

## Overview

The M16C/6S group are highly integrated single-chip microcomputers with PLC (Power Line Communication) modem core and AFE (Analog Front End) in a 64-pin plastic molded LQFP package, which incorporates IT800 PLC modem technology developed by Yitran Communications Ltd. M16C/60 Series CPU core enables a high level of code efficiency and high-speed operation. In addition, the implementation of Yitran's patented DCSK (Differential Code Shift Keying) spread spectrum modulation technique in the IT800 modem core enables extremely robust communication over the existing electrical wiring, with data rates up to 7.5Kbps. The M16C/6S complies with worldwide regulations (FCC part 15, ARIB and CENELEC bands) and is suitable for a variety of narrowband applications like smart metering and home networking.

## Applications

Power Line Communication

## -----Table of Contents-----

Overview .....	1	Serial I/O .....	92
Memory .....	10	Clock Synchronous serial I/O Mode .....	101
Central Processing Unit (CPU) .....	11	UART Mode .....	109
SFR .....	13	Special Mode .....	117
Reset .....	19	SI/O3 and SI/O4 .....	132
Processor Mode .....	23	Programmable I/O Ports .....	137
Clock Generation Circuit .....	27	Electrical Characteristics .....	149
Protection .....	46	Flash Memory Version .....	160
Interrupts .....	47	IT800AFE (Analog Front End) .....	184
Watchdog Timer .....	66	Usage Notes .....	189
DMAC .....	68	Appendix .....	199
Timers .....	78		
Timer A .....	79		

Specifications written in this manual are believed to be accurate, but are not guaranteed to be entirely free of error. Specifications in this manual may be changed for functional or performance improvements. Please make sure your manual is the latest edition.

## Performance Outline

Table 1.1.1 lists performance outline of M16C/6S group.

**Table 1.1.1. Performance outline of M16C/6S group**

Item		Performance
CPU	Number of basic instructions	91 instructions
	Minimum Instruction Execution time	65.1 ns (f(BCLK)= 15.36MHz, Vcc= 3.0V to 3.6V)
	Operation Mode	Single-chip mode
	Memory Space	1M Byte
	Memory Capacity	ROM RAM
Peripheral Function	Port	Input/Output : 21 pins, Input : 1 pin
	Multifunction Timer	Timer A : 16 bits x 5 channels,
	Serial I/O	2 channels Clock synchronous, UART, I <sup>2</sup> C bus <sup>(1)</sup> , 1 channel UART, I <sup>2</sup> C bus <sup>(1)</sup> , 2 channels Clock synchronous(*) (*) 1 channel is internally connected to IT800
	DMAC	2 channels
	Watchdog Timer	15 bits x 1 channel (with prescaler)
	Interrupt	21 internal and 3 external sources, 4 software sources, 7 levels
	Clock Generation Circuit	2 circuits Main clock generation circuit with PLL synthesizer (*), On-chip oscillator, (*) This circuit contains a built-in feedback resistor.
	Electrical Characteristics	Power supply voltage Power Consumption
Flash memory Version	Program/Erase Supply Voltage	3.0V to 3.6V (Topr= 0 to 60°C)
	Program and Erase Endurance	100 times or 1,000 times <sup>(2)</sup>
Power consumption		70mA (Vcc= VccA= 3.3V, f(XIN)= 5.12MHz)
Operating Ambient Temperature <sup>(2)</sup>		-20 to 85°C -40 to 85°C -40 to 105°C
Package		64-pin plastic mold LQFP

### Notes:

1. I<sup>2</sup>C Bus is a registered trademark of Koninklijke Philips Electronics N. V.
2. See Tables 1.1.5 and 1.1.6 Product code for increased program/erase cycle version, and version of expanded operating ambient temperature.

## IT800 PHY performance outline of M16C/6S group.

The IT800 PHY is a PLC optimized Physical Layer (PHY) which consists of IT800 modem core and internal AFE (Analog Front End). The implementation of Yitran's patented DCSK spread spectrum modulation technique in the IT800 modem core enables extremely robust communication over the existing electrical wiring, with data rates up to 7.5Kbps. In addition to the inherent interference immunity provided by the DCSK modulation, the IT800 PHY utilizes several mechanisms for enhanced communication robustness, such as forward short block soft decoding error correction algorithm and special synchronization algorithms.

The IT800PHY communicates M16C core with clock synchronous serial I/O, interrupt and input/output ports in M16C/6S (Note). M16C/6S requires external AFE. Table 1.1.2 lists IT800 PHY performance outline of M16C/6S group.

**Table 1.1.2. IT800 PHY performance outline of M16C/6S group**

Item		Performance
Features		High immunity to signal fading, various noise characteristics, impedance modulation and phase/frequency distortion High in-phase and cross-phase reliability
Modulation Technique		DCSK (Differential Code Shift Keying)* *Yitran patented modulation technique
Error Correction/Detection		Forward short-block soft decoding error correction mechanism, CRC-16
Complies with W/W Regulations		FCC, ARIB, EN50065-1-CENELEC
Data Rate & Frequency Band	FCC & ARIB	120-400 KHz 7.5Kbps Standard Mode (SM) 5.0Kbps Robust Mode (RM) 1.25Kbps Extremely Robust Mode (ERM)
	CENELEC	A-Band (Outdoor): 20-80 KHz B-Band (Indoor): 95-125 KHz 2.5 Kbps Robust Mode (RM) 0.625Kbps Extremely Robust Mode (ERM)
Internal AFE		10bit-D/A converter, preamp, 1bit-A/D converter x 3 channels

### Note:

Direct operation of IT800 PHY is not recommended. Since the Layer 2 (DLL) handles the channel access procedure, using the IT800DLL as such assures coexistence with other IT800 technology based products, regardless of the vendor, protocol and the application. There is NO such coexistence if the device is used in direct operation of IT800 PHY. Note that the coexistence here is not with other technologies but with other IT800 technology based products. As for details, please refer to the next section and Appendix.

## About Firmware

Renesas recommends IT800DLL (Product name: D2DL) as a Data Link Layer (DLL) of M16C/6S group. The IT800 DLL is a PLC optimized DLL especially for the products based on Yitran's IT800 technology. For availability of IT800DLL, please contact Renesas technical support representative. For more details about the IT800DLL advantages, please refer to Appendix.

### Block Diagram and PLC application Outline

Figure 1.1.1 is a block diagram of the M16C/6S group and PLC application Outline.

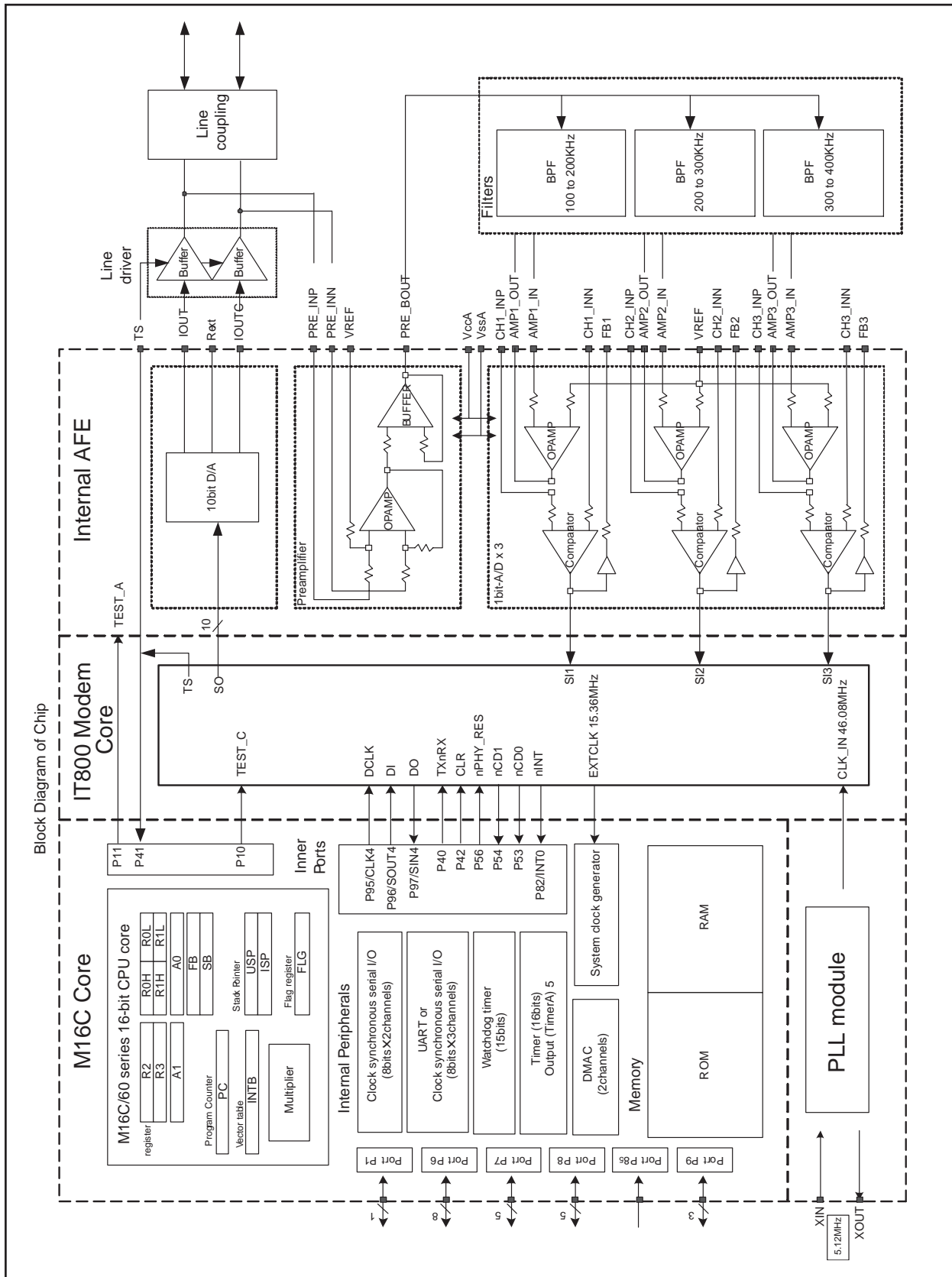


Figure 1.1.1. Block Diagram of Chip and PLC application Outline

### Product List

Tables 1.1.3 and 1.1.4 list the M16C/6S group product and Figure 1.1.2 shows the type numbers, memory sizes and packages.

**Table 1.1.3. Product List (1) M16C/6S**

Current of Dec. 2009

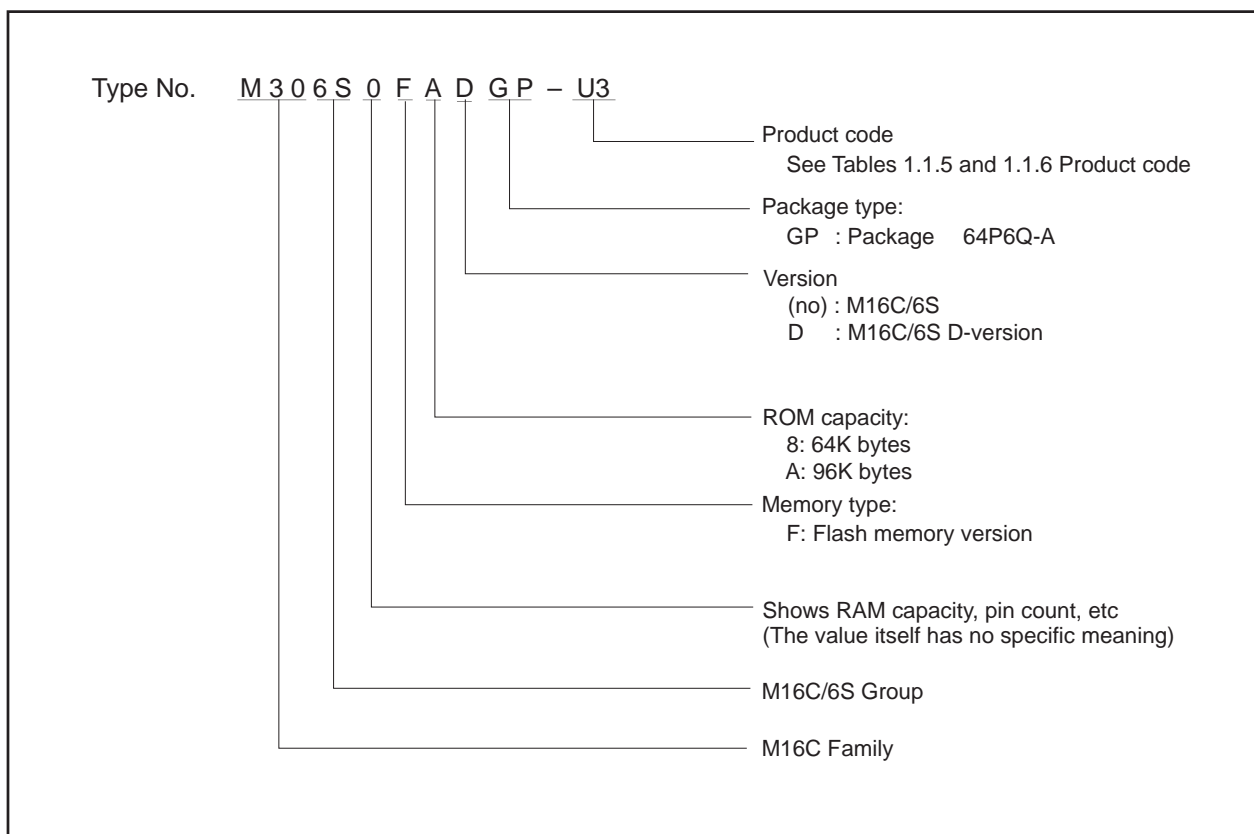
Type No.	ROM capacity	RAM capacity	Package type	Remarks	Product Code
M306S0FAGP	96K bytes	24K bytes	PLQP0064KB-A (64P6Q-A)	Flash memory	U3, U5, U7(D), U9(D)

(D): under development

**Table 1.1.4. Product List (2) M16C/6S D-version**

Current of Dec. 2009

Type No.	ROM capacity	RAM capacity	Package type	Remarks	Product Code
M306S0F8DGP	64K bytes	24K bytes	PLQP0064KB-A (64P6Q-A)	Flash memory	U3
M306S0FADGP	96K bytes	24K bytes	PLQP0064KB-A (64P6Q-A)	Flash memory	U3



**Figure 1.1.2. Type No., Memory Size, and Package**

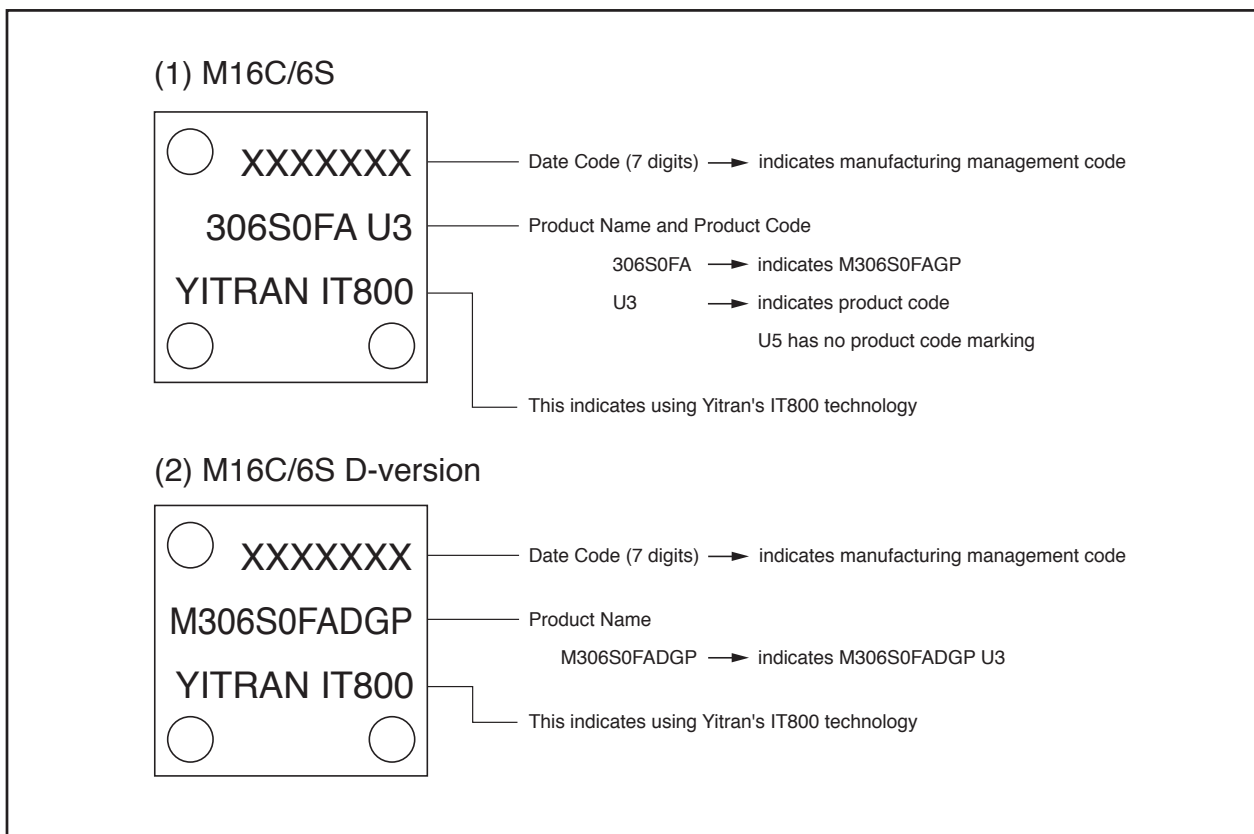
**Table 1.1.5. Product Code (1) M16C/6S**

Product Code	Package	Internal ROM		Microcomputer operating temperature
		E/W cycles	Temperature range	
U3	LEAD free	100	0°C to 60°C	-40°C to 85°C
U5				-20°C to 85°C
U7 (D)		1,000		-40°C to 85°C
U9 (D)				-20°C to 85°C

(D): under development

**Table 1.1.6. Product Code (2) M16C/6S D-version**

Product Code	Package	Internal ROM		Microcomputer operating temperature
		E/W cycles	Temperature range	
U3	LEAD free	100	0°C to 60°C	-40°C to 105°C



**Figure 1.1.3. Marking (Top View)**

### Pin Configuration

Figures 1.1.4 show the pin configurations (top view).

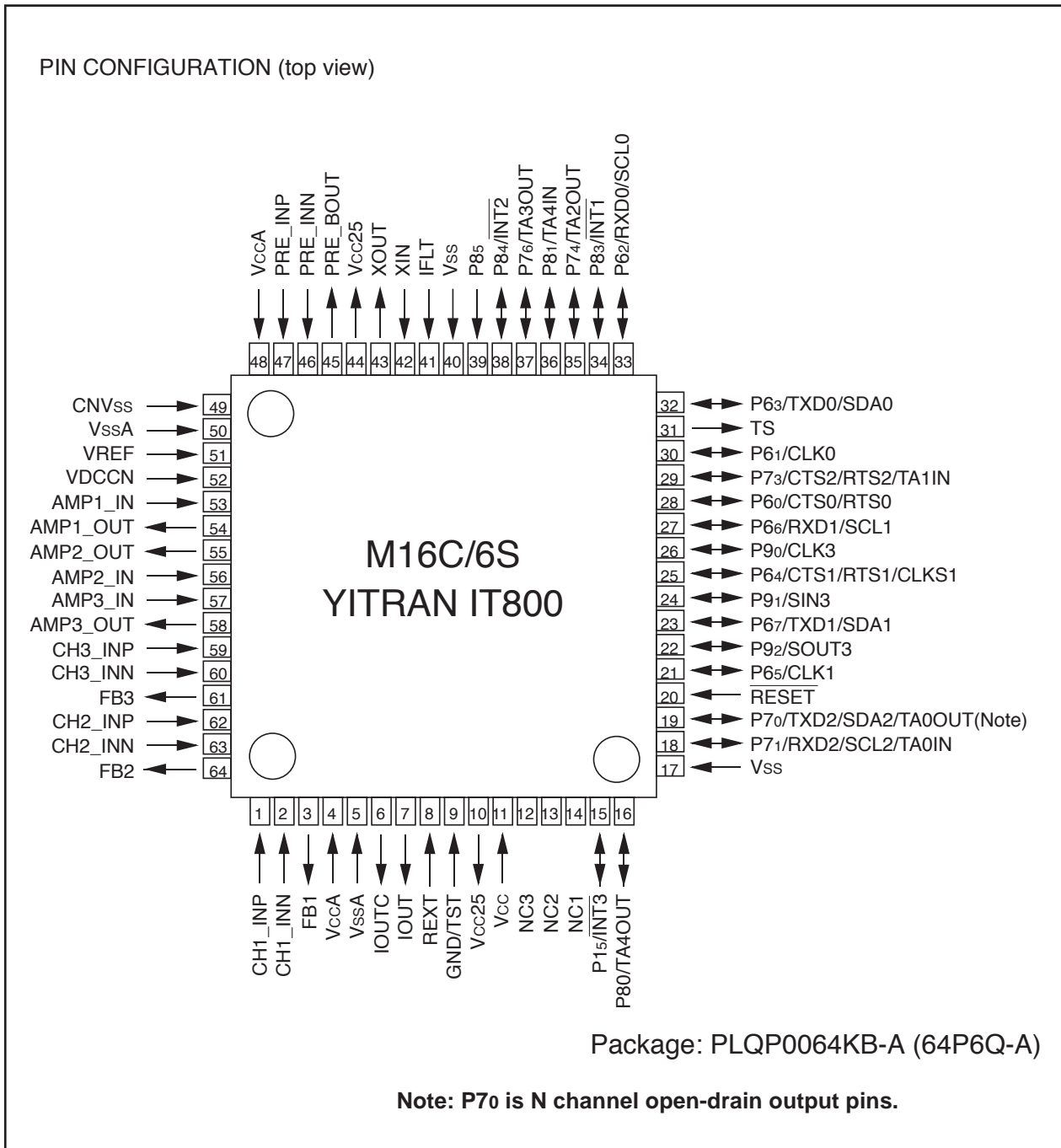


Figure 1.1.4. Pin Configuration (Top View)

Table 1.1.7 Pin Description (1)

Pin name	Signal name	I/O type	Function
Vcc, Vss	Power supply input		Apply 3.0V to 3.6V to the Vcc pin. Apply 0V to the Vss pin.
CNVss	CNVss	Input	This is a pin for changing flash memory mode. Usually, connect to Vss.
RESET	Reset input	Input	"L" on this input resets the microcomputer.
XIN	Clock input	Input	I/O pins for the main clock generation circuit. Connect a ceramic resonator or crystal oscillator between XIN and XOUT (3). To use the external clock, input the clock from XIN and leave XOUT open.
XOUT	Clock output	Output	
VCCA	Analog power supply input		This pin is a power supply input of analog circuit. Connect to Vcc.
VSSA	Analog power supply input		This pin is a power supply input of analog circuit. Connect to Vss.
GND/TST	Input for test	Input	This is input pin for test. Connect to Vss.
P15	I/O port P1	Input/output	This is an 1-bit I/O port equivalent to P6. By choosing by the program, it functions as an input pin of INT interrupt.
Vcc25	Power supply		It is the 2.5V power supply pin which is carrying out internal generating. Connect Vcc25 two pins each other and add stabilization capacity.
P60 to P67	I/O port P6	Input/output	This is the 8-bit I/O port of CMOS. It has a direction register for choosing I/O, and is made for every pin in an input port or an output port. An input port can choose the existence of pull-up resistance in a 4-bit unit by the program. It functions as an I/O pin of what is chosen by the program which are therefore UART0 and UART1.
P70, P71, P73, P74, P76	I/O port P7	Input/output	It is an I/O port with a function equivalent to P6 (however, P70 N channel open-drain output). By choosing by the program, it functions as an I/O pin of timers A0 to A3. Moreover, P70 to P73 function also as an I/O pin of UART2.
P80, P81, P83, P84, P85	I/O port P8	Input/output	P80, P81, P83, and P84 are I/O ports with a function equivalent to P6. By choosing by the program, P80 to P81 function as an I/O pin of timer A4, and P83 to P84 functions as an input pin of INT interruption.
	Input port P85	Input	P85 cannot be used as general input pin. Pull-up resistance cannot set up this pin. This pin must be pulled-up externally to Vcc, and be fixed to "H".
P90 to P92	I/O port P9	Input/output	It is an I/O port with a function equivalent to P6. It functions as an I/O pin of SILO3 by choosing by the program.
TS	Output port TS	Output	This is the pin with which IT800 controls ON/OFF of an output to the external transmission AMP at the time of power line communication.
P10	Inner port P10	keep Input	Port for IT800 Testing. Please set Direction Register PD1_0 "0" for keeping Input.
P11	Inner port P11	keep Input	Port for AFE Testing. Please set Direction Register PD1_1 "0" for keeping Input.
P40	Inner port P40	Output to IT800	I/O Port for communication between M16C Core and IT800 Modem. Please set Direction Register PD4_0 "1" to Output a signal to IT800. The output signal is connected to TxnRX of IT800 inside of chip.
P41	Inner port P41	Input from IT800	I/O Port for communication between M16C Core and IT800 Modem (Note.1). Please set Direction Register PD4_1 "0" to Input a signal from IT800. The TS signal is input from IT800 inside of chip.
P42	Inner port P42	Output to IT800	I/O Port for communication between M16C Core and IT800 Modem. Please set Direction Register PD4_2 "1" to Output a signal to IT800. The output signal is connected to CLR of IT800 inside of chip.
P53	Inner port P53	Input from IT800	I/O Port for communication between M16C Core and IT800 Modem (Note.1). Please set Direction Register PD5_3 "0" to Input a signal from IT800. The nCDO signal is input from IT800 inside of chip.
P54	Inner port P54	Input from IT800	I/O Port for communication between M16C Core and IT800 Modem (Note.1). Please set Direction Register PD5_4 "0" to Input a signal from IT800. The nCD1 signal is input from IT800 inside of chip.
P56	Inner port P56	Output to IT800	I/O Port for communication between M16C Core and IT800 Modem. Please set Direction Register PD5_6 "1" to Output a signal to IT800. The output signal is connected to nPHY_RES of IT800 inside of chip.
P82	Inner port P82	Input from IT800	I/O Port for communication between M16C Core and IT800 Modem (Note.1). nINT signal of IT800 is connected to this port inside of chip for carrying out Interrupt function by software.
P95	Inner port P95	Output to IT800	I/O Port for communication between M16C Core and IT800 Modem. This port is connected to DCLK signal of IT800 inside of chip as SILO4 CLK4 functional Output port by choosing by the program.
P96	Inner port P96	Output to IT800	I/O Port for communication between M16C Core and IT800 Modem. This port is connected to DI signal of IT800 inside of chip as SILO4 SOUT4 functional Output port by choosing by the program.
P97	Inner port P97	Input from IT800	I/O Port for communication between M16C Core and IT800 Modem. This port is connected to DI signal of IT800 inside of chip as SILO4 SOUT4 functional Input port by choosing by the program.

## NOTES:

1. In case of Direction Register "1", any signal is not output from M16C to assigned signal of IT800 . Refer to Programmable I/O Ports pages.
2. Ask the oscillator maker the oscillation characteristic.



**Table 1.1.8 Pin Description (2) (Analog pin)**

Pin name	I/O type	Function
PRE-BOUT	Output	This is a pre-amp buffer output.
PRE-INN	Input	This is a pre-amp differential signal input.
PRE-INP	Input	This is a pre-amp differential signal input.
VREF	Input	This is the reference voltage input of amplifier common to channels 1, 2, and 3.
VDCCN	Input	This is a pin for a test. Usually, please carry out a pull-up.
AMP1-IN	Input	This is a channel 1 amplifier input.
AMP1-OUT	Output	This is a channel 1 amplifier output.
AMP2-IN	Input	This is a channel 2 amplifier input.
AMP2-OUT	Output	This is a channel 2 amplifier output.
AMP3-IN	Input	This is a channel 3 amplifier input.
AMP3-OUT	Output	This is a channel 3 amplifier output.
CHI-INP	Input	This is a channel 1 comparator differential input.
CHI-INN	Input	This is a channel 1 comparator differential input.
FB1	Output	This is a channel 1 comparator feedback output.
CH2-INP	Input	This is a channel 2 comparator differential input.
CH2-INN	Input	This is a channel 2 comparator differential input.
FB2	Output	This is a channel 2 comparator feedback output.
CH3-INP	Input	This is a channel 3 comparator differential input.
CH3-INN	Input	This is a channel 3 comparator differential input.
FB3	Output	This is a channel 3 comparator feedback output.
IOUTC	Output	This is the differential current output of DAC.
IOUT	Output	This is the differential current output of DAC.
REXT	Input	This is for external reference resistor of DAC.
IFLT	Input	This is the pin for low path filters of PLL.

# Memory

Figure 1.2.1 is a memory map of the M16C/6S group. The address space extends the 1M bytes from address 00000<sub>16</sub> to FFFFF<sub>16</sub>.

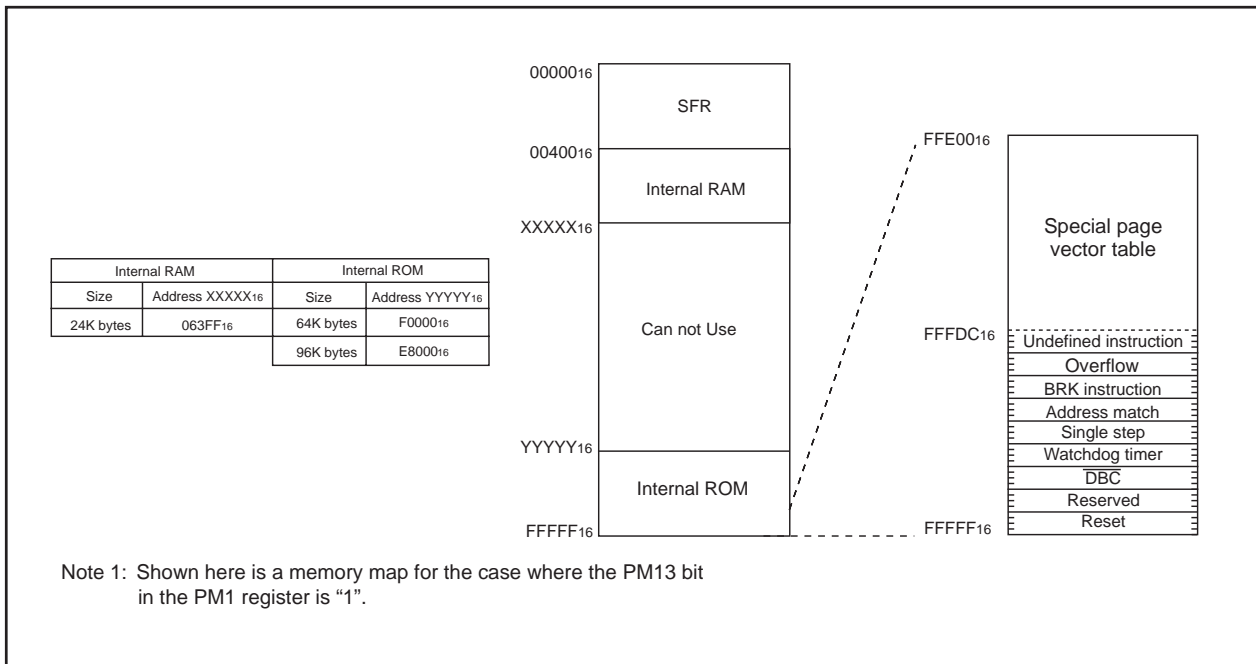
The internal ROM is allocated in a lower address direction beginning with address FFFFF<sub>16</sub>. For example, a 96-Kbyte internal ROM is allocated to the addresses from E8000<sub>16</sub> to FFFFF<sub>16</sub>.

The fixed interrupt vector table is allocated to the addresses from FFFDC<sub>16</sub> to FFFFF<sub>16</sub>. Therefore, store the start address of each interrupt routine here.

The internal RAM is allocated in an upper address direction beginning with address 00400<sub>16</sub>. For example, a 24-Kbytes internal RAM is allocated to the addresses from 00400<sub>16</sub> to 063FF<sub>16</sub>. In addition to storing data, the internal RAM also stores the stack used when calling subroutines and when interrupts are generated.

The SRF is allocated to the addresses from 00000<sub>16</sub> to 003FF<sub>16</sub>. Peripheral function control registers are located here. Of the SFR, any area which has no functions allocated is reserved for future use and cannot be used by users.

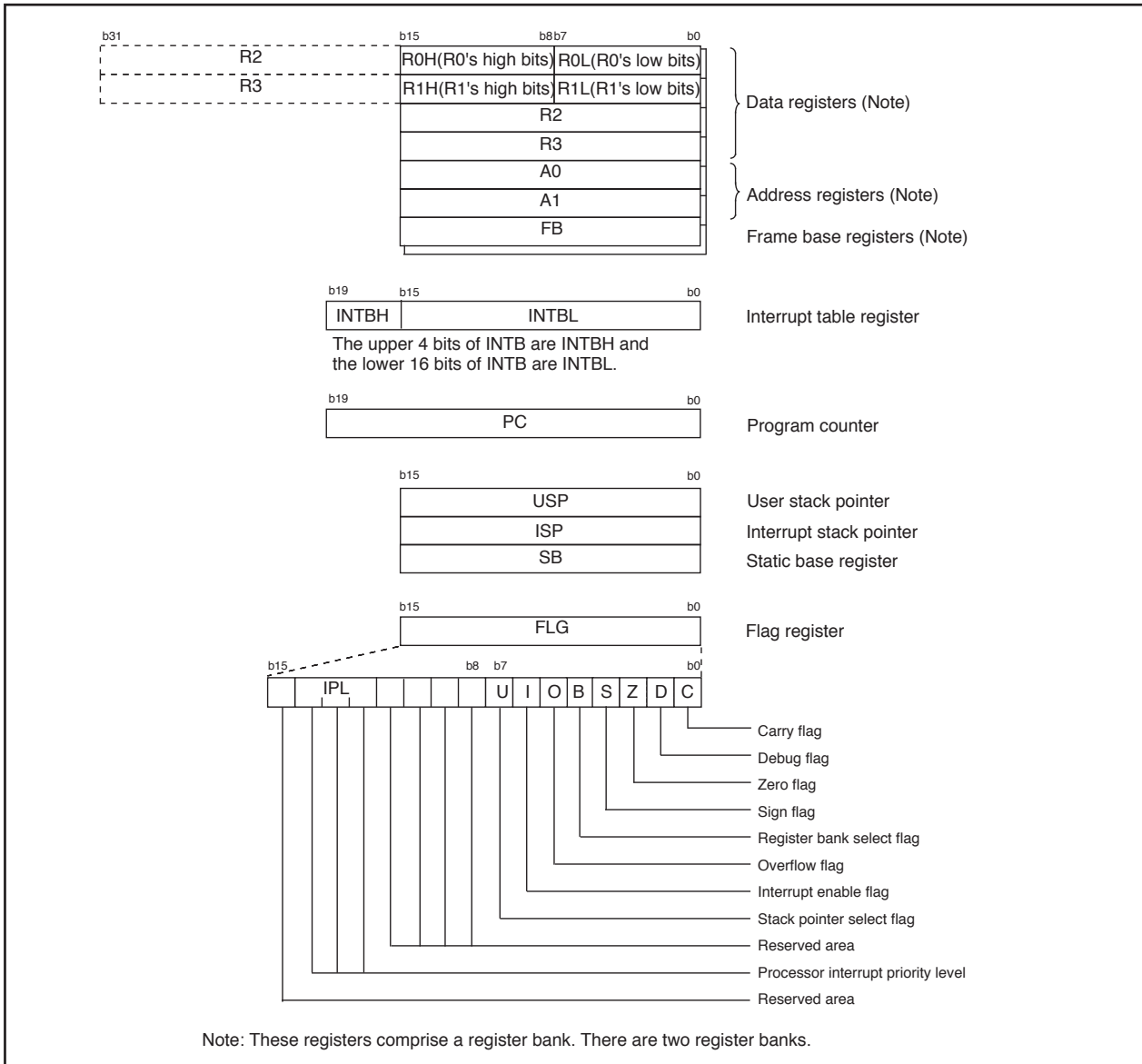
The special page vector table is allocated to the addresses from FFE00<sub>16</sub> to FFFDB<sub>16</sub>. This vector is used by the JMPS or JSRS instruction. For details, refer to the “M16C/60 and M16C/20 Series Software Manual.”



**Figure 1.2.1. Memory Map**

## Central Processing Unit (CPU)

Figure 1.3.1 shows the CPU registers. The CPU has 13 registers. Of these, R0, R1, R2, R3, A0, A1 and FB comprise a register bank. There are two register banks.



**Figure 1.3.1. Central Processing Unit Register**

### (1) Data Registers (R0, R1, R2 and R3)

The R0 register consists of 16 bits, and is used mainly for transfers and arithmetic/logic operations. R1 to R3 are the same as R0.

The R0 register can be separated between high (R0H) and low (R0L) for use as two 8-bit data registers. R1H and R1L are the same as R0H and R0L. Conversely, R2 and R0 can be combined for use as a 32-bit data register (R2R0). R3R1 is the same as R2R0.

### (2) Address Registers (A0 and A1)

The register A0 consists of 16 bits, and is used for address register indirect addressing and address register relative addressing. They also are used for transfers and logic/logic operations. A1 is the same as A0.

In some instructions, registers A1 and A0 can be combined for use as a 32-bit address register (A1A0).

### (3) Frame Base Register (FB)

FB is configured with 16 bits, and is used for FB relative addressing.

### (4) Interrupt Table Register (INTB)

INTB is configured with 20 bits, indicating the start address of an interrupt vector table.

### (5) Program Counter (PC)

PC is configured with 20 bits, indicating the address of an instruction to be executed.

### (6) User Stack Pointer (USP) and Interrupt Stack Pointer (ISP)

Stack pointer (SP) comes in two types: USP and ISP, each configured with 16 bits.

Your desired type of stack pointer (USP or ISP) can be selected by the U flag of FLG.

### (7) Static Base Register (SB)

SB is configured with 16 bits, and is used for SB relative addressing.

### (8) Flag Register (FLG)

FLG consists of 11 bits, indicating the CPU status.

- **Carry Flag (C Flag)**

This flag retains a carry, borrow, or shift-out bit that has occurred in the arithmetic/logic unit.

- **Debug Flag (D Flag)**

The D flag is used exclusively for debugging purpose. During normal use, it must be set to "0".

- **Zero Flag (Z Flag)**

This flag is set to "1" when an arithmetic operation resulted in 0; otherwise, it is "0".

- **Sign Flag (S Flag)**

This flag is set to "1" when an arithmetic operation resulted in a negative value; otherwise, it is "0".

- **Register Bank Select Flag (B Flag)**

Register bank 0 is selected when this flag is "0" ; register bank 1 is selected when this flag is "1".

- **Overflow Flag (O Flag)**

This flag is set to "1" when the operation resulted in an overflow; otherwise, it is "0".

- **Interrupt Enable Flag (I Flag)**

This flag enables a maskable interrupt.

Maskable interrupts are disabled when the I flag is "0", and are enabled when the I flag is "1". The I flag is cleared to "0" when the interrupt request is accepted.

- **Stack Pointer Select Flag (U Flag)**

ISP is selected when the U flag is "0"; USP is selected when the U flag is "1".

The U flag is cleared to "0" when a hardware interrupt request is accepted or an INT instruction for software interrupt Nos. 0 to 31 is executed.

- **Processor Interrupt Priority Level (IPL)**

IPL is configured with three bits, for specification of up to eight processor interrupt priority levels from level 0 to level 7.

If a requested interrupt has priority greater than IPL, the interrupt is enabled.

- **Reserved Area**

When write to this bit, write "0". When read, its content is indeterminate.

## SFR

Address	Register	Symbol	After reset
0000 <sub>16</sub>			
0001 <sub>16</sub>			
0002 <sub>16</sub>			
0003 <sub>16</sub>			
0004 <sub>16</sub>	Processor mode register 0 (Note 2)	PM0	XXXX0X00 <sub>2</sub> (CNV <sub>SS</sub> pin is "L")
0005 <sub>16</sub>	Processor mode register 1	PM1	00XX10X0 <sub>2</sub>
0006 <sub>16</sub>	System clock control register 0	CM0	01001000 <sub>2</sub>
0007 <sub>16</sub>	System clock control register 1	CM1	00100000 <sub>2</sub>
0008 <sub>16</sub>			
0009 <sub>16</sub>	Address match interrupt enable register	AIER	XXXXXX00 <sub>2</sub>
000A <sub>16</sub>	Protect register	PRCR	XX000000 <sub>2</sub>
000B <sub>16</sub>			
000C <sub>16</sub>	Oscillation stop detection register (Note 3)	CM2	0000X000 <sub>2</sub>
000D <sub>16</sub>			
000E <sub>16</sub>	Watchdog timer start register	WDTS	?? <sub>16</sub>
000F <sub>16</sub>	Watchdog timer control register	WDC	00?????? <sub>2</sub>
0010 <sub>16</sub>	Address match interrupt register 0	RMAD0	00 <sub>16</sub>
0011 <sub>16</sub>			00 <sub>16</sub>
0012 <sub>16</sub>			X0 <sub>16</sub>
0013 <sub>16</sub>			
0014 <sub>16</sub>	Address match interrupt register 1	RMAD1	00 <sub>16</sub>
0015 <sub>16</sub>			00 <sub>16</sub>
0016 <sub>16</sub>			X0 <sub>16</sub>
0017 <sub>16</sub>			
0018 <sub>16</sub>			
0019 <sub>16</sub>			
001A <sub>16</sub>			
001B <sub>16</sub>			
001C <sub>16</sub>			
001D <sub>16</sub>			
001E <sub>16</sub>	Processor mode register 2	PM2	XXX00000 <sub>2</sub>
001F <sub>16</sub>			
0020 <sub>16</sub>	DMA0 source pointer	SAR0	?? <sub>16</sub>
0021 <sub>16</sub>			?? <sub>16</sub>
0022 <sub>16</sub>			X? <sub>16</sub>
0023 <sub>16</sub>			
0024 <sub>16</sub>	DMA0 destination pointer	DAR0	?? <sub>16</sub>
0025 <sub>16</sub>			?? <sub>16</sub>
0026 <sub>16</sub>			X? <sub>16</sub>
0027 <sub>16</sub>			
0028 <sub>16</sub>	DMA0 transfer counter	TCR0	?? <sub>16</sub>
0029 <sub>16</sub>			?? <sub>16</sub>
002A <sub>16</sub>			
002B <sub>16</sub>			
002C <sub>16</sub>	DMA0 control register	DM0CON	00000?00 <sub>2</sub>
002D <sub>16</sub>			
002E <sub>16</sub>			
002F <sub>16</sub>			
0030 <sub>16</sub>	DMA1 source pointer	SAR1	?? <sub>16</sub>
0031 <sub>16</sub>			?? <sub>16</sub>
0032 <sub>16</sub>			X? <sub>16</sub>
0033 <sub>16</sub>			
0034 <sub>16</sub>	DMA1 destination pointer	DAR1	?? <sub>16</sub>
0035 <sub>16</sub>			?? <sub>16</sub>
0036 <sub>16</sub>			X? <sub>16</sub>
0037 <sub>16</sub>			
0038 <sub>16</sub>	DMA1 transfer counter	TCR1	?? <sub>16</sub>
0039 <sub>16</sub>			?? <sub>16</sub>
003A <sub>16</sub>			
003B <sub>16</sub>			
003C <sub>16</sub>	DMA1 control register	DM1CON	00000?00 <sub>2</sub>
003D <sub>16</sub>			
003E <sub>16</sub>			
003F <sub>16</sub>			

Note 1: The blank areas are reserved and cannot be used by users.

Note 2: The PM00 and PM01 bits do not change at software reset, watchdog timer reset and oscillation stop detection reset.

Note 3: The CM20, CM21, and CM27 bits do not change at oscillation stop detection reset.

X : Nothing is mapped to this bit

? : Undefined

Address	Register	Symbol	After reset
0040 <sub>16</sub>			
0041 <sub>16</sub>			
0042 <sub>16</sub>			
0043 <sub>16</sub>			
0044 <sub>16</sub>	INT3 interrupt control register	INT3IC	XX00?0002
0045 <sub>16</sub>			
0046 <sub>16</sub>	UART1 BUS collision detection interrupt control register	U1BCNIC	XXXX?0002
0047 <sub>16</sub>	UART0 BUS collision detection interrupt control register	U0BCNIC	XXXX?0002
0048 <sub>16</sub>	SI/O4 interrupt control register (S4IC)	S4IC	XX00?0002
0049 <sub>16</sub>	SI/O3 interrupt control register	S3IC	XX00?0002
004A <sub>16</sub>	UART2 Bus collision detection interrupt control register	BCNIC	XXXX?0002
004B <sub>16</sub>	DMA0 interrupt control register	DM0IC	XXXX?0002
004C <sub>16</sub>	DMA1 interrupt control register	DM1IC	XXXX?0002
004D <sub>16</sub>			
004E <sub>16</sub>			
004F <sub>16</sub>	UART2 transmit interrupt control register	S2TIC	XXXX?0002
0050 <sub>16</sub>	UART2 receive interrupt control register	S2RIC	XXXX?0002
0051 <sub>16</sub>	UART0 transmit interrupt control register	S0TIC	XXXX?0002
0052 <sub>16</sub>	UART0 receive interrupt control register	S0RIC	XXXX?0002
0053 <sub>16</sub>	UART1 transmit interrupt control register	S1TIC	XXXX?0002
0054 <sub>16</sub>	UART1 receive interrupt control register	S1RIC	XXXX?0002
0055 <sub>16</sub>	Timer A0 interrupt control register	TA0IC	XXXX?0002
0056 <sub>16</sub>	Timer A1 interrupt control register	TA1IC	XXXX?0002
0057 <sub>16</sub>	Timer A2 interrupt control register	TA2IC	XXXX?0002
0058 <sub>16</sub>	Timer A3 interrupt control register	TA3IC	XXXX?0002
0059 <sub>16</sub>	Timer A4 interrupt control register	TA4IC	XXXX?0002
005A <sub>16</sub>			
005B <sub>16</sub>			
005C <sub>16</sub>			
005D <sub>16</sub>	INT0 interrupt control register	INT0IC	XX00?0002
005E <sub>16</sub>	INT1 interrupt control register	INT1IC	XX00?0002
005F <sub>16</sub>	INT2 interrupt control register	INT2IC	XX00?0002
0060 <sub>16</sub>			
0061 <sub>16</sub>			
0062 <sub>16</sub>			
0063 <sub>16</sub>			
0064 <sub>16</sub>			
0065 <sub>16</sub>			
0066 <sub>16</sub>			
0067 <sub>16</sub>			
0068 <sub>16</sub>			
0069 <sub>16</sub>			
006A <sub>16</sub>			
006B <sub>16</sub>			
006C <sub>16</sub>			
006D <sub>16</sub>			
006E <sub>16</sub>			
006F <sub>16</sub>			
0070 <sub>16</sub>			
0071 <sub>16</sub>			
0072 <sub>16</sub>			
0073 <sub>16</sub>			
0074 <sub>16</sub>			
0075 <sub>16</sub>			
0076 <sub>16</sub>			
0077 <sub>16</sub>			
0078 <sub>16</sub>			
0079 <sub>16</sub>			
007A <sub>16</sub>			
007B <sub>16</sub>			
007C <sub>16</sub>			
007D <sub>16</sub>			
007E <sub>16</sub>			
007F <sub>16</sub>			

Note :The blank areas are reserved and cannot be used by users.

X : Nothing is mapped to this bit

? : Undefined

Address	Register	Symbol	After reset
0080 <sub>16</sub>			
0081 <sub>16</sub>			
0082 <sub>16</sub>			
0083 <sub>16</sub>			
0084 <sub>16</sub>			
0085 <sub>16</sub>			
0086 <sub>16</sub>			
≈			≈
01B0 <sub>16</sub>			
01B1 <sub>16</sub>			
01B2 <sub>16</sub>			
01B3 <sub>16</sub>			
01B4 <sub>16</sub>			
01B5 <sub>16</sub>	Flash memory control register 1 (Note 2)	FMR1	0?00??0?2
01B6 <sub>16</sub>			
01B7 <sub>16</sub>	Flash memory control register 0 (Note 2)	FMR0	??0000012
01B8 <sub>16</sub>	Address match interrupt register 2	RMAD2	00 <sub>16</sub>
01B9 <sub>16</sub>			00 <sub>16</sub>
01BA <sub>16</sub>			X0 <sub>16</sub>
01BB <sub>16</sub>	Address match interrupt enable register 2	AIER2	XXXXXX002
01BC <sub>16</sub>	Address match interrupt register 3	RMAD3	00 <sub>16</sub>
01BD <sub>16</sub>			00 <sub>16</sub>
01BE <sub>16</sub>			X0 <sub>16</sub>
01BF <sub>16</sub>			
≈			≈
0250 <sub>16</sub>			
0251 <sub>16</sub>			
0252 <sub>16</sub>			
0253 <sub>16</sub>			
0254 <sub>16</sub>			
0255 <sub>16</sub>			
0256 <sub>16</sub>			
0257 <sub>16</sub>			
0258 <sub>16</sub>			
0259 <sub>16</sub>			
025A <sub>16</sub>			
025B <sub>16</sub>			
025C <sub>16</sub>			
025D <sub>16</sub>			
025E <sub>16</sub>	Peripheral clock select register	PCLKR	000000112
025F <sub>16</sub>			
≈			≈
0330 <sub>16</sub>			
0331 <sub>16</sub>			
0332 <sub>16</sub>			
0333 <sub>16</sub>			
0334 <sub>16</sub>			
0335 <sub>16</sub>			
0336 <sub>16</sub>			
0337 <sub>16</sub>			
0338 <sub>16</sub>			
0339 <sub>16</sub>			
033A <sub>16</sub>			
033B <sub>16</sub>			
033C <sub>16</sub>			
033D <sub>16</sub>			
033E <sub>16</sub>			
033F <sub>16</sub>			

Note 1: The blank areas are reserved and cannot be used by users.

Note 2: This register is included in the flash memory version.

X : Nothing is mapped to this bit

? : Undefined

Address	Register	Symbol	After reset
0340 <sub>16</sub>			
0341 <sub>16</sub>			
0342 <sub>16</sub>			
0343 <sub>16</sub>			
0344 <sub>16</sub>			
0345 <sub>16</sub>			
0346 <sub>16</sub>			
0347 <sub>16</sub>			
0348 <sub>16</sub>			
0349 <sub>16</sub>			
034A <sub>16</sub>			
034B <sub>16</sub>			
034C <sub>16</sub>			
034D <sub>16</sub>			
034E <sub>16</sub>			
034F <sub>16</sub>			
0350 <sub>16</sub>			
0351 <sub>16</sub>			
0352 <sub>16</sub>			
0353 <sub>16</sub>			
0354 <sub>16</sub>			
0355 <sub>16</sub>			
0356 <sub>16</sub>			
0357 <sub>16</sub>			
0358 <sub>16</sub>			
0359 <sub>16</sub>			
035A <sub>16</sub>			
035B <sub>16</sub>			
035C <sub>16</sub>			
035D <sub>16</sub>			
035E <sub>16</sub>	Interrupt cause select register 2	IFSR2A	00XXXXXX2
035F <sub>16</sub>	Interrupt cause select register	IFSR	00 <sub>16</sub>
0360 <sub>16</sub>	SI/O3 transmit/receive register	S3TRR	?? <sub>16</sub>
0361 <sub>16</sub>			
0362 <sub>16</sub>	SI/O3 control register	S3C	010000002
0363 <sub>16</sub>	SI/O3 bit rate generator	S3BRG	?? <sub>16</sub>
0364 <sub>16</sub>	SI/O4 transmit/receive register	S4TRR	?? <sub>16</sub>
0365 <sub>16</sub>			
0366 <sub>16</sub>	SI/O4 control register	S4C	010000002
0367 <sub>16</sub>	SI/O4 bit rate generator	S4BRG	?? <sub>16</sub>
0368 <sub>16</sub>			
0369 <sub>16</sub>			
036A <sub>16</sub>			
036B <sub>16</sub>			
036C <sub>16</sub>	UART0 special mode register 4	U0SMR4	00 <sub>16</sub>
036D <sub>16</sub>	UART0 special mode register 3	U0SMR3	000X0X0X2
036E <sub>16</sub>	UART0 special mode register 2	U0SMR2	X00000002
036F <sub>16</sub>	UART0 special mode register	U0SMR	X00000002
0370 <sub>16</sub>	UART1 special mode register 4	U1SMR4	00 <sub>16</sub>
0371 <sub>16</sub>	UART1 special mode register 3	U1SMR3	000X0X0X2
0372 <sub>16</sub>	UART1 special mode register 2	U1SMR2	X00000002
0373 <sub>16</sub>	UART1 special mode register	U1SMR	X00000002
0374 <sub>16</sub>	UART2 special mode register 4	U2SMR4	00 <sub>16</sub>
0375 <sub>16</sub>	UART2 special mode register 3	U2SMR3	000X0X0X2
0376 <sub>16</sub>	UART2 special mode register 2	U2SMR2	X00000002
0377 <sub>16</sub>	UART2 special mode register	U2SMR	X00000002
0378 <sub>16</sub>	UART2 transmit/receive mode register	U2MR	00 <sub>16</sub>
0379 <sub>16</sub>	UART2 bit rate generator	U2BRG	?? <sub>16</sub>
037A <sub>16</sub>	UART2 transmit buffer register	U2TB	????????2
037B <sub>16</sub>			XXXXXXXX?2
037C <sub>16</sub>	UART2 transmit/receive control register 0	U2C0	000010002
037D <sub>16</sub>	UART2 transmit/receive control register 1	U2C1	000000102
037E <sub>16</sub>	UART2 receive buffer register	U2RB	????????2
037F <sub>16</sub>			?????XX?2

Note : The blank areas are reserved and cannot be used by users.

X : Nothing is mapped to this bit

? : Undefined



Address	Register	Symbol	After reset
0380 <sub>16</sub>	Count start flag	TABSR	0016
0381 <sub>16</sub>			
0382 <sub>16</sub>	One-shot start flag	ONSF	0016
0383 <sub>16</sub>	Trigger select register	TRGSR	0016
0384 <sub>16</sub>	Up-down flag	UDF	0016
0385 <sub>16</sub>			
0386 <sub>16</sub>	Timer A0 register	TA0	??16
0387 <sub>16</sub>			??16
0388 <sub>16</sub>	Timer A1 register	TA1	??16
0389 <sub>16</sub>			??16
038A <sub>16</sub>	Timer A2 register	TA2	??16
038B <sub>16</sub>			??16
038C <sub>16</sub>	Timer A3 register	TA3	??16
038D <sub>16</sub>			??16
038E <sub>16</sub>	Timer A4 register	TA4	??16
038F <sub>16</sub>			??16
0390 <sub>16</sub>			
0391 <sub>16</sub>			
0392 <sub>16</sub>			
0393 <sub>16</sub>			
0394 <sub>16</sub>			
0395 <sub>16</sub>			
0396 <sub>16</sub>	Timer A0 mode register	TA0MR	0016
0397 <sub>16</sub>	Timer A1 mode register	TA1MR	0016
0398 <sub>16</sub>	Timer A2 mode register	TA2MR	0016
0399 <sub>16</sub>	Timer A3 mode register	TA3MR	0016
039A <sub>16</sub>	Timer A4 mode register	TA4MR	0016
039B <sub>16</sub>			
039C <sub>16</sub>			
039D <sub>16</sub>			
039E <sub>16</sub>			
039F <sub>16</sub>			
03A0 <sub>16</sub>	UART0 transmit/receive mode register	U0MR	0016
03A1 <sub>16</sub>	UART0 bit rate generator	U0BRG	??16
03A2 <sub>16</sub>	UART0 transmit buffer register	U0TB	?????????
03A3 <sub>16</sub>			XXXXXXXX?2
03A4 <sub>16</sub>	UART0 transmit/receive control register 0	U0C0	000010002
03A5 <sub>16</sub>	UART0 transmit/receive control register 1	U0C1	000000102
03A6 <sub>16</sub>	UART0 receive buffer register	U0RB	?????????
03A7 <sub>16</sub>			?????XX?2
03A8 <sub>16</sub>	UART1 transmit/receive mode register	U1MR	0016
03A9 <sub>16</sub>	UART1 bit rate generator	U1BRG	??16
03AA <sub>16</sub>	UART1 transmit buffer register	U1TB	?????????
03AB <sub>16</sub>			XXXXXXXX?2
03AC <sub>16</sub>	UART1 transmit/receive control register 0	U1C0	000010002
03AD <sub>16</sub>	UART1 transmit/receive control register 1	U1C1	000000102
03AE <sub>16</sub>	UART1 receive buffer register	U1RB	?????????
03AF <sub>16</sub>			?????XX?2
03B0 <sub>16</sub>	UART transmit/receive control register 2	UCON	X00000002
03B1 <sub>16</sub>			
03B2 <sub>16</sub>			
03B3 <sub>16</sub>			
03B4 <sub>16</sub>			
03B5 <sub>16</sub>			
03B6 <sub>16</sub>			
03B7 <sub>16</sub>			
03B8 <sub>16</sub>	DMA0 request cause select register	DM0SL	0016
03B9 <sub>16</sub>			
03BA <sub>16</sub>	DMA1 request cause select register	DM1SL	0016
03BB <sub>16</sub>			
03BC <sub>16</sub>			
03BD <sub>16</sub>			
03BE <sub>16</sub>			
03BF <sub>16</sub>			

Note : The blank areas are reserved and cannot be used by users.

X : Nothing is mapped to this bit

? : Undefined

Address	Register	Symbol	After reset
03C0 <sub>16</sub>			
03C1 <sub>16</sub>			
03C2 <sub>16</sub>			
03C3 <sub>16</sub>			
03C4 <sub>16</sub>			
03C5 <sub>16</sub>			
03C6 <sub>16</sub>			
03C7 <sub>16</sub>			
03C8 <sub>16</sub>			
03C9 <sub>16</sub>			
03CA <sub>16</sub>			
03CB <sub>16</sub>			
03CC <sub>16</sub>			
03CD <sub>16</sub>			
03CE <sub>16</sub>			
03CF <sub>16</sub>			
03D0 <sub>16</sub>			
03D1 <sub>16</sub>			
03D2 <sub>16</sub>			
03D3 <sub>16</sub>			
03D4 <sub>16</sub>			
03D5 <sub>16</sub>			
03D6 <sub>16</sub>			
03D7 <sub>16</sub>			
03D8 <sub>16</sub>			
03D9 <sub>16</sub>			
03DA <sub>16</sub>			
03DB <sub>16</sub>			
03DC <sub>16</sub>			
03DD <sub>16</sub>			
03DE <sub>16</sub>			
03DF <sub>16</sub>			
03E0 <sub>16</sub>			
03E1 <sub>16</sub>	Port P1 register	P1	?? <sub>16</sub>
03E2 <sub>16</sub>			
03E3 <sub>16</sub>	Port P1 direction register	PD1	00 <sub>16</sub>
03E4 <sub>16</sub>			
03E5 <sub>16</sub>			
03E6 <sub>16</sub>			
03E7 <sub>16</sub>			
03E8 <sub>16</sub>	Port P4 register	P4	?? <sub>16</sub>
03E9 <sub>16</sub>	Port P5 register	P5	?? <sub>16</sub>
03EA <sub>16</sub>	Port P4 direction register	PD4	00 <sub>16</sub>
03EB <sub>16</sub>	Port P5 direction register	PD5	00 <sub>16</sub>
03EC <sub>16</sub>	Port P6 register	P6	?? <sub>16</sub>
03ED <sub>16</sub>	Port P7 register	P7	?? <sub>16</sub>
03EE <sub>16</sub>	Port P6 direction register	PD6	00 <sub>16</sub>
03EF <sub>16</sub>	Port P7 direction register	PD7	00 <sub>16</sub>
03F0 <sub>16</sub>	Port P8 register	P8	?? <sub>16</sub>
03F1 <sub>16</sub>	Port P9 register	P9	?? <sub>16</sub>
03F2 <sub>16</sub>	Port P8 direction register	PD8	00X00000 <sub>2</sub>
03F3 <sub>16</sub>	Port P9 direction register	PD9	00 <sub>16</sub>
03F4 <sub>16</sub>			
03F5 <sub>16</sub>			
03F6 <sub>16</sub>			
03F7 <sub>16</sub>			
03F8 <sub>16</sub>			
03F9 <sub>16</sub>			
03FA <sub>16</sub>			
03FB <sub>16</sub>			
03FC <sub>16</sub>	Pull-up control register 0	PUR0	00 <sub>16</sub>
03FD <sub>16</sub>	Pull-up control register 1	PUR1	0000000 <sub>2</sub>
03FE <sub>16</sub>	Pull-up control register 2	PUR2	00 <sub>16</sub>
03FF <sub>16</sub>	Port control register	PCR	00 <sub>16</sub>

Note 1: The blank areas are reserved and cannot be used by users.

X : Nothing is mapped to this bit

? : Undefined

# Reset

There are four types of resets: a hardware reset, a software reset, an watchdog timer reset, and an oscillation stop detection reset.

## Hardware Reset

A reset is applied using the  $\overline{\text{RESET}}$  pin. When an “L” signal is applied to the  $\overline{\text{RESET}}$  pin while the power supply voltage is within the recommended operating condition, the pins are initialized (see Table 1.5.1). The oscillation circuit is initialized and the main clock starts oscillating. When the input level at the  $\overline{\text{RESET}}$  pin is released from “L” to “H”, the CPU and SFR are initialized, and the program is executed starting from the address indicated by the reset vector. The internal RAM is not initialized. If the  $\overline{\text{RESET}}$  pin is pulled “L” while writing to the internal RAM, the internal RAM becomes indeterminate.

Figure 1.5.1 shows the example reset circuit. Figure 1.5.2 shows the reset sequence. Table 1.5.1 shows the statuses of the other pins while the  $\overline{\text{RESET}}$  pin is “L”. Figure 1.5.3 shows the CPU register status after reset. Refer to “SFR” for SFR status after reset.

1. When the power supply is stable

- (1) Apply an “L” signal to the  $\overline{\text{RESET}}$  pin.
- (2) Supply a clock for 20 cycles or more to the XIN pin.
- (3) Apply an “H” signal to the  $\overline{\text{RESET}}$  pin.

2. Power on

- (1) Apply an “L” signal to the  $\overline{\text{RESET}}$  pin.
- (2) Let the power supply voltage increase until it meets the recommended operating condition.
- (3) Wait  $t_d(\text{P-R})$  or more until the internal power supply stabilizes.
- (4) Supply a clock for 20 cycles or more to the XIN pin.
- (5) Apply an “H” signal to the  $\overline{\text{RESET}}$  pin.

## Software Reset

When the PM03 bit in the PM0 register is set to “1” (microcomputer reset), the microcomputer has its pins, CPU, and SFR initialized. Then the program is executed starting from the address indicated by the reset vector.

Select the main clock for the CPU clock source, and set the PM03 bit to “1” with main clock oscillation satisfactorily stable.

At software reset, some SFR’s are not initialized. Refer to “SFR”. Also, since the PM01 to PM00 bits in the PM0 register are not initialized, the processor mode remains unchanged.

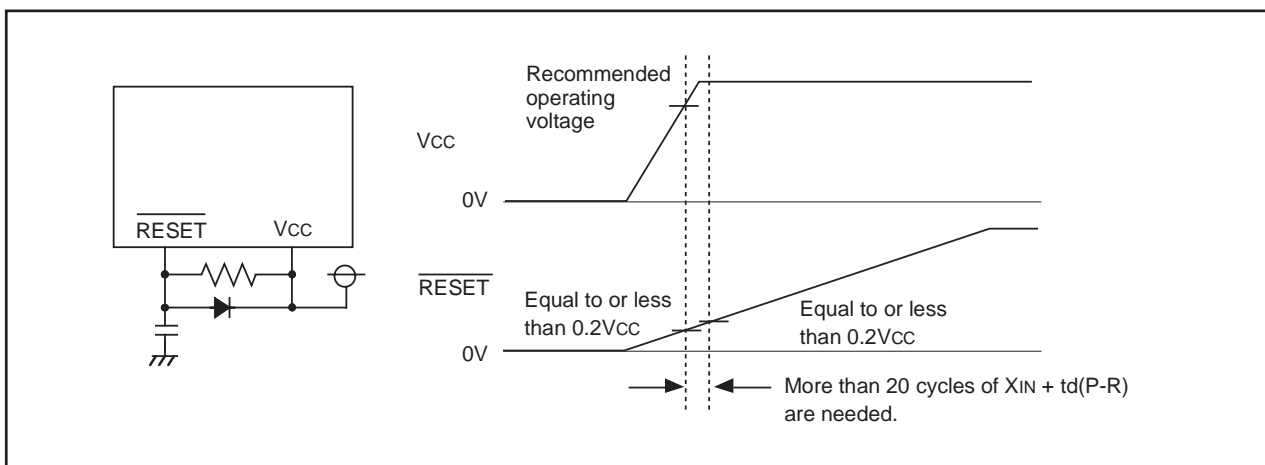


Figure 1.5.1. Example Reset Circuit

## Watchdog Timer Reset

Where the PM12 bit in the PM1 register is “1” (reset when watchdog timer underflows), the microcomputer initializes its pins, CPU and SFR if the watchdog timer underflows. Then the program is executed starting from the address indicated by the reset vector.

At watchdog timer reset, some SFR’s are not initialized. Refer to “SFR”. Also, since the PM01 to PM00 bits in the PM0 register are not initialized, the processor mode remains unchanged.

## Oscillation Stop Detection Reset

Where the CM27 bit in the CM2 register is “0” (reset at oscillation stop detection), the microcomputer initializes its pins, CPU and SFR, coming to a halt if it detects main clock oscillation circuit stop. Refer to the section “oscillation stop, re-oscillation detection function”.

At oscillation stop detection reset, some SFR’s are not initialized. Refer to the section “SFR”. Also, since the PM01 to PM00 bits in the PM0 register are not initialized, the processor mode remains unchanged.

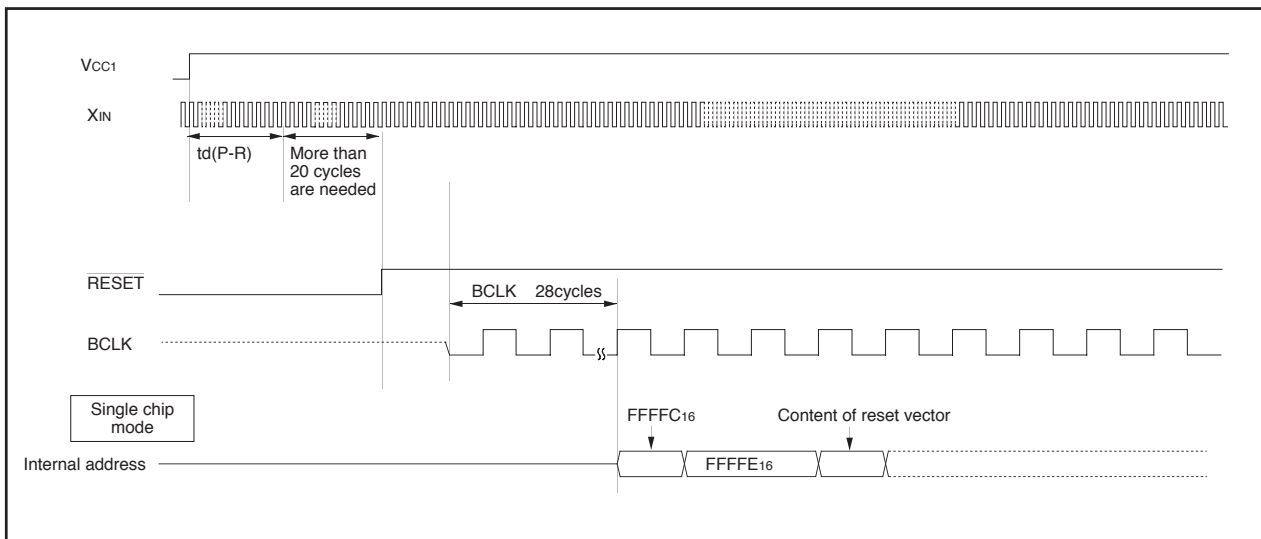
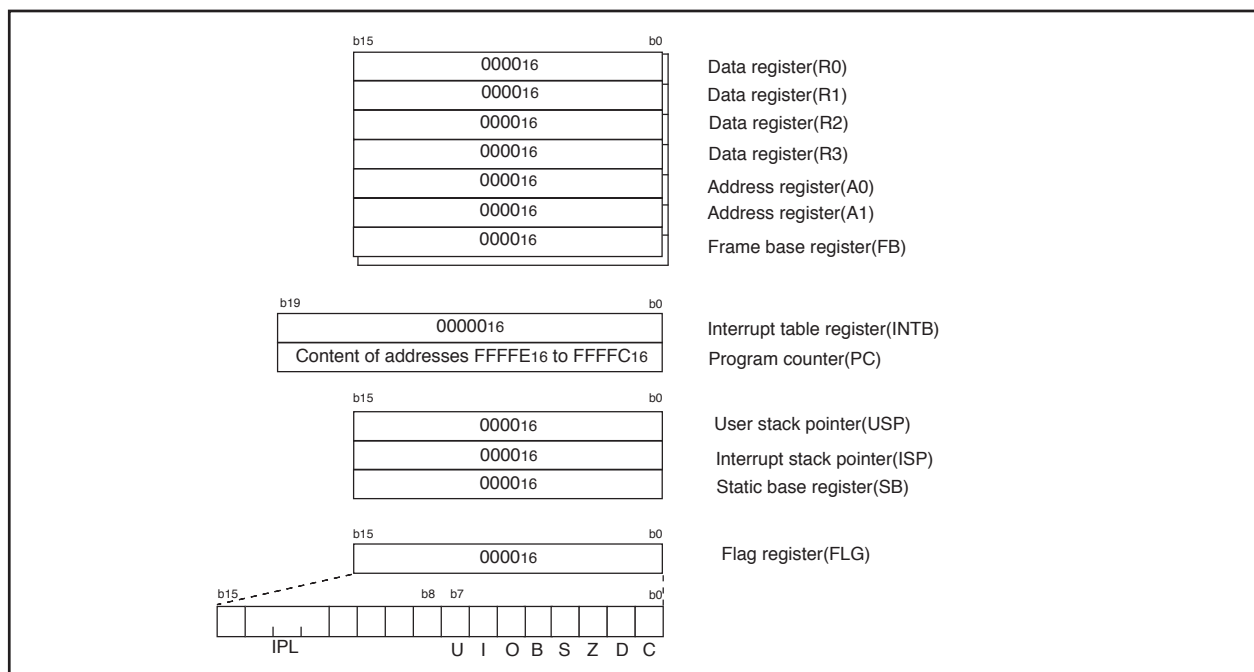


Figure 1.5.2. Reset Sequence

**Table 1.5.1. Pin Status When  $\overline{\text{RESET}}$  Pin Level is “L”**

Pin name	Status
	CNVss = Vss
P15, P60 to P67, P70, P71, P73, P74, P76, P80, P81, P83, P84, P85, P90 to P92, TS	Input port



**Figure 1.5.3. CPU Register Status After Reset**

## Processor Mode

### (1) Setting Processor Modes

Processor mode is available only: single-chip mode.

Processor mode is set by using the CNVss pin and the PM01 to PM00 bits in the PM0 register.

Table 1.6.1 shows the processor mode after hardware reset. Table 1.6.2 shows the PM01 to PM00 bits set values and processor modes. For setting Single-chip mode, CNVss should be kept Vss level. And PM01 to PM00 of PM0 register should be set "00."

**Table 1.6.1. Processor Mode After Hardware Reset**

CNVSS pin input level	Processor mode
Vss	Single-chip mode
Vcc	Flash Memory Mode

**Table 1.6.2. PM01 to PM00 Bits Set Values and Processor Modes**

PM01 to PM00 bits	Processor modes
002	Single-chip mode
012	Must not be set
102	
112	

Figure 1.6.4 show the memory map in single chip mode.

## (2) Setting PLC Mode

PLC mode is simply set by putting P15 High level during RESET.

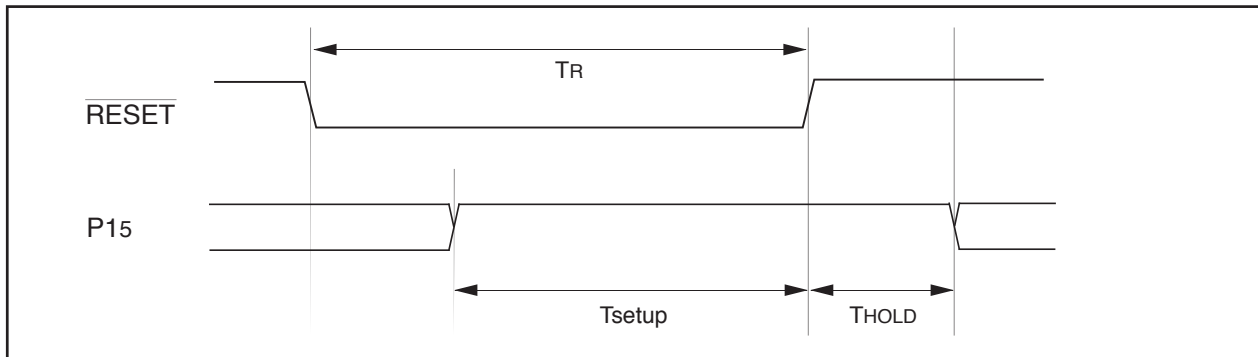


Figure 1.6.1. PLC mode by P15 simply setting

Table 1.6.3. RESET and P15 Input

	min	typ	max
TR	40us	—	—
Tset up	5us	—	—
THOLD	5us	—	—



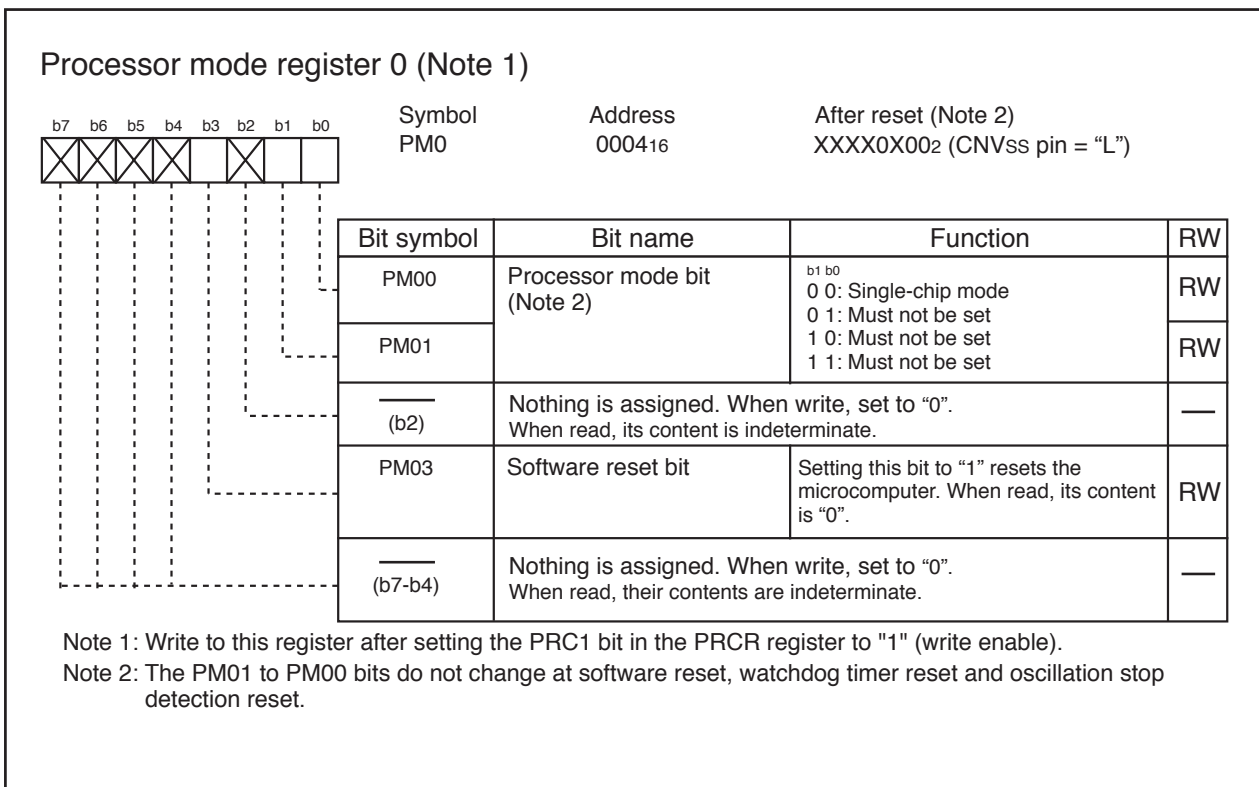


Figure 1.6.2. PM0 Register

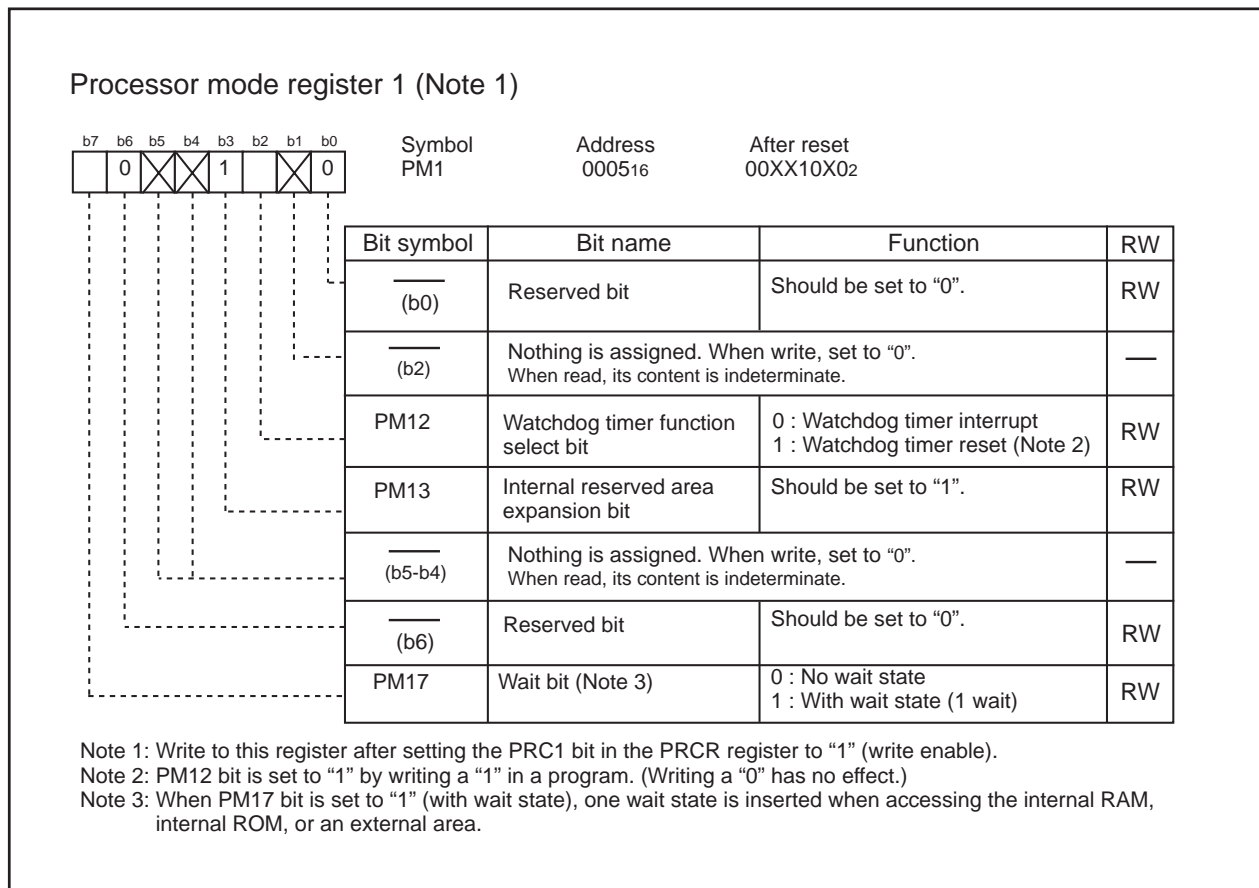
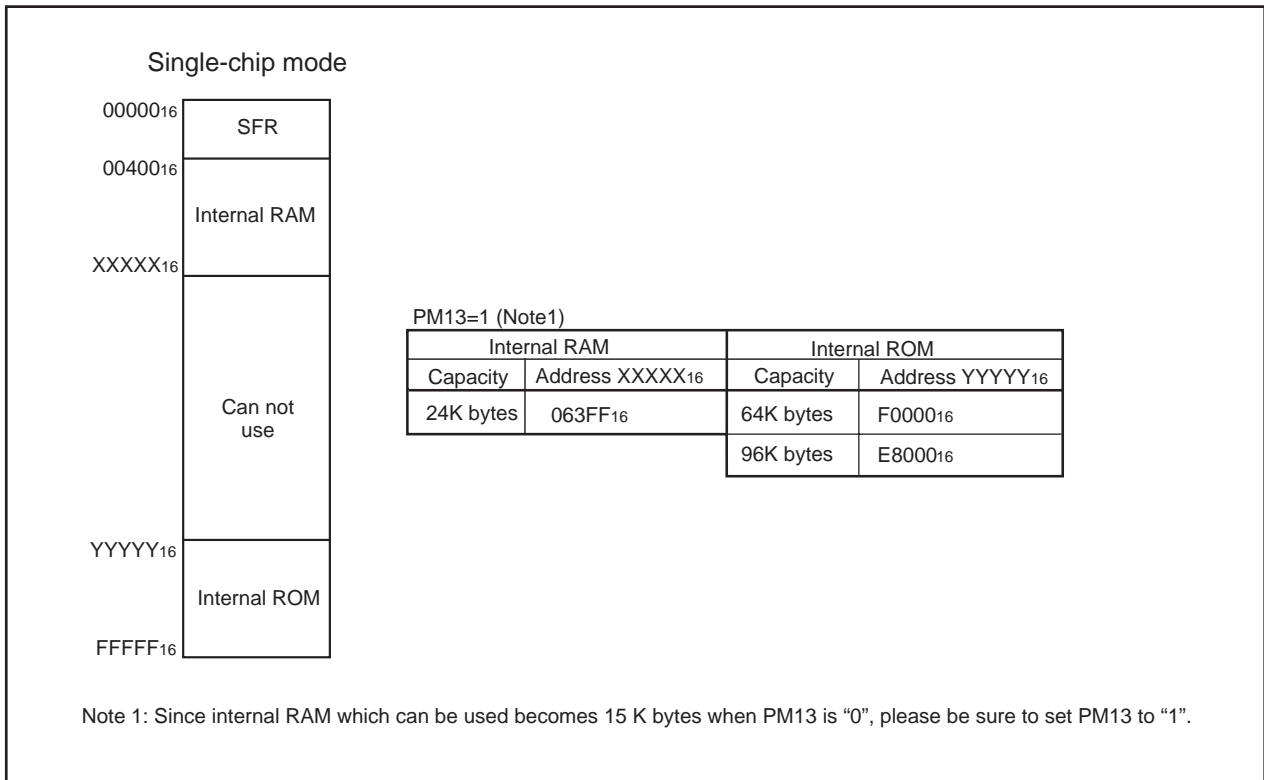


Figure 1.6.3. PM1 Register



**Figure 1.6.4. Memory Map in Single Chip Mode**

## Clock Generation Circuit

The clock generation circuit contains two oscillator circuits as follows:

- (1) Main clock oscillation circuit
- (2) On-chip Oscillator (oscillation stop detect function)

Table 1.7.1 lists the clock generation circuit specifications. Figure 1.7.1 shows the clock generation circuit. Figures 1.7.2 to 1.7.5 show the clock-related registers.

**Table 1.7.1. Clock Generation Circuit Specifications**

Item	Main clock oscillation circuit	On-chip Oscillator
Use of clock	<ul style="list-style-type: none"> <li>• CPU clock source</li> <li>• Peripheral function clock source</li> </ul>	<ul style="list-style-type: none"> <li>• CPU clock source</li> <li>• Peripheral function clock source</li> </ul>
Clock frequency	5.12 MHz	About 1 MHz
Usable oscillator	<ul style="list-style-type: none"> <li>• Crystal oscillator (Note 1)</li> </ul>	_____
Pins to connect oscillator	XIN, XOUT	_____
Oscillation stop, restart function	No	Presence
Oscillator status after reset	Oscillating	Stopped
Other	Externally derived clock can be input	_____

Note. Operating frequency must be 5.12 MHz, overall accuracy must be less than 150 ppm.

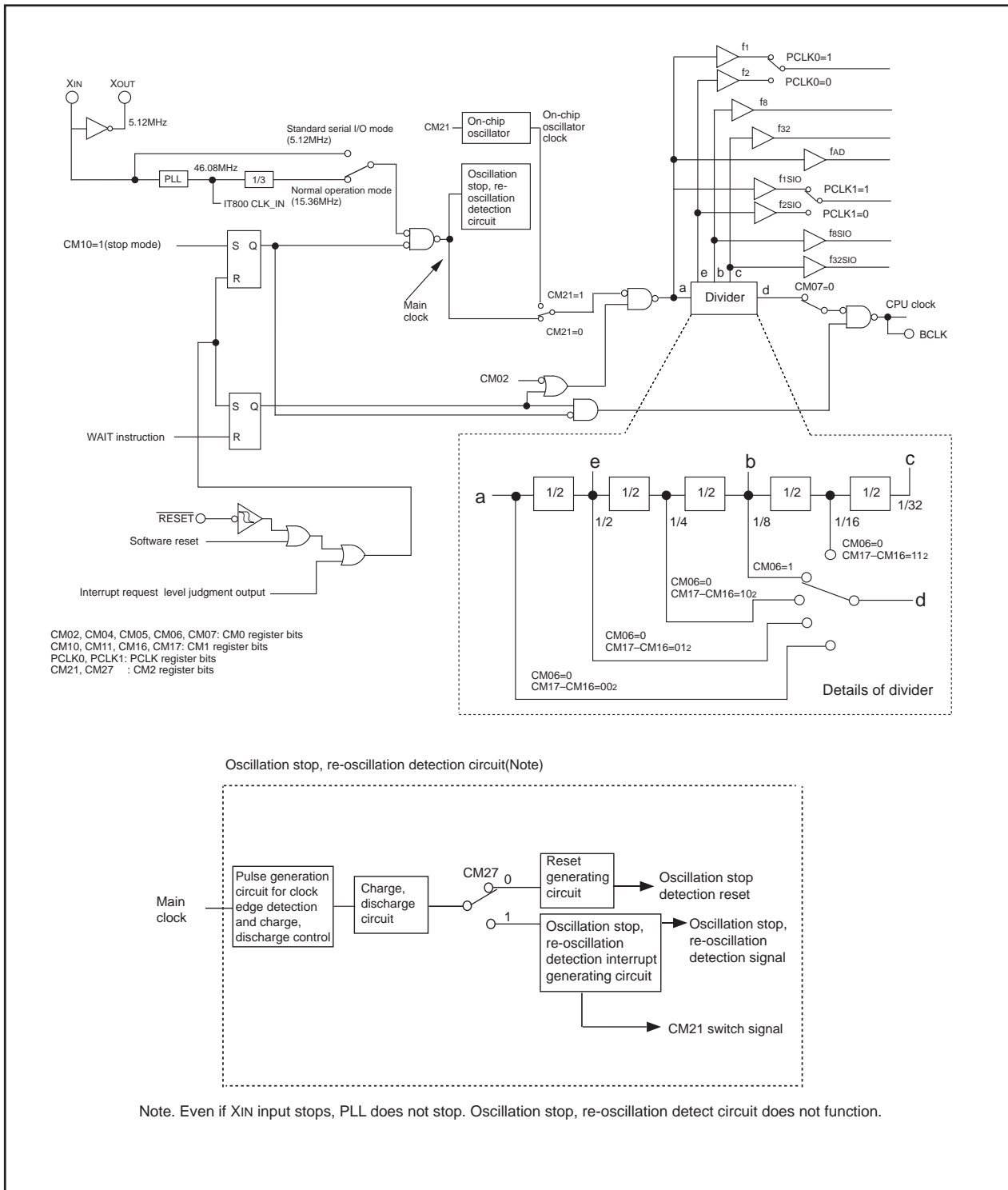


Figure 1.7.1. Clock Generation Circuit

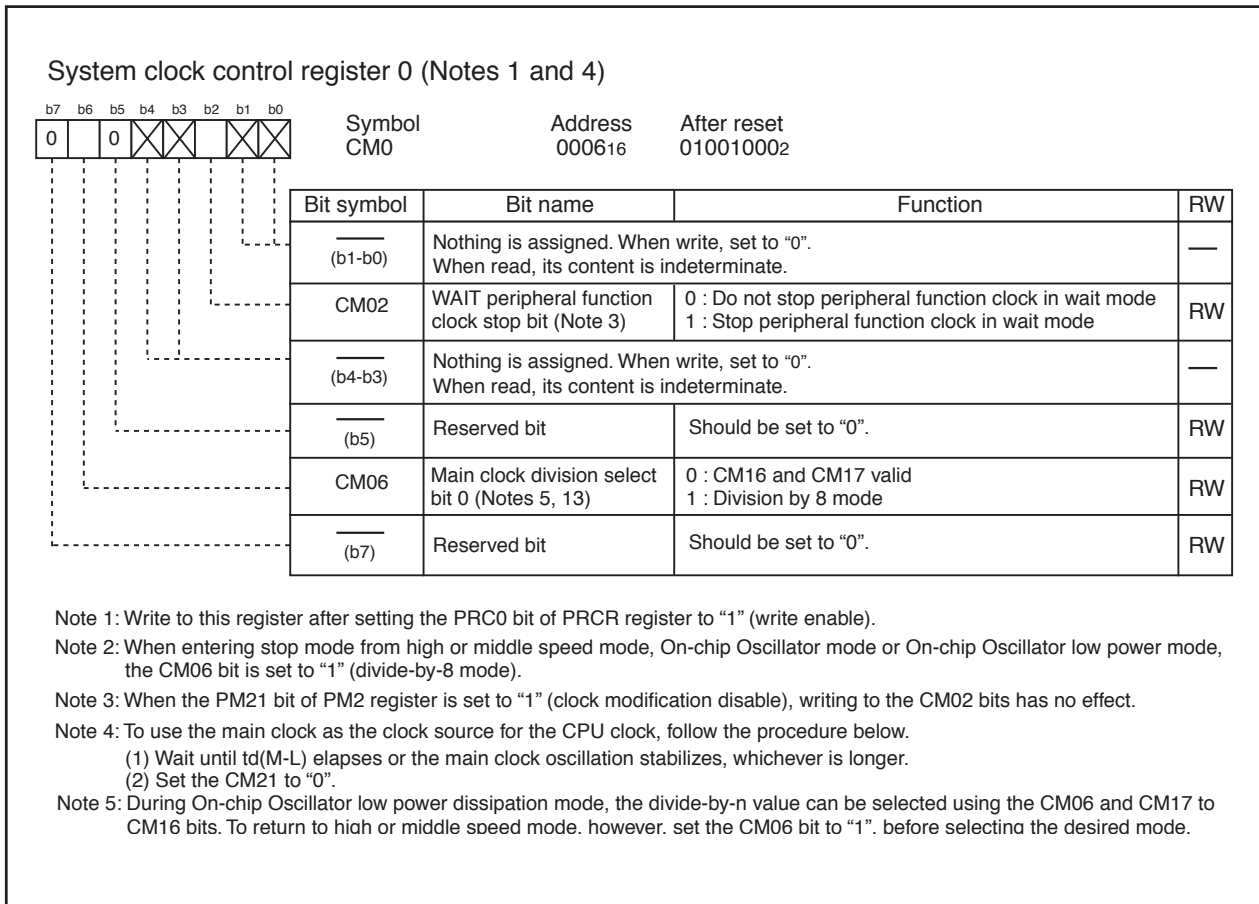


Figure 1.7.2. CM0 Register

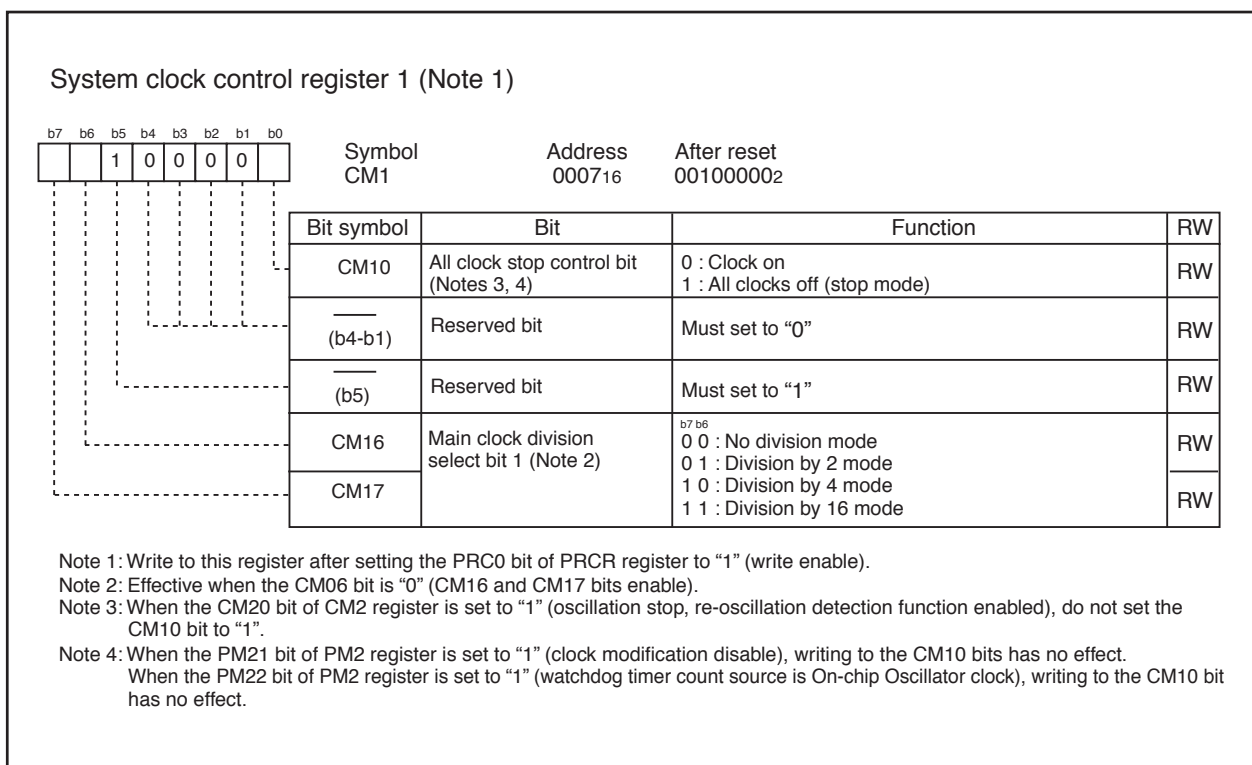


Figure 1.7.3. CM1 Register

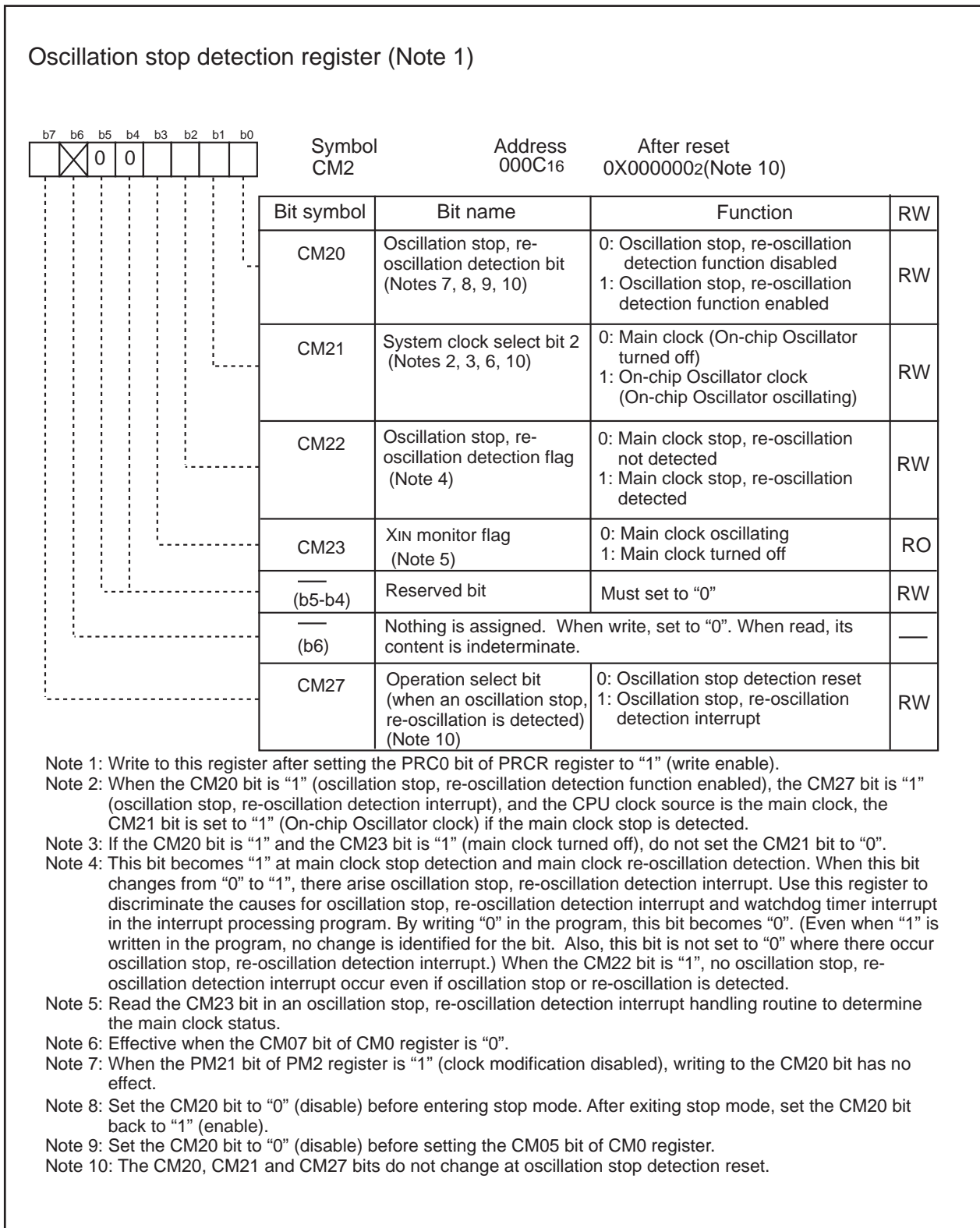


Figure 1.7.4. CM2 Register

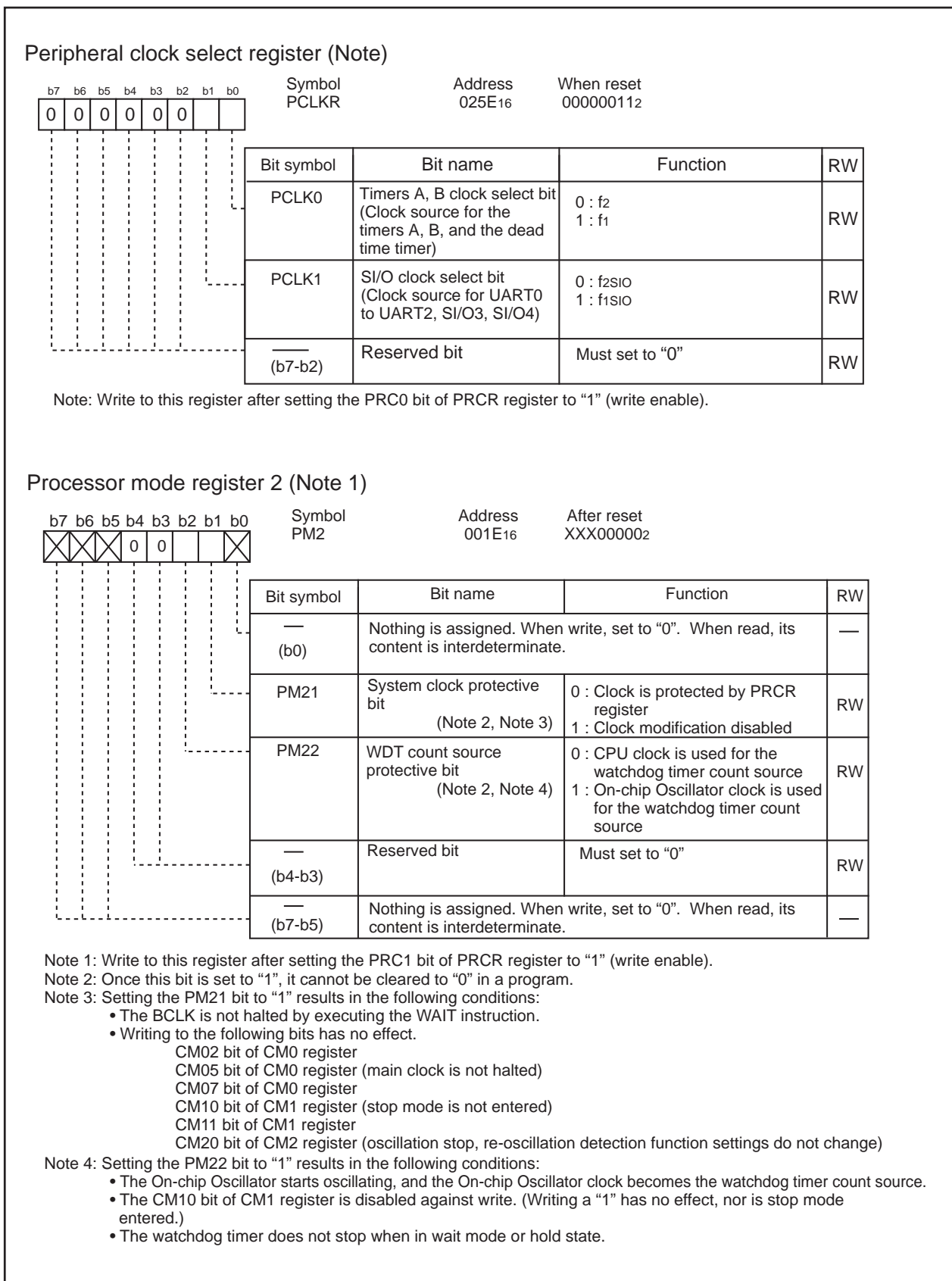


Figure 1.7.5. PCLKR Register and PM2 Register



The following describes the clocks generated by the clock generation circuit.

### (1) Main Clock

Main clock is supplied by IT800 with a tripled clock of XIN (main clock oscillator).

This clock is used as the clock source for the CPU and peripheral function clocks. The main clock oscillator circuit is configured by connecting a resonator between the XIN and XOUT pins. The main clock oscillator circuit contains a feedback resistor. The main clock oscillator circuit may also be configured by feeding an externally generated clock to the XIN pin. Figure 1.7.6 shows the examples of main clock connection circuit.

After reset, the main clock divided by 8 is selected for the CPU clock.

Even if XIN input stops, main clock oscillator does not stop.

During stop mode, all clocks of internal M16C core including the main clock are turned off. Refer to "power control".

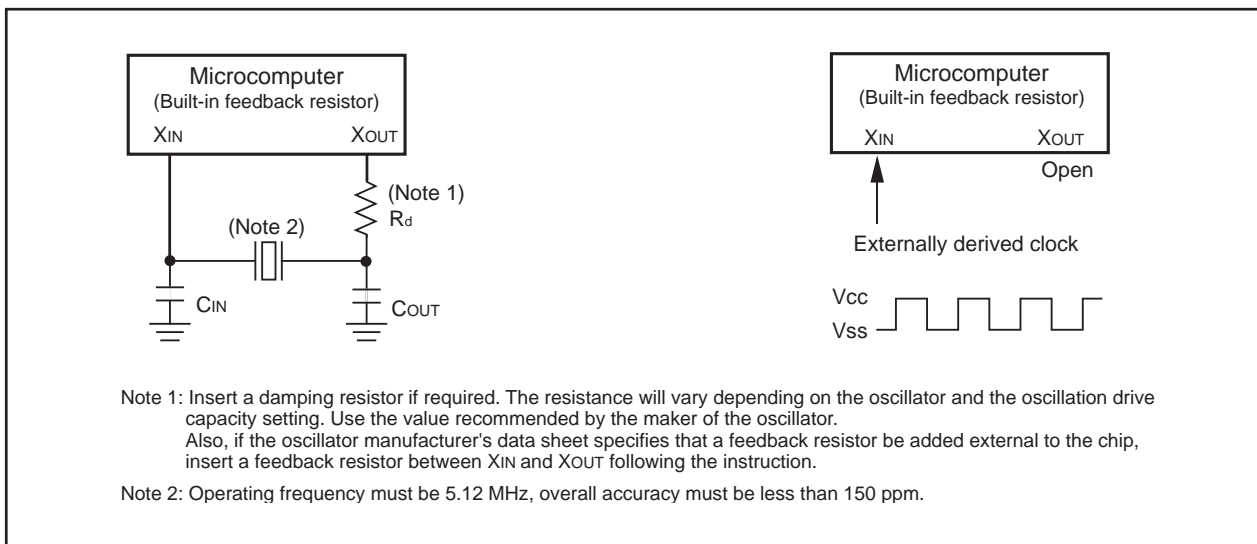


Figure 1.7.6. Examples of Main Clock Connection Circuit

### (3) On-chip Oscillator Clock

This clock, approximately 1 MHz, is supplied by a On-chip Oscillator. This clock is used as the clock source for the CPU and peripheral function clocks. In addition, if the PM22 bit of PM2 register is “1” (On-chip Oscillator clock for the watchdog timer count source), this clock is used as the count source for the watchdog timer.

After reset, the On-chip Oscillator clock is turned off. It is turned on by setting the CM21 bit of CM2 register to “1” (On-chip Oscillator clock), and is used as the clock source for the CPU and peripheral function clocks, in place of the main clock.

## CPU Clock and Peripheral Function Clock

Two type clocks: CPU clock to operate the CPU and peripheral function clocks to operate the peripheral functions.

### (1) CPU Clock and BCLK

These are operating clocks for the CPU and watchdog timer.

The clock source for the CPU clock can be chosen to be the main clock, or On-chip Oscillator clock.

If the main clock or On-chip Oscillator clock is selected as the clock source for the CPU clock, the selected clock source can be divided by 1 (undivided), 2, 4, 8 or 16 to produce the CPU clock. Use the CM06 bit in CM0 register and the CM17 to CM16 bits in CM1 register to select the divide-by-n value.

After reset, the main clock divided by 8 provides the CPU clock.

Note that when entering stop mode from high or middle speed mode or On-chip Oscillator mode, the CM06 bit of CM0 register is set to "1" (divide-by-8 mode).

### (2) Peripheral Function Clock(f1, f2, f8, f32, f1SIO, f2SIO, f8SIO, f32SIO)

These are operating clocks for the peripheral functions.

Of these,  $f_i$  ( $i = 1, 2, 8, 32$ ) and  $f_{iSIO}$  are derived from the main clock, or On-chip Oscillator clock by dividing them by  $i$ . The clock  $f_i$  is used for timers A, and  $f_{iSIO}$  is used for serial I/O.

When the WAIT instruction is executed after setting the CM02 bit of CM0 register to "1" (peripheral function clock turned off during wait mode), or when the microcomputer is in low power dissipation mode, the  $f_i$ , and  $f_{iSIO}$  clocks are turned off.

## Power Control

There are three power control modes. For convenience' sake, all modes other than wait and stop modes are referred to as normal operation mode here.

### (1) Normal Operation Mode

Normal operation mode is further classified into three modes.

In normal operation mode, because the CPU clock and the peripheral function clocks both are on, the CPU and the peripheral functions are operating. Power control is exercised by controlling the CPU clock frequency. The higher the CPU clock frequency, the greater the processing capability. The lower the CPU clock frequency, the smaller the power consumption in the chip. If the unnecessary oscillator circuits are turned off, the power consumption is further reduced.

Before the clock sources for the CPU clock can be switched over, the new clock source to which switched must be oscillating stably. If the new clock source is the main clock, allow a sufficient wait time in a program until it becomes oscillating stably.

Where the CPU clock source is changed from the On-chip Oscillator to the main clock, change the operation mode to the medium speed mode (divided by 8 mode) after the clock was divided by 8 (the CM06 bit of CM0 register was set to "1") in the On-chip Oscillator mode.

- **High-speed Mode**

The main clock divided by 1 provides the CPU clock.

- **Medium-speed Mode**

The main clock divided by 2, 4, 8 or 16 provides the CPU clock.

- **On-chip Oscillator Mode**

The On-chip Oscillator clock divided by 1 (undivided), 2, 4, 8 or 16 provides the CPU clock. The On-chip Oscillator clock is also the clock source for the peripheral function clocks.

**Table 1.7.2. Setting Clock Related Bit and Modes**

Modes		CM2 register	CM1 register	CM0 register
		CM21	CM17, CM16	CM06
High-speed mode		0	002	0
Medium-speed mode	divided by 2	0	012	0
	divided by 4	0	102	0
	divided by 8	0	—	1
	divided by 16	0	112	0
On-chip oscillator mode	divided by 1	1	002	0
	divided by 2	1	012	0
	divided by 4	1	102	0
	divided by 8	1	—	1
	divided by 16	1	112	0

Note: The divide-by-n value can be selected the same way as in On-chip Oscillator mode.

## (2) Wait Mode

In wait mode, the CPU clock is turned off, so are the CPU (because operated by the CPU clock) and the watchdog timer. However, if the PM22 bit of PM2 register is “1” (On-chip Oscillator clock for the watchdog timer count source), the watchdog timer remains active. Because the main clock, and On-chip Oscillator clock are on, the peripheral functions using these clocks keep operating.

### • Peripheral Function Clock Stop Function

If the CM02 bit is “1” (peripheral function clocks turned off during wait mode), the f1, f2, f8, f32, f1SIO, f8SIO, and f32SIO clocks are turned off when in wait mode, with the power consumption reduced that much.

### • Entering Wait Mode

The microcomputer is placed into wait mode by executing the WAIT instruction.

### • Pin Status During Wait Mode

Table 1.7.3 lists pin status during wait mode

### • Exiting Wait Mode

The microcomputer is moved out of wait mode by a hardware reset or peripheral function interrupt.

If the microcomputer is to be moved out of exit wait mode by a hardware reset, set the peripheral function interrupt priority ILVL2 to ILVL0 bits to “0002” (interrupts disabled) before executing the WAIT instruction.

The peripheral function interrupts are affected by the CM02 bit. If CM02 bit is “0” (peripheral function clocks not turned off during wait mode), all peripheral function interrupts can be used to exit wait mode. If CM02 bit is “1” (peripheral function clocks turned off during wait mode), the peripheral functions using the peripheral function clocks stop operating, so that only the peripheral functions clocked by external signals can be used to exit wait mode.

Table 1.7.4 lists the interrupts to exit wait mode.

**Table 1.7.3. Pin Status During Wait Mode**

Pin	Status
I/O ports	Retains status before wait mode

**Table 1.7.4. Interrupts to Exit Wait Mode**

Interrupt	CM02=0	CM02=1
Serial I/O interrupt	Can be used when operating with internal or external clock	Can be used when operating with external clock
Timer A interrupt	Can be used in all modes	Can be used in event counter mode
$\overline{\text{INT}}$ interrupt	Can be used	Can be used

If the microcomputer is to be moved out of wait mode by a peripheral function interrupt, set up the following before executing the WAIT instruction.

1. In the ILVL2 to ILVL0 bits of interrupt control register, set the interrupt priority level of the peripheral function interrupt to be used to exit wait mode.  
Also, for all of the peripheral function interrupts not used to exit wait mode, set the ILVL2 to ILVL0 bits to "0002" (interrupt disable).
2. Set the I flag to "1".
3. Enable the peripheral function whose interrupt is to be used to exit wait mode.  
In this case, when an interrupt request is generated and the CPU clock is thereby turned on, an interrupt routine is executed.

The CPU clock turned on when exiting wait mode by a peripheral function interrupt is the same CPU clock that was on when the WAIT instruction was executed.

### (3) Stop Mode

In stop mode, all the M16C core's internal oscillator circuits are turned off, so are the CPU clock and the peripheral function clocks. Therefore, the CPU and the peripheral functions clocked by these clocks stop operating.

However, the peripheral functions clocked by external signals keep operating. The following interrupts can be used to exit stop mode.

- $\overline{\text{INT}}$  interrupt
- Timer A, interrupt (when counting external pulses in event counter mode)
- Serial I/O interrupt (when external clock is selected)

#### • Entering Stop Mode

The microcomputer is placed into stop mode by setting the CM10 bit of CM1 register to "1" (all clocks turned off). At the same time, the CM06 bit of CM0 register is set to "1" (divide-by-8 mode).

Before entering stop mode, set the CM20 bit to "0" (oscillation stop, re-oscillation detection function disable).

#### • Pin Status in Stop Mode

Table 1.7.5 lists pin status during stop mode

#### • Exiting Stop Mode

The microcomputer is moved out of stop mode by a hardware reset or peripheral function interrupt.

If the microcomputer is to be moved out of stop mode by a hardware reset, set the peripheral function interrupt priority ILVL2 to ILVL0 bits to "0002" (interrupts disable) before setting the CM10 bit to "1".

If the microcomputer is to be moved out of stop mode by a peripheral function interrupt, set up the following before setting the CM10 bit to "1".

1. In the ILVL2 to ILVL0 bits of interrupt control register, set the interrupt priority level of the peripheral function interrupt to be used to exit stop mode.

Also, for all of the peripheral function interrupts not used to exit stop mode, set the ILVL2 to ILVL0 bits to "0002".

2. Set the I flag to "1".

3. Enable the peripheral function whose interrupt is to be used to exit stop mode.

In this case, when an interrupt request is generated and the CPU clock is thereby turned on, an interrupt service routine is executed.

Which CPU clock will be used after exiting stop mode by a peripheral function is determined by the CPU clock that was on when the microcomputer was placed into stop mode as follows:

If the CPU clock before entering stop mode was derived from the main clock: main clock divide-by-8

If the CPU clock before entering stop mode was derived from the On-chip Oscillator clock: On-chip Oscillator clock divide-by-8

**Table 1.7.5. Pin Status in Stop Mode**

Pin	Status
I/O ports	Retains status before stop mode



Figure 1.7.7 shows the state transition from normal operation mode to stop mode and wait mode.

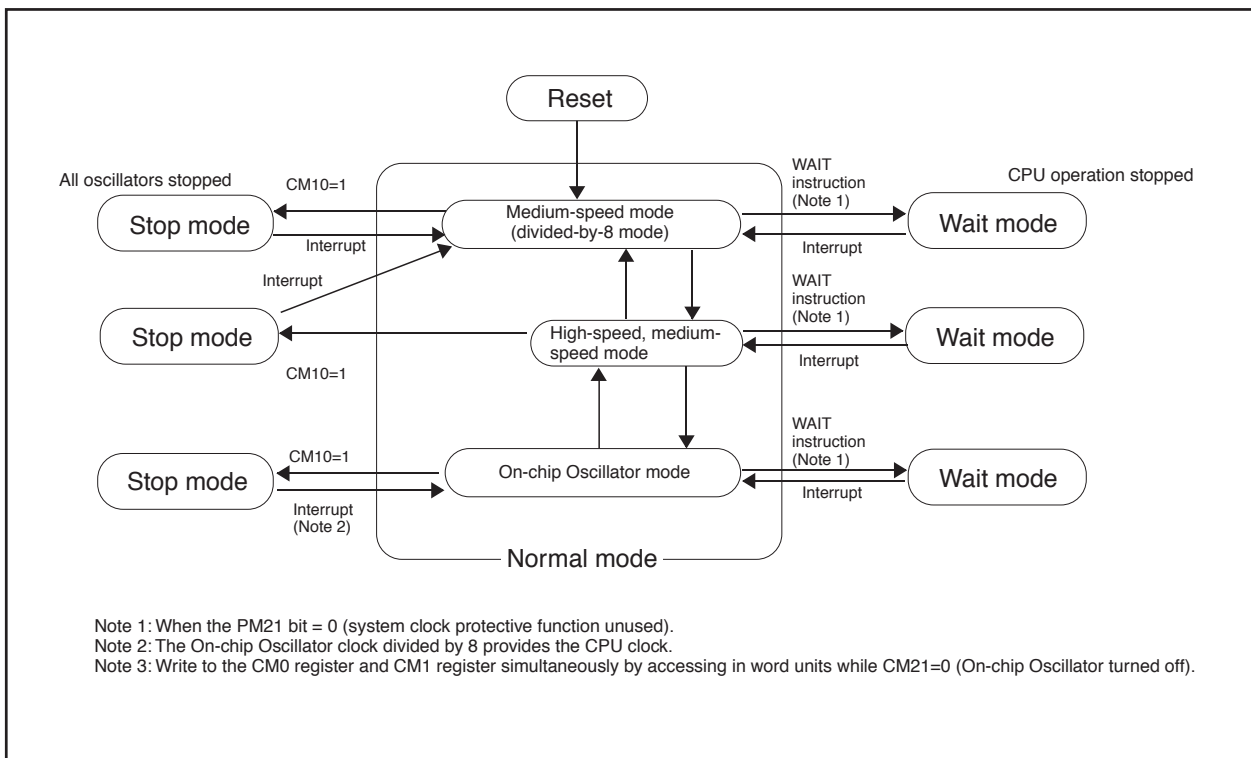


Figure 1.7.7. State Transition to Stop Mode and Wait Mode

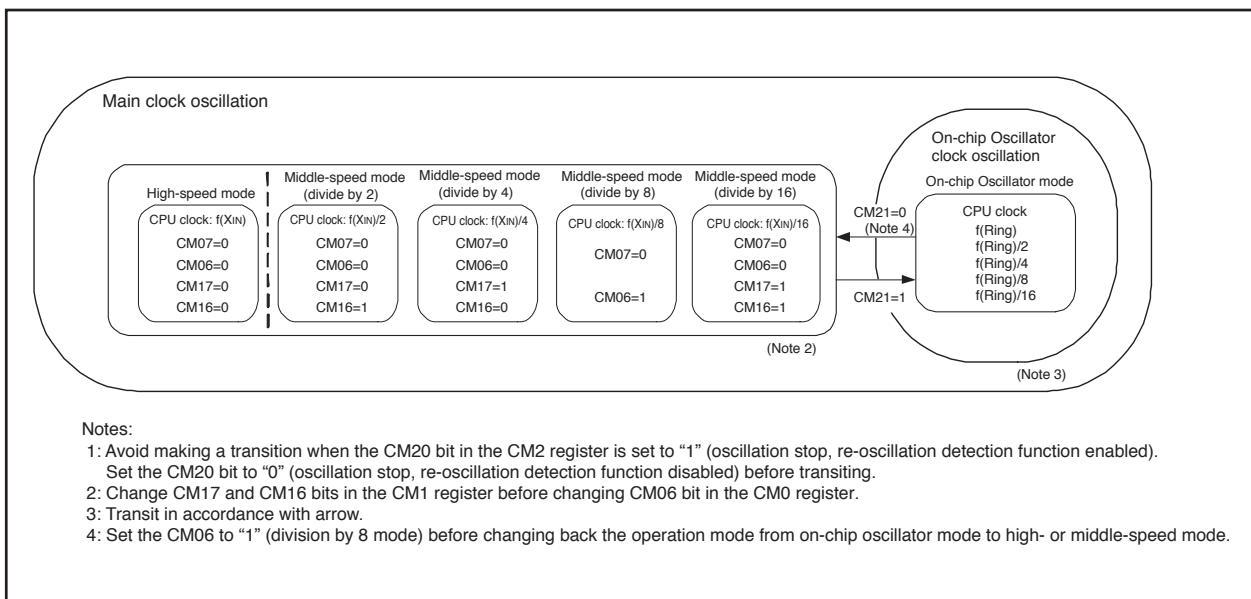


Figure 1.7.8. State Transition in Normal Mode

**Table 1.7.6. Allowed Transition and Setting**

		State after transition			
		High-speed mode, middle-speed mode	On-chip Oscillator mode	Stop mode	Wait mode
Current state	High-speed mode, middle-speed mode	See Table A	(7)	(8) <sup>1</sup>	(9)
	On-chip Oscillator mode	(6) <sup>2</sup>	See Table A	(8) <sup>1</sup>	(9)
	Stop mode	(10) <sup>3</sup>	(10) <sup>3</sup>	/	--
	Wait mode	(10)	(10)		--

--: Cannot transit

Table 1. State Transition with Main Clock Division Ratio in High- or Middle-speed Mode and On-chip Oscillator Mode.

		State after transition				
		No division	Divided by 2	Divided by 4	Divided by 8	Divided by 16
Current state	No division	/	(2)	(3)	(5)	(4)
	Divided by 2		(1)	(3)	(5)	(4)
	Divided by 4		(1)	(2)	(5)	(4)
	Divided by 8		(1)	(2)	(3)	(4)
	Divided by 16		(1)	(2)	(3)	(5)

Notes:

- Avoid making a transition when the CM21 bit is set to "1" (oscillation stop, re-oscillation detection function enabled). Set the CM21 bit to "0" (oscillation stop, re-oscillation detection function disabled) before transiting.
- Set the CM06 bit to "1" (division by 8 mode) before transiting from On-chip Oscillator mode to high- or middle-speed mode.
- When exiting stop mode, the CM06 bit is set to "1" (division by 8 mode).

Table B. Setting and Operation

	Setting	Operation
(1)	CM06 = 0, CM17 = 0, CM16 = 0	CPU clock no division mode
(2)	CM06 = 0, CM17 = 0, CM16 = 1	CPU clock division by 2 mode
(3)	CM06 = 0, CM17 = 1, CM16 = 0	CPU clock division by 4 mode
(4)	CM06 = 0, CM17 = 1, CM16 = 1	CPU clock division by 16 mode
(5)	CM06 = 1	CPU clock division by 8 mode
(6)	CM21 = 0	Main clock
(7)	CM21 = 1	On-chip Oscillator clock selected
(8)	CM10 = 1	Transition to stop mode
(9)	wait	Transition to wait mode
(10)	Hardware interrupt	Exit stop mode or wait mode

## System Clock Protective Function

When the main clock is selected for the CPU clock source, this function disables the clock against modifications in order to prevent the CPU clock from becoming halted by run-away.

If the PM21 bit of PM2 register is set to "1" (clock modification disabled), the following bits are protected against writes:

- CM02 bit in CM0 register
- CM10, CM11 bits in CM1 register
- CM20 bit in CM2 register

Before the system clock protective function can be used, the following register settings must be made:

- (1) Set the PRC1 bit of PRCR register to "1" (enable writes to PM2 register).
- (2) Set the PM21 bit of PM2 register to "1" (disable clock modification).
- (3) Set the PRC1 bit of PRCR register to "0" (disable writes to PM2 register).

Do not execute the WAIT instruction when the PM21 bit is "1".

### **Oscillation Stop and Re-oscillation Detect Function**

The oscillation stop and re-oscillation detect function is such that main clock oscillation circuit stop and re-oscillation are detected. At oscillation stop, re-oscillation detection, reset or oscillation stop, re-oscillation detection interrupt are generated. Which is to be generated can be selected using the CM27 bit of CM2 register. Main clock oscillator of M16C/6S does not stop even if Xin input stops. Oscillation stop re-oscillation detect circuit does not function.

## Protection

In the event that a program runs out of control, this function protects the important registers so that they will not be rewritten easily. Figure 1.8.1 shows the PRCR register. The following lists the registers protected by the PRCR register.

- Registers protected by PRC0 bit: CM0, CM1, CM2, and PCLKR registers
- Registers protected by PRC1 bit: PM0, PM1, PM2 registers
- Registers protected by PRC2 bit: PD9, S3C and S4C registers

Set the PRC2 bit to “1” (write enabled) and then write to any address, and the PRC2 bit will be cleared to “0” (write protected). The registers protected by the PRC2 bit should be changed in the next instruction after setting the PRC2 bit to “1”. Make sure no interrupts or DMA transfers will occur between the instruction in which the PRC2 bit is set to “1” and the next instruction. The PRC0 and PRC1 bits are not automatically cleared to “0” by writing to any address. They can only be cleared in a program.

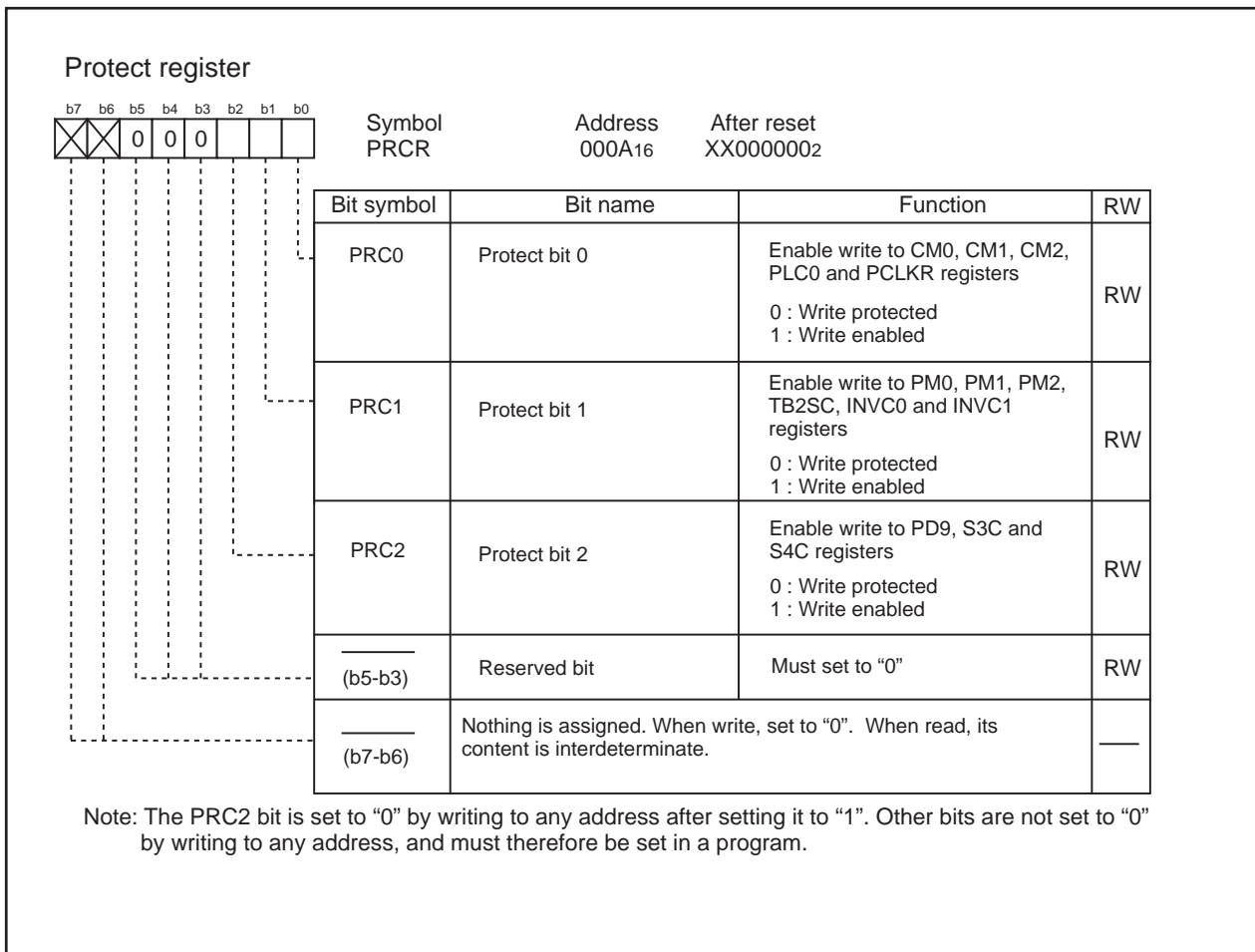
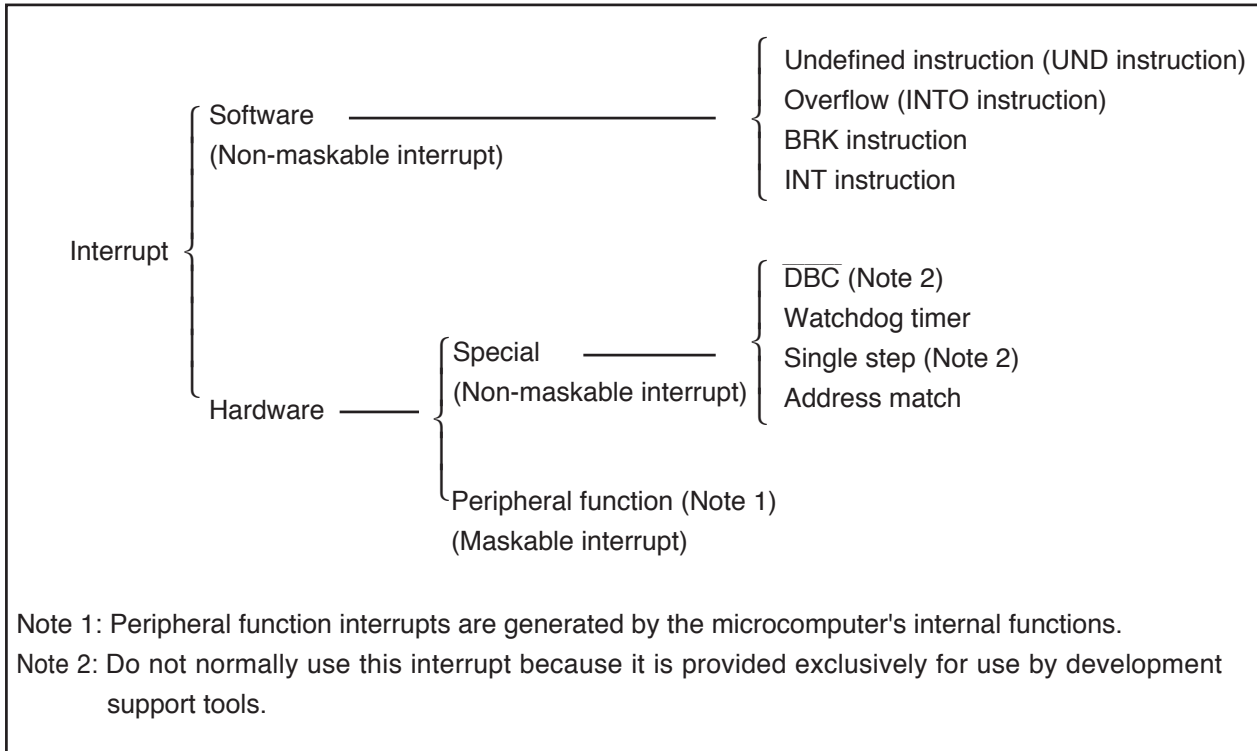


Figure 1.8.1. PRCR Register

# Interrupts

## Type of Interrupts

Figure 1.9.1 shows types of interrupts.



**Figure 1.9.1. Interrupts**

- Maskable Interrupt: An interrupt which can be enabled (disabled) by the interrupt enable flag (I flag) or whose interrupt priority **can be changed** by priority level.
- Non-maskable Interrupt: An interrupt which cannot be enabled (disabled) by the interrupt enable flag (I flag) or whose interrupt priority **cannot be changed** by priority level.

## Software Interrupts

A software interrupt occurs when executing certain instructions. Software interrupts are non-maskable interrupts.

- **Undefined Instruction Interrupt**

An undefined instruction interrupt occurs when executing the UND instruction.

- **Overflow Interrupt**

An overflow interrupt occurs when executing the INTO instruction with the O flag set to “1” (the operation resulted in an overflow). The following are instructions whose O flag changes by arithmetic:

ABS, ADC, ADCF, ADD, CMP, DIV, DIVU, DIVX, NEG, RMPA, SBB, SHA, SUB

- **BRK Interrupt**

A BRK interrupt occurs when executing the BRK instruction.

- **INT Instruction Interrupt**

An INT instruction interrupt occurs when executing the INT instruction. Software interrupt Nos. 0 to 63 can be specified for the INT instruction. Because software interrupt Nos. 4 to 31 are assigned to peripheral function interrupts, the same interrupt routine as for peripheral function interrupts can be executed by executing the INT instruction.

In software interrupt Nos. 0 to 31, the U flag is saved to the stack during instruction execution and is cleared to “0” (ISP selected) before executing an interrupt sequence. The U flag is restored from the stack when returning from the interrupt routine. In software interrupt Nos. 32 to 63, the U flag does not change state during instruction execution, and the SP then selected is used.



## Hardware Interrupts

Hardware interrupts are classified into two types — special interrupts and peripheral function interrupts.

### (1) Special Interrupts

Special interrupts are non-maskable interrupts.

- **DBC Interrupt**

Do not normally use this interrupt because it is provided exclusively for use by development support tools.

- **Watchdog Timer Interrupt**

Generated by the watchdog timer. Once a watchdog timer interrupt is generated, be sure to initialize the watchdog timer. For details about the watchdog timer, refer to the section "watchdog timer".

- **Oscillation Stop and Re-oscillation Detection Interrupt**

Generated by the oscillation stop and re-oscillation detection function. For details about the oscillation stop detection function, refer to the section "clock generating circuit".

- **Single-step Interrupt**

Do not normally use this interrupt because it is provided exclusively for use by development support tools.

- **Address Match Interrupt**

An address match interrupt is generated immediately before executing the instruction at the address indicated by the RMAD0 to RMAD3 register that corresponds to one of the AIER register's AIER0 or AIER1 bit or the AIER2 register's AIER20 or AIER21 bit which is "1" (address match interrupt enabled). For details about the address match interrupt, refer to the section "address match interrupt".

### (2) Peripheral Function Interrupts

Peripheral function interrupts are maskable interrupts and generated by the microcomputer's internal functions. The interrupt sources for peripheral function interrupts are listed in Table 1.9.1. For details about the peripheral functions, refer to the description of each peripheral function in this manual.

## Interrupts and Interrupt Vector

One interrupt vector consists of 4 bytes. Set the start address of each interrupt routine in the respective interrupt vectors. When an interrupt request is accepted, the CPU branches to the address set in the corresponding interrupt vector. Figure 1.9.2 shows the interrupt vector.

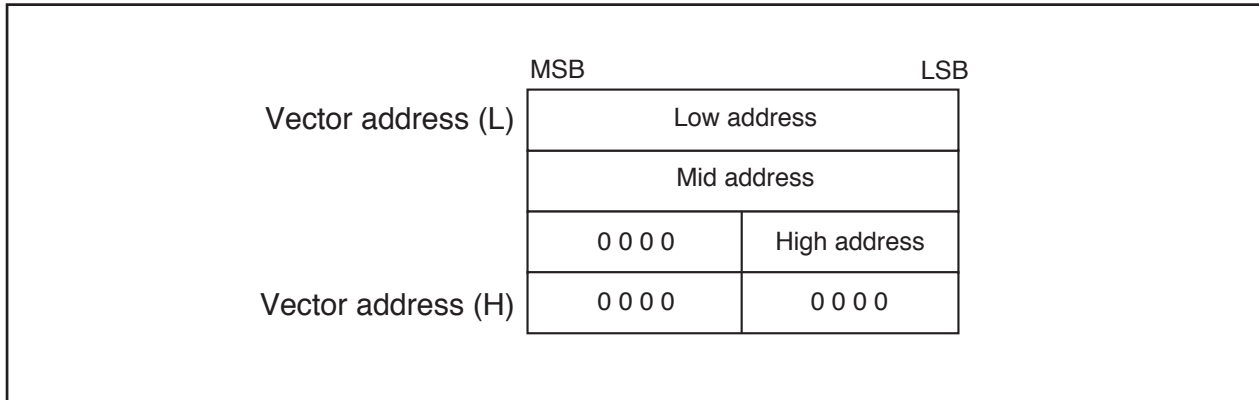


Figure 1.9.2. Interrupt Vector

### • Fixed Vector Tables

The fixed vector tables are allocated to the addresses from FFFDC<sub>16</sub> to FFFFF<sub>16</sub>. Table 1.9.1 lists the fixed vector tables. In the flash memory version of microcomputer, the vector addresses (H) of fixed vectors are used by the ID code check function. For details, refer to the section "flash memory rewrite disabling function".

Table 1.9.1. Fixed Vector Tables

Interrupt source	Vector table addresses Address (L) to address (H)	Remarks	Reference
Undefined instruction	FFFD <sub>C16</sub> to FFFD <sub>F16</sub>	Interrupt on UND instruction	M16C/60, M16C/20 series software manual
Overflow	FFFE <sub>016</sub> to FFFE <sub>316</sub>	Interrupt on INTO instruction	
BRK instruction	FFFE <sub>416</sub> to FFFE <sub>716</sub>	If the contents of address FFFE <sub>716</sub> is FF <sub>16</sub> , program execution starts from the address shown by the vector in the relocatable vector table.	
Address match	FFFE <sub>816</sub> to FFFEB <sub>16</sub>		Address match interrupt
Single step (Note)	FFFE <sub>C16</sub> to FFFE <sub>F16</sub>		
Watchdog timer Oscillation stop and re-oscillation detection	FFFF <sub>016</sub> to FFFF <sub>316</sub>		Watchdog timer  Clock generating circuit
DBC (Note)	FFFF <sub>416</sub> to FFFF <sub>716</sub>		
— (Reserved)	FFFF <sub>816</sub> to FFFF <sub>B16</sub>		
Reset	FFFF <sub>C16</sub> to FFFF <sub>F16</sub>		Reset

Note: Do not normally use this interrupt because it is provided exclusively for use by development support tools.

### • Relocatable Vector Tables

The 256 bytes beginning with the start address set in the INTB register comprise a relocatable vector table area. Table 1.9.2 lists the relocatable vector tables. Setting an even address in the INTB register results in the interrupt sequence being executed faster than in the case of odd addresses.

**Table 1.9.2. Relocatable Vector Tables**

Interrupt source	Vector address (Note 1) Address (L) to address (H)	Software interrupt number	Reference
BRK instruction (Note 5)	+0 to +3 (0000 <sub>16</sub> to 0003 <sub>16</sub> )	0	M16C/60, M16C/20 series software manual
———— (Reserved)		1 to 3	
INT $\bar{3}$	+16 to +19 (0010 <sub>16</sub> to 0013 <sub>16</sub> )	4	INT interrupt
———— (Reserved)	+20 to +23 (0014 <sub>16</sub> to 0017 <sub>16</sub> )	5	
Timer B4, UART1 bus collision <sup>(Note 4)</sup> detect	+24 to +27 (0018 <sub>16</sub> to 001B <sub>16</sub> )	6	Timer Serial I/O
Timer B3, UART0 bus collision <sup>(Note 4)</sup> detect	+28 to +31 (001C <sub>16</sub> to 001F <sub>16</sub> )	7	
SI/O4, INT $\bar{5}$ (Note 2)	+32 to +35 (0020 <sub>16</sub> to 0023 <sub>16</sub> )	8	INT interrupt Serial I/O
SI/O3, INT $\bar{4}$ (Note 2)	+36 to +39 (0024 <sub>16</sub> to 0027 <sub>16</sub> )	9	
UART 2 bus collision detection	+40 to +43 (0028 <sub>16</sub> to 002B <sub>16</sub> )	10	Serial I/O
DMA0	+44 to +47 (002C <sub>16</sub> to 002F <sub>16</sub> )	11	DMAC
DMA1	+48 to +51 (0030 <sub>16</sub> to 0033 <sub>16</sub> )	12	
———— (Reserved)	+52 to +55 (0034 <sub>16</sub> to 0037 <sub>16</sub> )	13	
———— (Reserved)	+56 to +59 (0038 <sub>16</sub> to 003B <sub>16</sub> )	14	
UART2 transmit, NACK2 (Note 3)	+60 to +63 (003C <sub>16</sub> to 003F <sub>16</sub> )	15	Serial I/O
UART2 receive, ACK2 (Note 3)	+64 to +67 (0040 <sub>16</sub> to 0043 <sub>16</sub> )	16	
UART0 transmit, NACK0 (Note 3)	+68 to +71 (0044 <sub>16</sub> to 0047 <sub>16</sub> )	17	
UART0 receive, ACK0 (Note 3)	+72 to +75 (0048 <sub>16</sub> to 004B <sub>16</sub> )	18	
UART1 transmit, NACK1 (Note 3)	+76 to +79 (004C <sub>16</sub> to 004F <sub>16</sub> )	19	
UART1 receive, ACK1 (Note 3)	+80 to +83 (0050 <sub>16</sub> to 0053 <sub>16</sub> )	20	
Timer A0	+84 to +87 (0054 <sub>16</sub> to 0057 <sub>16</sub> )	21	Timer
Timer A1	+88 to +91 (0058 <sub>16</sub> to 005B <sub>16</sub> )	22	
Timer A2	+92 to +95 (005C <sub>16</sub> to 005F <sub>16</sub> )	23	
Timer A3	+96 to +99 (0060 <sub>16</sub> to 0063 <sub>16</sub> )	24	
Timer A4	+100 to +103 (0064 <sub>16</sub> to 0067 <sub>16</sub> )	25	
———— (Reserved)	+104 to +107 (0068 <sub>16</sub> to 006B <sub>16</sub> )	26	
———— (Reserved)	+108 to +111 (006C <sub>16</sub> to 006F <sub>16</sub> )	27	
———— (Reserved)	+112 to +115 (0070 <sub>16</sub> to 0073 <sub>16</sub> )	28	
INT $\bar{0}$	+116 to +119 (0074 <sub>16</sub> to 0077 <sub>16</sub> )	29	INT interrupt
INT $\bar{1}$	+120 to +123 (0078 <sub>16</sub> to 007B <sub>16</sub> )	30	
INT $\bar{2}$	+124 to +127 (007C <sub>16</sub> to 007F <sub>16</sub> )	31	
Software interrupt (Note 5)	+128 to +131 (0080 <sub>16</sub> to 0083 <sub>16</sub> ) to +252 to +255 (00FC <sub>16</sub> to 00FF <sub>16</sub> )	32 to 63	M16C/60, M16C/20 series software manual

Note 1: Address relative to address in INTB.

Note 2: Set the IFSR register's IFSR6 and IFSR7 bits "0".

Note 3: During I<sup>2</sup>C mode, NACK and ACK interrupts comprise the interrupt source.

Note 4: Set the IFSR2A register's IFSR26 and IFSR27 bits "1".

Note 5: These interrupts cannot be disabled using the I flag.

## Interrupt Control

The following describes how to enable/disable the maskable interrupts, and how to set the priority in which order they are accepted. What is explained here does not apply to nonmaskable interrupts.

Use the FLG register's I flag, IPL, and each interrupt control register's ILVL2 to ILVL0 bits to enable/disable the maskable interrupts. Whether an interrupt is requested is indicated by the IR bit in each interrupt control register.

Figure 1.9.3 shows the interrupt control registers.

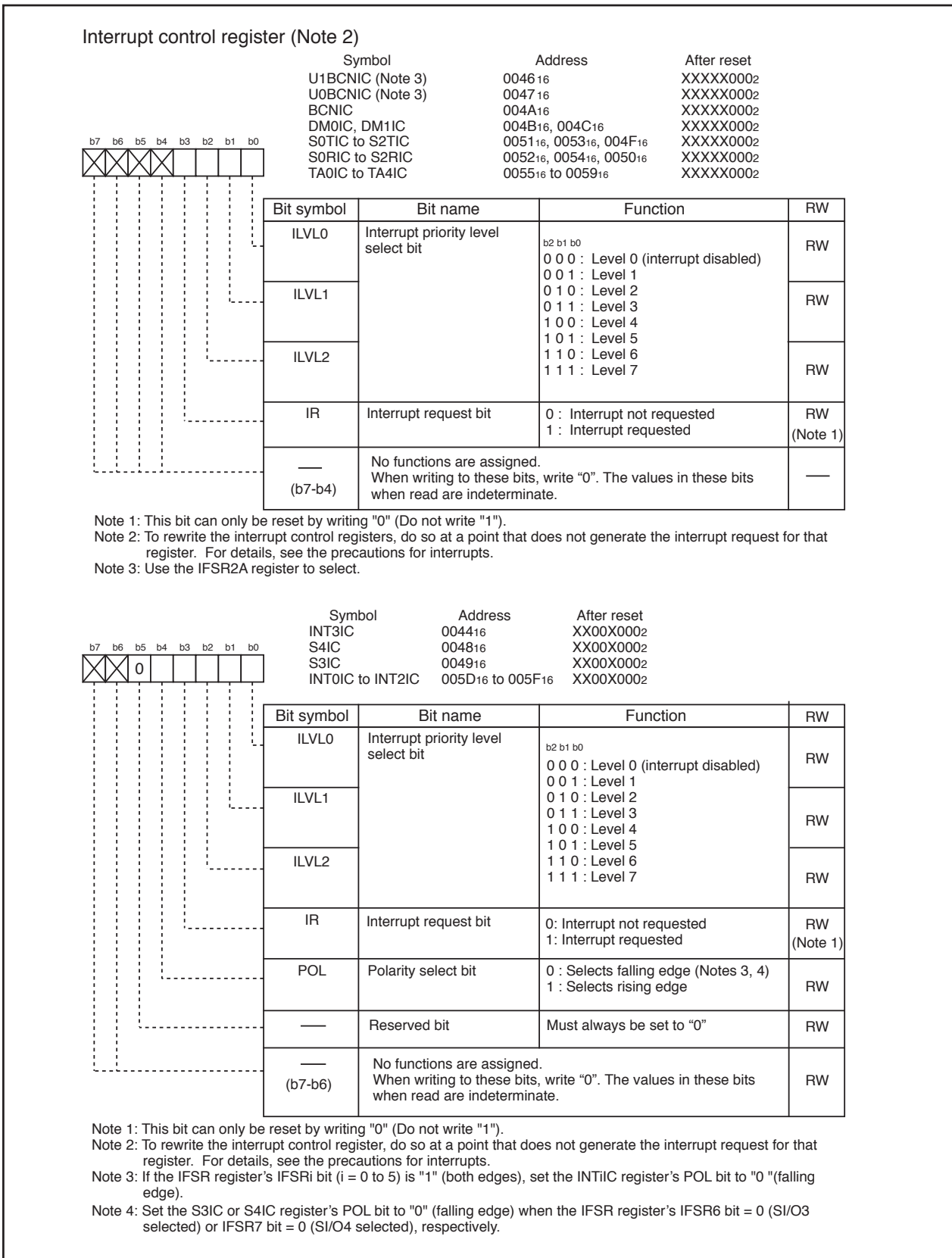


Figure 1.9.3. Interrupt Control Registers

## I Flag

The I flag enables or disables the maskable interrupt. Setting the I flag to “1” (= enabled) enables the maskable interrupt. Setting the I flag to “0” (= disabled) disables all maskable interrupts.

## IR Bit

The IR bit is set to “1” (= interrupt requested) when an interrupt request is generated. Then, when the interrupt request is accepted and the CPU branches to the corresponding interrupt vector, the IR bit is cleared to “0” (= interrupt not requested).

The IR bit can be cleared to “0” in a program. Note that do not write “1” to this bit.

## ILVL2 to ILVL0 Bits and IPL

Interrupt priority levels can be set using the ILVL2 to ILVL0 bits.


Table 1.9.3 shows the settings of interrupt priority levels and Table 1.9.4 shows the interrupt priority levels enabled by the IPL.

The following are conditions under which an interrupt is accepted:

- I flag = “1”
- IR bit = “1”
- interrupt priority level > IPL

The I flag, IR bit, ILVL2 to ILVL0 bits and IPL are independent of each other. In no case do they affect one another.

**Table 1.9.3. Settings of Interrupt Priority Levels**

ILVL2 to ILVL0 bits	Interrupt priority level	Priority order
0002	Level 0 (interrupt disabled)	————
0012	Level 1	Low  High
0102	Level 2	
0112	Level 3	
1002	Level 4	
1012	Level 5	
1102	Level 6	
1112	Level 7	

**Table 1.9.4. Interrupt Priority Levels Enabled by IPL**

IPL	Enabled interrupt priority levels
0002	Interrupt levels 1 and above are enabled
0012	Interrupt levels 2 and above are enabled
0102	Interrupt levels 3 and above are enabled
0112	Interrupt levels 4 and above are enabled
1002	Interrupt levels 5 and above are enabled
1012	Interrupt levels 6 and above are enabled
1102	Interrupt levels 7 and above are enabled
1112	All maskable interrupts are disabled

## Interrupt Sequence

An interrupt sequence — what are performed over a period from the instant an interrupt is accepted to the instant the interrupt routine is executed — is described here.

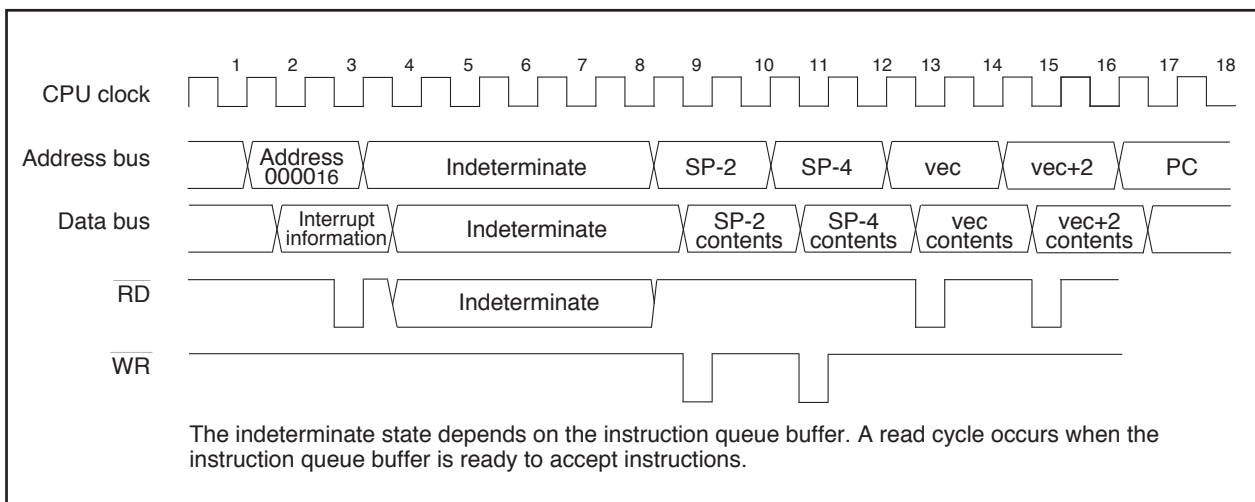
If an interrupt occurs during execution of an instruction, the processor determines its priority when the execution of the instruction is completed, and transfers control to the interrupt sequence from the next cycle. If an interrupt occurs during execution of either the SMOVB, SMOVF, SSTR or RMPA instruction, the processor temporarily suspends the instruction being executed, and transfers control to the interrupt sequence.

The CPU behavior during the interrupt sequence is described below. Figure 1.9.4 shows time required for executing the interrupt sequence.

- (1) The CPU gets interrupt information (interrupt number and interrupt request priority level) by reading the address 000016. Then it clears the IR bit for the corresponding interrupt to “0” (interrupt not requested).
- (2) The FLG register immediately before entering the interrupt sequence is saved to the CPU’s internal temporary register<sup>(Note 1)</sup>.
- (3) The I, D and U flags in the FLG register become as follows:
  - The I flag is cleared to “0” (interrupts disabled).
  - The D flag is cleared to “0” (single-step interrupt disabled).
  - The U flag is cleared to “0” (ISP selected).
 However, the U flag does not change state if an INT instruction for software interrupt Nos. 32 to 63 is executed.
- (4) The CPU’s internal temporary register <sup>(Note 1)</sup> is saved to the stack.
- (5) The PC is saved to the stack.
- (6) The interrupt priority level of the accepted interrupt is set in the IPL.
- (7) The start address of the relevant interrupt routine set in the interrupt vector is stored in the PC.

After the interrupt sequence is completed, the processor resumes executing instructions from the start address of the interrupt routine.

Note: This register cannot be used by user.



**Figure 1.9.4. Time Required for Executing Interrupt Sequence**

### Interrupt Response Time

Figure 1.9.5 shows the interrupt response time. The interrupt response or interrupt acknowledge time denotes a time from when an interrupt request is generated till when the first instruction in the interrupt routine is executed. Specifically, it consists of a time from when an interrupt request is generated till when the instruction then executing is completed ((a) in Figure 1.9.5) and a time during which the interrupt sequence is executed ((b) in Figure 1.9.5).

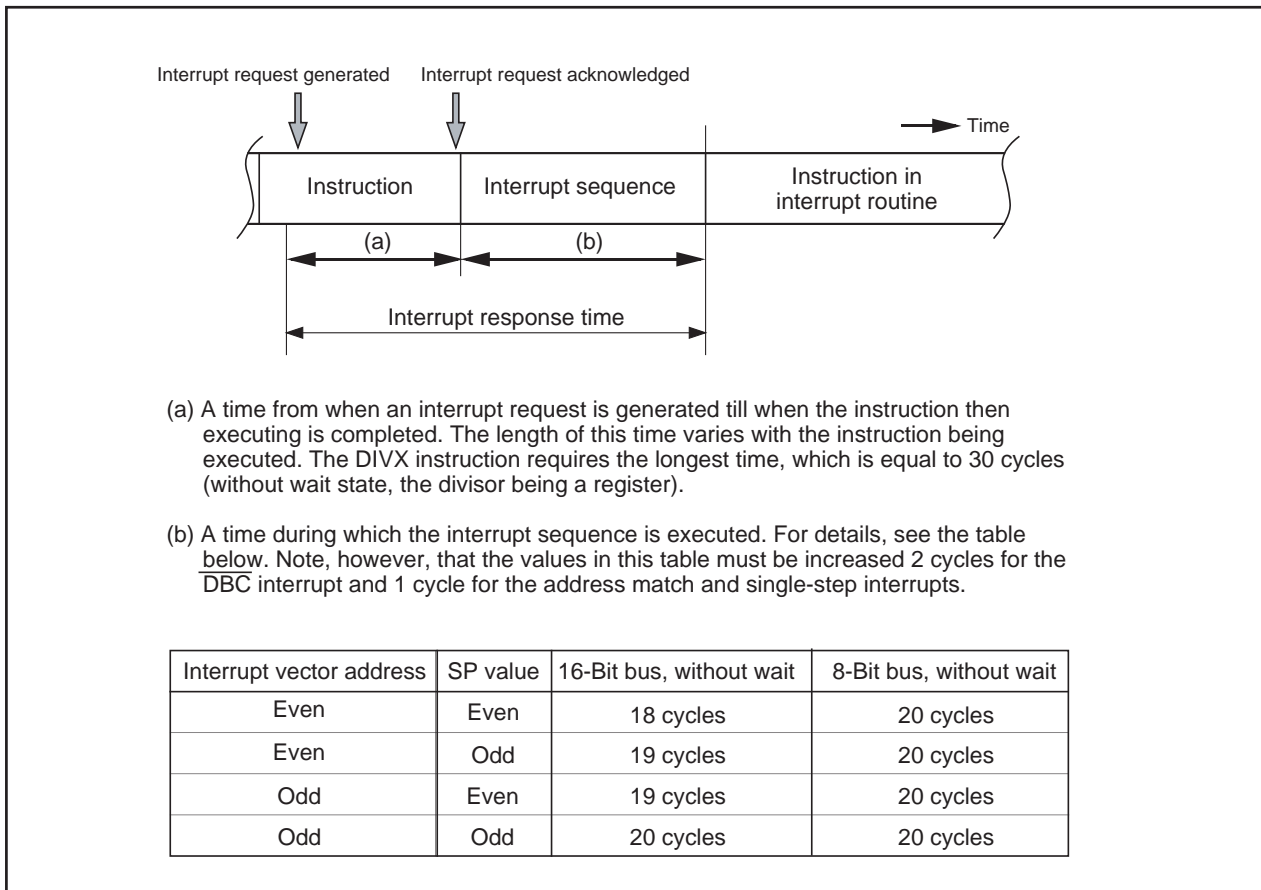


Figure 1.9.5. Interrupt response time

### Variation of IPL when Interrupt Request is Accepted

When a maskable interrupt request is accepted, the interrupt priority level of the accepted interrupt is set in the IPL.

When a software interrupt or special interrupt request is accepted, one of the interrupt priority levels listed in Table 1.9.5 is set in the IPL. Shown in Table 1.9.5 are the IPL values of software and special interrupts when they are accepted.

Table 1.9.5. IPL Level That is Set to IPL When A Software or Special Interrupt Is Accepted

Interrupt sources	Level that is set to IPL
Watchdog timer	7
Software, address match, $\overline{DBC}$ , single-step	Not changed



## Saving Registers

In the interrupt sequence, the FLG register and PC are saved to the stack.

At this time, the 4 high-order bits of the PC and the 4 high-order (IPL) and 8 low-order bits of the FLG register, 16 bits in total, are saved to the stack first. Next, the 16 low-order bits of the PC are saved. Figure 1.9.6 shows the stack status before and after an interrupt request is accepted.

The other necessary registers must be saved in a program at the beginning of the interrupt routine. Use the PUSHM instruction, and all registers except SP can be saved with a single instruction.

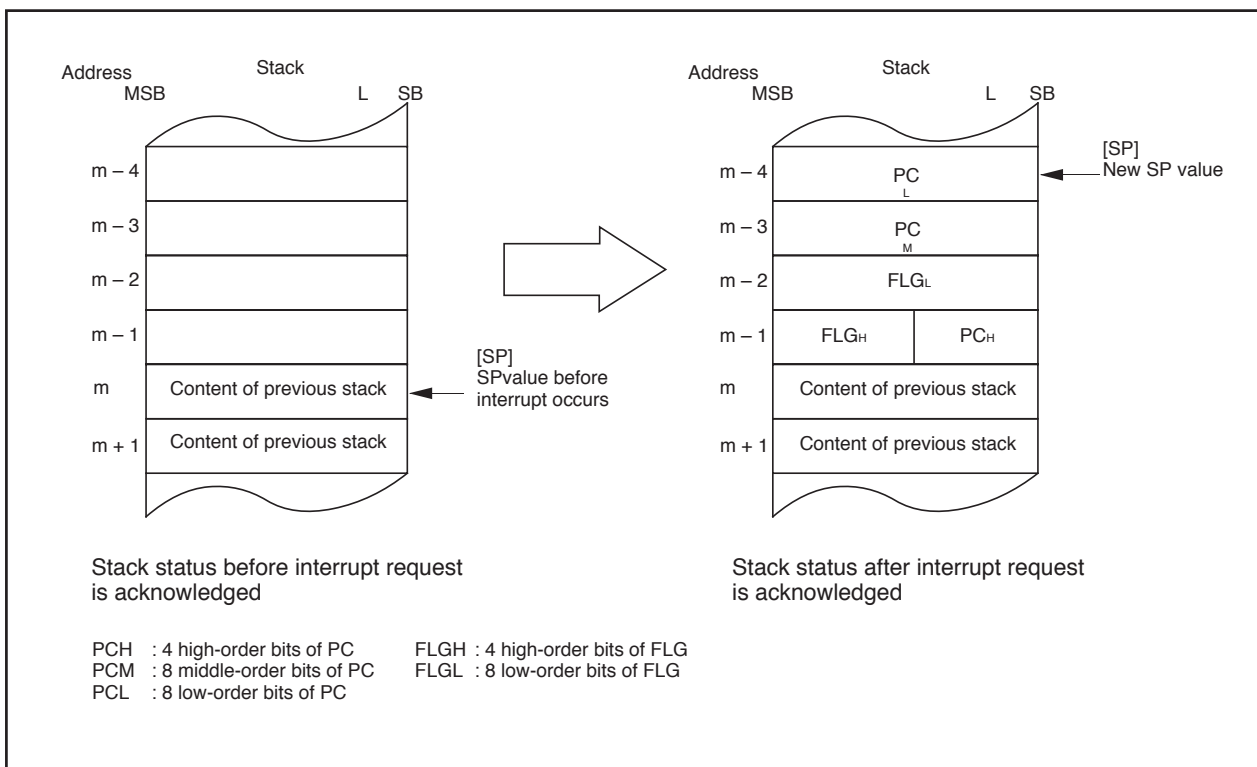


Figure 1.9.6. Stack Status Before and After Acceptance of Interrupt Request

The operation of saving registers carried out in the interrupt sequence is dependent on whether the SP<sup>(Note)</sup>, at the time of acceptance of an interrupt request, is even or odd. If the stack pointer <sup>(Note)</sup> is even, the FLG register and the PC are saved, 16 bits at a time. If odd, they are saved in two steps, 8 bits at a time. Figure 1.9.7 shows the operation of the saving registers.

Note: When any INT instruction in software numbers 32 to 63 has been executed, this is the SP indicated by the U flag. Otherwise, it is the ISP.

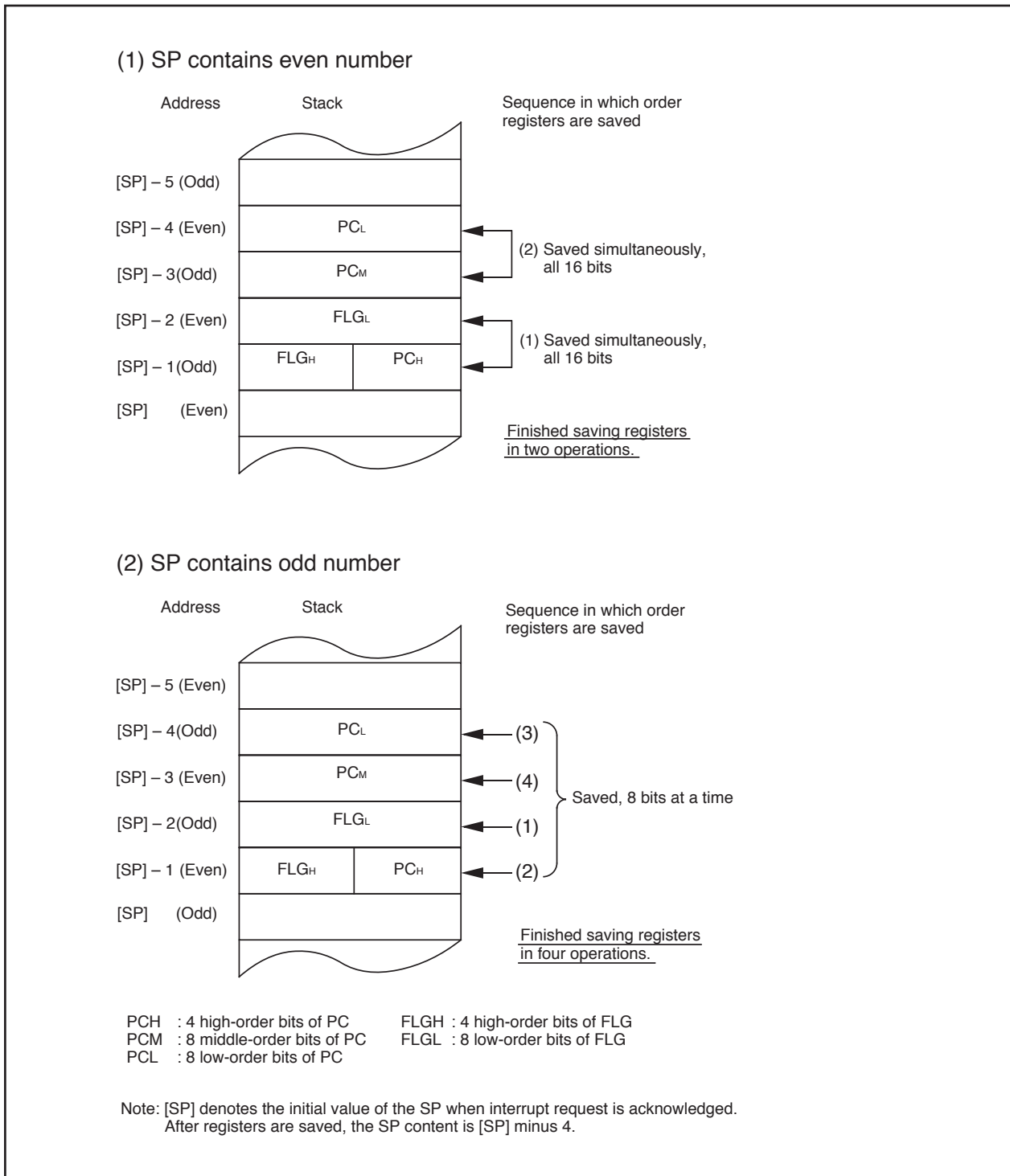


Figure 1.9.7. Operation of Saving Register

## Returning from an Interrupt Routine

The FLG register and PC in the state in which they were immediately before entering the interrupt sequence are restored from the stack by executing the REIT instruction at the end of the interrupt routine. Thereafter the CPU returns to the program which was being executed before accepting the interrupt request.

Return the other registers saved by a program within the interrupt routine using the POPM or similar instruction before executing the REIT instruction.

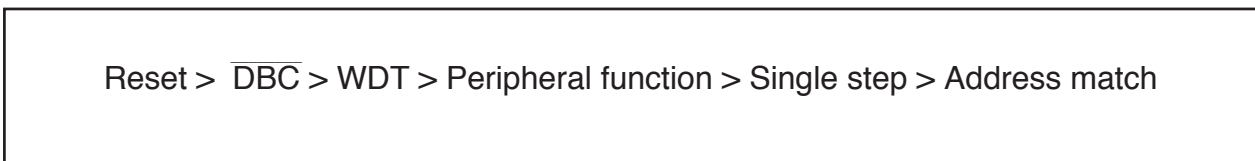
## Interrupt Priority

If two or more interrupt requests are generated while executing one instruction, the interrupt request that has the highest priority is accepted.

For maskable interrupts (peripheral functions), any desired priority level can be selected using the ILVL2 to ILVL0 bits. However, if two or more maskable interrupts have the same priority level, their interrupt priority is resolved by hardware, with the highest priority interrupt accepted.

The watchdog timer and other special interrupts have their priority levels set in hardware. Figure 1.9.8 shows the priorities of hardware interrupts.

Software interrupts are not affected by the interrupt priority. If an instruction is executed, control branches invariably to the interrupt routine.



**Figure 1.9.8. Hardware Interrupt Priority**

## Interrupt Priority Resolution Circuit

The interrupt priority resolution circuit is used to select the interrupt with the highest priority among those requested.

Figure 1.9.9 shows the circuit that judges the interrupt priority level.

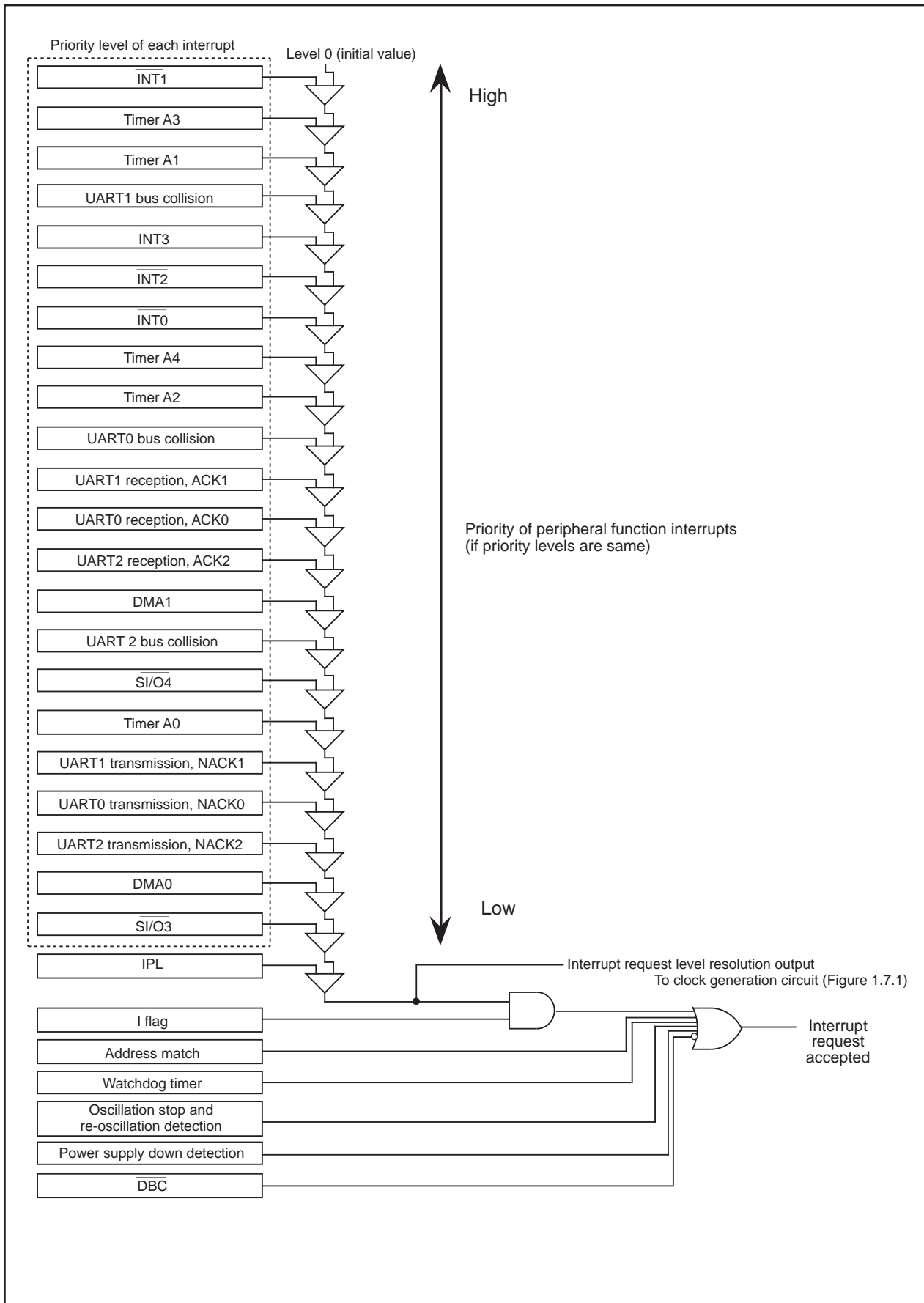


Figure 1.9.9. Interrupts Priority Select Circuit

## INT Interrupt

INT<sub>i</sub> interrupt (i=0 to 3) is triggered by the edges of external inputs. The edge polarity is selected using the IFSR register's IFSR<sub>i</sub> bit.

Figure 1.9.10 shows the IFSR and IFSR2A registers.

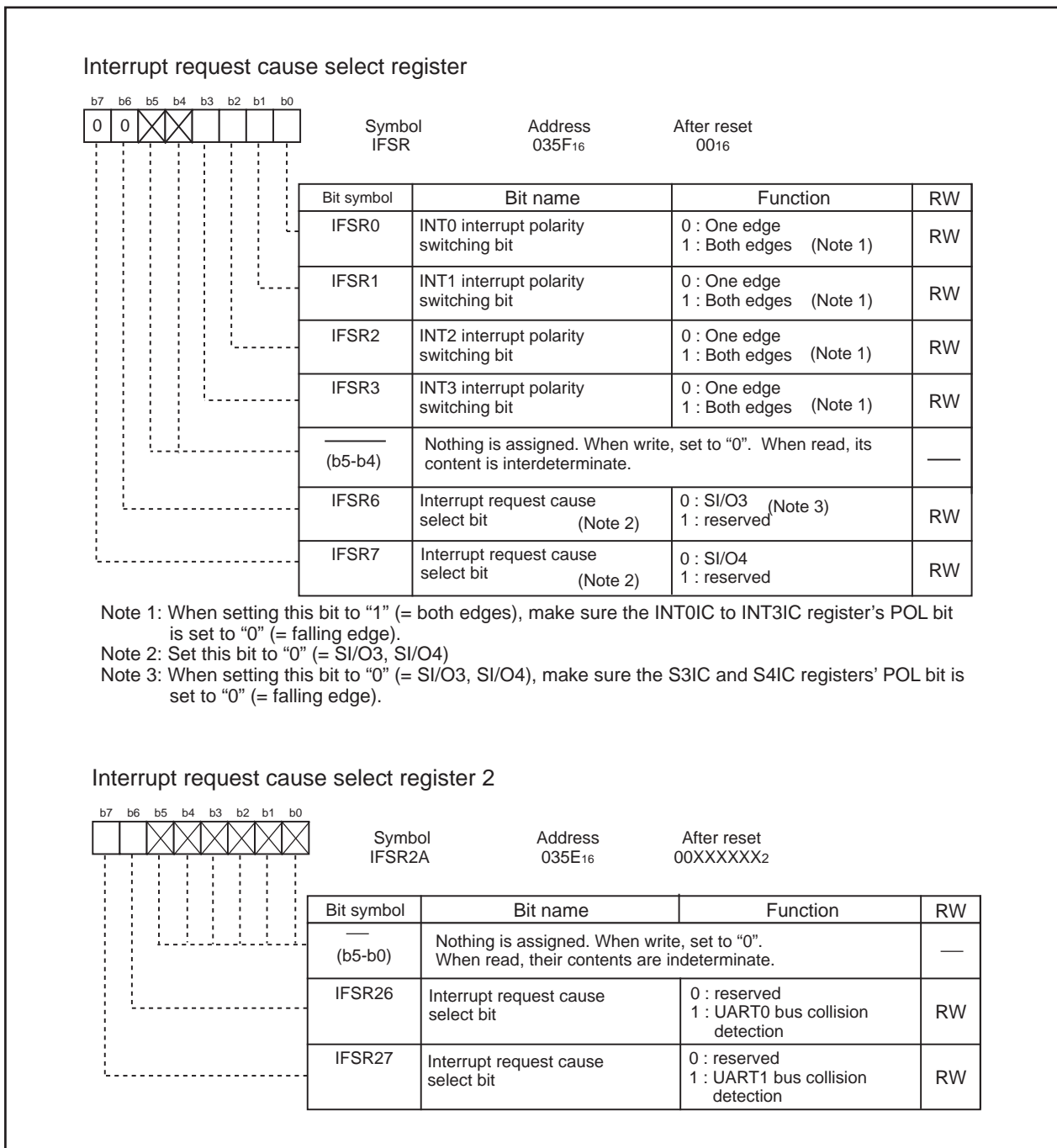


Figure 1.9.10. IFSR Register and IFSR2A Register

## Address Match Interrupt

An address match interrupt is generated immediately before executing the instruction at the address indicated by the RMADi register (i=0 to 3). Set the start address of any instruction in the RMADi register. Use the AIER register's AIER0 and AIER1 bits and the AIER2 register's AIER20 and AIER21 bits to enable or disable the interrupt. Note that the address match interrupt is unaffected by the I flag and IPL. For address match interrupts, the value of the PC that is saved to the stack area varies depending on the instruction being executed (refer to **Saving Registers**).

(The value of the PC that is saved to the stack area is not the correct return address.) Therefore, follow one of the methods described below to return from the address match interrupt.

- Rewrite the content of the stack and then use the REIT instruction to return.
- Restore the stack to its previous state before the interrupt request was accepted by using the POP or similar other instruction and then use a jump instruction to return.

Table 1.9.6 shows the value of the PC that is saved to the stack area when an address match interrupt request is accepted.

Figure 1.9.11 shows the AIER, AIER2, and RMAD0 to RMAD3 registers.

**Table 1.9.6. Value of the PC that is saved to the stack area when an address match interrupt request is accepted**

Instruction at the Address Indicated by the RMADi Register	Value of the PC that is saved to the stack area
<ul style="list-style-type: none"> <li>• 16-bit op-code instruction</li> <li>• Instruction shown below among 8-bit operation code instructions</li> </ul> <pre> ADD.B:S  #IMM8,dest  SUB.B:S  #IMM8,dest  AND.B:S  #IMM8,dest OR.B:S   #IMM8,dest  MOV.B:S  #IMM8,dest  STZ.B:S  #IMM8,dest STNZ.B:S #IMM8,dest  STZX.B:S #IMM81,#IMM82,dest CMP.B:S  #IMM8,dest  PUSHM   src         POPM   dest JMPS     #IMM8      JSRS     #IMM8 MOV.B:S  #IMM,dest  (However, dest=A0 or A1)           </pre>	The address indicated by the RMADi register +2
Instructions other than the above	The address indicated by the RMADi register +1

Value of the PC that is saved to the stack area : Refer to **12.5.7 Saving Registers**.

**Table 1.9.7. Relationship Between Address Match Interrupt Sources and Associated Registers**

Address match interrupt sources	Address match interrupt enable bit	Address match interrupt register
Address match interrupt 0	AIER0	RMAD0
Address match interrupt 1	AIER1	RMAD1
Address match interrupt 2	AIER20	RMAD2
Address match interrupt 3	AIER21	RMAD3

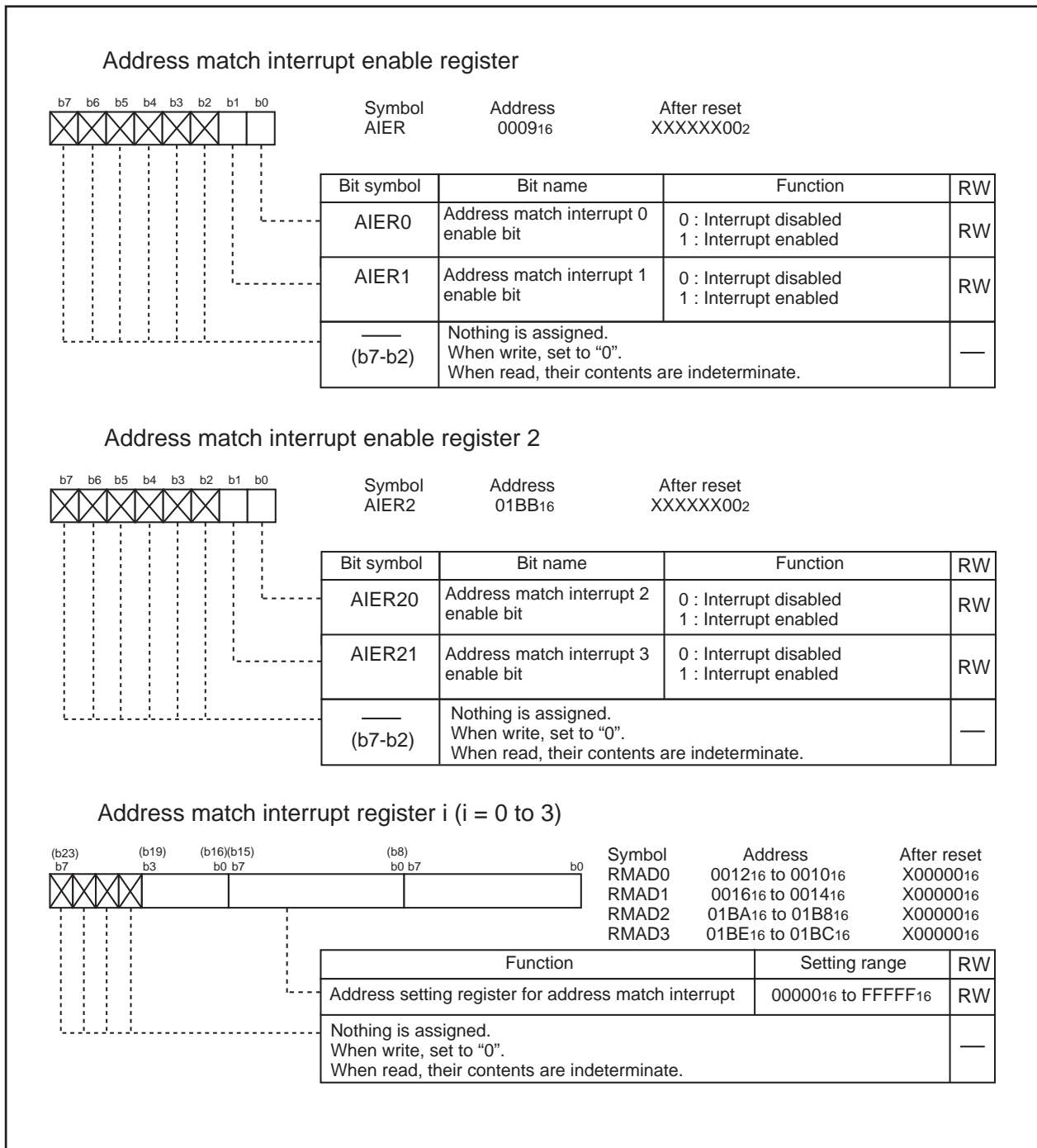


Figure 1.9.11. AIER Register, AIER2 Register and RMAD0 to RMAD3 Registers

## Precautions for Interrupts

### (1) Reading Address 00000<sub>16</sub>

- Do not read the address 00000<sub>16</sub> in a program. When a maskable interrupt request is accepted, the CPU reads interrupt information (interrupt number and interrupt request priority level) from the address 00000<sub>16</sub> during the interrupt sequence. At this time, the IR bit for the accepted interrupt is cleared to “0”. If the address 00000<sub>16</sub> is read in a program, the IR bit for the interrupt which has the highest priority among the enabled interrupts is cleared to “0”. This causes a problem that the interrupt is canceled, or an unexpected interrupt is generated.

### (2) SP Setting

- Set any value in the SP before accepting an interrupt. The SP is cleared to ‘0000<sub>16</sub>’ after reset. Therefore, if an interrupt is accepted before setting any value in the SP, the program may go out of control.

### (3) $\overline{\text{INT}}$ Interrupt

- Either an “L” level or an “H” level of at least 250 ns width is necessary for the signal input to the  $\overline{\text{INT}}_0$  through  $\overline{\text{INT}}_3$  pins regardless of the CPU clock.
- When the polarity of the  $\overline{\text{INT}}_0$  to  $\overline{\text{INT}}_3$  pins is changed, the IR bit is sometimes set to “1” (=interrupt requested). After changing the polarity, set the IR bit to “0” (=interrupt not requested). Figure 1.9.12 shows the procedure for changing the  $\overline{\text{INT}}$  interrupt generate factor.

### (4) Watchdog Timer Interrupt

- Initialize the watchdog timer after the watchdog timer interrupt occurs.

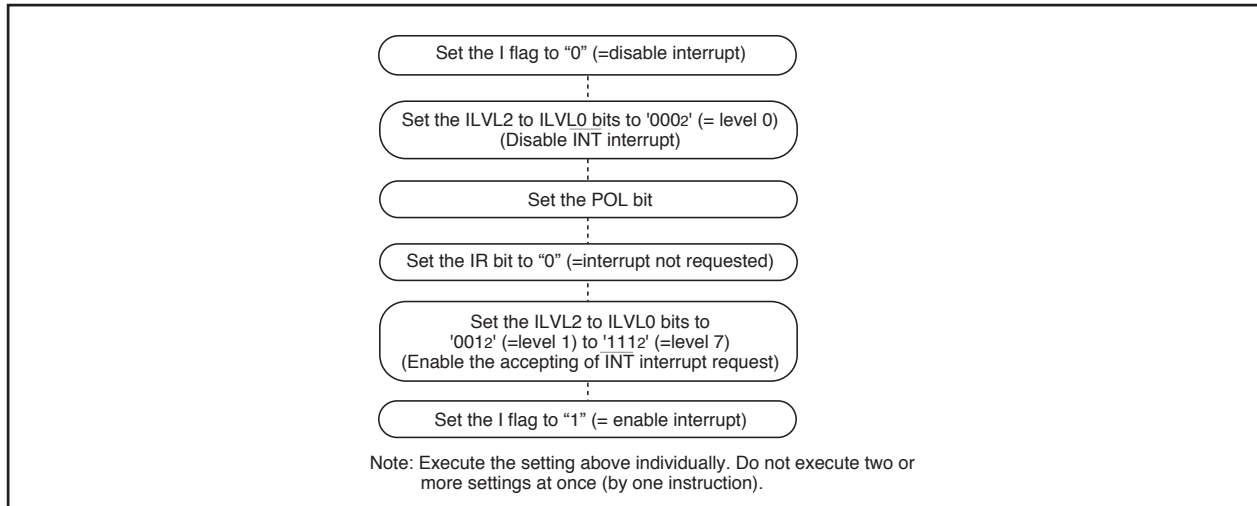


Figure 1.9.12. Switching Procedure for  $\overline{\text{INT}}$  Interrupt Request



## (5) Modifying Interrupt Control Register

- Each interrupt control register can only be modified while no interrupt requests corresponding to that register are generated. If interrupt requests managed by any interrupt control register are likely to occur, disable the interrupts before modifying the register. A sample program is shown below.

To modify any interrupt control register after disabling interrupts, be careful with the instructions used.

Modifying other than the IR bit

If an interrupt request corresponding to that register is generated while executing the instruction, the IR bit may not be set to "1" (= interrupt requested), with the result that the interrupt request is ignored. If this presents a problem, use the following instructions to modify the register.

Instructions to use: AND, OR, BCLR, BSET

Modifying the IR bit

Even when the IR bit is cleared to "0" (= interrupt not requested), it may not actually be cleared to "0" depending on the instruction used. Therefore, use the MOV instruction to clear the IR bit.

### Example 1:

```
INT_SWITCH1:
  FCLR  I           ; Disable interrupts.
  AND.B #00h, 0055h ; Set the TA0IC register to "0016".
  NOP                    ; Four NOP instructions are required when using HOLD function.
  NOP
  FSET  I           ; Enable interrupts.
```

### Example 2:

```
INT_SWITCH2:
  FCLR  I           ; Disable interrupts.
  AND.B #00h, 0055h ; Set the TA0IC register to "0016".
  MOV.W MEM, R0     ; Dummy read.
  FSET  I           ; Enable interrupts.
```

### Example 3:

```
INT_SWITCH3:
  PUSHC FLG         ; Push Flag register on to stack
  FCLR  I           ; Disable interrupts.
  AND.B #00h, 0055h ; Set the TA0IC register to "0016".
  POPC  FLG         ; Enable interrupts.
```

Why the FSET I instruction is preceded by two NOP instructions (four when using HOLD function) in Example 1 and why the FSET I instruction is preceded by a dummy read in Example 2  
This is to prevent the I flag from being set to "1" before writing to the interrupt control register for reasons of the instruction queue buffer.

## Watchdog Timer

The watchdog timer is the function of detecting when the program is out of control. Therefore, we recommend using the watchdog timer to improve reliability of a system. The watchdog timer contains a 15-bit counter which counts down the clock derived by dividing the CPU clock using the prescaler. Whether to generate a watchdog timer interrupt request or apply a watchdog timer reset as an operation to be performed when the watchdog timer underflows after reaching the terminal count can be selected using the PM12 bit of PM1 register. The PM12 bit can only be set to “1” (reset). Once this bit is set to “1”, it cannot be set to “0” (watchdog timer interrupt) in a program.

The pin, CPU and SFR initialized where the monitor timer underflows when the PM12 bit is “1” are the same as in software reset.

When the main clock is selected for CPU clock, the divide-by-N value for the prescaler can be chosen to be 16 or 128. The period of watchdog timer can be calculated as given below. The period of watchdog timer is, however, subject to an error due to the prescaler.

With main clock chosen for CPU clock

$$\text{Watchdog timer period} = \frac{\text{Prescaler dividing (16 or 128) X Watchdog timer count (32768)}}{\text{CPU clock}}$$

For example, when CPU clock = 16 MHz and the divide-by-N value for the prescaler= 16, the watchdog timer period is approx. 32.8 ms.

The watchdog timer is initialized by writing to the WDTS register. The prescaler is initialized after reset. Note that the watchdog timer and the prescaler both are inactive after reset, so that the watchdog timer is activated to start counting by writing to the WDTS register.

In stop mode and wait mode, the watchdog timer and prescaler are stopped. Counting is resumed from the held value when the modes or state are released.

Figure 1.10.1 shows the block diagram of the watchdog timer. Figure 1.10.2 shows the watchdog timer-related registers.

- Count source protective mode

In this mode, a On-chip Oscillator clock is used for the watchdog timer count source. The watchdog timer can be kept being clocked even when CPU clock stops as a result of run-away.

Before this mode can be used, the following register settings are required:

- (1) Set the PRC1 bit of PRCR register to “1” (enable writes to PM1 and PM2 registers).
- (2) Set the PM12 bit of PM1 register to “1” (reset when the watchdog timer underflows).
- (3) Set the PM22 bit of PM2 register to “1” (On-chip Oscillator clock used for the watchdog timer count source).
- (4) Set the PRC1 bit of PRCR register to “0” (disable writes to PM1 and PM2 registers).
- (5) Write to the WDTS register (watchdog timer starts counting).

Setting the PM22 bit to “1” results in the following conditions

- The On-chip Oscillator starts oscillating, and the On-chip Oscillator clock becomes the watchdog timer count source.

$$\text{Watchdog timer period} = \frac{\text{Watchdog timer count (32768)}}{\text{on-chip oscillator clock}}$$

- The CM10 bit of CM1 register is disabled against write. (Writing a “1” has no effect, nor is stop mode entered.)
- The watchdog timer does not stop when in wait mode.

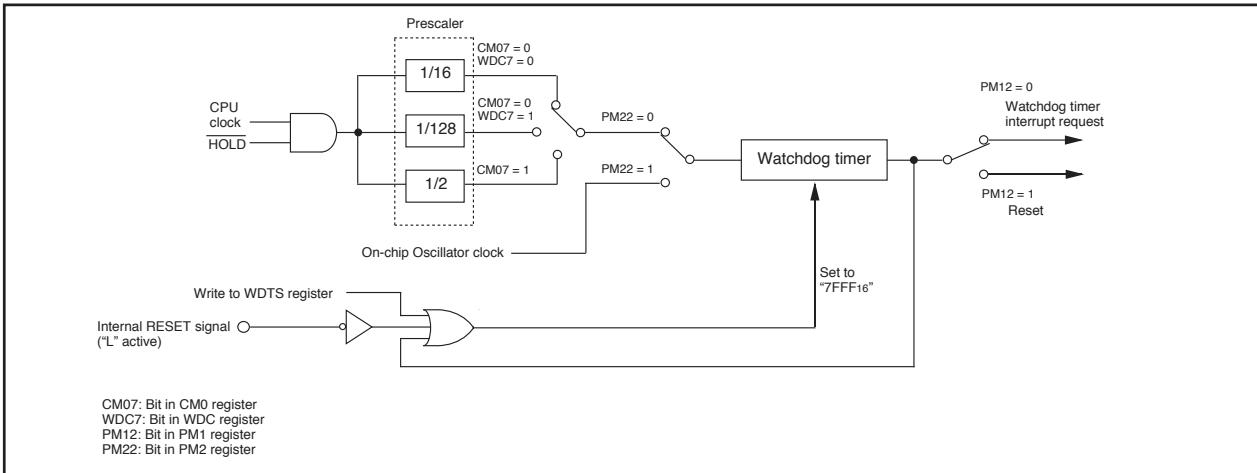


Figure 1.10.1. Watchdog Timer Block Diagram

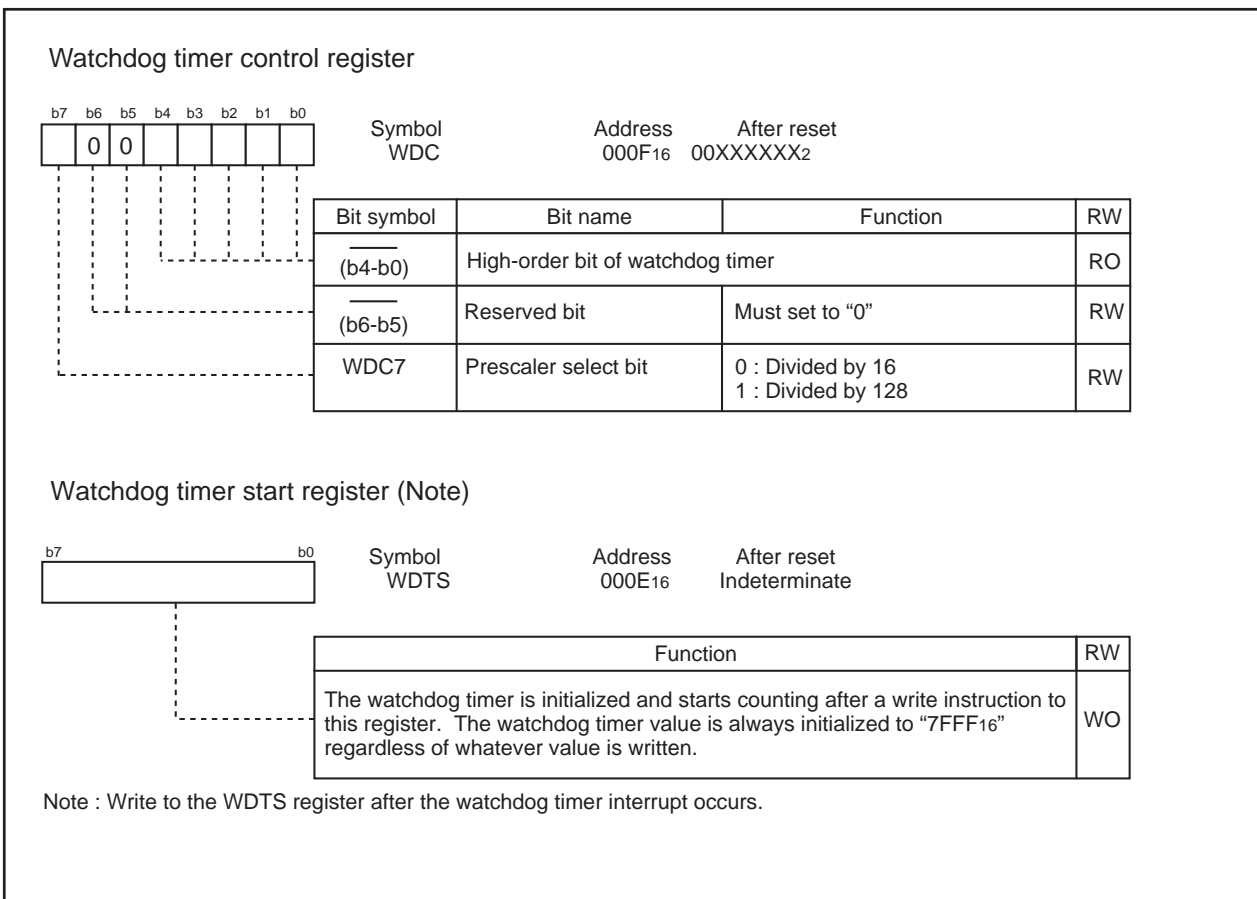
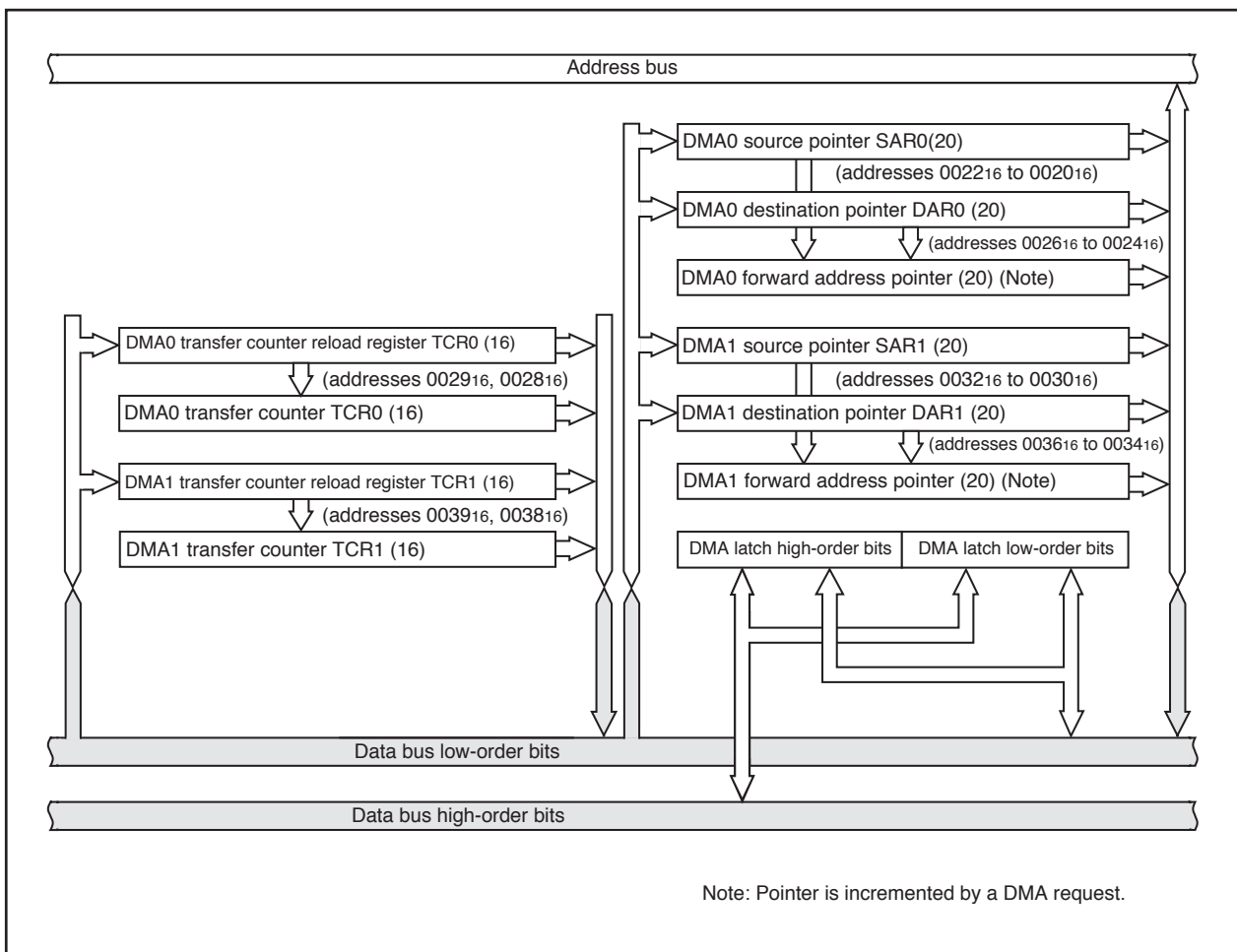


Figure 1.10.2. WDC Register and WDTS Register

## DMAC

The DMAC (Direct Memory Access Controller) allows data to be transferred without the CPU intervention. Two DMAC channels are included. Each time a DMA request occurs, the DMAC transfers one (8 or 16-bit) data from the source address to the destination address. The DMAC uses the same data bus as used by the CPU. Because the DMAC has higher priority of bus control than the CPU and because it makes use of a cycle steal method, it can transfer one word (16 bits) or one byte (8 bits) of data within a very short time after a DMA request is generated. Figure 1.11.1 shows the block diagram of the DMAC. Table 1.11.1 shows the DMAC specifications. Figures 1.11.2 to 1.11.4 show the DMAC-related registers.



**Figure 1.11.1. DMAC Block Diagram**

A DMA request is generated by a write to the DMiSL register ( $i = 0-1$ )'s DSR bit, as well as by an interrupt request which is generated by any function specified by the DMiSL register's DMS and DSEL3–DSEL0 bits. However, unlike in the case of interrupt requests, DMA requests are not affected by the I flag and the interrupt control register, so that even when interrupt requests are disabled and no interrupt request can be accepted, DMA requests are always accepted. Furthermore, because the DMAC does not affect interrupts, the interrupt control register's IR bit does not change state due to a DMA transfer.

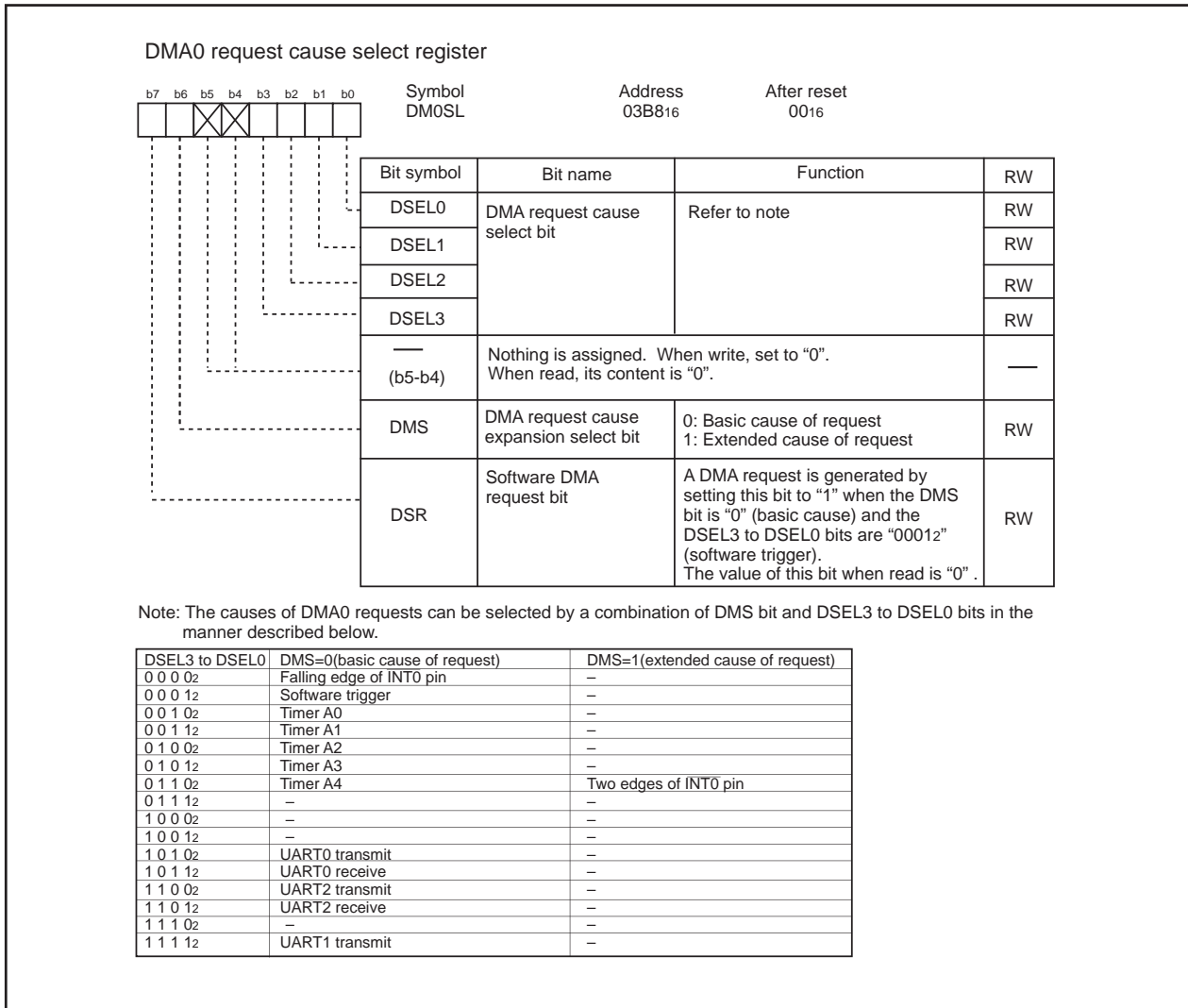
A data transfer is initiated each time a DMA request is generated when the DMiCON register's DMAE bit = "1" (DMA enabled). However, if the cycle in which a DMA request is generated is faster than the DMA transfer cycle, the number of transfer requests generated and the number of times data is transferred may not match. For details, refer to "DMA Requests".

**Table 1.11.1. DMAC Specifications**

Item	Specification	
No. of channels	2 (cycle steal method)	
Transfer memory space	<ul style="list-style-type: none"> <li>• From any address in the 1M bytes space to a fixed address</li> <li>• From a fixed address to any address in the 1M bytes space</li> <li>• From a fixed address to a fixed address</li> </ul>	
Maximum No. of bytes transferred	128K bytes (with 16-bit transfers) or 64K bytes (with 8-bit transfers)	
DMA request factors (Note 1, Note 2)	Falling edge of $\overline{INT0}$ or $\overline{INT1}$ Both edge of $\overline{INT0}$ or $\overline{INT1}$ Timer A0 to timer A4 interrupt requests UART0 transfer, UART0 reception interrupt requests UART1 transfer, UART1 reception interrupt requests UART2 transfer, UART2 reception interrupt requests SI/O3, SI/O4 interrupt requests Software triggers	
Channel priority	DMA0 > DMA1 (DMA0 takes precedence)	
Transfer unit	8 bits or 16 bits	
Transfer address direction	forward or fixed (The source and destination addresses cannot both be in the forward direction.)	
Transfer mode	•Single transfer	Transfer is completed when the DMA <sub>i</sub> transfer counter (i = 0–1) underflows after reaching the terminal count.
	•Repeat transfer	When the DMA <sub>i</sub> transfer counter underflows, it is reloaded with the value of the DMA <sub>i</sub> transfer counter reload register and a DMA transfer is continued with it.
DMA interrupt request generation timing	When the DMA <sub>i</sub> transfer counter underflowed	
DMA startup	Data transfer is initiated each time a DMA request is generated when the DMA <sub>i</sub> CON register's DMAE bit = "1" (enabled).	
DMA shutdown	•Single transfer	<ul style="list-style-type: none"> <li>•When the DMAE bit is set to "0" (disabled)</li> <li>•After the DMA<sub>i</sub> transfer counter underflows</li> </ul>
	•Repeat transfer	When the DMAE bit is set to "0" (disabled)
Reload timing for forward address pointer and transfer counter	When a data transfer is started after setting the DMAE bit to "1" (enabled), the forward address pointer is reloaded with the value of the SAR <sub>i</sub> or the DAR <sub>i</sub> pointer whichever is specified to be in the forward direction and the DMA <sub>i</sub> transfer counter is reloaded with the value of the DMA <sub>i</sub> transfer counter reload register.	

## Notes:

1. DMA transfer is not effective to any interrupt. DMA transfer is affected neither by the I flag nor by the interrupt control register.
2. The selectable causes of DMA requests differ with each channel.
3. Make sure that no DMAC-related registers (addresses 0020<sub>16</sub>–003F<sub>16</sub>) are accessed by the DMAC.



**Figure 1.11.2. DM0SL Register**

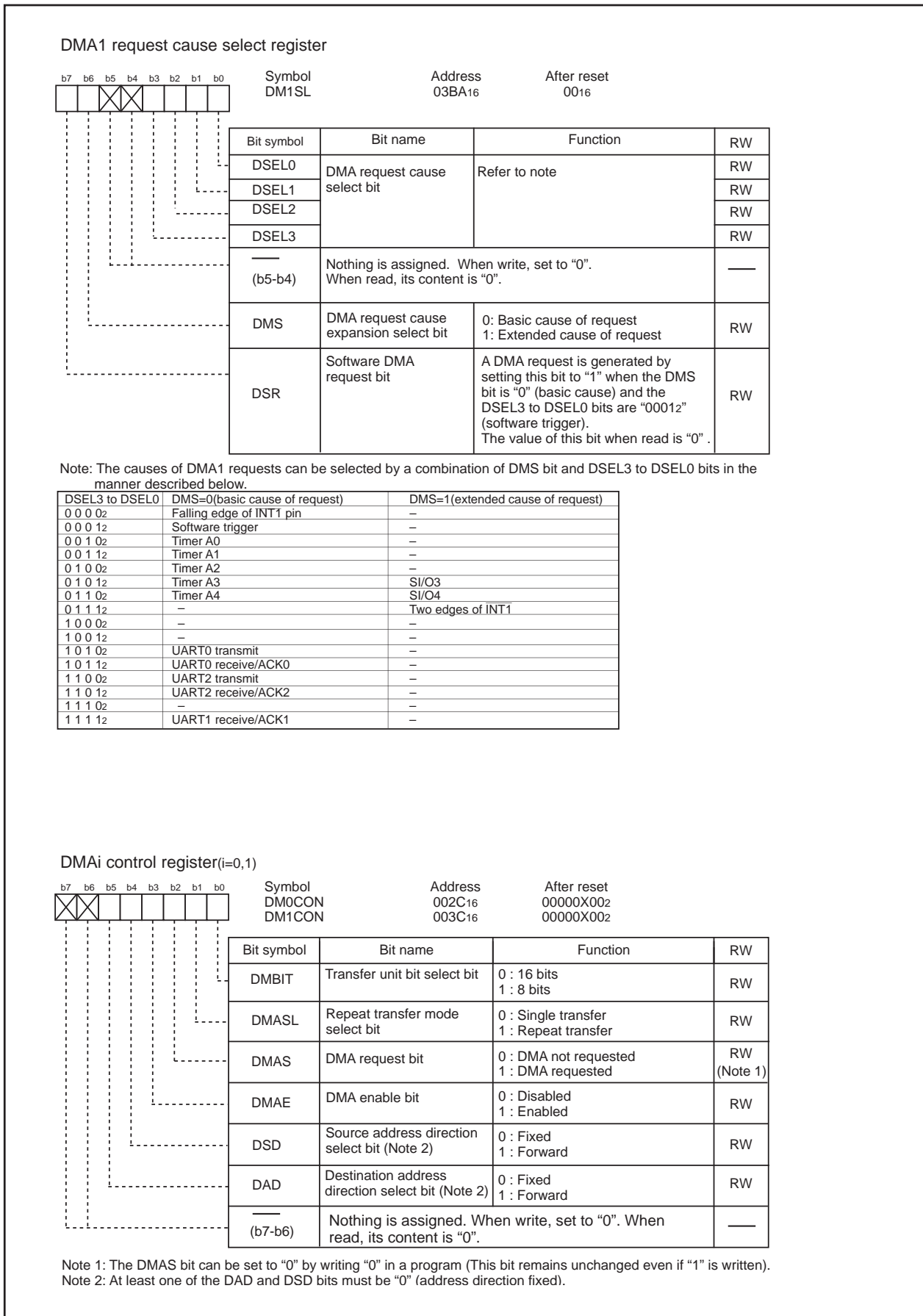


Figure 1.11.3. DM1SL Register, DM0CON Register, and DM1CON Registers

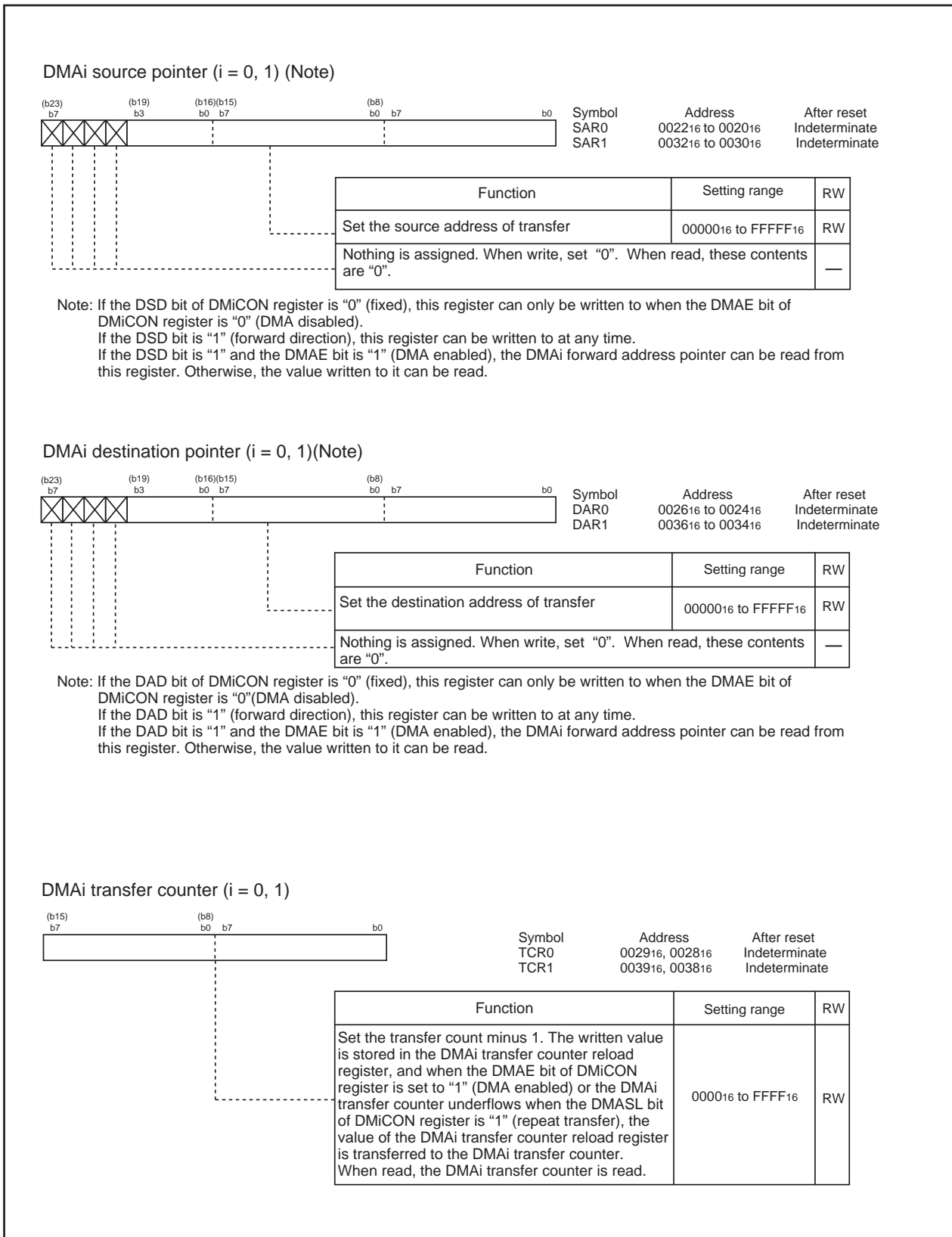


Figure 1.11.4. SAR0, SAR1, DAR0, DAR1, TCR0, and TCR1 Registers



## 1. Transfer Cycles

The transfer cycle consists of a memory or SFR read (source read) bus cycle and a write (destination write) bus cycle. The number of read and write bus cycles is affected by the source and destination addresses of transfer. Furthermore, the bus cycle itself is extended by a software wait or  $\overline{RDY}$  signal.

### (a) Effect of Source and Destination Addresses

If the transfer unit and data bus both are 16 bits and the source address of transfer begins with an odd address, the source read cycle consists of one more bus cycle than when the source address of transfer begins with an even address.

Similarly, if the transfer unit and data bus both are 16 bits and the destination address of transfer begins with an odd address, the destination write cycle consists of one more bus cycle than when the destination address of transfer begins with an even address.

### (b) Effect of Software Wait

For memory or SFR accesses in which one or more software wait states are inserted, the number of bus cycles required for that access increases by an amount equal to software wait states.

Figure 1.11.5 shows the example of the cycles for a source read. For convenience, the destination write cycle is shown as one cycle and the source read cycles for the different conditions are shown. In reality, the destination write cycle is subject to the same conditions as the source read cycle, with the transfer cycle changing accordingly. When calculating transfer cycles, take into consideration each condition for the source read and the destination write cycle, respectively. For example, when data is transferred in 16 bit units using an 8-bit bus ((2) in Figure 1.11.5), two source read bus cycles and two destination write bus cycles are required.

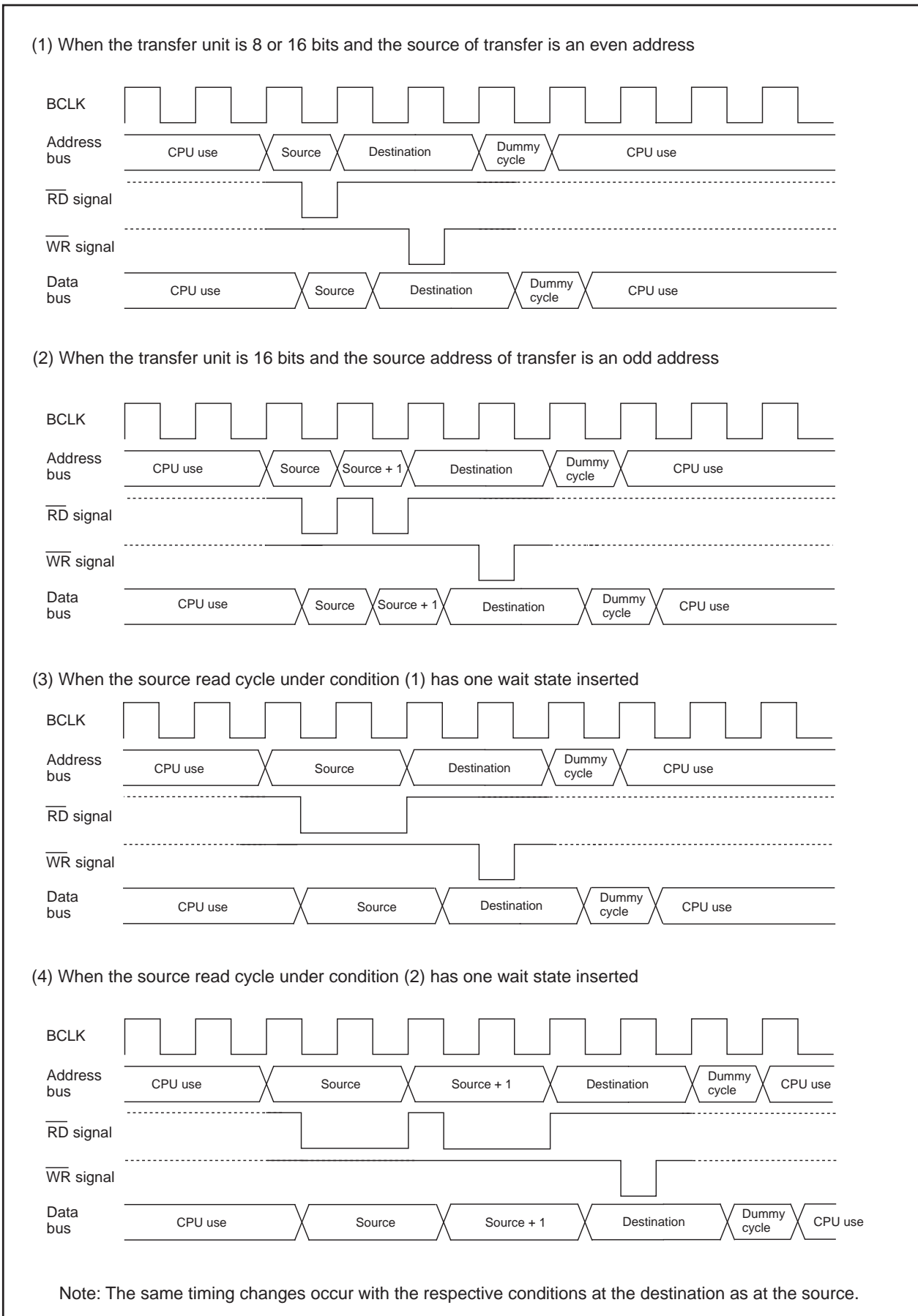


Figure 1.11.5. Transfer Cycles for Source Read

## 2. DMA Transfer Cycles

Any combination of even or odd transfer read and write addresses is possible. Table 1.11.2 shows the number of DMA transfer cycles. Table 1.11.3 shows the Coefficient j, k.

The number of DMAC transfer cycles can be calculated as follows:

$$\text{No. of transfer cycles per transfer unit} = \text{No. of read cycles} \times j + \text{No. of write cycles} \times k$$

**Table 1.11.2. DMA Transfer Cycles**

Transfer unit	Access address	No. of read cycles	No. of write cycles
8-bit transfers (DMBIT= "1")	Even	1	1
	Odd	1	1
16-bit transfers (DMBIT= "0")	Even	1	1
	Odd	2	2

**Table 1.11.3. Coefficient j, k**

	Internal ROM, RAM		SFR	
	No wait	With wait	1-wait <sup>2</sup>	2-wait <sup>2</sup>
j	1	2	2	3
k	1	2	2	3

Notes:

1. Depends on the set value of CSE register
2. Depends on the set value of PM20 bit in F

### 3. DMA Enable

When a data transfer starts after setting the DMAE bit in DMiCON register (i = 0, 1) to “1” (enabled), the DMAC operates as follows:

- (1) Reload the forward address pointer with the SARi register value when the DSD bit in DMiCON register is “1” (forward) or the DARi register value when the DAD bit of DMiCON register is “1” (forward).
- (2) Reload the DMAi transfer counter with the DMAi transfer counter reload register value.

If the DMAE bit is set to “1” again while it remains set, the DMAC performs the above operation. However, if a DMA request may occur simultaneously when the DMAE bit is being written, follow the steps below.

Step 1: Write “1” to the DMAE bit and DMAS bit in DMiCON register simultaneously.

Step 2: Make sure that the DMAi is in an initial state as described above (1) and (2) in a program.

If the DMAi is not in an initial state, the above steps should be repeated.

### 4. DMA Request

The DMAC can generate a DMA request as triggered by the cause of request that is selected with the DMS and DSEL3 to DSEL0 bits of DMiSL register (i = 0, 1) on either channel. Table 1.11.4 shows the timing at which the DMAS bit changes state.

Whenever a DMA request is generated, the DMAS bit is set to “1” (DMA requested) regardless of whether or not the DMAE bit is set. If the DMAE bit was set to “1” (enabled) when this occurred, the DMAS bit is set to “0” (DMA not requested) immediately before a data transfer starts. This bit cannot be set to “1” in a program (it can only be set to “0”).

The DMAS bit may be set to “1” when the DMS or the DSEL3 to DSEL0 bits change state. Therefore, always be sure to set the DMAS bit to “0” after changing the DMS or the DSEL3 to DSEL0 bits.

Because if the DMAE bit is “1”, a data transfer starts immediately after a DMA request is generated, the DMAS bit in almost all cases is “0” when read in a program. Read the DMAE bit to determine whether the DMAC is enabled.

**Table 1.11.4. Timing at Which the DMAS Bit Changes State**

DMA factor	DMAS bit of the DMiCON register	
	Timing at which the bit is set to “1”	Timing at which the bit is set to “0”
Software trigger	When the DSR bit of DMiCON register is set to “1”	<ul style="list-style-type: none"> <li>• Immediately before a data transfer starts</li> <li>• When set by writing “0” in a program</li> </ul>
Peripheral function	When the interrupt control register for the peripheral function that is selected by the DSEL3 to DSEL0 and DMS bits of DMiCON register has its IR bit set to “1”	

## Channel Priority and DMA Transfer Timing

If both DMA0 and DMA1 are enabled and DMA transfer request signals from DMA0 and DMA1 are detected active in the same sampling period (one period from a falling edge to the next falling edge of BCLK), the DMAS bit on each channel is set to "1" (DMA requested) at the same time. In this case, the DMA requests are arbitrated according to the channel priority, DMA0 > DMA1. The following describes DMAC operation when DMA0 and DMA1 requests are detected active in the same sampling period. Figure 1.11.6 shows an example of DMA transfer effected by external factors.

In Figure 1.11.6, because DMA0 and DMA1 requests occurred at the same time, DMA0 which has higher channel priority is accepted first and a DMA transfer on it starts. When DMA0 finishes one transfer unit, it relinquishes control of the bus to the CPU, and when the CPU finishes one bus access, DMA1 starts a transfer next and after completion of one transfer unit, returns control of the bus to the CPU.

Note that because there is only one DMAS bit on each channel, the number of times DMA is requested cannot be counted. Therefore, even if multiple DMA requests occurred before gaining control of the bus as in the case of DMA1 in Figure 1.11.6, the DMAS bit is set to "0" when control of the bus is gained and after completion of one transfer unit, control of the bus is returned to the CPU.

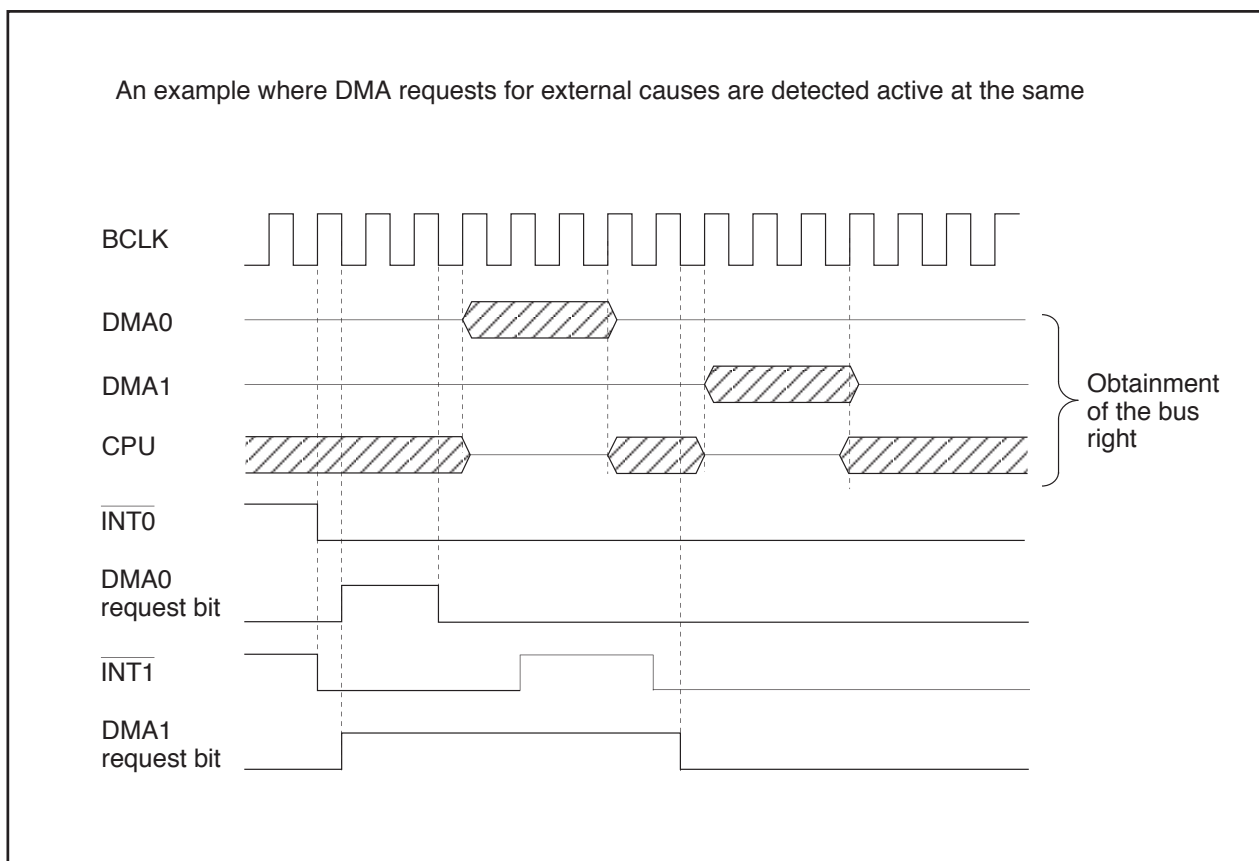


Figure 1.11.6. DMA Transfer by External Factors

# Timers

Five 16-bit timers, each capable of operating independently of the others. The count source for each timer acts as a clock, to control such timer operations as counting, reloading, etc. Figures 1.12.1 show block diagrams of timer A.

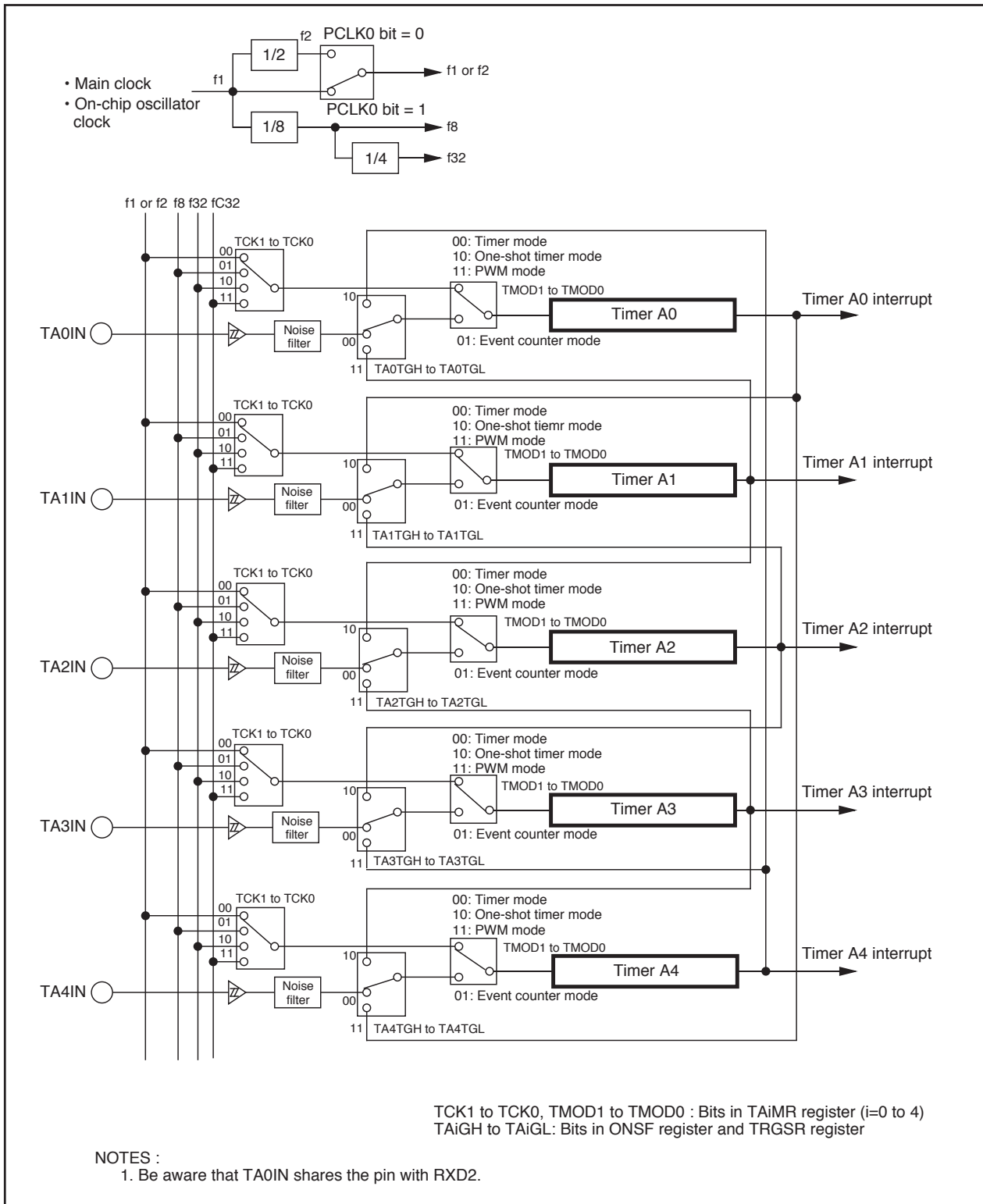


Figure 1.12.1. Timer A Configuration

### Timer A

Figure 1.12.2 shows a block diagram of the timer A. Figures 1.12.3 to 1.12.5 show registers related to the timer A.

The timer A supports the following four modes. Except in event counter mode, timers A0 to A4 all have the same function. Use the TMOD1 to TMOD0 bits of TAIiMR register (i = 0 to 4) to select the desired mode.

- Timer mode: The timer counts an internal count source.
- Event counter mode: The timer counts pulses from an external device or overflows and underflows of other timers.
- One-shot timer mode: The timer outputs a pulse only once before it reaches the minimum count "0000<sub>16</sub>."
- Pulse width modulation (PWM) mode: The timer outputs pulses in a given width successively.

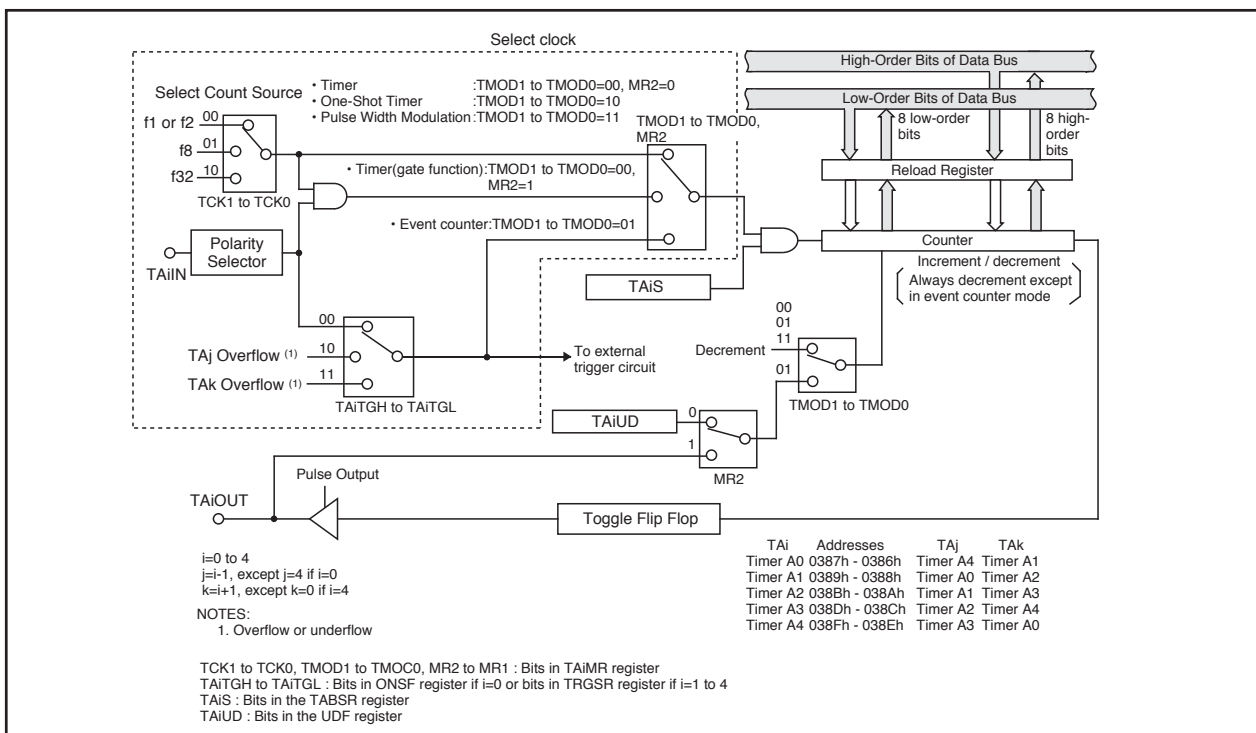


Figure 1.12.2. Timer A Block Diagram

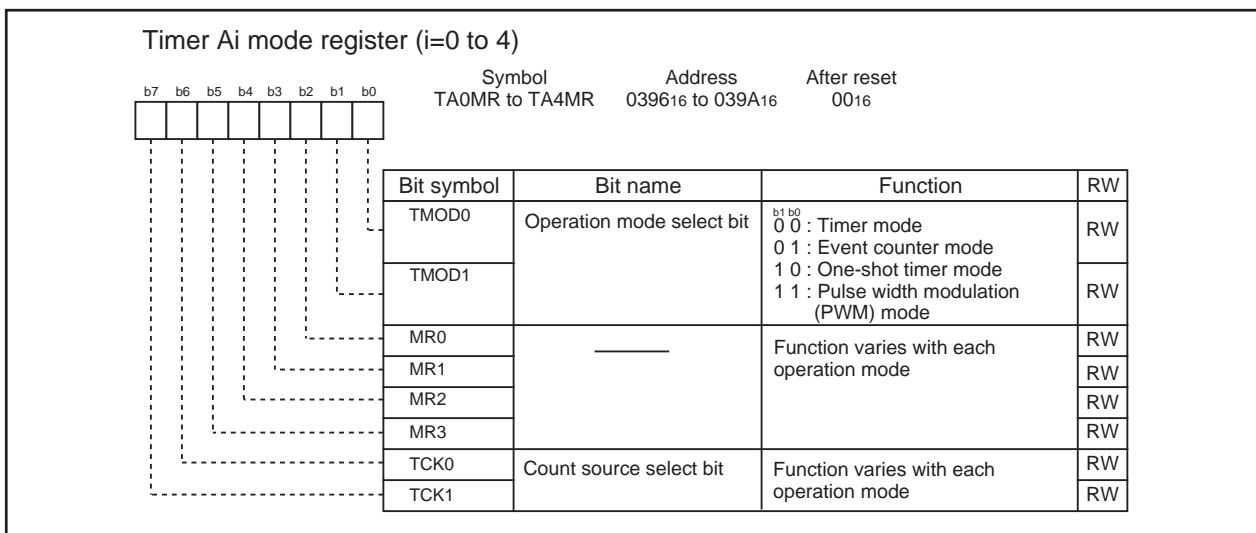


Figure 1.12.3. TA0MR to TA4MR Registers

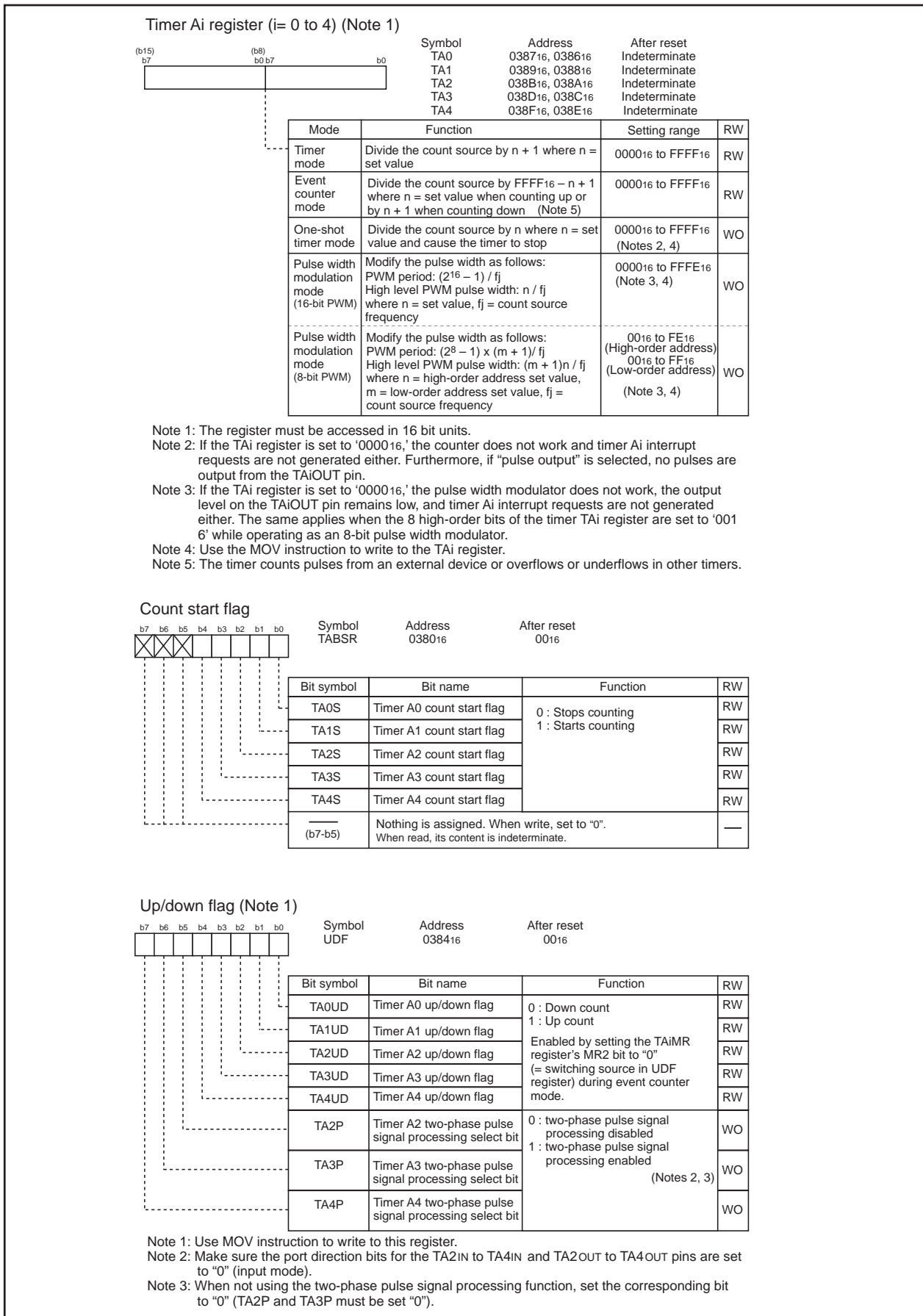


Figure 1.12.4. TA0 to TA4 Registers, TABSR Register, and UDF Register



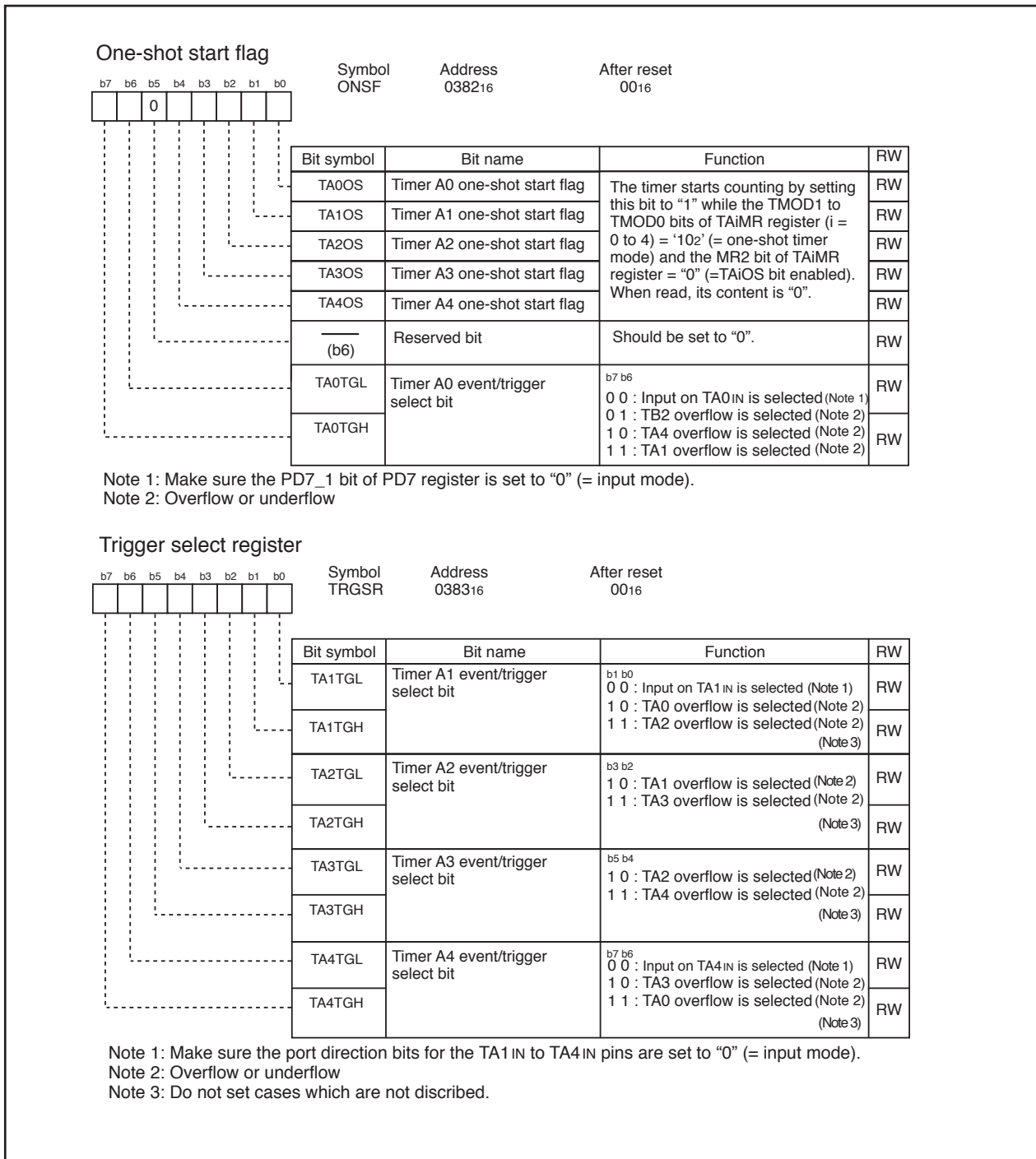


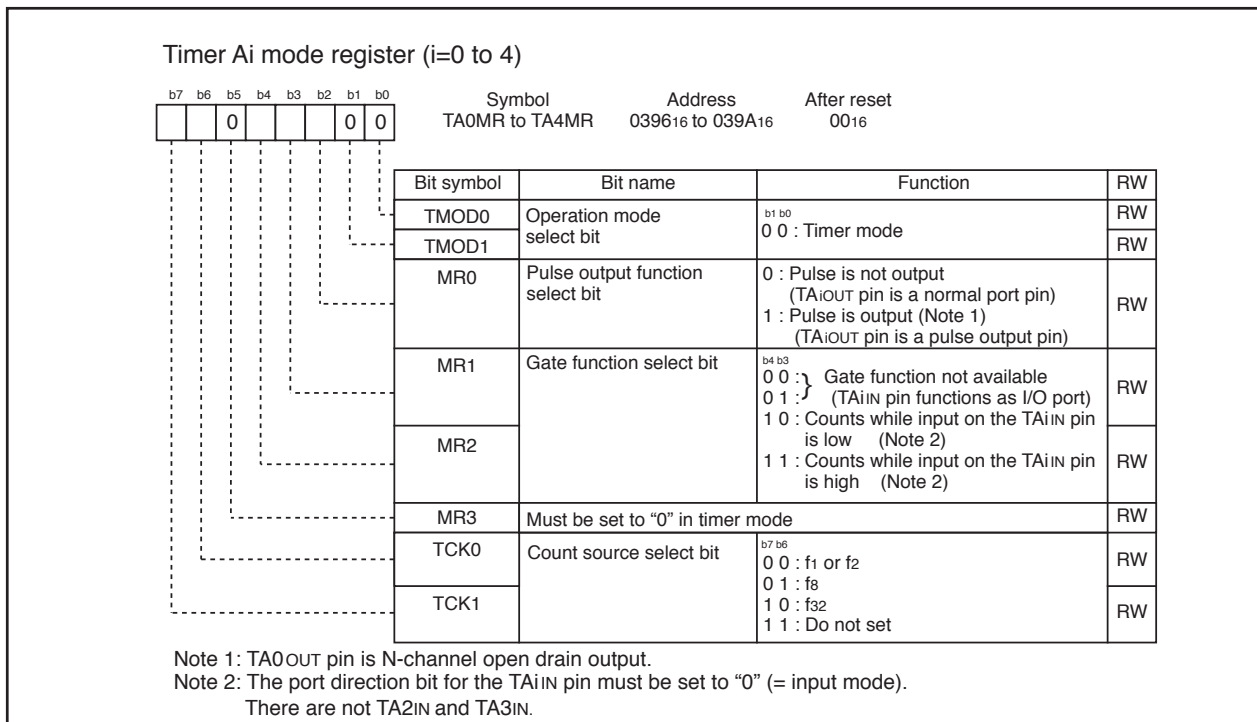
Figure 1.12.5. ONSF Register, TRGSR Register, and CPSRF Register

### 1. Timer Mode

In timer mode, the timer counts a count source generated internally (see Table 1.12.1). Figure 1.12.6 shows TAIiMR register in timer mode.

**Table 1.12.1. Specifications in Timer Mode**

Item	Specification
Count source	f1, f2, f8, f32
Count operation	<ul style="list-style-type: none"> <li>Down-count</li> <li>When the timer underflows, it reloads the reload register contents and continues counting</li> </ul>
Divide ratio	1/(n+1) n: set value of TAIiMR register (i= 0 to 4) 0000 <sub>16</sub> to FFFF <sub>16</sub>
Count start condition	Set TAIiS bit of TABSR register to "1" (= start counting)
Count stop condition	Set TAIiS bit to "0" (= stop counting)
Interrupt request generation timing	Timer underflow
TAiIN pin function	I/O port or gate input i≠2, 3
TAiOUT pin function	I/O port or pulse output
Read from timer	Count value can be read by reading TAIi register
Write to timer	<ul style="list-style-type: none"> <li>When not counting and until the 1st count source is input after counting start Value written to TAIi register is written to both reload register and counter</li> <li>When counting (after 1st count source input) Value written to TAIi register is written to only reload register (Transferred to counter when reloaded next)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>Gate function Counting can be started and stopped by an input signal to TAIiIN pin</li> <li>Pulse output function Whenever the timer underflows, the output polarity of TAIiOUT pin is inverted. When not counting, the pin outputs a low.</li> </ul>



**Figure 1.12.6. Timer Ai Mode Register in Timer Mode**

## 2. Event Counter Mode

In event counter mode, the timer counts pulses from an external device or overflows and underflows of other timers. Timer A4 can count two-phase external signals. Table 1.12.2 lists specifications in event counter mode (when not processing two-phase pulse signal). Table 1.12.3 lists specifications in event counter mode (when processing two-phase pulse signal with the timer A4). Figure 1.12.7 shows TAI<sub>MR</sub> register in event counter mode (when not processing two-phase pulse signal). Figure 1.12.8 shows TA4<sub>MR</sub> registers in event counter mode (when processing two-phase pulse signal with the timer A4).

**Table 1.12.2. Specifications in Event Counter Mode (when not processing two-phase pulse signal)**

Item	Specification
Count source	<ul style="list-style-type: none"> <li>External signals input to TAI<sub>IN</sub> pin (i=0 to 4) (effective edge can be selected in program)</li> <li>timer A<sub>j</sub> (j=i-1, except j=4 if i=0) overflows or underflows,</li> <li>timer A<sub>k</sub> (k=i+1, except k=0 if i=4) overflows or underflows</li> </ul>
Count operation	<ul style="list-style-type: none"> <li>Up-count or down-count can be selected by external signal or program</li> <li>When the timer overflows or underflows, it reloads the reload register contents and continues counting. When operating in free-running mode, the timer continues counting without reloading.</li> </ul>
Divided ratio	$1 / (FFFF_{16} - n + 1)$ for up-count $1 / (n + 1)$ for down-count    n : set value of TAI register    0000 <sub>16</sub> to FFFF <sub>16</sub>
Count start condition	Set TAI <sub>S</sub> bit of TABSR register to "1" (= start counting)
Count stop condition	Set TAI <sub>S</sub> bit to "0" (= stop counting)
Interrupt request generation timing	Timer overflow or underflow
TAI <sub>IN</sub> pin function	I/O port or count source input    i≠2, 3
TAI <sub>OUT</sub> pin function	I/O port, pulse output, or up/down-count select input
Read from timer	Count value can be read by reading TAI register
Write to timer	<ul style="list-style-type: none"> <li>When not counting and until the 1st count source is input after counting start Value written to TAI register is written to both reload register and counter</li> <li>When counting (after 1st count source input) Value written to TAI register is written to only reload register (Transferred to counter when reloaded next)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>Free-run count function Even when the timer overflows or underflows, the reload register content is not reloaded to it</li> <li>Pulse output function Whenever the timer underflows or underflows, the output polarity of TAI<sub>OUT</sub> pin is inverted . When not counting, the pin outputs a low.</li> </ul>

Timer Ai mode register (i=0 to 4)  
(When not using two-phase pulse signal processing)

Bit	Symbol	Address	After reset
b7			
b6			
b5	0		
b4			
b3			
b2			
b1	0		
b0	1		

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 1 : Event counter mode (Note 1)	RW
			RW
MR0	Pulse output function select bit	0 : Pulse is not output (TAiOUT pin functions as I/O port) 1 : Pulse is output (Note 2) (TAiOUT pin functions as pulse output pin)	RW
MR1	Count polarity select bit (Note 3)	0 : Counts external signal's falling edge 1 : Counts external signal's rising edge	RW
MR2	Up/down switching cause select bit	0 : UDF register 1 : Input signal to TAiOUT pin (Note 4)	RW
MR3	Must be set to "0" in event counter mode		RW
TCK0	Count operation type select bit	0 : Reload type 1 : Free-run type	RW
TCK1	Can be "0" or "1" when not using two-phase pulse signal processing		RW

Note 1: During event counter mode, the count source can be selected using the ONSF and TRGSR registers.

Note 2: TA0OUT pin is N-channel open drain output.

Note 3: Effective when the TAiGH and TAiGL bits of ONSF or TRGSR register are '002' (TAiIN pin input).

Note 4: Count down when input on TAiOUT pin is low or count up when input on that pin is high. The port direction bit for TAiOUT pin must be set to "0" (= input mode).

**Figure 1.12.7. TAiMR Register in Event Counter Mode (when not using two-phase pulse signal processing)**

Table 1.12.3. Specifications in Event Counter Mode (when processing two-phase pulse signal with timer A4)

Item	Specification
Count source	• Two-phase pulse signals input to TAIIN or TAIOUT pins (i = 4)
Count operation	• Up-count or down-count can be selected by two-phase pulse signal • When the timer overflows or underflows, it reloads the reload register contents and continues counting. When operating in free-running mode, the timer continues counting without reloading.
Divide ratio	1/ (FFFF <sub>16</sub> - n + 1) for up-count 1/ (n + 1) for down-count      n : set value of TAI register    0000 <sub>16</sub> to FFFF <sub>16</sub>
Count start condition	Set TAI <sub>i</sub> S bit of TABSR register to "1" (= start counting)
Count stop condition	Set TAI <sub>i</sub> S bit to "0" (= stop counting)
Interrupt request generation timing	Timer overflow or underflow
TAIIN pin function	Two-phase pulse input
TAIOUT pin function	Two-phase pulse input
Read from timer	Count value can be read by reading timer A4 register
Write to timer	• When not counting and until the 1st count source is input after counting start Value written to TAI register is written to both reload register and counter • When counting (after 1st count source input) Value written to TAI register is written to reload register (Transferred to counter when reloaded next)
Select function	<p>• Multiply-by-4 processing operation (timer A4)</p> <p>If the phase relationship is such that TAKIN(k=4) pin goes "H" when the input signal on TAKOUT pin is "H", the timer counts up rising and falling edges on TAKOUT and TAKIN pins. If the phase relationship is such that TAKIN pin goes "L" when the input signal on TAKOUT pin is "H", the timer counts down rising and falling edges on TAKOUT and TAKIN pins.</p>

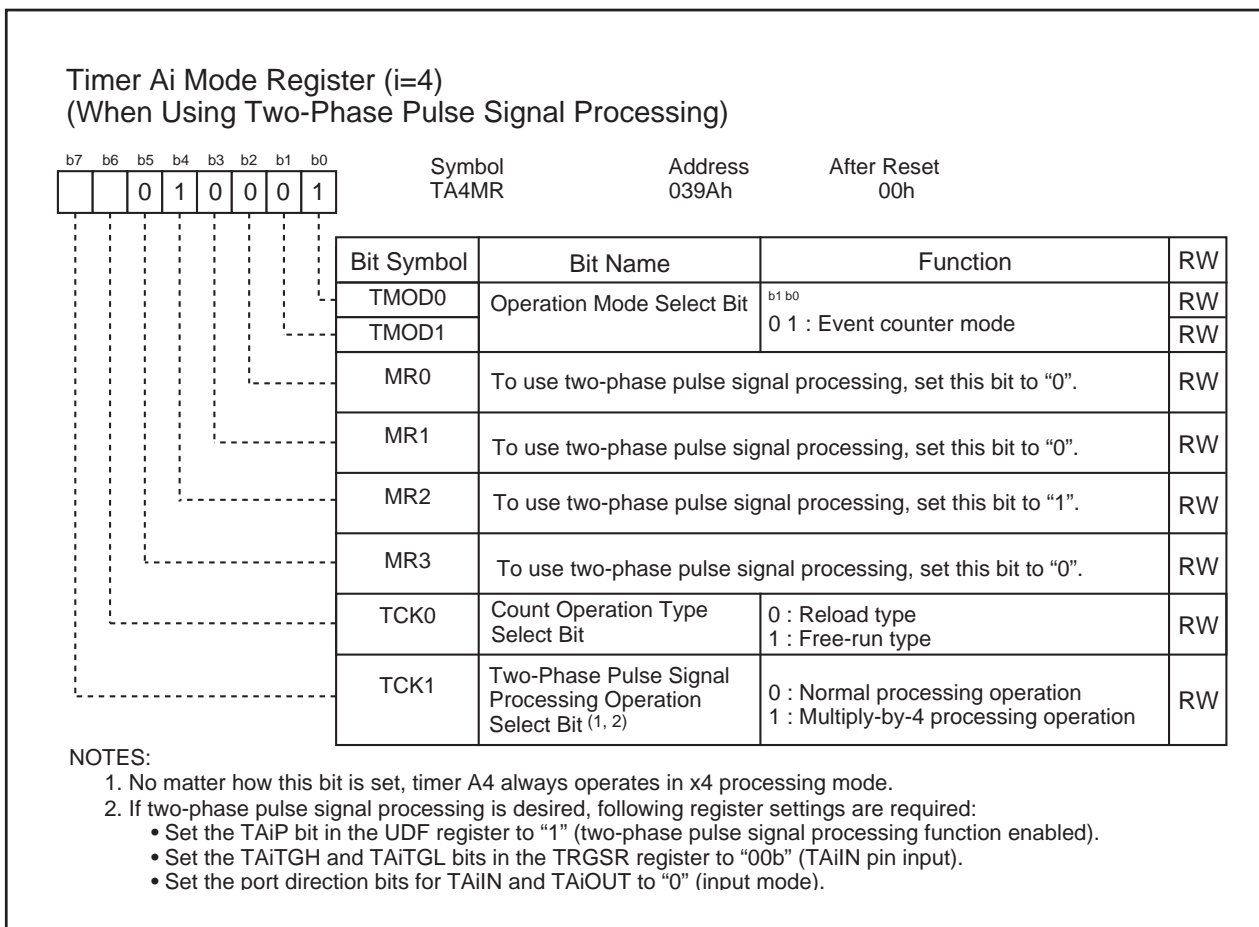


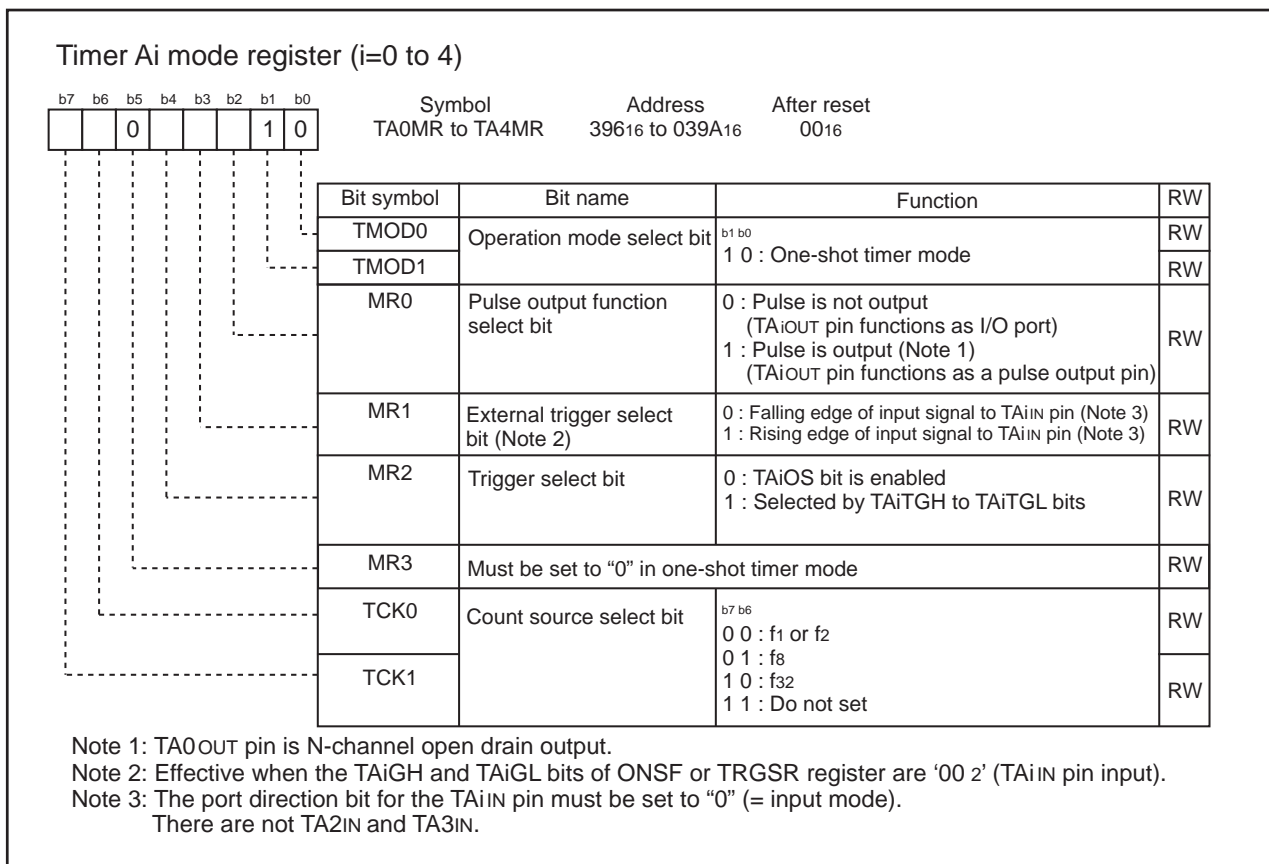
Figure 1.12.8. TA4MR Registers in Event Counter Mode (when using two-phase pulse signal processing with timer A4)

### 3. One-shot Timer Mode

In one-shot timer mode, the timer is activated only once by one trigger. (See Table 1.12.4.) When the trigger occurs, the timer starts up and continues operating for a given period. Figure 1.12.9 shows the TAI<sub>i</sub>MR register in one-shot timer mode.

**Table 1.12.4. Specifications in One-shot Timer Mode**

Item	Specification
Count source	f <sub>1</sub> , f <sub>2</sub> , f <sub>8</sub> , f <sub>32</sub>
Count operation	<ul style="list-style-type: none"> <li>Down-count</li> <li>When the counter reaches 0000<sub>16</sub>, it stops counting after reloading a new value</li> <li>If a trigger occurs when counting, the timer reloads a new count and restarts counting</li> </ul>
Divide ratio	1/n    n : set value of TAI register    0000 <sub>16</sub> to FFFF <sub>16</sub> However, the counter does not work if the divide-by-n value is set to 0000 <sub>16</sub> .
Count start condition	<p>TAiS bit of TABSR register = "1" (start counting) and one of the following triggers occurs.</p> <ul style="list-style-type: none"> <li>External trigger input from the TAI<sub>i</sub>N pin</li> <li>timer A<sub>j</sub> (j=i-1, except j=4 if i=0) overflow or underflow, timer A<sub>k</sub> (k=i+1, except k=0 if i=4) overflow or underflow</li> <li>The TAIOS bit of ONSF register is set to "1" (= timer starts)</li> </ul>
Count stop condition	<ul style="list-style-type: none"> <li>When the counter is reloaded after reaching "0000<sub>16</sub>"</li> <li>TAiS bit is set to "0" (= stop counting)</li> </ul>
Interrupt request generation timing	When the counter reaches "0000 <sub>16</sub> "
TAI <sub>i</sub> N pin function	I/O port or trigger input
TAI <sub>i</sub> OUT pin function	I/O port or pulse output
Read from timer	An indeterminate value is read by reading TAI register
Write to timer	<ul style="list-style-type: none"> <li>When not counting and until the 1st count source is input after counting start Value written to TAI register is written to both reload register and counter</li> <li>When counting (after 1st count source input) Value written to TAI register is written to only reload register (Transferred to counter when reloaded next)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>Pulse output function The timer outputs a low when not counting and a high when counting.</li> </ul>

Figure 1.12.9. TA<sub>i</sub>MR Register in One-shot Timer Mode



#### 4. Pulse Width Modulation (PWM) Mode

In PWM mode, the timer outputs pulses of a given width in succession (see Table 1.12.5). The counter functions as either 16-bit pulse width modulator or 8-bit pulse width modulator. Figure 1.12.10 shows TAI<sub>MR</sub> register in pulse width modulation mode. Figures 1.12.11 and 1.12.12 show examples of how a 16-bit pulse width modulator operates and how an 8-bit pulse width modulator operates.

**Table 1.12.5. Specifications in PWM Mode**

Item	Specification
Count source	f <sub>1</sub> , f <sub>2</sub> , f <sub>8</sub> , f <sub>32</sub>
Count operation	<ul style="list-style-type: none"> <li>Down-count (operating as an 8-bit or a 16-bit pulse width modulator)</li> <li>The timer reloads a new value at a rising edge of PWM pulse and continues counting</li> <li>The timer is not affected by a trigger that occurs during counting</li> </ul>
16-bit PWM	<ul style="list-style-type: none"> <li>High level width <math>n / f_j</math> n : set value of TAI register (i=0 to 4)</li> <li>Cycle time <math>(2^{16}-1) / f_j</math> fixed f<sub>j</sub>: count source frequency (f<sub>1</sub>, f<sub>2</sub>, f<sub>8</sub>, f<sub>32</sub>)</li> </ul>
8-bit PWM	<ul style="list-style-type: none"> <li>High level width <math>n \times (m+1) / f_j</math> n : set value of TAI<sub>MR</sub> register high-order address</li> <li>Cycle time <math>(2^8-1) \times (m+1) / f_j</math> m : set value of TAI<sub>MR</sub> register low-order address</li> </ul>
Count start condition	<ul style="list-style-type: none"> <li>TAiS bit of TABSR register is set to "1" (= start counting)</li> <li>The TAiS bit = 1 and external trigger input from the TAI<sub>IN</sub> pin</li> <li>The TAiS bit = 1 and one of the following external triggers occurs Timer A<sub>j</sub> (j=i-1, except j=4 if i=0) overflow or underflow, Timer A<sub>k</sub> (k=i+1, except k=0 if i=4) overflow or underflow</li> </ul>
Count stop condition	TAiS bit is set to "0" (= stop counting)
Interrupt request generation timing	PWM pulse goes "L"
TAi <sub>IN</sub> pin function	I/O port or trigger input
TAi <sub>OUT</sub> pin function	Pulse output
Read from timer	An indeterminate value is read by reading TAI register
Write to timer	<ul style="list-style-type: none"> <li>When not counting and until the 1st count source is input after counting start Value written to TAI register is written to both reload register and counter</li> <li>When counting (after 1st count source input) Value written to TAI register is written to only reload register (Transferred to counter when reloaded next)</li> </ul>

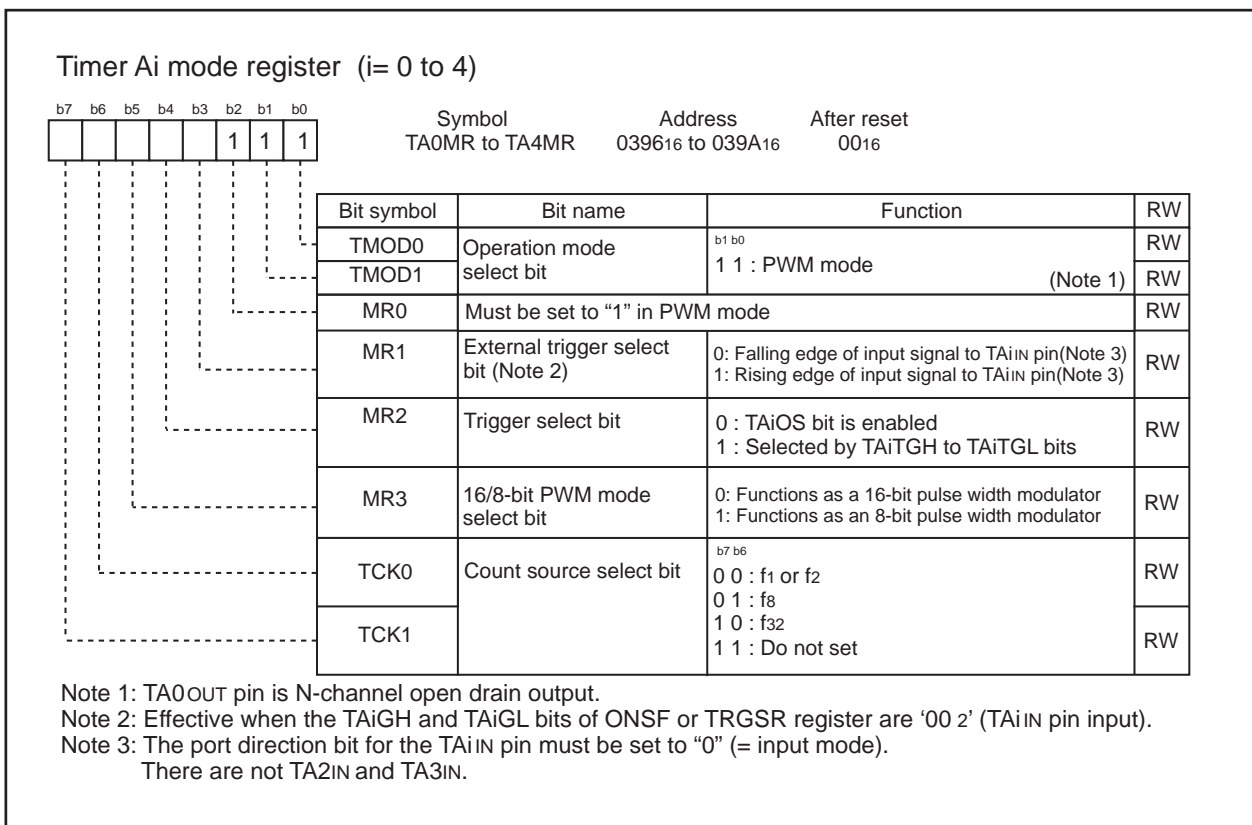


Figure 1.12.10. TAI<sub>i</sub>MR Register in PWM Mode

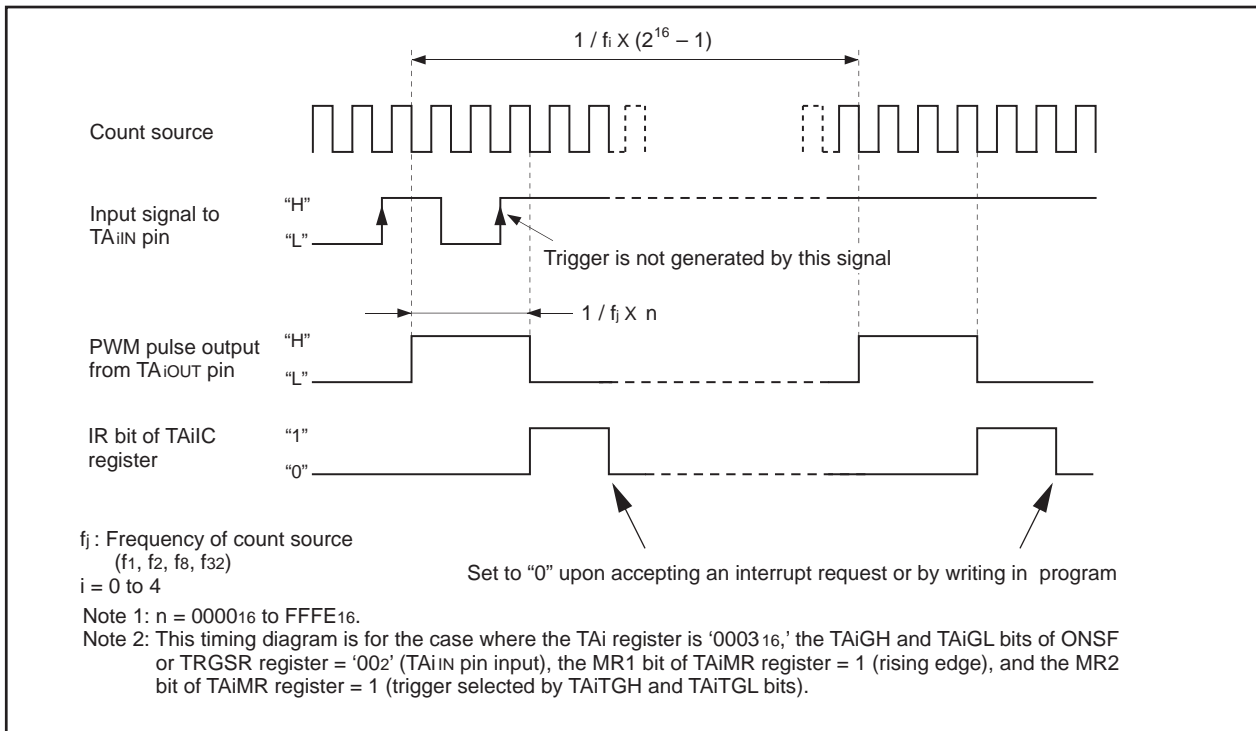


Figure 1.12.11. Example of 16-bit Pulse Width Modulator Operation

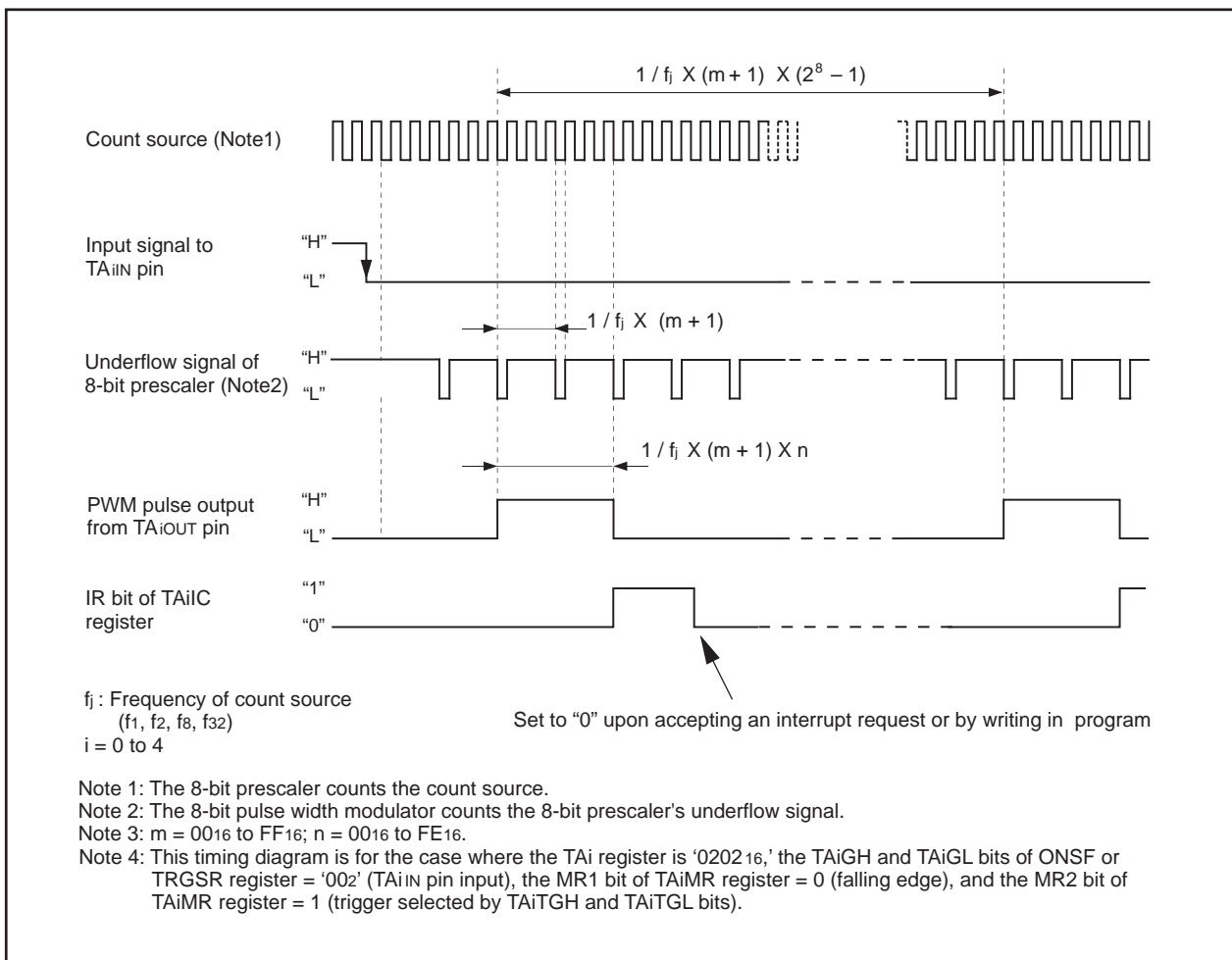


Figure 1.12.12. Example of 8-bit Pulse Width Modulator Operation

## Serial I/O

Serial I/O is configured with five channels: UART0 to UART2, SI/O3 and SI/O4.

### UARTi (i=0 to 2)

UARTi each have an exclusive timer to generate a transfer clock, so they operate independently of each other.

Figure 1.13.1 shows the block diagram of UARTi. Figure 1.13.2 shows the block diagram of the UARTi transmit/receive unit.

UARTi has the following modes:

- Clock synchronous serial I/O mode
- Clock asynchronous serial I/O mode (UART mode).
- Special mode 1 (I<sup>2</sup>C mode)
- Special mode 2

Figures 1.13.3 to 1.13.8 show the UARTi-related registers.

Refer to tables listing each mode for register setting.

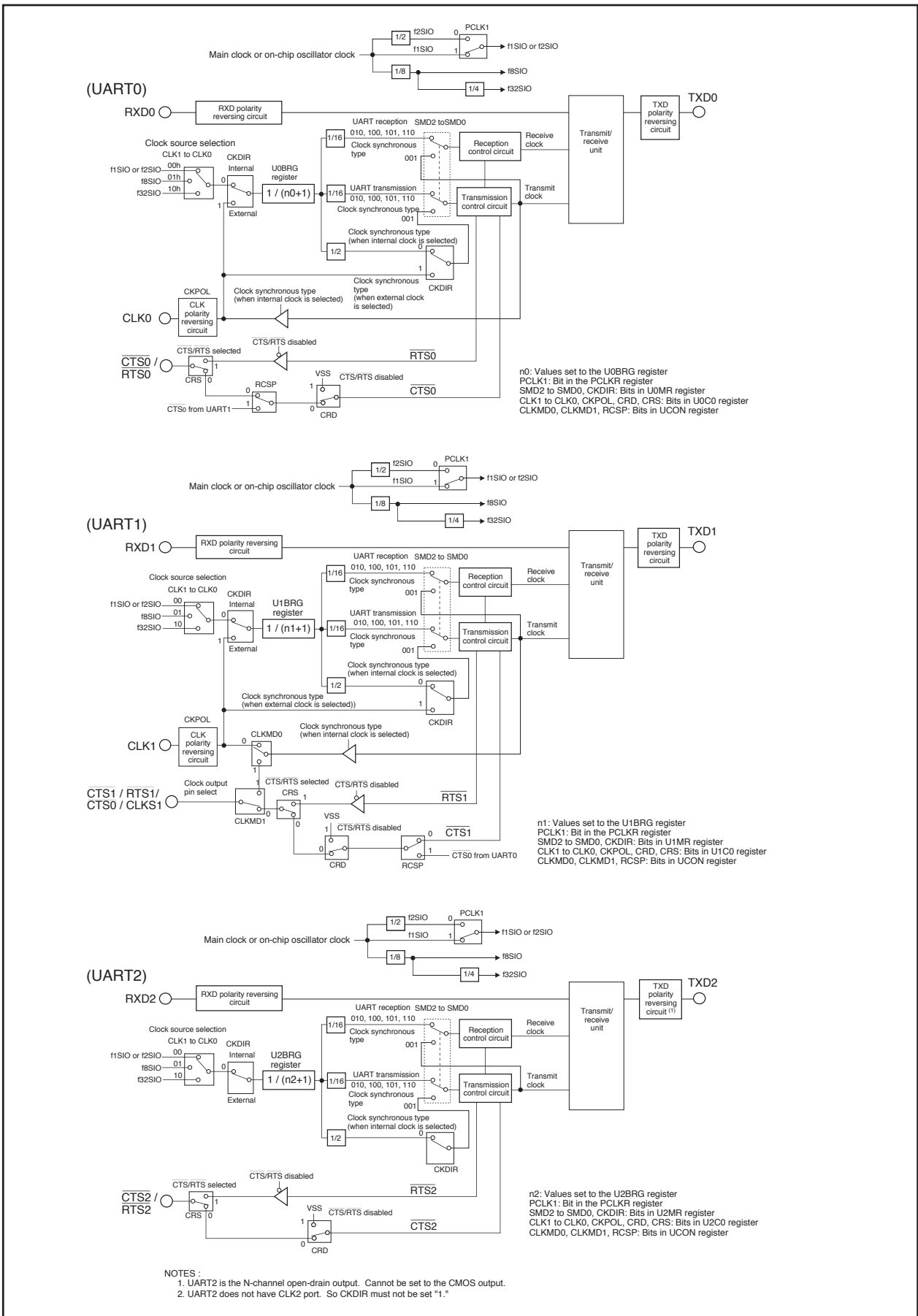


Figure 1.13.1. UARTi Block Diagram

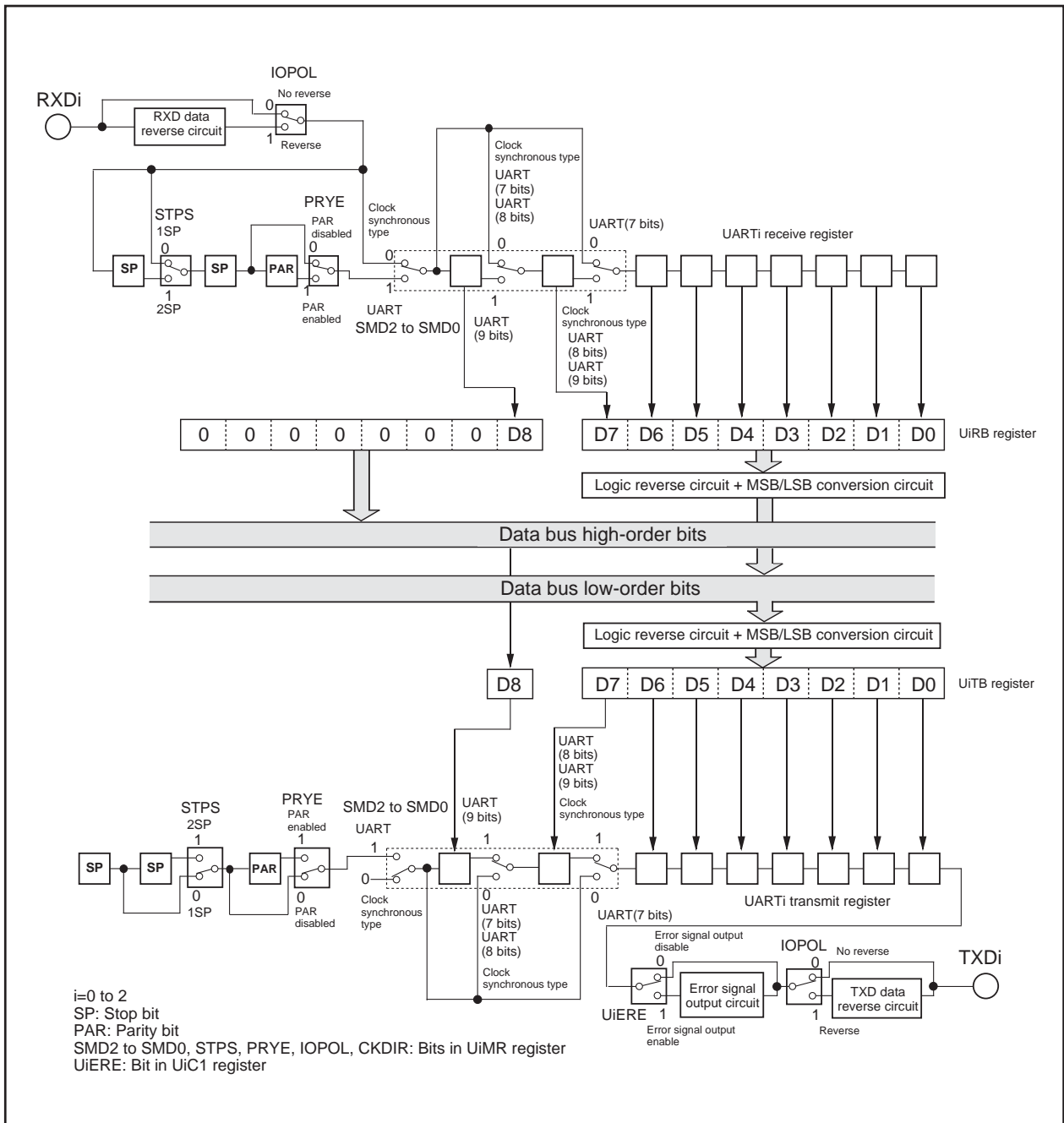


Figure 1.13.2. UARTi Transmit/Receive Unit

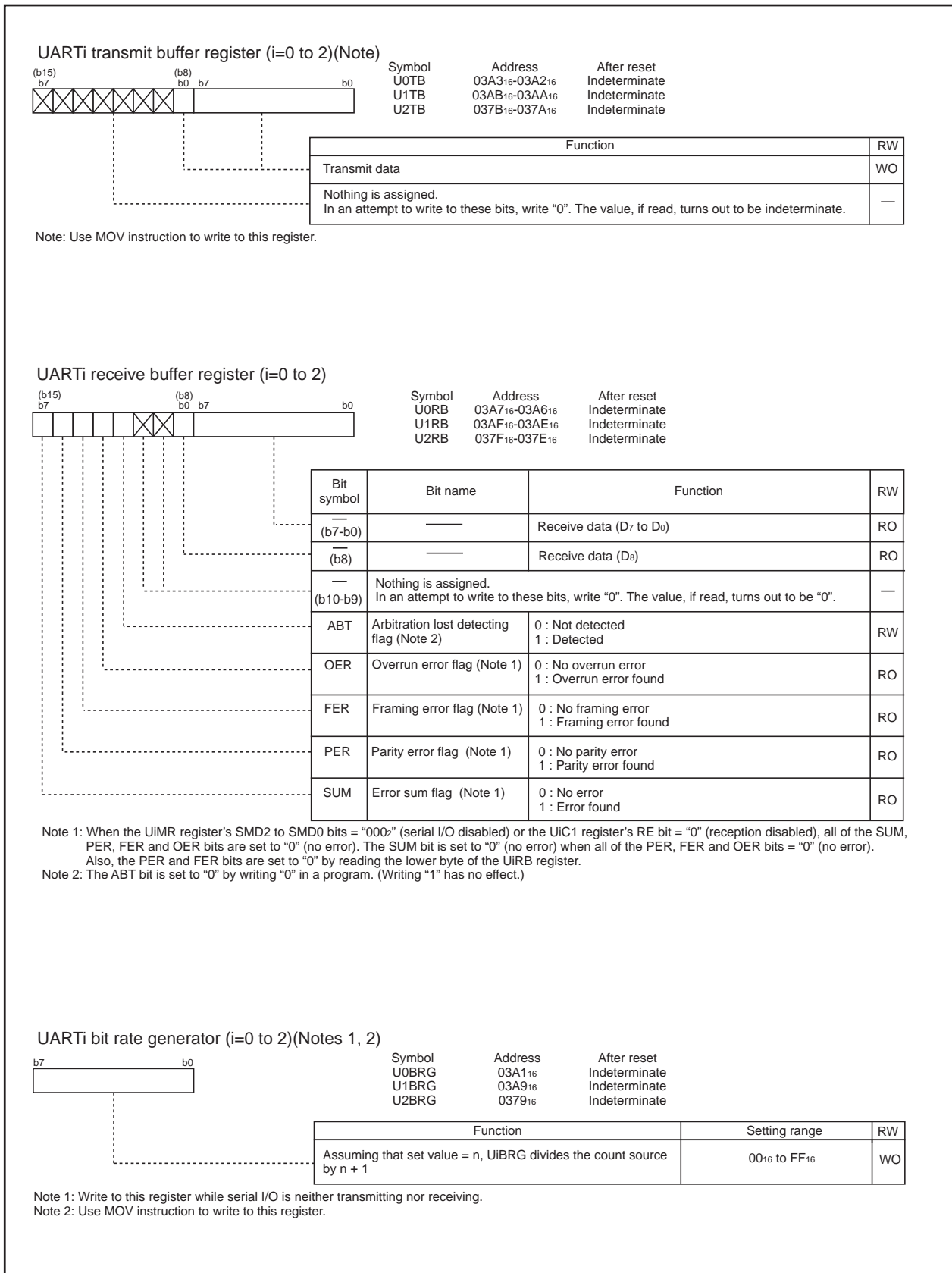


Figure 1.13.3. U0TB to U2TB Register, U0RB to U2RB Register, and U0BRG to U2BRG Register

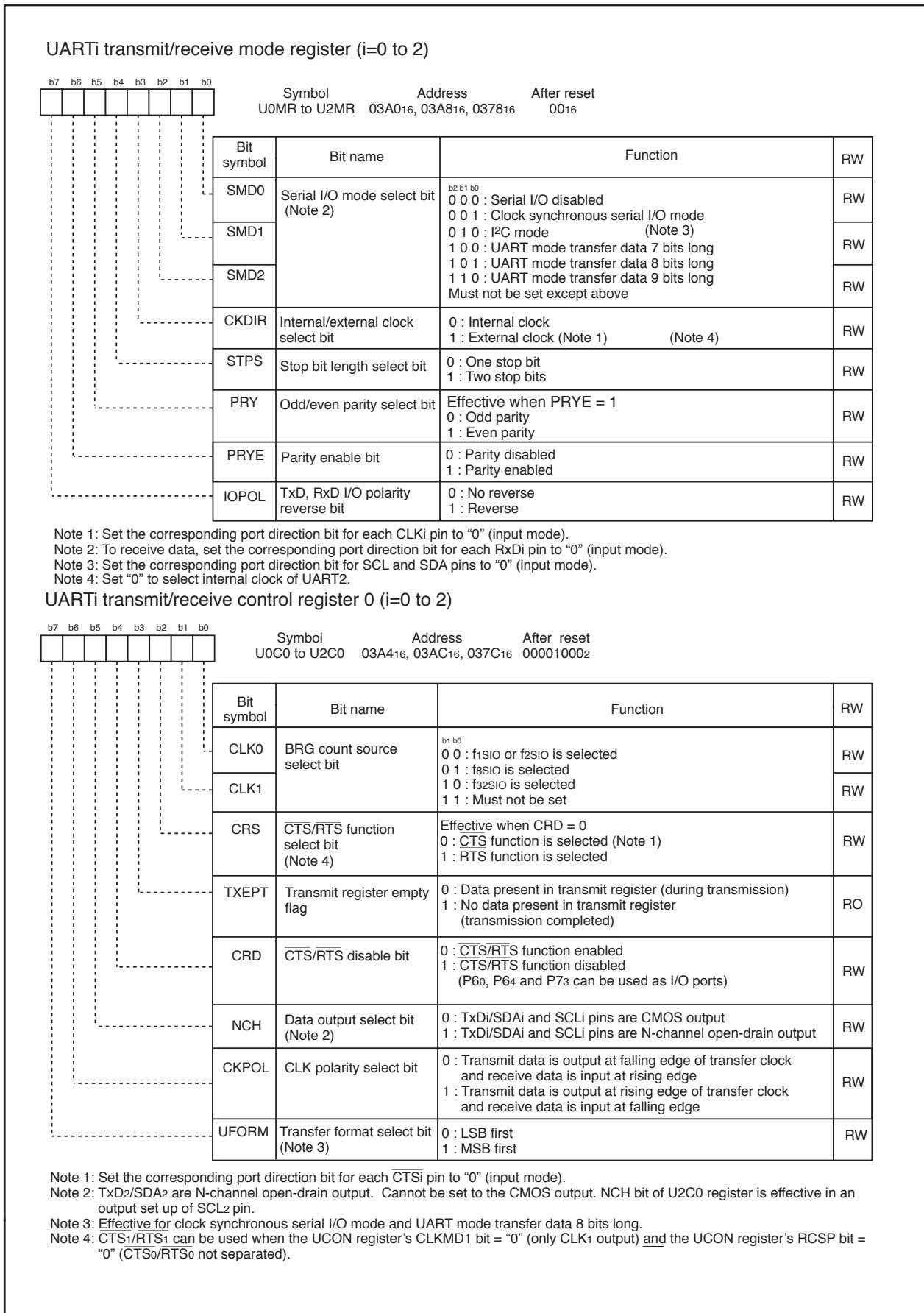


Figure 1.13.4. U0MR to U2MR Register and U0C0 to U2C0 Register



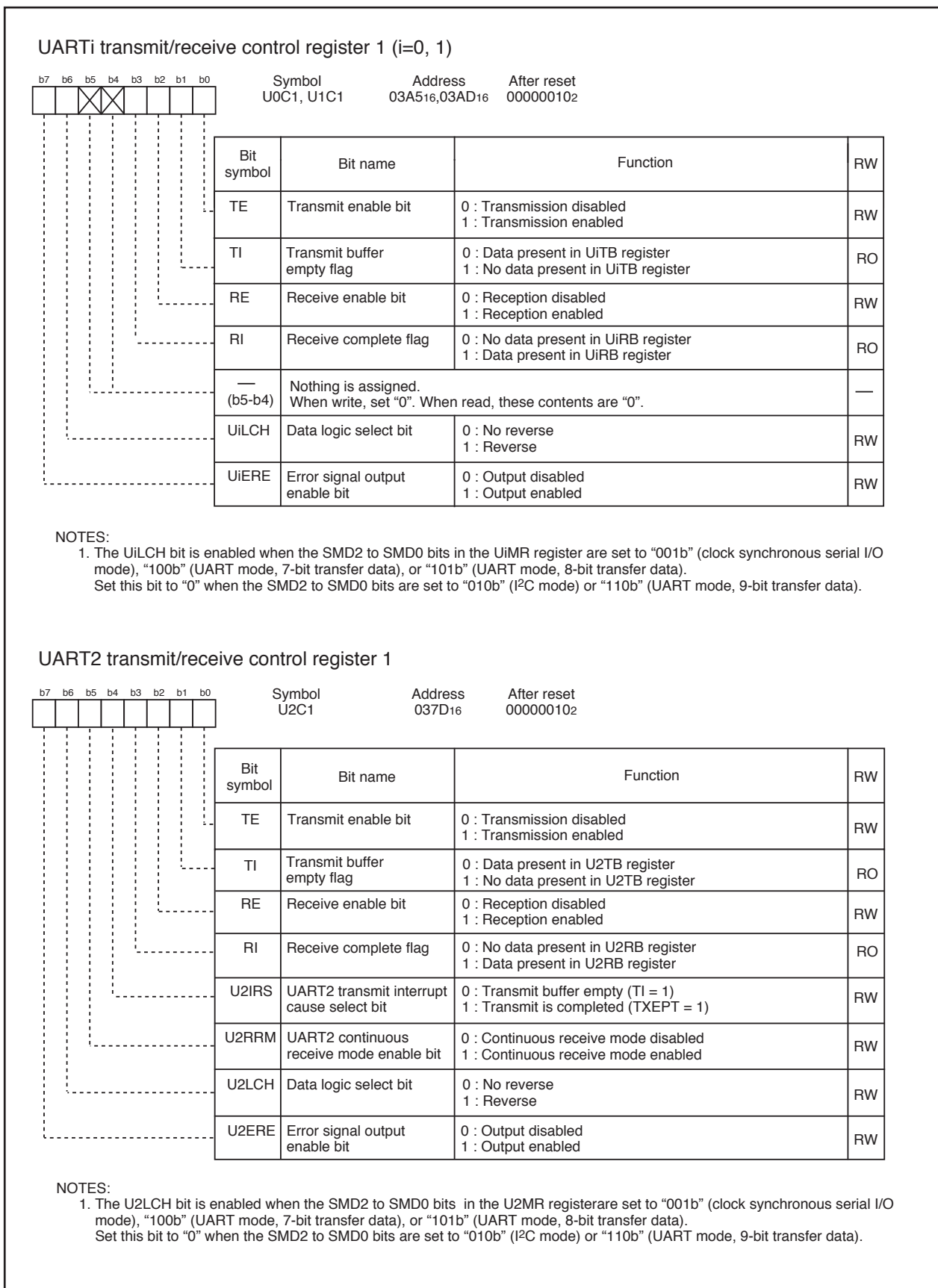


Figure 1.13.5. U0C1 to U2C1 Registers

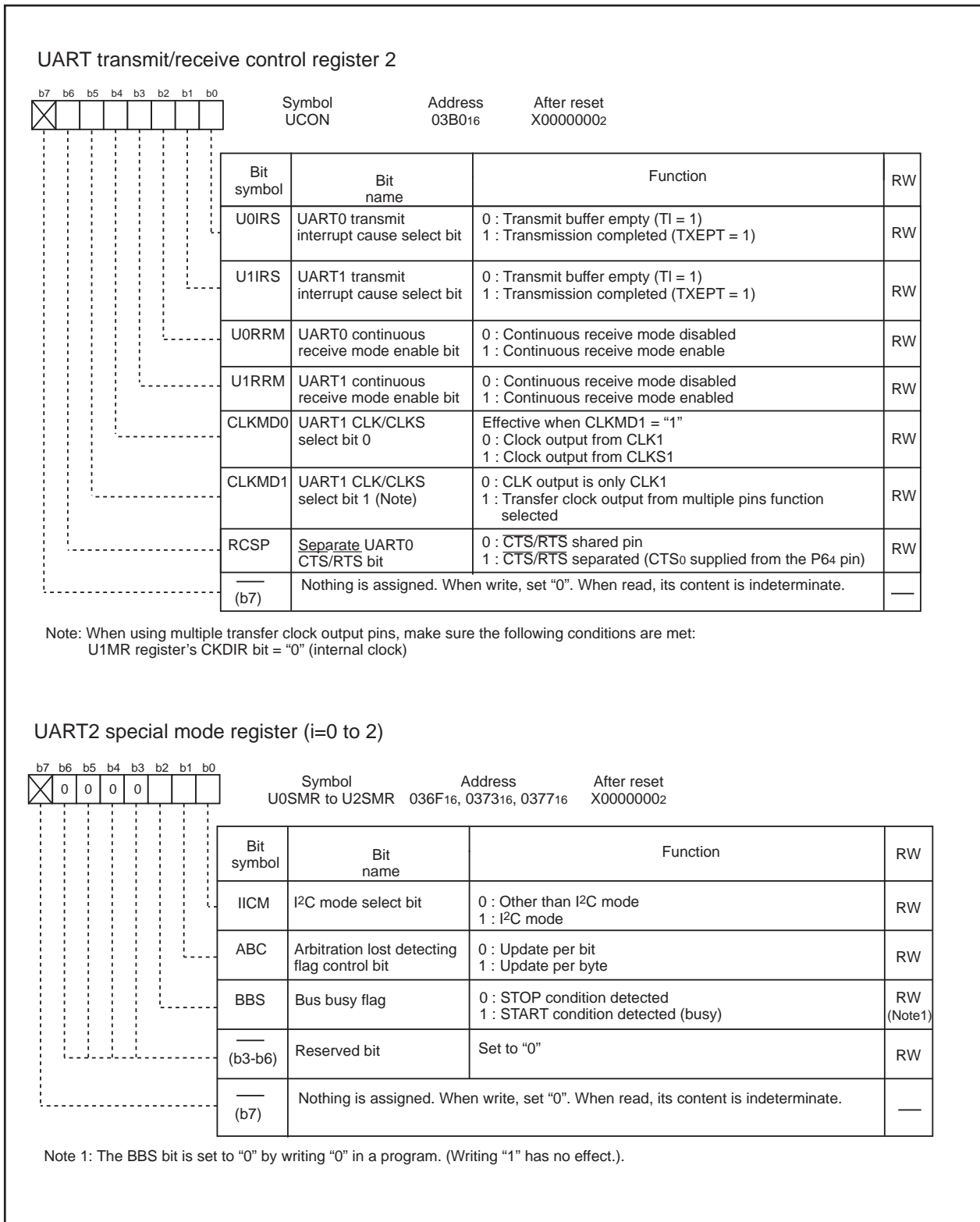


Figure 1.13.6. UCON Register and U0SMR to U2SMR Registers

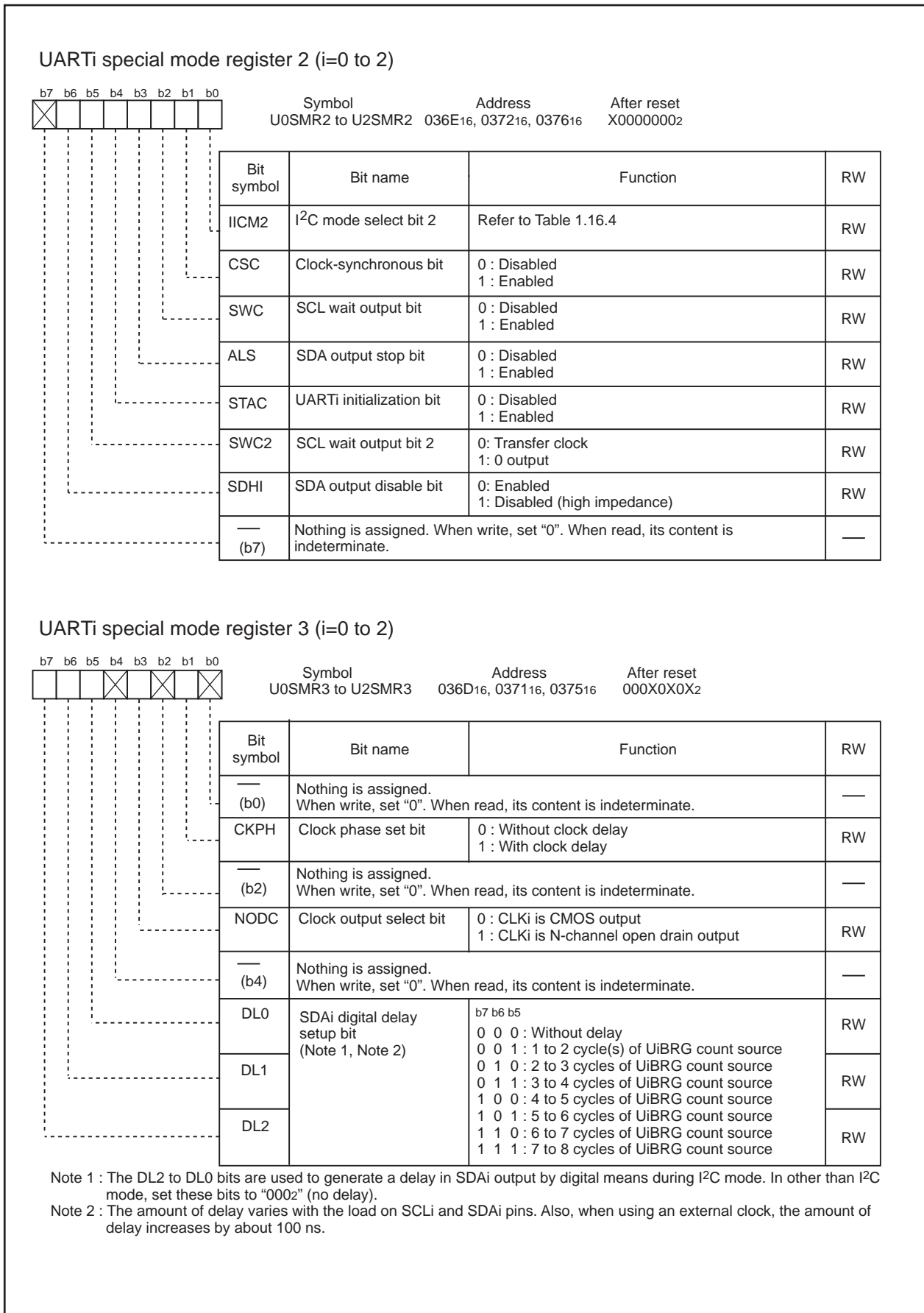


Figure 1.13.7. U0SMR2 to U2SMR2 Registers and U0SMR3 to U2SMR3 Registers

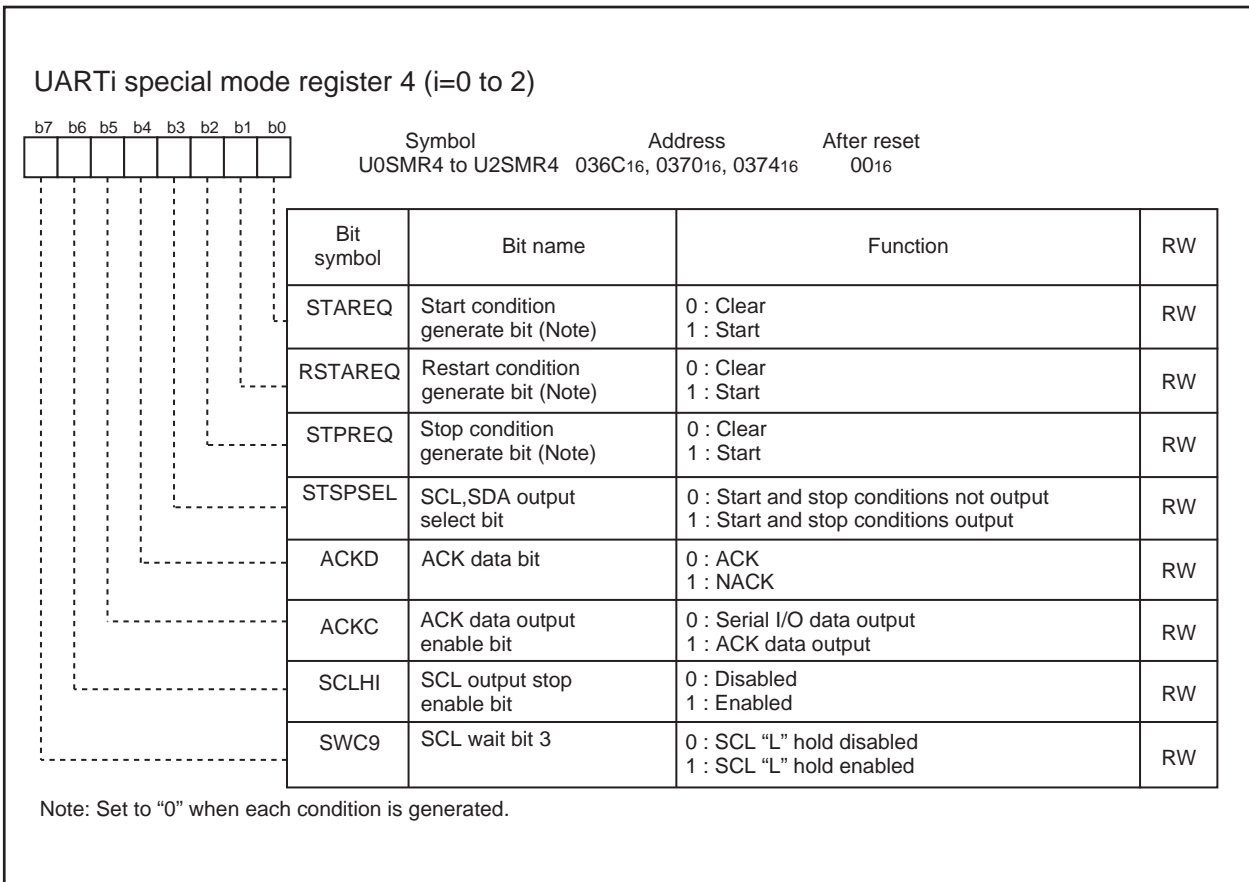


Figure 1.13.8. U0SMR4 to U2SMR4 Registers

## Clock Synchronous serial I/O Mode

The clock synchronous serial I/O mode uses a transfer clock to transmit and receive data. Table 1.14.1 lists the specifications of the clock synchronous serial I/O mode. Table 1.14.2 lists the registers used in clock synchronous serial I/O mode and the register values set. UART2 is not available in this mode.

**Table 1.14.1. Clock Synchronous Serial I/O Mode Specifications**

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• UiMR(i=0 to 1) register's CKDIR bit = "0" (internal clock) : $f_j / 2(n+1)$ $f_j = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}$ . n: Setting value of UiBRG register 0016 to FF16 • CKDIR bit = "1" (external clock) : Input from CLKi pin
Transmission, reception control	• Selectable from CTS function, RTS function or CTS/RTS function disable
Transmission start condition	• Before transmission can start, the following requirements must be met (Note 1) – The TE bit of UiC1 register= 1 (transmission enabled) – The TI bit of UiC1 register = 0 (data present in UiTB register) – If $\overline{CTS}$ function is selected, input on the $\overline{CTS}_i$ pin = "L"
Reception start condition	• Before reception can start, the following requirements must be met (Note 1) – The RE bit of UiC1 register= 1 (reception enabled) – The TE bit of UiC1 register= 1 (transmission enabled) – The TI bit of UiC1 register= 0 (data present in the UiTB register)
Interrupt request generation timing	• For transmission, one of the following conditions can be selected – The UiIRS bit (Note 3) = 0 (transmit buffer empty): when transferring data from the UiTB register to the UARTi transmit register (at start of transmission) – The UiIRS bit =1 (transfer completed): when the serial I/O finished sending data from the UARTi transmit register • For reception When transferring data from the UARTi receive register to the UiRB register (at completion of reception)
Error detection	• Overrun error (Note 2) This error occurs if the serial I/O started receiving the next data before reading the UiRB register and received the 7th bit of the next data
Select function	• CLK polarity selection Transfer data input/output can be chosen to occur synchronously with the rising or the falling edge of the transfer clock • LSB first, MSB first selection Whether to start sending/receiving data beginning with bit 0 or beginning with bit 7 can be selected • Continuous receive mode selection Reception is enabled immediately by reading the UiRB register • Switching serial data logic This function reverses the logic value of the transmit/receive data • Transfer clock output from multiple pins selection (UART1) The output pin can be selected in a program from two UART1 transfer clock pins that have been set • Separate $\overline{CTS}$ / $\overline{RTS}$ pins (UART0) $\overline{CTS}_0$ and $\overline{RTS}_0$ are input/output from separate pins

Note 1: When an external clock is selected, the conditions must be met while if the UiC0 register's CKPOL bit = "0" (transmit data output at the falling edge and the receive data taken in at the rising edge of the transfer clock), the external clock is in the high state; if the UiC0 register's CKPOL bit = "1" (transmit data output at the rising edge and the receive data taken in at the falling edge of the transfer clock), the external clock is in the low state.

Note 2: If an overrun error occurs, the value of UiRB register will be indeterminate. The IR bit of SiRIC register does not change.

Note 3: The U0IRS and U1IRS bits respectively are the UCON register bits 0 and 1.

**Table 1.14.2. Registers to Be Used and Settings in Clock Synchronous Serial I/O Mode**

Register	Bit	Function
UiTB(Note3)	0 to 7	Set transmission data
UiRB(Note3)	0 to 7	Reception data can be read
	OER	Overrun error flag
UiBRG	0 to 7	Set a transfer rate
UiMR(Note3)	SMD2 to SMD0	Set to "0012"
	CKDIR	Select the internal clock or external clock
	IOPOL	Set to "0"
UiC0	CLK1 to CLK0	Select the count source for the UiBRG register
	CRS	Select CTS or RTS to use
	TXEPT	Transmit register empty flag
	CRD	Enable or disable the CTS or RTS function
	NCH	Select TxDi pin output mode
	CKPOL	Select the transfer clock polarity
	UFORM	Select the LSB first or MSB first
UiC1	TE	Set this bit to "1" to enable transmission/reception
	TI	Transmit buffer empty flag
	RE	Set this bit to "1" to enable reception
	RI	Reception complete flag
	U2IRS (Note 1)	Select the source of UART2 transmit interrupt
	U2RRM (Note 1)	Set this bit to "1" to use continuous receive mode
	UiLCH	Set this bit to "1" to use inverted data logic
	UiERE	Set to "0"
UiSMR	0 to 7	Set to "0"
UiSMR2	0 to 7	Set to "0"
UiSMR3	0 to 2	Set to "0"
	NODC	Select clock output mode
	4 to 7	Set to "0"
UiSMR4	0 to 7	Set to "0"
UCON	U0IRS, U1IRS	Select the source of UART0/UART1 transmit interrupt
	U0RRM, U1RRM	Set this bit to "1" to use continuous receive mode
	CLKMD0	Select the transfer clock output pin when CLKMD1 = 1
	CLKMD1	Set this bit to "1" to output UART1 transfer clock from two pins
	RCSP	Set this bit to "1" to accept as input the UART0 CTS <sub>0</sub> signal from the P64 pin
	7	Set to "0"

Note 1: Set the U0C1 and U1C1 register bit 4 and bit 5 to "0". The U0IRS, U1IRS, U0RRM and U1RRM bits are in the UCON register.

Note 2: Not all register bits are described above. Set those bits to "0" when writing to the registers in clock synchronous serial I/O mode.

i=0 to 1

Table 1.14.3 lists the functions of the input/output pins during clock synchronous serial I/O mode. Table 1.14.3 shows pin functions for the case where the multiple transfer clock output pin select function is deselected. Table 1.14.4 lists the P64 pin functions during clock synchronous serial I/O mode. Note that for a period from when the UARTi operation mode is selected to when transfer starts, the TxDi pin outputs an “H”. (If the N-channel open-drain output is selected, this pin is in a high-impedance state.)

**Table 1.14.3. Pin Functions (When Not Select Multiple Transfer Clock Output Pin Function)**

Pin name	Function	Method of selection
TxDi (i = 0 to 2) (P63, P67)	Serial data output	(Outputs dummy data when performing reception only)
RxDi (P62, P66)	Serial data input	PD6 register's PD6_2 bit=0, PD6_6 bit=0 (Can be used as an input port when performing transmission only)
CLKi (P61, P65)	Transfer clock output	UiMR register's CKDIR bit=0
	Transfer clock input	UiMR register's CKDIR bit=1 PD6 register's PD6_1 bit=0, PD6_5 bit=0
CTS $\bar$ /RTSi (P60, P64)	CTS $\bar$ input	UiC0 register's CRD bit=0 UiC0 register's CRS bit=0 PD6 register's PD6_0 bit=0, PD6_4 bit=0
	RTS output	UiC0 register's CRD bit=0 UiC0 register's CRS bit=1
	I/O port	UiC0 register's CRD bit=1

**Table 1.14.4. P64 Pin Functions**

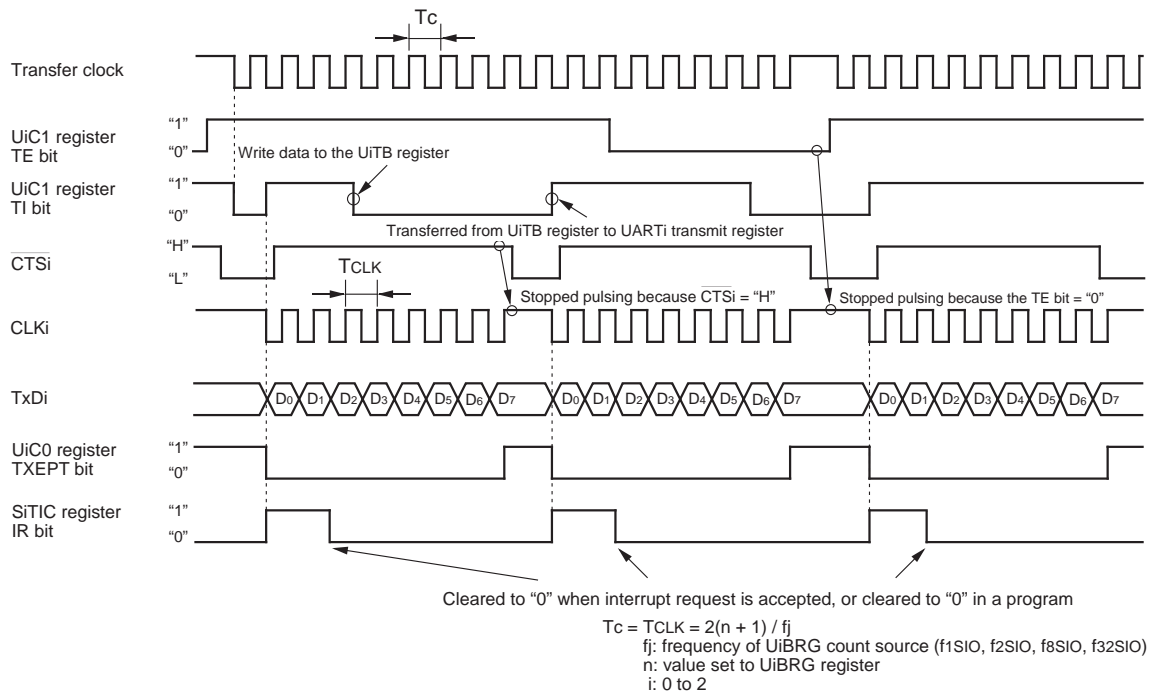
Pin function	Bit set value					
	U1C0 register		UCON register			PD6 register
	CRD	CRS	RCSP	CLKMD1	CLKMD0	PD6_4
P64	1	—	0	0	—	Input: 0, Output: 1
CTS $\bar$ 1	0	0	0	0	—	0
RTS1	0	1	0	0	—	—
CTS0(Note1)	0	0	1	0	—	0
CLKS1	—	—	—	1(Note 2)	1	—

Note 1: In addition to this, set the UOC0 register's CRD bit to “0” (CTS $\bar$ 0/RTS0 enabled) and the U0C0 register's CRS bit to “1” (RTS0 selected).

Note 2: When the CLKMD1 bit = 1 and the CLKMD0 bit = 0, the following logic levels are output:

- High if the U1C0 register's CLKPOL bit = 0
- Low if the U1C0 register's CLKPOL bit = 1

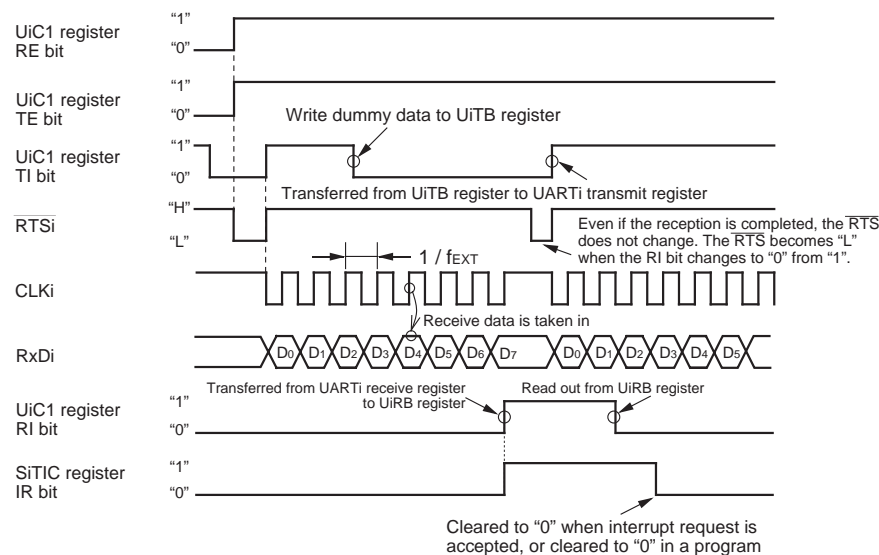
(1) Example of transmit timing (when internal clock is selected)



The above timing diagram applies to the case where the register bits are set as follows:

- UiMR register CKDIR bit = 0 (internal clock)
- UiC0 register CRD bit = 0 (CTS/RTS enabled), CRS bit = 0 (CTS selected)
- UiC0 register CKPOL bit = 0 (transmit data output at the falling edge and receive data taken in at the rising edge of the transfer clock)
- UIRS bit = 0 (an interrupt request occurs when the transmit buffer becomes empty): U0IRS bit is the UCON register bit 0, U1IRS bit is the UCON register bit 1, and U2IRS bit is the U2C1 register bit 4

(2) Example of receive timing (when external clock is selected)



The above timing diagram applies to the case where the register bits are set as follows:

- UiMR register CKDIR bit = 1 (external clock)
- UiC0 register CRD bit = 0 (CTS/RTS enabled), CRS bit = 1 (RTS selected)
- UiC0 register CKPOL bit = 0 (transmit data output at the falling edge and receive data taken in at the rising edge of the transfer clock)

Make sure the following conditions are met when input to the CLKi pin before receiving data is high:

- UiC0 register TE bit = 1 (transmit enabled)
- UiC0 register RE bit = 1 (Receive enabled)
- Write dummy data to the UiTB register

fEXT: frequency of external clock

Figure 1.14.1. Transmit and Receive Operation



### Counter Measure for Communication Error Occurs

If a communication error occurs while transmitting or receiving in clock synchronous serial I/O mode, follow the procedures below.

- Resetting the UiRB register (i=0 to 2)

- (1) Set the RE bit in the UiC1 register to "0" (reception disabled)

- (2) Set the SMD2 to SMD0 bits in the UiMR register to "000b" (Serial I/O disabled)

- (3) Set the SMD2 to SMD0 bits in the UiMR register to "001b" (Clock synchronous serial I/O mode)

- (4) Set the RE bit in the UiC1 register to "1" (reception enabled)

- Resetting the UiTB register (i=0 to 2)

- (1) Set the SMD2 to SMD0 bits in the UiMR register "000b" (Serial I/O disabled)

- (2) Set the SMD2 to SMD0 bits in the UiMR register "001b" (Clock synchronous serial I/O mode)

- (3) "1" is written to RE bit in the UiC1 register (reception enabled), regardless of the TE bit in the UiCi register

### (a) CLK Polarity Select Function

Use the UiC0 register (i = 0 to 1)'s CKPOL bit to select the transfer clock polarity. Figure 1.14.2 shows the polarity of the transfer clock.

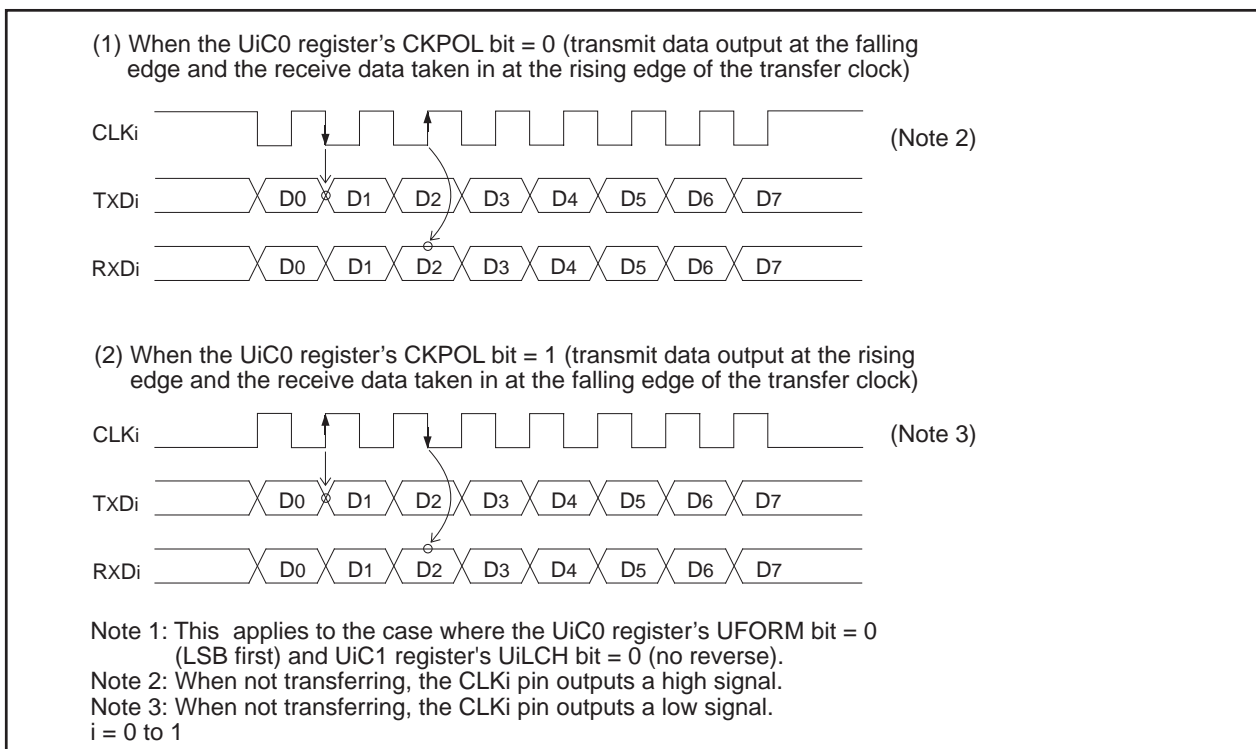
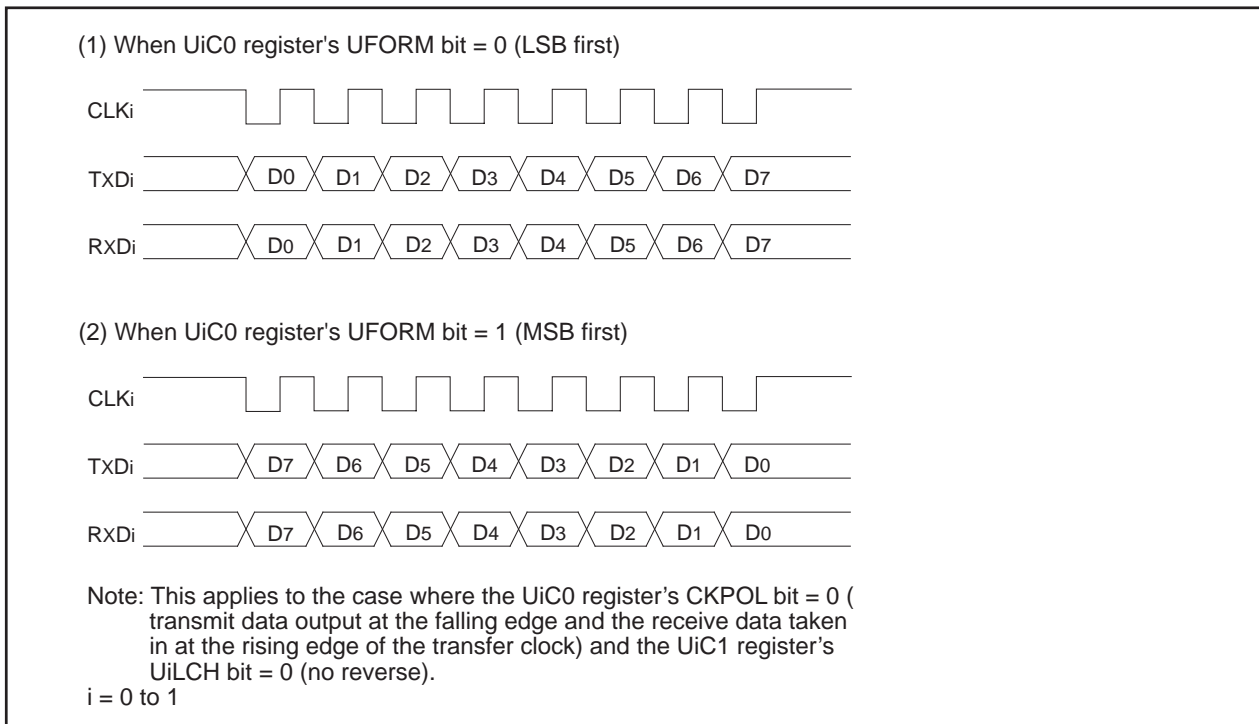


Figure 1.14.2. Transfer Clock Polarity

**(b) LSB First/MSB First Select Function**

Use the UiC0 register (i = 0 to 1)'s UFORM bit to select the transfer format. Figure 1.14.3 shows the transfer format.



**Figure 1.14.3. Transfer Format**

**(c) Continuous Receive Mode**

In continuous receive mode, receive operation becomes enable when the receive buffer register is read. It is not necessary to write dummy data into the transmit buffer register to enable receive operation in this mode. However, a dummy read of the receive buffer register is required when starting the operation mode.

When the UiRRM bit (i = 0 to 1) = 1 (continuous receive mode), the UiC1 register's TI bit is set to "1" (data present in the UiTB register) by reading the UiRB register. In this case, i.e., UiRRM bit = 1, do not write dummy data to the UiTB register in a program. The U0RRM and U1RRM bits are the UCON register bit 2 and bit 3, respectively.

**(d) Serial Data Logic Switching Function**

When the UiC1 register (i = 0 to 1)'s UiLCH bit = 1 (reverse), the data written to the UiTB register has its logic reversed before being transmitted. Similarly, the received data has its logic reversed when read from the UiRB register. Figure 1.14.4 shows serial data logic.

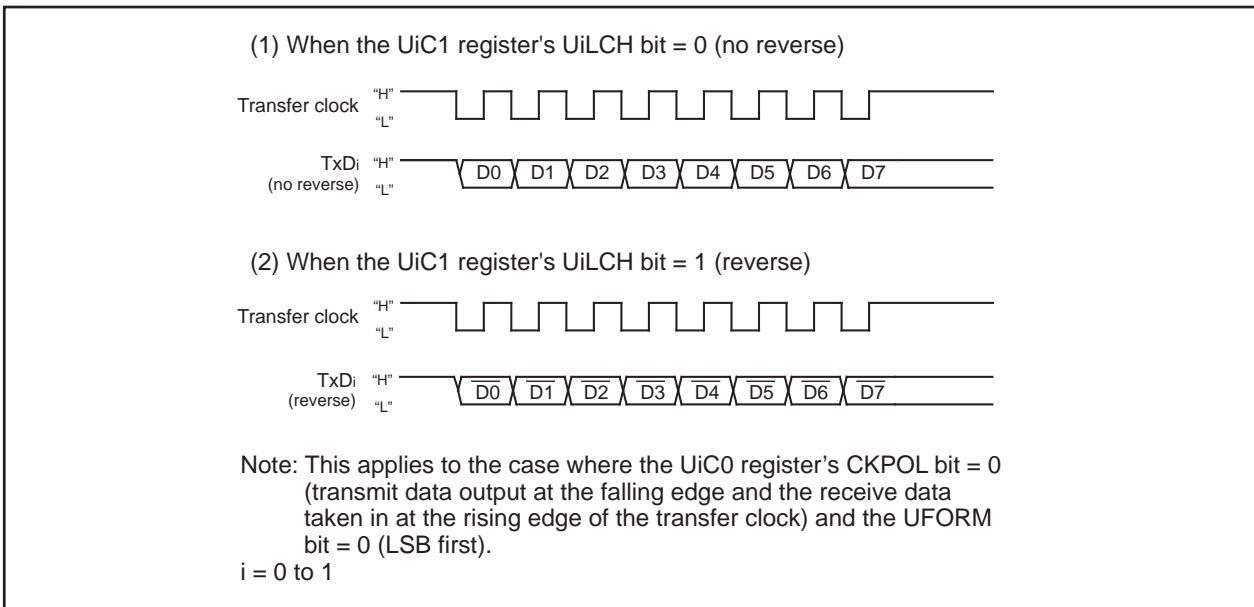


Figure 1.14.4. Serial Data Logic Switching

**(e) Transfer Clock Output From Multiple Pins (UART1)**

Use the UCON register's CLKMD1 to CLKMD0 bits to select one of the two transfer clock output pins. (See Figure 1.14.5.) This function can be used when the selected transfer clock for UART1 is an internal clock.

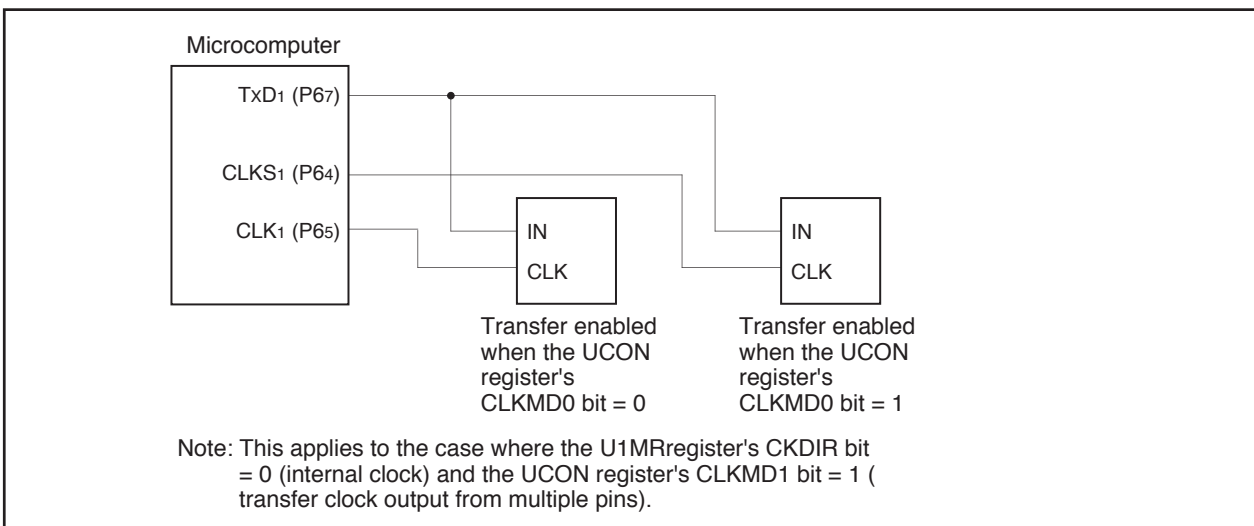


Figure 1.14.5. Transfer Clock Output From Multiple Pins

### $\overline{\text{CTS}}/\overline{\text{RTS}}$ Function

When the  $\overline{\text{CTS}}$  function is used transmit and receive operation start when “L” is applied to the  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  ( $i=0$  to 2) pin. Transmit and receive operation begins when the  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is held “L”. If the “L” signal is switched to “H” during a transmit or receive operation, the operation stops before the next data.

When the  $\overline{\text{RTS}}$  function is used, the  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin outputs on “L” signal when the microcomputer is ready to receive. The output level becomes “H” on the first falling edge of the  $\text{CLK}_i$  pin.

- CRD bit in UIC0 register = 1 (  $\overline{\text{CTS}}/\overline{\text{RTS}}$  function disabled)  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is programmable I/O function
- CRD bit = 0, CRS bit = 0 ( $\overline{\text{CTS}}$  function is selected)  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is  $\overline{\text{CTS}}$  function
- CRD bit = 0, CRS bit = 1 ( $\overline{\text{RTS}}$  function is selected)  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is  $\overline{\text{RTS}}$  function

### (f) $\overline{\text{CTS}}/\overline{\text{RTS}}$ Separate Function (UART0)

This function separates  $\overline{\text{CTS}}_0/\overline{\text{RTS}}_0$ , outputs  $\overline{\text{RTS}}_0$  from the P60 pin, and accepts as input the  $\overline{\text{CTS}}_0$  from the P64 pin. To use this function, set the register bits as shown below.

- U0C0 register's CRD bit = 0 (enables UART0  $\overline{\text{CTS}}/\overline{\text{RTS}}$ )
- U0C0 register's CRS bit = 1 (outputs UART0  $\overline{\text{RTS}}$ )
- U1C0 register's CRD bit = 0 (enables UART1  $\overline{\text{CTS}}/\overline{\text{RTS}}$ )
- U1C0 register's CRS bit = 0 (inputs UART1  $\overline{\text{CTS}}$ )
- UCON register's RCSP bit = 1 (inputs  $\overline{\text{CTS}}_0$  from the P64 pin)
- UCON register's CLKMD1 bit = 0 ( $\text{CLKS}_1$  not used)

Note that when using the  $\overline{\text{CTS}}/\overline{\text{RTS}}$  separate function, UART1  $\overline{\text{CTS}}/\overline{\text{RTS}}$  separate function cannot be used.

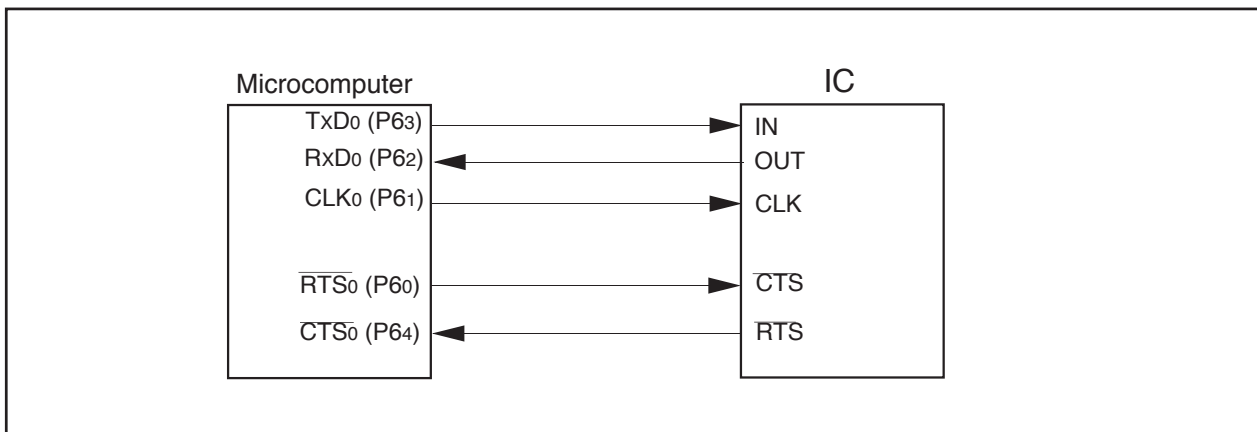


Figure 1.14.6.  $\overline{\text{CTS}}/\overline{\text{RTS}}$  Separate Function

## Clock Asynchronous Serial I/O (UART) Mode

The UART mode allows transmitting and receiving data after setting the desired transfer rate and transfer data format. Table 1.15.1 lists the specifications of the UART mode.

**Table 1.15.1. UART Mode Specifications**

Item	Specification
Transfer data format	<ul style="list-style-type: none"> <li>• Character bit (transfer data): Selectable from 7, 8 or 9 bits</li> <li>• Start bit: 1 bit</li> <li>• Parity bit: Selectable from odd, even, or none</li> <li>• Stop bit: Selectable from 1 or 2 bits</li> </ul>
Transfer clock	<ul style="list-style-type: none"> <li>• UiMR(i=0 to 2) register's CKDIR bit = 0 (internal clock) : <math>f_j / 16(n+1)</math>  <math>f_j = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}</math>. n: Setting value of UiBRG register 00<sub>16</sub> to FF<sub>16</sub></li> <li>• CKDIR bit = "1" (external clock) : <math>f_{EXT} / 16(n+1)</math> (Note 3)  <math>f_{EXT}</math>: Input from CLKi pin. n: Setting value of UiBRG register 00<sub>16</sub> to FF<sub>16</sub></li> </ul>
Transmission, reception control	<ul style="list-style-type: none"> <li>• Selectable from <math>\overline{CTS}</math> function, <math>\overline{RTS}</math> function or <math>\overline{CTS}/\overline{RTS}</math> function disable</li> </ul>
Transmission start condition	<ul style="list-style-type: none"> <li>• Before transmission can start, the following requirements must be met <ul style="list-style-type: none"> <li>– The TE bit of UiC1 register= 1 (transmission enabled)</li> <li>– The TI bit of UiC1 register = 0 (data present in UiTB register)</li> <li>– If <math>\overline{CTS}</math> function is selected, input on the <math>\overline{CTS}_i</math> pin = "L"</li> </ul> </li> </ul>
Reception start condition	<ul style="list-style-type: none"> <li>• Before reception can start, the following requirements must be met <ul style="list-style-type: none"> <li>– The RE bit of UiC1 register= 1 (reception enabled)</li> <li>– Start bit detection</li> </ul> </li> </ul>
Interrupt request generation timing	<ul style="list-style-type: none"> <li>• For transmission, one of the following conditions can be selected <ul style="list-style-type: none"> <li>– The UiIRS bit (Note 2) = 0 (transmit buffer empty): when transferring data from the UiTB register to the UARTi transmit register (at start of transmission)</li> <li>– The UiIRS bit =1 (transfer completed): when the serial I/O finished sending data from the UARTi transmit register</li> </ul> </li> <li>• For reception <ul style="list-style-type: none"> <li>When transferring data from the UARTi receive register to the UiRB register (at completion of reception)</li> </ul> </li> </ul>
Error detection	<ul style="list-style-type: none"> <li>• Overrun error (Note 1)  This error occurs if the serial I/O started receiving the next data before reading the UiRB register and received the bit one before the last stop bit of the next data</li> <li>• Framing error  This error occurs when the number of stop bits set is not detected</li> <li>• Parity error  This error occurs when if parity is enabled, the number of 1's in parity and character bits does not match the number of 1's set</li> <li>• Error sum flag  This flag is set (= 1) when any of the overrun, framing, and parity errors is encountered</li> </ul>
Select function	<ul style="list-style-type: none"> <li>• LSB first, MSB first selection  Whether to start sending/receiving data beginning with bit 0 or beginning with bit 7 can be selected</li> <li>• Serial data logic switch  This function reverses the logic of the transmit/receive data. The start and stop bits are not reversed.</li> <li>• TxD, RxD I/O polarity switch  This function reverses the polarities of the TxD pin output and RxD pin input. The logic levels of all I/O data is reversed.</li> <li>• Separate <math>\overline{CTS}/\overline{RTS}</math> pins (UART0)  <math>\overline{CTS}_0</math> and <math>\overline{RTS}_0</math> are input/output from separate pins</li> </ul>

Note 1: If an overrun error occurs, the value of UiRB register will be indeterminate. The IR bit of SiRIC register does not change.

Note 2: The U0IRS and U1IRS bits respectively are the UCON register bits 0 and 1; the U2IRS bit is the U2C1 register bit 4.

Note 3: CKDIR of U2MR must be set "0" to select internal clock.

**Table 1.15.2. Registers to Be Used and Settings in UART Mode**

Register	Bit	Function
UiTB	0 to 8	Set transmission data (Note 1)
UiRB	0 to 8	Reception data can be read (Note 1)
	OER,FER,PER,SUM	Error flag
UiBRG	0 to 7	Set a transfer rate
UiMR	SMD2 to SMD0	Set these bits to '100 <sub>2</sub> ' when transfer data is 7 bits long Set these bits to '101 <sub>2</sub> ' when transfer data is 8 bits long Set these bits to '110 <sub>2</sub> ' when transfer data is 9 bits long
	CKDIR	Select the internal clock or external clock
	STPS	Select the stop bit
	PRY, PRYE	Select whether parity is included and whether odd or even
	IOPOL	Select the TxD/RxD input/output polarity
UiC0	CLK0, CLK1	Select the count source for the UiBRG register
	CRS	Select CTS or RTS to use
	TXEPT	Transmit register empty flag
	CRD	Enable or disable the CTS or RTS function
	NCH	Select TxDi pin output mode (Note 2)
	CKPOL	Set to "0"
	UFORM	LSB first or MSB first can be selected when transfer data is 8 bits long. Set this bit to "0" when transfer data is 7 or 9 bits long.
UiC1	TE	Set this bit to "1" to enable transmission
	TI	Transmit buffer empty flag
	RE	Set this bit to "1" to enable reception
	RI	Reception complete flag
	U2IRS (Note 2)	Select the source of UART2 transmit interrupt
	U2RRM (Note 2)	Set to "0"
	UiLCH	Set this bit to "1" to use inverted data logic
	UiERE	Set to "0"
UiSMR	0 to 7	Set to "0"
UiSMR2	0 to 7	Set to "0"
UiSMR3	0 to 7	Set to "0"
UiSMR4	0 to 7	Set to "0"
UCON	U0IRS, U1IRS	Select the source of UART0/UART1 transmit interrupt
	U0RRM, U1RRM	Set to "0"
	CLKMD0	Invalid because CLKMD1 = 0
	CLKMD1	Set to "0"
	RCSP	Set this bit to "1" to accept as input the UART0 CTS <sub>0</sub> signal from the P64 pin
	7	Set to "0"

Note 1: The bits used for transmit/receive data are as follows: Bit 0 to bit 6 when transfer data is 7 bits long; bit 0 to bit 7 when transfer data is 8 bits long; bit 0 to bit 8 when transfer data is 9 bits long.

Note 2: Set the U0C1 and U1C1 registers bit 4 to bit 5 to "0". The U0IRS, U1IRS, U0RRM and U1RRM bits are included in the UCON register.

Note 3: TxD2 pin is N channel open-drain output. Set the U2C0 register's NCH bit to "0".

i=0 to 2

Table 1.15.3 lists the functions of the input/output pins during UART mode. Table 1.15.4 lists the P64 pin functions during UART mode. Note that for a period from when the UARTi operation mode is selected to when transfer starts, the TxDi pin outputs an “H”. (If the N-channel open-drain output is selected, this pin is in a high-impedance state.)

**Table 1.15.3. I/O Pin Functions**

Pin name	Function	Method of selection
TxDi (i = 0 to 2) (P63, P67, P70)	Serial data output	(Outputs dummy data when performing reception only)
RxDi (P62, P66, P71)	Serial data input	PD6 register's PD6_2 bit=0, PD6_6 bit=0, PD7 register's PD7_1 bit=0 (Can be used as an input port when performing transmission only)
CLKi (P61, P65)	Input/output port	UiMR register's CKDIR bit=0
	Transfer clock input	UiMR register's CKDIR bit=1 PD6 register's PD6_1 bit=0, PD6_5 bit=0, PD7 register's PD7_2 bit=0
$\overline{\text{CTS}}_i/\text{RTS}_i$ (P60, P64, P73)	$\overline{\text{CTS}}$ input	UiC0 register's CRD bit=0 UiC0 register's CRS bit=0 PD6 register's PD6_0 bit=0, PD6_4 bit=0, PD7 register's PD7_3 bit=0
	RTS output	UiC0 register's CRD bit=0 UiC0 register's CRS bit=1
	Input/output port	UiC0 register's CRD bit=1

**Table 1.15.4. P64 Pin Functions**

Pin function	Bit set value				
	U1C0 register		UCON register		PD6 register
	CRD	CRS	RCSP	CLKMD1	PD6_4
P64	1	—	0	0	Input: 0, Output: 1
$\overline{\text{CTS}}_1$	0	0	0	0	0
$\text{RTS}_1$	0	1	0	0	—
$\overline{\text{CTS}}_0$ (Note)	0	0	1	0	0

Note: In addition to this, set the U0C0 register's CRD bit to “0” ( $\overline{\text{CTS}}_0/\text{RTS}_0$  enabled) and the U0C0 register's CRS bit to “1” ( $\text{RTS}_0$  selected).

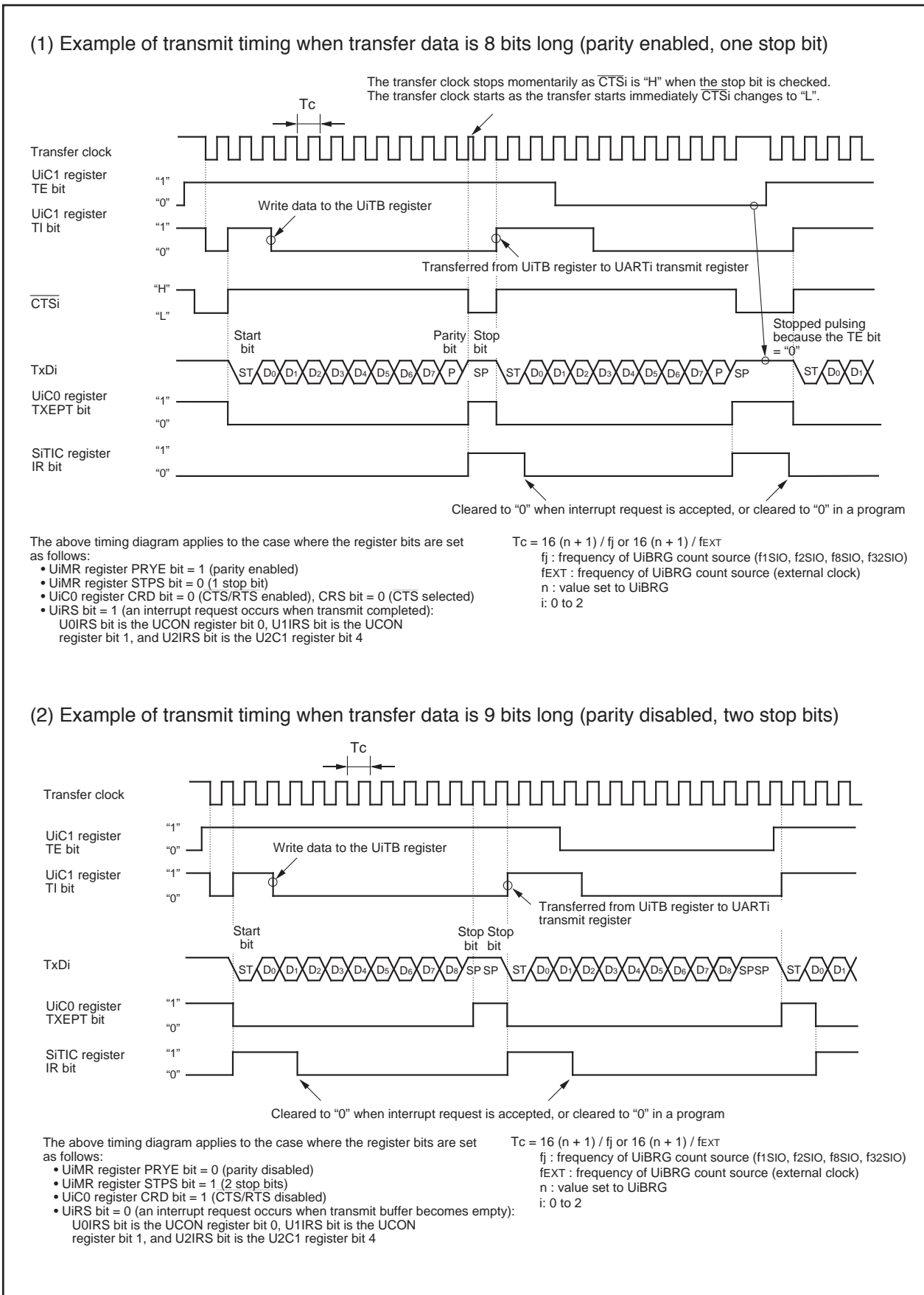


Figure 1.15.1. Transmit Operation



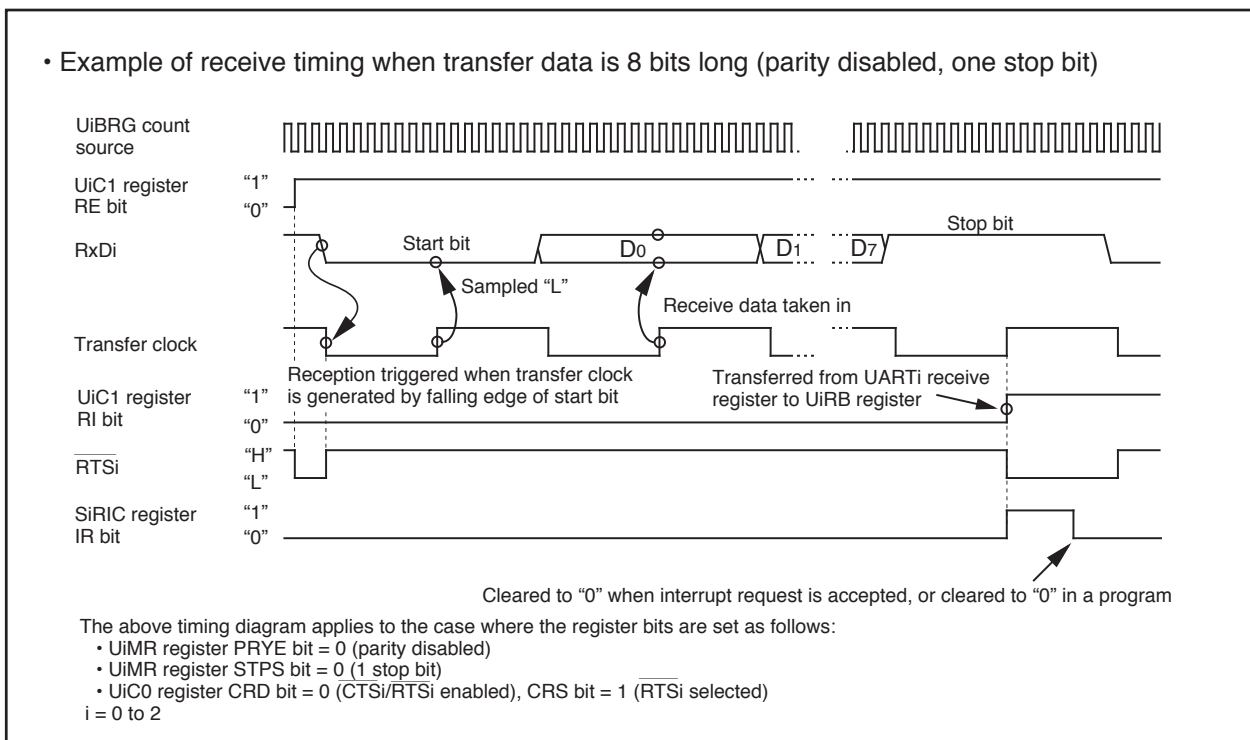


Figure 1.15.2. Receive Operation

**Bit Rates**

In UART mode, the frequency set by the UiBRG register (i=0 to 2) divided by 16 become the bit rates. Table 1.15.5 lists example of bit rates and settings.

Table 1.15.5. Example of Bit Rates and Settings

Bit Rate (bps)	Count Source of BRG	Peripheral Function Clock : 16MHz		Peripheral Function Clock : 24MHz	
		Set Value of BRG : n	Actual Time (bps)	Set value of BRG : n	Actual Time (bps)
1200	f8	103 (67h)	1202	155 (96h)	1202
2400	f8	51 (33h)	2404	77 (46h)	2404
4800	f8	25 (19h)	4808	38 (26h)	4808
9600	f1	103 (67h)	9615	155 (96h)	9615
14400	f1	68 (44h)	14493	103 (67h)	14423
19200	f1	51 (33h)	19231	77 (46h)	19231
28800	f1	34 (22h)	28571	51 (33h)	28846
31250	f1	31 (1Fh)	31250	47 (2Fh)	31250
38400	f1	25 (19h)	38462	38 (26h)	38462
51200	f1	19 (13h)	50000	28 (1Ch)	51724

### Counter Measure for Communication Error Occurs

If a communication error occurs while transmitting or receiving in UART mode, follow the procedures below.

- Resetting the UiRB register (i=0 to 2)
  - (1) Set the RE bit in the UiC1 register to “0” (reception disabled)
  - (2) Set the RE bit in the UiC1 register to “1” (reception enabled)
  
- Resetting the UiTB register (i=0 to 2)
  - (1) Set the SMD2 to SMD0 bits in the UiMR register “000b” (Serial I/O disabled)
  - (2) Set the SMD2 to SMD0 bits in the UiMR register “001b”, “101b”, “110b”.
  - (3) “1” is written to RE bit in the UiC1 register (reception enabled), regardless of the TE bit in the UiCi register

#### (a) LSB First/MSB First Select Function

As shown in Figure 1.15.3, use the UiC0 register's UFORM bit to select the transfer format. This function is valid when transfer data is 8 bits long.

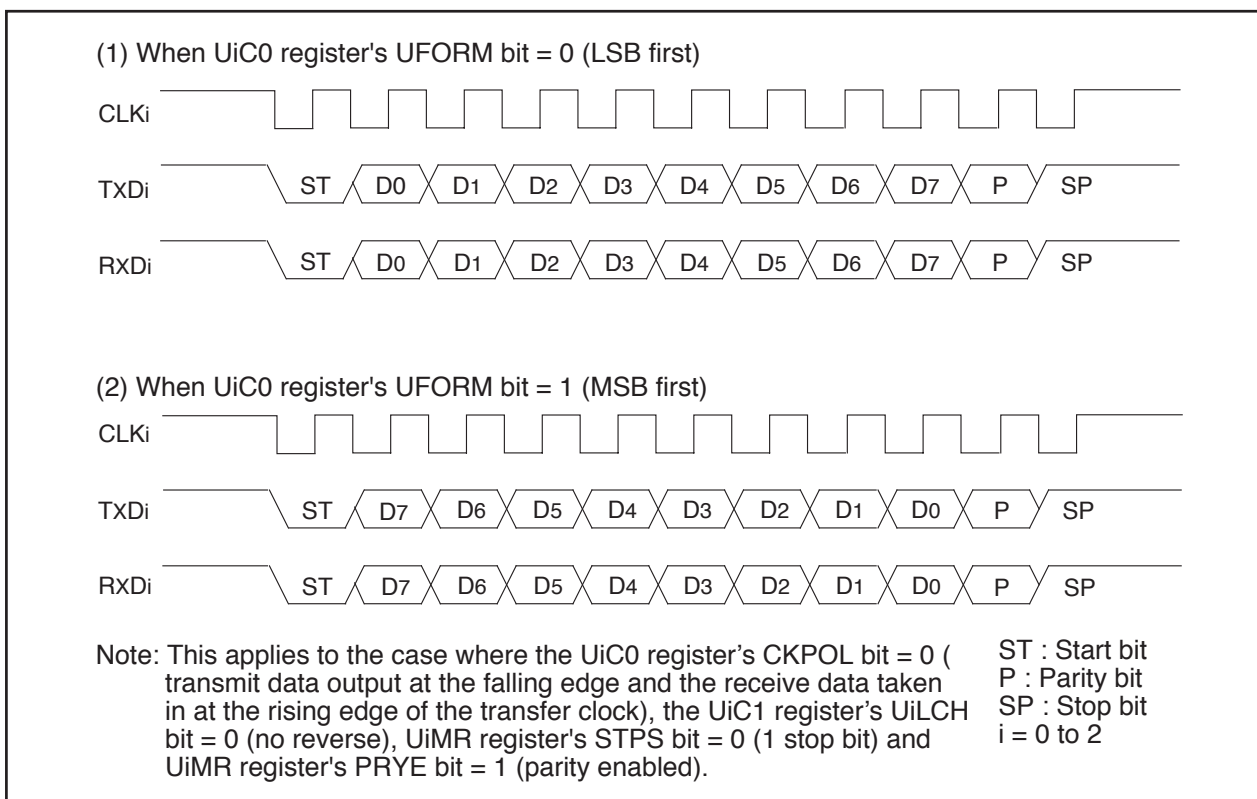
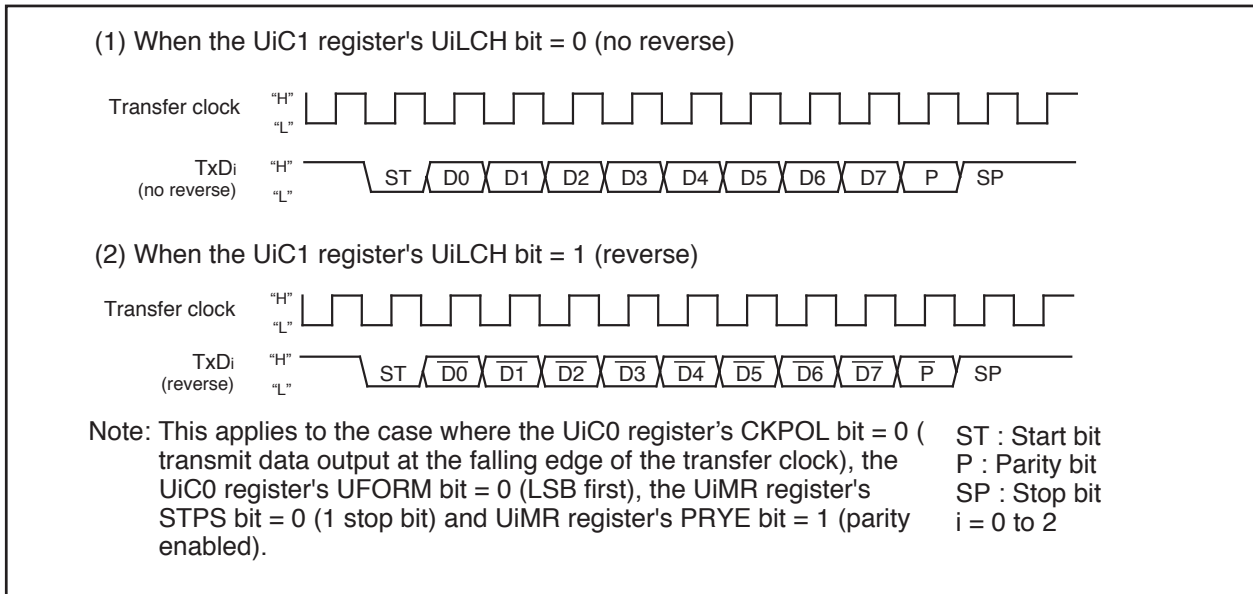


Figure 1.15.3. Transfer Format

**(b) Serial Data Logic Switching Function**

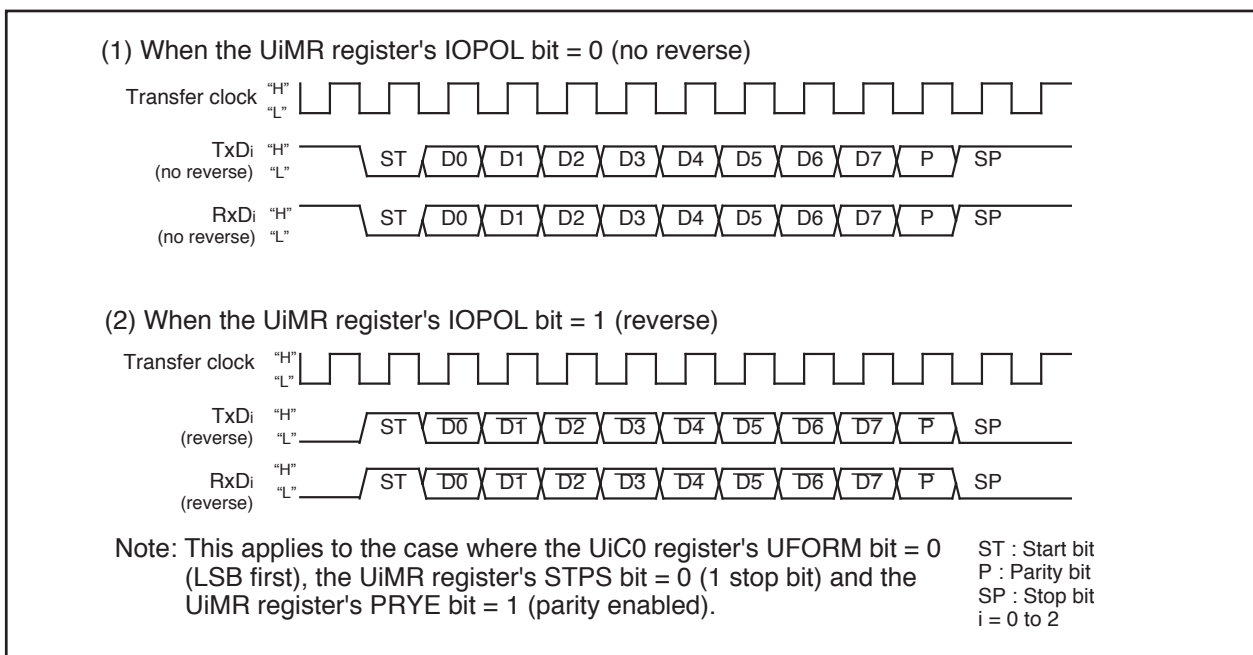
The data written to the UiTB register has its logic reversed before being transmitted. Similarly, the received data has its logic reversed when read from the UiRB register. Figure 1.15.4 shows serial data logic.



**Figure 1.15.4. Serial Data Logic Switching**

**(c) TxD and RxD I/O Polarity Inverse Function**

This function inverses the polarities of the TxDi pin output and RxDi pin input. The logic levels of all input/output data (including the start, stop and parity bits) are inverted. Figure 1.15.5 shows the TxD pin output and RxD pin input polarity inverse.



**Figure 1.15.5. TxD and RxD I/O Polarity Inverse**

### $\overline{\text{CTS}}/\overline{\text{RTS}}$ Function

When the  $\overline{\text{CTS}}$  function is used transmit operation start when “L” is applied to the  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  ( $i=0$  to 2) pin. Transmit operation begins when the  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is held “L”. If the “L” signal is switched to “H” during a transmit operation, the operation stops before the next data.

When the  $\overline{\text{RTS}}$  function is used, the  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin outputs on “L” signal when the microcomputer is ready to receive. The output level becomes “H” on the first falling edge of the CLK<sub>i</sub> pin.

- CRD bit in U<sub>i</sub>C0 register = 1 (disable  $\overline{\text{CTS}}/\overline{\text{RTS}}$  function of UART0)  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is programmable I/O function
- CRD bit = 0, CRS bit = 0 ( $\overline{\text{CTS}}$  function is selected)  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is  $\overline{\text{CTS}}$  function
- CRD bit = 0, CRS bit = 1 ( $\overline{\text{RTS}}$  function is selected)  $\overline{\text{CTS}}_i/\overline{\text{RTS}}_i$  pin is  $\overline{\text{RTS}}$  function

#### (d) $\overline{\text{CTS}}/\overline{\text{RTS}}$ Separate Function (UART0)

This function separates  $\overline{\text{CTS}}_0/\overline{\text{RTS}}_0$ , outputs  $\overline{\text{RTS}}_0$  from the P60 pin, and accepts as input the  $\overline{\text{CTS}}_0$  from the P64 pin. To use this function, set the register bits as shown below.

- U0C0 register's CRD bit = 0 (enables UART0  $\overline{\text{CTS}}/\overline{\text{RTS}}$ )
- U0C0 register's CRS bit = 1 (outputs UART0  $\overline{\text{RTS}}$ )
- U1C0 register's CRD bit = 0 (enables UART1  $\overline{\text{CTS}}/\overline{\text{RTS}}$ )
- U1C0 register's CRS bit = 0 (inputs UART1  $\overline{\text{CTS}}$ )
- UCON register's RCSP bit = 1 (inputs  $\overline{\text{CTS}}_0$  from the P64 pin)
- UCON register's CLKMD1 bit = 0 (CLKS<sub>1</sub> not used)

Note that when using the  $\overline{\text{CTS}}/\overline{\text{RTS}}$  separate function, UART1  $\overline{\text{CTS}}/\overline{\text{RTS}}$  separate function cannot be used.

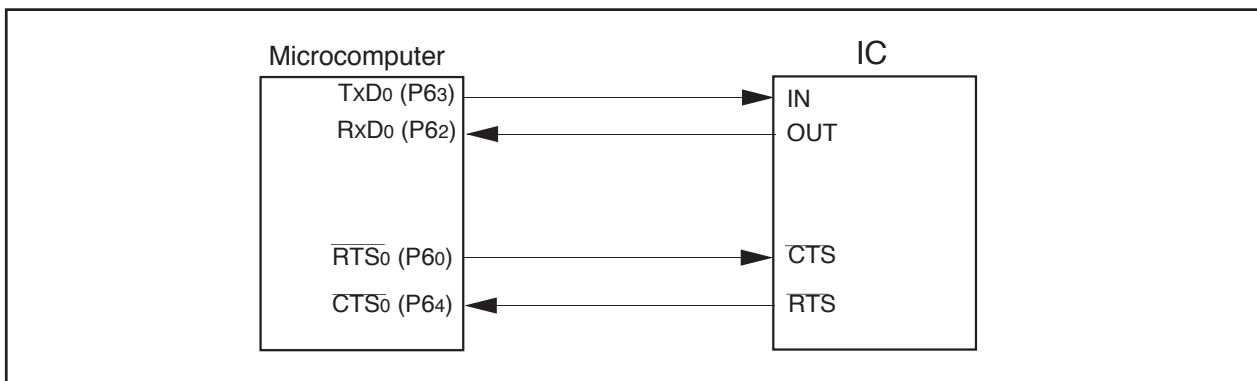


Figure 1.15.6.  $\overline{\text{CTS}}/\overline{\text{RTS}}$  Separate Function

## Special Mode 1 (I<sup>2</sup>C mode)

I<sup>2</sup>C mode is provided for use as a simplified I<sup>2</sup>C interface compatible mode. Table 1.16.1 lists the specifications of the I<sup>2</sup>C mode. Table 1.16.2 lists the registers used in the I<sup>2</sup>C mode and the register values set. Figure 1.16.1 shows the block diagram for I<sup>2</sup>C mode. Figure 1.16.2 shows SCLi timing.

As shown in Table 1.16.4, the microcomputer is placed in I<sup>2</sup>C mode by setting the SMD2 to SMD0 bits to '0102' and the IICM bit to "1". Because SDAi transmit output has a delay circuit attached, SDAi output does not change state until SCLi goes low and remains stably low.

**Table 1.16.1. I<sup>2</sup>C Mode Specifications**

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• During master UiMR(i=0 to 2) register's CKDIR bit = "0" (internal clock) : $f_j / 2^{(n+1)}$ $f_j = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}$ . n: Setting value of UiBRG register 00 <sub>16</sub> to FF <sub>16</sub> • During slave CKDIR bit = "1" (external clock) : Input from CLKi pin
Transmission start condition	• Before transmission can start, the following requirements must be met (Note 1) – The TE bit of UiC1 register= 1 (transmission enabled) – The TI bit of UiC1 register = 0 (data present in UiTB register)
Reception start condition	• Before reception can start, the following requirements must be met (Note 1) – The RE bit of UiC1 register= 1 (reception enabled) – The TE bit of UiC1 register= 1 (transmission enabled) – The TI bit of UiC1 register= 0 (data present in the UiTB register)
Interrupt request generation timing	When start or stop condition is detected, acknowledge undetected, and acknowledge detected
Error detection	• Overrun error (Note 2) This error occurs if the serial I/O started receiving the next data before reading the UiRB register and received the 8th bit of the data
Select function	• Arbitration lost Timing at which the UiRB register's ABT bit is updated can be selected • SDAi digital delay No digital delay or a delay of 2 to 8 UiBRG count source clock cycles selectable • Clock phase setting With or without clock delay selectable

Note 1: When an external clock is selected, the conditions must be met while the external clock is in the high state.

Note 2: If an overrun error occurs, the value of UiRB register will be indeterminate. The IR bit of SiRIC register does not change.

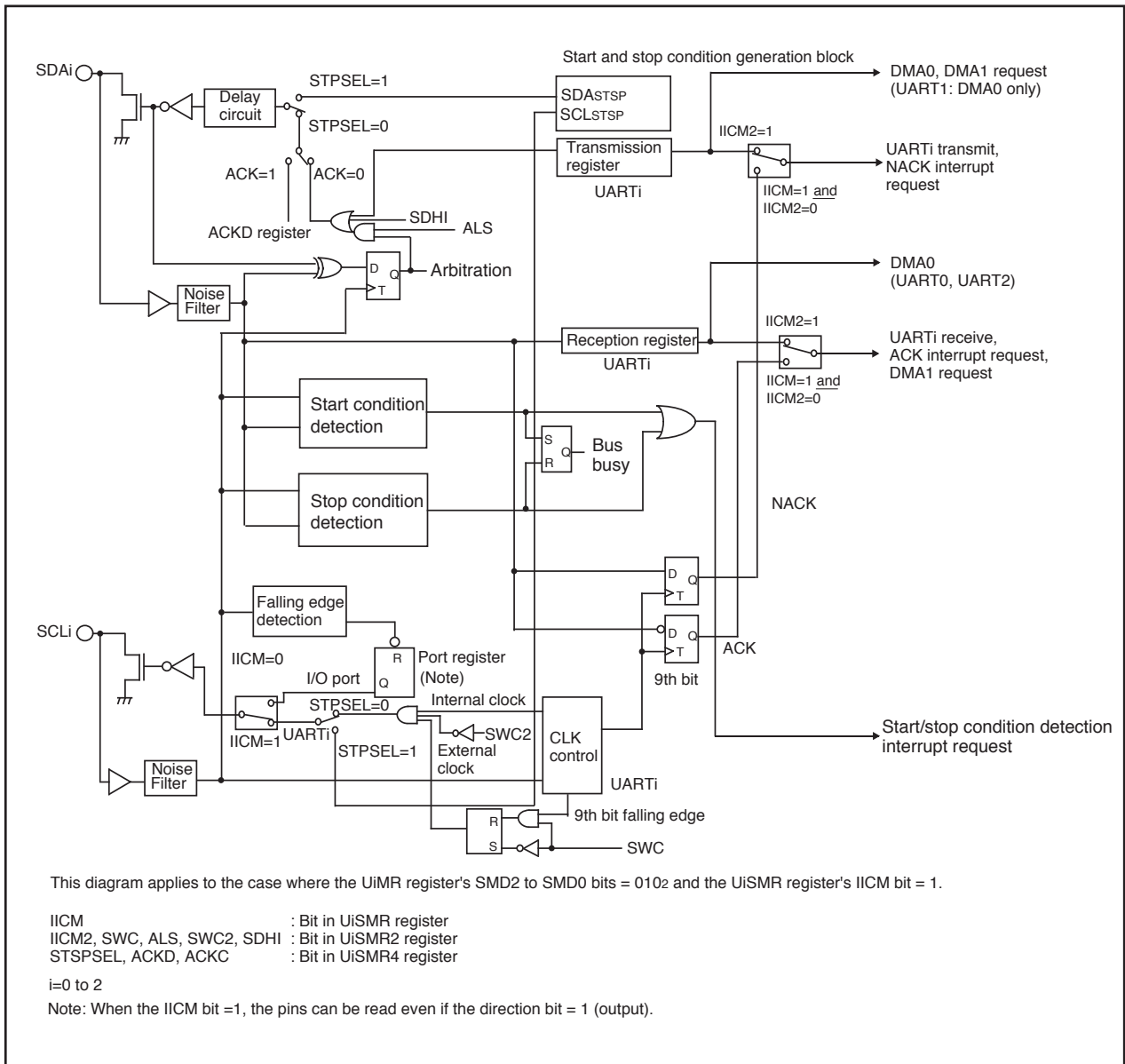


Figure 1.16.1. I²C Mode Block Diagram

Table 1.16.2. Registers to Be Used and Settings in I<sup>2</sup>C Mode (1) (Continued)

Register	Bit	Function	
		Master	Slave
UiTB <sup>3</sup>	0 to 7	Set transmission data	Set transmission data
UiRB <sup>3</sup>	0 to 7	Reception data can be read	Reception data can be read
	8	ACK or NACK is set in this bit	ACK or NACK is set in this bit
	ABT	Arbitration lost detection flag	Invalid
	OER	Overrun error flag	Overrun error flag
UiBRG	0 to 7	Set a transfer rate	Invalid
UiMR <sup>3</sup>	SMD2 to SMD0	Set to '0102'	Set to '0102'
	CKDIR	Set to "0"	Set to "1"
	IOPOL	Set to "0"	Set to "0"
UiC0	CLK1, CLK0	Select the count source for the UiBRG register	Invalid
	CRS	Invalid because CRD = 1	Invalid because CRD = 1
	TXEPT	Transmit buffer empty flag	Transmit buffer empty flag
	CRD	Set to "1"	Set to "1"
	NCH	Set to "1" <sup>2</sup>	Set to "1" <sup>2</sup>
	CKPOL	Set to "0"	Set to "0"
	UFORM	Set to "1"	Set to "1"
UiC1	TE	Set this bit to "1" to enable transmission	Set this bit to "1" to enable transmission
	TI	Transmit buffer empty flag	Transmit buffer empty flag
	RE	Set this bit to "1" to enable reception	Set this bit to "1" to enable reception
	RI	Reception complete flag	Reception complete flag
	U2IRS <sup>1</sup>	Invalid	Invalid
	U2RRM <sup>1</sup> , UiLCH, UiERE	Set to "0"	Set to "0"
UiSMR	IICM	Set to "1"	Set to "1"
	ABC	Select the timing at which arbitration-lost is detected	Invalid
	BBS	Bus busy flag	Bus busy flag
	3 to 7	Set to "0"	Set to "0"
UiSMR2	IICM2	Refer to Table 1.16.4.	Refer to Table 1.16.4.
	CSC	Set this bit to "1" to enable clock synchronization	Set to "0"
	SWC	Set this bit to "1" to have SCLi output fixed to "L" at the falling edge of the 9th bit of clock	Set this bit to "1" to have SCLi output fixed to "L" at the falling edge of the 9th bit of clock
	ALS	Set this bit to "1" to have SDAi output stopped when arbitration-lost is detected	Set to "0"
	STAC	Set to "0"	Set this bit to "1" to initialize UARTi at start condition detection
	SWC2	Set this bit to "1" to have SCLi output forcibly pulled low	Set this bit to "1" to have SCLi output forcibly pulled low
	SDHI	Set this bit to "1" to disable SDAi output	Set this bit to "1" to disable SDAi output
UiSMR3	0, 2, 4 and NODC	Set to "0"	Set to "0"
	CKPH	Refer to Table 1.16.4	Refer to Table 1.16.4
	DL2 to DL0	Set the amount of SDAi digital delay	Set the amount of SDAi digital delay

i=0 to 2

Notes:

1. Set the U0C1 and U1C1 register bit 4 and bit 5 to "0". The U0IRS, U1IRS, U0RRM and U1RRM bits are in the UCON register.
2. TxD2 pin is N channel open-drain output. Set the NCH bit in the U2C0 register to "0".
3. Not all register bits are described above. Set those bits to "0" when writing to the registers in I<sup>2</sup>C mode.

**Table 1.16.3. Registers to Be Used and Settings in I<sup>2</sup>C Mode (2) (Continued)**

Register	Bit	Function	
		Master	Slave
UiSMR4	STAREQ	Set this bit to "1" to generate start condition	Set to "0"
	RSTAREQ	Set this bit to "1" to generate restart condition	Set to "0"
	STPREQ	Set this bit to "1" to generate stop condition	Set to "0"
	STSPSEL	Set this bit to "1" to output each condition	Set to "0"
	ACKD	Select ACK or NACK	Select ACK or NACK
	ACKC	Set this bit to "1" to output ACK data	Set this bit to "1" to output ACK data
	SCLHI	Set this bit to "1" to have SCLi output stopped when stop condition is detected	Set to "0"
	SWC9	Set to "0"	Set this bit to "1" to set the SCLi to "L" hold at the falling edge of the 9th bit of clock
IFSR2A	IFSR26, ISFR27	Set to "1"	Set to "1"
UCON	U0IRS, U1IRS	Invalid	Invalid
	2 to 7	Set to "0"	Set to "0"

i=0 to 2



Table 1.16.4. I<sup>2</sup>C Mode Functions

Function	Clock Synchronous Serial I/O Mode (SMD2 to SMD0 = 001b, IICM = 0)	I <sup>2</sup> C Mode (SMD2 to SMD0 = 010b, IICM = 1)			
		IICM2 = 0 (NACK/ACK interrupt)		IICM2 = 1 (UART transmit/ receive interrupt)	
		CKPH = 0 (No clock delay)	CKPH = 1 (Clock delay)	CKPH = 0 (No clock delay)	CKPH = 1 (Clock delay)
Factor of Interrupt Number 6, 7 and 10 (1.5, 7)	—————	Start condition detection or stop condition detection (See Table 1.16.5 STSPSEL Bit Functions)			
Factor of Interrupt Number 15, 17 and 19 (1.6)	UARTi transmission Transmission started or completed (selected by UiIRS)	No acknowledgment detection (NACK) Rising edge of SCLi 9th bit	UARTi transmission Rising edge of SCLi 9th bit	UARTi transmission Falling edge of SCLi next to the 9th bit	
Factor of Interrupt Number 16, 18 and 20 (1.6)	UARTi reception When 8th bit received CKPOL = 0 (rising edge) CKPOL = 1 (falling edge)	Acknowledgment detection (ACK) Rising edge of SCLi 9th bit	UARTi reception Falling edge of SCLi 9th bit		
Timing for Transferring Data From the UART Reception Shift Register to the UiRB Register	CKPOL = 0 (rising edge) CKPOL = 1 (falling edge)	Rising edge of SCLi 9th bit	Falling edge of SCLi 9th bit	Falling and rising edges of SCLi 9th bit	
UARTi Transmission Output Delay	Not delayed	Delayed			
Functions of P6_3, P6_7 and P7_0 Pins	TXDi output	SDAi input/output			
Functions of P6_2, P6_6 and P7_1 Pins	RXD <sub>i</sub> input	SCLi input/output			
Functions of P6_1, P6_5 and P7_2 Pins	CLK <sub>i</sub> input or output selected	————— (Cannot be used in I <sup>2</sup> C mode)			
Noise Filter Width	15ns	200ns			
Read RXDi and SCLi Pin Levels	Possible when the corresponding port direction bit = 0	Always possible no matter how the corresponding port direction bit is set			
Initial Value of TXDi and SDAi Outputs	CKPOL = 0 (H) CKPOL = 1 (L)	The value set in the port register before setting I <sup>2</sup> C mode (2)			
Initial and End Values of SCLi	—————	H	L	H	
DMA1 Factor (6)	UARTi reception	Acknowledgment detection (ACK)	UARTi reception Falling edge of SCLi 9th bit		
Store Received Data	1st to 8th bits of the received data are stored into bits 7 to 0 in the UiRB register	1st to 8th bits of the received data are stored into bits 7 to 0 in the UiRB register	1st to 7th bits of the received data are stored into bits 6 to 0 in the UiRB register. 8th bit is stored into bit 8 in the UiRB register.		
			1st to 8th bits are stored into bits 7 to 0 in the UiRB register (3)		
Read Received Data	The UiRB register status is read	Bits 6 to 0 in the UiRB register (4) are read as bits 7 to 1. Bit 8 in the UiRB register is read as bit 0.			

i = 0 to 2

## NOTES:

- If the source or cause of any interrupt is changed, the IR bit in the interrupt control register for the changed interrupt may inadvertently be set to "1" (interrupt requested). (Refer to **Changing the Interrupt Generate Factor**.)  
If one of the bits shown below is changed, the interrupt source, the interrupt timing, etc. change. Therefore, always be sure to clear the IR bit to "0" (interrupt not requested) after changing those bits.  
SMD2 to SMD0 bits in the UiMR register, IICM bit in the UiSMR register, IICM2 bit in the UiSMR2 register, CKPH bit in the UiSMR3 register
- Set the initial value of SDAi output while the SMD2 to SMD0 bits in the UiMR register = 000b (serial I/O disabled).
- Second data transfer to UiRB register (Rising edge of SCLi 9th bit)
- First data transfer to UiRB register (Falling edge of SCLi 9th bit)
- See **Figure 1.16.4 STSPSEL Bit Functions**.
- See **Figure 1.16.2 Transfer to UiRB Register and Interrupt Timing**
- When using UART0, be sure to set the IFSR26 bit in the IFSR2A register to "1" (factor of interrupt: UART0 bus collision).  
When using UART1, be sure to set the IFSR27 bit to "1" (factor of interrupt: UART1 bus collision).

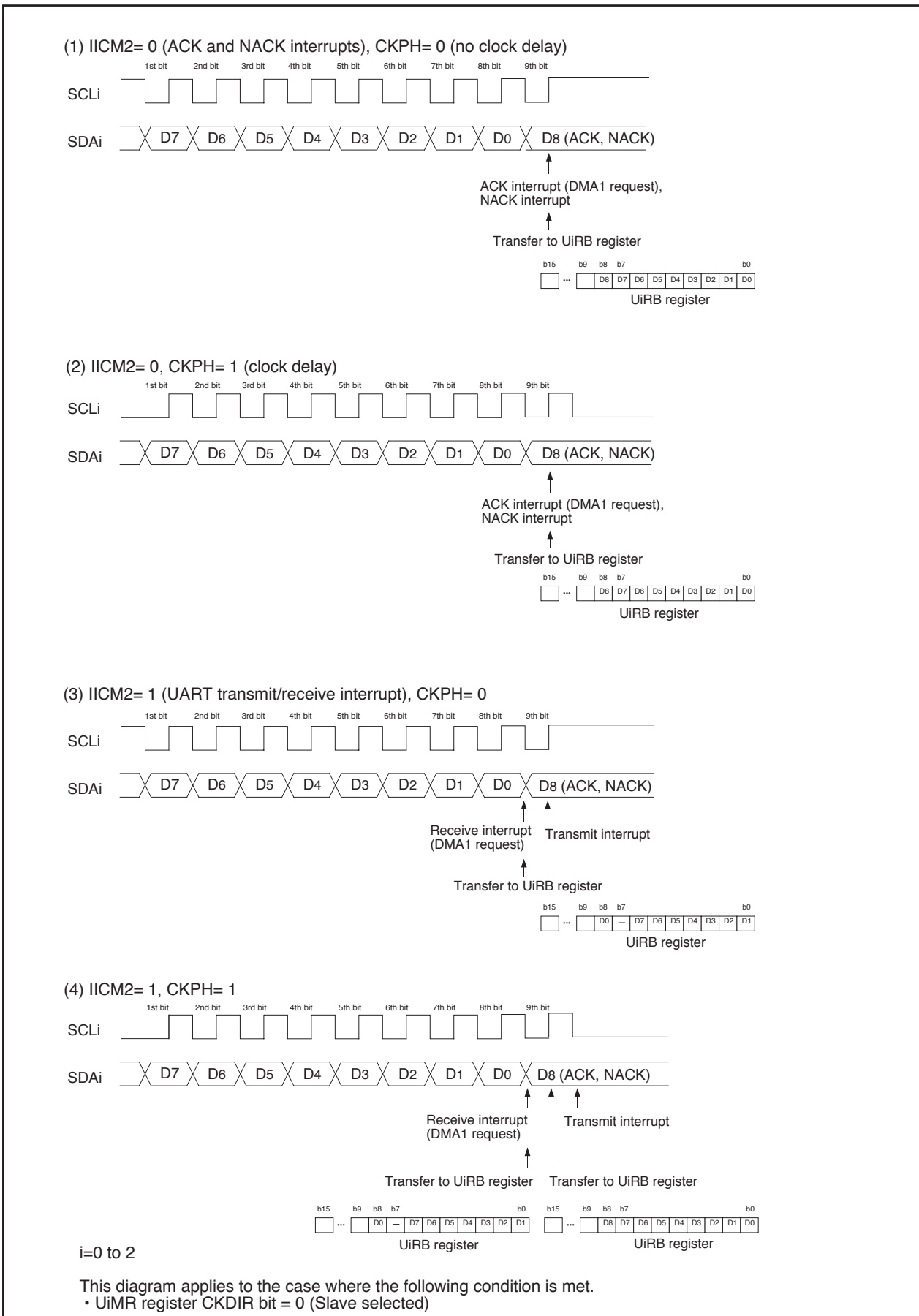


Figure 1.16.2. Transfer to UiRB Register and Interrupt Timing

### • Detection of Start and Stop Condition

Whether a start or a stop condition has been detected is determined.

A start condition-detected interrupt request is generated when the SDA<sub>i</sub> pin changes state from high to low while the SCL<sub>i</sub> pin is in the high state. A stop condition-detected interrupt request is generated when the SDA<sub>i</sub> pin changes state from low to high while the SCL<sub>i</sub> pin is in the high state.

Because the start and stop condition-detected interrupts share the interrupt control register and vector, check the UiSMR register's BBS bit to determine which interrupt source is requesting the interrupt.

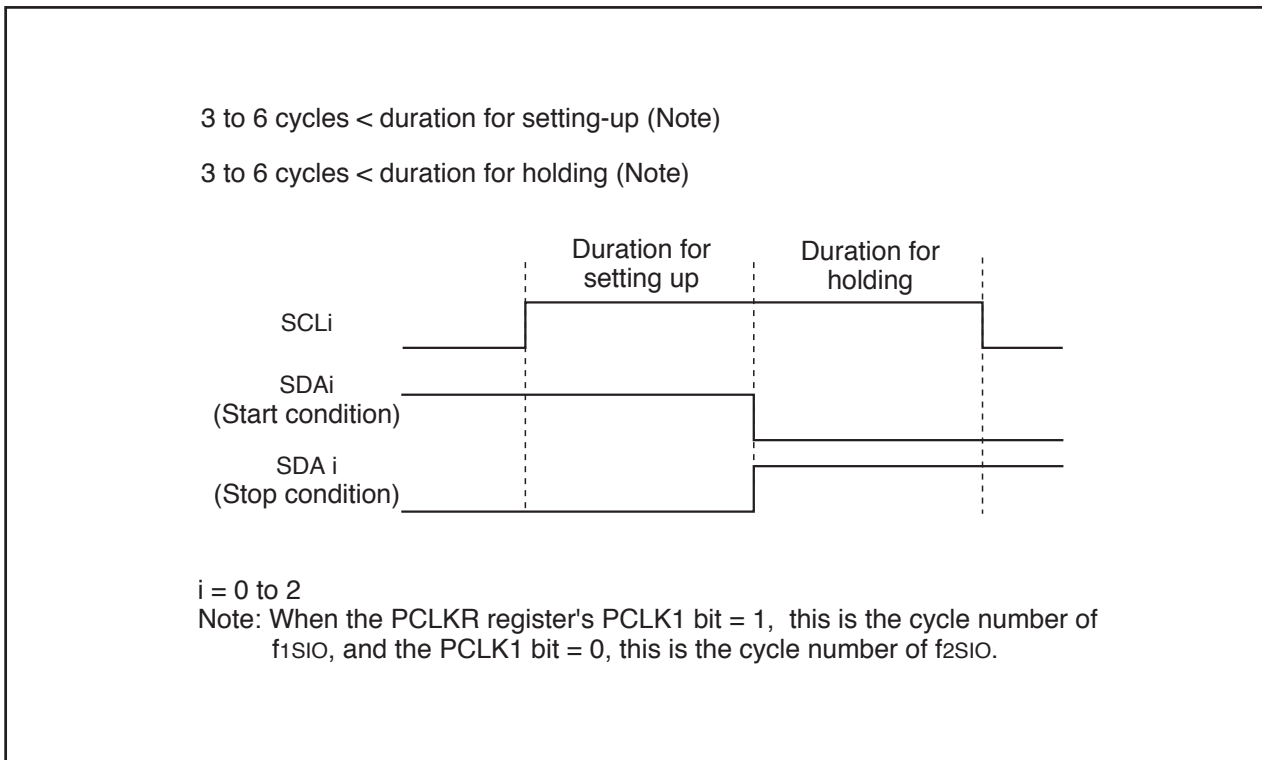


Figure 1.16.3. Detection of Start and Stop Condition

### • Output of Start and Stop Condition

A start condition is generated by setting the STAREQ bit in the UiSMR4 register ( $i = 0$  to 2) to "1" (start).

A restart condition is generated by setting the RSTAREQ bit in the UiSMR4 register to "1" (start).

A stop condition is generated by setting the STPREQ bit in the UiSMR4 register to "1" (start).

The output procedure is described below.

- (1) Set the STAREQ bit, RSTAREQ bit or STPREQ bit to "1" (start).
- (2) Set the STSPSEL bit in the UiSMR4 register to "1" (output).

The function of the STSPSEL bit is shown in Table 1.16.5 and Figure 1.16.4.

Table 1.16.5. STSPSEL Bit Functions

Function	STSPSEL = 0	STSPSEL = 1
Output of SCLi and SDAi pins	Output of transfer clock and data Output of start/stop condition is accomplished by a program using ports (not automatically generated in hardware)	Output of a start/stop condition according to the STAREQ, RSTAREQ and STPREQ bit
Star/stop condition interrupt request generation timing	Start/stop condition detection	Finish generating start/stop condition

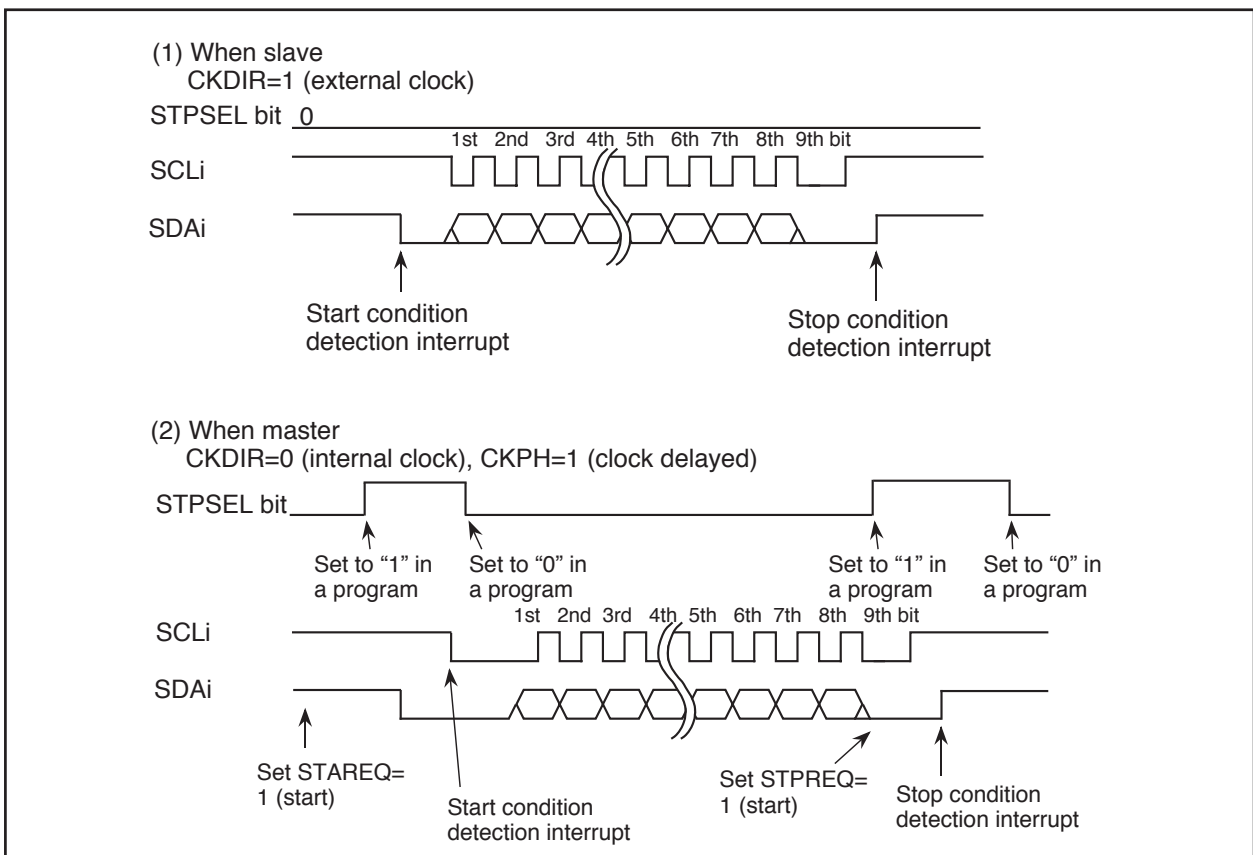


Figure 1.16.4. STSPSEL Bit Functions

#### • Arbitration

Unmatching of the transmit data and SDAi pin input data is checked synchronously with the rising edge of SCLi. Use the UiSMR register's ABC bit to select the timing at which the UiRB register's ABT bit is updated. If the ABC bit = 0 (updated bitwise), the ABT bit is set to "1" at the same time unmatching is detected during check, and is cleared to "0" when not detected. In cases when the ABC bit is set to "1", if unmatching is detected even once during check, the ABT bit is set to "1" (unmatching detected) at the falling edge of the clock pulse of 9th bit. If the ABT bit needs to be updated bitwise, clear the ABT bit to "0" (undetected) after detecting acknowledge in the first byte, before transferring the next byte.

Setting the UiSMR2 register's ALS bit to "1" (SDA output stop enabled) causes arbitration-lost to occur, in which case the SDAi pin is placed in the high-impedance state at the same time the ABT bit is set to "1" (unmatching detected).

### • Transfer Clock

Data is transmitted/received using a transfer clock like the one shown in Figure 1.16.4.

The UiSMR2 register's CSC bit is used to synchronize the internally generated clock (internal SCLi) and an external clock supplied to the SCLi pin. In cases when the CSC bit is set to "1" (clock synchronization enabled), if a falling edge on the SCLi pin is detected while the internal SCLi is high, the internal SCLi goes low, at which time the UiBRG register value is reloaded with and starts counting in the low-level interval. If the internal SCLi changes state from low to high while the SCLi pin is low, counting stops, and when the SCLi pin goes high, counting restarts.

In this way, the UARTi transfer clock is comprised of the logical product of the internal SCLi and SCLi pin signal. The transfer clock works from a half period before the falling edge of the internal SCLi 1st bit to the rising edge of the 9th bit. To use this function, select an internal clock for the transfer clock. The UiSMR2 register's SWC bit allows to select whether the SCLi pin should be fixed to or freed from low-level output at the falling edge of the 9th clock pulse.

If the UiSMR4 register's SCLHI bit is set to "1" (enabled), SCLi output is turned off (placed in the high-impedance state) when a stop condition is detected.

Setting the UiSMR2 register's SWC2 bit = 1 (0 output) makes it possible to forcibly output a low-level signal from the SCLi pin even while sending or receiving data. Clearing the SWC2 bit to "0" (transfer clock) allows the transfer clock to be output from or supplied to the SCLi pin, instead of outputting a low-level signal.

If the UiSMR4 register's SWC9 bit is set to "1" (SCL hold low enabled) when the UiSMR3 register's CKPH bit = 1, the SCLi pin is fixed to low-level output at the falling edge of the clock pulse next to the ninth. Setting the SWC9 bit = 0 (SCL hold low disabled) frees the SCLi pin from low-level output.

### • SDA Output

The data written to the UiTB register bit 7 to bit 0 (D7 to D0) is sequentially output beginning with D7. The ninth bit (D8) is ACK or NACK.

The initial value of SDAi transmit output can only be set when IICM = 1 (I<sup>2</sup>C mode) and the UiMR register's SMD2 to SMD0 bits = '0002' (serial I/O disabled).

The UiSMR3 register's DL2 to DL0 bits allow to add no delays or a delay of 2 to 8 UiBRG count source clock cycles to SDAi output.

Setting the UiSMR2 register's SDHI bit = 1 (SDA output disabled) forcibly places the SDAi pin in the high-impedance state. Do not write to the SDHI bit synchronously with the rising edge of the UARTi transfer clock. This is because the ABT bit may inadvertently be set to "1" (detected).

### • SDA Input

When the IICM2 bit = 0, the 1st to 8th bits (D7 to D0) of received data are stored in the UiRB register bit 7 to bit 0. The 9th bit (D8) is ACK or NACK.

When the IICM2 bit = 1, the 1st to 7th bits (D7 to D1) of received data are stored in the UiRB register bit 6 to bit 0 and the 8th bit (D0) is stored in the UiRB register bit 8. Even when the IICM2 bit = 1, providing the CKPH bit = 1, the same data as when the IICM2 bit = 0 can be read out by reading the UiRB register after the rising edge of the corresponding clock pulse of 9th bit.

**• ACK and NACK**

If the STSPSEL bit in the UiSMR4 register is set to “0” (start and stop conditions not generated) and the ACKC bit in the UiSMR4 register is set to “1” (ACK data output), the value of the ACKD bit in the UiSMR4 register is output from the SDAi pin.

If the IICM2 bit = 0, a NACK interrupt request is generated if the SDAi pin remains high at the rising edge of the 9th bit of transmit clock pulse. An ACK interrupt request is generated if the SDAi pin is low at the rising edge of the 9th bit of transmit clock pulse.

If ACKi is selected for the cause of DMA1 request, a DMA transfer can be activated by detection of an acknowledge.

**• Initialization of Transmission/Reception**

If a start condition is detected while the STAC bit = 1 (UARTi initialization enabled), the serial I/O operates as described below.

- The transmit shift register is initialized, and the content of the UiTB register is transferred to the transmit shift register. In this way, the serial I/O starts sending data synchronously with the next clock pulse applied. However, the UARTi output value does not change state and remains the same as when a start condition was detected until the first bit of data is output synchronously with the input clock.
- The receive shift register is initialized, and the serial I/O starts receiving data synchronously with the next clock pulse applied.
- The SWC bit is set to “1” (SCL wait output enabled). Consequently, the SCLi pin is pulled low at the falling edge of the ninth clock pulse.

Note that when UARTi transmission/reception is started using this function, the TI does not change state. Note also that when using this function, the selected transfer clock should be an external clock.

## Special Mode 2

Multiple slaves can be serially communicated from one master. Synchronous clock polarity and phase are selectable. Table 1.16.6 lists the specifications of Special Mode 2. Table 1.16.7 lists the registers used in Special Mode 2 and the register values set. Figure 1.16.5 shows communication control example for Special Mode 2. UART2 is not available in this mode.

**Table 1.16.6. Special Mode 2 Specifications**

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• Master mode UiMR(i=0 to 1) register's CKDIR bit = "0" (internal clock) : $f_j / 2^{(n+1)}$ $f_j = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}$ . n: Setting value of UiBRG register 00 <sub>16</sub> to FF <sub>16</sub> • Slave mode CKDIR bit = "1" (external clock selected) : Input from CLKi pin
Transmit/receive control	Controlled by input/output ports
Transmission start condition	• Before transmission can start, the following requirements must be met (Note 1) – The TE bit of UiC1 register= 1 (transmission enabled) – The TI bit of UiC1 register = 0 (data present in UiTB register)
Reception start condition	• Before reception can start, the following requirements must be met (Note 1) – The RE bit of UiC1 register= 1 (reception enabled) – The TE bit of UiC1 register= 1 (transmission enabled) – The TI bit of UiC1 register= 0 (data present in the UiTB register)
Interrupt request generation timing	• For transmission, one of the following conditions can be selected – The UiIRS bit of UiC1 register = 0 (transmit buffer empty): when transferring data from the UiTB register to the UARTi transmit register (at start of transmission) – The UiIRS bit =1 (transfer completed): when the serial I/O finished sending data from the UARTi transmit register • For reception When transferring data from the UARTi receive register to the UiRB register (at completion of reception)
Error detection	• Overrun error (Note 2) This error occurs if the serial I/O started receiving the next data before reading the UiRB register and received the 7th bit of the next data
Select function	• Clock phase setting Selectable from four combinations of transfer clock polarities and phases

Note 1: When an external clock is selected, the conditions must be met while if the UiC0 register's CKPOL bit = "0" (transmit data output at the falling edge and the receive data taken in at the rising edge of the transfer clock), the external clock is in the high state; if the UiC0 register's CKPOL bit = "1" (transmit data output at the rising edge and the receive data taken in at the falling edge of the transfer clock), the external clock is in the low state.

Note 2: If an overrun error occurs, the value of UiRB register will be indeterminate. The IR bit of SiRIC register does not change.

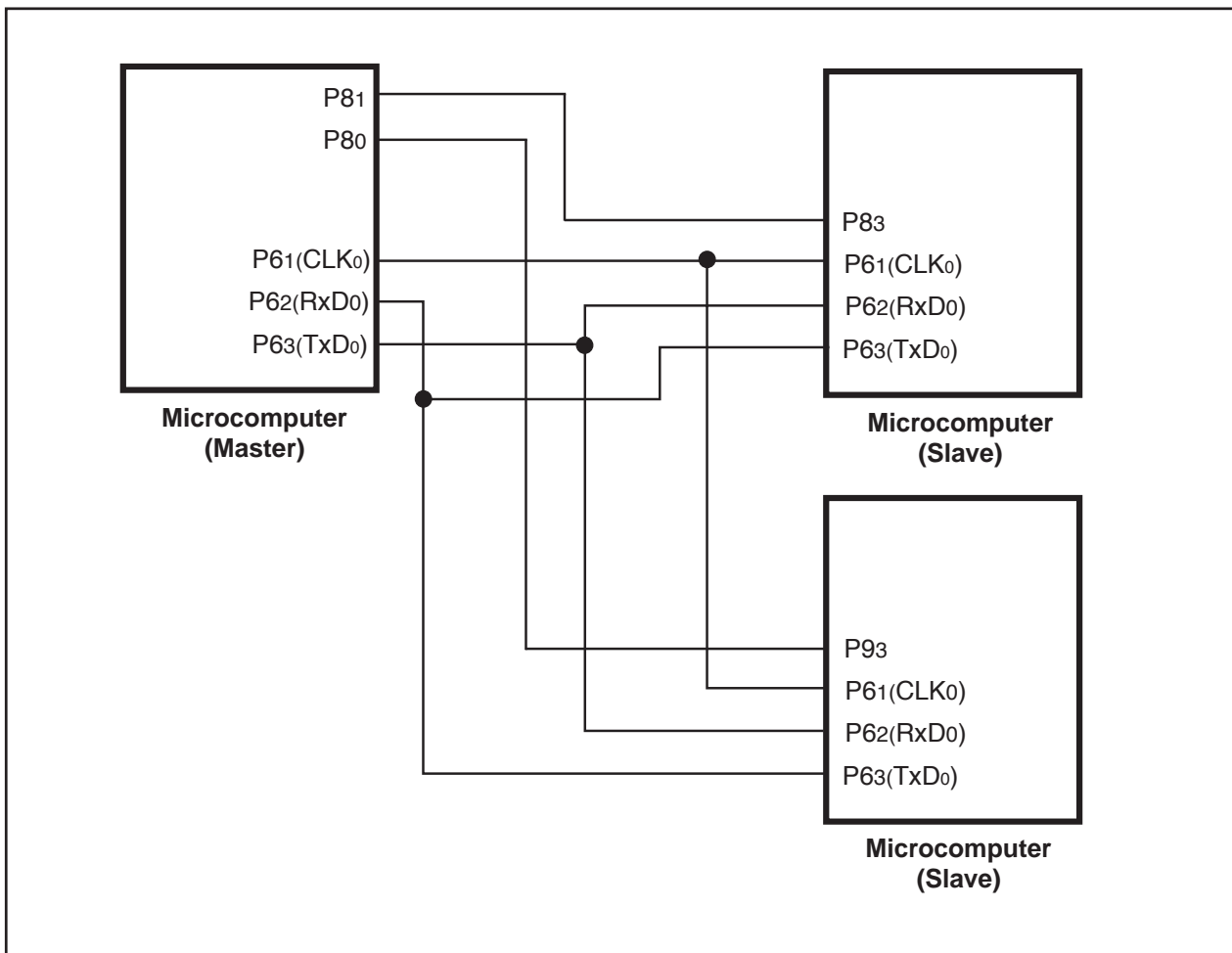


Figure 1.16.5. Serial Bus Communication Control Example (UART0)



**Table 1.16.7. Registers to Be Used and Settings in Special Mode 2**

Register	Bit	Function
UiTB(Note2)	0 to 7	Set transmission data
UiRB(Note2)	0 to 7	Reception data can be read
	OER	Overflow error flag
UiBRG	0 to 7	Set a transfer rate
UiMR(Note2)	SMD2 to SMD0	Set to '0012'
	CKDIR	Set this bit to "0" for master mode or "1" for slave mode
	IOPOL	Set to "0"
UiC0	CLK1, CLK0	Select the count source for the UiBRG register
	CRS	Invalid because CRD = 1
	TXEPT	Transmit register empty flag
	CRD	Set to "1"
	NCH	Select TxDi pin output format
	CKPOL	Clock phases can be set in combination with the UiSMR3 register's CKPH bit
	UFORM	Set to "0"
UiC1	TE	Set this bit to "1" to enable transmission
	TI	Transmit buffer empty flag
	RE	Set this bit to "1" to enable reception
	RI	Reception complete flag
	U2IRS (Note 1)	Select UART2 transmit interrupt cause
	U2RRM(Note 1), U2LCH, UiERE	Set to "0"
UiSMR	0 to 7	Set to "0"
UiSMR2	0 to 7	Set to "0"
UiSMR3	CKPH	Clock phases can be set in combination with the UiC0 register's CKPOL bit
	NODC	Set to "0"
	0, 2, 4 to 7	Set to "0"
UiSMR4	0 to 7	Set to "0"
UCON	U0IRS, U1IRS	Select UART0 and UART1 transmit interrupt cause
	U0RRM, U1RRM	Set to "0"
	CLKMD0	Invalid because CLKMD1 = 0
	CLKMD1, RCSP, 7	Set to "0"

Note 1: Set the U0C0 and U1C1 register bit 4 and bit 5 to "0". The U0IRS, U1IRS, U0RRM and U1RRM bits are in the UCON register.

Note 2: Not all register bits are described above. Set those bits to "0" when writing to the registers in Special Mode 2.

i = 0 to 1

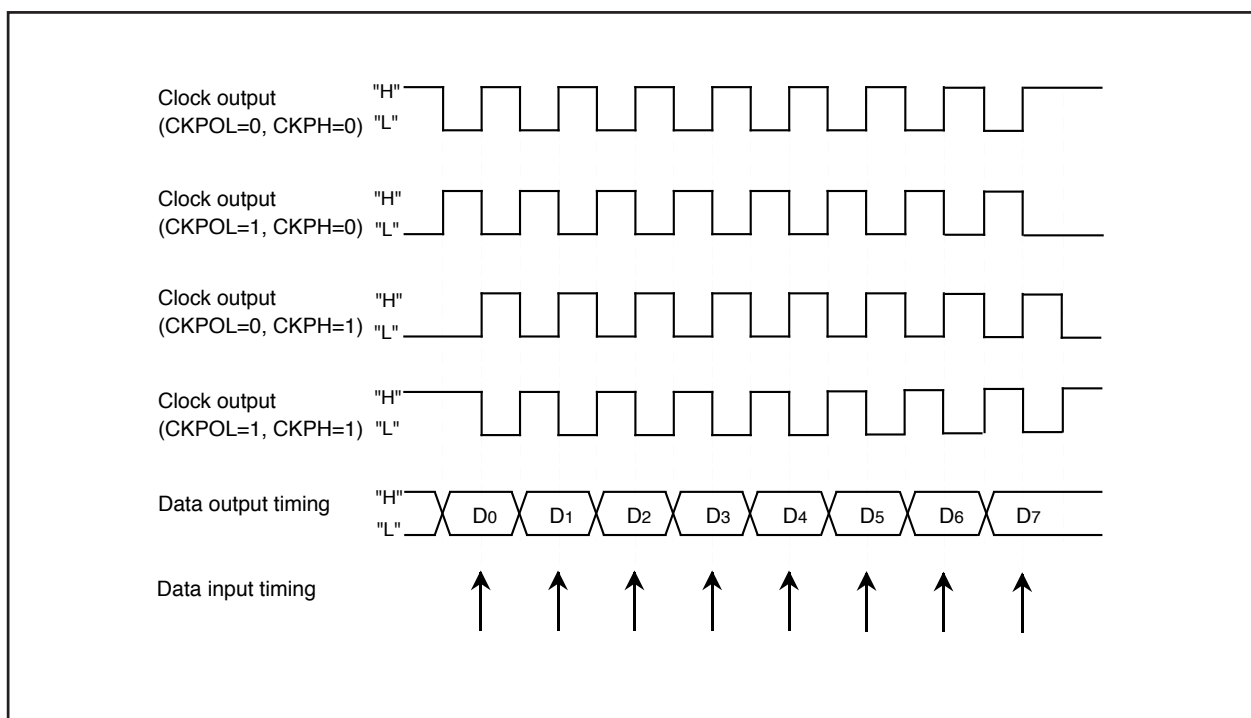
• **Clock Phase Setting Function**

One of four combinations of transfer clock phases and polarities can be selected using the UiSMR3 register's CKPH bit and the UiC0 register's CKPOL bit.

Make sure the transfer clock polarity and phase are the same for the master and slaves to be communicated.

Figure 1.16.6 shows the transmission and reception timing in master (internal clock).

Figure 1.16.7 shows the transmission and reception timing (CKPH=0) in slave (external clock) while Figure 1.16.8 shows the transmission and reception timing (CKPH=1) in slave (external clock).



**Figure 1.16.6. Transmission and Reception Timing in Master Mode (Internal Clock)**

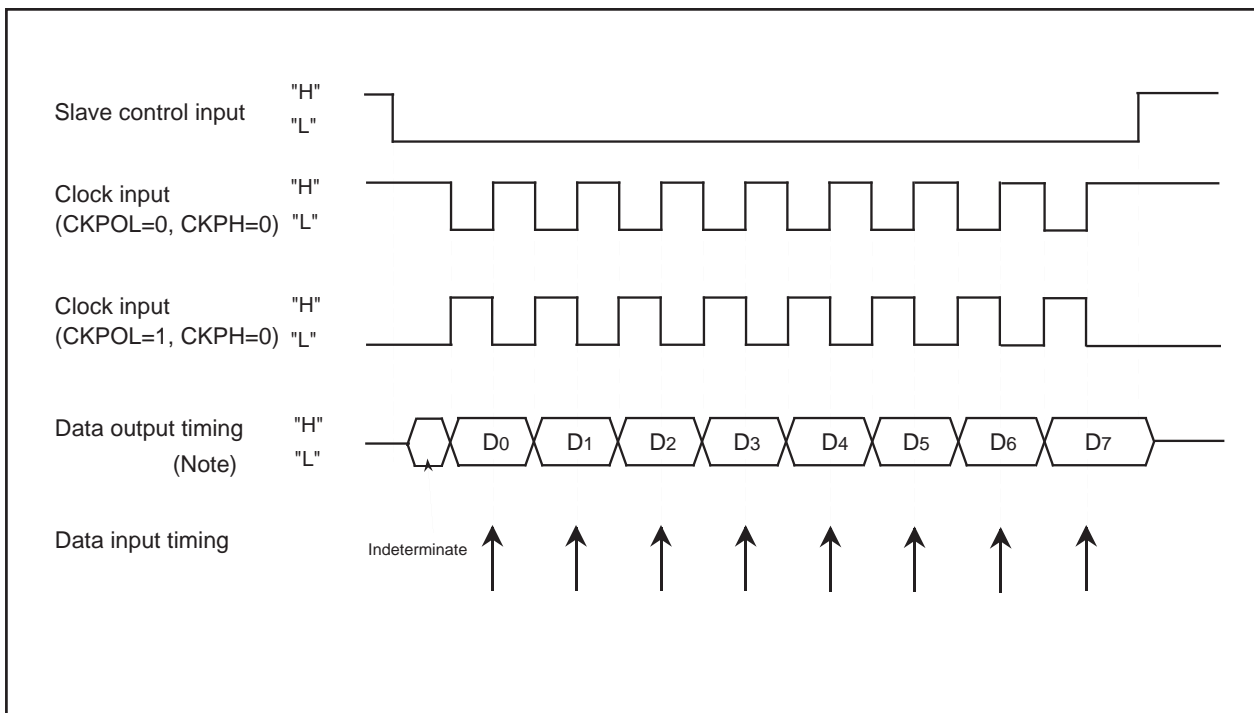


Figure 1.16.7. Transmission and Reception Timing (CKPH=0) in Slave Mode (External Clock)

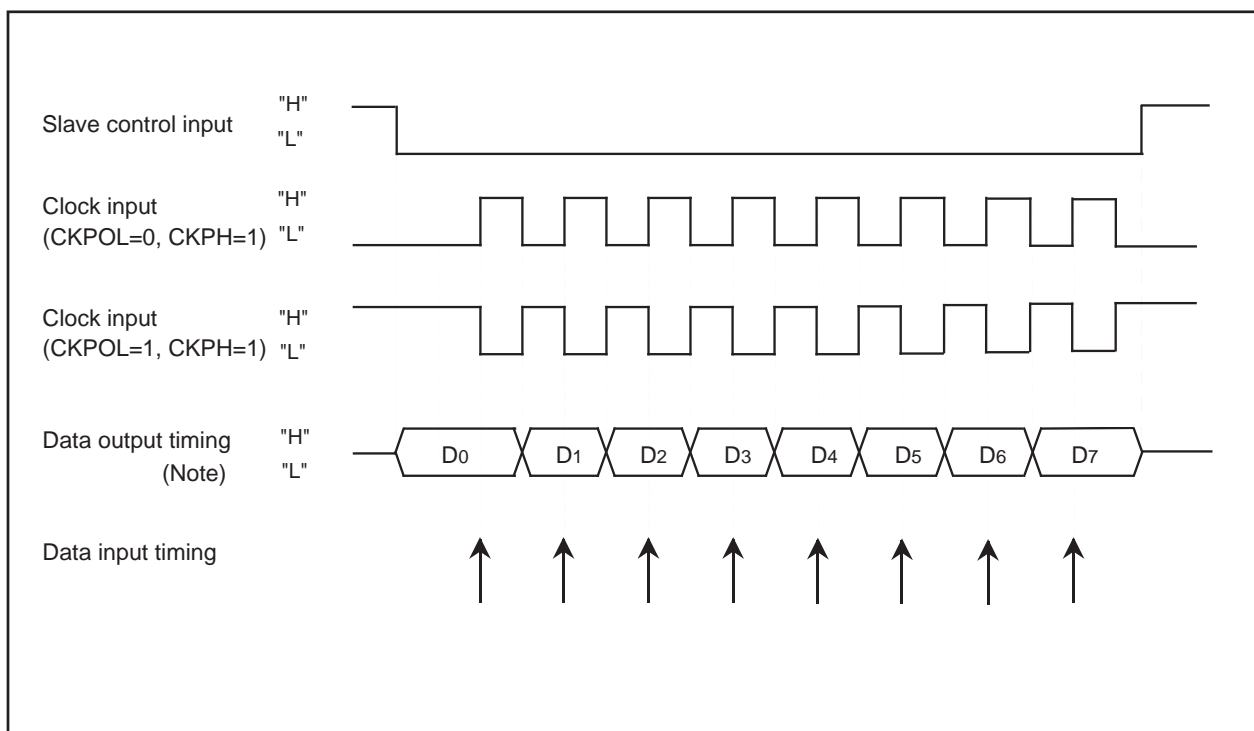


Figure 1.16.8. Transmission and Reception Timing (CKPH=1) in Slave Mode (External Clock)

## SI/O3 and SI/O4

SI/O3 and SI/O4 are exclusive clock-synchronous serial I/Os.

Figure 1.17.1 shows the block diagram of SI/O3 and SI/O4, and Figure 1.17.2 shows the SI/O3 and SI/O4-related registers. SI/O4 is directly connected to IT800 internally.

Table 1.17.1 shows the specifications of SI/O3 and SI/O4.

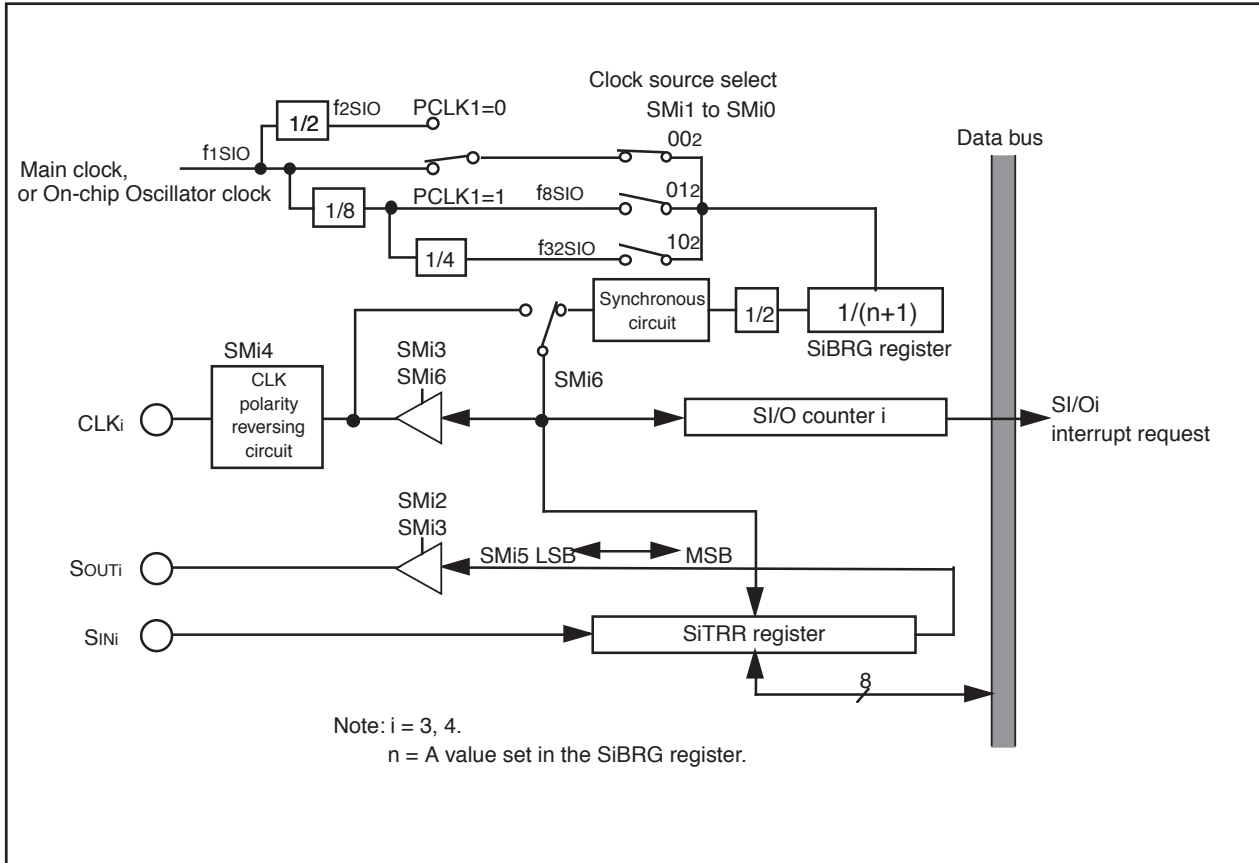


Figure 1.17.1. SI/O3 and SI/O4 Block Diagram

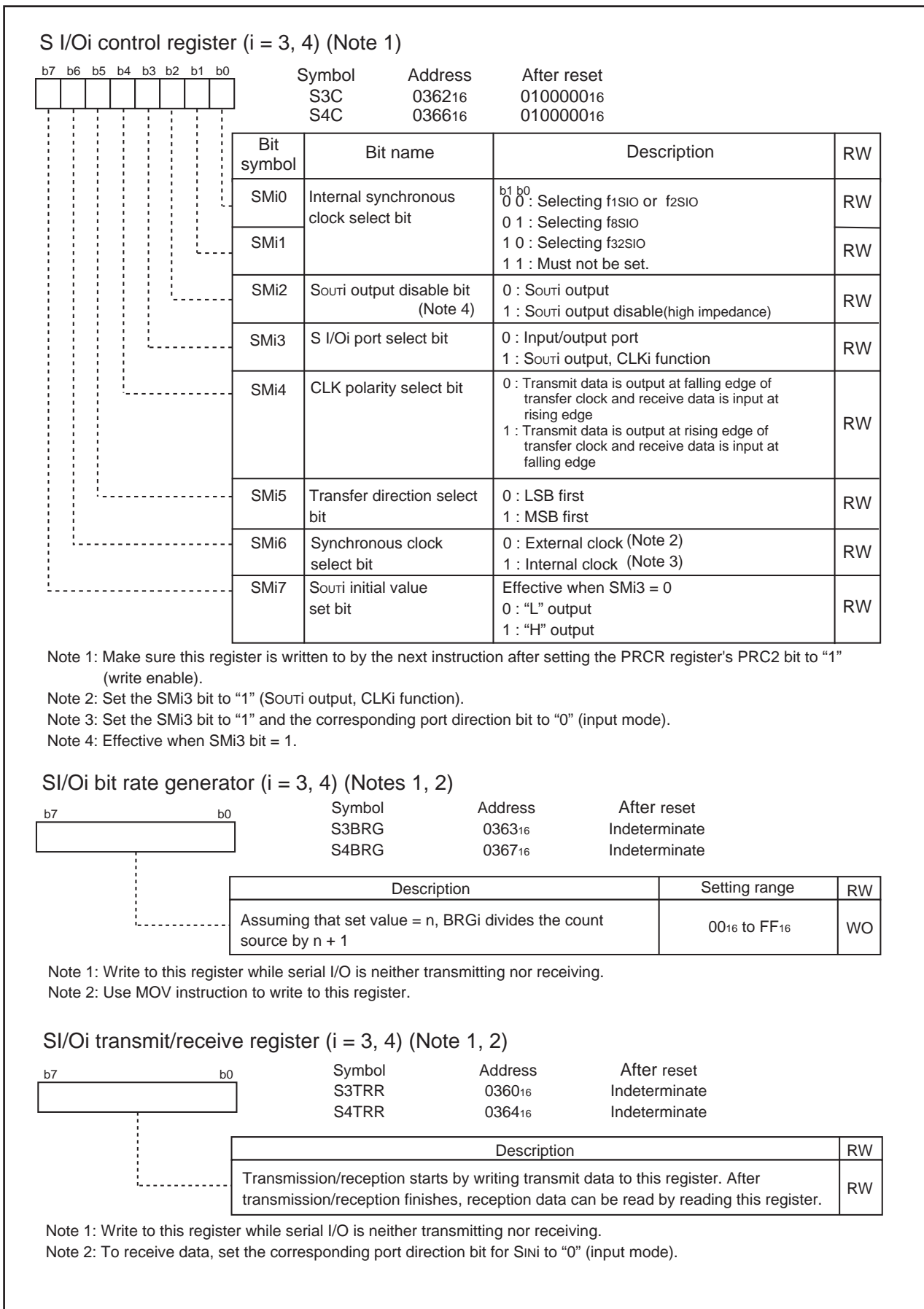


Figure 1.17.2. S3C and S4C Registers, S3BRG and S4BRG Registers, and S3TRR and S4TRR Registers

**Table 1.17.1. SI/O3 and SI/O4 Specifications**

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• SiC (i=3, 4) register's SMi6 bit = "1" (internal clock) : $f_j / 2^{(n+1)}$ $f_j = f_{1SIO}, f_{8SIO}, f_{32SIO}$ . n=Setting value of SiBRG register 0016 to FF16. • SMi6 bit = "0" (external clock) : Input from CLKi pin (Note 1)
Transmission/reception start condition	• Before transmission/reception can start, the following requirements must be met Write transmit data to the SiTRR register (Notes 2, 3)
Interrupt request generation timing	• When SiC register's SMi4 bit = 0 The rising edge of the last transfer clock pulse (Note 4) • When SMi4 = 1 The falling edge of the last transfer clock pulse (Note 4)
CLKi pin function	I/O port, transfer clock input, transfer clock output
SOUTi pin function	I/O port, transmit data output, high-impedance
SINi pin function	I/O port, receive data input
Select function	• LSB first or MSB first selection Whether to start sending/receiving data beginning with bit 0 or beginning with bit 7 can be selected • Function for setting an SOUTi initial value set function When the SiC register's SMi6 bit = 0 (external clock), the SOUTi pin output level while not transmitting can be selected. • CLK polarity selection Whether transmit data is output/input timing at the rising edge or falling edge of transfer clock can be selected.

Note 1: To set the SiC register's SMi6 bit to "0" (external clock), follow the procedure described below.

- If the SiC register's SMi4 bit = 0, write transmit data to the SiTRR register while input on the CLKi pin is high. The same applies when rewriting the SiC register's SMi7 bit.
- If the SMi4 bit = 1, write transmit data to the SiTRR register while input on the CLKi pin is low. The same applies when rewriting the SMi7 bit.
- Because shift operation continues as long as the transfer clock is supplied to the SI/Oi circuit, stop the transfer clock after supplying eight pulses. If the SMi6 bit = 1 (internal clock), the transfer clock automatically stops.

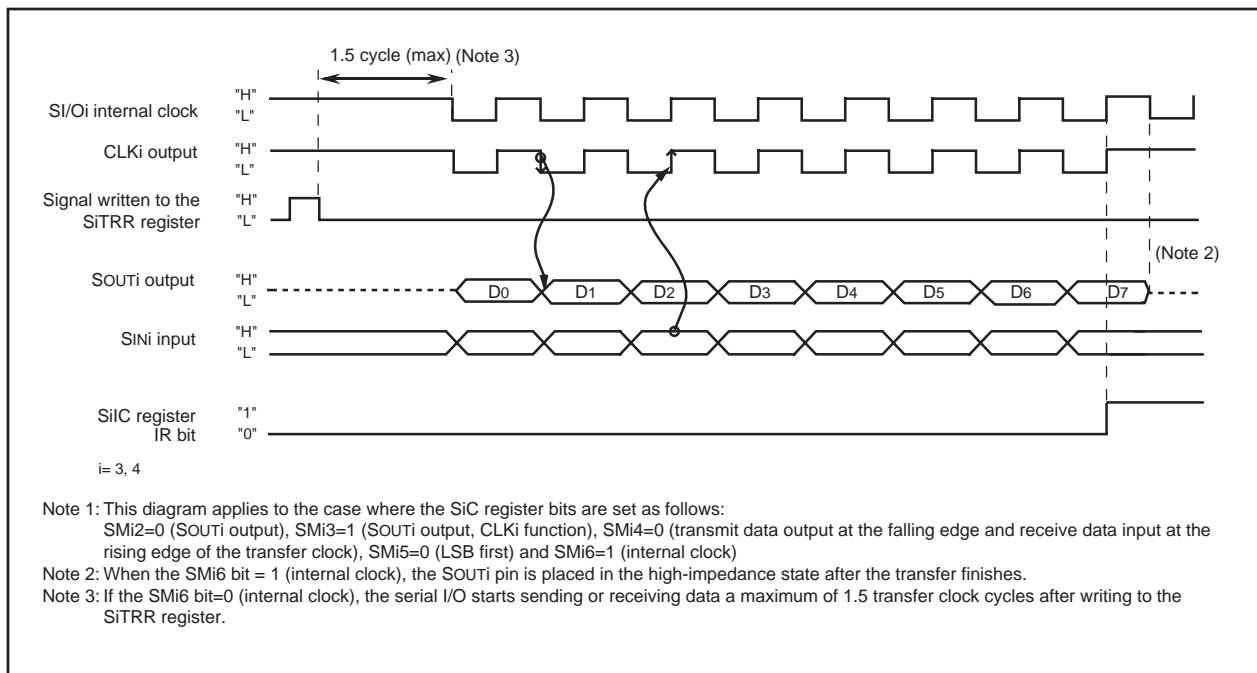
Note 2: Unlike UART0 to UART2, SI/Oi (i = 3 to 4) is not separated between the transfer register and buffer. Therefore, do not write the next transmit data to the SiTRR register during transmission.

Note 3: When the SiC register's SMi6 bit = 1 (internal clock), SOUTi retains the last data for a 1/2 transfer clock period after completion of transfer and, thereafter, goes to a high-impedance state. However, if transmit data is written to the SiTRR register during this period, SOUTi immediately goes to a high-impedance state, with the data hold time thereby reduced.

Note 4: When the SiC register's SMi6 bit = 1 (internal clock), the transfer clock stops in the high state if the SMi4 bit = 0, or stops in the low state if the SMi4 bit = 1.

**(a) SI/Oi Operation Timing**

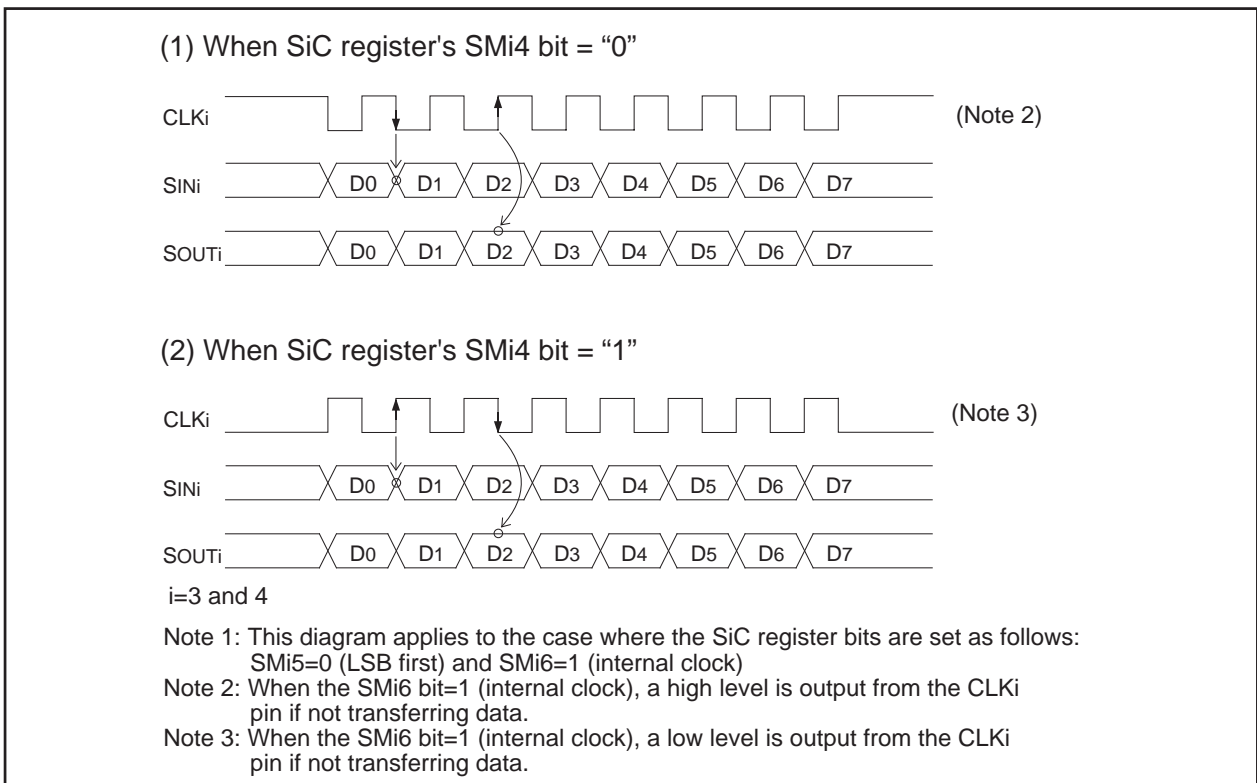
Figure 1.17.3 shows the SI/Oi operation timing



**Figure 1.17.3. SI/Oi Operation Timing**

**(b) CLK Polarity Selection**

The SiC register's SMI4 bit allows selection of the polarity of the transfer clock. Figure 1.17.4 shows the polarity of the transfer clock.



**Figure 1.17.4. Polarity of Transfer Clock**

### (c) Functions for Setting an SOUT<sub>i</sub> Initial Value

If the SiC register's SMi6 bit = 0 (external clock), the SOUT<sub>i</sub> pin output can be fixed high or low when not transferring. Figure 1.17.5 shows the timing chart for setting an SOUT<sub>i</sub> initial value and how to set it.

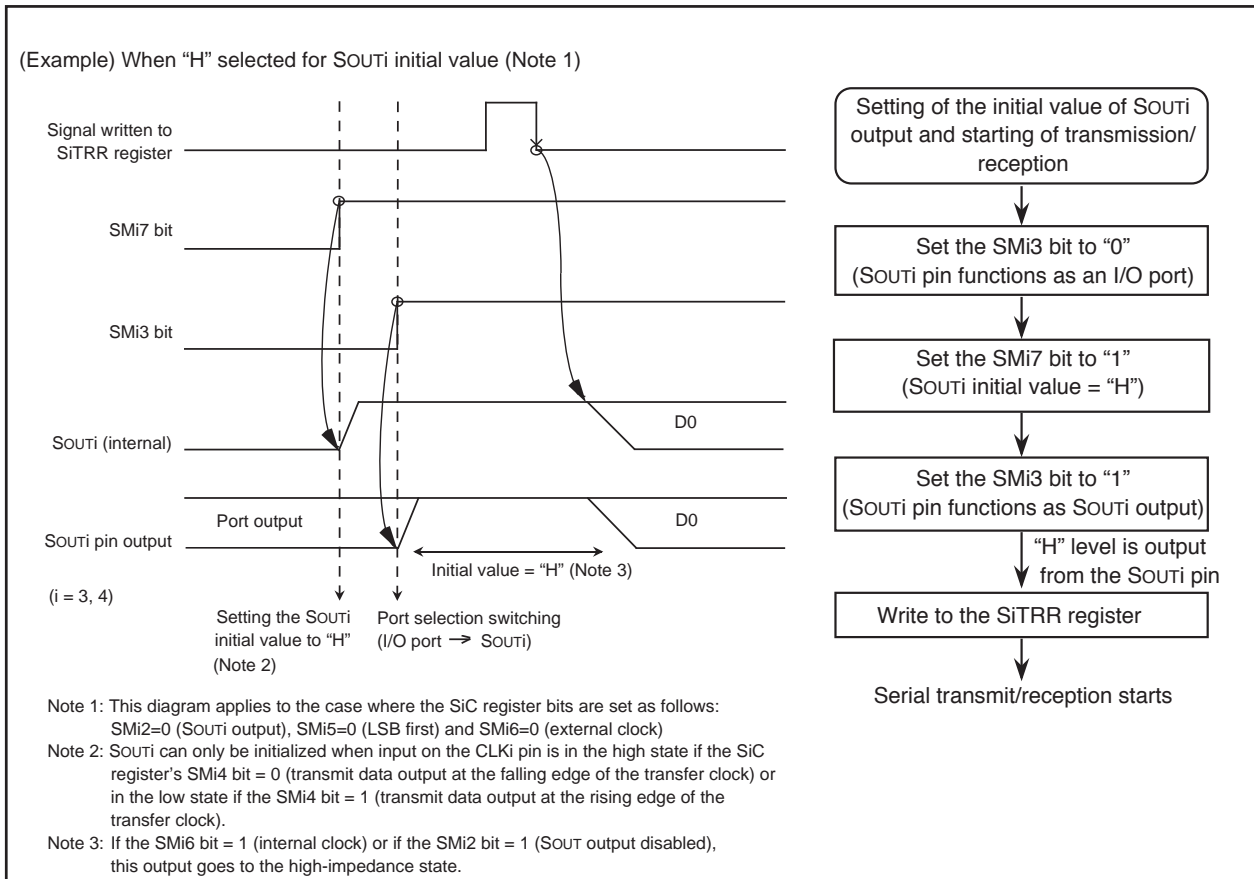


Figure 1.17.5. SOUT<sub>i</sub>'s Initial Value Setting



## Programmable I/O Ports

The programmable input/output ports (hereafter referred to simply as “I/O ports”) consist of 55 lines P1, P4 to P9 (except P85). Each port can be set for input or output every line by using a direction register, and can also be chosen to be or not be pulled high every 4 lines. P85 is an input-only port and does not have a pull-up resistor. There is no external connections for port P1\_0 to P1\_4, P1\_6 to P1\_7, P4\_0 to P4\_7, P5\_0 to P5\_7, P7\_2, P7\_5, P7\_7, P8\_2, P8\_6 to P8\_7, P9\_3 to P9\_7.

Figures 1.18.1 to 1.18.4 show the I/O ports. Figure 1.18.5 shows the I/O pins.

Each pin functions as an I/O port, a peripheral function input/output.

For details on how to set peripheral functions, refer to each functional description in this manual. If any pin is used as a peripheral function input, set the direction bit for that pin to “0” (input mode). Any pin used as an output pin for peripheral functions is directed for output no matter how the corresponding direction bit is set.

### (1) Port Pi Direction Register (PDi Register, i, 4 to 9)

Figure 1.18.6 shows the direction registers.

This register selects whether the I/O port is to be used for input or output. The bits in this register correspond one for one to each port.

No direction register bit for P85 is available.

### (2) Port Pi Register (Pi Register, i = 1, 4 to 9)

Figure 1.18.7 show the Pi registers.

Data input/output to and from external devices are accomplished by reading and writing to the Pi register. The Pi register consists of a port latch to hold the input/output data and a circuit to read the pin status. For ports set for input mode, the input level of the pin can be read by reading the corresponding Pi register, and data can be written to the port latch by writing to the Pi register.

For ports set for output mode, the port latch can be read by reading the corresponding Pi register, and data can be written to the port latch by writing to the Pi register. The data written to the port latch is output from the pin. The bits in the Pi register correspond one for one to each port.

### (3) Pull-up Control Register 0 to Pull-up Control Register 2 (PUR0 to PUR2 Registers)

Figure 1.18.8 shows the PUR0 to PUR2 registers.

The PUR0 to PUR2 register bits can be used to select whether or not to pull the corresponding port high in 4 bit units. The port chosen to be pulled high has a pull-up resistor connected to it when the direction bit is set for input mode.

### (4) Port Control Register

Figure 1.18.9 shows the port control register.

When the P1 register is read after setting the PCR register’s PCR0 bit to “1”, the corresponding port latch can be read no matter how the PD1 register is set.

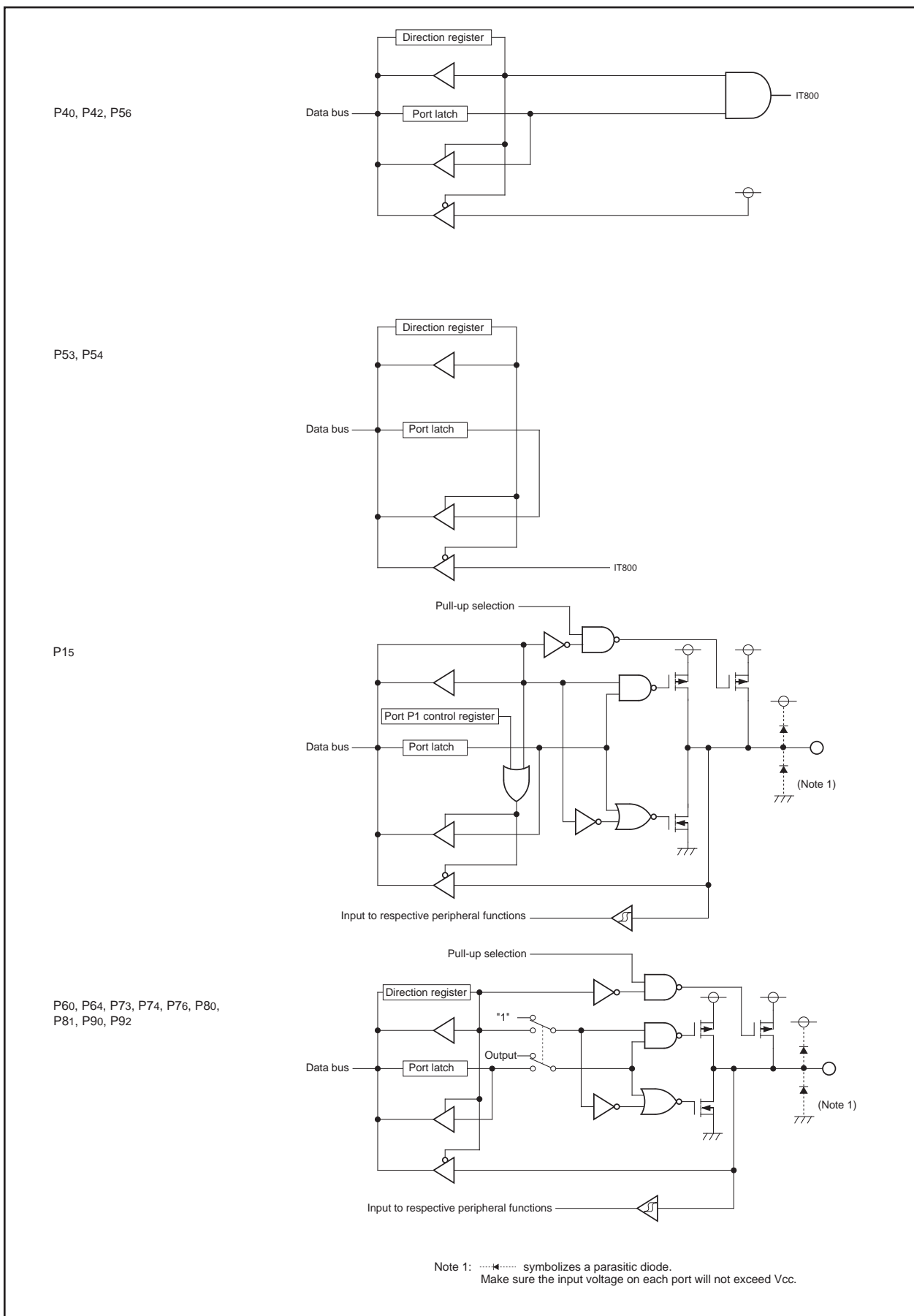


Figure 1.18.1. I/O Ports (1)

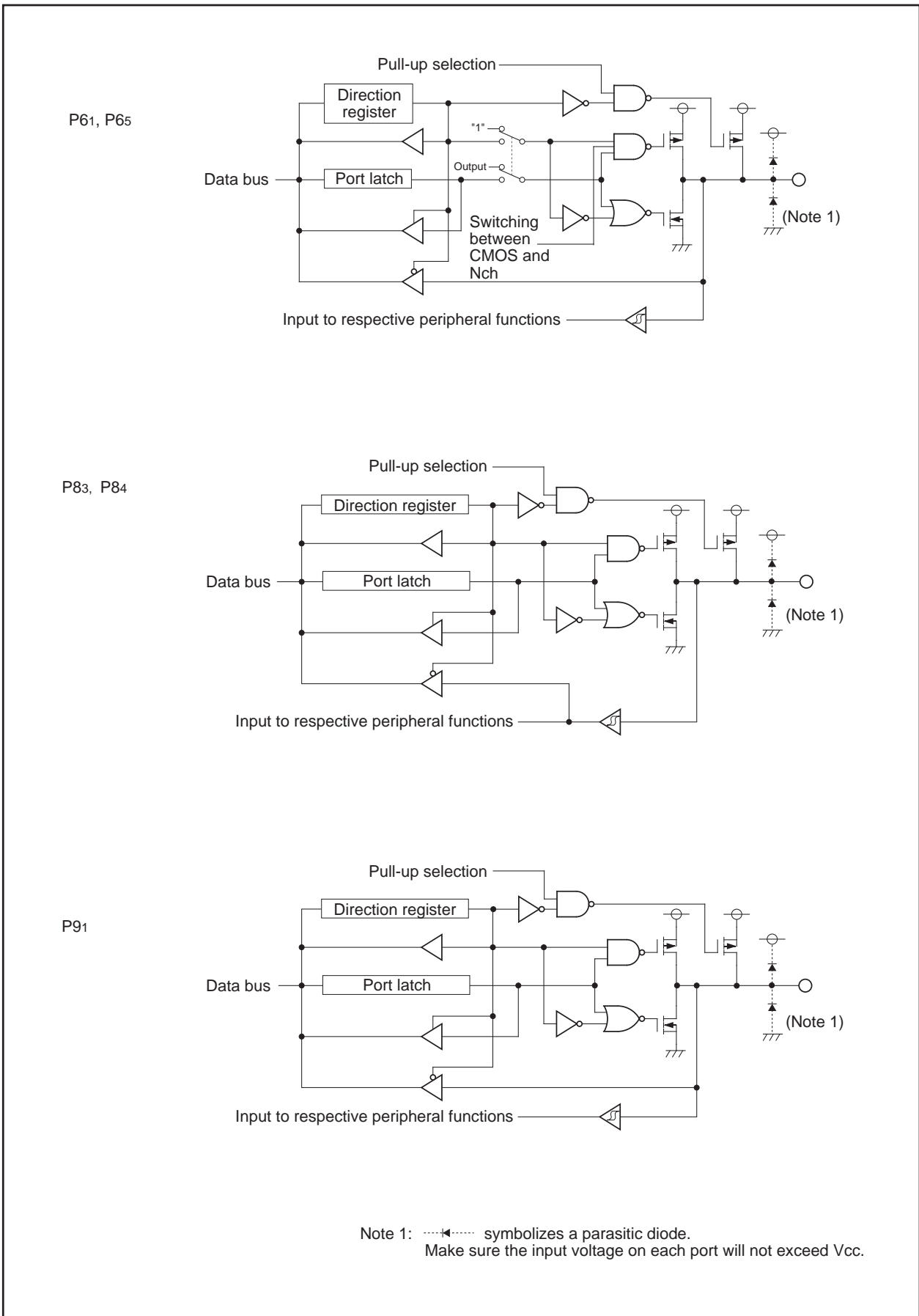


Figure 1.18.2. I/O Ports (2)

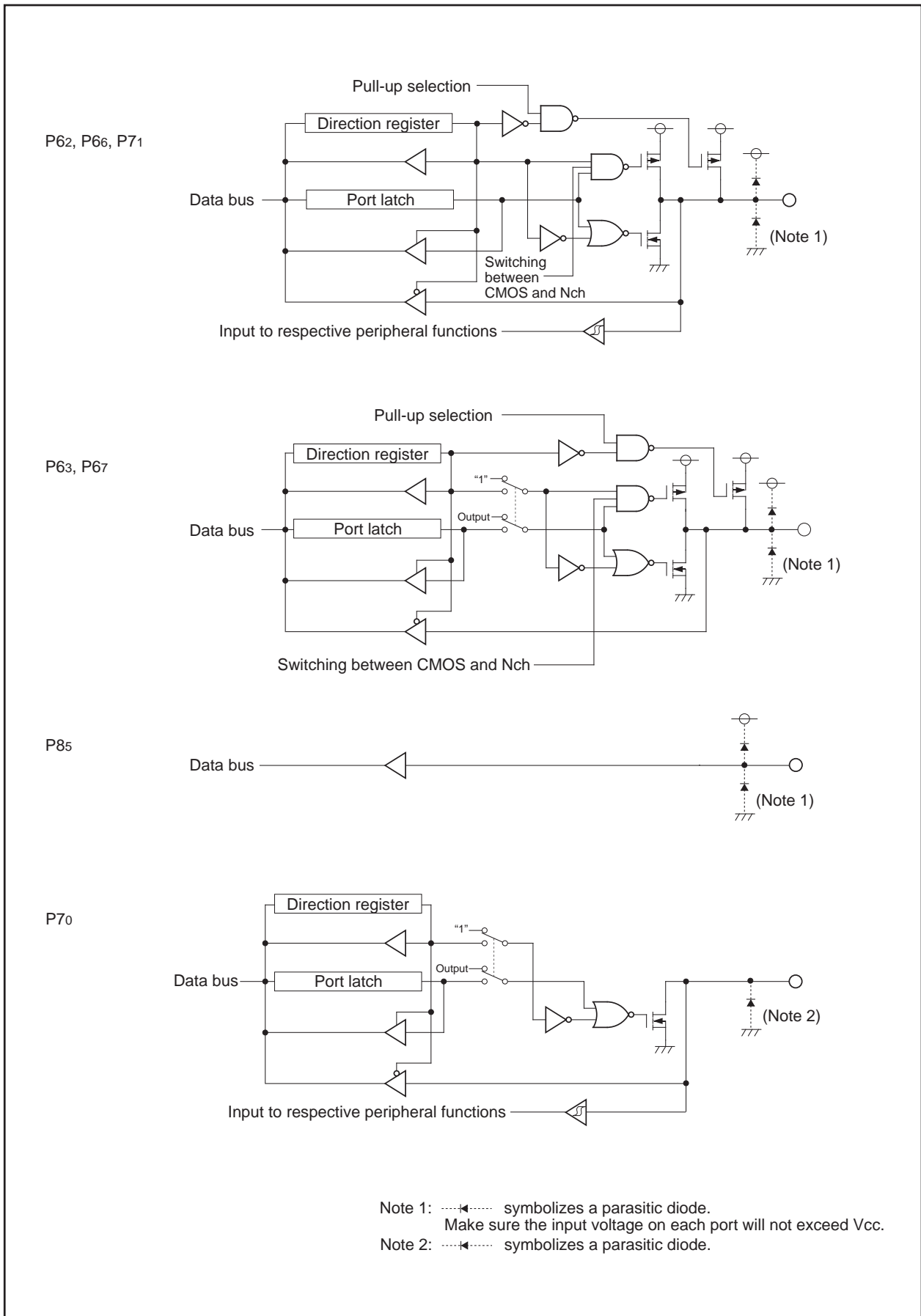


Figure 1.18.3. I/O Ports (3)

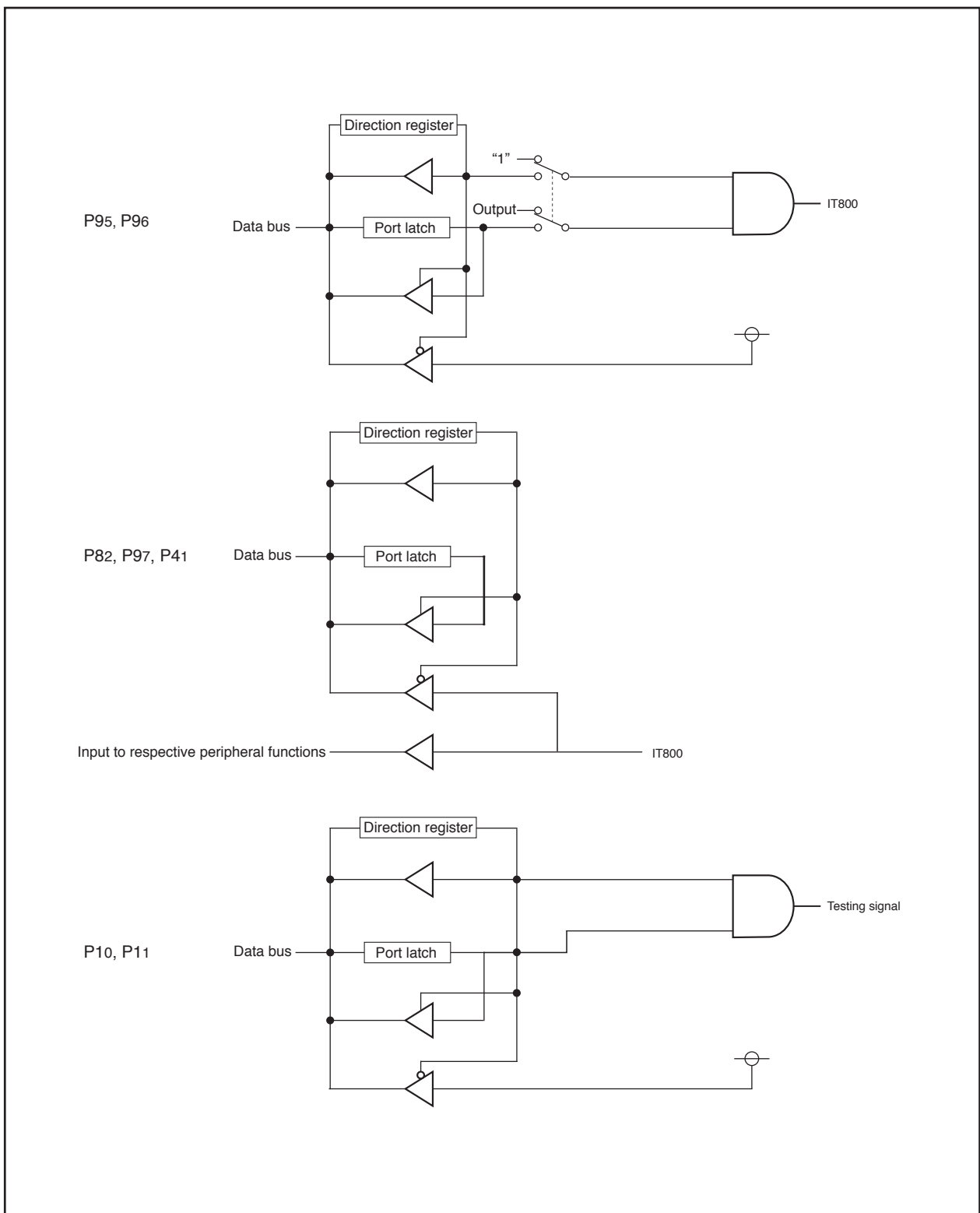


Figure 1.18.4. I/O Ports (4)

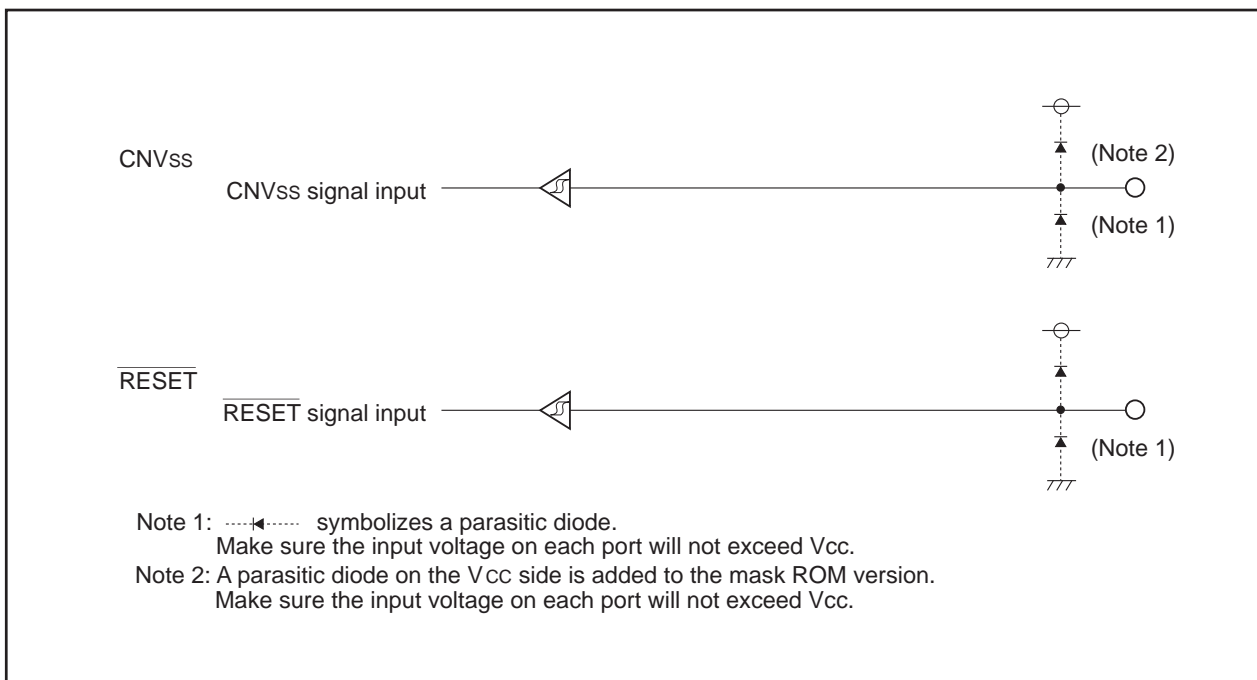


Figure 1.18.5. I/O Pins

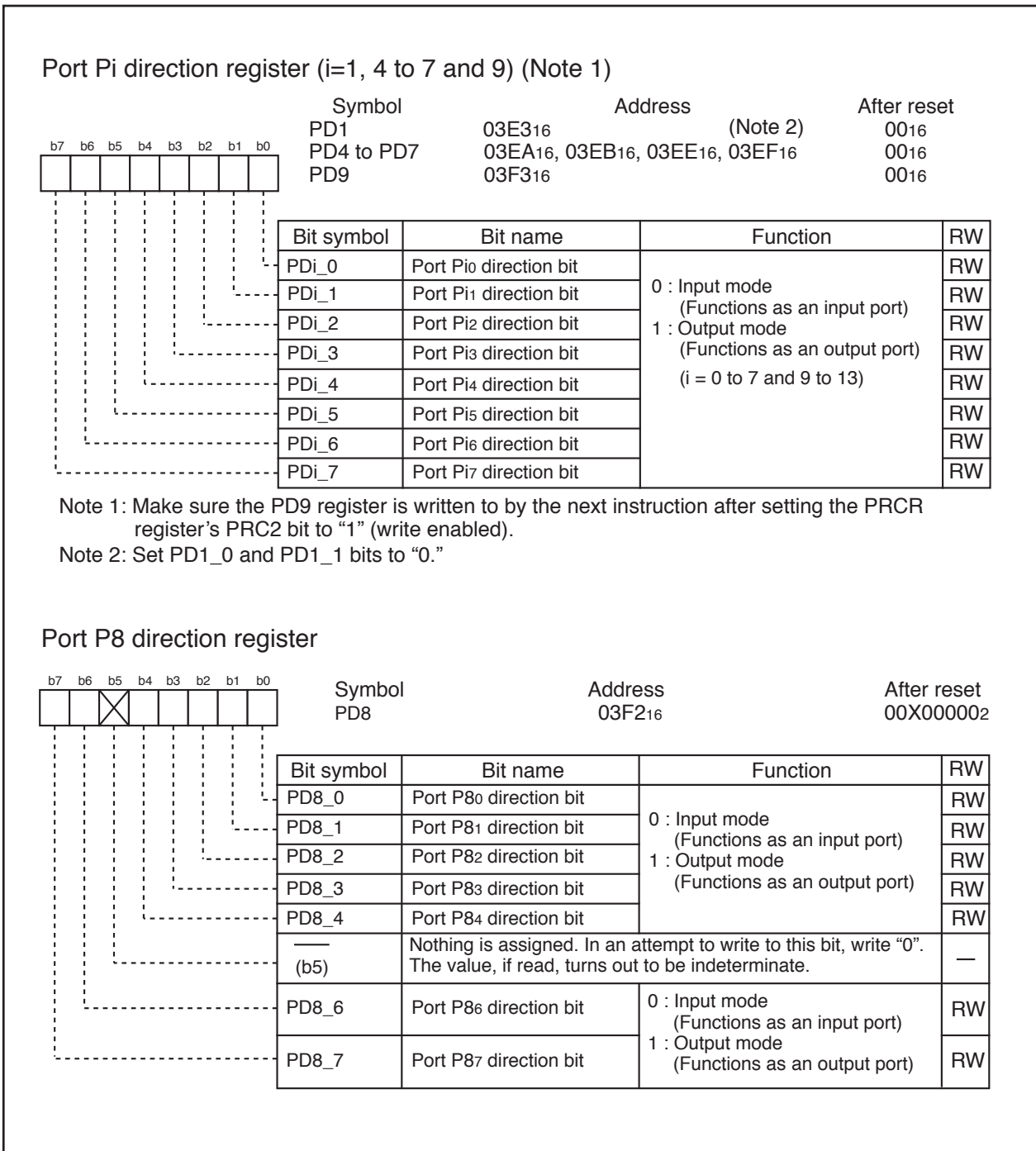


Figure 1.18.6. PD1, PD4 to PD9 Registers

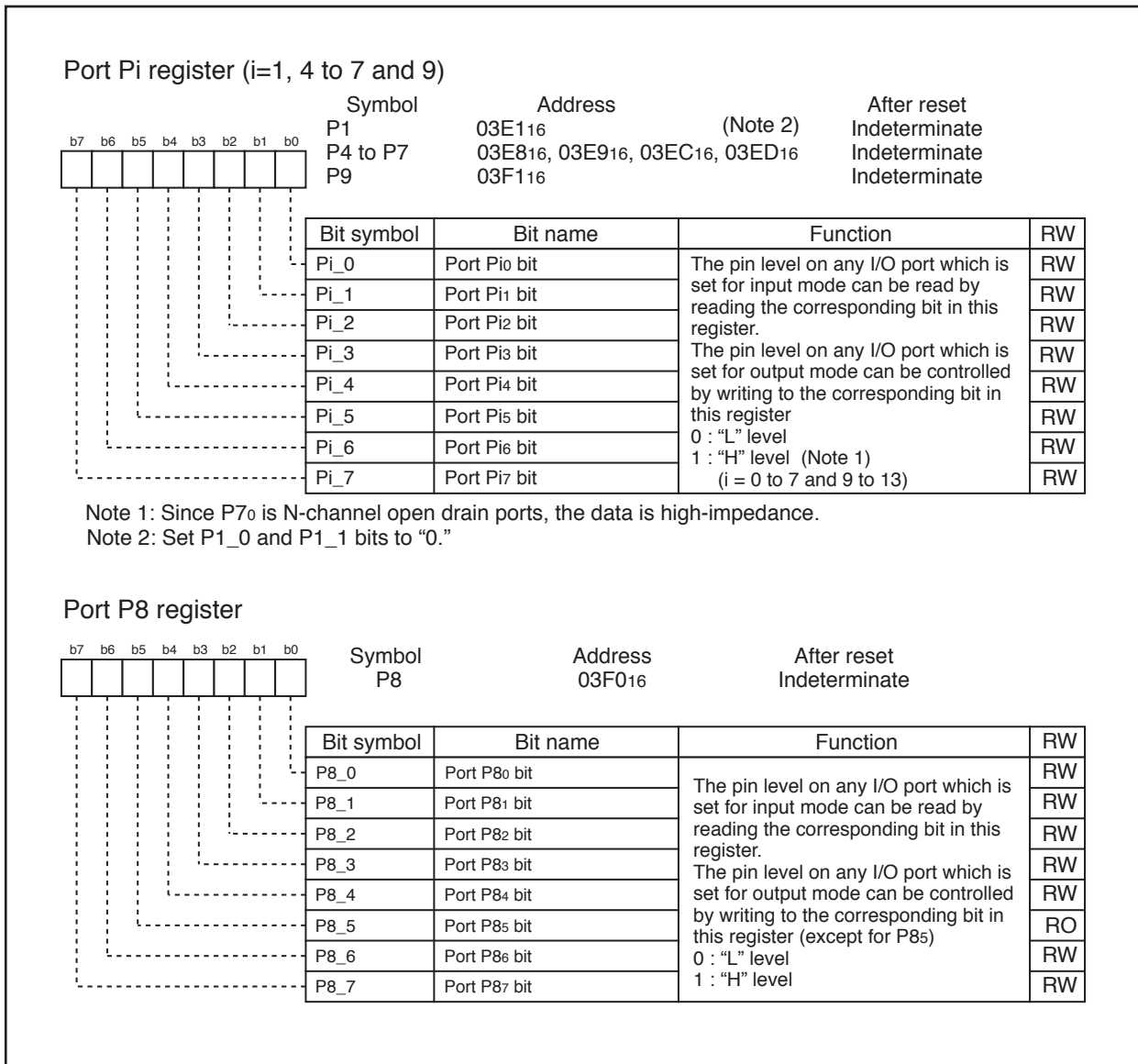


Figure 1.18.7. P1, P4 to P9



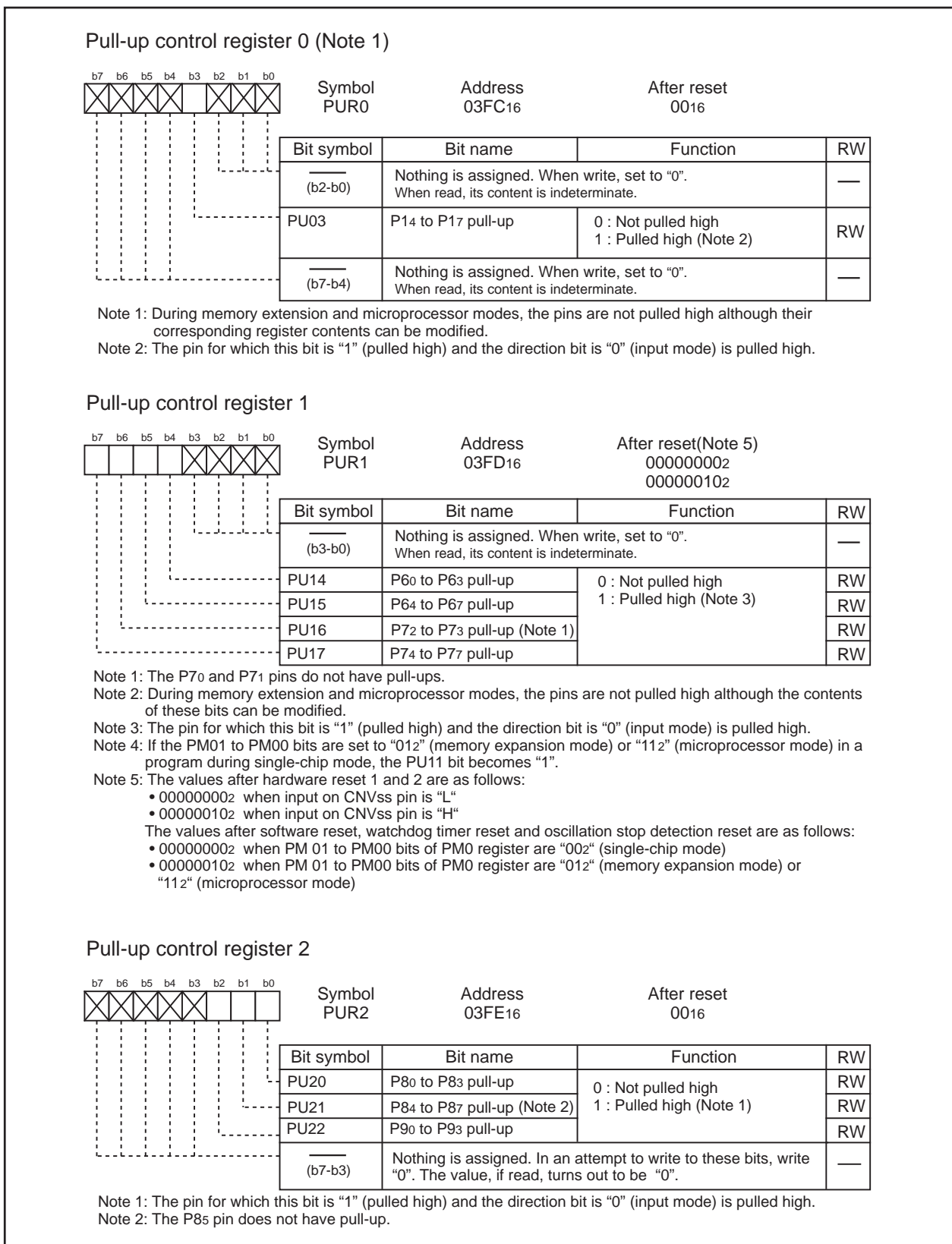


Figure 1.18.8. PUR0 to PUR2 Registers

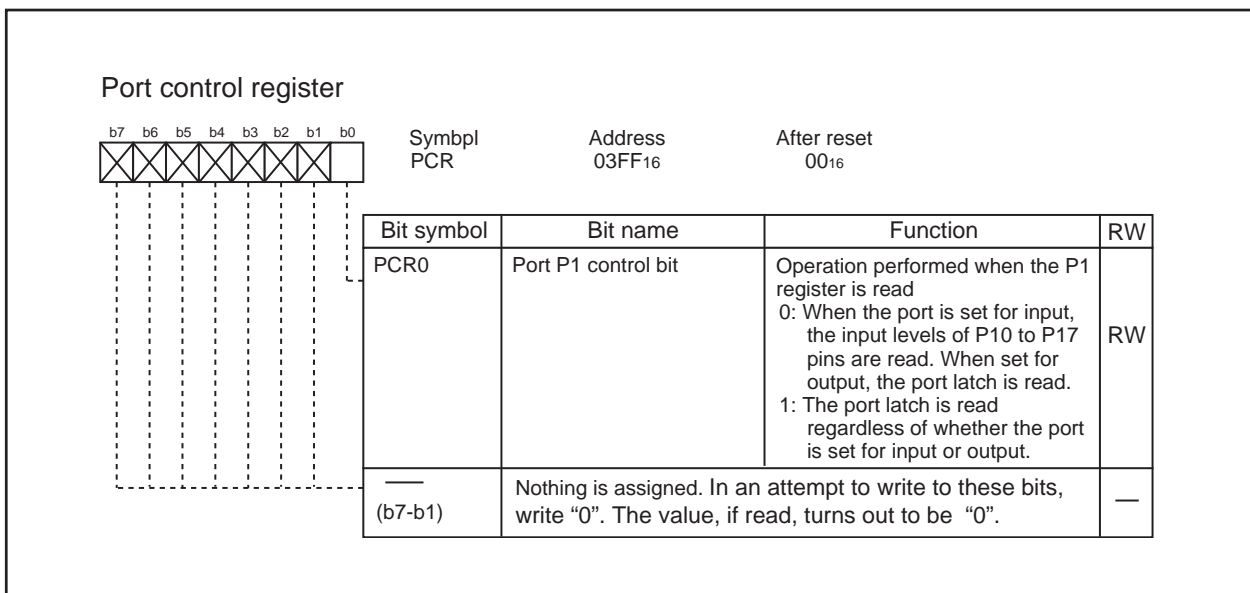


Figure 1.18.9. PCR Register

**Table 1.18.1. Unassigned Pin Handling in Single-chip Mode (Excluding Analog Pins)**

Pin name	Connection
Ports P1, P6 to P9 (excluding P85)	After setting for input mode, connect every pin to V <sub>SS</sub> via a resistor(pull-down); or after setting for output mode, leave these pins open. (2, 3, 4)
XOUT (Note 1)	Open
P85	Connect via resistor to V <sub>CC</sub> (pull-up)
V <sub>CCA</sub>	Connect to V <sub>CC</sub>
V <sub>SSA</sub>	Connect to V <sub>SS</sub>

**NOTES:**

1. With external clock input to X<sub>IN</sub> pin.
2. When setting the port for output mode and leave it open, be aware that the port remains in input mode until it is switched to output mode in a program after reset. For this reason, the voltage level on the pin becomes indeterminate, causing the power supply current to increase while the port remains in input mode. Furthermore, by considering a possibility that the contents of the direction registers could be changed by noise or noise-induced runaway, it is recommended that the contents of the direction registers be periodically reset in software, for the increased reliability of the program.
3. Make sure the unused pins are processed with the shortest possible wiring from the microcomputer pins (within 2 cm).
4. When the port P7\_0 is set for output mode, make sure a low-level signal is output from the pin. The port P7\_0 is N-channel open-drain output.

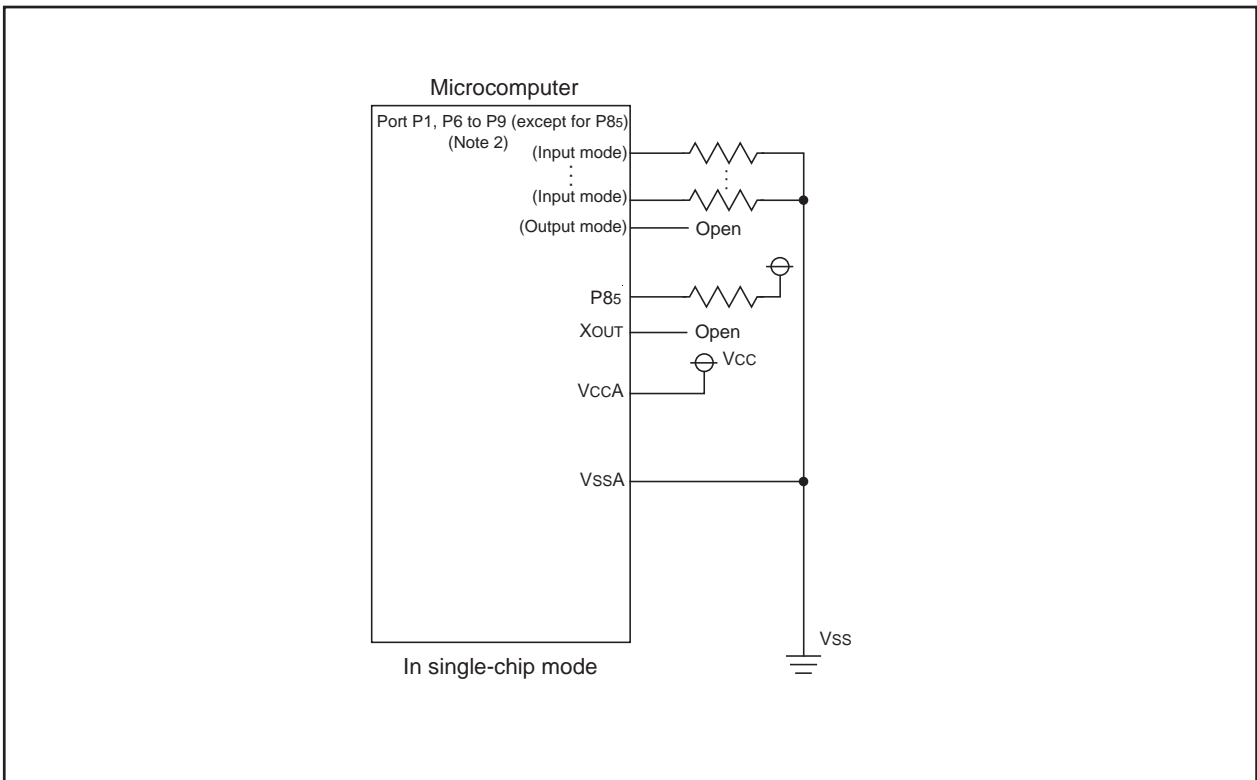


Figure 1.18.10. Unassigned Pins Handling (Excluding Analog Pins)

## Electrical Characteristics

**Table 1.19.1. Absolute Maximum Ratings**

Symbol	Parameter		Condition	Rated value	Unit
V <sub>CC</sub>	Supply voltage		V <sub>CC</sub> =V <sub>CCA</sub>	-0.3 to 4.2	V
V <sub>CCA</sub>	Analog supply voltage		V <sub>CC</sub> =V <sub>CCA</sub>	-0.3 to 4.2	V
V <sub>I</sub>	Input voltage	RESET, CNV <sub>SS</sub> P60 to P67, P71, P73, P74, P76, P80 to P85, P90 to P92, X <sub>IN</sub>		-0.3 to V <sub>CC</sub> +0.3	V
		P70		-0.3 to 6.5	V
V <sub>O</sub>	Output voltage	P60 to P67, P71, P73, P74, P76, P80 to P84, P90 to P92, X <sub>OUT</sub> , TS		-0.3 to V <sub>CC</sub> +0.3	V
		P70		-0.3 to 6.5	V
P <sub>d</sub>	Power dissipation			500	mW
T <sub>opr</sub>	Operating ambient temperature			-20 to 85/-40 to 85/-40 to 105 (Note 1)	°C
T <sub>stg</sub>	Storage temperature			-65 to 150	°C

Note 1: Refer to Tables 1.1.5 and 1.1.6 Product code.

Table 1.19.2. Recommended Operating Conditions (Note 1)

Symbol	Parameter		Standard			Unit
			Min.	Typ.	Max.	
V <sub>CC</sub>	Supply voltage		3.0	3.3	3.6	V
V <sub>CC</sub> A	Analog supply voltage			V <sub>CC</sub>		V
V <sub>SS</sub>	Supply voltage			0		V
V <sub>SS</sub> A	Analog supply voltage			0		V
V <sub>IH</sub>	HIGH input voltage	P60 to P67, P71, P73, P74, P76, P80 to P84, P90 to P92 X <sub>IN</sub> , RESET, CNV <sub>SS</sub>	0.8V <sub>CC</sub>		V <sub>CC</sub>	V
		P70	0.8V <sub>CC</sub>		6.5	V
V <sub>IL</sub>	LOW input voltage	P60 to P67, P70, P71, P73, P74, P76, P80 to P84, P90 to P92 X <sub>IN</sub> , RESET, CNV <sub>SS</sub>	0		0.2V <sub>CC</sub>	V
I <sub>OH</sub> (peak)	HIGH peak output current	P60 to P67, P71, P73, P74, P76, P80 to P84, P90 to P92, TS			-10.0	mA
I <sub>OH</sub> (avg)	HIGH average output current	P60 to P67, P71, P73, P74, P76, P80 to P84, P90 to P92, TS			-5.0	mA
I <sub>OL</sub> (peak)	LOW peak output current	P60 to P67, P70, P71, P73, P74, P76, P80 to P84, P90 to P92, TS			10.0	mA
I <sub>OL</sub> (avg)	LOW average output current	P60 to P67, P70, P71, P73, P74, P76, P80 to P84, P90 to P92, TS			5.0	mA
f (X <sub>IN</sub> )	Main clock input oscillation frequency (Note 4)	V <sub>CC</sub> =3.0 to 3.6V		5.12		MHz
f (Ring)	Ring oscillation frequency			1		MHz
f (BCLK)	CPU operation clock			15.36		MHz
T <sub>SU</sub> (PLL)	PLL frequency synthesizer stabilization wait time	V <sub>CC</sub> =3.0V			50	ms

Note 1: Referenced to V<sub>CC</sub> = 3.0 to 3.6V at Topr = -20 to 85 °C/-40 to 85 °C/-40 to 105 °C unless otherwise specified.

Note 2: The mean output current is the mean value within 100ms.

Note 3: The total I<sub>OL</sub> (peak) for all ports must be 80mA max, the total I<sub>OH</sub> (peak) for all ports must be -40mA max.

**Table 1.19.3. Flash Memory Version Electrical Characteristics (Note 1)**

Symbol	Parameter	Standard			Unit
		Min.	Typ. (Note 2)	Max	
–	Erase/Write cycle (Note 3)	100/1000 (Note 4, 6 )			cycle
–	Word program time (V <sub>cc</sub> =3.3V, T <sub>opr</sub> =25°C)		75	600	μs
–	Block erase time	8Kbyte block		9	s
		16Kbyte block		9	s
		32Kbyte block		9	s
t <sub>PS</sub>	Flash Memory Circuit Stabilization Wait Time			15	μs
–	Data retention time (Note 5)	20			year

Note 1: When not otherwise specified, V<sub>cc</sub> = 3.0 to 3.6V; T<sub>opr</sub> = 0 to 60 °C.

Note 2: V<sub>CC</sub> = 3.3V; T<sub>opr</sub> = 25 °C.

Note 3: Program and Erase Endurance refers to the number of times a block erase can be performed. If the program and erase endurance is n (n=100, 1,000), each block can be erased n times. For example, if a 8Kbytes block 0 is erased after writing 1 word data 4096 times, each to a different address, this counts as one program and erase endurance. Data cannot be written to the same address more than once without erasing the block. (Rewrite prohibited)

Note 4: Maximum number of E/W cycles for which operation is guaranteed.

Note 5: T<sub>opr</sub> = 55°C.

Note 6: The program area for U3 and U5 is 100 E/W cycles; the program area for U7 and U9 is 1,000 E/W cycles.

Note 7: Customers desiring E/W failure rate information should contact their Renesas technical support representative.

**Table 1.19.4. Flash Memory Version Program/Erase Voltage and Read Operation Voltage Characteristics (at T<sub>opr</sub> = 0 to 60°C)**

Flash program, erase voltage	Flash read operation voltage
V <sub>cc</sub> = 3.3 V ± 0.3 V	V <sub>cc</sub> = 3.0 to 3.6 V

**Table 1.19.5. Power Supply Circuit Timing Characteristics**

Symbol	Parameter	Measuring condition	Standard			Unit
			Min.	Typ.	Max.	
td(P-R)	Time for internal power supply stabilization during powering-on	$V_{CC} = 3.0$ to $3.6V$			2	ms
td(R-S)	STOP release time				150	$\mu s$
td(M-L)	Time for internal power supply stabilization when main clock oscillation starts				50	$\mu s$

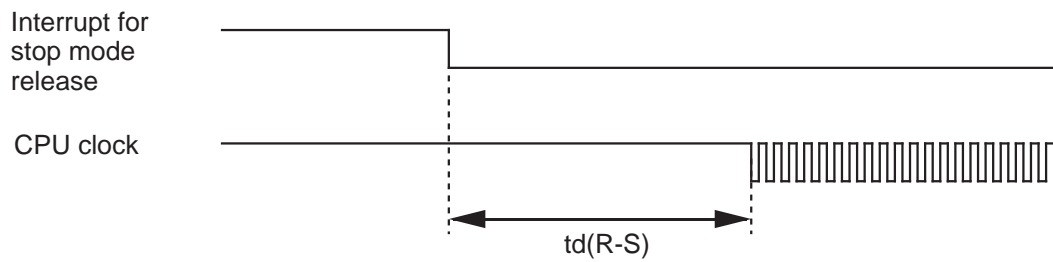




Table 1.19.6. Electrical Characteristics (Note)

Symbol	Parameter		Measuring condition	Standard			Unit
				Min.	Typ.	Max.	
V <sub>OH</sub>	HIGH output voltage	P60 to P67, P71, P73, P74, P76, P80 to P84, P90 to P92, TS	I <sub>OH</sub> = -1mA	V <sub>CC</sub> -0.5		V <sub>CC</sub>	V
V <sub>OH</sub>	HIGH output voltage	X <sub>OUT</sub>	I <sub>OH</sub> = -0.1mA	V <sub>CC</sub> -0.5		V <sub>CC</sub>	V
V <sub>OL</sub>	LOW output voltage	P60 to P67, P71, P73, P74, P76, P80 to P84, P90 to P92, TS	I <sub>OL</sub> =1mA			0.5	V
V <sub>OL</sub>	LOW output voltage	X <sub>OUT</sub>	I <sub>OL</sub> =0.1mA			0.5	V
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis	TA0IN, TA1IN, TA4IN, INT1 to INT3, CTS0 to CTS2, SCL, SDA, CLK0 to CLK4, TA4OUT, RxD0 to RxD2, S <sub>N3</sub>		0.2		0.8	V
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis	RESET		0.2	(0.7)	1.8	V
I <sub>IH</sub>	HIGH input current	P60 to P67, P70, P71, P73, P74, P76, P80 to P84, P90 to P92 X <sub>IN</sub> , RESET, CNV <sub>SS</sub>	V <sub>I</sub> =3V			4.0	μA
I <sub>IL</sub>	LOW input current	P60 to P67, P70, P71, P73, P74, P76, P80 to P84, P90 to P92 X <sub>IN</sub> , RESET, CNV <sub>SS</sub>	V <sub>I</sub> =0V			-4.0	μA
R <sub>PULLUP</sub>	Pull-up resistance	P60 to P67, P71, P73, P74, P76, P80 to P84, P90 to P92	V <sub>I</sub> =0V	66	160	500	kΩ
R <sub>XIN</sub>	Feedback resistance	X <sub>IN</sub>			3.0		MΩ

Note : Referenced to V<sub>CC</sub> = 3.0 to 3.6V, V<sub>SS</sub> = 0V at T<sub>opr</sub> = -20 to 85 °C/-40 to 85 °C/-40 to 105 °C, f(BCLK) = 15.36 MHz unless otherwise specified.

**Table 1.19.7. Electrical Characteristics (2) (Note 1)**

Symbol	Parameter		Measuring condition		Standard			Unit
					Min.	Typ.	Max.	
I <sub>CC</sub>	Power supply current (V <sub>CC</sub> =2.7 to 3.6V)	In single-chip mode, the output pins are open and other pins are V <sub>SS</sub>	Flash memory	f(BCLK)=15.36 MHz, No division		70	95	mA
			Flash memory Program	f(BCLK)=10MHz, V <sub>CC1</sub> =3.0V		TBD		mA
			Flash memory Erase	f(BCLK)=10MHz, V <sub>CC1</sub> =3.0V		TBD		mA

Note : Referenced to V<sub>CC</sub> = 3.0 to 3.6V, V<sub>SS</sub> = 0V at Topr = -20 to 85 °C/-40 to 85 °C/-40 to 105 °C, f(BCLK) = 15.36 MHz unless otherwise specified.

**Timing Requirements****( $V_{CC} = 3V$ ,  $V_{SS} = 0V$ , at  $T_{opr} = -20$  to  $85^{\circ}C$ /-  $40$  to  $85^{\circ}C$ /-  $40$  to  $105^{\circ}C$  unless otherwise specified)****Table 1.19.8. External Clock Input**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c$	External clock input cycle time	195.3		ns
$t_{w(H)}$	External clock input HIGH pulse width	80		ns
$t_{w(L)}$	External clock input LOW pulse width	80		ns
$t_r$	External clock rise time		18	ns
$t_f$	External clock fall time		18	ns

**Timing Requirements**(V<sub>CC</sub> = 3V, V<sub>SS</sub> = 0V, at T<sub>opr</sub> = – 20 to 85°C/– 40 to 85°C/– 40 to 105°C unless otherwise specified)**Table 1.19.9. Timer A Input (Counter Input in Event Counter Mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
t <sub>c</sub> (TA)	TAiIN input cycle time	150		ns
t <sub>w</sub> (TAH)	TAiIN input HIGH pulse width	60		ns
t <sub>w</sub> (TAL)	TAiIN input LOW pulse width	60		ns

**Table 1.19.10. Timer A Input (Gating Input in Timer Mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
t <sub>c</sub> (TA)	TAiIN input cycle time	600		ns
t <sub>w</sub> (TAH)	TAiIN input HIGH pulse width	300		ns
t <sub>w</sub> (TAL)	TAiIN input LOW pulse width	300		ns

**Table 1.19.11. Timer A Input (External Trigger Input in One-shot Timer Mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
t <sub>c</sub> (TA)	TAiIN input cycle time	300		ns
t <sub>w</sub> (TAH)	TAiIN input HIGH pulse width	150		ns
t <sub>w</sub> (TAL)	TAiIN input LOW pulse width	150		ns

**Table 1.19.12. Timer A Input (External Trigger Input in Pulse Width Modulation Mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
t <sub>w</sub> (TAH)	TAiIN input HIGH pulse width	150		ns
t <sub>w</sub> (TAL)	TAiIN input LOW pulse width	150		ns

**Table 1.19.13. Timer A Input (Counter Increment/decrement Input in Event Counter Mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
t <sub>c</sub> (UP)	TAiOUT input cycle time	3000		ns
t <sub>w</sub> (UPH)	TAiOUT input HIGH pulse width	1500		ns
t <sub>w</sub> (UPL)	TAiOUT input LOW pulse width	1500		ns
t <sub>su</sub> (UP-TiN)	TAiOUT input setup time	600		ns
t <sub>h</sub> (TiN-UP)	TAiOUT input hold time	600		ns

**Table 1.19.14. Timer A Input (Two-phase Pulse Input in Event Counter Mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
t <sub>c</sub> (TA)	TAiIN input cycle time	2		μs
t <sub>su</sub> (TAiN-TAOUT)	TAiOUT input setup time	500		ns
t <sub>su</sub> (TAOUT-TAiN)	TAiIN input setup time	500		ns

**Timing Requirements****( $V_{CC} = 3V$ ,  $V_{SS} = 0V$ , at  $T_{opr} = -20$  to  $85^{\circ}C$ /- 40 to  $85^{\circ}C$ /- 40 to  $105^{\circ}C$  unless otherwise specified)****Table 1.19.15. Serial I/O**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(CK)}$	CLKi input cycle time	300		ns
$t_{w(CKH)}$	CLKi input HIGH pulse width	150		ns
$t_{w(CKL)}$	CLKi input LOW pulse width	150		ns
$t_{d(C-Q)}$	TxDi output delay time		160	ns
$t_{h(C-Q)}$	TxDi hold time	0		ns
$t_{su(D-C)}$	RxDi input setup time	100		ns
$t_{h(C-D)}$	RxDi input hold time	90		ns

**Table 1.19.16. External Interrupt  $\overline{INTi}$  Input**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{w(INH)}$	$\overline{INTi}$ input HIGH pulse width	380		ns
$t_{w(INL)}$	$\overline{INTi}$ input LOW pulse width	380		ns

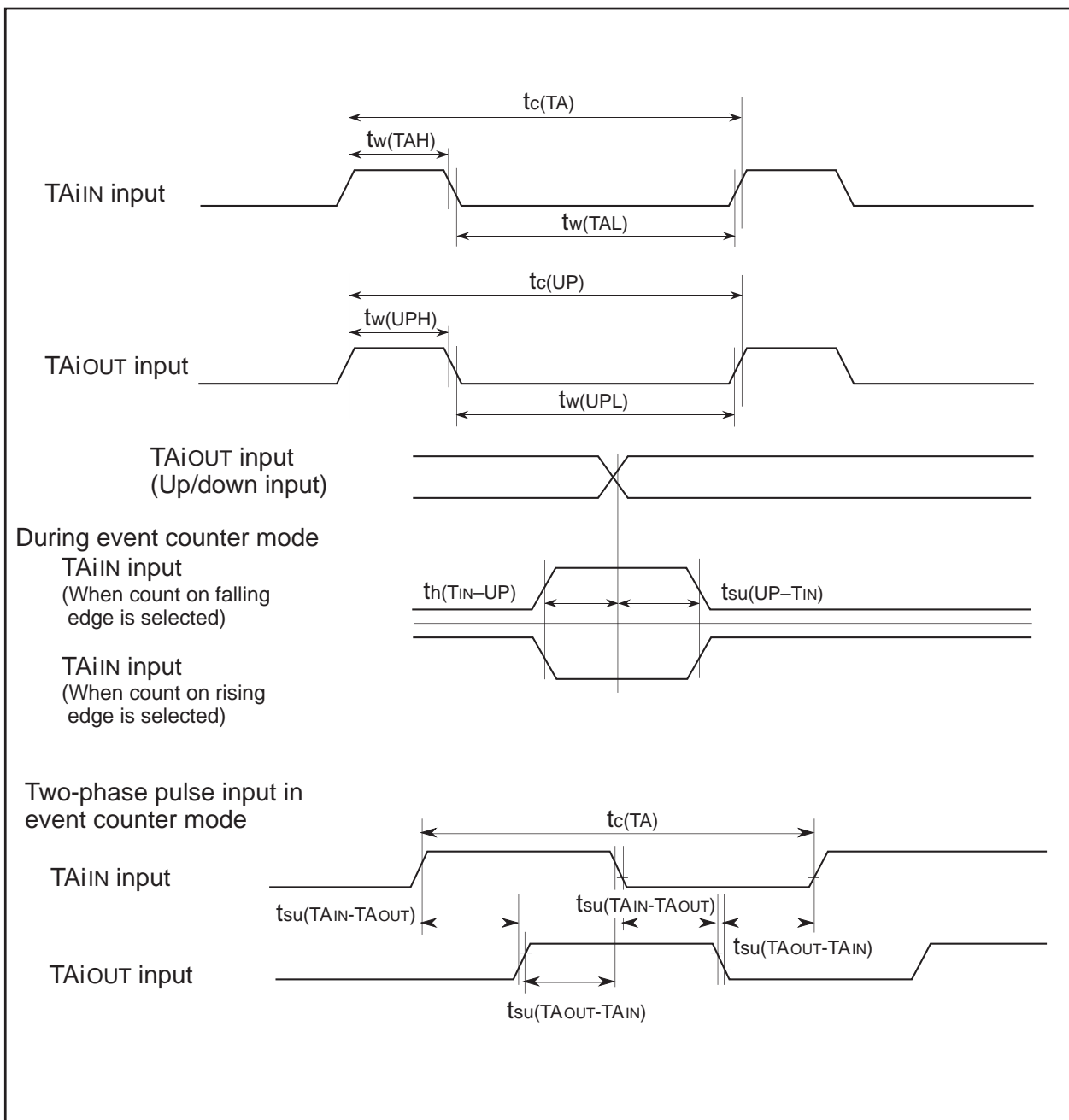


Figure 1.19.1. Timing Diagram (1)

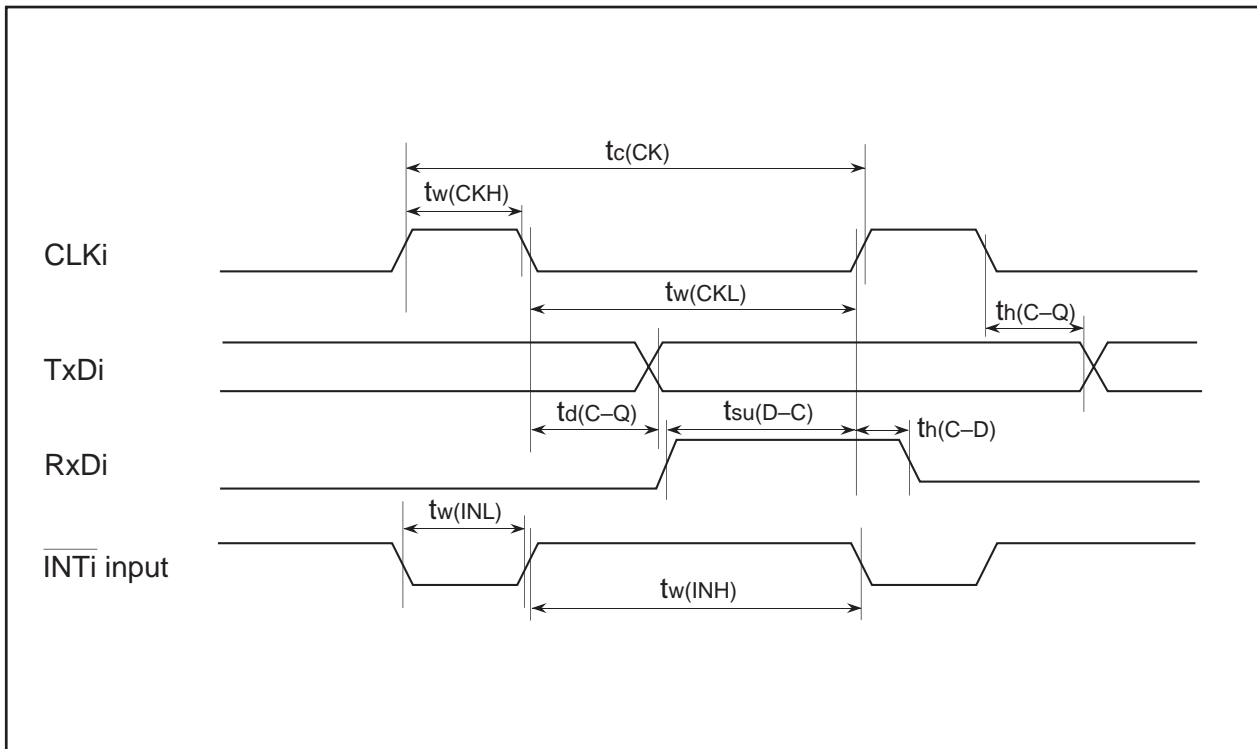


Figure 1.19.2. Timing Diagram (2)

# Flash Memory Version

## Flash Memory Performance

The flash memory version has three modes—CPU rewrite, standard serial input/output, and parallel input/output modes—in which its internal flash memory can be operated on.

Note: About the parallel programmer of exclusive use, there is no schedule of development at the present time.

Table 1.20.1 shows the outline performance of flash memory version (see Table 1.1.1 for the items not listed in Table 1.20.1.).

**Table 1.20.1. Flash Memory Version Specifications**

Item		Specification
Flash memory operating mode		2 modes (CPU rewrite, standard serial I/O)
Erase block		See Figure 1.20.1 Flash Memory Block Diagram
Program method		In units of word
Erase method		Block erase
Program, erase control method		Program and erase controlled by software command
Protect method		The block 0 and block 1 are write protected by bit FMR02.
Number of commands		5 commands
Program/Erase Endurance(Note)	Block 0 to 4 (program area)	100 times or 1,000 times (See Tables 1.1.5 and 1.1.6 Product code.)
Data Retention		20 years (Topr = 55°C)
ROM code protection		Standard serial I/O mode is supported.

Note: Program and erase endurance definition

Program and erase endurance are the erase endurance of each block. If the program and erase endurance are n times (n=100,1,000), each block can be erased n times. For example, if a 8-Kbyte block 0 is erased after writing 1 word data 4096 times, each to different addresses, this is counted as one program and erasure.

However, data cannot be written to the same address more than once without erasing the block. (Rewrite disabled)

**Table 1.20.2. Flash Memory Rewrite Modes Overview**

Flash memory rewrite mode	CPU rewrite mode	Standard serial I/O mode
Function	The user ROM area is rewritten by executing software commands from the CPU. EW0 mode: Can be rewritten in any area other than the flash memory EW1 mode: Can be rewritten in the flash memory	The user ROM area is rewritten by using a dedicated serial programmer. Standard serial I/O mode 1: Clock sync serial I/O Standard serial I/O mode 2: UART
Areas which can be rewritten	User ROM area	User ROM area
Operation mode	Single chip mode Memory expansion mode (EW0 mode) Boot mode (EW0 mode)	Boot mode
ROM programmer	None	Serial programmer

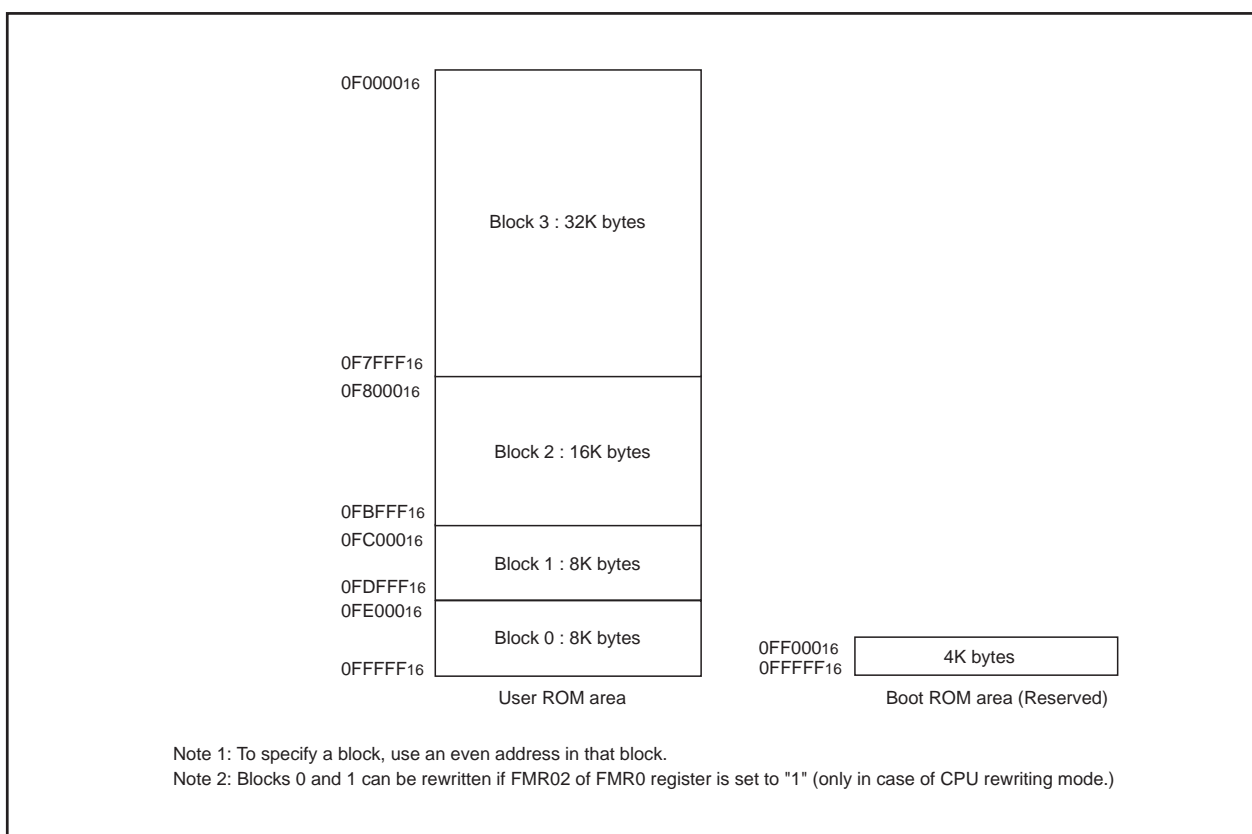


## 1. Memory Map

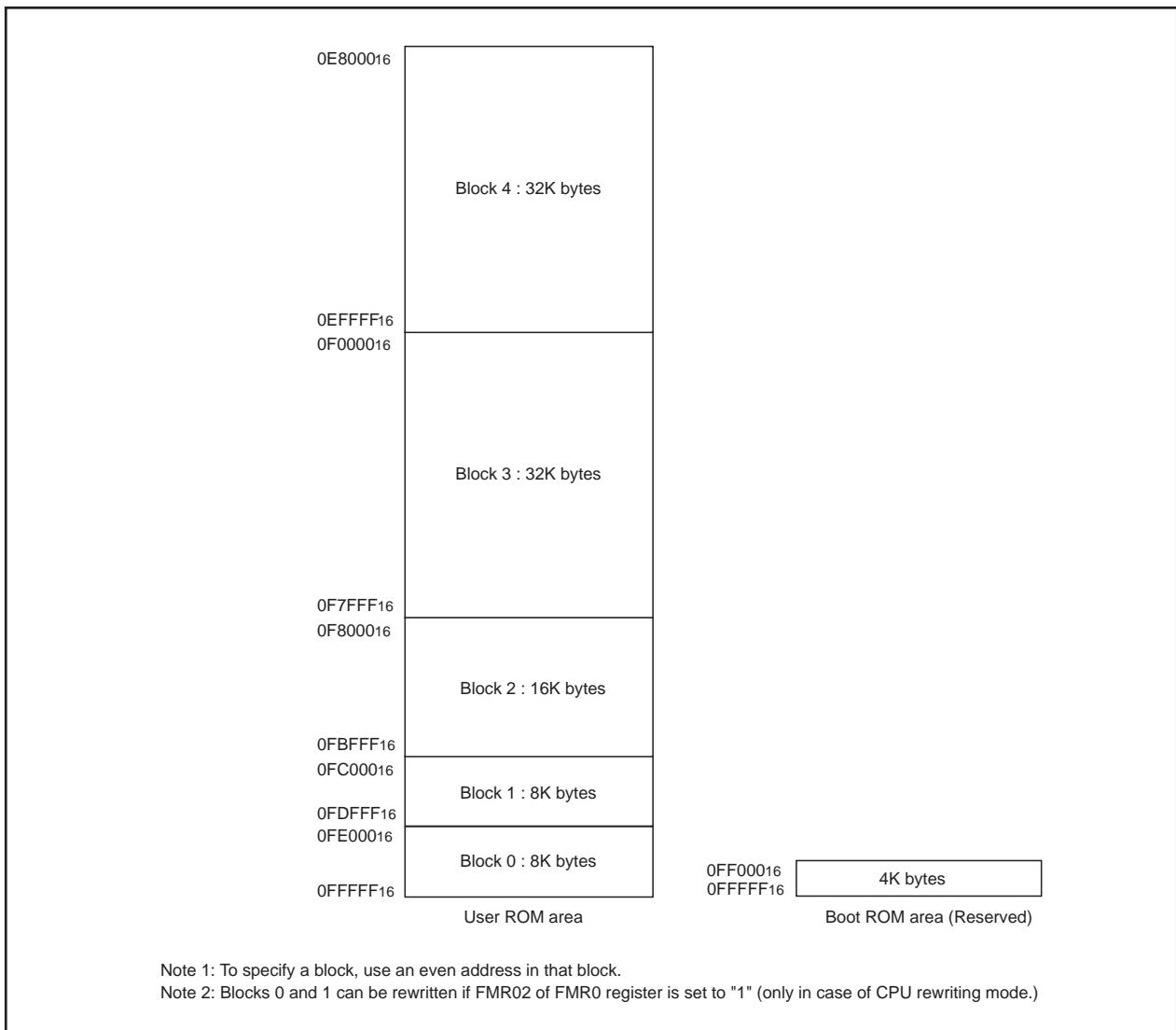
The ROM in the flash memory version is separated between a user ROM area and a boot ROM area. Figures 1.20.1 and 1.20.2 show the block diagram of flash memory.

The user ROM area is divided into several blocks, so that memory can be erased one block at a time. The user ROM area can be rewritten in all of CPU rewrite, standard serial input/output, and parallel input/output modes.

The boot ROM area is reserved. The rewrite control program for standard serial I/O mode is stored in this area before shipment, so that the area cannot be rewritten.



**Figure 1.20.1. Flash Memory Block Diagram (ROM Capacity 64K bytes)**



**Figure 1.20.2. Flash Memory Block Diagram (ROM Capacity 96K bytes)**

## Boot Mode

After a hardware reset which is performed by applying a high-level signal to the CNVss and P15 pins, the microcomputer is placed in boot mode, thereby executing the program in the boot ROM area.

The boot ROM area contains a standard serial input/output mode based rewrite control program which was stored in it when shipped from the factory.

## Functions To Prevent Flash Memory from Rewriting

To prevent the flash memory from being read or rewritten easily, parallel input/output mode has a ROM code protect and standard serial input/output mode has an ID code check function.

### • ROM Code Protect Function

The ROM code protect function inhibits the flash memory from being read or rewritten during parallel input/output mode. Figure 1.20.3 shows the ROMCP register.

The ROMCP register is located in the user ROM area. The ROMCP1 bit consists of two bits. The ROM code protect function is enabled by clearing one or both of two ROMCP1 bits to "0" when the ROMCR bits are not '002,' with the flash memory thereby protected against reading or rewriting. Conversely, when the ROMCR bits are '002' (ROM code protect removed), the flash memory can be read or rewritten. Once the ROM code protect function is enabled, the ROMCR bits cannot be changed during parallel input/output mode. Therefore, use standard serial input/output or other modes to rewrite the flash memory.

### • ID Code Check Function

Use this function in standard serial input/output mode. Unless the flash memory is blank, the ID codes sent from the programmer and the ID codes written in the flash memory are compared to see if they match. If the ID codes do not match, the commands sent from the programmer are not accepted. The ID code consists of 8-bit data, the areas of which, beginning with the first byte, are 0FFFDF<sub>16</sub>, 0FFFE3<sub>16</sub>, 0FFFEB<sub>16</sub>, 0FFFEF<sub>16</sub>, 0FFFF3<sub>16</sub>, 0FFFF7<sub>16</sub>, and 0FFFFB<sub>16</sub>. Prepare a program in which the ID codes are preset at these addresses and write it in the flash memory.

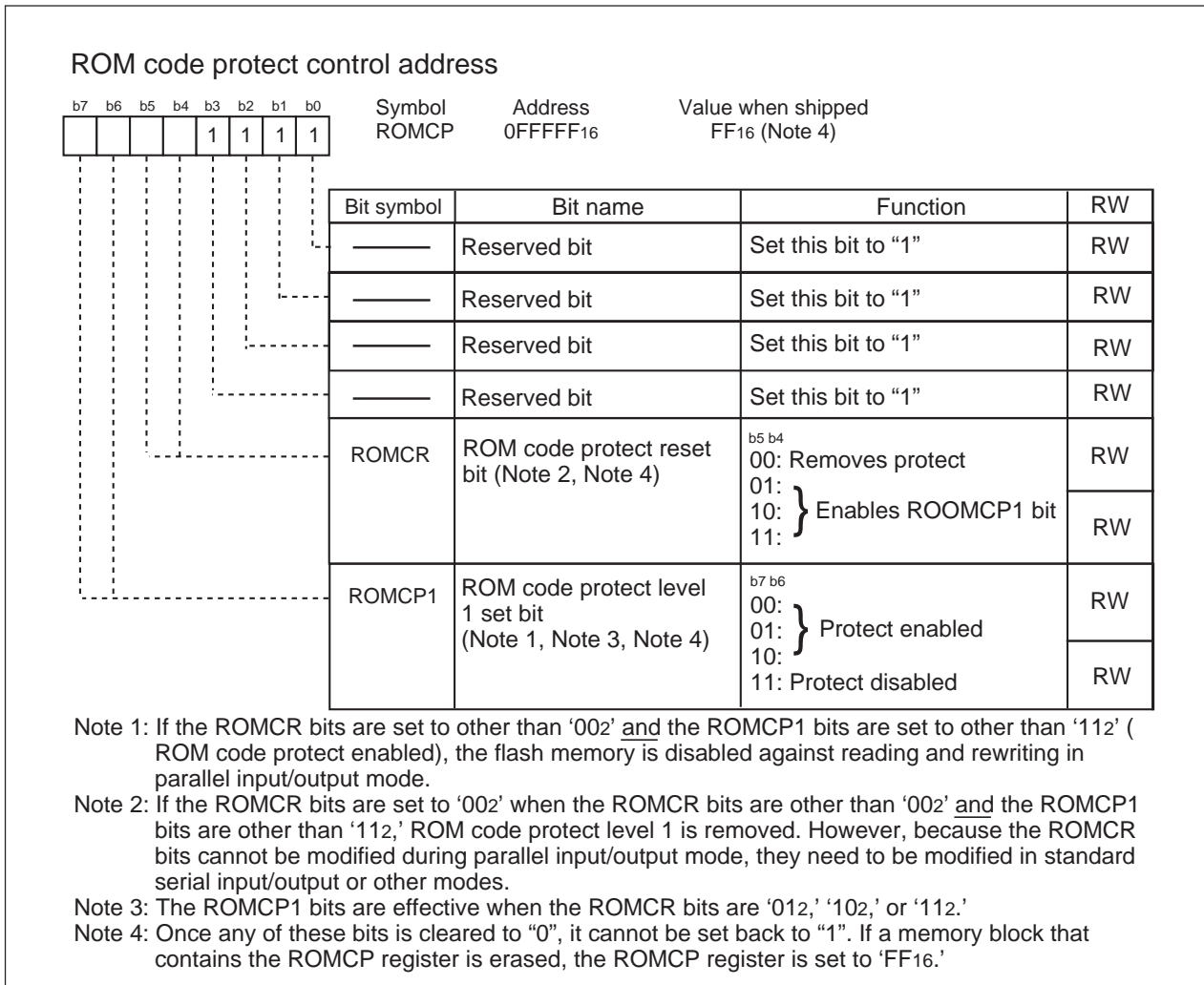


Figure 1.20.3. ROMCP Register

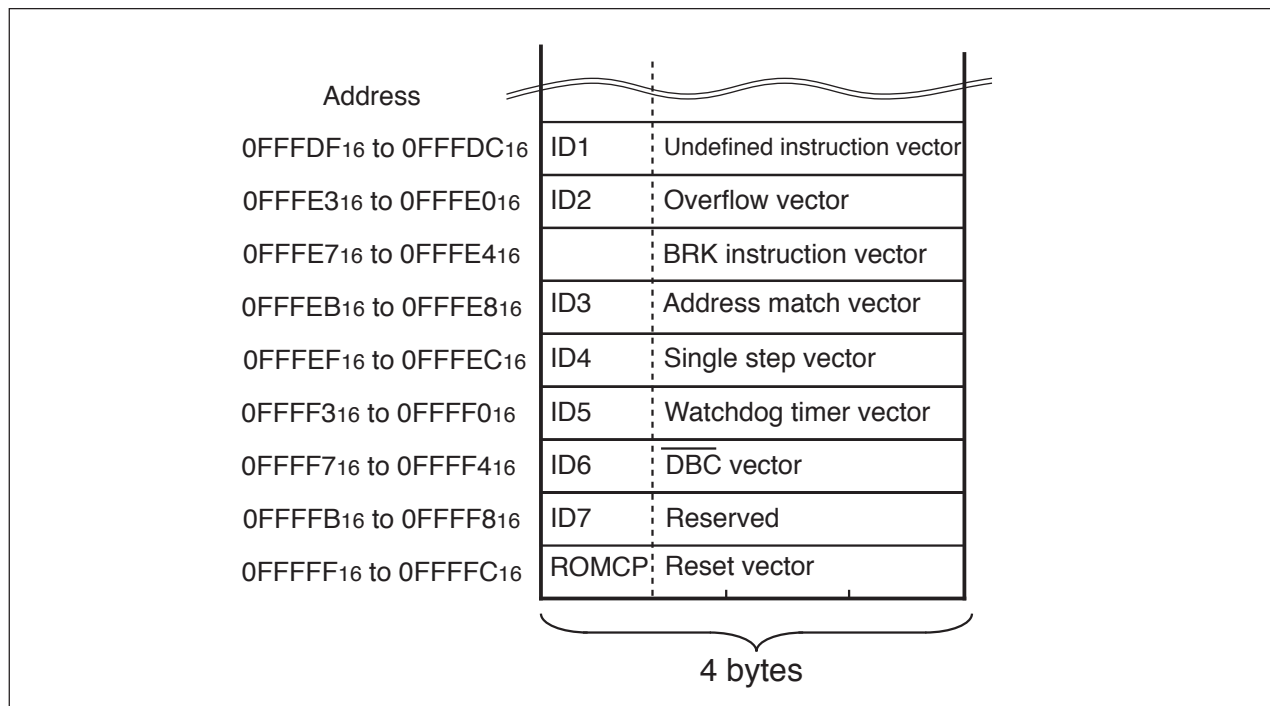


Figure 1.20.4. Address for ID Code Stored

## CPU Rewrite Mode

In CPU rewrite mode, the user ROM area can be rewritten by executing software commands from the CPU. Therefore, the user ROM area can be rewritten directly while the microcomputer is mounted on-board without having to use a ROM programmer, etc.

Make sure the Program and the Block Erase commands are executed only on each block in the user ROM area.

During CPU rewrite mode, the user ROM area be operated on in either Erase Write 0 (EW0) mode or Erase Write 1 (EW1) mode. Table 1.21.1 lists the differences between Erase Write 0 (EW0) and Erase Write 1 (EW1) modes.

**Table 1.21.1. EW0 Mode and EW1 Mode**

Item	EW0 mode	EW1 mode
Operation mode	• Single chip mode	Single chip mode
Areas in which a rewrite control program can be located	• User ROM area	User ROM area
Areas in which a rewrite control program can be executed	Must be transferred to any area other than the flash memory (e.g., RAM) before being executed	Can be executed directly in the user ROM area
Areas which can be rewritten	User ROM area	User ROM area However, this does not include the area in which a rewrite control program exists
Software command limitations	None	• Program, Block Erase command Cannot be executed on any block in which a rewrite control program exists • Read Status Register command Cannot be executed
Modes after Program or Erase	Read Status Register mode	Read Array mode
CPU status during Auto Write and Auto Erase	Operating	Hold state (I/O ports retain the state in which they were before the command was executed) <sup>(Note)</sup>
Flash memory status detection	• Read the FMR0 register's FMR00, FMR06, and FMR07 bits in a program • Execute the Read Status Register command to read the status register's SR7, SR5, and SR4 flags.	Read the FMR0 register's FMR00, FMR06, and FMR07 bits in a program

Note: Make sure no interrupts (except NMI and watchdog timer interrupts) and DMA transfers will occur.

- **EW0 Mode**

The microcomputer is placed in CPU rewrite mode by setting the FMR0 register's FMR01 bit to "1" (CPU rewrite mode enabled), ready to accept commands. In this case, because the FMR1 register's FMR11 bit = 0, EW0 mode is selected. The FMR01 bit can be set to "1" by writing "0" and then "1" in succession. Use software commands to control program and erase operations. Read the FMR0 register or status register to check the status of program or erase operation at completion.

- **EW1 Mode**

EW1 mode is selected by setting FMR11 bit to "1" (by writing "0" and then "1" in succession) after setting the FMR01 bit to "1" (by writing "0" and then "1" in succession).

Read the FMR0 register to check the status of program or erase operation at completion. The status register cannot be read during EW1 mode.

When an erase/program operation is initiated the CPU halts all program execution until the operation is completed.

Figure 1.21.1 shows the FMR0 and FMR1 registers.

### **FMR00 Bit**

This bit indicates the operating status of the flash memory. The bit is “0” when the Program or Erase is running; otherwise, the bit is “1”.

### **FMR01 Bit**

The microcomputer is made ready to accept commands by setting the FMR01 bit to “1” (CPU rewrite mode).

### **FMR02 Bit**

When FMR02 bit is “0” (rewriting is disable), block 0 and block 1 do not receive the command of a program and block erase.

### **FMSTP Bit**

This bit is provided for initializing the flash memory control circuits, as well as for reducing the amount of current consumed in the flash memory. The internal flash memory is disabled against access by setting the FMSTP bit to “1”. Therefore, the FMSTP bit must be written to by a program in other than the flash memory.

In the following cases, set the FMSTP bit to “1”:

- When flash memory access resulted in an error while erasing or programming in EW0 mode (FMR00 bit not reset to “1” (ready))

### **FMR06 Bit**

This is a read-only bit indicating the status of auto program operation. The bit is set to “1” when a program error occurs; otherwise, it is cleared to “0”. For details, refer to the description of the full status check.

### **FMR07 Bit**

This is a read-only bit indicating the status of auto erase operation. The bit is set to “1” when an erase error occurs; otherwise, it is cleared to “0”. For details, refer to the description of the full status check.

Figure 1.21.2 and 1.21.3 show the setting and resetting of EW0 mode and EW1 mode, respectively.

### **FMR11 Bit**

Setting this bit to “1” places the microcomputer in EW1 mode.

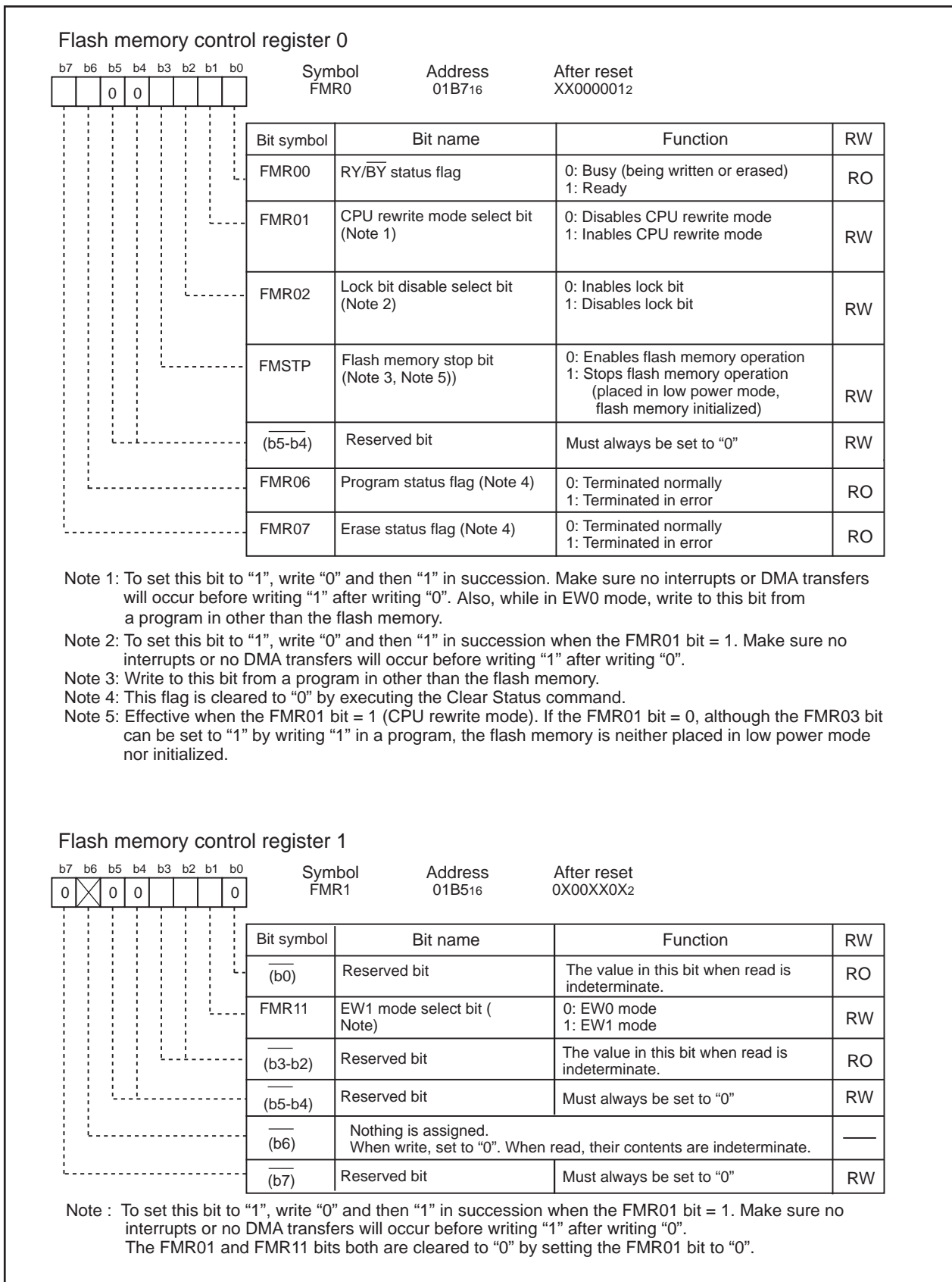
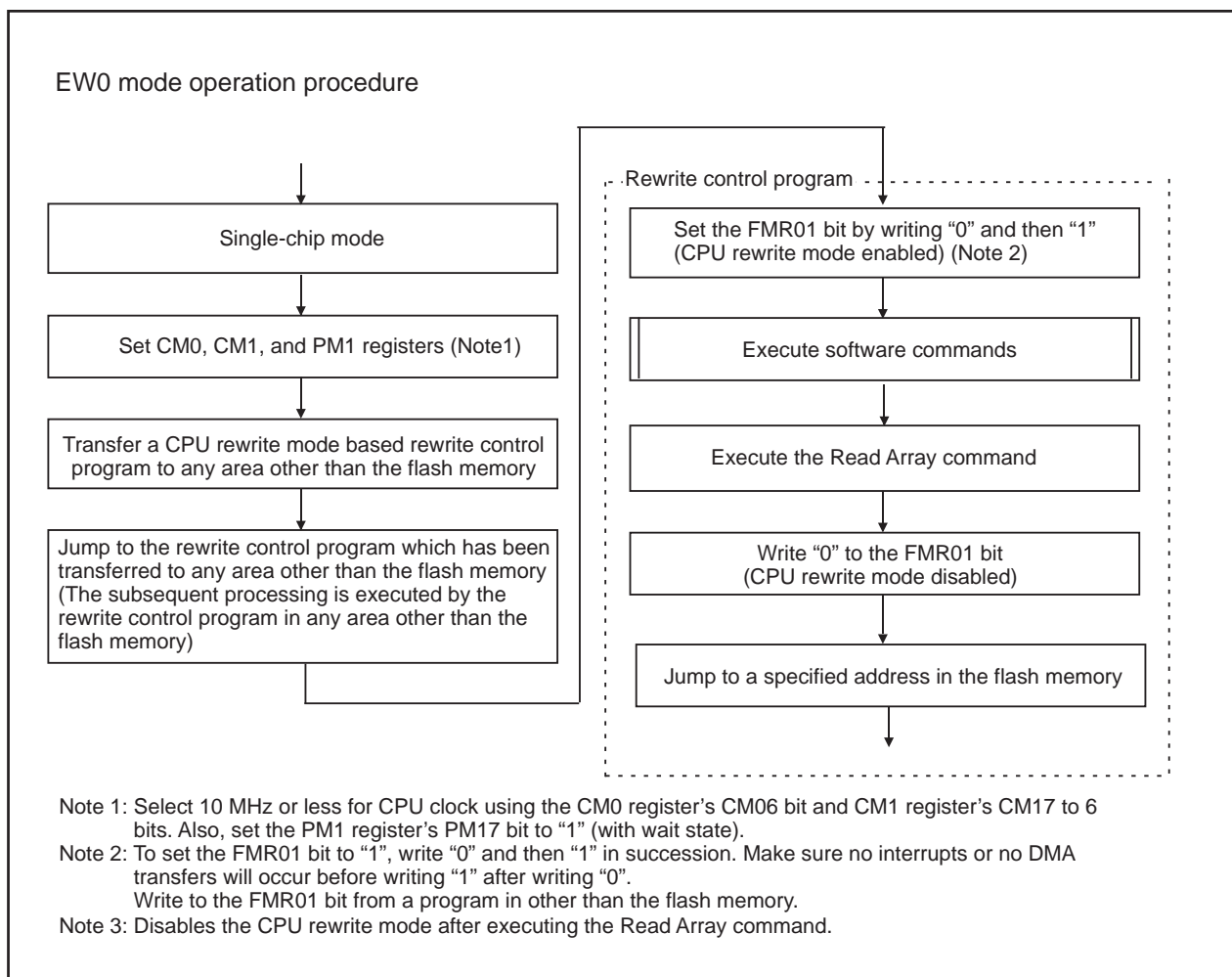


Figure 1.21.1. FIDR Register and FMR0 and FMR1 Registers





**Figure 1.21.2. Setting and Resetting of EW0 Mode**

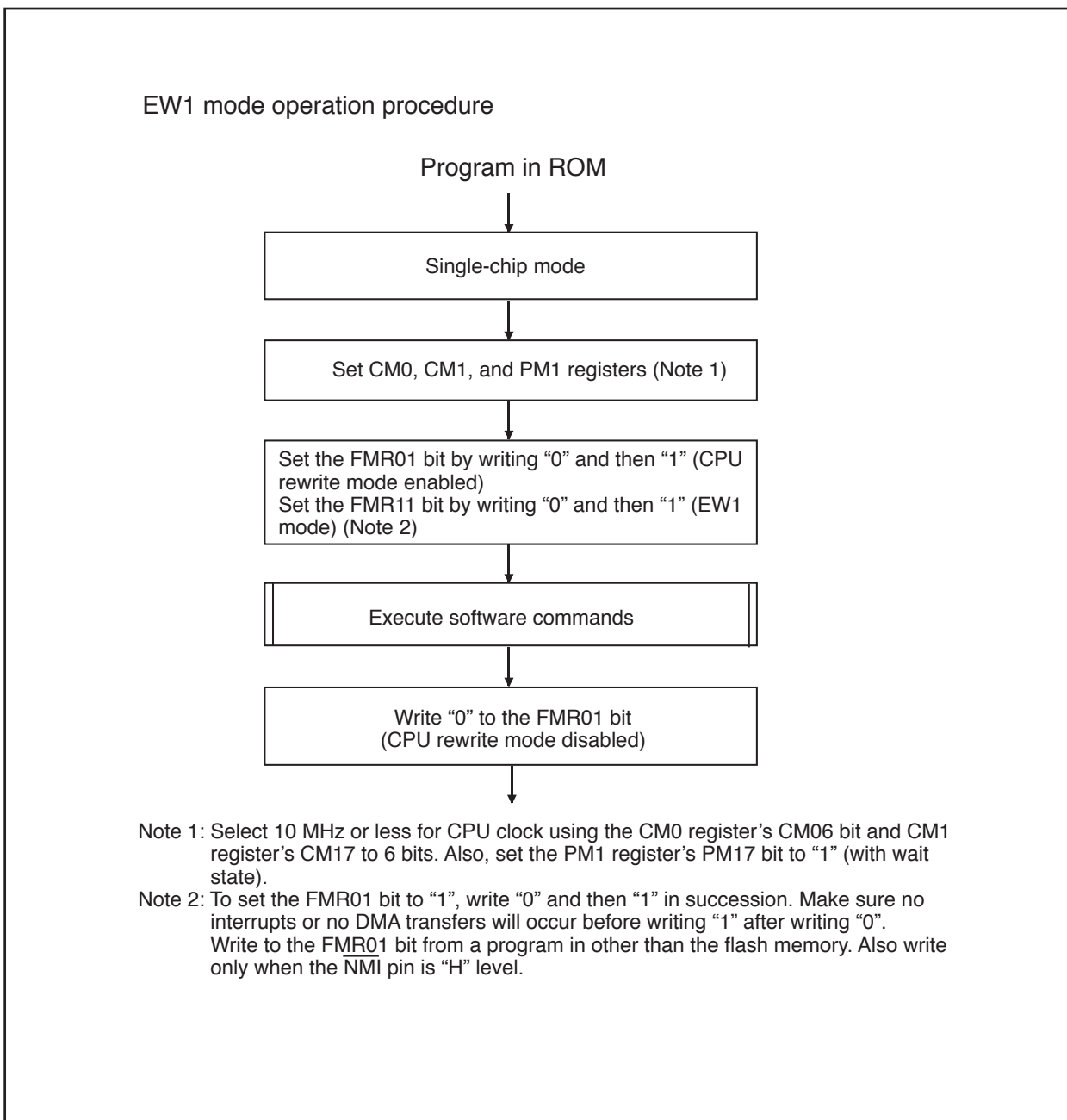


Figure 1.21.3. Setting and Resetting of EW1 Mode

## Precautions on CPU Rewrite Mode

Described below are the precautions to be observed when rewriting the flash memory in CPU rewrite mode.

### (1) Operation Speed

Before entering CPU rewrite mode (EW0 or EW1 mode), select 10 MHz or less for BCLK using the CM06 bit in the CM0 register and the CM17 to CM16 bits in the CM1 register. Also, set the PM17 bit in the PM1 register to “1” (with wait state).

### (2) Instructions to Prevent from Using

The following instructions cannot be used in EW0 mode because the flash memory's internal data is referenced: UND instruction, INTO instruction, JMPS instruction, JSRS instruction, and BRK instruction

### (3) Interrupts

EW0 Mode

- Any interrupt which has a vector in the variable vector table can be used providing that its vector is transferred into the RAM area.
- The  $\overline{\text{NMI}}$  and watchdog timer interrupts can be used because the FMR0 register and FMR1 register are initialized when one of those interrupts occurs. The jump addresses for those interrupt service routines should be set in the fixed vector table.

Because the rewrite operation is halted when a  $\overline{\text{NMI}}$  or watchdog timer interrupt occurs, the rewrite program must be executed again after exiting the interrupt service routine.

- The address match interrupt cannot be used because the flash memory's internal data is referenced.

EW1 Mode

- Make sure that any interrupt which has a vector in the variable vector table or address match interrupt will not be accepted during the auto program or auto erase period.
- Avoid using watchdog timer interrupts.
- The WDT interrupt can be used because the FMR0 register and FMR1 register are initialized when this interrupt occurs. The jump address for the interrupt service routine should be set in the fixed vector table.

Because the rewrite operation is halted when a WDT interrupt occurs, the rewrite program must be executed again after exiting the interrupt service routine.

### (4) How to Access

To set the FMR01, FMR02, or FMR11 bit to “1”, write “0” and then “1” in succession. This is necessary to ensure that no interrupts or DMA transfers will occur before writing “1” after writing “0”.

### (5) Writing in the User ROM Space

EW0 Mode

- If the power supply voltage drops while rewriting any block in which the rewrite control program is stored, a problem may occur that the rewrite control program is not correctly rewritten and, consequently, the flash memory becomes unable to be rewritten thereafter. In this case, standard serial I/O or parallel I/O mode should be used.

EW1 Mode

- Avoid rewriting any block in which the rewrite control program is stored.

**(6) DMA Transfer**

In EW1 mode, make sure that no DMA transfers will occur while the FMR0 register's FMR00 bit = 0 (during the auto program or auto erase period).

**(7) Writing Command and Data**

Write the command code and data at even addresses.

**(8) Wait Mode**

When shifting to wait mode, set the FMR01 bit to "0" (CPU rewrite mode disabled) before executing the WAIT instruction.

**(9) Stop Mode**

When shifting to stop mode, the following settings are required:

- Set the FMR01 bit to "0" (CPU rewrite mode disabled) and disable DMA transfers before setting the CM10 bit to "1" (stop mode).
- Execute the JMP.B instruction subsequent to the instruction which sets the CM10 bit to "1" (stop mode)

```
Example program  BSET      0, CM1      ; Stop mode
                  JMP.B     L1
```

L1:

Program after returning from stop mode

## Software Commands

Software commands are described below. The command code and data must be read and written in 16-bit units, to and from even addresses in the user ROM area. When writing command code, the 8 high-order bits (D<sub>11</sub>–D<sub>8</sub>) are ignored.

**Table 1.21.2. Software Commands**

Command	First bus cycle			Second bus cycle		
	Mode	Address	Data (D <sub>0</sub> to D <sub>7</sub> )	Mode	Address	Data (D <sub>0</sub> to D <sub>7</sub> )
Read array	Write	X	xxFF <sub>16</sub>			
Read status register	Write	X	xx70 <sub>16</sub>	Read	X	SRD
Clear status register	Write	X	xx50 <sub>16</sub>			
Program	Write	WA	xx40 <sub>16</sub>	Write	WA	WD
Block erase	Write	X	xx20 <sub>16</sub>	Write	BA	xxD0 <sub>16</sub>

SRD: Status register data (D<sub>7</sub> to D<sub>0</sub>)

WA: Write address (Make sure the address value specified in the the first bus cycle is the same even address as the write address specified in the second bus cycle.)

WD: Write data (16 bits)

BA: Uppermost block address (even address, however)

X: Any even address in the user ROM area

x: High-order 8 bits of command code (ignored)

### Read Array Command (FF<sub>16</sub>)

This command reads the flash memory.

Writing 'xxFF<sub>16</sub>' in the first bus cycle places the microcomputer in read array mode. Enter the read address in the next or subsequent bus cycles, and the content of the specified address can be read in 16-bit units.

Because the microcomputer remains in read array mode until another command is written, the contents of multiple addresses can be read in succession.

However, when you use Read array command immediately after a program command, please read data in the following procedure.

- (1) FF<sub>16</sub>, FF<sub>16</sub>, FF<sub>16</sub>, and FF<sub>16</sub> are written to 4 arbitrary continuous addresses.
- (2) The head address of (1) is specified in Read array mode.
- (3) (2) is repeated until the read value and FFFF<sub>16</sub> are in agreement.
- (4) The head address +2 of (1) is specified.
- (5) (4) is repeated until the read value and FFFF<sub>16</sub> are in agreement.
- (6) Arbitrary addresses are specified.

### Read Status Register Command (70<sub>16</sub>)

This command reads the status register.

Write 'xx70<sub>16</sub>' in the first bus cycle, and the status register can be read in the second bus cycle. (Refer to "Status Register.") When reading the status register too, specify an even address in the user ROM area.

Do not execute this command in EW1 mode.

### Clear Status Register Command (50<sub>16</sub>)

This command clears the status register to “0”.

Write ‘xx50<sub>16</sub>’ in the first bus cycle, and the FMR06 to FMR07 bits in the FMR0 register and SR4 to SR5 in the status register will be cleared to “0”.

### Program Command (40<sub>16</sub>)

This command writes data to the flash memory in 1 word (2 byte) units.

Write ‘xx40<sub>16</sub>’ in the first bus cycle and write data to the write address in the second bus cycle, and an auto program operation (data program and verify) will start. Make sure the address value specified in the first bus cycle is the same even address as the write address specified in the second bus cycle.

Check the FMR00 bit in the FMR0 register to see if auto programming has finished. The FMR00 bit is “0” during auto programming and set to “1” when auto programming is completed.

Check the FMR06 bit in the FMR0 register after auto programming has finished, and the result of auto programming can be known. (Refer to “Full Status Check.”)

Writing over already programmed addresses is inhibited.

Moreover, when FMR02 bit of FMR0 register is “0” (rewriting is disable), the program command to block 0 and block 1 is not received. Just behind a program command, when you execute commands other than a program command, please make it the address value which was specified by the 2nd bus cycle of a program command and which writes in and specifies the same address as an address by the 1st bus cycle of the following command.

In EW1 mode, do not execute this command on any address at which the rewrite control program is located.

In EW0 mode, the microcomputer goes to read status register mode at the same time auto programming starts, making it possible to read the status register. The status register bit 7 (SR7) is cleared to “0” at the same time auto programming starts, and set back to “1” when auto programming finishes. In this case, the microcomputer remains in read status register mode until a read command is written next. The result of auto programming can be known by reading the status register after auto programming has finished.

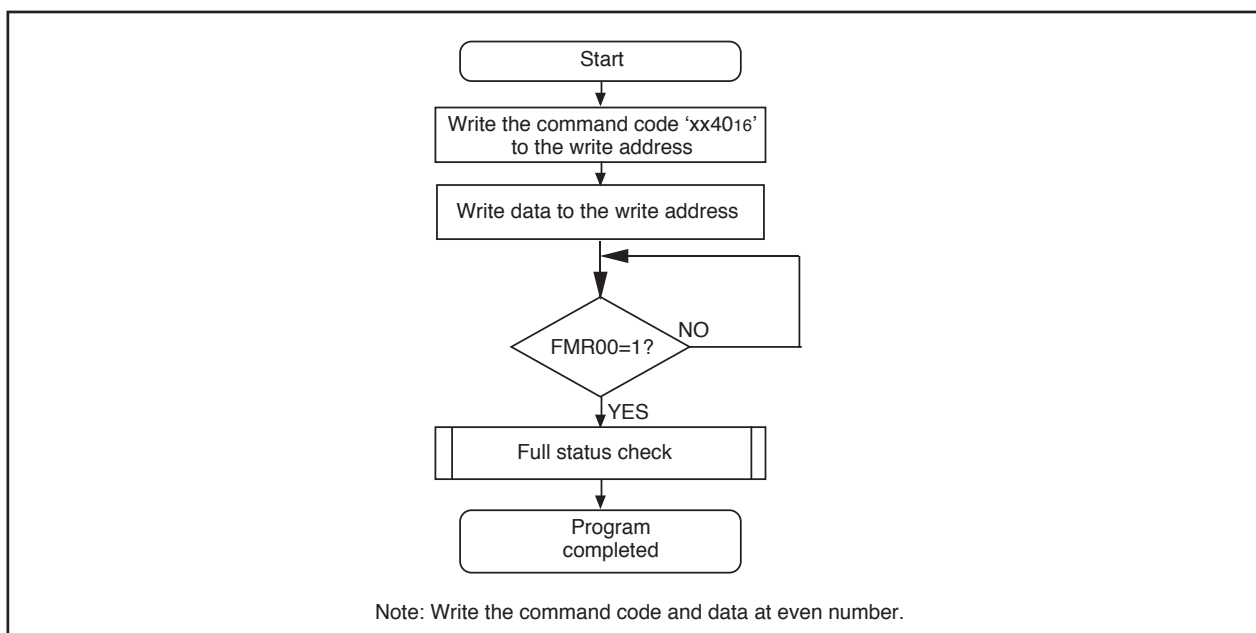


Figure 1.21.4. Program Command

### Block Erase

Write 'xx20<sub>16</sub>' in the first bus cycle and write 'xxD0<sub>16</sub>' to the uppermost address of a block (even address, however) in the second bus cycle, and an auto erase operation (erase and verify) will start. Check the FMR0 register's FMR00 bit to see if auto erasing has finished.

The FMR00 bit is "0" during auto erasing and set to "1" when auto erasing is completed.

Check the FMR0 register's FMR07 bit after auto erasing has finished, and the result of auto erasing can be known. (Refer to "Full Status Check.")

Moreover, when FMR02 bit of FMR0 register is "0" (rewriting is disable), the block erase command to block 0 and block 1 is not received.

Figure 1.21.5 shows an example of a block erase flowchart.

Each block can be protected against erasing by a lock bit. (Refer to "Data Protect Function.")

Writing over already programmed addresses is inhibited.

In EW1 mode, do not execute this command on any address at which the rewrite control program is located.

In EW0 mode, the microcomputer goes to read status register mode at the same time auto erasing starts, making it possible to read the status register. The status register bit 7 (SR7) is cleared to "0" at the same time auto erasing starts, and set back to "1" when auto erasing finishes. In this case, the microcomputer remains in read status register mode until the Read Array or Read Lock Bit Status command is written next.

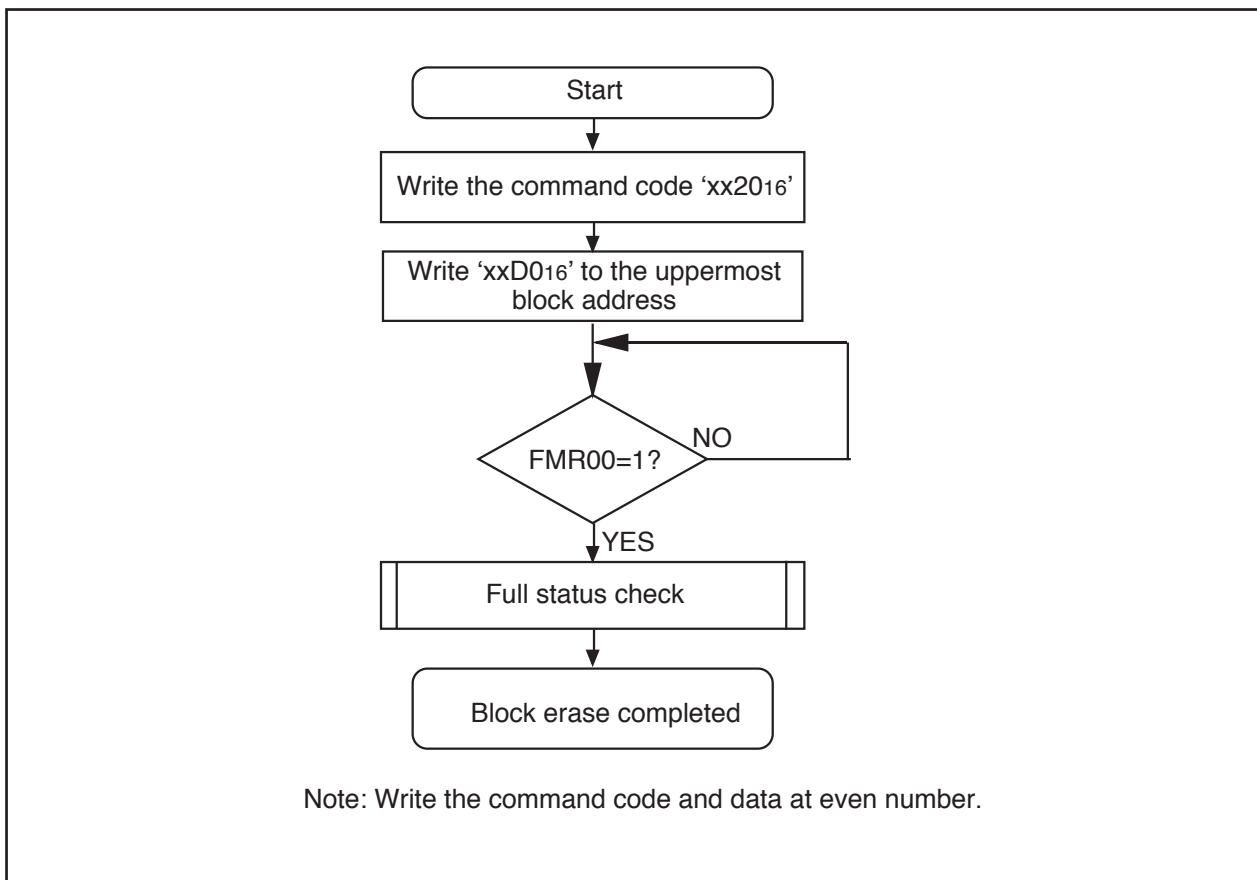


Figure 1.21.5. Block Erase Command

## Status Register

The status register indicates the operating status of the flash memory and whether an erase or programming operation terminated normally or in error. The status of the status register can be known by reading the FMR0 register's FMR00, FMR06, and FMR07 bits.

Table 1.21.3 shows the status register.

In EW0 mode, the status register can be read in the following cases:

- (1) When a given even address in the user ROM area is read after writing the Read Status Register command
- (2) When a given even address in the user ROM area is read after executing the Program, Block Erase, Erase All Unlocked Block, or Lock Bit Program command but before executing the Read Array command.

### Sequencer Status (SR7 and FMR00 Bits )

The sequence status indicates the operating status of the flash memory. SR7 = 0 (busy) during auto programming and auto erase is set to "1" (ready) at the same time the operation finishes.

### Erase Status (SR5 and FMR07 Bits)

Refer to "Full Status Check."

### Program Status (SR4 and FMR06 Bits)

Refer to "Full Status Check."

**Table 1.21.3. Status Register**

- D0 to D7: Indicates the data bus which is read out when the Read Status Register command is executed.

Status register bit	FMR0 register bit	Status name	Contents		Value after reset
			"0"	"1"	
SR7 (D7)	FMR00	Sequencer status	Busy	Ready	1
SR6 (D6)	—	Reserved	-	-	—
SR5 (D5)	FMR07	Erase status	Terminated normally	Terminated in error	0
SR4 (D4)	FMR06	Program status	Terminated normally	Terminated in error	0
SR3 (D3)	—	Reserved	-	-	—
SR2 (D2)	—	Reserved	-	-	—
SR1 (D1)	—	Reserved	-	-	—
SR0 (D0)	—	Reserved	-	-	—

- The FMR07 bit (SR5) and FMR06 bit (SR4) are cleared to "0" by executing the Clear Status Register command.
- When the FMR07 bit (SR5) or FMR06 bit (SR4) = 1, the Program and Block Erase are not accepted.



## Full Status Check

When an error occurs, the FMR0 register's FMR06 to FMR07 bits are set to "1", indicating occurrence of each specific error. Therefore, execution results can be verified by checking these status bits (full status check). Table 1.21.4 lists errors and FMR0 register status. Figure 1.21.6 shows a full status check flowchart and the action to be taken when each error occurs.

**Table 1.21.4. Errors and FMR0 Register Status**

FMR00 register (SRD register) status		Error	Error occurrence condition
FMR07 (SR5)	FMR06 (SR4)		
1	1	Command sequence error	<ul style="list-style-type: none"> <li>• When any commands are not written correctly</li> <li>• A value other than 'xxD016' or 'xxFF16' is written in the second bus cycle of the block erase command (Note 1)</li> <li>• When the block erase command is executed on protected blocks</li> <li>• When the program command is executed on protected blocks</li> </ul>
1	0	Erase error	<ul style="list-style-type: none"> <li>• When the block erase command is executed on unprotected blocks but the blocks are not automatically erased correctly</li> </ul>
0	1	Program error	<ul style="list-style-type: none"> <li>• When the program command is executed on unprotected blocks but the blocks are not automatically programmed correctly.</li> </ul>

Note 1: The flash memory enters read array mode by writing command code 'xxFF16' in the second bus cycle of these commands. The command code written in the first bus cycle becomes invalid.

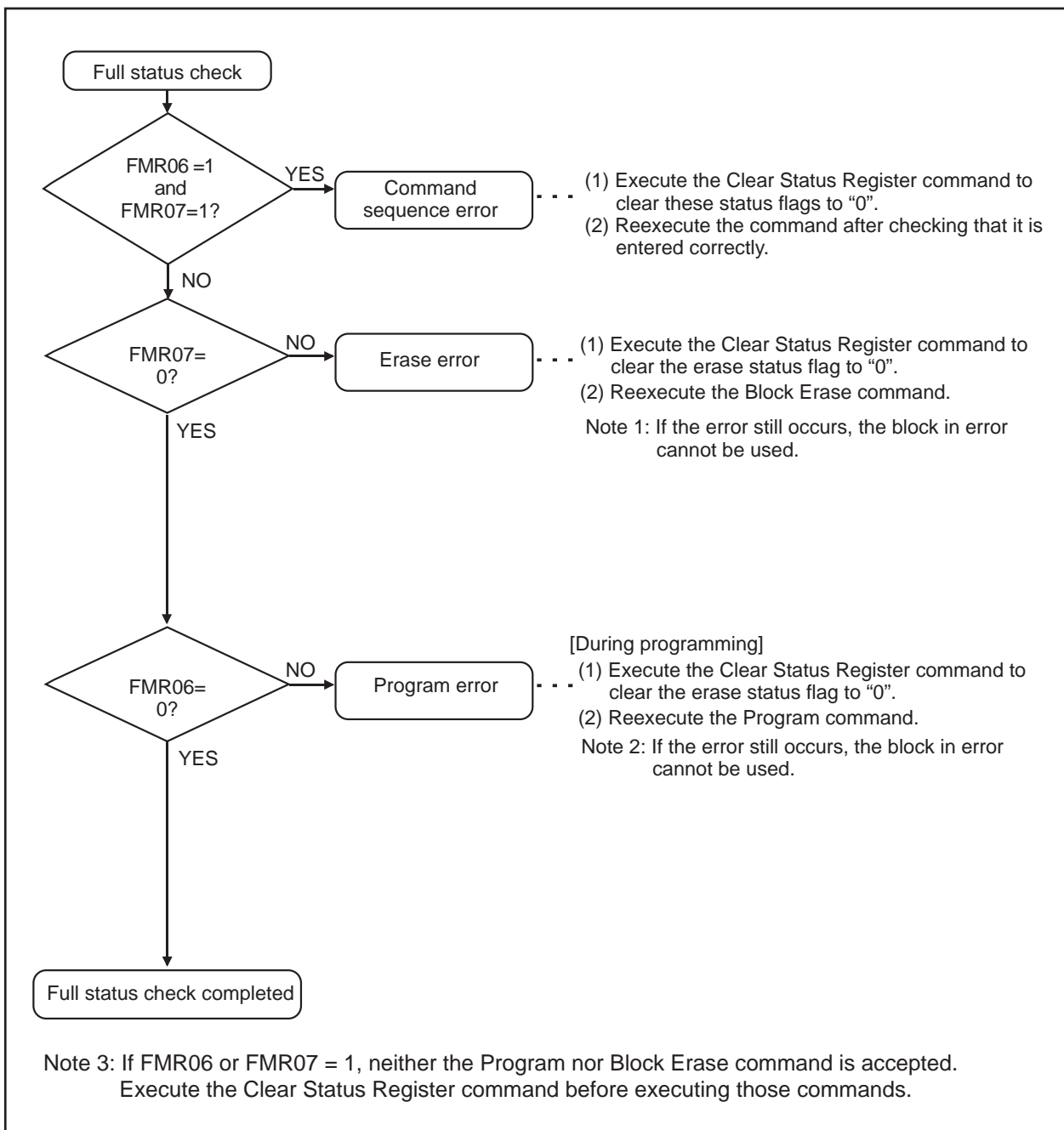


Figure 1.21.6. Full Status Check and Handling Procedure for Each Error

## Standard Serial I/O Mode

The standard serial I/O mode inputs and outputs the software commands, addresses and data needed to operate (read, program, erase, etc.) the internal flash memory using the serial I/O port UART1. The serial I/O mode transfers the data serially in 8-bit units.

In the standard serial I/O mode the CPU executes a control program for flash memory rewrite (using the CPU's rewrite mode), rewrite data input and so forth. It is started when both the P15 (CE) pin and the CNVss pin are in "H" level after the reset is released. (In normal operation mode, set CNVss pin to "L" level.) The main clock f1 is 5.12 MHz in this mode.

This control program is written in the boot ROM area when the product is shipped.

There are actually two standard serial I/O modes: mode 1, which is clock synchronized, and mode 2, which is asynchronous. Standard serial I/O switches between mode 1 (clock synchronous) and mode 2 (clock asynchronous) depending on the level of CLK1 pin when the reset is released.

To use standard serial I/O mode 1 (clock synchronous), set the CLK1 pin to "H" level and release the reset. The operation uses the four UART1 pins CLK1, RxD1, TxD1 and RTS1 (BUSY). The CLK1 pin is the transfer clock input pin through which an external transfer clock is input. The TxD1 pin is for CMOS output. The RTS1 (BUSY) pin outputs an "L" level when ready for reception and an "H" level when reception starts. In mode 1, be sure the TxD1 (P67) pin is at high before reset being deasserted.

To use standard serial I/O mode 2 (clock asynchronous), set the CLK1 pin to "L" level and release the reset. The operation uses the two UART1 pins RxD1 and TxD1.

In standard serial input/output mode, the user ROM area can be rewritten while the microcomputer is mounted on-board by using a serial programmer suitable for the M16C/62P group. For more information about serial programmers, contact the manufacturer of your serial programmer. For details on how to use, refer to the user's manual included with your serial programmer.

Table 1.22.1 lists pin functions (flash memory standard serial input/output mode). Figures 1.22.1 to 1.22.2 show pin connections for serial input/output mode.

## ID Code Check Function

This function determines whether the ID codes sent from the serial programmer and those written in the flash memory match. (Refer to the description of the functions to inhibit rewriting flash memory version.)

**Table 1.22.1. Functional explanation of pin (flash memory standard serial I/O mode)**

Pin	Signal name	I/O	Description
Vcc,Vss	Power supply input		Apply the voltage guaranteed for Program and Erase to Vcc pin. Apply 0 V to Vss pin.
CNVss	CNVss	I	Connect to Vcc pin.
RESET	Reset input	I	Reset input pin. While $\overline{\text{RESET}}$ pin is "L" level, input a 20 cycle or longer clock to XIN pin.
XIN	Clock input	I	Connect a ceramic resonator or crystal oscillator between XIN and XOUT pins. To input an externally generated clock, input it to XIN pin and open XOUT pin.
XOUT	Clock output	O	
VDCCN			Input "H" level signal.
VCCA, VSSA	Analog power supply input	I	Please connect AVcc to Vcc and connect AVss to Vss.
VREF	Standard voltage input	I	Input the standard voltage of a preamplifier and an operational amplifier.
P60 to P63	Input port P6	I	Input "H" or "L" level signal or open.
P64/ $\overline{\text{RTS1}}$	BUSY output	O	Standard serial I/O mode 1: BUSY signal output pin Standard serial I/O mode 2: Monitors the boot program operation check signal output pin.
P65/CLK1	SCLK input	I	Standard serial I/O mode 1: Serial clock input pin Standard serial I/O mode 2: Input "L."
P66/RxD1	RxD input	I	Serial data input pin.
P67/TxD1	TxD output	O	Serial data output pin. (Note 1)
P70, P71, P73, P74, P76	Input port P7	I	Input "H" or "L" level signal or open.
P80, P81, P83, P84, P85	Input port P8	I	Input "H" or "L" level signal or open.
P15	$\overline{\text{CE}}$ input	I	Input "H" level signal.
P90 to P92	Input port P11	I	Input "H" or "L" level signal or open.

Note 1: When using standard serial input/output mode 1, the TxD pin must be held high while the  $\overline{\text{RESET}}$  pin is pulled low. Therefore, connect this pin to Vcc1 via a resistor. Because this pin is directed for data output after reset, adjust the pull-up resistance value in the system so that data transfers will not be affected.

### Example of Circuit Application in the Standard Serial I/O Mode

Figure 1.22.1 and 1.22.2 show example of circuit application in standard serial I/O mode 1 and mode 2, respectively. Refer to the user's manual for serial programmer to handle pins controlled by a serial programmer.

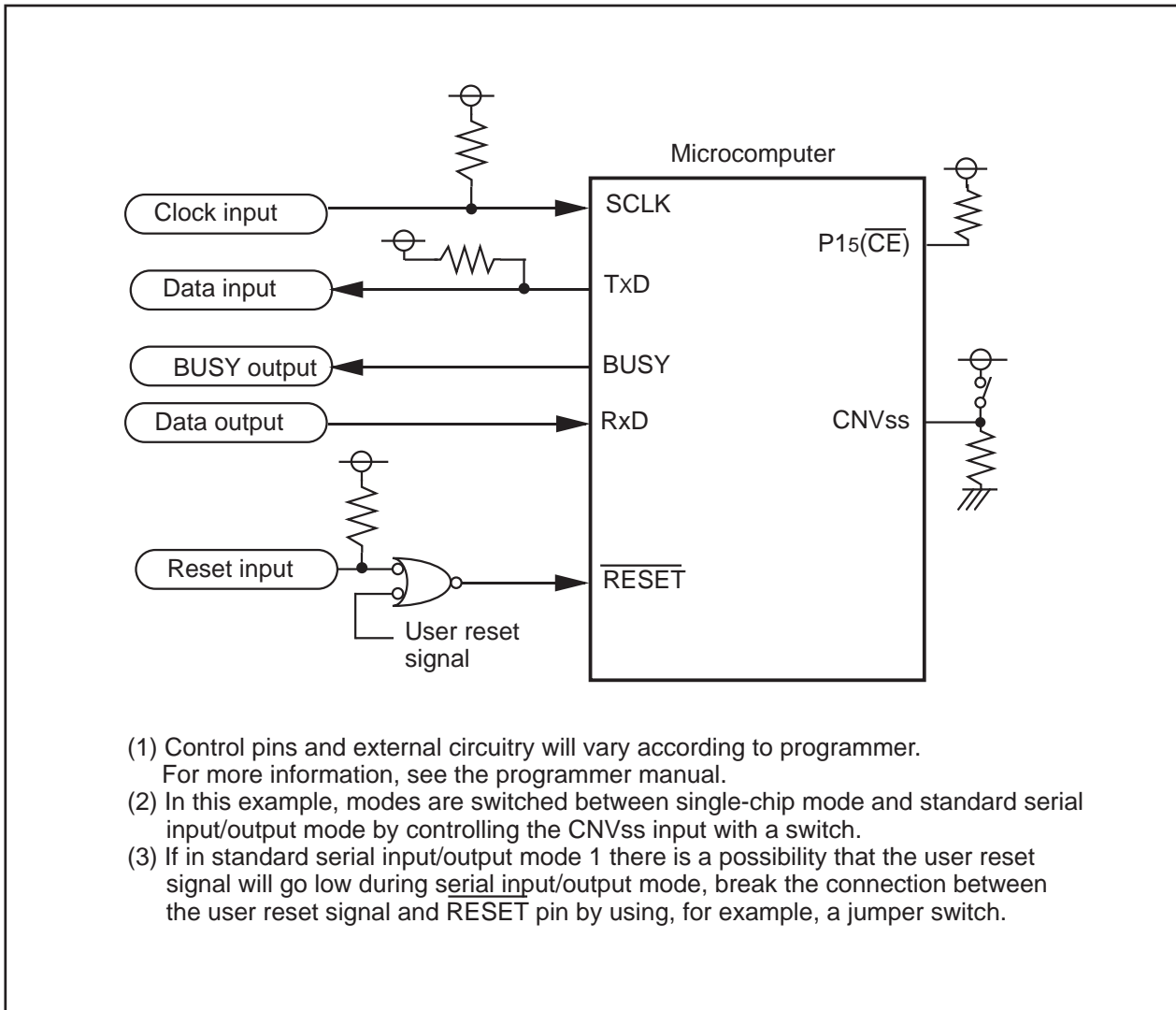


Figure 1.22.1. Circuit Application in Standard Serial I/O Mode 1

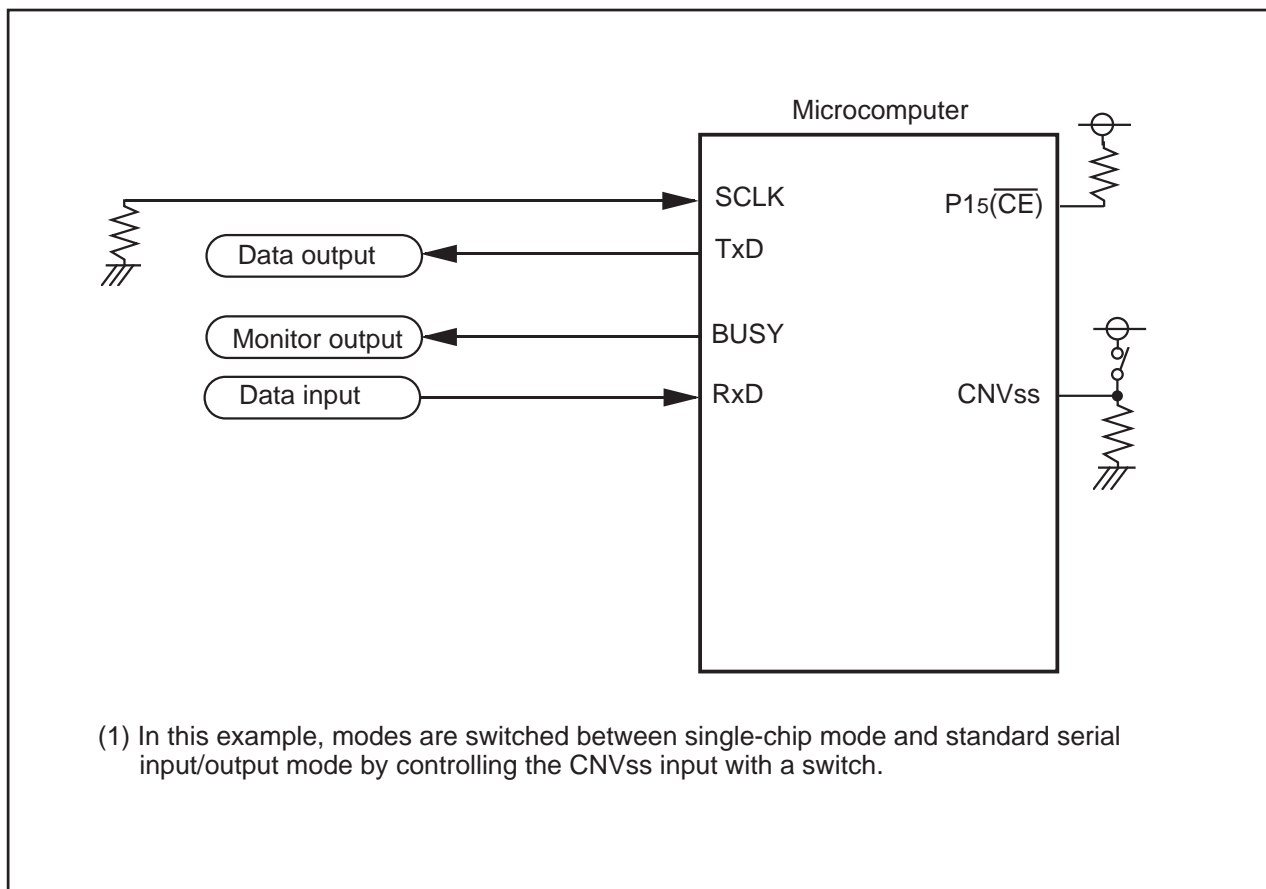


Figure 1.22.2. Circuit Application in Standard Serial I/o Mode 2

## ROM Code Protect Function

The ROM code protect function inhibits the flash memory from being read or rewritten. (Refer to the description of the functions to inhibit rewriting flash memory version.)

## IT800AFE (Analog Front End)

### 1. The block diagram of analog front end

Analog front end (AFE) part is the circuit located between M16C/6S and a power line, and M16C/6S build in DAC, preamplifier, and ADC. There are the following two signal circuits in AFE.

(1) Transmitting circuit: It drives with the signal from the logic of M16C/6S inside, and this consists of DAC with a differential output, Output filter for spectrum adjustment, Differential line driver amplifier and line coupling circuit which drive power line.

(2) Receiving circuit: This consists of Line coupling circuit, Differential preamplifier, Input filter group and ADC. Line coupling circuit is common to both transmission and reception.

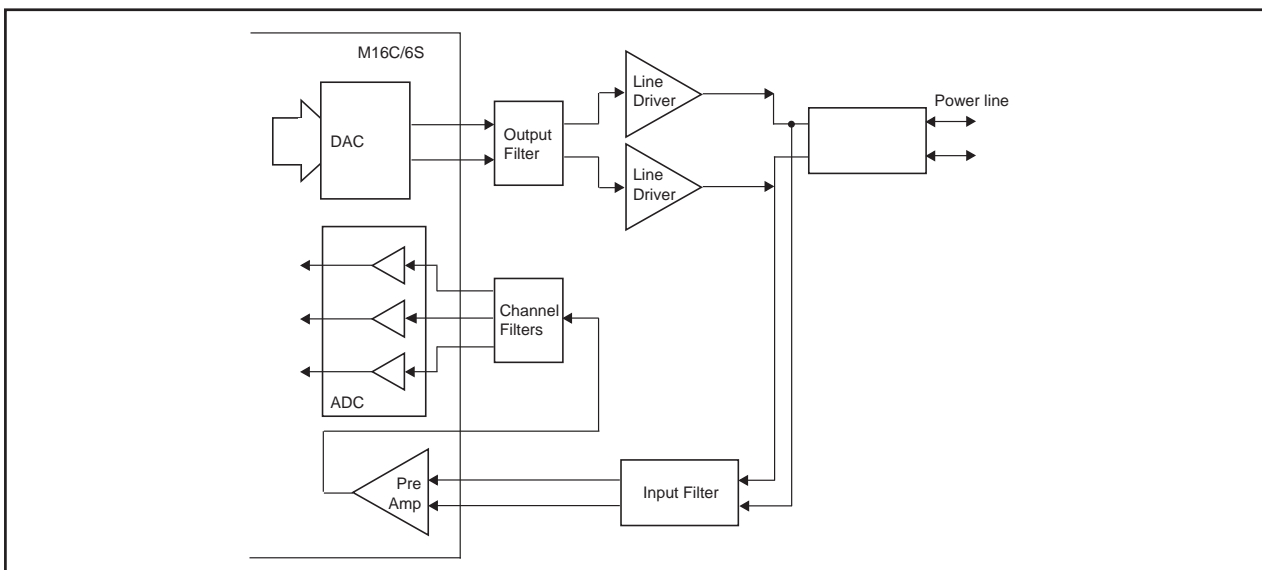


Figure 1.23.1 Analog Front End circuit block diagram

#### (1) Transmitting path

The characteristics required for transmitting path are as follows.

- (a) A suitable output signal level is obtained on specific load.
- (b) Frequency characteristic is flat in signal bandwidth.
- (c) The signal level outside a zone: It is less than each regulation.

This system assumes operation in the following three fundamental signal zones.

- (a) The U.S., Japan: 120k to 400kHz
- (b) Europe in door: 95k to 125kHz (CENELEC B Band)
- (c) Europe out door: 20k to 80kHz (CENELEC A Band)

It is necessary to perform the change of a signal zone by change of adjustment of change of the configuration of IT800, the analog filter to a signal zone, or output electric power, and the output impedance of the line driver stage (notes).

Note. Please refer to the following standard and style about the conditions outside a signal level or a zone.

- (a) The U.S.: FCC standard, part 15
- (b) Europe: CENELEC standard, EN 50065-1
- (c) Japan: ARIB STD-T84



## About DAC (Digital to Analog Converter)

DAC of built-in in M16C/6S is current output type 10 bit DAC. The standard level of output current can be set up by the external resistance linked to a built-in standard power supply. A DAC circuit and a load circuit are shown in Figure 1.23.2. The typical characteristic is shown in Table 1.23.1.

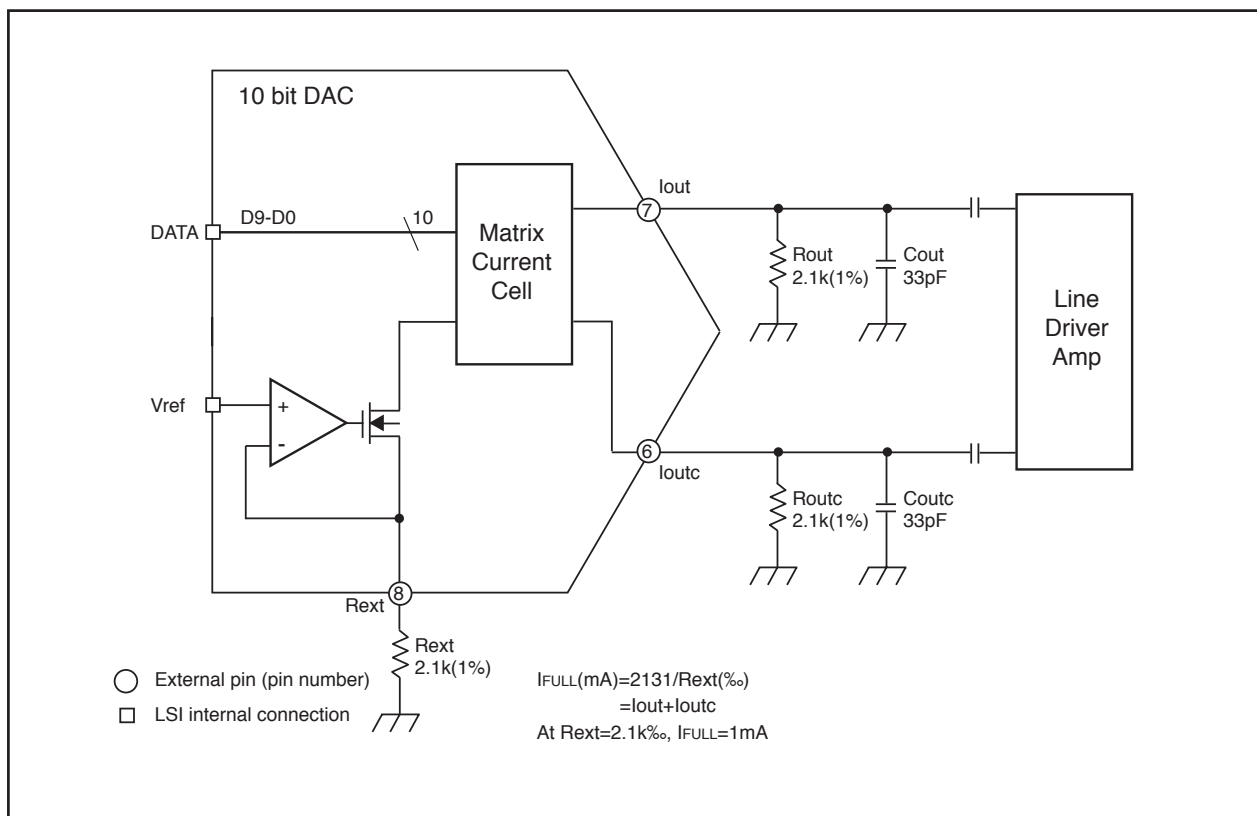


Figure 1.23.2 DAC (Digital to Analog Converter)

Table 1.23.1 DAC Electrical Characteristics (V<sub>dd</sub>=3.3±0.3V, T<sub>a</sub> = -20 to 85°C/-40 to 85°C/-40 to 105°C unless otherwise specified)

Parameter	Symbol	Condition	Electrical Characteristics			Unit
			Min	Typ	Max	
Standard voltage	Vref	Rext=2.1k		0.3		V
Standard current	Iref	Rext=2.1k		0.14		mA
External standard register	Rext			2.1		k
DAC output register	Rout/Routc	Rext=2.1k			2.0	k
Full-scale current	I <sub>FULL</sub>	Rext=2.1k	0.9	1	1.1	mA
Maximum output voltage	Voutmax	Rout/Routc=2.1k			V <sub>dd</sub> -1	V

In the circumference of a DAC circuit, it is cautious of the following point and a constant is set up.

(a) Full-scale output current of DAC

Although the current by which DAC is outputted to a pin ( $I_{out}/I_{outc}$ ) according to the change width of an internal bit changes, total is fixed and serves as full-scale output current (IFULL). This IFULL can be set up by the following formula.

$$IFULL = 2131 / R_{ext} (\text{mA})$$

The unit of  $R_{ext}$  is  $\Omega$

Since about 1mA of an IFULL value is the optimal value,  $R_{ext}$  is 2.1k $\Omega$ .

DAC is total and  $I_{out}/I_{outc}$  changes between 0 to IFULL at 1024 steps by the full range for a change width of 10 bits.

The value of  $I_{out}/I_{outc}$  is expressed with the following formula if the step value which converted the 10-bit binary input into decimal is set to  $d$ . ( $d=0$  to 1023)

$$I_{out} = IFULL / 1023 * d$$

$$I_{outc} = IFULL / 1023 * (1023 - d)$$

(b) About DAC load resistance

Resistance ( $R_{out}/R_{outc}$ ) is connected to a DAC output pin. In this resistance, signal voltage occurs by the current ( $I_{out}/I_{outc}$ ) which flows, respectively. In order to maintain the linearity of  $I_{out}/I_{outc}$ , it is necessary to set up  $R_{out}/R_{outc}$  so that this potential may not exceed maximum 2V. Since  $I_{out}/I_{outc}$  is mostly set to one half of IFULL(s) at the time of a non-signal (balanced state), the signal voltage ( $V_{out}/V_{outc}$ ) of a DAC output pin is set to below:

$$V_{out} = R_{out} \times I_{out}$$

$$V_{outc} = R_{outc} \times I_{outc}$$

Since the recommendation value of  $V_{out}/V_{outc}$  at the time of a non-signal is about 1V, in  $R_{ext}=2.1\text{k}\Omega$ ,  $R_{out}/R_{outc}$  serves as 2.1k $\Omega$ .

The capacitor ( $C_{out}/C_{outc}$ ) for output filters cuts the high frequency ingredient of DAC, and sets it up on balance with necessary zone frequency.

## (2) Receiving circuit

### (a) Preamplifier

A preamplifier circuit consists of two CMOS operational amplifiers as shown in Figure 1.23.3, the first opamp is connected as gain stage with gain of 20dB and the second opamp is connected as a voltage follower for driving of external filter.

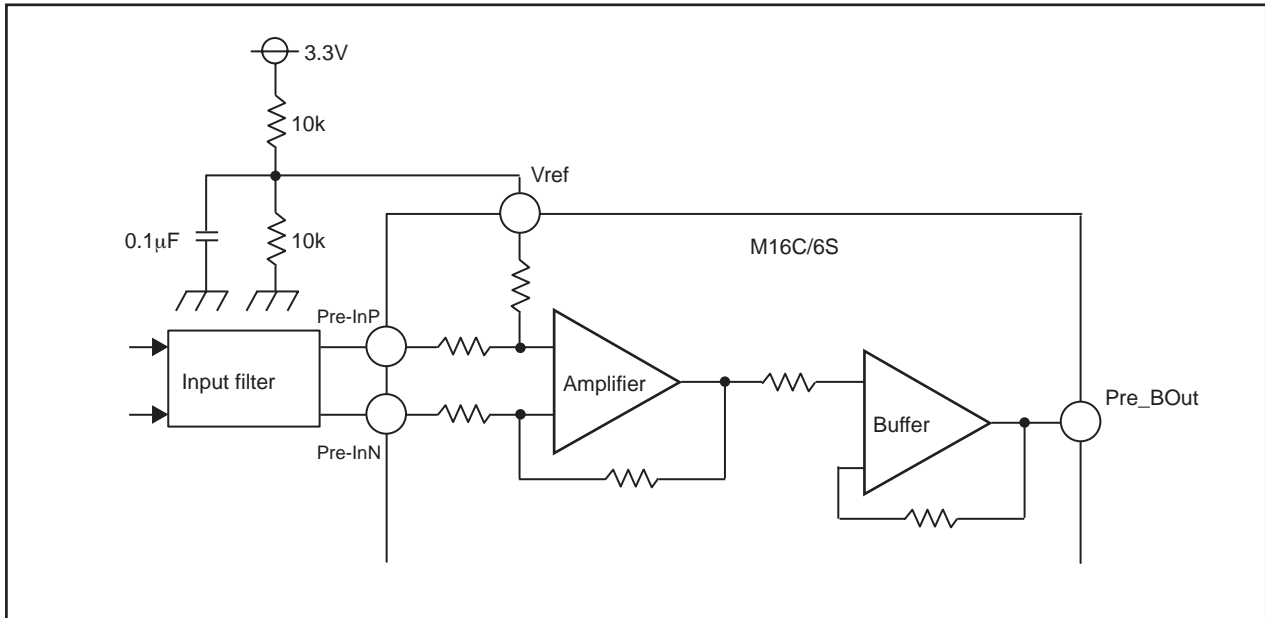


Figure 1.23.3 Consists of Preamp circuit

Table 1.23.2 Preamp Electrical Characteristics (Ta = 25°C unless otherwise specified)

Parameter	Symbol	Condition	Electrical Characteristics			Unit
			Min	Typ	Max	
Standard voltage	Vdd		3.0	3.3	3.6	V
Input off-set voltage	Vof				15	mV
Open loop gain	Gvo	No-load	70			dB
Gain band width	BW	Gvo=0dB	5			MHz
Common mode input range	CMIR	DC to 1MHz	0.7		Vdd-0.7	V
Common mode rejection ratio	CMRR	DC to 1MHz	40			dB
Power supply rejection ratio	PSRR	DC to 1MHz	30			dB

## (b)ADC (Analog to Digital Converter)

The output of the channel filter (maximum of three) connected outside is connected to 1 bit ADC which consists of an operational amplifier and a comparator.

M16C/6S build in ADC of three equivalent performances. The circuit of one ADC has composition shown in the following figure.

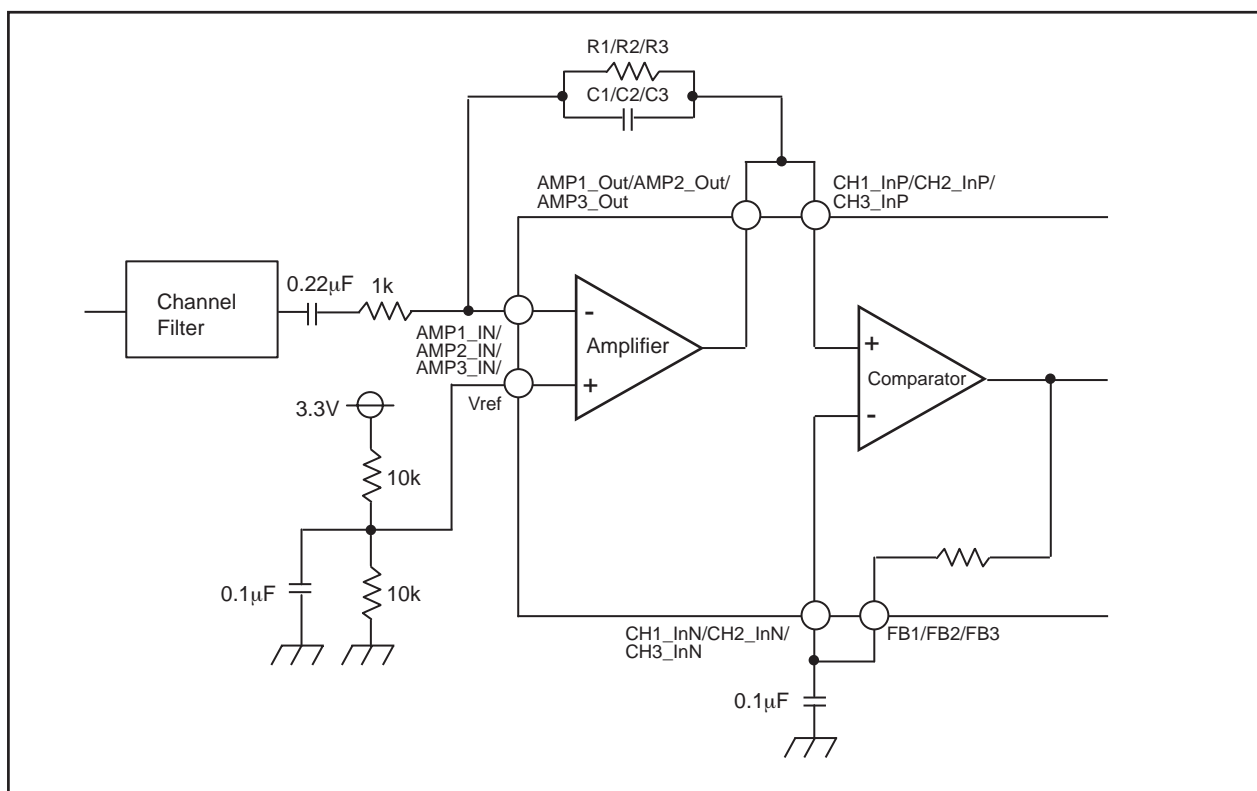


Figure 1.23.4 ADC block diagram

Three ADC use all three by the signal zone to be used. The constant of a filter is set up as shown in the following table.

Table 1.23.3 The example of an ADC part constant setting

Parts	Application characteristic	C1	C2	C3	Unit
Capacitor	FCC/ARIB	33	22	10	pF
	CECLEC-B	33	—	—	pF
Parts	Application characteristic	R1	R2	R3	Unit
Register	FCC/ARIB	10	10	12	kΩ
	CECLEC-B	10	—	—	kΩ

## Usage Notes

### Register Setting

Immediate values should be set in the registers containing write-only bits. When establishing a new value by modifying a previous value, write the previous value into RAM as well as the register. Change the contents of the RAM and then transfer the new value to the register.

## Power Control

When entering wait mode, insert a JMP.B instruction before a WAIT instruction. Do not execute any instructions which can generate a write to RAM between the JMP.B and WAIT instructions. Disable the DMA transfers, if a DMA transfer may occur between the JMP.B and WAIT instructions. After the WAIT instruction, insert at least 4 NOP instructions. When entering wait mode, the instruction queue reads ahead the instructions following WAIT, and depending on timing, some of these may execute before the MCU enters wait mode.

Program example when entering wait mode

```

Program Example:  JMP.B    L1    ; Insert JMP.B instruction before WAIT instruction
                  L1:
                  FSET    I    ;
                  WAIT    ; Enter wait mode
                  NOP     ; More than 4 NOP instructions
                  NOP
                  NOP
                  NOP

```

When entering stop mode, insert a JMP.B instruction immediately after executing an instruction which sets the CM10 bit in the CM1 register to 1, and then insert at least 4 NOP instructions. When entering stop mode, the instruction queue reads ahead the instructions following the instruction which sets the CM10 bit to 1 (all clock stops), and, some of these may execute before the MCU enters stop mode or before the interrupt routine for returning from stop mode.

Program example when entering stop mode

```

Program Example:  FSET    I
                  BSET    CM10 ; Enter stop mode
                  JMP.B   L2    ; Insert JMP.B instruction
                  L2:
                  NOP     ; More than 4 NOP instructions
                  NOP
                  NOP
                  NOP

```

Stop mode and wait mode is cancelled by a hardware reset or an interrupt. If an interrupt is to be used to cancel stop mode and wait mode that interrupt must first have been enabled, and the priority level of the interrupt which is not used to cancel must have been changed to 0.

## Changing the Interrupt Generate Factor

If the interrupt generate factor is changed, the IR bit in the interrupt control register for the changed interrupt may inadvertently be set to 1 (interrupt requested). If you changed the interrupt generate factor for an interrupt that needs to be used, be sure to clear the IR bit for that interrupt to 0 (interrupt not requested).

“Changing the interrupt generate factor” referred to here means any act of changing the source, polarity or timing of the interrupt assigned to each software interrupt number. Therefore, if a mode change of any peripheral function involves changing the generate factor, polarity or timing of an interrupt, be sure to clear the IR bit for that interrupt to 0 (interrupt not requested) after making such changes. Refer to the description of each peripheral function for details about the interrupts from peripheral functions.

Figure 1.24.1 shows the procedure for changing the interrupt generate factor.

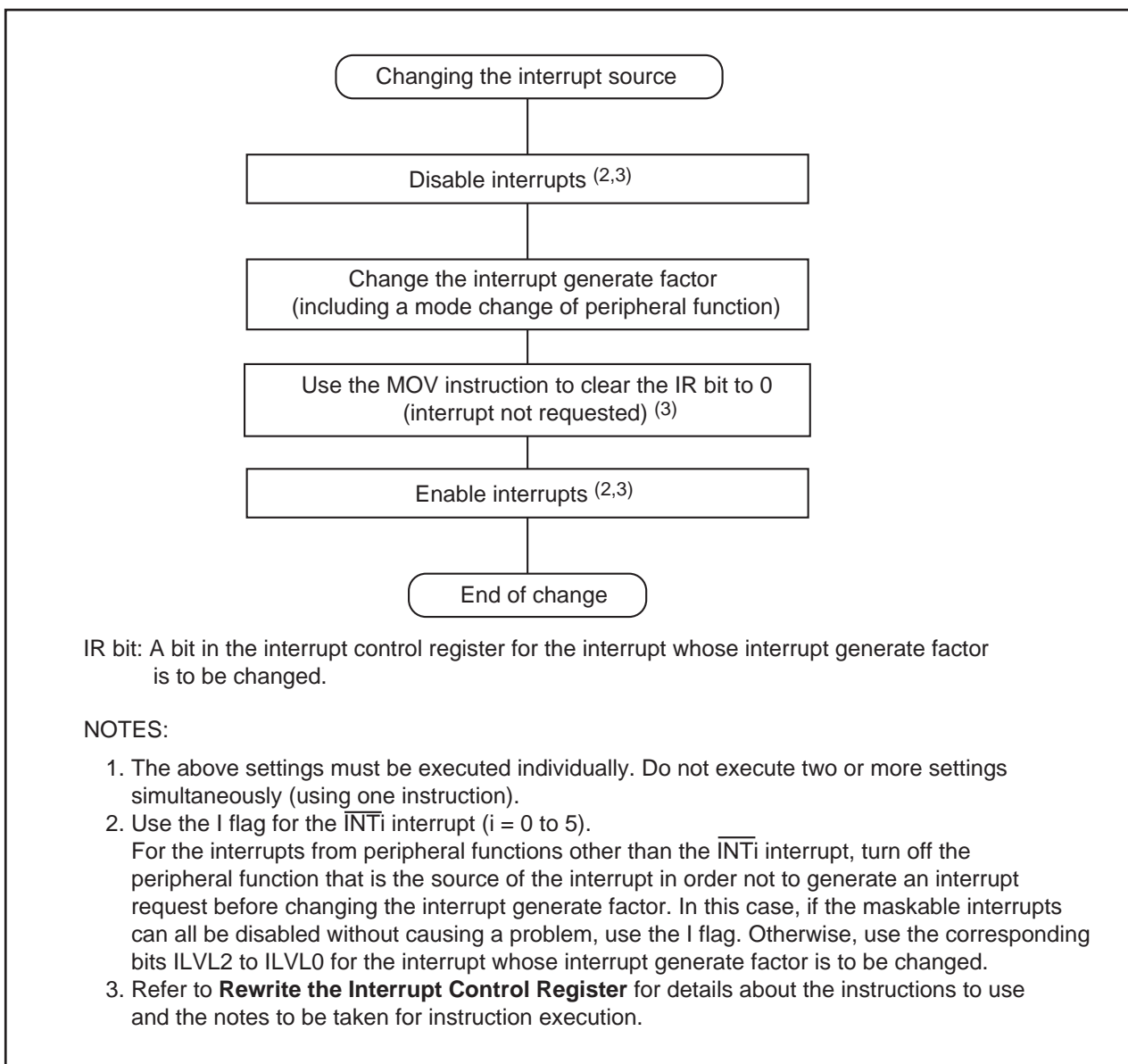


Figure 1.24.1 Procedure for Changing the Interrupt Generate Factor

## Watchdog Timer Interrupt

Initialize the watchdog timer after the watchdog timer interrupt occurs.

## DMAC

### Write to DMAE Bit in DMiCON Register (i = 0 to 1)

When both of the conditions below are met, follow the steps below.

(a) Conditions

- The DMAE bit is set to 1 again while it remains set (DMAi is in an active state).
- A DMA request may occur simultaneously when the DMAE bit is being written.

(b) Procedure

- (1) Write 1 to the DMAE bit and DMAS bit in DMiCON register simultaneously <sup>(1)</sup>.
- (2) Make sure that the DMAi is in an initial state <sup>(2)</sup> in a program.

If the DMAi is not in an initial state, the above steps should be repeated.

#### NOTES:

1. The DMAS bit remains unchanged even if 1 is written. However, if 0 is written to this bit, it is set to 0 (DMA not requested). In order to prevent the DMAS bit from being modified to 0, 1 should be written to the DMAS bit when 1 is written to the DMAE bit. In this way the state of the DMAS bit immediately before being written can be maintained.  
Similarly, when writing to the DMAE bit with a read-modify-write instruction, 1 should be written to the DMAS bit in order to maintain a DMA request which is generated during execution.
2. Read the TCRi register to verify whether the DMAi is in an initial state. If the read value is equal to a value which was written to the TCRi register before DMA transfer start, the DMAi is in an initial state. (If a DMA request occurs after writing to the DMAE bit, the value written to the TCRi register is 1.) If the read value is a value in the middle of transfer, the DMAi is not in an initial state.



## Timers

### Timer A

#### Timer A (Timer Mode)

The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TAI<sub>MR</sub> (i = 0 to 4) register and the TAI register before setting the TAI<sub>S</sub> bit in the TABSR register to 1 (count starts). Always make sure the TAI<sub>MR</sub> register is modified while the TAI<sub>S</sub> bit remains 0 (count stops) regardless whether after reset or not.

While counting is in progress, the counter value can be read out at any time by reading the TAI register. However, if the TAI register is read at the same time the counter is reloaded, the read value is always FFFF<sub>16</sub>. If the TAI register is read after setting a value in it, but before the counter starts counting, the read value is the one that has been set in the register.

#### Timer A (Event Counter Mode)

The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TAI<sub>MR</sub> (i = 0 to 4) register, the TAI register, the UDF register, bits TAZIE, TA0TGL, and TA0TGH in the ONSF register and the TRGSR register before setting the TAI<sub>S</sub> bit in the TABSR register to 1 (count starts). Always make sure bits TAZIE, TA0TGL, and TA0TGH in the TAI<sub>MR</sub> register, the UDF register, the ONSF register, and the TRGSR register are modified while the TAI<sub>S</sub> bit remains 0 (count stops) regardless whether after reset or not.

While counting is in progress, the counter value can be read out at any time by reading the TAI register. However, if the TAI register is read at the same time the counter is reloaded, the read value is always FFFF<sub>16</sub> when the timer counter underflows and 0000<sub>16</sub> when the timer counter overflows. If the TAI register is read after setting a value in it, but before the counter starts counting, the read value is the one that has been set in the register.

## Timer A (One-shot Timer Mode)

The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TAIMR (i = 0 to 4) register, the TAI register, bits TA0TGL and TA0TGH in the ONSF register and the TRGSR register before setting the TAI<sub>S</sub> bit in the TABSR register to 1 (count starts).

Always make sure bits TA0TGL and TA0TGH in the TAIMR register, the ONSF register, and the TRGSR register are modified while the TAI<sub>S</sub> bit remains 0 (count stops) regardless whether after reset or not.

When setting TAI<sub>S</sub> bit to 0 (count stop), the followings occur:

- A counter stops counting and a content of reload register is reloaded.
- TAI<sub>OUT</sub> pin outputs "L".
- After one cycle of the CPU clock, the IR bit in TAI<sub>IC</sub> register is set to 1 (interrupt request).

Output in one-shot timer mode synchronizes with a count source internally generated. When the external trigger has been selected, a maximum delay of one cycle of the count source occurs between the trigger input to TAI<sub>IN</sub> pin and output in one-shot timer mode.

The IR bit is set to 1 when timer operation mode is set with any of the following procedures:

- Select one-shot timer mode after reset.
- Change an operation mode from timer mode to one-shot timer mode.
- Change an operation mode from event counter mode to one-shot timer mode.

To use the timer A<sub>i</sub> interrupt (the IR bit), set the IR bit to 0 after the changes listed above have been made.

When a trigger occurs while the timer is counting, the counter reloads the reload register value, and continues counting after a second trigger is generated and the counter is decremented once. To generate a trigger while counting, space more than one cycle of the timer count source from the first trigger and generate again.

When selecting the external trigger for the count start conditions in timer A one-shot timer mode, do generate an external trigger 300ns before the count value of timer A is set to 0000<sub>16</sub>. The one-shot timer does not continue counting and may stop.

## Timer A (Pulse Width Modulation Mode)

The timer remains idle after reset. Set the mode, count source, counter value, etc. using bits TA0TGL and TA0TGH in the TAIiMR ( $i = 0$  to 4) register, the TAI register, the ONSF register and the TRGSR register before setting the TAIiS bit in the TABSR register to 1 (count starts).

Always make sure bits TA0TGL and TA0TGH in the TAIiMR register, the ONSF register and the TRGSR register are modified while the TAIiS bit remains 0 (count stops) regardless whether after reset or not.

The IR bit is set to 1 when setting a timer operation mode with any of the following procedures:

- Select the PWM mode after reset.
- Change an operation mode from timer mode to PWM mode.
- Change an operation mode from event counter mode to PWM mode.

To use the timer Ai interrupt (interrupt request bit), set the IR bit to 0 by program after the above listed changes have been made.

When setting TAIiS register to 0 (count stop) during PWM pulse output, the following action occurs:

- Stop counting.
- When TAIiOUT pin is output "H", output level is set to "L" and the IR bit is set to 1.
- When TAIiOUT pin is output "L", both output level and the IR bit remains unchanged.

## Serial I/O

### Clock-Synchronous Serial I/O

#### Transmission/reception

With an external clock selected, and choosing the  $\overline{\text{RTS}}$  function, the output level of the  $\overline{\text{RTSi}}$  pin goes to “L” when the data-receivable status becomes ready, which informs the transmission side that the reception has become ready. The output level of the  $\overline{\text{RTSi}}$  pin goes to “H” when reception starts. So if the  $\overline{\text{RTSi}}$  pin is connected to the  $\overline{\text{CTSi}}$  pin on the transmission side, the circuit can transmission and reception data with consistent timing. With the internal clock, the  $\overline{\text{RTS}}$  function has no effect.

#### Transmission

When an external clock is selected, the conditions must be met while if the CKPOL bit in the UiC0 register is set to 0 (transmit data output at the falling edge and the receive data taken in at the rising edge of the transfer clock), the external clock is in the high state; if the CKPOL bit in the UiC0 register is set to 1 (transmit data output at the rising edge and the receive data taken in at the falling edge of the transfer clock), the external clock is in the low state.

- The TE bit in UiC1 register is set to 1 (transmission enabled)
- The TI bit in UiC1 register is set to 0 (data present in UiTB register)
- If CTS function is selected, input on the  $\overline{\text{CTSi}}$  pin is set to “L”

#### Reception

In operating the clock-synchronous serial I/O, operating a transmitter generates a shift clock. Fix settings for transmission even when using the device only for reception. Dummy data is output to the outside from the TxDi pin when receiving data.

When an internal clock is selected, set the TE bit in the UiC1 register ( $i = 0$  to 2) to 1 (transmission enabled) and write dummy data to the UiTB register, and the shift clock will thereby be generated. When an external clock is selected, set the TE bit in the UiC1 register ( $i = 0$  to 2) to 1 and write dummy data to the UiTB register, and the shift clock will be generated when the external clock is fed to the CLKi input pin.

When successively receiving data, if all bits of the next receive data are prepared in the UARTi receive register while the RE bit in the UiC1 register ( $i = 0$  to 2) is set to 1 (data present in the UiRB register), an overrun error occurs and the UiRB register OER bit is set to 1 (overrun error occurred). In this case, because the content of the UiRB register is undefined, a corrective measure must be taken by programs on the transmit and receive sides so that the valid data before the overrun error occurred will be retransmitted. Note that when an overrun error occurred, the SiRIC register IR bit does not change state.

To receive data in succession, set dummy data in the lower-order byte of the UiTB register every time reception is made.

When an external clock is selected, make sure the external clock is in high state if the CKPOL bit is set to 0, and in low state if the CKPOL bit is set to 1 before the following conditions are met:

- The RE bit in the UiC1 register is set to 1 (reception enabled)
- The TE bit in the UiC1 register is set to 1 (transmission enabled)
- The TI bit in the UiC1 register= 0 (data present in the UiTB register)

## UART Mode

### Special Mode 1 (I<sup>2</sup>C bus Mode)

When generating start, stop and restart conditions, set the STSPSEL bit in the U2SMR4 register to 0 and wait for more than half cycle of the transfer clock before setting each condition generate bit (STAREQ, RSTAREQ and STPREQ) from 0 to 1.

### SI/O3, SI/O4

The SOUT<sub>i</sub> default value which is set to the SOUT<sub>i</sub> pin by the SMi7 bit approximately 10ns may be output when changing the SMi3 bit from 0 (I/O port) to 1 (SOUT<sub>i</sub> output and CLKfunction) while the SMi2 bit in the SiC (i=3 and 4) to 0 (SOUT<sub>i</sub> output) and the SMi6 bit is set to 1 (internal clock). And then the SOUT<sub>i</sub> pin is held high-impedance.

If the level which is output from the SOUT<sub>i</sub> pin is a problem when changing the SMi3 bit from 0 to 1, set the default value of the SOUT<sub>i</sub> pin by the SMi7 bit.

### Programmable I/O Ports

The input threshold voltage of pins differs between programmable input/output ports and peripheral functions.

Therefore, if any pin is shared by a programmable input/output port and a peripheral function and the input level at this pin is outside the range of recommended operating conditions V<sub>IH</sub> and V<sub>IL</sub> (neither “high” nor “low”), the input level may be determined differently depending on which side—the programmable input/output port or the peripheral function—is currently selected.

## Flash Memory Version

### Functions to Inhibit Rewriting Flash Memory Rewrite

ID codes are stored in addresses 0FFFFDF<sub>16</sub>, 0FFFE3<sub>16</sub>, 0FFFE<sub>16</sub>, 0FFFEF<sub>16</sub>, 0FFFF3<sub>16</sub>, 0FFFF7<sub>16</sub>, and 0FFFFB<sub>16</sub>. If wrong data are written to these addresses, the flash memory cannot be read or written in standard serial I/O mode.

The ROMCP register is mapped in address 0FFFFFF<sub>16</sub>. If wrong data is written to this address, the flash memory cannot be read or written in parallel I/O mode.

In the flash memory version of MCU, these addresses are allocated to the vector addresses (“H”) of fixed vectors. The b3 to b0 in address 0FFFFFF<sub>16</sub> are reserved bits. Set these bits to 1111<sub>2</sub>.

### Regarding Programming/Erase Times and Execution Time

As the number of programming/erasure times increases, so does the execution time for software commands (Program, and Block Erase).

The software commands are aborted by hardware reset 1, brown-out detection reset (hardware reset 2), and watchdog timer interrupt. If a software command is aborted by such reset or interrupt, the affected block must be erased before reexecuting the aborted command.

### Definition of Programming/Erase Times

“Number of programs and erasure” refers to the number of erasure per block.

If the number of program and erasure is n (n=100 1,000) each block can be erased n times. For example, if a 8K byte block A is erased after writing 1 word data 4096 times, each to a different address, this is counted as one program and erasure. However, data cannot be written to the same address more than once without erasing the block. (Rewrite prohibited)

### Boot Mode

An undefined value is sometimes output in the I/O port until the internal power supply becomes stable when “H” is applied to the CNVss pin and “L” is applied to the RESET pin.

When setting the CNVss pin to “H”, the following procedure is required:

- (1) Apply an “L” signal to the RESET pin and the CNVss pin.
- (2) Bring VCC to more than 2.7V, and wait at least 2 msec. (Internal power supply stable waiting time)
- (3) Apply an “H” signal to the CNVss pin.
- (4) Apply an “H” signal to the RESET pin.

When the CNVss pin is “H” and RESET pin is “L”, P67 pin is connected to the pull-up resistor.

## APPENDIX



## IT800DLL Explanatory Note

June 2, 2005

## IT800 Data Link Layer – Functionality and Advantages

## 1. Scope

This document describes the functionality of the Data Link Layer (DLL) implemented in the products based on Yitran's IT800 technology for narrowband networking using the power line wiring.

The purpose of this document is to emphasize the advantages of the embedded algorithms and mechanisms of the DLL and to analyze the scenarios of using other implementations of a Data Link Layer.

## 2. Network Reference Module

The DLL is the 2nd layer of the 7 layers in the OSI network reference model illustrated in Figure 1(a). According to this model, data is transported upstream and downstream between subsequent layers, where the communication between layers of different communication nodes is basically carried out between the same layers [Figure 1(b)].

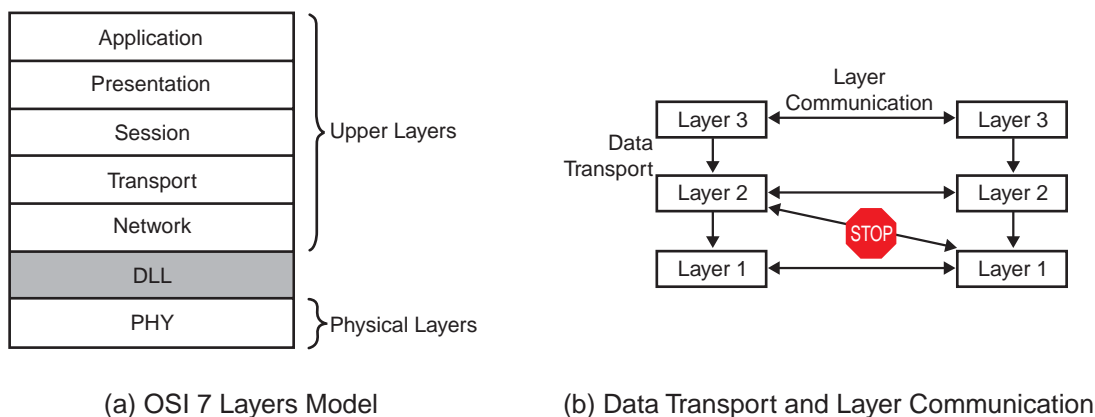


Figure 1: Network Reference Model

The Physical Layer (PHY) defines the electrical specifications for activating, the physical link between the communicating network system nodes. The DLL algorithms are designed to establish and manage a unique access channel between any two nodes as optimally and fairly as possible while minimizing the probability of collisions.

The IT800 DLL was developed by Yitran especially for the IT800 PHY and using the DLL optimize the utilization of the PHY and thus optimizes the overall performance.

The IT800 DLL implementation manages the time critical sessions of the interface with the PHY and uses the knowledge of the PHY implementation for optimal interrupt decoding and efficient interface interactions. The DLL takes advantage of various PHY features such as multiple transportation modes and the ability to create special packets and signals, enabling optimal utilization of the media channel while providing the most reliable data transfer.

## IT800DLL Explanatory Note

June 2, 2005

## 3. IT800DLL Main functions

The DLL main functions and mechanisms are described in the following table:

Function	Description
Carrier sense	Provides the carrier detection (CD) signal for triggering the media access algorithms as a function of the PHY correlator output, which indicates the probability of a signal on the line.
Channel access prioritization	Determines the sequence and time of packet transmissions for a PLC node, where the highest priority packet nodes participate in the DLL contention for the media access.
Adaptive back-off	Spreads the time over which a PLC node contends for the channel using a uniform distribution of the transmissions number over a given period of time (Yitran patent pending). This mechanism provides high efficiency of the network functionality, optimal utilization of the channel and sustains a viable network even in cases where many nodes contend for the channel simultaneously.
Acknowledgement	Serves for informing the transmitting node about the success or failure of a packet delivery to a target node by means of a traffic-free acknowledgement window.
Repetitive un-acknowledgement	This mechanism is used for transmitting packets a pre-determined number of times without requiring a reception acknowledgement from the target node.
Multiple hop broadcast	Retransmits single-network broadcast packets and CNC (Control Network Channel) messages to all the nodes participating in the same logical network.
Fragmentation and reassembly	Transfers packets longer than the maximal packet size allowed by the PHY by means of fragmentation in the transmitting node and reassembly in the receiving node
Packet filtering	Filters received packets according to their type for transferring only pre-defined types of messages to the upper layers and rejecting impostor node or other types of messages that are not required.

As can be seen from the table above, the IT800 DLL implements enhanced functionality, provides best performance in terms of channel access and throughput (based on Carrier Sense signaling, Adaptive Back-off algorithm and packet prioritization), reliable data transfer (using acknowledgement and repetition mechanisms, multi-hop transmissions and packet filtering) and allows easy integration of IT800 based solution with different protocol stacks. Another important advantage of using IT800DLL is assuring the co-existence between different IT800 based products operating in the same environment.



**IT800DLL Explanatory Note**

June 2, 2005

**4. Other DLL implementations**

In case of using other implementations of the DLL or considering such implementations, please be advised to consider the following issues:

1. **Co-existence:** Implementing a different channel access scheme, prioritization levels, packet formats and so forth, will not allow your product to co-exist with other IT800 based products that may share the same medium. Using the IT800 will enable such co-existence regardless of the product, the vendor, the application and the protocol used in top of the DLL.
2. **Required Knowledge, Time to Market and Cost:** Interfacing the DLL is much easier than interfacing the PHY. Physical layer interface includes handling of a several interrupts, critical timing sections and so forth. To be able to do so you require very good understanding of the both the power line medium and the PHY internal mechanisms and will significantly increase the development resources and schedule and as a result the time to market and product cost. In addition note that not all the PHY internal mechanism can be disclosed as some are in the status of patent-pending.
3. **Overall performance:** IT800DLL has been tested and integrated into different Control and Automation products over many years. The implementation has showed the best performance for Narrowband Power Line communication modem, when all the algorithms implemented in it were verified using the networking simulations and in real systems.

**5. Summary**

As a conclusion, we highly recommend to using IT800 DLL implementation in all type of products using the IT800 PHY, rather than using the latter by itself.

In case you still wish to self implement the DLL layer, please be sure to consult RENESAS support engineers.

REVISION HISTORY

M16C/6S Group

Rev.	Date	Description	
		Page	Summary
1.00	Jun 25, 2003	-	First edition issued
2.00	Aug 18, 2003	P1	L7 is changed.
		P2	T1.1.1 is changed.
		P3	F1.1.1 is changed.
		P4	T1.1.2 is changed.
		P5	F1.1.2 is changed.
		P7	T1.1.3 is changed.
		P8	T1.1.4 is changed.
		P9	L5 and F1.2.1 are changed.
		P12	Table of register is changed.
		P14	Table of register is changed.
		P15	Table of register is changed.
		P20	F1.5.2 is changed.
		P24	F1.6.1 is changed.
		P25	F1.6.2 is changed.
		P26	F1.6.3 is changed.
		P27	L2 and F1.7.1 are changed.
		P29	F1.7.2 is changed.
		P30	F1.7.3 is changed.
		P33	L4 to L5 and F1.7.6 are changed.
		P36	L5 is changed.
		P37	T1.7.2 is changed.
		P42	F1.7.8 is changed.
		P43	T1.7.6 is changed.
		P45	T1.7.7 is changed.
		P52	T1.9.2 is changed.
		P54	T1.9.3 is changed.
		P62	L2 is changed.
P83	F1.12.5 is changed.		
P84	F1.12.6 is changed.		
P85	L3 and T1.12.2 are changed.		
P88	F1.12.8 is changed.		
P90	F1.12.9 is changed.		
P92	F1.12.10 is changed.		
P94	L13 is changed.		
P95	F1.13.1 is changed.		
P100	F1.13.6 is changed.		
P107	F1.14.2 and F1.14.3 are changed.		
P119	T1.16.2 is changed.		

Rev.	Date	Description	
		Page	Summary
2.00	Aug 18, 2003	P121	PT1.16.4 is changed.
		P127	T1.16.6 is changed.
		P128	F1.16.5 is changed.
		–	Special Mode 3 is delated.
		P139	L10 to L11 and L17 to L19 and L21 and L30 to L32 and L38 to L40 are changed.
		P140	F1.18.1 is changed.
		P141	F1.18.2 is changed.
		P143	F1.18.4 is changed.
		P147	F1.18.8 is changed.
		P149	T1.18.1 is changed.
		P151	T1.19.1 is changed.
		P152	T1.19.2 is changed.
		P153	T1.19.4 is changed.
		P154	T1.19.5 is changed.
		P155	T1.19.6 is changed.
		P157	L2 and T1.19.8 are changed.
		P158	L2 is changed.
		P159	L2 is changed.
		P162	L3 to L4 are delated. T1.20.2 is changed.
		P163	1.Memory Map is changed. F1.20.1 is changed.
		P164	Boot Mode is changed.
		P166	L5 and T1.21.1 are changed.
			P168 is changed.
		P169	F1.21.2 is changed.
		P170	F1.21.3 is changed.
		P171	F1.21.4 is changed.
		P174	T1.21.2 is changed. Read Array Command is changed.
		P175	Program Command is changed.
		P176	Block Erase is changed.
		P177	Sequencer Status and T1.21.3 are changed.
		P178	T1.21.4 is changed.
		P179	F1.21.7 is changed.
		P181	T1.22.1 is changed.
P182	F1.22.1 is changed.		
P183	F1.22.2 is changed.		
P184	Parallel I/O Mode and User ROM and Boot ROM Areas are delated.		

Rev.	Date	Description	
		Page	Summary
2.01	Oct 27, 2003	P3 P7 P22 P23 P141 P143 P144 P178	L1 to L2 are changed. F1.1.1 is changed. T1.1.3 is changed. L3 to L4 are changed. (3) Setting PLC Mode is changed. T1.6.4 and F1.6.1 and F1.6.2 are changed. F1.18.4 is changed. F1.18.6 is changed. F1.18.7 is changed. Standard Serial I/O Mode is changed.
3.00	Jul 22, 2004	P1 P2 P4 P5 P6 P7 P9 P22 P23 P26 P27 P32 P33 P36 P37 P38 P44 P45 P46 P48 P49 P55 P58 P59 Between P60 to P61 P63 P79 P137 P140 P147 P148 P149 P150 P151 P152	Table of Contents is changed. T1.1.1 is changed. T1.1.2 is changed. F1.1.2 is changed. F1.1.3 is changed. T1.1.3 is changed. F1.2.1 is changed. Text is changed. T1.6.1 is delated. Text is changed. T1.6.4 and F1.6.2 are delated. Table name of T1.6.3 is added. T1.7.1 is changed. F1.7.1 is changed. Text is changed. F1.7.6 is changed. Text is changed. Text is changed. T1.7.4 is changed. Text is changed. T1.7.7, F1.7.9 and text are delated. Text is changed. F1.9.1 is changed. Text is changed (NMI Interrupt is delated.) T1.9.1 is changed. T1.9.5 is changed. F.1.9.8 is changed. F.1.9.9 is changed. Text is delated (NMI Interrupt is delated.) Text is delated (NMI Interrupt is delated.) F.1.12.4 is changed. Text is changed. F.1.18.3 is changed. T.1.18.1 is changed. F.1.18.10 is changed. T.1.19.1 is changed. T.1.19.2 is changed. Deletion of a voltage desplay of a header. Deletion of a voltage desplay of a header. Deletion of a voltage desplay of a header.

Rev.	Date	Description	
		Page	Summary
3.00	Jul 22, 2004	P153	T.1.19.6 is changed. Deletion of a voltage display of a header.
		P154	T.1.19.7 is changed. Deletion of a voltage display of a header.
		155	Deletion of a voltage display of a header.
		P156	Deletion of a voltage display of a header.
		P157	T.1.19.15 is changed.
		P163	T.1.20.3 is changed.
		P167	T.1.21.2 is changed.
		P168	T.1.21.3 is changed.
		P169	T.1.21.4 is changed.
		P170	Text is changed.
		P180	IT800AFE (Analog Front End) is added.
		P181	F.1.23.2 is changed.
		P183	Text is changed. T.1.23.2 is changed.
		P184	F.1.23.4 is changed.
3.01	Feb 17, 2005	-	Words standardized (On-chip Oscillator).
		P7	T.1.1.3 addition and changed.
		P34	L11-L12 addition and changed.
		P38	L2-L5 addition and delated.
		P42	T.1.7.6 part of delated.
		P157	L19 is delated.
4.00	Aug 05, 2005	P1	Text is changed.
		P5	F.1.1.2 is changed. T.1.1.3 and F.1.1.3 are added.
		P7	T.1.1.4 is changed.
		P56	F.1.9.6 is changed.
		P57	F.1.9.7 is changed.
		P61	Text is changed. F.1.9.11 is changed.
		P66	Text is changed. F.1.10.1 is changed.
		P77	F.1.12.1 is changed.
		P78	F.1.12.2 is changed.
		P84	T.1.12.3 is changed.
		P85	F.1.12.8 is changed.
		P86	P86 is added.
		P88	T.1.12.5 is changed.
		P93	F.1.13.1 is changed.
		P94	F.1.13.2 is changed.
		P97	F.1.13.5 is changed.
P105	L1-L13 are added.		
P106	L5-L8 are added.		
P108	L1-L11 are added.		

Rev.	Date	Description	
		Page	Summary
4.00	Aug 05, 2005	P110 P114 P116 P118 P119 P121 P123 P130 P137 P147 P151 P160 P165 P176 P188 to P190	T.1.15.2 is changed. L1-L11 are added. L1-L10 are added. F.1.16.1 is changed. T.1.16.2 is changed. T.1.16.4 is changed. L8-L16 are added. Text is changed. L2, L5-L6 and L12 are changed. Note is added. T.1.19.3 is changed. T.1.20.1 is changed. L12-L13 are added. T.1.21.4 is changed. Appendix is added.
5.00	Apr 24, 2009	1 2 3 5 6 8 13 24 25 28 33 41 42 61 66 78	Overview "AMR (Automatic Meter Reading)" → "smart metering" Table 1.1.1 Operating Ambient Temperature "-40 to 105°C" is added. Table 1.1.2 "Error Correction" → "Error Correction/Detection" About Firmware "(Product name: D2DL)" is added. Table 1.1.3 is revised. Table 1.1.4 is added. Figure 1.1.2 is revised. Table 1.1.5 and 1.1.6 are added. Figure 1.1.3 is revised. Table 1.1.7 P85 is revised. Address 0004 <sub>16</sub> After reset "00000000 <sub>2</sub> " → "XXXX0X00 <sub>2</sub> " Address 0005 <sub>16</sub> After reset "00001000 <sub>2</sub> " → "00XX1010 <sub>2</sub> " Table 1.6.3 title is added. Figure 1.6.2 After reset "00000000 <sub>2</sub> " → "XXXX0X00 <sub>2</sub> " "00000011 <sub>2</sub> (CNVss pin ="H")" is deleted. Figure 1.6.3 After reset "0X001000 <sub>2</sub> " → "00XX10X0 <sub>2</sub> " Figure 1.7.1 is revised. (1) Main Clock "Xin" → "XIN" "Figure 1.7.8...transition." is deleted. Figure 1.7.8 Note 1 to Note 4 are revised. Figure 1.9.10 Note 1 "INT5IC" → "INT3IC" Watchdog Timer "If a sub-clock is...no matter how the WDC7 bit is set." is deleted. Figure 1.12.1 is revised.

Rev.	Date	Description	
		Page	Summary
5.00	Apr 24, 2009	92	"UARTi (i=0 to 2)" is revised.
		117	Table 1.16.1 "Error detection" is revised.
		147	Table 1.18.1 title "(Excluding Analog Pins)" is added.
		148	Figure 1.18.10 title "(Excluding Analog Pins)" is added.
		149	Table 1.19.1 Topr is revised, Note 1 is revised.
		150	Table 1.19.2 "Vcc1" → "Vcc", Note 1 is revised.
		152	Table 1.19.5 Note is deleted.
		153	Table 1.19.6 Note is revised.
		154	Table 1.19.7 Standard "95" is added. Note is revised.
		155 to 157	Timing Requirements "-20 to 85°C" → "-20 to 85°C/-40 to 85°C/-40 to 105°C"
		160	parallel writer → parallel programmer Table 1.20.1 "Block 0 to 5" → "Block 0 to 4" "See Figure 20.2.1 to 20.2.3" → "See Figure 1.20.1"
		161	"The boot ROM area is located ... in parallel input/output mode." → "The boot ROM area is reserved. The rewrite control program ... the area cannot be rewritten."
		164	"In CPU mode, only the user ROM area ... the boot ROM area cannot be rewritten." is deleted.
		178	Standard Serial I/O Mode "The main clock f1 is 5.12 MHz in this mode." is added.
		180	serial writer → serial programmer
		183	(1)Transmitting path is revised.
		184	Table 1.23.1 title "-20 to 85°C" → "-20 to 85°C/-40 to 85°C/-40 to 105°C"
		186	Table 1.23.2 title "-25°C" → "25°C"
		187	Figure 1.23.4 is revised.
		188 to 197	"Usage Notes" is added.
200	4. Other DLL implementations "the last three years" → "many years"		
5.01	Dec 10, 2009	2	Table 1.1.1 Memory Capacity; ROM is revised
		5	Table 1.1.4 "M306S0F8DGP" is added, Figure 1.1.2 ROM capacity "8: 64K bytes" is added
		10	Figure 1.2.1 Internal ROM "64K bytes" is added, Note 1 is revised
		26	Figure 1.6.4 Internal ROM "64K bytes" is added
		161	Figure 1.20.1 is added
		162	Figure 1.20.2 title "(ROM Capacity 96K bytes)" is added.

Notes:

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 has used reasonable care in compiling the information included in this document, but 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 undersea 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 implantations
  - (3) healthcare intervention (e.g., excision, administration of medication, etc.)
  - (4) any other purposes that pose a direct threat to human lifeRenesas 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.



**RENESAS SALES OFFICES**

<http://www.renesas.com>

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

**Renesas Technology America, Inc.**  
450 Holger Way, San Jose, CA 95134-1368, U.S.A  
Tel: <1> (408) 382-7500, Fax: <1> (408) 382-7501

**Renesas Technology Europe Limited**  
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.  
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

**Renesas Technology (Shanghai) Co., Ltd.**  
Unit 204, 205, AZIACenter, No.1233 Lujiazui Ring Rd, Pudong District, Shanghai, China 200120  
Tel: <86> (21) 5877-1818, Fax: <86> (21) 6887-7858/7898

**Renesas Technology Hong Kong Ltd.**  
7th Floor, North Tower, World Finance Centre, Harbour City, Canton Road, Tsimshatsui, Kowloon, Hong Kong  
Tel: <852> 2265-6688, Fax: <852> 2377-3473

**Renesas Technology Taiwan Co., Ltd.**  
10th Floor, No.99, Fushing North Road, Taipei, Taiwan  
Tel: <886> (2) 2715-2888, Fax: <886> (2) 3518-3399

**Renesas Technology Singapore Pte. Ltd.**  
1 Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632  
Tel: <65> 6213-0200, Fax: <65> 6278-8001

**Renesas Technology Korea Co., Ltd.**  
Kukje Center Bldg. 18th Fl., 191, 2-ka, Hangang-ro, Yongsan-ku, Seoul 140-702, Korea  
Tel: <82> (2) 796-3115, Fax: <82> (2) 796-2145

**Renesas Technology Malaysia Sdn. Bhd**  
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No.18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia  
Tel: <603> 7955-9390, Fax: <603> 7955-9510