1. Abstract

This document describes the UART version program downloader for the R8C/35C Group.

2. Introduction

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

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

This application note can be used with other R8C Family MCUs which have the same special function registers (SFRs) as the above group. Check the manual for any modifications to functions. Careful evaluation is recommended before using the program described in this application note.
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 P1_5 and selects either to use the program downloader or the user program. The program downloader operates when port P1_5 is high, and the user program operates when port P1_5 is low.
- The virtual fixed vector table is allocated to block 1 to use the fixed vector table interrupt in the user program.
- UART0 clock asynchronous serial I/O (UART) is used to communicate with a programmer.
- CMOS output is selected for the TXD0 pin.
- The communication format is as follows:
  - Bit rate: approximately 9600 bps
  - Transfer data length: 8 bits
  - Stop bit: 1 bit
  - Parity bit: None
- 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).
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.
3.2 Timing after Reset

The operating program after reset chooses either the program downloader or the user program. The MCU enters either program according to the P1_5/RXD0 pin level applied to the MCU during (1). Before reset is deasserted, a programmer must determine the input level of the P1_5/RXD0 pin, and hold that level during (1).

Figure 3.5 Signal Control Timing after Reset

(1) While the program downloader or user program selection signal is held: Approximately 3.5 ms. \(^{(1)}\)

The reset sequence is included in above time.

Note:
1. Time when the low-speed on-chip oscillator (60 kHz (min)) operates.
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_r835c.inc file.
## 3.4 Registers

<table>
<thead>
<tr>
<th>Port P1 Register</th>
<th>P1 [address 00E1h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7</td>
<td>b0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| Port P1_4 bit | 1: High level |

<table>
<thead>
<tr>
<th>Port P1 Direction Register</th>
<th>PD1 [address 00E3h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7</td>
<td>b0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| Port PD1_4 bit | 1: Output mode (functions as TXD0 pin) |
| Port PD1_5 bit | 0: Input mode (functions as RXD0 pin) |

<table>
<thead>
<tr>
<th>Watchdog Timer Reset Register</th>
<th>WDTR [address 000Dh]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7</td>
<td>b0</td>
</tr>
<tr>
<td>0/1</td>
<td>0/1</td>
</tr>
</tbody>
</table>

Writing 00h and then FFh to this register initializes the watchdog timer.

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

<table>
<thead>
<tr>
<th>UART0 Pin Select Register</th>
<th>U0SR [address 0188h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7</td>
<td>b0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| TXD0 pin select bit | 1: P1_4 assigned |
| RXD0 pin select bit | 1: P1_5 assigned |

<table>
<thead>
<tr>
<th>Protect Register</th>
<th>PRCR [address 000Ah]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7</td>
<td>b0</td>
</tr>
<tr>
<td></td>
<td>0/1</td>
</tr>
</tbody>
</table>

Enables writing to registers CM0, CM1, CM3, OCD, FRA0, FRA1, FRA2, and FRA3.
0: Write disabled
1: Write enabled
### System Clock Control Register 0 (CM0, address 0006h)

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0/1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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 (CM1, address 0007h)

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></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**
  - 0 0: No division mode

### Oscillation Stop Detection Register (OCD, address 000Ch)

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

- **System clock select bit**
  - 0: XIN clock selected

### Interrupt Control Register (S0TIC, address 0051h)

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

### Interrupt Control Register (S0RIC, address 0052h)

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Interrupt priority level select bit**
  - 0 0 0: Level 0 (interrupt disabled)
UART0 Transmit/Receive Mode Register  
U0MR [address 00A0h]

- **b7 b0**
  - 00000101
  - **Serial I/O mode select bit**
  - 101: UART mode, transfer data 8 bits long
  - **Internal/external clock select bit**
  - 0: Internal clock
  - **Stop bit length select bit**
  - 0: 1 stop bit
  - **Odd/even parity select bit**
  - This bit is not used. Set this bit to 0.
  - **Parity enable bit**
  - 0: Parity disabled

UART0 Transmit/Receive Control Register 0  
U0C0 [address 00A4h]

- **b7 b0**
  - 00000000
  - **BRG count source select bit**
  - 0 0: f1 selected
  - **Data output select bit**
  - 0: TXD0 pin set to CMOS output
  - **Transfer format select bit**
  - 0: LSB first

UART0 Transmit/Receive Control Register 1  
U0C1 [address 00A5h]

- **b7 b0**
  - 00001100
  - **Transmit enable bit**
  - 0: Transmission disabled
  - 1: Transmission enabled
  - **Receive enable bit**
  - 0: Reception disabled
  - 1: Reception enabled
  - **Receive complete flag**
  - 0: No data in the U0RB register
  - 1: Data present in the U0RB register
  - **UART0 transmit interrupt source select bit**
  - 1: Transmission completed (TXEPT = 1)
  - **Set this bit to 0 in UART mode.**
UART0 Bit Rate Register  U0BRG [address 00A1h]

Bit rate setting
Set to 129-1.

UART0 Transmit Buffer Register  U0TB [addresses 00A3h to 00A2h]

Transmit data

UART0 Receive Buffer Register  U0RB [addresses 00A7h to 00A6h]

Receive data
Flash Memory Status Register  FST [address 01B2h]

- b7: Program error flag
  - 0: No program error
  - 1: Program error
- b6: Erase error/blank check error flag
  - 0: No erase error/blank check error
  - 1: Erase error/blank check error
- b5: Ready/busy status flag
  - 0: Busy
  - 1: Ready

Flash Memory Control Register 0  FMR0 [address 01B4h]

- b7: CPU rewrite mode select bit
  - 0: CPU rewrite mode disabled
  - 1: CPU rewrite mode enabled
- b6: EW1 mode select bit
  - 0: EW0 mode
- b5: Flash memory stop bit
  - 0: Flash memory operates.
- b4: Erase/write error interrupt enable bit
  - 0: Erase/write error interrupt disabled
- b3: Flash access error interrupt enable bit
  - 0: Flash access error interrupt disabled
- b2: Flash ready status interrupt enable bit
  - 0: Flash ready status interrupt disabled
## Flash Memory Control Register 1 - FMR1

[address 01B5h]

<table>
<thead>
<tr>
<th>7</th>
<th>0</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **b7 (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>
<thead>
<tr>
<th>Table 3.1</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assigned Memory</strong></td>
<td><strong>Size</strong></td>
</tr>
<tr>
<td>ROM</td>
<td>1226 bytes</td>
</tr>
<tr>
<td>RAM</td>
<td>427 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3.2</th>
<th>RAM and Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>Size</td>
</tr>
<tr>
<td>ram_execute</td>
<td>128 bytes</td>
</tr>
<tr>
<td>status_flags</td>
<td>1 byte</td>
</tr>
<tr>
<td>reset_blank</td>
<td>–</td>
</tr>
<tr>
<td>srd1</td>
<td>1 byte</td>
</tr>
<tr>
<td>srd08</td>
<td>–</td>
</tr>
<tr>
<td>srd09</td>
<td>–</td>
</tr>
<tr>
<td>srd10</td>
<td>–</td>
</tr>
<tr>
<td>srd11</td>
<td>–</td>
</tr>
<tr>
<td>srd12</td>
<td>–</td>
</tr>
<tr>
<td>srd13</td>
<td>–</td>
</tr>
<tr>
<td>srd14</td>
<td>–</td>
</tr>
<tr>
<td>srd15</td>
<td>–</td>
</tr>
<tr>
<td>srd</td>
<td>1 byte</td>
</tr>
<tr>
<td>address</td>
<td>4 bytes</td>
</tr>
<tr>
<td>temp</td>
<td>32 bytes</td>
</tr>
<tr>
<td>rx_data</td>
<td>2 bytes</td>
</tr>
<tr>
<td>tx_data</td>
<td>1 byte</td>
</tr>
<tr>
<td>page_buffer</td>
<td>256 bytes</td>
</tr>
</tbody>
</table>
### 3.6 Flowchart

(1) Startup processing

1. **reset**
   - `bclr pd_5` ; Set port 1_5 input mode.
2. **Program downloader selected?**
   - **Yes** (program downloader selected)
     - `ldc #boot_stack,SP`
     - `reset`
     - `ldc #SB_base,SB`
     - `ldintb #VECTOR_ADR`
     - `bset prc0`
     - `bset cm13`
     - `bclr cm05`
     - `bclr cm14` ; Select low-speed on-chip oscillator.
     - `mov.b #00000101b,u0sr` ; Assign TXD0 to port P1_4 and RXD0 to port P1_5.
     - `mov.b #00010000b,pd1` ; Set port P1_4/TXD0 output.
     - `mov.b #00010000b,p1` ; Initialize port P1.
     - `mov.w #0,a1` ; XIN clock on
     - `wait until oscillation stabilizes 2040 > a1`
     - `inc.w a1`
     - `bclr cm07` ; Select XIN clock.
     - `bclr ocd2` ; Select XIN clock as the system clock.
     - `and.b #00111111b,cm1` ; Select CPU clock no division.
     - `bclr cm06` ; Enable bits CM16 and CM17.
     - `bclr prc0` ; Set system clock control register protect.
     - `Watchdog timer reset` ; Disable system clock control register protect.
     - `wdt_reset` ; Select XIN-XOUT pin.
   - **No** (user program selected)
     - `ldc #00h,FLG` ; Initialize FLG.
     - `bclr cm14` ; Select XIN-Clock.
     - `mov.b #00000101b,u0sr` ; Assign TXD0 to port P1_4 and RXD0 to port P1_5.
     - `mov.b #00010000b,pd1` ; Set port P1_4/TXD0 output.
     - `mov.b #00010000b,p1` ; Initialize port P1.
     - `mov.w #0,a1` ; XIN clock on
     - `Watchdog timer reset` ; Disable system clock control register protect.
     - `wdt_reset` ; Select XIN-XOUT pin.
   - **Watchdog timer reset**
     - `wdt_reset`
     - `mov.b #00000101b,u0sr` ; Assign TXD0 to port P1_4 and RXD0 to port P1_5.
     - `mov.b #00010000b,pd1` ; Set port P1_4/TXD0 output.
     - `mov.b #00010000b,p1` ; Initialize port P1.
     - `mov.w #0,a1` ; XIN clock on
     - `Watchdog timer reset` ; Disable system clock control register protect.
     - `wdt_reset` ; Select XIN-XOUT pin.

![Flowchart Diagram](image-url)
mov.b  #00000000b,s0ric
mov.b  #00000000b,s0tic
bclr  te_u0c1
bclr  re_u0c1
mov.b  #0000101b,u0mr
mov.b  #0000100b,u0c0
mov.b  #0001001b,u0c1
mov.b  #129-1,u0brg
bset  te_u0c1
bset  re_u0c1
mov.w  #0,r0
mov.w  #(RAM_END+1-RAM_TOP)/2,r3
mov.w  #RAM_TOP,a1
sstr.w

mov.b  #00000101b,u0mr
mov.b  #00001000b,u0c0
mov.b  #0001001b,u0c1
mov.b  #00000000b,s0ric
mov.b  #00000000b,s0tic
bset  re_u0c1
bset  te_u0c1
mov.w  #0,r0
mov.w  #(RAM_END+1-RAM_TOP)/2,r3
mov.w  #RAM_TOP,a1
sstr.w

#0fffh == User_Reset_VEC

No

#0fffh == User_Reset_VEC+2

No

Yes

Yes

or.b  #00001100b,srd1
bset  reset_blank

command_handler
(2) Command handler

```assembly
; Command handler

; Byte receive processing
get_byte

mov.b rx_data,r0l

#READ_STATUS == r0l

; No

#VERSION_OUTPUT == r0l

; No

#ID_CHECK == r0l

; No

mov.b sr1,r0h

and.b #0ch,r0h

; No

#0ch == r0h

; Yes (ID identified)

#PAGE_READ == r0l

; No

#PAGE_PROGRAM == r0l

; No

#BLOCK_ERASE == r0l

; No

#CLEAR_STATUS == r0l

; No

; command_error

mov.b #0000000b,s0ric

mov.b #0000000b,s0tic

bclr te_u0c1

bclr re_u0c1

mov.b #0000101b,u0mr

mov.b #0001000b,u0c0

mov.b #0001001b,u0c1

mov.b #129-1,u0brg

bset te_u0c1

bset re_u0c1

; Initialize UART0.
```

Yes (ID identified)
### (3) Subroutine 1

**cpu_fast**

- `bset pro0`
- `and.b #00111111b,cm1`
- `bclr cm06`
- `bclr pro0`

**rts**

**cpu_slow**

- `bset pro0`
- `bset cm06`
- `bclr prc0`

**rts**

**wdt_reset**

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

**rts**

**get_byte**

- **Watchdog timer reset**
  - **wdt_reset**
  - `ri_u0c1 == 1`
  - Yes (data in U0RB)
    - `mov.w u0rb_rx_data`
  - No

**rts**

When WDT_USE is enabled

- **reset watchdog timer**
- **reset watchdog timer**

**Watchdog timer reset:

- **wdt_reset**
- **Watchdog timer reset**
- **Wait until receive operation is completed.**
- **Store received data.**

- **Yes (data in U0RB)**
- **No**
- **ri_u0c1 == 1**
- **mov.w u0rb_rx_data**

**rts**
(4) Subroutine 2

<table>
<thead>
<tr>
<th>put_byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov.b tx_data,u0tb</td>
</tr>
<tr>
<td>mov.b #00000000b,s0tic</td>
</tr>
<tr>
<td>Watchdog timer reset</td>
</tr>
<tr>
<td>wdt_reset</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>ir_s0tic == 1</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>rts</td>
</tr>
</tbody>
</table>

; Set transmit data.
; Clear transmit interrupt request bit.
; Wait until transmit operation is completed.

(5) Page read

<table>
<thead>
<tr>
<th>page_read</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov.b #00h,address</td>
</tr>
<tr>
<td>Byte receive processing</td>
</tr>
<tr>
<td>get_byte</td>
</tr>
<tr>
<td>mov.b rx_data,address+1</td>
</tr>
<tr>
<td>Byte receive processing</td>
</tr>
<tr>
<td>get_byte</td>
</tr>
<tr>
<td>mov.b rx_data,address+2</td>
</tr>
<tr>
<td>mov.b #00h,address+3</td>
</tr>
<tr>
<td>Watchdog timer reset</td>
</tr>
<tr>
<td>wdt_reset</td>
</tr>
<tr>
<td>mov.w address,a0</td>
</tr>
<tr>
<td>mov.w address+2,a1</td>
</tr>
<tr>
<td>lde.b [a1a0],tx_data</td>
</tr>
<tr>
<td>Byte transmit processing</td>
</tr>
<tr>
<td>put_byte</td>
</tr>
<tr>
<td>add.w #1,a0</td>
</tr>
<tr>
<td>mov.w a0,r0</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>#00h == r0 !</td>
</tr>
<tr>
<td>Yes (page read completed)</td>
</tr>
<tr>
<td>command_handler</td>
</tr>
</tbody>
</table>

; Set low-order address.
; Set middle-order address.
; Set high-order address.
; Set read data as transmit data.
; Change read address (+1).
; Page read completed?
(6) Page program

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

Byte receive processing
get_byte

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

Byte receive processing
get_byte

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).

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

mov.b #080h == r0l

mov.w FLASH_PROGRAM, a0

RAM execute processing
ram_run

command_handler
```

- Page program data receive completed?
- ; Set low-order address.
- ; Set middle-order address.
- ; Set high-order address.
- ; Store program data to buffer.
- ; Change storage address (+1).
(7) Block erase

```
block_erase

mov.b #0feh,address ; Set low-order address.

Byte receive processing
get_byte

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

Byte receive processing
get_byte

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

mov.b #00h,address+3

Byte receive processing
get_byte

mov.w #FLASH_BLOCK_ERASE,a0
RAM execute processing
ram_run

command_handler

#000h == rx_data ; Confirmation command OK?

#0d0h == rx_data ; Confirmation command OK?

#00f0h > address+1
No

#0100h <= address+1
Yes (other than block 0)

#000h > address+1
No

#0100h <= address+1
Yes (other than block 0)
```

(Confirmation command error)
(8) Read status register

```
read_status

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

Watchdog timer reset
wdt_reset

mov.b   srd,tx_data

Byte transmit processing
put_byte

mov.b   srd1,tx_data

Byte transmit processing
put_byte

command_handler
```

(9) Clear status register

```
clear_status

mov.w   #FLASH_CLEAR_STATUS,a0

RAM execute processing
ram_run

command_handler
```
(10) ID check

```
(id_check)

mov.w #0,a0

get_byte

mov.b rx_data,temp[a0] ; Store received ID to the temporary buffer.
add.w #1,a0 ; Change number of received data (+1).

#7 == a0

; 7-byte receive completed?

reset_blank == 1

Yes (reception completed)

No

Yes (user reset blank)

No

bset srd10

bclr srd11

mov.w #7,r2

temp-1[a0] == r1l

Yes

lde.w ID_code_addr-2[a0],address+2 ; 7-byte confirmation completed?

lde.w ID_code_addr-4[a0],address

mov.w r2,a0

shl.w #2,a0

mov.w address+2,a1

mov.w address,a0

lde.b [a1a0],r1l ; Read user ID.

mov.w r2,a0

(r2 = r2 - 1) == 0

No (no user ID matched)

Yes

ID_code_addr:

.lword 0efdfh ; ID1
.lword 0efe3h ; ID2
.lword 0efebh ; ID3
.lword 0efefh ; ID4
.lword 0eff3h ; ID5
.lword 0eff7h ; ID6
.lword 0effbh ; ID7
```
(11) Version output function

```
.version_output

lde.b version_string+1,bx_data

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

Byte transmit processing
put_byte

lde.b version_string,bx_data

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

Byte transmit processing
put_byte

lde.b usr_ver_h,bx_data

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

Byte transmit processing
put_byte

lde.b usr_ver_l,bx_data

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

Byte transmit processing
put_byte

command_handler
```

```
.section version_area,ROMDATA
.org version_string

.word 0100h
```
(12) RAM execute routine

```assembly
ram_run
  lde.w (ram_function_table+4)[a0],r3
  lde.b (ram_function_table+2)[a0],r1h
  lde.w (ram_function_table+0)[a0],a0
  mov.w #ram_execute,a1
  smovf.b
```

- `ram_run`: Set number of transfer data.
- `lde.w`: Set transfer source address (high order).
- `lde.b`: Set transfer source address (middle order and low order).
- `lde.w`: Set transfer destination address.

```assembly
CPU clock divided-by-8 setting
  cpu_slow

RAM execute processing
  ram_execute

CPU clock divided-by-1 setting
  cpu_fast
```

```assembly
rts
```

---

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

```assembly
flash_clear_status
  bclr fmr01
  bset fmr01
  and.b #00000010b,fmr0
```

- `bclr`: Enable CPU rewrite mode.
- `bset`: Select EW0 and disable each interrupt.
- `and.b`: Issue clear status register command.
- `mov.b`: Issue read array command.
- `mov.b`: Initialize flash memory control register 0.

```assembly
COMMAND_WRITE FLASH_ADDRESS, CLEAR_STATUS
COMMAND_WRITE FLASH_ADDRESS, PAGE_READ
```

```assembly
flash_clear_status_end
```

---

```
ram_function_table:
  .lword flash_clear_status
  .word flash_clear_status_end - flash_clear_status
  .lword flash_program
  .word flash_program_end - flash_program
  .lword flash_block_erase
  .word flash_block_erase_end - flash_block_erase
```
(14) Page program to flash memory (execute in RAM)

```assembly
; Enable CPU rewrite mode.
bcir fmr01
bset fmr01
and.b #00000010b,fmr0
or.b #11110000b,fmr1
and.b #00001111b,fmr1
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #0,r3
;
; Enable rewriting blocks A, B, C, and D.
and.b #00000010b,fmr0
or.b #11110000b,fmr1
and.b #00001111b,fmr1
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #0,r3
;
; Issue program command.
COMMAND_WRITE [a1a0], 040h
ste.b r0l,[a1a0]
;
; Set program data offset to A0.
;
; Set program addresses (middle order and low order) to A0.
mov.w a0,r2
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Select EW0 and disable each interrupt.
;
; Set program data offset + 1.
;
; Set program data.
;
; Set program addresses (middle order and low order) to A0.
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #0,r3
;
; Initialize program data offset.
;
; Enable rewrite mode.
bcir fmr13
bset fmr13
mov.w r3,a0
mov.b page_buffer[a0],r0l
mov.w r2,a0
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.
;
; Initialize flash memory control register 0.
mov.b #11110000b,fmr1
mov.b #000h,fmr0
;
; Disable lock bit.
bcir fmr13
bset fmr13
mov.w r3,a0
mov.b page_buffer[a0],r0l
mov.w r2,a0
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Issue program command.
COMMAND_WRITE [a1a0], 040h
ste.b r0l,[a1a0]
;
; Set program data.
;
; Set program addresses (middle order and low order) to A0.
mov.w a0,r2
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.
mov.b #11110000b,fmr1
mov.b #000h,fmr0
;
; Enable rewrite mode.
bcir fmr01
bset fmr01
and.b #00000010b,fmr0
or.b #11110000b,fmr1
and.b #00001111b,fmr1
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #0,r3
;
; Issue program command.
COMMAND_WRITE [a1a0], 040h
ste.b r0l,[a1a0]
;
; Set program data.
;
; Set program addresses (middle order and low order) to A0.
mov.w a0,r2
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.
mov.b #11110000b,fmr1
mov.b #000h,fmr0
;
; Enable rewrite mode.
bcir fmr01
bset fmr01
and.b #00000010b,fmr0
or.b #11110000b,fmr1
and.b #00001111b,fmr1
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #0,r3
;
; Issue program command.
COMMAND_WRITE [a1a0], 040h
ste.b r0l,[a1a0]
;
; Set program data.
;
; Set program addresses (middle order and low order) to A0.
mov.w a0,r2
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.
mov.b #11110000b,fmr1
mov.b #000h,fmr0
;
; Enable rewrite mode.
bcir fmr01
bset fmr01
and.b #00000010b,fmr0
or.b #11110000b,fmr1
and.b #00001111b,fmr1
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #0,r3
;
; Issue program command.
COMMAND_WRITE [a1a0], 040h
ste.b r0l,[a1a0]
;
; Set program data.
;
; Set program addresses (middle order and low order) to A0.
mov.w a0,r2
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.
mov.b #11110000b,fmr1
mov.b #000h,fmr0
;
; Enable rewrite mode.
bcir fmr01
bset fmr01
and.b #00000010b,fmr0
or.b #11110000b,fmr1
and.b #00001111b,fmr1
mov.w address,a0
mov.w address+2,a1
mov.w a0,r2
mov.w #0,r3
;
; Issue program command.
COMMAND_WRITE [a1a0], 040h
ste.b r0l,[a1a0]
;
; Set program data.
;
; Set program addresses (middle order and low order) to A0.
mov.w a0,r2
add.w #1,r3
add.w #1,r2
#100h == r3
Yes (page program completed)
;
; Enable lock bit and disable rewriting data flash blocks A, B, C, and D.
mov.b #11110000b,fmr1
mov.b #000h,fmr0
;
; Enable rewrite mode.
(15) Block erase to flash memory (execute in RAM)

```assembly
; Enable CPU rewrite mode.
bclr fmr01

; Select EW0 and disable each interrupt.
bset fmr01

and.b #00000010b,fmr0

; Disable lock bit.
bclr fmr13

or.b #11110000b,fmr1

; Enable rewriting blocks A, B, C, and D.
and.b #00001111b,fmr1

mov.w address_a0

mov.w address+2,a1

COMMAND_WRITE [a1a0], BLOCK_ERASE

; Issue block erase command.
mov.b #0d0h,r1l

; Write confirmation command.
mov.w address_a0

ste.b r1l,[a1a0]

; Reset watchdog timer.

; Wait for ready.

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

; Initialize flash memory control register 0.
mov.b #11110000b,fmr1

mov.b #00h,fmr0

or.b #11110000b,fmr1

and.b #00001111b,fmr1

and.b #00000010b,fmr0

mov.b #00h,fmr0

mov.b #11110000b,fmr1

mov.b #00h,fmr0

rts

flash_block_erase_end
```

(16) Command write macro

```assembly
COMMAND_WRITE.MACRO address,data

; Reset watchdog timer.
mov.b #000h,wdtr

; Reset watchdog timer.
mov.b #0ffh,wdtr

; Wait for ready.

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

; Initialize flash memory control register 0.

; Write command.
mov.b #data,r1l

ste.b r1l,address

; Write command.

ENDM
```
(17) System interrupt handling

```
sys_undefined
  jmp.a 0efdch
  User undefined instruction interrupt

sys_overflow
  jmp.a 0efe0h
  User overflow interrupt

sys_brk
  jmp.a 0efe4h
  User BRK instruction interrupt

sys_addrmatch
  jmp.a 0efe8h
  User address match interrupt

sys_wdt
  jmp.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 Command</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></td>
<td></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></td>
<td></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></td>
<td></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>SRD</td>
<td>SRD1</td>
<td></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>User version</td>
<td></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>Command</th>
<th>Address</th>
<th>Data</th>
<th>Up to 259th byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer to MCU</td>
<td>FFh</td>
<td>Middle-order address</td>
<td>High-order address</td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td>Data 0</td>
<td>Up to Data 255</td>
<td></td>
</tr>
</tbody>
</table>

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

4.2.3  Procedure
(1) The page read command FFh is received at the first byte.
(2) The middle-order address is received at the second byte and the high-order address is received at the third byte.
(3) The content in the low-order address 00h is sequentially transmitted from the fourth byte.
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></td>
<td>Up to data</td>
</tr>
</tbody>
</table>

Programmer to MCU

<table>
<thead>
<tr>
<th>1st byte</th>
<th>2nd byte</th>
<th>3rd byte</th>
<th>4th byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>41h</td>
<td>Middle-order address</td>
<td>High-order address</td>
<td>Data 0</td>
</tr>
</tbody>
</table>

MCU to Programmer

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

4.3.3 Procedure

1. The page program command 41h is received at the first byte.
2. The middle-order address is received at the second byte and the high-order address is received at the third byte.
3. The programming data to the low-order address 00h is received from the fourth byte.

When the programming data is less than 256 bytes, transmit FFh for the shortage. When programming data is more than 257 bytes, the data at the 257th byte is considered to be the data in the next command. If an error occurs during programming, SR4 becomes 1 (program status ends in error).

After executing this command, confirm the status of the flash memory with the read status register command.
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>Block address</th>
<th>Confirmation command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer to MCU</td>
<td>20h</td>
<td>Middle-order address</td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td></td>
<td>High-order address</td>
</tr>
</tbody>
</table>

4.4.3 Procedure

1. The block erase command 20h is received at the first byte.
2. The middle-order address is received at the second byte and the high-order address is received at the third byte.
3. The confirmation command D0h is received at the fourth byte.

After receiving the 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, SR5 becomes 1 (erase status ends in error).

After executing this command, confirm the status of the flash memory with the read status register command.
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></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></td>
<td>SRD output</td>
<td>SRD1 output</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SRD: Status register data
SRD1: Status register data 1

4.5.3 Procedure

1. The read status register command 70h is received at the first byte.
2. SRD is transmitted at the second byte.
3. SRD1 is transmitted at the third byte.
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>Ready</td>
</tr>
<tr>
<td>SR6 (bit 6)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR5 (bit 5)</td>
<td>Erase status</td>
<td>Error</td>
</tr>
<tr>
<td>SR4 (bit 4)</td>
<td>Program status</td>
<td>Error</td>
</tr>
<tr>
<td>SR3 (bit 3)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR2 (bit 2)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR1 (bit 1)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR0 (bit 0)</td>
<td></td>
<td>Reserved</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 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></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR14 (bit 6)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR13 (bit 5)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR12 (bit 4)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR11 (bit 3)</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>SR10 (bit 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SR9  (bit 1)</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>SR8  (bit 0)</td>
<td></td>
<td>Reserved</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>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>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 clear status register command 50h is received at the first byte.
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>Command</th>
<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>Programmer to MCU</td>
<td>F5h</td>
<td>ID1</td>
<td>ID2</td>
<td>ID3</td>
<td>Up to ID7</td>
</tr>
<tr>
<td>MCU to Programmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.7.3  Procedure
(1) The ID check function command F5h is received at the first byte.
(2) ID1 to ID7 are received from the second byte to the eighth byte, respectively.

After receiving the ID, the ID check starts. However, a user program area blank product returns the wait state for the control command from the programmer without performing ID check. When ID1 to ID7 all match, SR11 to SR10 become 11b (verified). If any of the IDs do not match, SR11 to SR10 become 01b (verify not matched).
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>Programmer to MCU</td>
<td>FBh</td>
<td></td>
<td></td>
<td></td>
<td>User</td>
</tr>
<tr>
<td>Version</td>
<td>MCU to Programmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Program downloader</td>
</tr>
</tbody>
</table>

4.8.3 Procedure

1. The version information output function command FBh is received at the first byte.
2. The program downloader version is transmitted at the high-order second byte first and then the low-order third byte.
3. The user program version is transmitted at the high-order fourth byte first and then low-order fifth byte.

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 bl_r835c.a30 file)

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

User version data
(in the sect30.inc file for 5. User Program Example)

```asm
User_Ver .equ 0010h  ; User version (Ver.0.10)
```
5. 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.

5.1 Function

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

5.2 Memory Map

Figure 5.1 shows a User Program Memory Map.
Figure 5.1  User Program Memory Map
5.3 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 5.1):

• 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).
6. Programmer Example

6.1 Control Pins

(1) Pins TXD and RXD
   These pins are for transmitting and receiving in UART mode.

(2) RESET pin
   This pin controls an MCU reset from the programmer.

(3) 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 6.1 Programmer Configuration
6.2 Programmer Functions

The following are the functions necessary for the programmer:

- Blank Check
- Erase
- Program
- Verify
- Read

6.3 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]

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

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
6.4 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

- **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

**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**

**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**
6.5 Program

Data (program) in the MCU’s specified on-chip flash memory area automatically or manually is programmed.
6.6 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.

![Diagram of Verify process]

- **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

- **Transmit page read command FFh, middle-order address, and high-order address**
  - Receive data (256 bytes)
  - **Verify Check OK**
6.7 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.

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

---

**Diagram:**

- 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
7. **Sample Programming Code**
A sample program can be downloaded from the Renesas Electronics website. To download, click “Application Notes” in the left-hand side menu of the R8C Family page.

8. **Reference Documents**
R8C/35C Group User’s Manual: Hardware Rev.1.00
The latest version can be downloaded from the Renesas Electronics website.

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

**Website and Support**
Renesas Electronics website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Apr 30, 2010</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
**General Precautions in the Handling of MPU/MCU Products**

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

<table>
<thead>
<tr>
<th>1. Handling of Unused Pins</th>
<th>Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.</th>
</tr>
</thead>
<tbody>
<tr>
<td>— 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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Processing at Power-on</th>
<th>The state of the product is undefined at the moment when power is supplied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>— 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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Prohibition of Access to Reserved Addresses</th>
<th>Access to reserved addresses is prohibited.</th>
</tr>
</thead>
<tbody>
<tr>
<td>— 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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Clock Signals</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>— 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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Differences between Products</th>
<th>Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.</th>
</tr>
</thead>
<tbody>
<tr>
<td>— The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.</td>
<td></td>
</tr>
</tbody>
</table>
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 example. 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 damages 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 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.

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 malfunction under certain usage conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by 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.