repeating the write operation if this function terminates with a write error.

If the number of blocks to be transferred exceeds 65,535, break up the write processing into multiple function calls. Care must be taken when this function is called from upper-layer application programs such as the M3S-TFAT-Tiny module.

Block size is 512 bytes.

3.8 R_SDC_SPI_GetCardStatus()

This function acquires the card status information.

Format

```
sdc_spi_status_t R_SDC_SPI_GetCardStatus(
uint32_t                    card_no,
sdc_spi_card_status_t     *p_sdc_spi_card_status
)
```

Parameters

card_no

SD Card number The number of the SD Card used (numbering starts at 0)

*p_sdc_spi_card_status

Card status information structure pointer

card_sector_size: User area block count

max_block_number : Maximum block count

write_protect: Write protect information (see Table 3-3 Write Protect Information (write_protect))

csd_structure: CSD information

0: Standard-Capacity Card (SDSC)

1: High-Capacity Card (SDHC, SDXC)

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function acquires the card status information of the SD Card, and then stores it in a card status information structure.

Table 3-3 Write Protect Information (write_protect)

Macro Definition	Value (Bits)	Definition
SDC_SPI_WP_OFF	0x00	Write protect released state
SDC_SPI_WP_HW	0x01	Hardware write protect state
SDC_SPI_WP_TEMP	0x02	The TEMP_WRITE_PROTECT bit in the CSD register is set.
SDC_SPI_WP_PERM	0x04	The PERM_WRITE_PROTECT bit in the CSD register is set.
SDC_SPI_WP_ROM	0x10	SD ROM

Example

```
sdc_spi_card_status_t  sdc_spi_card_status;

if (R_SDC_SPI_GetCardStatus(SDC_SPI_CARD0, &sdc_spi_card_status) !=
SDC_SPI_SUCCESS)
{
    /* Error */
}
```

Special Notes

Before running this function, it is necessary to perform driver open processing by the R_SDC_SPI_Open() function and initialization by the R_SDC_SPI_Initialize() function.

3.9 R_SDC_SPI_GetCardInfo()

This function acquires the SD Card register information.

Format

```
sdc_spi_status_t R_SDC_SPI_GetCardInfo(
uint32_t                card_no,
sdc_spi_card_reg_t     *p_sdc_spi_card_reg
)
```

Parameters

card_no

SD Card number — The number of the SD Card used (numbering starts at 0)

*p_sdc_spi_card_reg

SD Card register information structure pointer

ocr[1]: SD memory OCR information

cid[4]: SD memory CID information

csd[4]: SD memory CSD information

scr[2]: SD memory SCR information

sdstatus[4]: SD memory SD Status information

Return Values

SDC_SPI_SUCCESS	Successful operation
SDC_SPI_ERR	There is a problem with parameter settings.

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function acquires the SD Card register information, and then stores it in the SD Card register information structure.

Example

```
sdc_spi_card_reg_t     sdc_spi_card_reg;

if (R_SDC_SPI_GetCardInfo(SDC_SPI_CARD0, &sdc_spi_card_reg) !=
SDC_SPI_SUCCESS)
{
          /* Error */
}
```

Special Notes

Before running this function, it is necessary to perform driver open processing by the R_SDC_SPI_Open() function and initialization by the R_SDC_SPI_Initialize() function.

3.10 R_SDC_SPI_SetLogHdlAddress()

This function sets the LONGQ FIT module handler address.

Format

```
sdc_spi_status_t R_SDC_SPI_SetLogHdlAddress(
uint32_t    user_long_que
)
```

Parameters

user_long_que

Handler address of the LONGQ FIT module

Return Values

SDC_SPI_SUCCESS	Successful operation

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function sets the LONGQ FIT module handler address in the SD Card driver.

Example

```
#define SDC_SPI_USER_LONGQ_MAX                      (8)
#define SDC_SPI_USER_LONGQ_BUFSIZE              (SDC_SPI_USER_LONGQ_MAX * 4)
#define SDC_SPI_USER_LONGQ_IGN_OVERFLOW  (1)

uint32_t  g_sdc_spi_user_longq_buf[SDC_SPI_USER_LONGQ_BUFSIZE];
static longq_hdl_t     sdc_spi_user_long_que;
longq_err_t                  err = LONGQ_SUCCESS;
uint32_t               user_long_que = 0;

err = R_LONGQ_Open(g_sdc_spi_user_longq_buf,
                   SDC_SPI_USER_LONGQ_BUFSIZE,
                   SDC_SPI_USER_LONGQ_IGN_OVERFLOW,
                   &sdc_spi_user_long_que);
if (LONGQ_SUCCESS != err)
{
           /* Error */
}
user_long_que = (uint32_t)sdc_spi_user_long_que;
if (R_SDC_SPI_SetLogHdlAddress(user_long_que) != SDC_SPI_SUCCESS)
{
           /* Error */
}
```

Special Notes

This function performs the preparatory processing required to acquire an error log by using the LONGQ FIT module. This processing should be performed before the R_SDC_SPI_Open() is called.

The LONGQ FIT module needs to be added to the project separately.

3.11 R_SDC_SPI_Log()

This function acquires an error log.

Format

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

Parameters

flg

0x00000001 (fixed value)

fid

0x0000003f (fixed value)

line

0x00001fff (fixed value)

Return Values

0	Successful operation

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function acquires an error log.

To terminate error log acquisition, call this function.

Example

```
#define USER_DRIVER_ID      (1)
#define USER_LOG_MAX        (63)
#define USER_LOG_ADR_MAX    (0x00001fff)

sdc_spi_cfg_t               sdc_spi_config;

/* ==== Please add the processing to set the pins. ==== */

sdc_spi_config.mode = 0x0000;
sdc_spi_config.voltage = 0x00200000;
if (R_SDC_SPI_Initialize(SDC_SPI_CARD0, &sdc_spi_config, SDC_SPI_MODE_MEM) !=
SDC_SPI_SUCCESS)
{
    /* Error */
    R_SDC_SPI_Log(USER_DRIVER_ID, USER_LOG_MAX, USER_LOG_ADR_MAX);
}
```

Special Notes

The LONGQ FIT module needs to be added to the project separately.

3.12 R_SDC_SPI_GetVersion()

This function acquires the version information for the driver.

Format

```
uint32_t R_SDC_SPI_GetVersion(
void
)
```

Parameters

None

Return Values

Upper 2 bytes	Major version (decimal)
Lower 2 bytes	Minor version (decimal)

Properties

A prototype declaration for this function appears in r_sdc_spi_rx_if.h.

Description

This function returns the version information for the driver.

Example

```
uint32_t version;
version = R_SDC_SPI_GetVersion();
```

Special Notes

None

4. Pin Setting

To control SD Card using the SPI mode SD Memory Card driver, the pins shown in "1.4 Hardware Settings" is used. Each pin is used by the SPI mode SD Memory Card driver, the MEMDRV FIT module shown in "1.1 SPI Mode SD Memory Card Driver", and either the SCI FIT module or the RSPI FIT module. To enable each module to use the pins, peripheral functions must be properly assigned to the pins.

Function assignments to the pins can be set using the "Smart Configurator" of the integrated development environment "e^2 studio". Refer to Table 4-1 for setting each pin. Depending on the settings, source files "Pin.c", "e_sci_rx_pinset.c", and "r_rspi_rx_pinset" containing functions that control the pin are generated.

Table 4-1 Pin settings for controlling SD Card in SPI mode

Pin	Peripheral Function Assignment	Modules to Call Pin Setting Function		Pin Setting Function (Generated by Smart Configurator)
		FIT module	Function	
CDn*1 *5 CSn(1 *5 WPn(*1 *5	GPIO	SPI mode SD Memory Card	• R_SDC_SPI_Open()	• GPIO FIT module*5 R_GPIO_PinDirectionSet ()
SCKm*2 SMOSIm*2 SMISOm*2	SCI*3	MEMDRV FIT module	• r_memdrv_sci.c sci_init_ports()	• r_sci_pinset.c R_SCI_PinSet_SCIm()*2
RSPCKm*2 MOSIm*2 MISOm*2	RSPI*2	MEMDRV FIT module	• r_memdrv_rspi.c rspi_init_ports()	• r_rspi_pinset.c R_RSPI_PinSet_RSPIm()*2

Note1 : n indicates the card number.

Note2 : m indicates the channel number. The number varies depending on the channel used.

Note3 : When using RSPI, SCI settings are not required.

Note4 : When using SCI, RSPI settings are not required.

Note5 : Do not use pin setting functions generated by the Smart Configurator, use GPIO FIT module to control the pins.

For the pin setting sequence, refer to "4.1 SD Card Insertion and Power-On " and "4.2 SD Card Removal and Power-Off Sequence".

Refer to the demo project for practical coding examples.

4.1 SD Card Insertion and Power-On Sequence

SD Card insertion and power-on sequence are shown in Figure 4-1 and Table 4-2. Insert the SD Card after the R_SDC_SPI_Open() function has completed successfully, and when the power supply to the SD Card is being provided, and the SCI or RSPI output pins are set to L level.

Here, "power supply" indicates the power supply circuit for the SD Card, and "power control pin" indicates the RX microcontroller pin assigned to control the output of the power supply circuit for SD Card.

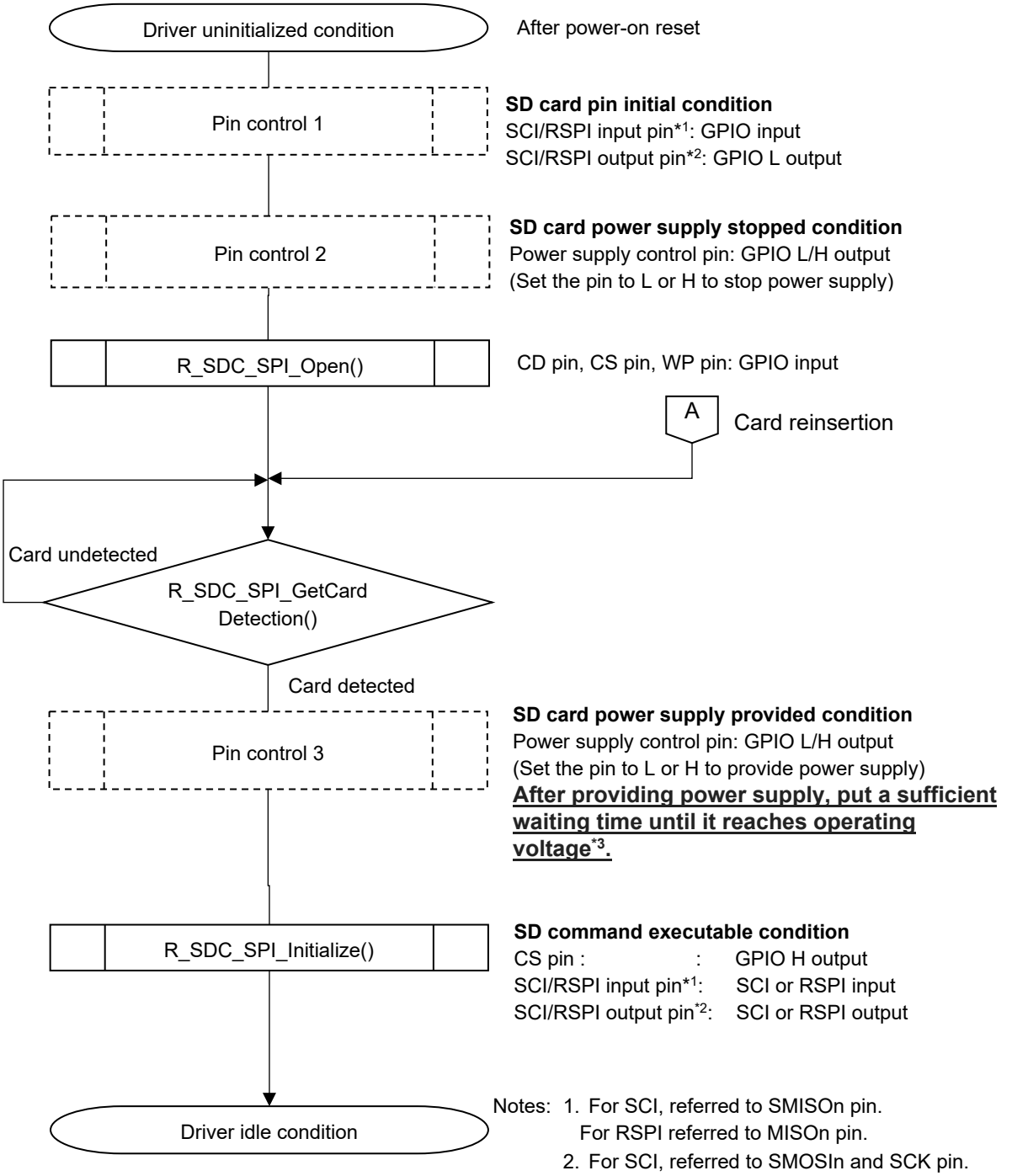


Figure 4-1 SD Card Insertion and Power-On Sequence

Table 4-2 Pin Control for SD Card Insertion

Processing	Target Pin	Pin Settings	Pin Condition after Execution
Pin control 1	SCI input pin or RSPI input pin*1	PMR setting: General I/O port PCR setting: Input pull-up resistor disabled*3 PDR setting: Input MPC setting: General I/O PMR setting: General I/O port	GPIO input
	SCI output pin or RSPI output pin*2	PMR setting: General I/O port DSCR setting: High-drive output PCR setting: Input pull-up resistor disabled*3 PODR setting: L output PDR setting: Output MPC setting: General I/O	GPIO L output
Pin control 2*5	Power supply control pin	PMR setting: General I/O PCR setting: Input pull-up resistor disabled*4 PODR setting: L or H output (Set to L or H to stop power supply) PDR setting: Output	GPIO L/H output (Power supply stopped condition)
Pin control 3*6	Power supply control pin	PODR setting: L or H output (Set to L or H to provide power supply)	GPIO L/H output (Power supply provided condition)

Notes: 1. For SCI, configure to SMISO pin. For RSPI, configure to MISO pin.
2. For SCI, configure to SMOSI and SCK pins. For RSPI, configure to MOSI and RSPCK pins.
3. Pull-up the pin externally to the microcontroller, and disable the microcontroller's internal pull-up resistor.
4. Depending on the system to be used this driver, enable or disable the microcontroller's built-in input pull-up resistor.
5. In the demo project, r_sdc_spi_demo_power_init() is used for the processing.
6. In the demo project, r_sdc_spi_demo_power_on() is used for the processing.

4.2 SD Card Removal and Power-Off Sequence

SD Card removal and power-off sequence are shown in Figure 4-2 and Table 4-3. Remove the SD Card after the R_SDC_SPI_End() function has completed successfully in the driver idle state, and when the power supply to the SD Card is stopped. Also, if the SD Card is removed unexpectedly, stop the power supply by the same sequence.

Here, "power supply" indicates the power supply circuit for the SD Card, and "power control pin" indicates the RX microcontroller pin assigned to control the output of the power supply circuit for SD Card.

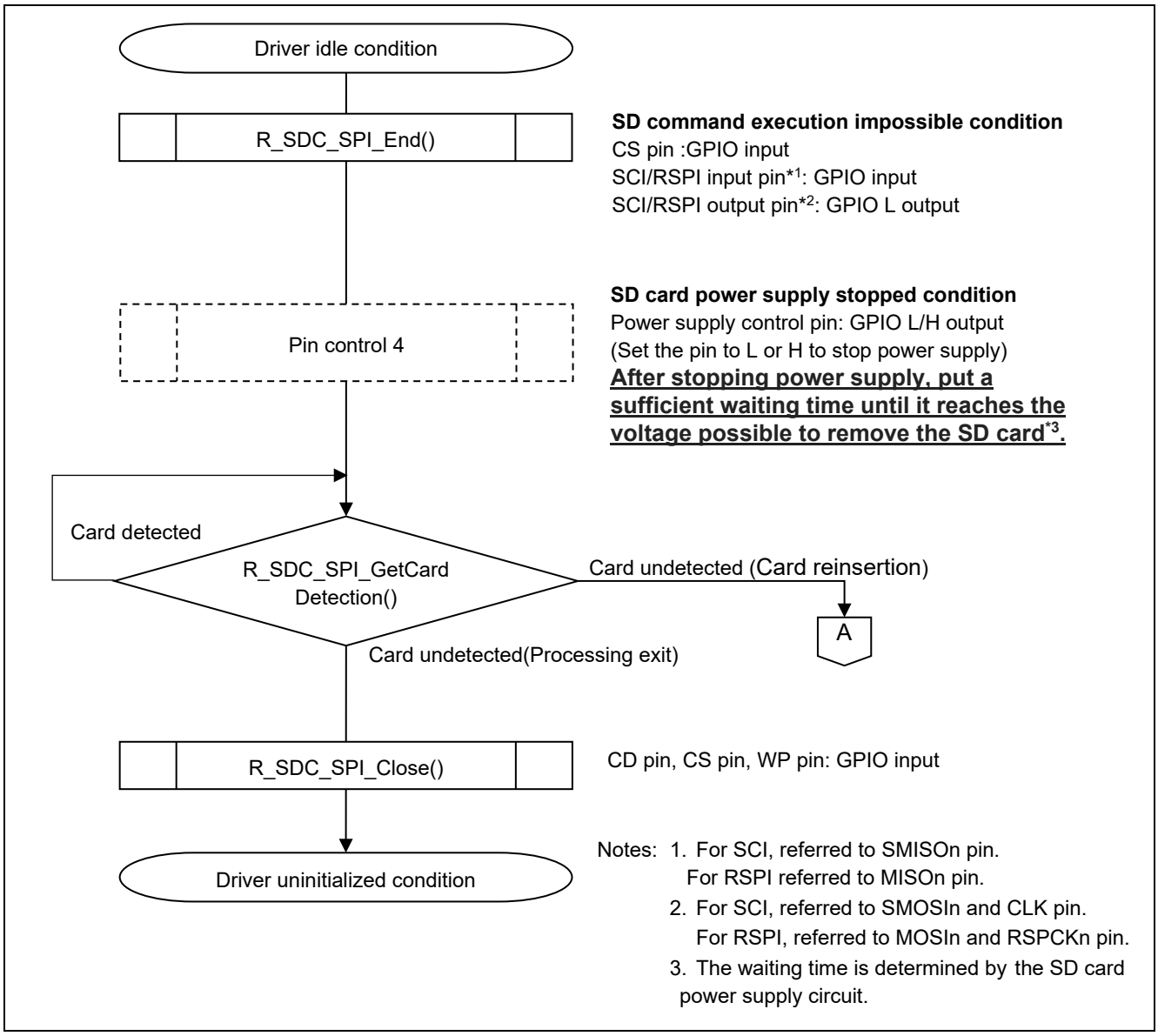


Figure 4-2 SD Card Removal and Power-Off Sequence

Table 4-3 Pin Control for SD Card Removal

Processing	Target Pin	Pin Settings	Pin Condition after Execution
Pin control 4*1	Power supply control pin	PODR setting: L or H output (Set to L or H to stop power supply)	GPIO L/H output (Power supply stopped condition)

Note 1. In the demo project, r_sdc_spi_demo_power_off() is used for the processing.

5. Demo project

5.1 Overview

This application note includes a demo project to explain how to use the SPI mode SD Memory Card driver.

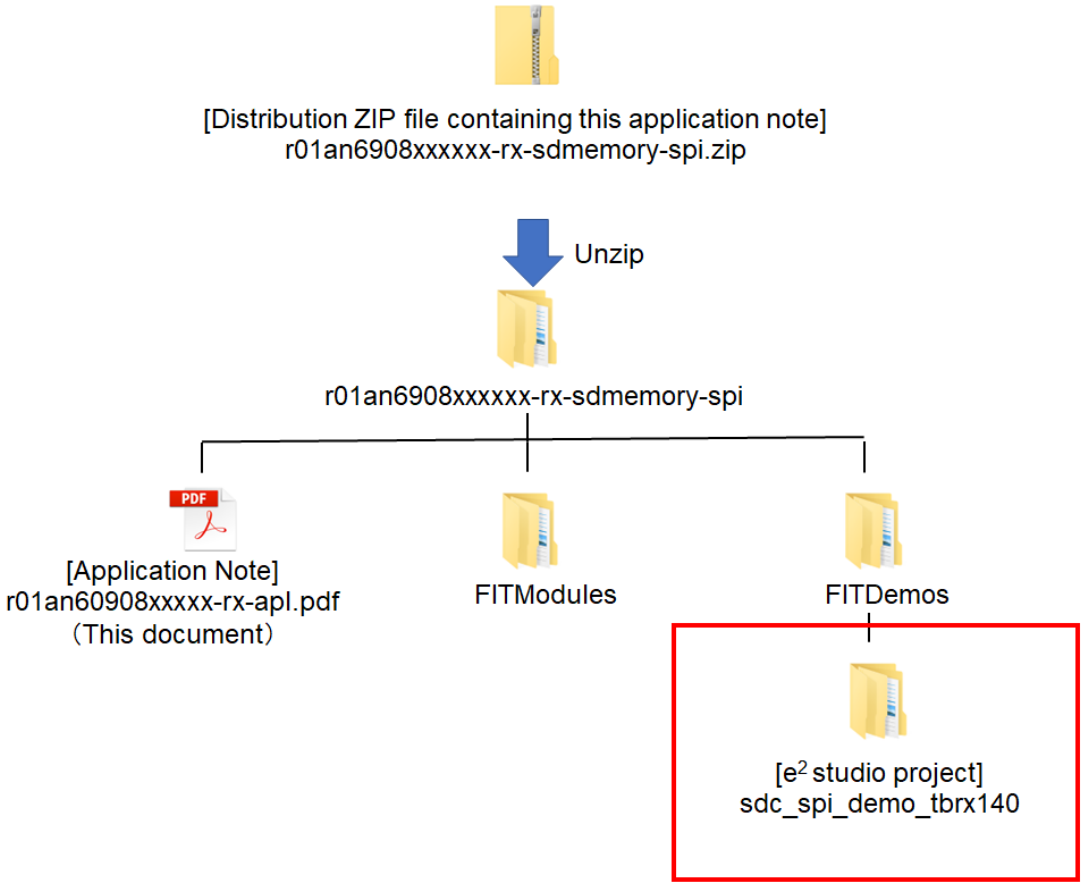


Figure 5-1 Files of the Application Note

The demo project performs the following processing in sequence.

- SD Card insertion and power-on
- Read/write processing to the SD Card using the driver
- SD Card removal and power-off

The demo project performs read/write processing for one SD Card.
The demo project performs read/write processing by CPU transfer using SCI module.
For the details, refer to “5.4 Demo Project Flowchart”

5.2 Operation Confirmation Environment

Table 5-1 shows the hardware, settings, and other conditions for running the demo project.
Figure 5-2 shows the hardware configuration consists of the evaluation board and SD Card.
Figure 5-3 shows the Pin connection between RX140 and the SD Card.
Figure 5-4 shows the rework of the evaluation board.

Refer to the Tables and the Figures, configure the operating environment for the demo project.

Table 5-1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F51403ADFM (RX140 Group)
Operating frequency	HOCO : 48 MHz System clock (ICLK): 48 MHz (HOCO ×1) Peripheral module clocks B (PCLKB): 24 MHz (PLL ×1/2)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e² studio Version: 2023-10
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.05.00
	Compiler option -lang = c99
iodefine.h version	V 1.00
Endian	Little endian or big endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Target Board for RX140 (Product No. RTK5RX1400CxxxxxBJ)
Pmod™ SD Card connector	Digilent® PmodSD™ Pmod™ and PmodSD™ is a trademarks of Digilent Inc.
SD Card	SDHC memory card

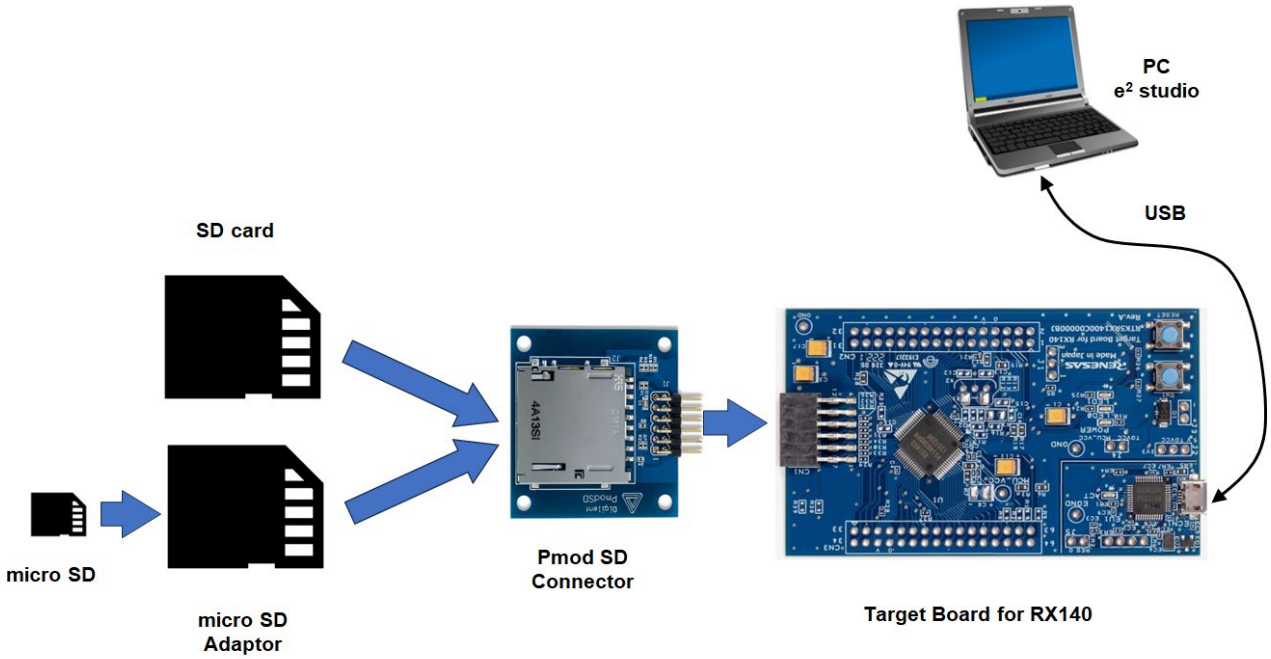


Figure 5-2 Operation Environment for Demo Project

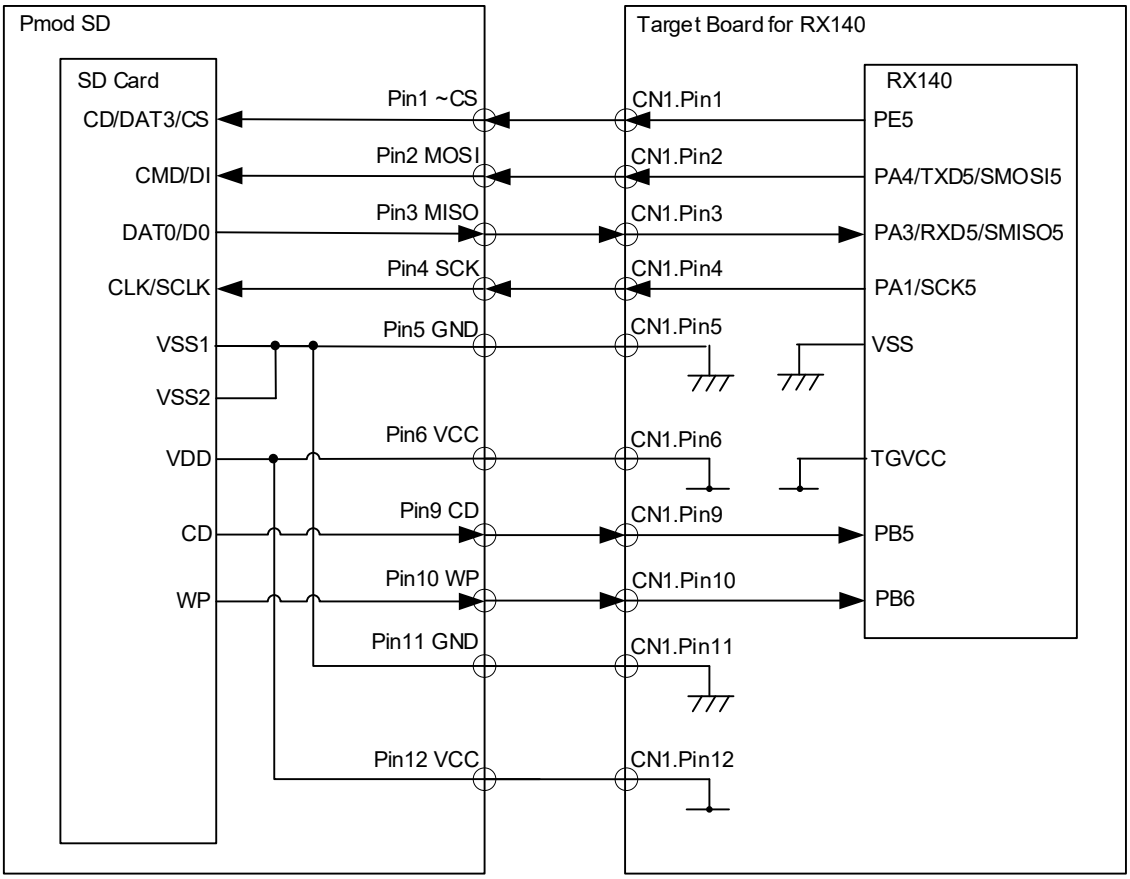


Figure 5-3 Pin Connection for Target Board for RX140 and SD Card

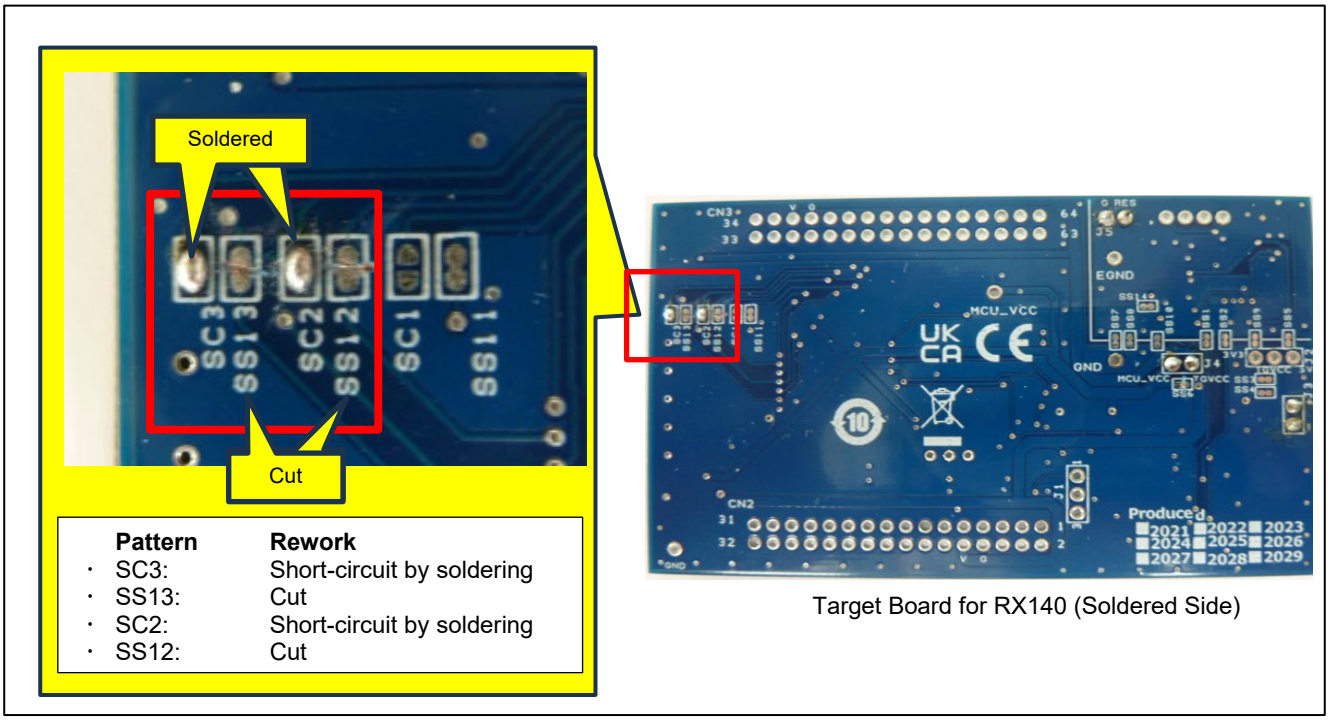


Figure 5-4 Rework of Target Board for RX140

5.3 Compile Settings

All configurable options the Demo Project that can be set at build time are located in the file ″r_sdc_spi_rx_demo_pin_config.h″.

Configuration options in r_sdc_spi_rx_demo_pin_config.h	
#define SDC_SPI_DEMO_CFG_POWER_CONTROL (0) Note: The default value is "0".	The definition is specified if a general-purpose input/output port for the power supply control for the SD Card is enabled or not. Set to 1: The SD Card power supply control is enabled. Set to 0: The SD Card power supply control is disabled.
#define SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE (1) Note: The default value is "1 (high level supplied)".	This definition is specified if an SD Card power supply control is required or not. Set to 1: A high level is supplied to the port that controls the SD Card power supply circuit to enable the SD Card power supply circuit. Set to 0: A low level is supplied to the port that controls the SD Card power supply circuit to enable the SD Card power supply circuit.
#define SDC_SPI_DEMO_POWER_ON_WAIT (100) Note: The default value is "100 (100 ms wait)".	This definition is specified the wait time for the power supply is reached the operating voltage after enable the power supply circuit started power supply to the SD Card. When set to 1, the waiting period is 1ms. Set according to the power supply circuit for the SD Card.
#define SDC_SPI_DEMO _POWER_OFF_WAIT (100) Note: The default value is "100 (100 ms wait)".	This definition is specified the wait time for the power supply is reached the voltage possible for SD Card removal. When set to 1, the waiting period is 1ms. Set according to the power supply circuit for the SD Card.
#define SDC_SPI_DEMO_POWER_CARDx_PORT (‘2’) Note: The default number is " ‘2’ ". Note: "x" in CARDx indicates an SD Card number (x = 0)	Specify the definition to the port number of the power supply a control pin assigned for SD Card number x. Put the number in single quotation marks ' '.
#define SDC_SPI_DEMO_POWER_CARDx_BIT (‘0’) Note: "x" in CARDx indicates an SD Card number (x = 0)	Specify the definition to the bit number of the power supply a control pin assigned for SD Card number x. Put the number in single quotation marks ' '.

5.4 Demo Project Flowchart

The demo project performs the processing shown in Table 5-2 on the SD Card connected to the RX140.

Figure 5-5 and Figure 5-6 shows the demo project processing.

Table 5-2 Operation of Demo Project

	Operation	Description
1)	SD Card insertion and power-on	● After detecting the insertion of SD Card, starts providing power to the SD Card. *1 ● After started providing power, waiting until the operating voltage is reached, and performs the subsequent processing. ● The waiting time is specified by SDC_SPI_CFG_POWER_ON_WAIT.
2)	Read/write to SD Card using this driver	Write and read 4 blocks (2048 bytes) and confirm that the data was written correctly.
3)	SD Card removal and power-off	● Stop providing power to the SD Card.*1 ● After stopped providing power, waiting until it reaches the voltage possible to remove the SD Card, the SD Card can be removed. ● The waiting time is specified by SDC_SPI_CFG_POWER_OFF_WAIT.

Note 1. In the demo project, SDC_SPI_DEMO_CFG_POWER_CONTROL is set to (0) to disable processing for power control pin setting. Because the Target Board for RX140, the operating environment for the demo project, is not implemented with a power supply circuit for the SD Card. Therefore, at the same time the board is powered, the SD Card is also powered.
For hardware implemented with a power supply circuit for the SD Card, set SDC_SPI_DEMO_CFG_POWER_CONTROL to (1), enables the processing to start providing power after detection of the SD Card insertion.

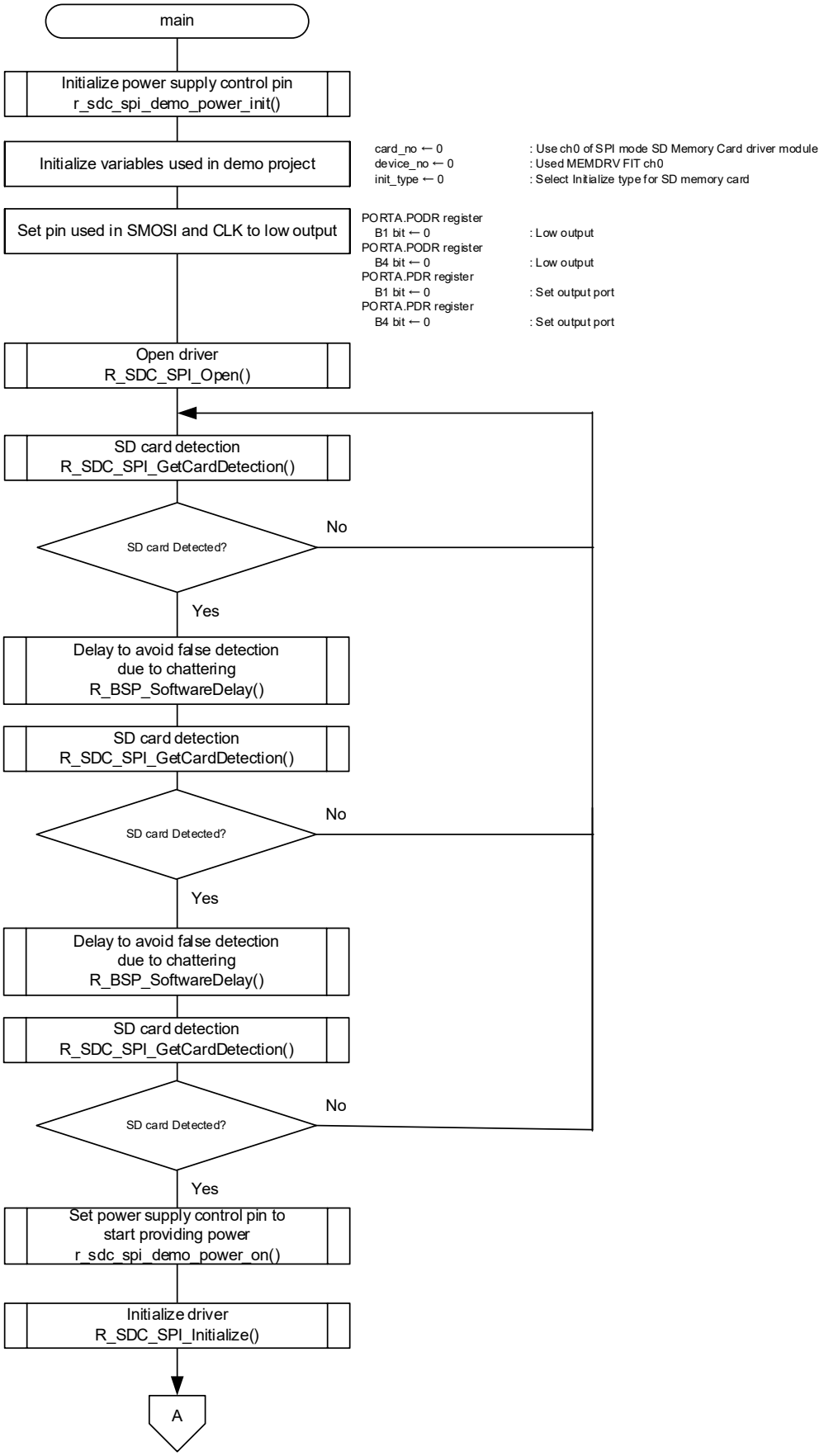


Figure 5-5 Flowchart for Demo Project (1/2)

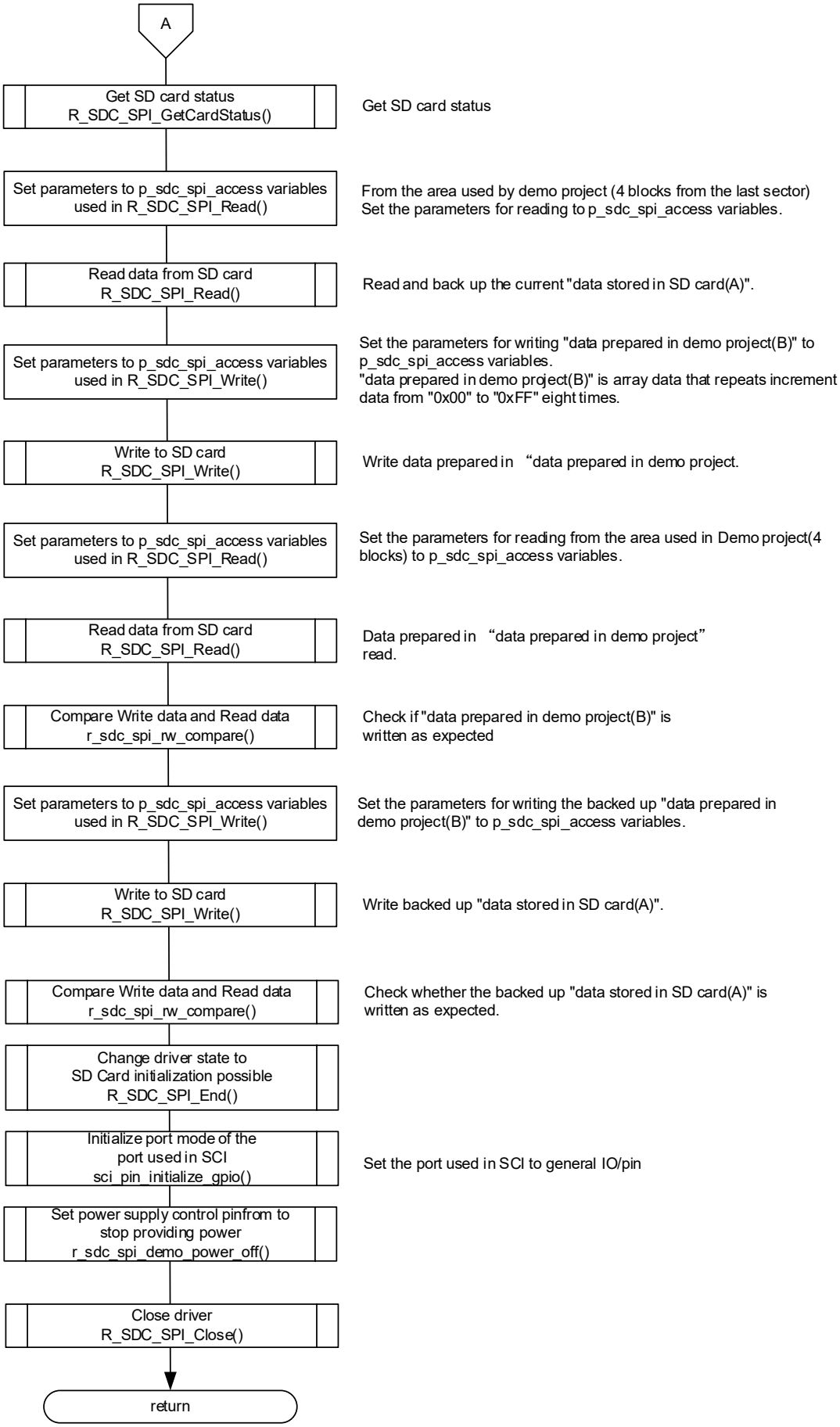
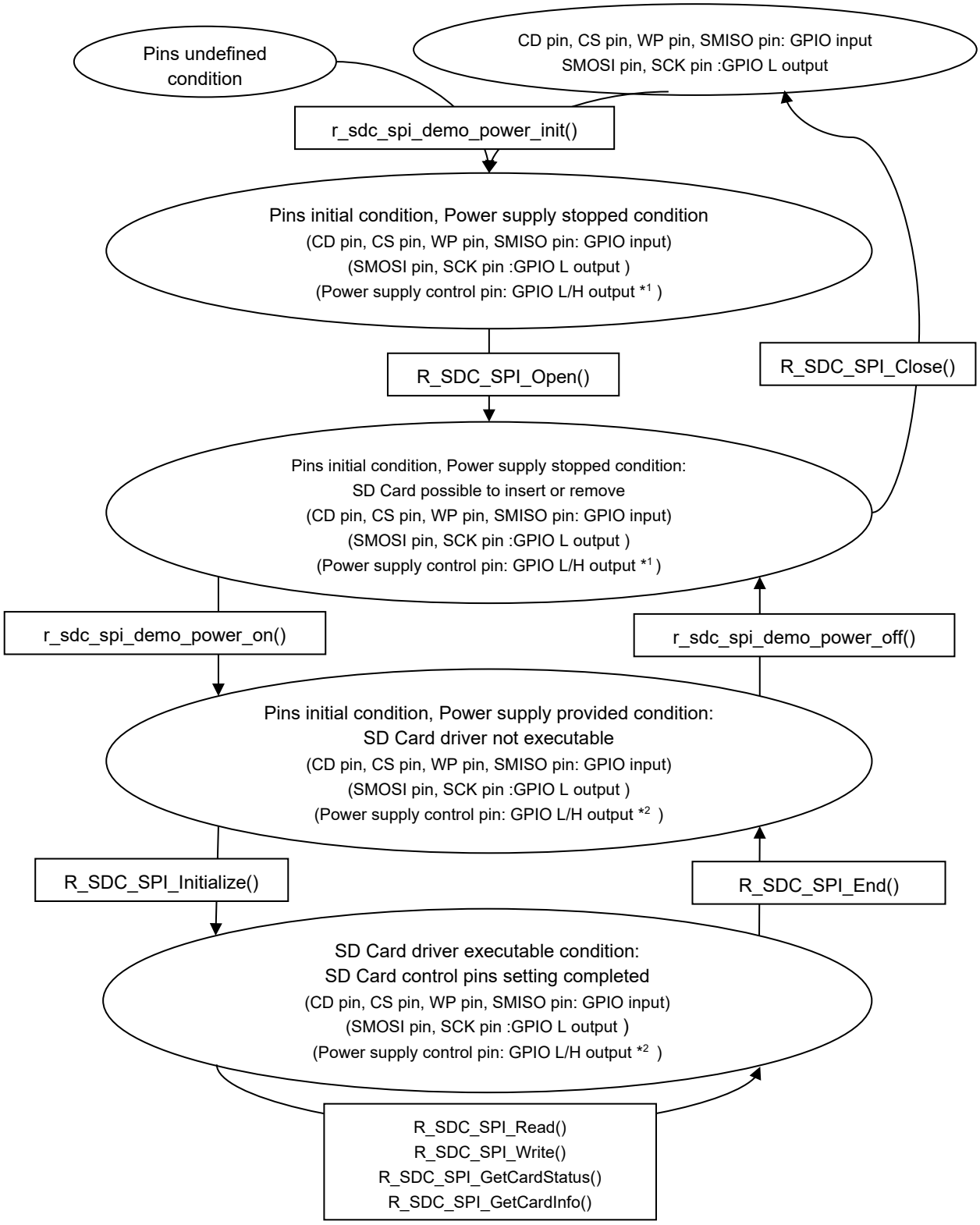


Figure 5-6 Flowchart for Demo Project (2/2)

5.5 Pin Condition Transition

Figure 5-7 Transition of Pin Conditions shows the transition focuses on the conditions of the RX140 pins connected to the SD Card pins.



Notes: 1. Set the pin to L or H to stop power supply.
2. Set the pin to L or H to provide power supply.

Figure 5-7 Transition of Pin Conditions

5.6 Files

Table 5-3 lists the files used in the demo project.

Table 5-3 Files Used in the Demo Project

File Name	Overview
r_sdc_spi_rx_demo.c	The C source file containing the main function of the demo project. The functions of main.c performs the operation shown in refer to “5.4 Demo Project Flowchart” using the FIT module APIs such as the SD Card driver.
r_sdc_spi_rx_demo.c	The C source file containing sub functions used in the demo project.
r_sdc_spi_rx_demo_config.h	The header file for configuration options for the demo project.

5.7 API Functions

The demo project consists of the main function and sub functions. These functions are described in main.c and r_sdc_spi_rx_demo.c.

Table 5-4 API Functions

Function	Functional Overview
main()	Main function of demo project. ・ Performs the operation shown in “5.4 Demo Project Flowchart”by using following four of sub functions. ・ Calls SPI mode SD Memory Card driver API functions to perform read and write processing
r_sdc_spi_demo_power_init()	Initialize the power supply control pin*1 settings
r_sdc_spi_demo_power_on()	Set the power supply control pin*1 to provide power supply to the SD Card
r_sdc_spi_demo_power_off()	Set the power supply control pin*1 to stop power supply to SD Card
r_sdc_spi_rw_compare()	Compare read data and write data

Note 1. Here, "power supply" indicates the power supply circuit for the SD Card, and "power control pin" indicates the RX microcontroller pin assigned to control the output of the power supply circuit for SD Card. Because the Target Board for RX140, the operating environment for the demo project, is not implemented with a power supply circuit for the SD Card. The demo project is designed for hardware that implements an SD card power supply circuit.

(1) r_sdc_spi_demo_power_init()

The function initializes the setting of the power supply control pin.

Format

```
void r_sdc_spi_demo_power_init(
uint32_t card_no
)
```

Parameters

card_no

SD Card number	The SD Card number to be controlled. (numbering starts at 0)

Return Values

None

Description

The function initializes the settings of the power control pin to control the SD Card power supply circuit.

The power control pin is the RX microcontroller pin specified in config.h.

Sets the power control pin as shown below.

- Sets the port mode register (PMR) to general I/O port.
- Sets the pull up control register (PCR) to disable the input pull-up resistor.
- Sets Port Output Data Register (PODR) the pin output to either L or H.
  Specify L or H to stop power supply according to the power supply circuit.
  (The setting in SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE referred to)
- Set PDR to output.

Special Notes

This function was designed to control the power supply for SD Card on the SD Card control system hardware.

Follow "4.1 SD Card Insertion and Power-On Sequence" for providing power to the SD Card. In the sequence, "Pin control 2" is this function which is the process to initialize the setting of the power supply control pin.

Note that, the Target Board for RX140, the operating environment for the demo project, is not implement with a power supply circuit for the SD Card. Therefore, the process to initialize the power control pin in the function is configured to disable by setting SDC_SPI_DEMO_CFG_POWER_CONTROL to (0) in r_sdc_spi_rx_demo_config.h.
For hardware implemented with a power supply circuit, setting SDC_SPI_DEMO_CFG_POWER_CONTROL to (1) enables the process to initialize the power supply control pin.

(2) r_sdc_spi_demo_power_on()

The function sets the power supply control pin to start providing power supply to the SD Card.

Format

```
bool r_sdc_spi_demo_power_on(
  uint32_t card_no
)
```

Parameters

card_no

SD Card number	The SD Card number to be controlled (numbering starts at 0)

Return Values

true	*Successful operation*
false	*error*

Description

Sets the output level of the power supply control pin to start providing power to the SD Card.

- Sets Port Output Data Register (PODR) the pin output to either L or H.
  Specify L or H to provide power supply according to the power supply circuit
  (The setting in SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE referred to)
- After the time set by SDC_SPI_CFG_POWER_ON_WAIT in r_sdc_spi_rx_demo_config.h has elapsed, the execution result of this function is returned as a return value.

Special Notes

- After started providing power, execute the R_BSP_SoftwareDelay() function to wait until the operating voltage is reached.
  The waiting time is specified by SDC_SPI_CFG_POWER_ON_WAIT, which referred to "5.5 Compile settings".
- Execute the r_sdc_spi_demo_power_init() function to initialize the power supply control pin before executing this function.
- This function was designed to control the power supply for SD Card on the SD Card control system hardware.
  Follow "4.1 SD Card Insertion and Power-On Sequence" for providing power to the SD Card. In the sequence, "Pin control 3" is this function which is the process to initialize the setting of the power supply control pin.

  Note that, the Target Board for RX140, the operating environment for the demo project, is not implement with a power supply circuit for the SD Card. Therefore, the process to initialize the power control pin in the function is configured to disable by setting SDC_SPI_DEMO_CFG_POWER_CONTROL to (0) in r_sdc_spi_rx_demo_config.h.
  For hardware implemented with a power supply circuit, setting SDC_SPI_DEMO_CFG_POWER_CONTROL to (1) enables the process to initialize the power supply control pin.

(3) r_sdc_spi_demo_power_off()

The function sets the power supply control pin to stop providing power supply to the SD Card.

Format

```
bool r_sdc_spi_demo_power_off(
  uint32_t card_no
)
```

Parameters

card_no

SD Card number	The SD Card number to be controlled. (numbering starts at 0)

Return Values

true	*Successful operation*
false	*error*

Description

Sets the output level of the power supply control pin to stop providing power to the SD Card.

- Sets Port Output Data Register (PODR) the pin output to either L or H.
  Specify L or H to stop power supply according to the power supply circuit
  (The setting in SDC_SPI_DEMO_CFG_POWER_HIGH_ACTIVE referred to)
- After the time set by SDC_SPI_CFG_POWER_OFF_WAIT in r_sdc_spi_rx_demo_config.h has elapsed, the execution result of this function is returned as a return value.

Special Notes

- After stopped providing power, execute the R_BSP_SoftwareDelay() function to wait until the voltage possible to remove the SD Card.
  The waiting time is specified by SDC_SPI_CFG_POWER_OFF_WAIT, which referred to "5.5 Compile settings".
- Execute the r_sdc_spi_demo_power_init() function to initialize the power supply control pin before executing this function.
- This function was designed to control the power supply for SD Card on the SD Card control system hardware.
  Follow "4.2 SD Card Removal and Power-Off Sequence" for providing power to the SD Card. In the sequence, "Pin control 4" is this function which is the process to initialize the setting of the power supply control pin.

  Note that, the Target Board for RX140, the operating environment for the demo project, is not implement with a power supply circuit for the SD Card. Therefore, the process to initialize the power control pin in the function is configured to disable by setting SDC_SPI_DEMO_CFG_POWER_CONTROL to (0) in r_sdc_spi_rx_demo_config.h.
  For hardware implemented with a power supply circuit, setting SDC_SPI_DEMO_CFG_POWER_CONTROL to (1) enables the process to initialize the power supply control pin.

(4) r_sdc_spi_rw_compare()

This function compares write the data stored in the write data buffer and read data buffer, and return the result.

Format

```
bool r_sdc_spi_rw_compare(
uint32_t * p_buf_w
uint32_t * p_buf_r
uint32_t cnt
)
```

Parameters

* p_buf_w
  Write data buffer pointer
  Please set the buffer in units of 512 bytes.

* p_buf_r
  Read data buffer pointer
  Please set the buffer in units of 512 bytes.

cnt
compare block count

Return Values

true	Data in the Write data buffer and the Read data buffer are matched.
false	Data in the Write data buffer and the Read data buffer are not matched.

Description

Compares data stored in the write data buffer and the read data buffer specified by arguments, return the result.

Special Notes

None

5.8 Downloading Demo Projects

Notes for the case of the FIT module obtained from RX Driver Package are described here.

Demo projects are not included in the RX Driver Package. When using the demo project, the FIT module needs to be downloaded. To download the FIT module, right click on this application note and select "Sample Code (download)" from the context menu in the *Smart Brower* >> *Application Notes* tab.

5.9 Importing a Project into e² studio

Demo projects are not included in the RX Driver Package. When using the demo project, the FIT module

Follow the steps below to import your project into e² studio. Pictures may be different depending on the version of e² studio to be used.

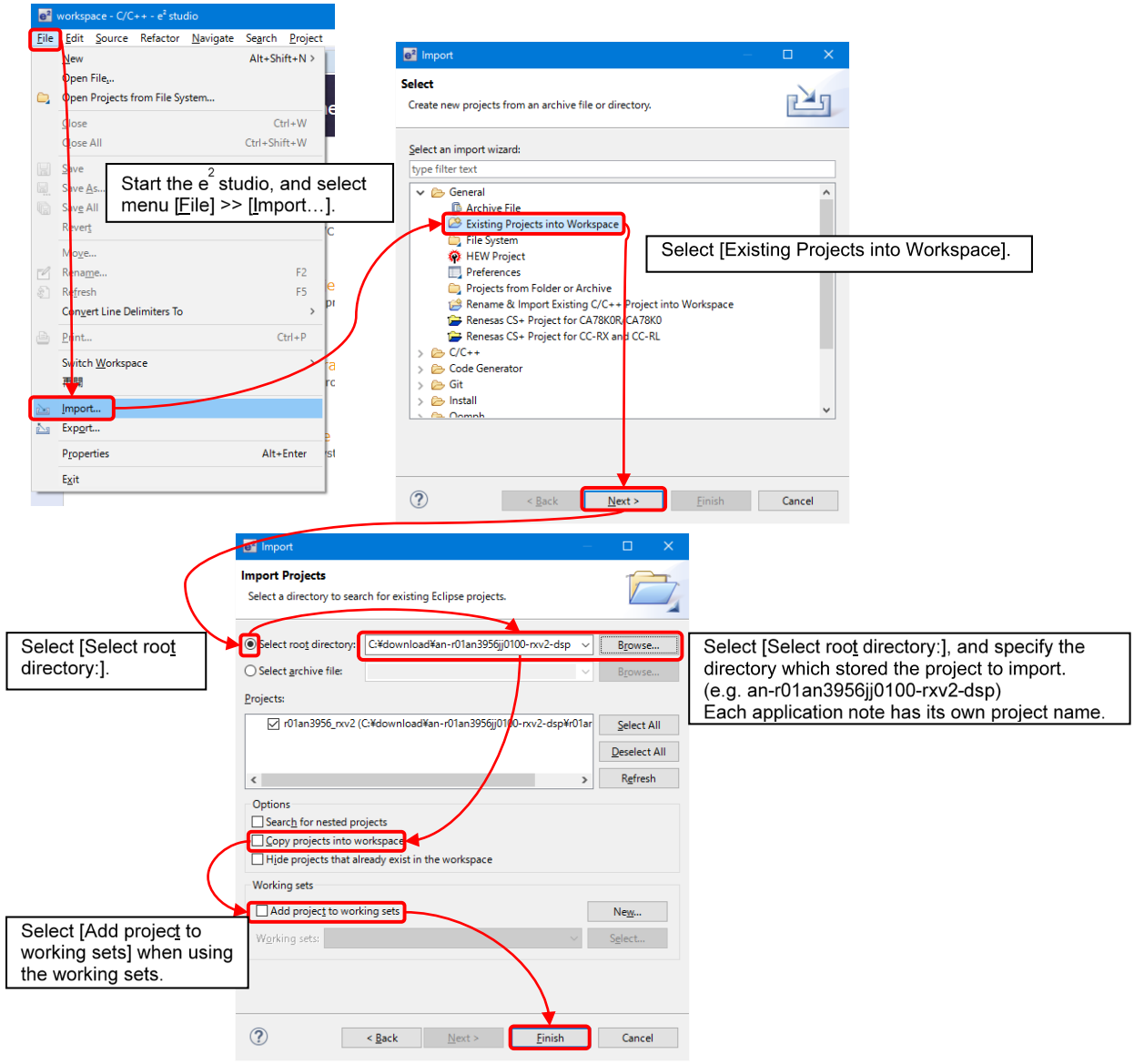


Figure 5-8 Importing a Project into e² studio

5.10 Notes for the Demo project

The demo project runs on the condition shown in "5.2 Operation Confirmation Environment", Target Board for RX140 connected with the Pmod SD Card connector.
By the condition, when power is supplied to the Target Board for RX140, also to the Pmod SD Card connector's power pin.
And power must be supplied to SD Card while SD Card is connected.
Therefore, connect SD Card before starting power supply to the Target Board for RX140.
The other way, remove SD Card after the power supply to the Target Board for RX140 has stopped.

6. Appendices

6.1 Operation Confirmation Environment

This section describes operation confirmation environment for the driver.

Table 6-1 Operation Confirmation Environment

Item	Contents
Integrated development environment	Renesas Electronics e² studio 2023-10 IAR Embedded Workbench for Renesas 5.10.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V.3.05.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.03.00.202311 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 5.10.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Version of the module	Ver.1.00
Board used	Target Board for RX140 (product No.: RTK5RX1400CxxxxxBJ)

6.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"

- Using e^2 studio:

  Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"

When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)".

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_sdc_spi_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

7. Appendices

User's Manual: Hardware
The latest versions 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 CC-RX User's Manual (R20UT3248)
The latest version can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec. 28, 2023	-	First edition issued.

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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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.

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.