

RX Family

R20AN0046EJ0100

Rev.1.00

M3S-MMCSLBR1M: MultiMediaCard Device Driver for SPI

Feb 21, 2011

Introduction

This manual shows the software configuration of MultiMediaCard device driver for the RX family and how to use it.

Target Device

RX Family

Contents

1. Summary	2
2. Program type definitions	4
3. Device Driver	5
4. Setup Examples	11
5. Method for connecting to MCU and MCU resource for use with	17
6. Notes for Application Development.....	19
7. Sample program usage.....	21
Revision Record	26
General Precautions in the Handling of MPU/MCU Products.....	27

1. Summary

1.1 Purpose

The purpose is to provide an interface that connects RX family MCU to MultiMediaCard (hereafter referred to as "MMC") in SPI mode.

This manual provides information to create the application.

1.2 Function Description

This device driver (hereafter referred to as "MMC driver") is software that enables communication with MMC by RX family.

This software uses RX MCU hardware dependent resource to access to MMC in SPI mode.

MMC driver

- Reference MMCA System Specifications; Version 3.2
- This is only used in MMC SPI mode
- This is a block type device driver that defines one sector as 512Byte.
The commands of `READ_MULTIPLE_BLOCK` and `WRITE_MULTIPLE_BLOCK` are used.
As for cards that not support aforesaid two `MULTIPLE_BLOCK` commands, it is operated by commands of `READ_SINGLE_BLOCK` and `WRITE_SINGLE_BLOCK`.
- It supports multiple devices controlled by CS signals.
- It is independent of OS.
- MMC driver in this manual: Ver. 1.23 Release01

1.3 File configuration

Directory Configuration <Directory name> ,File name		Reference
\com	<DIR>	Common Function Directory
	r_mtl_com.c	Common Function
	r_mtl_com2.h	Common Header file
	r_mtl_com.h	Common Header file
	r_mtl_endi.c	Common Function
	r_mtl_mem.c	Common Function
	r_mtl_str.c	Common Function
	r_mtl_tim.c	Common Function
	r_mtl_tim.h	Common Header file
\mmc	<DIR>	Directory of Device Driver for MMC
	r_mmc.h	Common Header File
	r_mmc_io.c r_mmc_io.h	I/O Module for SPI Mode
	r_mmc_mmc.c	MMC Module for SPI Mode
	r_mmc_sfr.h.	Individual Definitions of SFR
	r_mmc_sub.c r_mmc_sub.h	Sub Module for SPI Mode
	r_mmc_usr.c	API for SPI Mode
\sample	<DIR>	Sample Program Directory
	r_testmain.c r_testmain.h	Sample Program for Operation Verification

2. Program type definitions

This section gives the details about the type definitions used in the program.

Datatype	Typedef
unsigned char	uint8_t
unsigned short	uint16_t
unsigned long	uint32_t
signed char	int8_t
signed short	int16_t
signed long	int32_t

3. Device Driver

3.1 Device driver function details

Initialization function

Function Name	Function description
R_mmc_Init_Driver ()	Slot initialization process

Function of device control

Function Name	Function description
R_mmc_Init_Slot()	Slot initialization process
R_mmc_Detach()	Slot stop process
R_mmc_Chk_Detect()	Insertion check process

Data access control function

Function Name	Function description
R_mmc_Read_Data()	Data reading process
R_mmc_Write_Data()	Data writing process
R_mmc_Get_MmcInfo()	MMC information obtaining process

Command list of internal use

Command index	Name of command
CMD0	GO_IDLE_STATE
CMD1	SEND_OP_COND
CMD9	SEND_CSD
CMD10	SEND_CID
CMD12	STOP_TRANSMISSION
CMD13	SEND_STATUS
CMD17	READ_SINGLE_BLOCK
CMD18	READ_MULTIPLE_BLOCK
CMD24	WRITE_BLOCK
CMD25	WRITE_MULTIPLE_BLOCK
CMD58	READ_OCR
CMD59	CRC_ON_OFF

Note: User needs to respond to unsupported commands

3.2 function details

3.2.1 Initialization process of driver (R_mmc_Init_Driver)

clause	detail
Function Name	void R_mmc_Init_Driver(void)
Argument	None
Function	Initialize driver. Initialize SFR for card control. The following process is done in every slot. (1) Open card control port. (2) Initialize card control RAM. Execute once when the system starts up.
Return Value	None

3.2.2 Initialization of card slot (R_mmc_Init_Slot)

clause	detail
Function Name	int16_t R_mmc_Init_Slot(uint8_t SlotNo)
Argument	uint8_t SlotNo : Slot number
Function	Initialize card slot. Initialize card control RAM. Initialization of card. Execute when card insertion is detected.
Return Value	Returns initialization result. MMC_OK : Successful operation MMC_ERR_PARAM : Parameter error MMC_ERR_HARD : Hardware error MMC_ERR_CRC : CRC error MMC_ERR_IDEL : Idle state error MMC_ERR_OTHER : Other error

3.2.3 Card slot stop process (R_mmc_Detach)

clause	detail
Function Name	int16_t R_mmc_Detach(uint8_t SlotNo)
Argument	uint8_t SlotNo : Slot number
Function	Process when removing card from designated slot. -Initialize card control SFR. -Open card control port. -Initialize card control RAM. Execute when card removal is detected.
Return Value	Returns removal result. MMC_OK : Successful operation MMC_ERR_PARAM : Parameter error

3.2.4 Card insertion checking process (R_mmc_Chk_Detect)

clause	detail
Function Name	int16_t R_mmc_Chk_Detect(uint8_t SlotNo, uint8_t* pDetSts)
Argument	uint8_t SlotNo : Slot number uint8_t* pDetSts : Buffer pointer for card insertion condition
Function	Check the condition of card being inserted. If returns "MMC_OK", The port status of card detecting will be in buffer 'pDetSts'. — MMC_TRUE :The port status of card detecting is active — MMC_FALSE: The port status of card detecting Non is non-active Cannot remove chattering in this process. Remove chattering in upper system if needed. Recommend confirming card insertion by periodic polling.
Return Value	Returns the check result. MMC_OK : Successful operation MMC_ERR_PARAM : Parameter error

3.2.5 Data reading process (R_mmc_Read_Data)

clause	detail
Function Name	int16_t R_mmc_Read_Data(uint8_t SlotNo, uint32_t BlkNo, uint16_t BlkCnt, uint8_t FAR* pData, uint8_t Mode)
Argument	uint8_t SlotNo : Slot number uint32_t BlkNo : Block number to start readout uint16_t BlkCnt : Number of readout blocks uint8_t FAR* pData : Pointer to the area where the data which is read must be stored uint8_t Mode : Transfer mode of reading data
Function	Readout the data from card by block (512byte) Readout the data in the designated number of blocks from the designated block. Choose MMC_MODE_NORMAL(transfers data to the designated buffer 'pData'.) in "Mode". The readout from MMC is possible among MMC information handed from R_mmc_Get_MmcInfo() function only when card classification (MmcInfo.Card) is not 'MMC_CARD_UNDETECT'. Maximum block number is 'pMmcInfo.MaxBlkNum' from the "R_mmc_Get_MmcInfo()" function. Maximum number of blocks is 'pMmcInfo.MaxBlkNum' +1.
Return Value	Returns the result of reading. MMC_OK : Successful operation MMC_ERR_PARAM : Parameter error MMC_ERR_HARD : Hardware error MMC_ERR_CRC : CRC error MMC_ERR_OTHER : Other error

3.2.6 Data writing process (R_mmc_Write_Data)

clause	detail
Function Name	int16_t R_mmc_Write_Data(uint8_t SlotNo, uint32_t BlkNo, uint16_t BlkCnt, uint8_t FAR* pData, uint8_t Mode)
Argument	uint8_t SlotNo : Slot number uint32_t BlkNo : Block number to start writing uint16_t BlkCnt : Number of writing blocks uint8_t FAR* pData : Pointer to the area where the data which is written must be stored uint8_t Mode : Transfer mode of writing data
Function	<p>Write the data to card by block (512byte). Write the data in the designated number of blocks to the designated block. Choose MMC_MODE_NORMAL(This is a mode that transfers data from the designated buffer 'pData') in "Mode".</p> <p>The transfers to MMC is possible among MMC information handed from R_mmc_Get_MmcInfo() function only when card classification (MmcInfo.Card) is not 'MMC_CARD_UNDETECT'.</p> <p>Maximum block number is 'pMmcInfo.MaxBlkNum' from the "R_mmc_Get_MmcInfo()" function.</p> <p>Maximum number of blocks is 'pMmcInfo.MaxBlkNum' +1.</p>
Return Value	Returns the result of writing. MMC_OK : Successful operation MMC_ERR_PARAM : Parameter error MMC_ERR_HARD : Hardware error MMC_ERR_WP : Write-protection error MMC_ERR_OTHER : Other errors

3.2.7 Card information obtaining process (R_mmc_Get_MmcInfo)

clause	detail
Function Name	int16_t R_mmc_Get_MmcInfo(uint8_t SlotNo, MMC_INFO* pMmcInfo)
Argument	uint8_t SlotNo : Slot number MMC_INFO* pMmcInfo : Buffer pointer for card information
Function	<p>It returns MMC information. The buffer 'pMmcInfo' holds card information.</p> <p>pMmcInfo.Card : Card types - MMC_CARD_UNDETECT : Card not detected - MMC_CARD_MMC : MMC - MMC_CARD_OTHER : Other card</p> <p>pMmcInfo.WProtect : Write-protection status - MMC_NO_PROTECT : Write-protection cancel - bit1: MMC_W_PROTECT_SOFT : Software write-protection</p> <p>pMmcInfo.MemSize : Card capacity(byte) pMmcInfo.MaxBlkNum : Maximum block number of the media</p> <p>When 'pMmcInfo.MemSize' is 0xFFFFFFFF, 'pMmcInfo.MaxBlkNum' +1 indicates the number of the media and the size is ('pMmcInfo.MaxBlkNum'+1)*512.</p>
Return Value	Returns the result of obtaining card information MMC_OK : Successful operation MMC_ERR_PARAM : Parameter error MMC_ERR_OTHER : Other errors

3.3 Data Structure

Data structure is showed as follow.

Definition of Card Information Data Structure

```

typedef struct {
    uint8_t    Card;           /* Card type                */
    uint8_t    WProtect;      /* Write-protection status  */
    uint32_t   MemSize;       /* Card capacity            */
    uint32_t   MaxBlkNum;     /* The number of the max blocks */
} MMC_INFO;                 /* total 12byte            */

```

3.4 Definitions

Definitions are showed as follow.

```

/*----- Definitions of return value -----*/
#define MMC_OK                (int16_t)( 0)          /* Successful operation      */
#define MMC_ERR_PARAM         (int16_t)(-1)         /* Parameter error          */
#define MMC_ERR_HARD          (int16_t)(-2)         /* Hardware error           */
#define MMC_ERR_CRC           (int16_t)(-3)         /* CRC error                */
#define MMC_ERR_WP            (int16_t)(-4)         /* Write-protection error   */
#define MMC_ERR_MBLKCMD       (int16_t)(-5)         /* Multi-block command error */
#define MMC_ERR_IDLE          (int16_t)(-6)         /* Idle state error         */
#define MMC_ERR_OTHER         (int16_t)(-7)         /* Other error              */

/*----- Definitions of flag -----*/
#define MMC_TRUE              (uint8_t)0x01         /* Flag "ON"                */
#define MMC_FALSE             (uint8_t)0x00         /* Flag "OFF"               */

/*----- Definition of card type -----*/
#define MMC_CARD_UNDETECT     (uint8_t)0x00         /* Card is not found        */
#define MMC_CARD_MMC          (uint8_t)0x01         /* MMC                      */
#define MMC_CARD_OTHER        (uint8_t)0xFF         /* Other card               */

/*----- Definitions of write-protection status -----*/
#define MMC_NO_PROTECT        (uint8_t)0x00         /* None setting             */
#define MMC_W_PROTECT_SOFT    (uint8_t)0x02         /* Software write-protection */

```

4. Setup Examples

4.1 r_mtl_XXX : Variable Data Setup Example

This section is for setting the resources of each user system.
The setting should be made in the [/**SET**/] comment of each file.
An excerpt of each file is provided with detailed comments.
The following are an example in case of RX610.

4.1.1 r_mtl_com.h

This file is a common header file.

(1) Define the SFR header file

Include the header file that contains the definition of the MCU function registers.
This file must be included due to device drivers accessing ports designated in it.
The following is an example of the setting when not using the Renesas MCU SFR header file.
When using MMC driver, include the SFR header file.

```
/* In order to use definitions of MCU SFR area,          */ /** SET **/  
/* include the header file of MCU SFR definition.      */ /** SET **/  
#include "iodefine.h" /* definition of MCU SFR        */ /** SET **/
```

(2) Define the software loop timer

When using the loop timer, include following header file.
The loop timer process is used for waiting duration of device driver.
The following is an example of the setting when using the software loop timer.

```
/* When not using the loop timer, put the following 'include' as comments. */  
#include "r_mtl_tim.h"
```

(3) Define Endian type

Set the following definitions to configure the File System to the Endian type in the user system.
In the case of RX family, please appoint it by the setting of the endian of the compiler option.

```
#if ( (defined(__LIT)) || (!defined(__BIG)) )  
#define MTL_MCU_LITTLE /* Little Endian          */ /** SET **/  
#endif
```

(4) Define fast operation of Endian process

When using M16C, define 'MTL_ENDI_HISPEED(high-speed function)'.
When using the RX family, put the following 'define' as comments.

```
//#define MTL_ENDI_HISPEED /* Uses the high-speed function. */ /** SET **/
```

(5) Specify type of user standard library

Specify the type of standard library in the user system.

When using the library bundled with the compiler for the processes stated below, add the listed define definitions as comments.

When using the optimized library, define the optimized library.

```

/* Specify the type of user standard library.                **/ ** SET **/
/* When using the compiler-bundled library for the following processes, **/ ** SET **/
/* put the following 'define' as comments.                  **/ ** SET **/
/* memcmp()/memmove()/memcpy()/memset()/strcat()/strcmp()/strcpy()/strlen() **/ ** SET **/
#define MTL_USER_LIB          /* use optimized library      */ /* ** SET **/

```

(6) Define the RAM area for access by process group

When using the RX family, define MTL_MEM_NEAR.

```

/* Define the RAM area to be accessed by the user process. */ /* SET **/
/* Efficient operations for standard functions and processes are applied. **/ ** SET **/
// #define MTL_MEM_FAR /* Defines 'FAR' as 'far' attribute for RAM area. (For M16C
Family) **/ ** SET **/
#define MTL_MEM_NEAR /* No far/near attribute for RAM area. */ /* SET **/

```

4.1.2 r_mtl_tim.h

When including r_mtl_tim.h, it is enable.

The value depends on clock frequency and wait of MCU.

Set the software timer to be used for internal operations.

```

/* Define the counter value for the timer.                */
/* Specify according to the user MCU, clock and wait requirements. */
/*                                                         */

/* Set the reference value to 10% more than the actual calculated value. */
/*=====*/

/*=====*/
#ifdef MTL_TIM_RX600__12_5MHz_noWait_Ix8
/* Setting for 12.5MHz no wait Ix8 = 100MHz(Compile Option "-optimize=2 -
size")*/
#define MTL_T_1US          24      /* loop Number of 1us      */
#define MTL_T_2US          50      /* loop Number of 2us      */
#define MTL_T_4US          100     /* loop Number of 4us      */
#define MTL_T_5US          125     /* loop Number of 5us      */
#define MTL_T_10US         201     /* loop Number of 10us     */
#define MTL_T_20US         501     /* loop Number of 20us     */
#define MTL_T_30US         751     /* loop Number of 30us     */
#define MTL_T_50US        1251     /* loop Number of 50us     */
#define MTL_T_100US        2501    /* loop Number of 100us    */
#define MTL_T_200US        5020    /* loop Number of 200us    */
#define MTL_T_300US        7525    /* loop Number of 300us    */
#define MTL_T_400US        9999    /* loop Number of 400us    */
#define MTL_T_1MS          24997   /* loop Number of 1ms      */
#endif

```

4.2 MMC Driver : Variable Data Setup Example

This section is for setting the resources of each user system.
The setting should be made in the [/**SET**/] comment of each file.
An excerpt of each file is provided with detailed comments.
The following are an example in case of RX610.

4.2.1 r_mmc.h(Common header file)

(1) Define number of slots (devices) and slot number

Specify number of slots (devices) and slot number.

```

/* Define number of required card slots. (1-N slots) */
/* Define slot number in accordance with the number of card slots to be connected. */
/*-----*/
/* Define number of slots (devices). */
#define MMC_SLOT_NUM      1          /* 1slots          */ /** SET **/

/* Define slot number. */
#define MMC_SLOT0        0          /* Slot 0          */ /** SET **/
#define MMC_SLOT1        1          /* Slot 1          */ /** SET **/

```

(2) Define use of single block commands with SPI mode

Do not make any changes.

```

/* When use the card which does not support a multi-block command, please define it. */
/* Use single block commands in the case of the card which does not support multiple block */
/* commands. */
#if 1          /** SET **/
#define MMC_SBLK_CMD      /* Support single block commands */ /** SET **/
#endif

```

(3) Define card type

Define MMC_SUPPORT_MMC.

```

/*-----*/
/* Please define the media to support. */
/*-----*/
#define MMC_SUPPORT_MMC      /* MMC          */ /** SET **/

```

4.2.2 r_mmc_sfr.h (Header file for SFR)**(1) Define resources**

The data transfer depends on MCU resource for use with.
Select one of the following for use as your system.

```

/*-----*/
/* Define the combination of the MCU's resources. */
/*-----*/
#define MMC_OPTION_1 /* Low speed */ /* SCI */ /* SET */
#define MMC_OPTION_2 /* High speed */ /* SCI + CRC calculation circuit */ /* SET */

```

(2) Define control ports

Define a control port to use to a connection circuit.

```

/*-----*/
/* Define the control port. */
/*-----*/

#define MMC_DR_DATAO PORT2.DR.BIT.B6 /* MMC DataOut */ /* SET */
#define MMC_PORT_DATAI PORT2.PORT.BIT.B5 /* MMC DataIn */ /* SET */
#define MMC_DR_CLK PORT2.DR.BIT.B7 /* MMC CLK */ /* SET */
#define MMC_DDR_DATAO PORT2.DDR.BIT.B6 /* MMC DataOut */ /* SET */
#define MMC_DDR_DATAI PORT2.DDR.BIT.B5 /* MMC DataIn */ /* SET */
#define MMC_DDR_CLK PORT2.DDR.BIT.B7 /* MMC CLK */ /* SET */
#define MMC_ICR_DATAI PORT2.ICR.BIT.B5 /* MMC DataIn */ /* SET */

#define MMC_MSTPCR_SCI SYSTEM.MSTPCRB.BIT.MSTPB30 /* SCI Module stop setting*/ /* SET */

#define MMC_DR_CS0 PORT7.DR.BIT.B0 /* MMC CS0 (Negative-true logic) */ /* SET */
#define MMC_DDR_CS0 PORT7.DDR.BIT.B0 /* MMC CS0 (Negative-true logic) */ /* SET */

#define MMC_PORT_DETECT0 PORT7.PORT.BIT.B1 /* MMC DETECT0 (Negative-true logic)*/ /* SET */
#define MMC_DDR_DETECT0 PORT7.DDR.BIT.B1 /* MMC DETECT0 (Negative-true logic)*/ /* SET */
#define MMC_ICR_DETECT0 PORT7.ICR.BIT.B1 /* DETECT0 DataIn */ /* SET */

#if (MMC_SLOT_NUM > 1)
#define MMC_DR_CS1 /* FLASH CS1(Negative-true logic) */ /* SET */
#define MMC_DDR_CS1 /* FLASH CS1(Negative-true logic) */ /* SET */

#define MMC_PORT_DETECT1 /* MMC DETECT1 (Negative-true logic)*/ /* SET */
#define MMC_DDR_DETECT1 /* MMC DETECT1 (Negative-true logic)*/ /* SET */
#define MMC_ICR_DETECT1 /* DETECT1 DataIn */ /* SET */
#endif /* #if (MMC_SLOT_NUM > 1) */

```

(3) Define bit rate

As for transfer speed setting, it is necessary to meet tODLY of both Identification mode and Data Transfer mode in spec.

In addition, it is necessary to meet tOD (100KHz <= tOD <= 400KHz) at Identification mode and tPP (0.1MHz <= tPP <= 20MHz (*)) at Data Transfer mode.

The frequency of tOD and tPP mean the frequency of SCLK in this device driver.

As for maximum frequency, make a confirmation of each MCU datasheet.

```

/*-----*/
/* Define the value of the bit rate register according to a communication baud rate.          */
/* Set the frequency of CLK to 6MHz or less.                                              */
/* The possible maximum transfer frequency of CLK is depends on hardware circuit          */
/* and MCU conditions.                                                                    */
/* Refer to MCU hardware manual/memory card specifications and specify the buad rate.     */
/* When operating card with SPI mode,                                                    */
/* specify the following two definitions of Identification mode and Data Transfer mode.     */
/* Specify the definition to meet tODLY of both Identification mode and Data Transfer mode. */
/* In addition, meet tOD (100KHz <= tOD <= 400KHz) at Identification mode                */
/* and tPP (0.1MHz <= tPP <= 20MHz ) at Data Transfer mode.                             */
/* The maximum frequency depends on MCU type.                                            */
/*                                                                                          */
/*BRR = (PCLK / (8 * 2 ^ (2n - 1) * B)) - 1
/*PCLK: Operating frequency [MHz]
/*B   : Bit rate [bit/s]
/*n   : Determined by the SMR settings shown in the following table.
/*
/*  CKS1 | CKS0 | n
/* -----+-----+-----
/*  0    |  0    |  0
/*  0    |  1    |  1
/*  1    |  0    |  2
/*  1    |  1    |  3
/*-----*/
/* PCLK = 50MHz, n=0 */
#define MMC_UBRG_IDENTIFICATION (uint8_t)0x1f /* BRR identification mode setting*/ /** SET **/
/*                                     +------+ 391KHz          */ /** SET **/
#define MMC_UBRG_D_TRANSFER (uint8_t)0x01 /* BRR data Transfer mode setting */ /** SET **/
/*                                     +------+ 6.25MHz         */ /** SET **/

```

* : When I use SCI of the RX family, I cannot produce the maximum clock frequency of card specifications.
With each data sheet, please confirm maximum clock frequency.

5. Method for connecting to MCU and MCU resource for use with

5.1 MCU resource for use with

This software controls as follows:

Data input/output is controlled by clock synchronous serial I/O (internal clock).

Allocate CMOS output port and set CMOS output of the clock synchronous serial I/O in order to perform high-speed processing.

Please do CMOS output setting.

The transmission control detects the space of the transmission buffer, and use a transmission interrupt request bit without using an interrupt. Therefore, I set it about an interrupt as follows.

interrupt level is level 0 (interrupt is prohibited in it).

Connect Card CS# pin to RX Port and control it by RX general port setting.

Resources	RX610
SCI (clocked synchronous mode)	M
DMA	Unused
CRC operation circuit	R
Port for CS#: 1port/Card	M
Port for Card detection: 1port/Card	M
Port for Power Control: 1port/Card	M

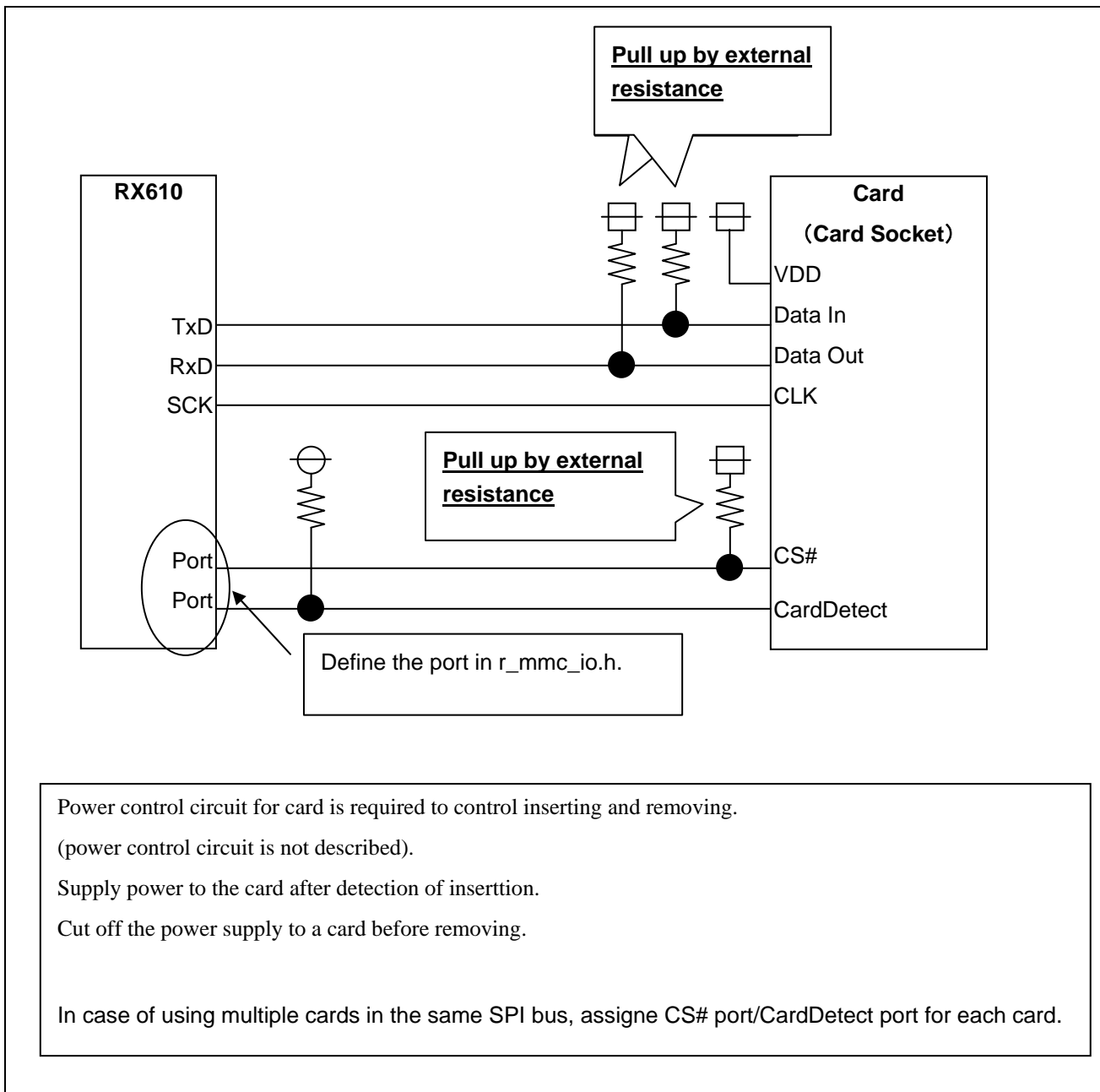
M: mandatory

R: recommended (high-speed processing is enabled when unique resource of RX is used)

Therefore, RX family MCU in SCI Clocked Synchronous mode (LSB first) can be operated.

5.2 Method for connecting to MCU

An example of connecting to RX610 is showed.
In case of other RX family MCUs, the same connection is made



6. Notes for Application Development

6.1 Notes for use

- Configure the software according to the hardware.
- Remove card after deactivation, setting signals between MCU and card into Hi-z and cutting off power supply to card.
- In case that insertion/removable circuit is not realized, inserting/removing card might cause the power source to be unstable and reset MCU.

6.2 Notes for embedding

6.2.1 Files for including

Include "r_mmc.h" and "r_mtl_com.h" when embedding this device driver.

It is necessary for include to do "r_mtl_com.h" first.

6.2.2 Notes for linking

RX Family C/C++ Compiler V.1.00.00.001 or later.

6.3 Reference value of ROM size

Approx. 6.1kByte

Compile conditions

MCU: Renesas RX family

Compiler: RX Family C/C++ Compiler V.1.00.00.001

(optimize level =2, optimize method = size)

6.4 Static RAM size for use with

64byte

Compile conditions

MCU: Renesas RX family

Compiler: RX Family C/C++ Compiler V.1.00.00.001

(optimize level =2, optimize method = size)

6.5 Stack size

Function	Stack size
R_mmc_Init_Driver	24
R_mmc_Init_Slot	112
R_mmc_Detach	24
R_mmc_Chk_Detect	12
R_mmc_Read_Data	120
R_mmc_Write_Data	128
R_mmc_Get_MmclInfo	16

6.6 Notes on insertion/removal of the card

Enable to detect the insertion or removal of the card using the function "R_mmc_Chk_Detect()" with the card detection pins, which comes with the card connector.

Therefore, it is recommended to detect the insertion or removal of the card by software polling.

The driver returns an error when the card is removed in a data transmission eventually.

However, the driver may not return an error when the card is removed momentarily in a data transmission in case of the following conditions:

1. Data transmission is operated properly when no response error from the card occurs, because the driver cannot detect the insertion or removal of the card by software polling.
2. The driver may recognize the completion of writing to the card when the card is inserted or removed momentarily in writing stage. This is because of the specification that a writing completion will be detected by "H" signal of DataIn pin. DataIn pin is pulled up.

Please avoid this problem by the system hardware such as a hardware interrupt control and polling period time etc.

6.7 Note of the Hi-z setting processing of the port about the exclusion and adding of the card

In the insertion of the card, Please insert card after having set CS#, DataIn, DataOut, and Clock terminal in Hi-z. Please supply the power supply to a card afterwards.

In the extraction of the card, After the power supply stop to a card, After the power supply supply stop to a card, please extract a card, after having set CS#, DataIn, DataOut, and Clock terminal in Hi-z.

The CS#, DataIn, DataOut, CLK terminal of the card is assigned to SIO or a port terminal of MCU, but does not process Hi-z by this driver because the case that the port is assigned to other resources is expected.

Therefore, please make the Hi-z processing of the MCU terminal in the high rank side in the exclusion and adding of the card.

7. Sample program usage

This section explains the sample program for MMC driver usage.

7.1 Outline

Build the sample software workspace and download the abs file to the RSK*.

After the "Reset Go" button is clicked, program starts running.

After having inserted MMC, using last 10 blocks of the memory in the MMC, compares it with the reading and writing of data.

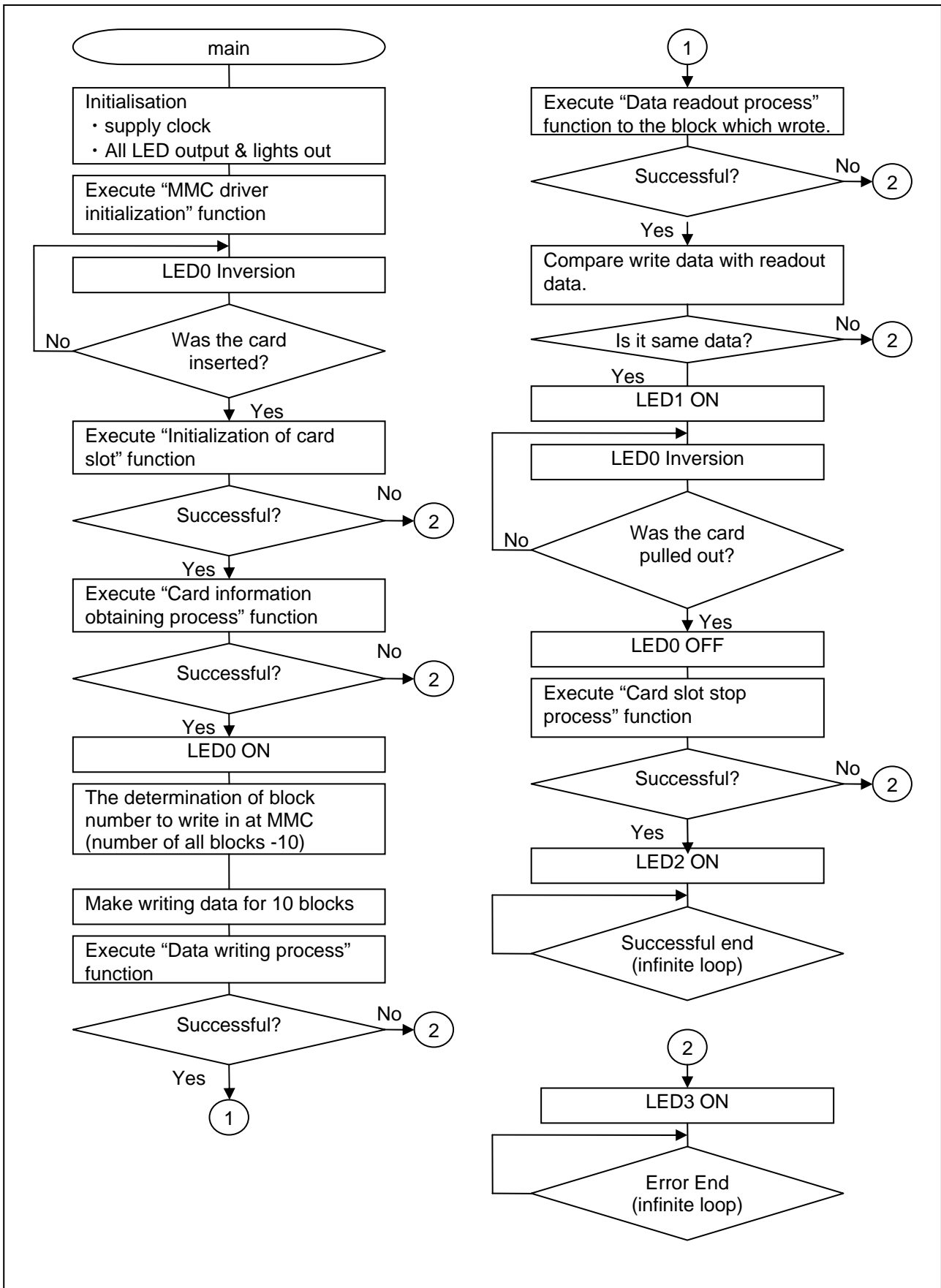
After having executed the above, extracts a card, and it is the movement end.

Displays the progress of the program and a result with LED. Please refer to a list shown below for the contents of the indication.

LED0	ON : A card is put. OFF : A card is inserted. BLINK : Require the drawing of the card or the insertion
LED1	ON : Write/Read Execution Successful.
LED2	ON : Program Successful
LED3	ON : Program Error

(*)RSK refers to
Renesas Starter Kit for RSKRX610 - R0K556108C000BR

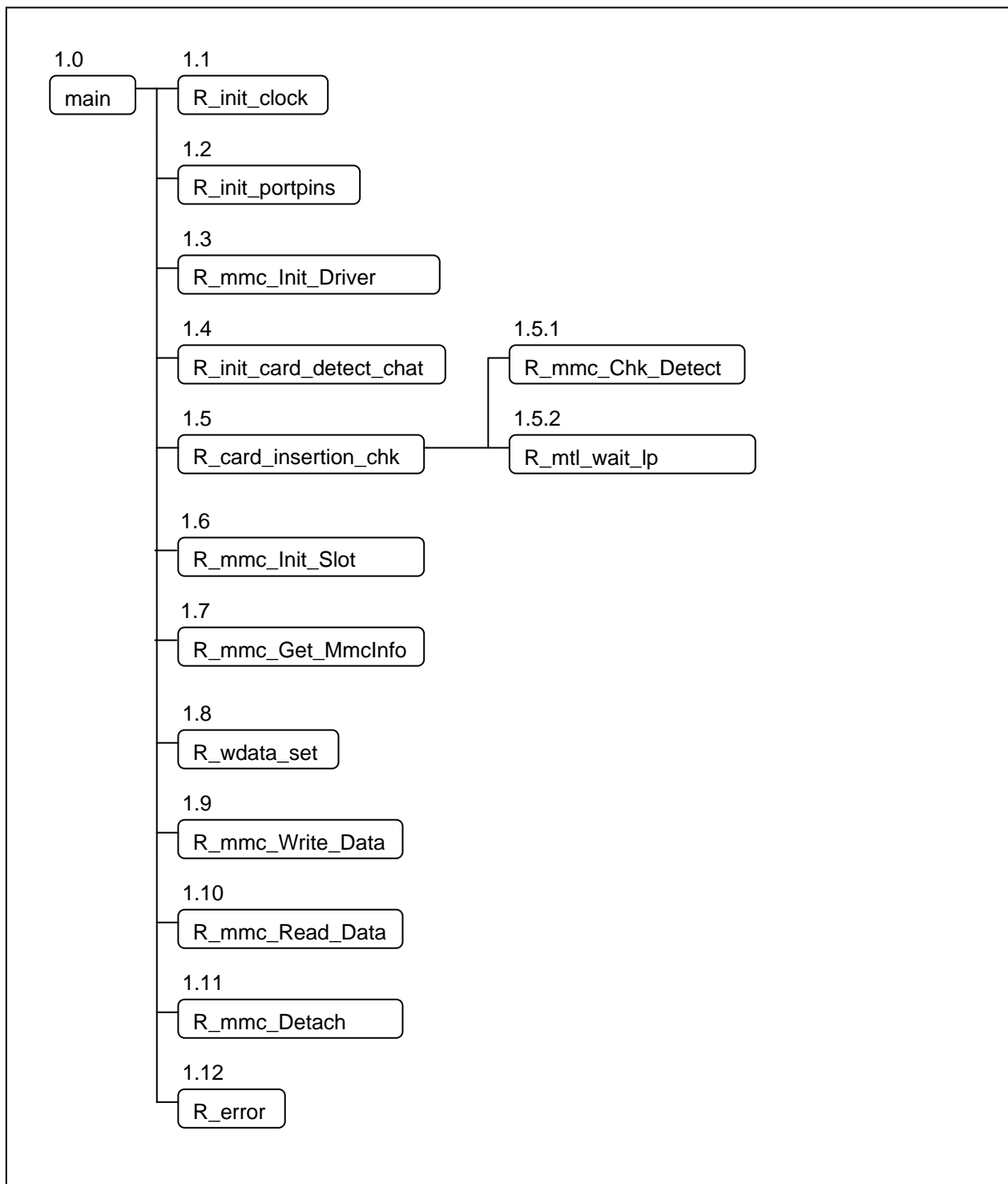
7.2 Flow



7.3 Function list

No	Function Name	Outline
1.0	main	When MMC is inserted, reads and writes a random value to a card and compares each.
1.1	R_init_clock	The clock of the microcomputer and other clock related registers are initialized.
1.2	R_init_portpins	Initializes the port pins for LEDs.
1.3	R_mmc_Init_Driver	Initializes the MMC driver processing. - This is API function.
1.4	R_init_card_detect_chat	Initializes memory for the card detection.
1.5	R_card_insertion_chk	Checks a having card or not state.
1.5.1	R_mmc_Chk_Detect	Returns a having card or not state. - This is API function.
1.5.2	R_mtl_wait_lp	It is time waiting processing before confirming the next terminal state.
1.6	R_mmc_Init_Slot	When a card is inserted, I execute initialization processing for the card. - This is API function.
1.7	R_mmc_Get_MmcInfo	Gets card information. - This is API function.
1.8	R_wdata_set	Makes data to write in at MMC.
1.9	R_mmc_Write_Data	Writes data to a MMC. - This is API function.
1.10	R_mmc_Read_Data	Reads data from a MMC. - This is API function.
1.11	R_mmc_Detach	When a card is pulled up, I initialize it for a slot. - This is API function.
1.12	R_error	Error handling function

7.4 Function chart



Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb.21.2011	-	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

- The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable.

When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

- The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

- All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
- Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
- Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
- When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
- Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
- Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
"Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
- You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
- Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
- Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
- Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

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

Renesas Electronics America Inc.

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

Renesas Electronics Canada Limited

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

Renesas Electronics Europe Limited

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

Renesas Electronics Europe GmbH

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

Renesas Electronics (China) Co., Ltd.

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

Renesas Electronics (Shanghai) Co., Ltd.

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

Renesas Electronics Hong Kong Limited

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

Renesas Electronics Taiwan Co., Ltd.

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

Renesas Electronics Singapore Pte. Ltd.

1 HarbourFront Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.

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

Renesas Electronics Korea Co., Ltd.

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