To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)
Send any inquiries to http://www.renesas.com/inquiry.
Notice

1. 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.

2. 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.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please 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.
1. Abstract

This document describes the I²C version program downloader for the R8C/35A Group.

2. Introduction

The application example described in this document applies to the following MCU and parameter:

- MCU: R8C/35A Group
- XIN clock frequency: 20 MHz

The sample program may include operations of unused bit functions for the convenience of the SFR bit layout. Set the values according to the operating conditions of the user system.
3. Program Downloader Overview

3.1 Downloader Specifications

- The system program (including program downloader process) is allocated to block 0.
- The program downloader erases and writes mainly to user programs other than the user program in block 0. The program downloader ignores rewrite operations to block 0.
- EW0 mode is used by the program downloader for rewriting the CPU.
- In a reset start, the program downloader checks the state of port P3_3 and selects either to use the program downloader or the user program. The program downloader operates when port P3_3 is high, and the user program operates when port P3_3 is low.
- The virtual fixed vector table is allocated to block 1 to use the fixed vector table interrupt in the user program.
- \(^{1} \text{C}-\text{bus slave mode is used to communicate with a programmer.}\)
- 1010101b is used for the slave address.
- Refer to 4. Downloader Communication Protocol for the communication protocol.

Figure 3.1 shows an example of a Connection, Figure 3.2 shows the Transfer Format, Figure 3.3 shows the Memory Map (32 Kbyte ROM MCU), and Figure 3.4 shows an example of the System Interrupt Operation (Overflow Interrupt).

![Connection Figure](image1)

![Transfer Format Figure](image2)
Figure 3.3 Memory Map (32 Kbyte ROM MCU)

Notes:
1. ID code checked by the program downloader whether IDs match the IDs transmitted from the programmer.
2. ID code checked in boot mode whether IDs match the IDs transmitted from the programmer.
3. Do not use these interrupts. These are for use with development tools only.
4. The setting values for the option function select register (OFS) and option function select register 2 (OFS2) can be set by the user.

Figure 3.4 System Interrupt Operation (Overflow Interrupt)
3.2 Timing after Reset

When the program downloader is activated, input a high to P3_3 pin before reset is deasserted and retain high until the program downloader is selected.

![Signal Control Timing after Reset Diagram](image)

**Figure 3.5 Signal Control Timing after Reset**

3.3 Initial Settings

1. Option function select register (OFS)
   The OFS register is assigned to the highest-order address 0FFFFh in the fixed vector table. Set the OFS register by a program of the program downloader.

2. Option function select register 2 (OFS2)
   The OFS2 register is assigned to 0FFDBh in the reserved area. Set the OFS2 register by a program of the program downloader.

3. Watchdog timer
   When using the watchdog timer, enable the WDT_USE definition in the fla_r835a.inc file.

Note:
1. Time when the low-speed on-chip oscillator (60 kHz (min)) operates.
# 3.4 Registers

### Port P3 Direction Register

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
<th>Port PD3_3 bit</th>
<th>Port PD3_5 bit</th>
<th>Port PD3_7 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0: Input mode</td>
<td>0: Input mode</td>
<td>0: Input mode</td>
</tr>
</tbody>
</table>

*Port PD3_3 bit:*
0: Input mode

*Port PD3_5 bit:*
0: Input mode (functions as SCL pin)

*Port PD3_7 bit:*
0: Input mode (functions as SDA pin)

### Watchdog Timer Reset Register

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
<th>Writing 00h and then FFh to this register initializes the watchdog timer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

*Note:*
1. This setting is unnecessary when the watchdog timer is not used.

### Protect Register

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
<th>Enables writing to registers CM0, CM1, CM3, OCD, FRA0, FRA1, FRA2, and FRA3.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0: Write disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Write enabled</td>
</tr>
</tbody>
</table>

### System Clock Control Register 0

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

*XIN clock (XIN-XOUT) stop bit:*
0: XIN clock oscillates

*CPU clock division select bit 0:*
0: Bits CM16 and CM17 in CM1 register enabled
1: Divide-by-8 mode

*XIN, XCIN clock select bit:*
0: XIN clock

### System Clock Control Register 1

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Port XIN-XOUT switch bit:*
1: XIN-XOUT pin

*Low-speed on-chip oscillator stop bit:*
0: Low-speed on-chip oscillator on

*CPU clock division select bit 1:*
00: No division mode
Oscillation Stop Detection Register  OCD [address 000Ch]

```
  b7 b0
  0 1
```

- System clock select bit
- 0: XIN clock selected

Interrupt Control Register  IICIC [address 004Fh]

```
  b7 b0
  0 0 0 0 0 0 0
```

- Interrupt priority level select bit
- 000: Level 0 (interrupt disabled)

Module Standby Control Register  MSTCR [address 0008h]

```
  b7 b0
  0 0
```

- SSU, I²C-bus standby bit
- 0: Active

SSU/IIC Pin Select Register  SSUIICSR [address 018Ch]

```
  b7 b0
  1 0
```

- SSU/I²C-bus switch bit
- 1: I²C-bus function selected

IIC bus Status Register  ICSR [address 019Ch]

```
  b7 b0
  0 0 0 0 0 0
```

- Slave address recognition flag
  - This flag is set to 1 when the first frame immediately after the start condition matches bits SVA0 to SVA6 in the SAR register in slave receive mode (slave address detection and general call address detection)
- Stop condition detection flag
  - This flag is set to 1 when a stop condition is detected after the frame is transferred.
- No acknowledge detection flag
  - This flag is set to 1 when no ACKnowledge is detected from the receive device after transmission.
- Receive data register full flag
  - This flag is set to 1 when receive data is transferred from registers ICDRS to ICDRR.
- Transmit end flag
  - This flag is set to 1 at the rising edge of the 9th clock cycle of the SCL signal while the TDRE bit is set to 1.
- Transmit data empty flag
  - This flag is set to 1 when:
    - Data is transferred from registers ICDRT to ICDRS and the CDRT register is empty
    - The TRS bit in the ICCR1 register is set to 1 (transmit mode)
    - A start condition is generated (including retransmission)
    - Slave receive mode is changed to slave transmit mode
IIC bus Control Register 1    ICCR1 [address 0198h]

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Transmit clock select bits 3 to 0
- This bit is used for maintaining the setup time in transmit mode of slave mode. The time is 10Tcyc when the CKS3 bit is set to 0 and 20Tcyc when the CKS3 bit is set to 1. \( (1Tcyc = 1/f_1(s)) \)
- Transfer/receive select bit, Master/slave select bit
- 00: Slave Receive Mode
- Receive disable bit
- After reading the ICDRR register while the TRS bit is set to 0
- 0: Next receive operation continues
- \( ^2 \)C-bus interface enable bit
- 1: This module is enabled for transfer operations (Pins SCL and SDA are in a bus drive state)

IIC bus Control Register 2    ICCR2 [address 0199h]

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- Set each bit to initialize.
- SCL monitor flag
- 0: SCL pin is set to “L”.
- 1: SCL pin is set to “H”.
- Set each bit to initialize.

IIC bus Mode Register     ICMR [address 019Ah]

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bit counters 2 to 0
- 000: 9 bits
- MSB-first/LSB-first select bit
- 0: Data transfer with MSB-first

IIC Interrupt Enable Register     ICIER [address 019Bh]

<table>
<thead>
<tr>
<th>b7</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Transmit acknowledge select bit
- 0: In receive mode, 0 is transmitted as the acknowledge bit.
- Acknowledge bit detection select bit
- 1: When the receive acknowledge bit is set to 1, continuous transfer is halted.
IIC bus Transmit Data Register 1     ICDRT [address 0194h]

This register stores transmit data. When the ICDRS register is detected as empty, the stored transmit data item is transferred to the ICDRS register and data transmission starts. When the next unit of transmit data is written to the ICDRT register while data is transmitted to the ICDRS register, continuous transmission is enabled.

IIC bus Receive Data Register 1     ICDRR [address 0196h]

This register stores receive data. When the ICDRS register receives 1 byte of data, the receive data is transferred to the ICDRR register and the next receive operation is enabled.

Flash Memory Status Register     FST [address 01B2h]

Program error flag
0: No program error
1: Program error

Erase error/blank check error flag
0: No erase error/blank check error
1: Erase error/blank check error

Ready/busy status flag
0: Busy
1: Ready
### Flash Memory Control Register 0 (FMR0 [address 01B4h])

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7</td>
<td>CPU rewrite mode select bit</td>
<td>0/1</td>
</tr>
<tr>
<td>b6</td>
<td>EW1 mode select bit</td>
<td>0</td>
</tr>
<tr>
<td>b5</td>
<td>Flash memory stop bit</td>
<td>0</td>
</tr>
<tr>
<td>b4</td>
<td>Erase/write error interrupt enable bit</td>
<td>0</td>
</tr>
<tr>
<td>b3</td>
<td>Flash access error interrupt enable bit</td>
<td>0</td>
</tr>
<tr>
<td>b2</td>
<td>Flash ready status interrupt enable bit</td>
<td>0</td>
</tr>
</tbody>
</table>

**Explanation:**
- **CPU rewrite mode select bit:**
  - 0: CPU rewrite mode disabled
  - 1: CPU rewrite mode enabled
- **EW1 mode select bit:**
  - 0: EW0 mode
- **Flash memory stop bit:**
  - 0: Flash memory operates
- **Erase/write error interrupt enable bit:**
  - 0: Erase/write error interrupt disabled
- **Flash access error interrupt enable bit:**
  - 0: Flash access error interrupt disabled
- **Flash ready status interrupt enable bit:**
  - 0: Flash ready status interrupt disabled

### Flash Memory Control Register 1 (FMR1 [address 01B5h])

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7</td>
<td>Lock bit disable select bit</td>
<td>0/1</td>
</tr>
<tr>
<td>b6</td>
<td>Data flash block A rewrite disable bit</td>
<td>0/1</td>
</tr>
<tr>
<td>b5</td>
<td>Data flash block B rewrite disable bit</td>
<td>0/1</td>
</tr>
<tr>
<td>b4</td>
<td>Data flash block C rewrite disable bit</td>
<td>0/1</td>
</tr>
<tr>
<td>b3</td>
<td>Data flash block D rewrite disable bit</td>
<td>0/1</td>
</tr>
</tbody>
</table>

**Explanation:**
- **Lock bit disable select bit:**
  - 0: Lock bit enabled
  - 1: Lock bit disabled
- **Data flash block A rewrite disable bit:**
  - 0: Rewrite enabled (software command acceptable)
  - 1: Rewrite disabled (software command not acceptable, no error occurred)
- **Data flash block B rewrite disable bit:**
  - 0: Rewrite enabled (software command acceptable)
  - 1: Rewrite disabled (software command not acceptable, no error occurred)
- **Data flash block C rewrite disable bit:**
  - 0: Rewrite enabled (software command acceptable)
  - 1: Rewrite disabled (software command not acceptable, no error occurred)
- **Data flash block D rewrite disable bit:**
  - 0: Rewrite enabled (software command acceptable)
  - 1: Rewrite disabled (software command not acceptable, no error occurred)
### 3.5 Memory

#### Table 3.1 Memory

<table>
<thead>
<tr>
<th>Assigned Memory</th>
<th>Size</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>1461 bytes</td>
<td>System program only (including fixed vector and variable vector table)</td>
</tr>
<tr>
<td>RAM</td>
<td>427 bytes</td>
<td>System program only</td>
</tr>
</tbody>
</table>

#### Table 3.2 RAM and Definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ram_execute</td>
<td>128 bytes</td>
<td>EW0 mode program area</td>
</tr>
<tr>
<td>status_flags</td>
<td>1 byte</td>
<td>Serial flag area</td>
</tr>
<tr>
<td>reset_blank</td>
<td>–</td>
<td>User program blank flag</td>
</tr>
<tr>
<td>srd1</td>
<td>1 byte</td>
<td>SRD1 register</td>
</tr>
<tr>
<td>srd08</td>
<td>–</td>
<td>SR8 bit</td>
</tr>
<tr>
<td>srd09</td>
<td>–</td>
<td>SR9 bit</td>
</tr>
<tr>
<td>srd10</td>
<td>–</td>
<td>SR10 bit</td>
</tr>
<tr>
<td>srd11</td>
<td>–</td>
<td>SR11 bit</td>
</tr>
<tr>
<td>srd12</td>
<td>–</td>
<td>SR12 bit</td>
</tr>
<tr>
<td>srd13</td>
<td>–</td>
<td>SR13 bit</td>
</tr>
<tr>
<td>srd14</td>
<td>–</td>
<td>SR14 bit</td>
</tr>
<tr>
<td>srd15</td>
<td>–</td>
<td>SR15 bit</td>
</tr>
<tr>
<td>srd</td>
<td>1 byte</td>
<td>SRD register</td>
</tr>
<tr>
<td>address</td>
<td>4 bytes</td>
<td>Address data</td>
</tr>
<tr>
<td>temp</td>
<td>32 bytes</td>
<td>Temporary (used with the stack area)</td>
</tr>
<tr>
<td>rx_data</td>
<td>1 byte</td>
<td>Receive data</td>
</tr>
<tr>
<td>tx_data</td>
<td>1 byte</td>
<td>Transmit data</td>
</tr>
<tr>
<td>page_buffer</td>
<td>256 bytes</td>
<td>Page buffer</td>
</tr>
</tbody>
</table>
3.6 Flowchart

(1) Startup handling 1

- **reset**
  - `bclr pd3_3` ; Set port P3_3 to input mode.

- **Program downloader selected? (Port P3_3 = H?)**
  - Yes (program downloader selected)
    - `ldc #00h,FLG` ; Initialize FLG.
    - `ldc #boot_stack,SP` ; Set SP.
    - `ldc #SB_base,SB` ; Set SB.
    - `ldintb #VECTOR_ADR` ; Set INTB.
  - No (user program selected)
    - `bclr prc0` ; Disable system clock control register protect.
    - `bclr cm14` ; Select low-speed on-chip oscillator.
    - `bset cm13` ; Select XIN-XOUT pin.
    - `bclr cm05` ; XIN clock on
    - `mov.w #0,a1` ; Select XIN clock.

- **Wait until oscillation stabilizes 2040 > a1**
  - Yes
    - `inc.w a1`
  - No

- **Watchdog timer reset**
  - `wdt_reset`

- **bset prc0**
  - `bclr cm07`
  - `bclr ocd2`
  - `and.b #00111111b,cm1` ; Select XIN clock as the system clock.
  - `bclr cm06`
  - `bclr prc0` ; Select CPU clock no division.
  - `Watchdog timer reset`
  - `wdt_reset`

- **iic_initial**
(2) Startup handling 2

```
and.b  #01010111b,pd3
mov.b  #00000000b,iioic
bcfr  mstiic
bset  iisel
bcfr  stop_icr
bset  ice_icr1
mov.b  #10010000b,iicr1
mov.b  #01110000b,iicr2
mov.b  #00000000b,icmr
mov.b  #00000100b,icier
and.b  #00000000b,icsr
mov.b  #SLAVE_ADDRESS,sar
(Note 1)
clear_static_ram
mov.w  #0,r0
mov.w  #(RAM_END+1-RAM_TOP)/2,r3
mov.w  #RAM_TOP,a1
sstr.w
user_reset_blank_check

#0fffh == User_Reset_VEC

Yes
No
#0fffh == User_Reset_VEC+2

Yes
No
or.b  #00001100b,srd1
bset  reset_blank

command_handler

; Set ports P3_3, P3_5/SCL, P3_7/SDA to input mode.

; Initialize I2C.

; Clear RAM.

; Check user reset blank.

; Set ID identification.

; Set user reset blank.

Note:
1. SLAVE_ADDRESS = 0AAh
(3) Command handler

```
(3) Command handler

Watchdog timer reset
  
  no

  aas_icsr == 1

  yes

  bcr aas_icsr
  
  bcr addit_icier
  
  mov.b icdr.rx_data

Byte receive handling
  
  get_byte

No (r0h = r0h - 1) == 0

Yes (command receive completed)

mov.b rx_data,r0l

#READ_STATUS == r0l

No

#VERSION_OUTPUT == r0l

No

#ID_CHECK == r0l

No

mov.b rdi,r0h

andi.b #0f0h

#0ch == r0h

yes (ID identified)

#PAGE_READ == r0l

No

#PAGE_PROGRAM == r0l

No

#BLOCK_ERASE == r0l

No

#CLEAR_STATUS == r0l

No

command_error

Stop condition receive handling
  
  stop_wait
```

; Wait until start condition + slave address are detected.

; Clear slave address recognition flag.

; Set ACK.

; Perform dummy read (slave address + RW).

; Command receive completed?

; ID identified?

; ID identified?
(4) Subroutine 1

```
(4) Subroutine 1

<table>
<thead>
<tr>
<th>cpu_fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>bset prc0</td>
</tr>
<tr>
<td>and.b #00111111b,cm1</td>
</tr>
<tr>
<td>bclr cm06</td>
</tr>
<tr>
<td>bclr prc0</td>
</tr>
<tr>
<td>rts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cpu_slow</th>
</tr>
</thead>
<tbody>
<tr>
<td>bset prc0</td>
</tr>
<tr>
<td>bset cm06</td>
</tr>
<tr>
<td>bclr prc0</td>
</tr>
<tr>
<td>rts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>wdt_reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>When WDT_USE is enabled</td>
</tr>
<tr>
<td>mov.b #000h,wdtr</td>
</tr>
<tr>
<td>mov.b #0ffh,wdtr</td>
</tr>
<tr>
<td>rts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>get_byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov.b #0,r0h</td>
</tr>
</tbody>
</table>

Watchdog timer reset

wtl_reset

stop_icsr == 1

Y (stop condition detected)

No

aas_icsr == 1

Y (slave address detected)

No

rdrf_icsr == 1

Y (reception completed)

No

Stop icsr == 1

Y (stop condition detected)

No

Aas_icsr == 1

Y (slave address detected)

No

Rdrf_icsr == 1

Y (reception completed)

When WDT_USE is enabled

mov.b icdrr,rx_data ; Store received data.

mov.b #1,r0h ; Set normal end value.

rts |
```
(5) Subroutine 2

```
put_byte

mov.b #0,r0h ; Set error end value.
```

```
Watchdog timer reset
wdr_reset

No

tdre_iccr2 == 1

Yes (no data in U0TB register)

mov.b tx_data,icdrt ; Set transmit data. → Release SCL line.
```

```
Watchdog timer reset
wdr_reset

No

tend_iccr == 1

Yes (transmission completed)

nackf_iccr == 1

Yes (NACK detected)
```

```
Yes (transmission completed)

mov.b #1,r0h ; Set normal end value.

bclr tend_iccr ; Clear transmission complete flag.
```

```
Watchdog timer reset
wdr_reset

No

tscl_iccr2 == 1

Yes (SCL pin is "H").

bclr trs_iccr1 ; Set to receive mode.

mov.b icdrr.rx_data ; Perform dummy read. → Release SCL line.

and.b #00101111b,icsr ; Clear flags TDRE, TEND, and NACKF.
```

```
No (SCL pin is "L").
```

```
rts
```
(6) Subroutine 3

**Restart Wait**

```
mov.b #0,r0h ; Set error end value.
```

**Watchdog Timer Reset**

```
wdt_reset
```

**Stop _icsr == 1**

Yes (stop condition detected)

No

**Aas _icsr == 1**

Yes (slave address matched?)

No

**Bclr aas _icsr**

; Clear slave address recognition flag.

```
mov.b icdrr,rx_data ; Perform dummy read (address + R/W)
```

**Trs _iccr1 == 1**

Yes (R/W direction bit = receive mode)

No (R/W direction bit = receive mode)

```
mov.b #1,r0h ; Set normal end value.
```

**Bclr trs _iccr1 ; Clear slave address recognition flag.**

```
and.b #00000000b,icsr
```

**Rts**

---

**Stop Wait**

**Watchdog Timer Reset**

```
wdt_reset
```

**Stop _icsr == 1**

Yes (stop condition detected)

No

**Aas _icsr == 1**

Yes (slave address matched?)

No

**Bclr aas _icsr**

; Clear slave address recognition flag.

```
mov.b icdrr,rx_data ; Perform dummy read (address + R/W)
```

**Trs _iccr1 == 1**

Yes (R/W direction bit = transmit mode)

No (R/W direction bit = receive mode)

```
mov.b #0ffh,tx_data
```

**Byte Receive Handling**

```
get_byte
```

**Byte Transmit Handling**

```
put_byte
```

---

**Bclr nackf _icsr ; Clear NACK detection flag.**

---

Yes (stop condition detected)

Yes (slave address detected)

No

Yes (R/W direction bit = transmit mode)

No  (R/W direction bit = receive mode)

Yes (R/W direction bit = transmit mode)

No  (R/W direction bit = receive mode)

---

**Bclr nackf _icsr ; Clear NACK detection flag.**

---

Yes (stop condition detected)

Yes (slave address detected)

No

Yes (R/W direction bit = transmit mode)

No  (R/W direction bit = receive mode)

---

Yes (stop condition detected)

Yes (slave address detected)

No

Yes (R/W direction bit = transmit mode)

No  (R/W direction bit = receive mode)

---
(7) Page read

page_read

mov.b #00h,address ; Set low-order address.

get_byte

(r0h = r0h - 1) == 0

No (stop condition or slave address detected)

Yes

mov.b rx_data,address+1 ; Set middle-order address.

get_byte

(r0h = r0h - 1) == 0

No (stop condition or slave address detected)

Yes

mov.b rx_data,address+2 ; Set high-order address.

mov.b #00h,address+3

restart_wait

(r0h = r0h - 1) == 0

No (stop condition detected or R/W direction bit error occurred)

Yes

mov.w address,a0

mov.w address+2,a1

lde.b [a1a0],tx_data ; Set read data as transmit data.

put_byte

add.w #1,a0 ; Change read address (+1)

mov.w a0,r1 ; Page read completed?

No

#00h == r1

Yes (page read completed)

(r0h = r0h - 1) == 0

No (stop condition detected or NACK detected)

Yes

Stop condition receive handling

stop_wait

command_handler

command_error
(8) Page program

page_program

mov.b #00h,address ; Set low-order address.

get_byte

( (r0h = r0h - 1) == 0 )

Yes

mov.b rx_data,address+1 ; Set middle-order address.

get_byte

( (r0h = r0h - 1) == 0 )

Yes

mov.b rx_data,address+2 ; Set high-order address.

mov.b #00h,address+3

mov.w #0,a0

mov.b rx_data,page_buffer[a0] ; Store program data to buffer.

add.w #1,a0 ; Change storage address (+1)

No

#0100h == a0 ; Page program data receive completed?

Yes (reception completed)

Stop condition receive handling

stop_wait

#0010h > address+1 ; Full status check performed, and ready/busy determined

No

#0100h <= address+1 ; Page program data receive completed?

Yes (other than block 0)

No

#0100h <= address+1 ; Page program data receive completed?

Yes (other than block 0)

No

mov.b fst,r0l

and.b #10110000h,r0l

No

#080h == r0l ; Full status check performed, and ready/busy determined

Yes (ready and no error occurred)

mov.w #FLASH_PROGRAM,a0

RAM execute handling

ram_run

command_handler

command_error
(9) Block erase

```
block_erase

mov.b #0feh,address ; Set low-order address.
;
Byte receive handling
get_byte

(r0h = r0h - 1) == 0

Yes

mov.b rx_data,address+1 ; Set middle-order address.

Yes

Byte receive handling
get_byte

(r0h = r0h - 1) == 0

Yes

mov.b rx_data,address+2 ; Set high-order address.

Yes

mov.b #00h,address+3

;
Byte receive handling
get_byte

(r0h = r0h - 1) == 0

Yes

#0dh = rx_data

Yes

Stop condition receive handling
stop_wait

#00f0h > address+1

Yes (other than block 0)

No

#0100h <= address+1

Yes (other than block 0)

mov.w #FLASH_BLOCK_ERASE,a0

RAM execute handling
ram_run

command_handler

command_error
```
(10) Read status register

```
read_status

Restart condition receive handling
restart_wait

(r0h = r0h - 1) == 0

Yes

mov.b fst.r0l
and.b #10110000b,r0l
mov.b r0l,srd

Watchdog timer reset
wdt_reset

mov.b srd,bx_data

Byte transmit handling
put_byte

(r0h = r0h - 1) == 0

Yes

mov.b srd1,bx_data

Byte transmit handling
put_byte

Stop condition receive handling
stop_wait

command_handler
```

No (stop condition or RW direction bit error occurred)

(11) Clear status register

```
clear_status

Stop condition receive handling
stop_wait

mov.w #FLASH_CLEAR_STATUS,a0

RAM execute handling
ram_run

command_handler
```
(12) ID check

id_check

| mov.w  #0,a0 |

Byte receive handling

get_byte

| (r0h = r0h - 1) == 0 |

| mov.b  rx_data,temp[a0] |

| add.w  #1,a0 |

| #7 == a0 |

Stop condition receive handling

stop_wait

| reset_blank == 1 |

| mov.w  #7,r2 |

Stop condition receive handling

stop_wait

| bset  srd10 |

| bclr  srd11 |

| mov.w  #7,r2 |

| mov.w  r2,a0 |

| shl.w  #2,a0 |

| lde.w  ID_code_addr-4[a0],address+2 |

| mov.w  address,a0 |

| mov.w  address+2,a1 |

| lde.b  [a1a0],r11 |

| mov.w  r2,a0 |

| temp-1[a0] == r11 |

| (r2 = r2 - 1) == 0 |

| bset  srd11 |

command_handler

ID_code_addr:

| .word  0efdfh ; ID1 |
| .word  0efe3h ; ID2 |
| .word  0efebh ; ID3 |
| .word  0eff3h ; ID4 |
| .word  0eff7h ; ID5 |
| .word  0effbh ; ID6 |

command_error

| ; Store received ID stored to the temporary buffer. |

| ; Change number of received data (+1). |

| ; 7-byte receive completed? |

| ; Set user ID address. |

| ; Set to no ID code matched. |

| ; Set to ID identified. |

| ; Read user ID. |

| ; No (stop condition or slave address detected) |

| ; No (user reset blank) |

| ; No (no user ID matched) |

| ; 7-byte confirmation completed? |

| ; Yes (confirmation completed) |

| ; Yes (reception completed) |

| ; Yes (user reset blank) |

| ; No (stop condition or slave address detected) |

| ; No (user reset blank) |

| ; No (no user ID matched) |

| ; Yes (confirmation completed) |

| ; Yes (reception completed) |

| ; Yes (user reset blank) |

| ; No (stop condition or slave address detected) |

| ; No (user reset blank) |

| ; No (no user ID matched) |

| ; Yes (confirmation completed) |

| ; Yes (reception completed) |

| ; Yes (user reset blank) |

| ; No (stop condition or slave address detected) |

| ; No (user reset blank) |

| ; No (no user ID matched) |

| ; Yes (confirmation completed) |

| ; Yes (reception completed) |

| ; Yes (user reset blank) |

| ; No (stop condition or slave address detected) |

| ; No (user reset blank) |

| ; No (no user ID matched) |

| ; Yes (confirmation completed) |

| ; Yes (reception completed) |

| ; Yes (user reset blank) |
(13) Version output function

version_output

Restart condition receive handling
restart_wait

( r0h = r0h - 1 ) == 0

Yes

lde.b version_string+1,tx_data

: Set program downloader version data (high order) as transmit data.

Byte transmit handling
put_byte

( r0h = r0h - 1 ) == 0

Yes

lde.b version_string.tx_data

: Set program downloader version data (low order) as transmit data.

Byte transmit handling
put_byte

( r0h = r0h - 1 ) == 0

Yes

lde.b usr_ver_h.tx_data

: Set user version data (high order) as transmit data.

Byte transmit handling
put_byte

( r0h = r0h - 1 ) == 0

Yes

lde.b usr_ver_l.tx_data

: Set user version data (low order) as transmit data.

Byte transmit handling
put_byte

Stop condition receive handling
stop_wait

command_handler

command_error

Yes

No (stop condition or R/W direction bit error occurred)

No (stop condition detected or R/W direction bit error occurred)

Yes

No (stop condition or NACK detected)

No (stop condition or NACK detected)

Yes

No (stop condition or NACK detected)

No (stop condition or NACK detected)
(14) RAM execute routine

```asm
ram_run

lde.w (ram_function_table+4)[a0],r3 ; Set number of transfer data.
lde.b (ram_function_table+2)[a0],r1h ; Set transfer source address (high order).
lde.w (ram_function_table+0)[a0],a0 ; Set transfer source addresses (middle order and low order).
mov.w #ram_execute,a1 ; Set transfer destination address.
smovf.b

CPU clock divided-by-8 setting handling

cpu_slow

RAM execute handling

ram_execute

CPU clock divided-by-1 setting handling

cpu_fast

rts
```

(15) Clear status register to flash memory (execute in RAM)

```asm
flash_clear_status

bclr fmr01 ; Enable CPU rewrite mode.
bset fmr01
and.b #00000010b,fmr0

COMMAND_WRITE FLASH_ADDRESS, CLEAR_STATUS ; Issue clear status register command.
COMMAND_WRITE FLASH_ADDRESS, PAGE_READ ; Issue read array command.
mov.b #00h,fmr0 ; Initialize flash memory control register 0.

rts

flash_clear_status_end
```
Page program to flash memory (execute in RAM)

```assembly
flash_program

; Enable CPU rewrite mode.
and.b #00000010b,fmr0
or.b #11110000b,fmr1
and.b #00001111b,fmr1

; Enable rewriting blocks A, B, C, and D.
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #00h,fmr0

; Initialize flash memory control register 0.

; Enable rewriting blocks A, B, C, and D.
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #00h,fmr0

; Set program address to A1A0.

; Issue program command.
COMMAND_WRITE [a1a0], 040h
ste.b r0l,[a1a0]

; Write program data.

; Set program data.

; Set program addresses (middle order and low order) to A0.

; Set program data offset to A0.

; Set program data offset + 1.
add.w #1,r3

; Set program address + 1.
add.w #1,r2

#100h == r3

; Page program completed?

; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.

; Initialize flash memory control register 0.

rts

flash_program_end
```

When WDT_USE is enabled

- `mov.b #000h,wdtr` - Reset watchdog timer.
- `mov.b #0ffh,wdtr` - Reset watchdog timer.

Yes (program error occurred)

- `add.w #1,r3`
- `add.w #1,r2`

Yes (page program completed)

- `mov.b #11110000b,fmr1`
- `mov.b #00h,fmr0`

No (busy)

Yes (ready)
(17) Block erase to flash memory (execute in RAM)

```assembly
flash_block_erase

bclr fmr01
bset fmr01
and.b #00000010b,fmr0
bclr fmr13
bset fmr13
or.b #11110000b,fmr1
and.b #00001111b,fmr1
mov.w address,a0
mov.w address+2,a1

; Enable CPU rewrite mode.

; Disable lock bit.

; Enable rewriting blocks A, B, C, and D.

; Set block address to A1A0.

COMMAND_WRITE [a1a0], BLOCK_ERASE

mov.b #0d0h,r1l
ste.b r1l,[a1a0]

; Write confirmation command.

; Disable lock bit.

; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.

; Initialize flash memory control register 0.

; Wait for ready.

; Reset watchdog timer.

; Reset watchdog timer.

; When WDT_USE is enabled

COMMAND_WRITE.MACRO address,data

mov.b #000h,wdtr
mov.b #0ffh,wdtr

; Reset watchdog timer.

; When WDT_USE is enabled

mov.b #data,r1l
ste.b r1l,address ; Write command.

.ENDM

(18) Command write macro
```
(19) System interrupt

**sys_undefined**

```
jmpi.a 0efdch
```

User undefined instruction interrupt

**sys_overflow**

```
jmpi.a 0efe0h
```

User overflow interrupt

**sys_brk**

```
jmpi.a 0efe4h
```

User BRK instruction interrupt

**sys_addmatch**

```
jmpi.a 0efe8h
```

User address match interrupt

**sys_wdt**

```
jmpi.a 0eff0h
```

User watchdog timer interrupt
4. Downloader Communication Protocol

4.1 Commands

4.1.1 Control Command List

Control commands are listed below.

<table>
<thead>
<tr>
<th>Control commands</th>
<th>1 Byte</th>
<th>2 Bytes</th>
<th>3 Bytes</th>
<th>4 Bytes</th>
<th>5 Bytes</th>
<th>6 Bytes</th>
<th>7 Bytes or More</th>
<th>ID Unchecked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page read</td>
<td>FFH</td>
<td>Middle-order address</td>
<td>High-order address</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Up to data</td>
<td>Not acceptable</td>
</tr>
<tr>
<td>Page program</td>
<td>41H</td>
<td>Middle-order address</td>
<td>High-order address</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Up to data</td>
<td>Not acceptable</td>
</tr>
<tr>
<td>Block erase</td>
<td>20H</td>
<td>Middle-order address</td>
<td>High-order address</td>
<td>D0H</td>
<td></td>
<td></td>
<td></td>
<td>Not acceptable</td>
</tr>
<tr>
<td>Read status register</td>
<td>70H</td>
<td></td>
<td>SRD</td>
<td>SRD1</td>
<td></td>
<td></td>
<td></td>
<td>Acceptable</td>
</tr>
<tr>
<td>Clear status register</td>
<td>50H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not acceptable</td>
</tr>
<tr>
<td>ID check function</td>
<td>F5H</td>
<td>ID1</td>
<td>ID2</td>
<td>ID3</td>
<td>ID4</td>
<td>ID5</td>
<td>Up to ID7</td>
<td>Acceptable</td>
</tr>
<tr>
<td>Version information output function</td>
<td>FBH</td>
<td>Program downloader version</td>
<td></td>
<td>User version</td>
<td></td>
<td></td>
<td></td>
<td>Acceptable</td>
</tr>
</tbody>
</table>

SRD: Status register data
SRD1: Status register data 1

Notes:
1. The shadowed areas show a transfer from the MCU (program downloader) to a programmer, the rest show a transfer from a programmer to the MCU (program downloader).
2. User program area blank product IDs are identified and all commands can be accepted.
3. The number of receive data is not checked and the timeout error is not processed in the downloader. When transmitting a command, make sure there is no excess or shortage of data.
4.2 Page Read

4.2.1 Operation

The page read command reads the specified user ROM area in the flash memory in units of 256 bytes. Specify the area to be read by the high-order addresses (A16 to A23) and middle-order addresses (A8 to A15). The target bytes are the 256 bytes from addresses xxxx00h to xxxxFFh.

4.2.2 Packet

<table>
<thead>
<tr>
<th></th>
<th>1st byte</th>
<th>2nd byte</th>
<th>3rd byte</th>
<th>4th byte</th>
<th>Up to 259th byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Up to data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Programmer to MCU     | FFh      | Middle-order address | High-order address |          |                  |
| MCU to Programmer     |          |                      |                     | Data 0   | Up to Data 255   |

Data 0: Low-order address is 00h
Data 255: Low-order address is FFh

4.2.3 Procedure

1. The slave address and R/W direction bit are received after receiving the start condition.
2. Page read command FFh is received at the first byte.
3. The middle-order address is received at the second byte and the high-order address is received at the third byte.
4. The slave address and R/W direction bit are received after receiving the restart condition.
5. The content in low-order address 00h is sequentially transmitted from the fourth byte.
6. The start condition is received after transmitting the last data.

When NACK is detected after transmitting the last data, the SCL line is released and the MCU waits for the stop condition. When ACK is detected, the dummy data (FFh) continues being transmitted until NACK is detected.

Figure 4.1 Page Read
4.3 Page Program

4.3.1 Operation

The page program command programs the data to the specified user ROM area in the flash memory in units of 256 bytes. Specify the area to be programmed by the high-order addresses (A16 to A23) and middle-order addresses (A8 to A15). The target bytes are the 256 bytes from addresses xxxx00h to xxxxFFh.

4.3.2 Packet

<table>
<thead>
<tr>
<th>1st byte</th>
<th>2nd byte</th>
<th>3rd byte</th>
<th>4th byte</th>
<th>Up to 259th byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Address</td>
<td>Data</td>
<td>Up to</td>
<td></td>
</tr>
<tr>
<td>Programmer to MCU</td>
<td>41h</td>
<td>Middle-order address</td>
<td>High-order address</td>
<td>Data 0</td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data 0: Low-order address is 00h
Data 255: Low-order address is FFh

4.3.3 Procedure

(1) The slave address and R/W direction bit are received after receiving the start condition.
(2) Page program command 41h is received at the first byte.
(3) The middle-order address is received at the second byte and the high-order address is received at the third byte.
(4) The programming data to low-order address 00h is received from the fourth byte.
(5) The stop condition is received after receiving the last data.

When the stop condition is detected after receiving 256-byte programming data, programming starts. When the programming data is less than 256 bytes, transmit FFh for the shortage. When the programming data is less than 256 bytes and the stop condition is detected, the received commands are presumed to be command errors and programming is not performed. When the programming data is more than 257 bytes, the MCU waits for the stop condition while write data from the 257th byte is being discarded. When the stop condition is detected, a write operation of the 256 bytes already received starts. If an error occurs during programming, the SR4 bit becomes 1 (program status ends in error).

After executing this command, confirm the status of the flash memory with the read status register command.

Figure 4.2 Page Program

The shadowed areas show a transmission from the MCU to a programmer, the rest show a transmission from a programmer to the MCU.

ST: Start condition
SP: Stop condition
W: R/W direction bit, write = 0 (programmer to MCU)
4.4 Block Erase

4.4.1 Operation

The block erase command erases a specified block area in the user ROM area of the flash memory. Specify a block area by the eight high-order bits (A16 to A23) and eight middle-order bits (A8 to A15) at a given address of the block to be erased.

4.4.2 Packet

<table>
<thead>
<tr>
<th>Command</th>
<th>1st byte</th>
<th>2nd byte</th>
<th>3rd byte</th>
<th>4th byte</th>
<th>Up to 259th byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer to MCU</td>
<td>20h</td>
<td>Middle-order address</td>
<td>High-order address</td>
<td>D0h</td>
<td></td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.4.3 Procedure

(1) The slave address and R/W direction bit are received after receiving the start condition.
(2) Block erase command 20h is received at the first byte.
(3) The middle-order address is received at the second byte and the high-order address is received at the third byte.
(4) Confirmation command D0h is received at the fourth byte.
(5) The stop condition is received.

When the stop condition is detected after receiving confirmation command D0h, erasing to the specified block starts. The erase operation sets the contents of the flash memory to FFh. If an error occurs, the SR5 bit becomes 1 (erase status ends in error).

After executing this command, confirm the status of the flash memory with the read status register command.

![Figure 4.3 Block Erase](image-url)

The shadowed areas show a transmission from the MCU to a programmer, the rest show a transmission from a programmer to the MCU.

ST: Start condition
SP: Stop condition
W: R/W direction bit, write = 0 (programmer to MCU)
4.5 Read Status Register

4.5.1 Operation

The read status register command confirms the operating status of the flash memory.

4.5.2 Packet

<table>
<thead>
<tr>
<th></th>
<th>1st byte</th>
<th>2nd byte</th>
<th>3rd byte</th>
<th>4th byte</th>
<th>Up to 259th byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>SRD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer to MCU</td>
<td>70h</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td>SRD output</td>
<td>SRD1 output</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SRD: Status register data
SRD1: Status register data 1

4.5.3 Procedure

1. The slave address and R/W direction bit are received after receiving the start condition.
2. Read status register command 70h is received at the first byte.
3. The slave address and R/W direction bit are received after receiving the restart condition.
4. SRD is transmitted at the second byte.
5. SRD1 is transmitted at the third byte.
6. The stop condition is received.

When NACK is detected after transmitting SRD1, the SCL line is released and the MCU waits for the stop condition. When ACK is detected, the dummy data (FFh) continues being transmitted until NACK is detected.

- ST: Start condition
- RT: Restart condition
- SP: Stop condition
- W: R/W direction bit, write = 0 (programmer to MCU)
- R: R/W direction bit, read = 1 (MCU to programmer)

The shadowed areas show a transmission from the MCU to a programmer, the rest show a transmission from a programmer to the MCU.

Figure 4.4 Read Status Register
### 4.5.4 SRD Register

<table>
<thead>
<tr>
<th>Each Bit of SRD</th>
<th>Status Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR7 (bit 7)</td>
<td>Sequencer status</td>
<td>1 (Ready) / 0 (Busy)</td>
</tr>
<tr>
<td>SR6 (bit 6)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR5 (bit 5)</td>
<td>Erase status</td>
<td>1 (Error) / 0 (Completed normally)</td>
</tr>
<tr>
<td>SR4 (bit 4)</td>
<td>Program status</td>
<td>1 (Error) / 0 (Completed normally)</td>
</tr>
<tr>
<td>SR3 (bit 3)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR2 (bit 2)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR1 (bit 1)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR0 (bit 0)</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

1. **Sequencer status**
   The sequencer status shows the operating status of the flash memory. This bit becomes 0 (busy) during auto-programming or auto-erasing. This bit becomes 1 (ready) during auto-programming or auto-erasing.

2. **Erase status**
   The erase status shows the erase operating status. If an error occurs, this bit becomes 1. This bit is set to 0 when the clear status register command is executed.

3. **Program status**
   The program status shows the programming status. If an error occurs, this bit becomes 1. This bit is set to 0 when the clear status register command is executed.

Both bits SR5 and SR4 become 1 in the following cases:
- The defined command is not written correctly.
- Data other than values which can be written to the second bus cycle data of the block erase command (D0h or FFh) is written in the cycle to input the block erase confirmation command. When FFh is written, the MCU enters read array mode and the command is canceled.

4. **Reserved bit**
   When read, the content is undefined.
### 4.5.5 SRD1 Register

<table>
<thead>
<tr>
<th>Each Bit of SRD1</th>
<th>Status Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR15 (bit 7)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR14 (bit 6)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR13 (bit 5)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR12 (bit 4)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR11 (bit 3)</td>
<td>ID check</td>
<td></td>
</tr>
<tr>
<td>SR10 (bit 2)</td>
<td>ID check</td>
<td>00: Not checked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01: ID Not matched</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11: Checked</td>
</tr>
<tr>
<td>SR9 (bit 1)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>SR8 (bit 0)</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

1. **ID check**
   - These bits indicate the ID check results.
2. **Reserved bit**
   - When read, the content is undefined.
4.6  Clear Status Register

4.6.1  Operation

The clear status register command initializes a status register. Initialize the status register before executing the erase or the page program to the flash memory.

4.6.2  Packet

<table>
<thead>
<tr>
<th></th>
<th>1st byte</th>
<th>2nd byte</th>
<th>3rd byte</th>
<th>4th byte</th>
<th>Up to 259th byte</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer to MCU</td>
<td>50h</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.6.3  Procedure

1. The slave address and R/W direction bit are received after receiving the start condition.
2. The clear status register command 50h is received at the first byte.
3. The stop condition is received.

When the stop condition is detected after receiving the clear status register command 50h, the status register is initialized.

---

**Figure 4.5  Clear Status Register**

- ST: Start condition
- SP: Stop condition
- W: R/W direction bit, write = 0 (programmer to MCU)
4.7 ID Check Function

4.7.1 Operation

This function compares the ID received from the programmer and the user ID code stored in the virtual fixed vector address. The ID check results are stored in SR11 to SR10 in the SRD1 register.

4.7.2 Packet

<table>
<thead>
<tr>
<th>1st Byte</th>
<th>2nd Byte</th>
<th>3rd Byte</th>
<th>4th Byte</th>
<th>Up to 8th Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>ID</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Programmer to MCU: F5h, ID1, ID2, ID3, Up to ID7

MCU to Programmer: 

4.7.3 Procedure

1. The slave address and R/W direction bit are received after receiving the start condition.
2. ID check function command F5h is received at the first byte.
3. ID1 to ID7 are received from the second byte to the eighth byte, respectively.
4. The stop condition is received after receiving ID7.

When the stop condition is detected after receiving ID7, the ID check starts. However, a user program area blank product returns to waiting for the start condition state without performing ID check. When ID1 to ID7 all match, bits SR11 to SR10 become 11b (verified). If any of the IDs do not match, bits SR11 to SR10 become 01b (verify not matched).

The shadowed areas show a transmission from the MCU to a programmer, the rest show a transmission from a programmer to the MCU.

Figure 4.6   ID Check Function
4.8 Version Information Output Function

4.8.1 Operation

This function transmits version information of the program downloader and user program.

4.8.2 Packet

<table>
<thead>
<tr>
<th></th>
<th>1st Byte</th>
<th>2nd Byte</th>
<th>3rd Byte</th>
<th>4th Byte</th>
<th>5th Byte</th>
<th>Up to 259th Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer to MCU</td>
<td>F8h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td>Program downloader</td>
<td>User</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.8.3 Procedure

1. The slave address and R/W direction bit are received after receiving the start condition.
2. The version information output function command F8h is received at the first byte.
3. The slave address and R/W direction bit are received after receiving the restart condition.
4. The program downloader version is transmitted at the high-order second byte first and then the low-order third byte.
5. The user program version is transmitted at the high-order fourth byte first and then low-order fifth byte.
6. The stop condition is received.

When NACK is detected after transmitting the user program version, the SCL line is released and the MCU waits for the stop condition. When ACK is detected, the dummy data (FFh) continues being transmitted until NACK is detected.

The shadowed areas show a transmission from the MCU to a programmer, the rest show a transmission from a programmer to the MCU.

Figure 4.7 Version Information Output Function

ST: Start condition
RT: Restart condition
SP: Stop condition
W: R/W direction bit, write = 0 (programmer to MCU)
R: R/W direction bit, read = 1 (MCU to programmer)
4.8.4 Version Data

For the example shown below, the program download version is transmitted after 01h is set to the high order and 00h to the low order of the program downloader version, and 00h is set to the high order and 10h to the low order of the user version.

When the program downloader version is Ver.1.00 and the user version is Ver.0.10:

Program downloader version data
(in the bt_r835a.a30 file)

.org version_string
.word 0100h ; Program Downloader version (Ver.1.00)

User version data
(in the sect30.inc file for 6. User Program Example)

User_Ver .equ 0010h ; User version (Ver.0.10)
5. Error Handling

5.1 Command Error
When commands other than control commands shown in 4.1.1 Control Command List are received, the received commands are presumed to be command errors and a dummy read of the I²C receive buffer or transmission of dummy data (FFh) repeats until the stop condition is detected. When the stop condition is detected, the MCU returns to waiting for the start condition state.

5.2 Stop Condition Detection Error
When the stop condition is detected in a communication, the command in this communication is disregarded and the MCU returns to waiting for the start condition state.

5.3 Data Excess or Shortage
When the received data exceeds the number of bytes selected by the command, the MCU waits for the stop condition while reading and discarding the excess data. When the stop condition is detected, command handling is performed using the data initially received. When the data received is less than the selected number of bytes, the command is disregarded and command handling is not performed.
When ACK is detected and data continues being transmitted after transmitting the number of bytes data selected by the command, dummy data (FFh) continues being transmitted until NACK is detected. When NACK is detected, the SCL line is released and the MCU waits for the stop condition. When the stop condition is detected, the MCU returns to waiting for the start condition state.
6. User Program Example

The program downloader rewrites the user programs other than the user program in block 0 according to the programmer. An example of the user program is shown below.

6.1 Function

The LEDs connected to I/O ports P3_1, P3_3, P3_4, and P3_6 light.

6.2 Memory Map

Figure 6.1 shows a User Program Memory Map.

---

**Figure 6.1 User Program Memory Map**

---

Notes:
1. ID code checked by the program downloader whether IDs match the IDs transmitted from the programmer.
2. Do not use these interrupts. These are for use with development tools only.
3. Contents in red need to be changed in the automatic generating file. Refer to (3) Edit automatic generating file in 6.3 Initial Setting for more details.
6.3 Initial Settings

(1) Vector table
   Allocate the virtual fixed vector table to block 1 to use an interrupt by the user program.

(2) ID code
   Set an ID code in the virtual fixed vector table. Do not opt to generate an ID code file when compiling.

(3) Edit automatic generating file
   When the project type is made in the Application and the initial setting file is automatically generated by the High-performance Embedded Workshop (HEW), change the sect30.inc file and nc_define.inc file as follows (see Figure 6.1):
   • Change the allocation address of the locatable table to 0EEDCh, and the virtual fixed vector table to 0EFDCh.
   • Set an additional ID code to the virtual fixed vector table.
   • Add the symbol definition of the user version data and user version data setting to the virtual fixed vector table.
   • Comment out the assembler expansion function direction instructions “.ID” (set an ID code) and “.OFSREG” (set a value to the OFS register).
7. Programmer Example

7.1 Control Pins

(1) Pins SCL and SDA
These pins are for transmitting and receiving in I²C-bus.

(2) P3_3 pin
This pin is used to select the program downloader or the user program.

(3) RESET pin
This pin controls MCU resets from the programmer.

(4) Pins VCC and VSS
Adjust high level from the programmer to the MCU’s VCC level and low level from the programmer to the MCU’s VSS level, respectively.

Figure 7.1 Programmer Configuration

7.2 Programmer Communication Protocol

• I²C-bus master mode is used to communicate with the program downloader.
• When the restart condition is transmitted for the read command, the R/W direction bit is switched as follows:
  • Set the R/W direction bit to 0 (programmer to MCU) when the slave address is transmitted after transmitting the start condition.
  • Set the R/W direction bit to 1 (MCU to programmer) when the slave address is transmitted after transmitting the restart condition.
7.3 Programmer Functions

The following are the functions necessary for the programmer:

- Blank Check
- Erase
- Program
- Verify
- Read

7.4 Blank Check

Data (program) in the specified area automatically or manually is read from the MCU’s on-chip flash memory. The programmer confirms that all read data is blank (FFh).

![Blank Check Diagram]

- When received data is
  - All FFh
  - Go to next step
  - Other than above
  - Process stopped by blank check error
- When the blank check for the specified area is not completed
  - Return to page read process
  - When blank check is completed
  - Blank check OK

- When the blank check is completed
  - Blank check OK
7.5 Erase

Data (program) in the MCU’s specified on-chip flash memory blocks automatically or manually is erased.

Bits SR11 and SR10 in SRD1
- 00b (not verified), 01b (verification not matched)
  → Go to ID check
- 11b (verified)
  → Go to erase without ID check

Bits SR11 and SR10 in SRD1
- 01b (verification not matched)
  → Process stopped by ID check error
- 11b (verified)
  → Go to next step

SR5 bit in SRD1
- 1 (Error)
  → Process stopped by erase error
- 0 (Completed normally)
  → Go to next step

• When erase for the specified blocks is not completed
  → Return to clear status process
• When erase for the specified blocks is completed
  → Erase OK

Programmer

Transmit read status register command 70h
Receive SRD and SRD1

Start ID check

Transmit ID check commands F5h and ID1 to ID7
Receive SRD and SRD1

Transmit read status register command 70h
Receive SRD and SRD1

ID Check OK

Start Erase

Transmit clear status register command 50h
Transmit block erase command 20h, middle-order address, high-order address, and D0h
Transmit read status register command 70h
Receive SRD and SRD1

Erase OK
7.6 Program

Data (program) in the MCU’s specified on-chip flash memory area automatically or manually is programmed.

![Diagram of program flow]

- **Bits SR11 and SR10 in SRD1**
  - 00b (not verified), 01b (verification not matched) → Go to ID check
  - 11b (verified) → Go to program without ID check

- **SR4 bit in SRD**
  - 1 (Error) → Process stopped by program error
  - 0 (Completed normally) → Go to next step

- When program for the specified area is not completed → Return to clear status process
- When erase for the specified area is completed → Program OK

**Program OK**
7.7 Verify

Data (program) in the specified area automatically or manually is read from the MCU’s on-chip flash memory. The programmer compares the read data with the memory data (program) in the programmer to confirm that they match.

**Bits SR11 and SR10 in SRD1**
- 00b (not verified), 01b (verification not matched) → Go to ID check
- 11b (verified) → Go to verify check without ID check

**Bits SR11 and SR10 in SRD1**
- 01b (verification not matched) → Process stopped by ID check error
- 11b (verified) → Go to next step

- When received data and programmed data do not match → Process stopped by verify check error
- Other than the above → Go to next step

- When verify check for the specified area is not completed → Return to page read process
- When verify check for the specified area is completed → Verify check OK
7.8 Read

This function allows reading the data (program) in the automatically or manually specified area from the MCU with on-chip flash memory.

The programmer stores the read data in its internal memory.

![Diagram of the read process]

- Bits SR11 and SR10 in SRD1
  - 00b (not verified), 01b (verification not matched) → Go to ID check
  - 11b (verified) → Go to read without ID check

- Bits SR11 and SR10 in SRD1
  - 01b (verification not matched) → Process stopped by ID check error
  - 11b (verified) → Go to next step

- When read for the specified area is not completed → Return to page read process
- When read for the specified area is completed → Read OK

- Transmit read status register command 70h
- Receive SRD and SRD1

- Start ID check
- Transmit ID check commands F5h and ID1 to ID7
- Receive SRD and SRD1
- Transmit read status register command 70h
- Receive SRD and SRD1

- ID Check OK

- Start read
- Transmit page read command FFh, middle-order address, and high-order address
- Receive data (256 bytes)

- Store received data in programmer internal memory

- Read OK
8. **Sample Programming Code**

A sample program can be downloaded from the Renesas Technology website.  
To download, click “Application Notes” in the left-hand side menu on the R8C/35A Group page.

9. **Reference Documents**

**Hardware Manual**
R8C/35A Group Hardware Manual    Rev.0.40
The latest version can be downloaded from the Renesas Technology website.

**Technical News/Technical Update**
The latest information can be downloaded from the Renesas Technology website.
# Website and Support

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

Inquiries  
http://www.renesas.com/inquiry  
csc@renesas.com

## REVISION HISTORY

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Dec 29, 2009</td>
<td>– First Edition issued</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.

2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.

3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.

4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)

5. Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.

6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guarantees regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.

7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or underwater communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.

8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implants
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.

9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.

10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.

12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.

13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.

© 2009. Renesas Technology Corp., All rights reserved. Printed in Japan