M16C/6S Group

SINGLE-CHIP 16-BIT CMOS MICROCOMPUTER

REJ03B0014-0300Z
Rev.3.00
Jul 22, 2004

Overview

The M16C/6S group of single-chip microcomputers are built using the high-performance silicon gate CMOS process using a M16C/60 Series CPU core and are packaged in a 64-pin plastic molded LQFP. These single-chip microcomputers operate using sophisticated instructions featuring a high level of instruction efficiency. With 1M bytes of address space, they are capable of executing instructions at high speed. The M16C/6S group are suitable for Power Line Communications (PLC) for home networking included advanced technologies (called IT800) developed by ITRAN Corporations.

Applications

Power Line Communication

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

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
Number of basic instructions		91 instructions
Shortest instruction execution time		65.1 ns (f(BCLK)= 15.36MHz, V_{CC}= 3.0V to 3.6V)
Memory capacity	ROM	96K Byte
	RAM	24K Byte
I/O port		21 ports
Input port	$P8_5$	1 bit x 1
Multifunction timer		16 bits x 5 channels (TA0, TA1, TA2, TA3, TA4)
Serial I/O		3 channels (UART0, UART1, UART2) UART, clock synchronous, I^{2}C bus[1] (option[2]) 2 channels (SI/O3, SI/O4) Clock synchronous SI/O4 is internally connected to IT800
DMAC		2 channels (trigger: 18 sources)
Watchdog timer		15 bits x 1 (with prescaler)
Interrupt		21 internal and 3 external sources, 4 software sources, 7 levels
Clock generation circuit		2 circuits • Main clock } (This circuit contains a built-in feedback resistor) • Ring oscillator
Power supply voltage		3.0V to 3.6V
Flash memory	Program/erase voltage	3.3V ± 0.3V (Topr= 0 to 60°C)
	Number of program/erase	100 times
Power consumption		70mA (V_{CC}= V_{CCA}= 3.3V, f(X_{IN})= 5.12MHz)
I/O characteristics	I/O withstand voltage	3.6V
	Output current	5mA
Operating ambient temperature		-20 to 85°C
Device configuration		CMOS high performance silicon gate
Package		64-pin plastic mold LQFP

Notes:

1. I^{2}C Bus is a registered trademark of PHILIPS.
2. If you desire this option, please so specify.

Block Diagram and PLC application Outline

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

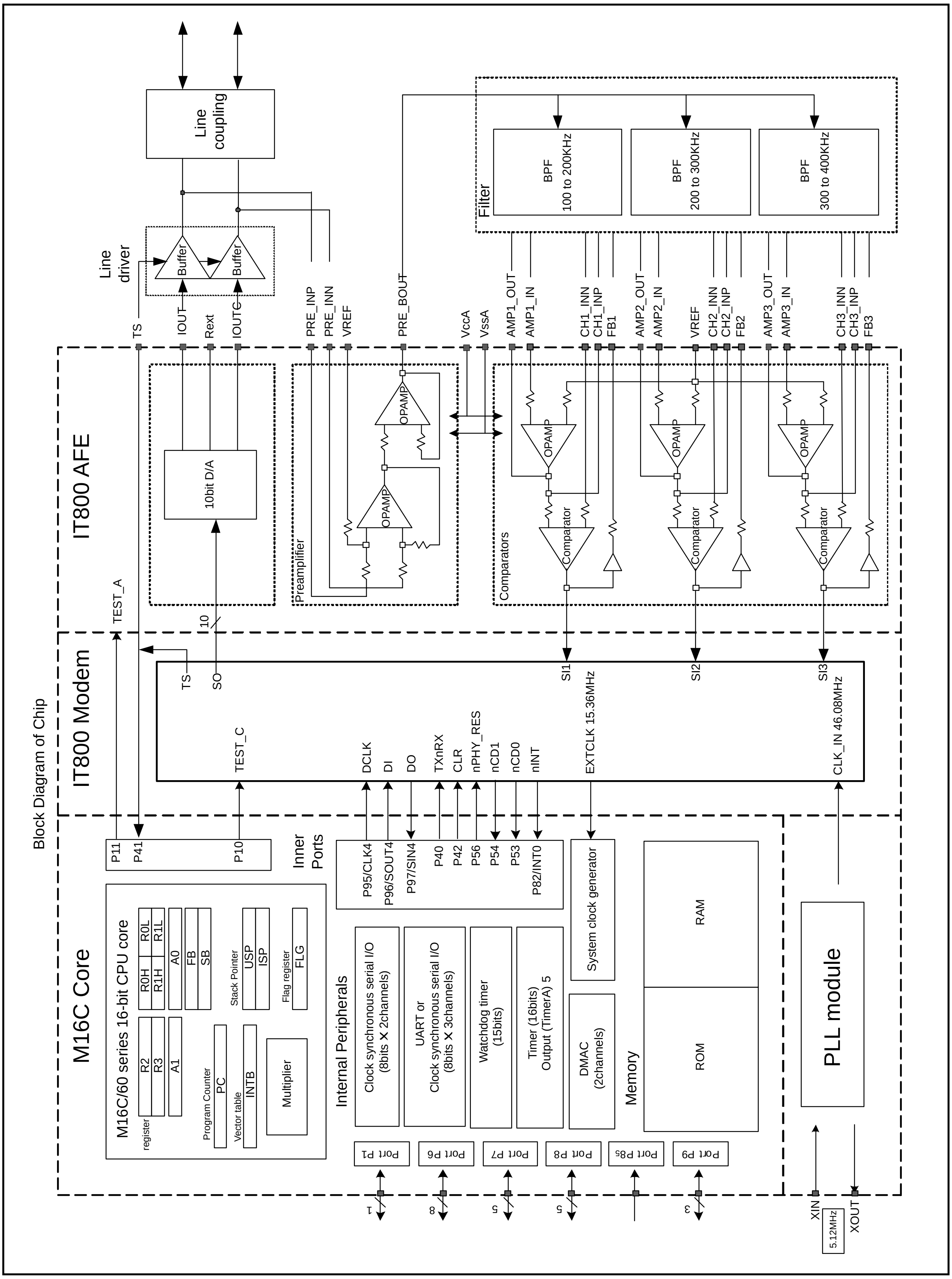


Figure 1.1.1. Block Diagram of Chip and PLC application Outline

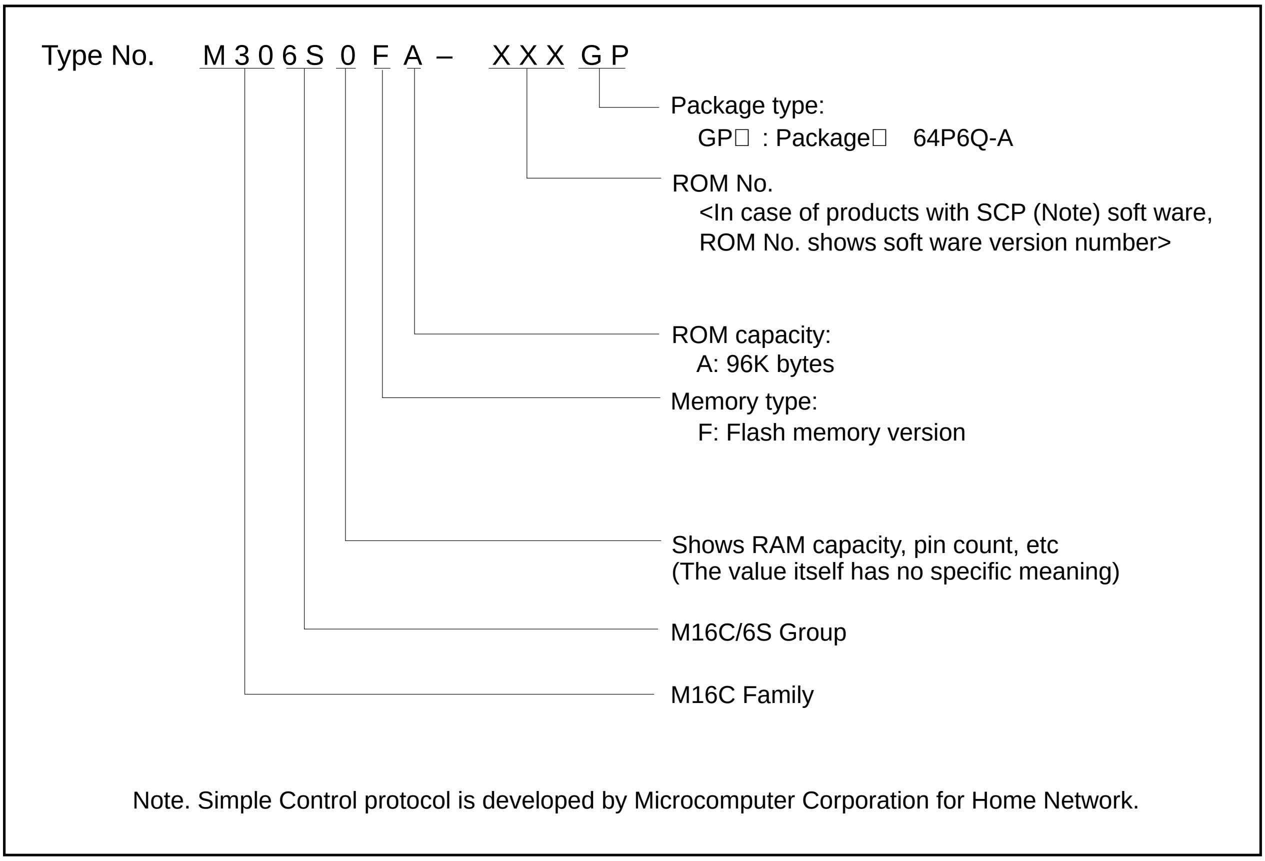
Product List

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

Table 1.1.2. Product List (1)

April 2004

Type No.	ROM capacity	RAM capacity	Package type	Remarks
M306S0FA-XXXGP	96K bytes	24K bytes	64P6Q-A	Flash Memory version


Type No. M 3 0 6 S 0 F A – X X X G P
Package type:
GP□ : Package□ 64P6Q-A
ROM No.
<In case of products with SCP (Note) soft ware,
ROM No. shows soft ware version number>
ROM capacity:
A: 96K bytes
Memory type:
F: Flash memory version
Shows RAM capacity, pin count, etc
(The value itself has no specific meaning)
M16C/6S Group
M16C Family

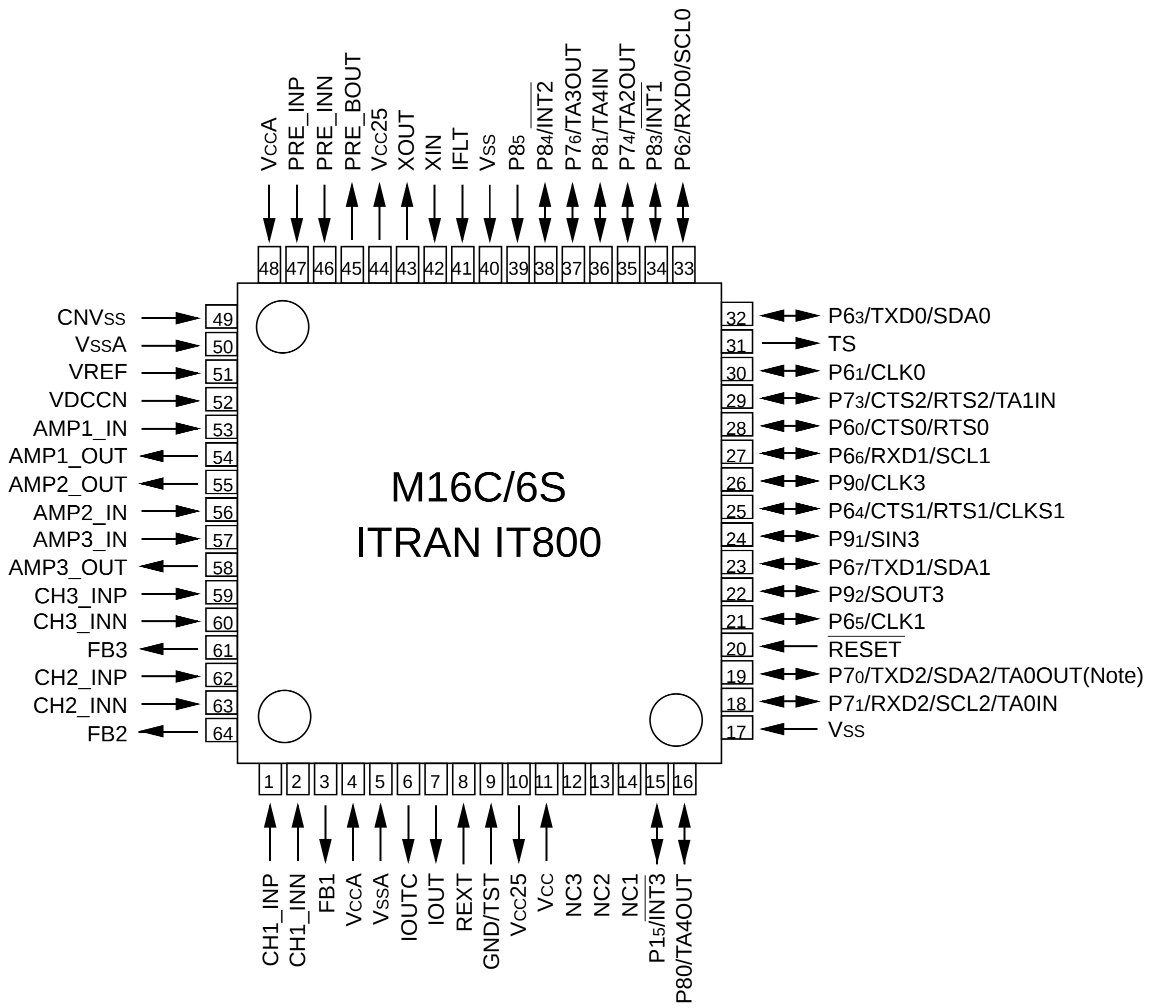
Note. Simple Control protocol is developed by Microcomputer Corporation for Home Network.

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

Pin Configuration

Figures 1.1.3 show the pin configurations (top view).

PIN CONFIGURATION (top view)



Package: 64P6Q-A

Note: P7$_0$ is N channel open-drain output pins.

Figure 1.1.3. Pin Configuration (Top View)

Table 1.1.3 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.
$\overline{\text{RESET}}$	Reset input	Input	“L” on this input resets the microcomputer.
XIN XOUT	Clock input Clock output	Input Output	These pins are provided for the main clock generating circuit input/output. Connect a ceramic resonator or crystal between the XIN and the XOUT pins. To use an externally derived clock, input it to the XIN pin and leave the XOUT pin open.
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.
$P1_5$	I/O port P1	Input/output	This is an 1-bit I/O port equiralent 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.
$P6_0$ to $P6_7$	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.
$P7_0$, $P7_1$, $P7_3$, $P7_4$, $P7_6$	I/O port P7	Input/output	It is an I/O port with a function equivalent to P6 (however, $P7_0$ N channel open-drain output). By choosing by the program, it functions as an I/O pin of timers A0 to A3. Moreover, $P7_0$ to $P7_3$ function also as an I/O pin of UART2.
$P8_0$, $P8_1$, $P8_3$, $P8_4$, $P8_5$	I/O port P8	Input/output	$P8_0$, $P8_1$, $P8_3$, and $P8_4$ are I/O ports with a function equivalent to P6. By choosing by the program, $P8_0$ to $P8_1$ function as an I/O pin of timer A4, and $P8_3$ to $P8_4$ functions as an input pin of $\overline{\text{INT}}$ interruption.
	Input port $P8_5$	Input	$P8_5$ is a port with the common use only for inputs. Pull-up resistance cannot set up this pin. This pin must be pulled-up externally to Vcc.
$P9_0$ to $P9_2$	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 SI/O3 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.
$P1_0$	Inner port $P1_0$	keep Input	Port for Testing. Please set Direction Register PD1_0 “0” for keeping Input.
$P1_1$	Inner port $P1_1$	keep Input	Port for Testing. Please set Direction Register PD1_1 “0” for keeping Input.
$P4_0$	Inner port $P4_0$	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 singal is connected to TXnRX of IT800 inside of chip.
$P4_1$	Inner port $P4_1$	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 singal is input from IT800 inside of chip.
$P4_2$	Inner port $P4_2$	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 singal is connected to CLR of IT800 inside of chip.
$P5_3$	Inner port $P5_3$	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 nCD0 singal is input from IT800 inside of chip.
$P5_4$	Inner port $P5_4$	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 singal is input from IT800 inside of chip.
$P5_6$	Inner port $P5_6$	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 singal is connected to nPHY_RES of IT800 inside of chip.
$P8_2$	Inner port $P8_2$	Input from IT800	I/O Port for communication between M16C Core and IT800 Modem (Note.1). nINT singal of IT800 is connected to this port inside of chip for carring out Interrupt function by software.
$P9_5$	Inner port $P9_5$	Output to IT800	I/O Port for communication between M16C Core and IT800 Modem. This port is connected to DCLK singal of IT800 inside of chip as SI/O4 CLK4 functional Output port by choosing by the program.
$P9_6$	Inner port $P9_6$	Output to IT800	I/O Port for communication between M16C Core and IT800 Modem. This port is connected to DI singal of IT800 inside of chip as SI/O4 Sout4 functional Output port by choosing by the program.
$P9_7$	Inner port $P9_7$	Input from IT800	I/O Port for communication between M16C Core and IT800 Modem. This port is connected to DI singal of IT800 inside of chip as SI/O4 Sout4 functional Input port by choosing by the program.

Note 1: In case of Direction Register “1”, any signal is not output from M16C to asigned signal of IT800 .
Refer to Programable I/O Ports pages.

Table 1.1.4 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_{16} to $FFFFF_{16}$.

The internal ROM is allocated in a lower address direction beginning with address $FFFFF_{16}$. For example, a 96-Kbyte internal ROM is allocated to the addresses from $E8000_{16}$ to $FFFFF_{16}$.

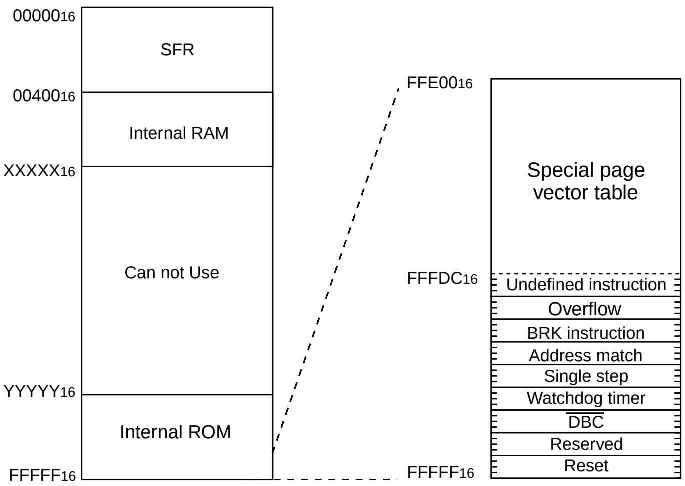
The fixed interrupt vector table is allocated to the addresses from $FFFDC_{16}$ to $FFFFF_{16}$. Therefore, store the start address of each interrupt routine here.

The internal RAM is allocated in an upper address direction beginning with address 00400_{16}. For example, a 24-Kbytes internal RAM is allocated to the addresses from 00400_{16} to $063FF_{16}$. 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_{16} to $003FF_{16}$. 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 $FFE00_{16}$ to $FFFDB_{16}$. This vector is used by the JMPS or JSRS instruction. For details, refer to the “M16C/60 and M16C/20 Series Software Manual.”

Internal RAM		Internal ROM	
Size	Address $XXXXX_{16}$	Size	Address $YYYYY_{16}$
24K bytes	$063FF_{16}$	96K bytes	$E8000_{16}$



Note 1: Shown here is a memory map for the case where the PM10 bit in the PM1 register is “1” and the PM13 bit in the PM1 register is “1”.

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.

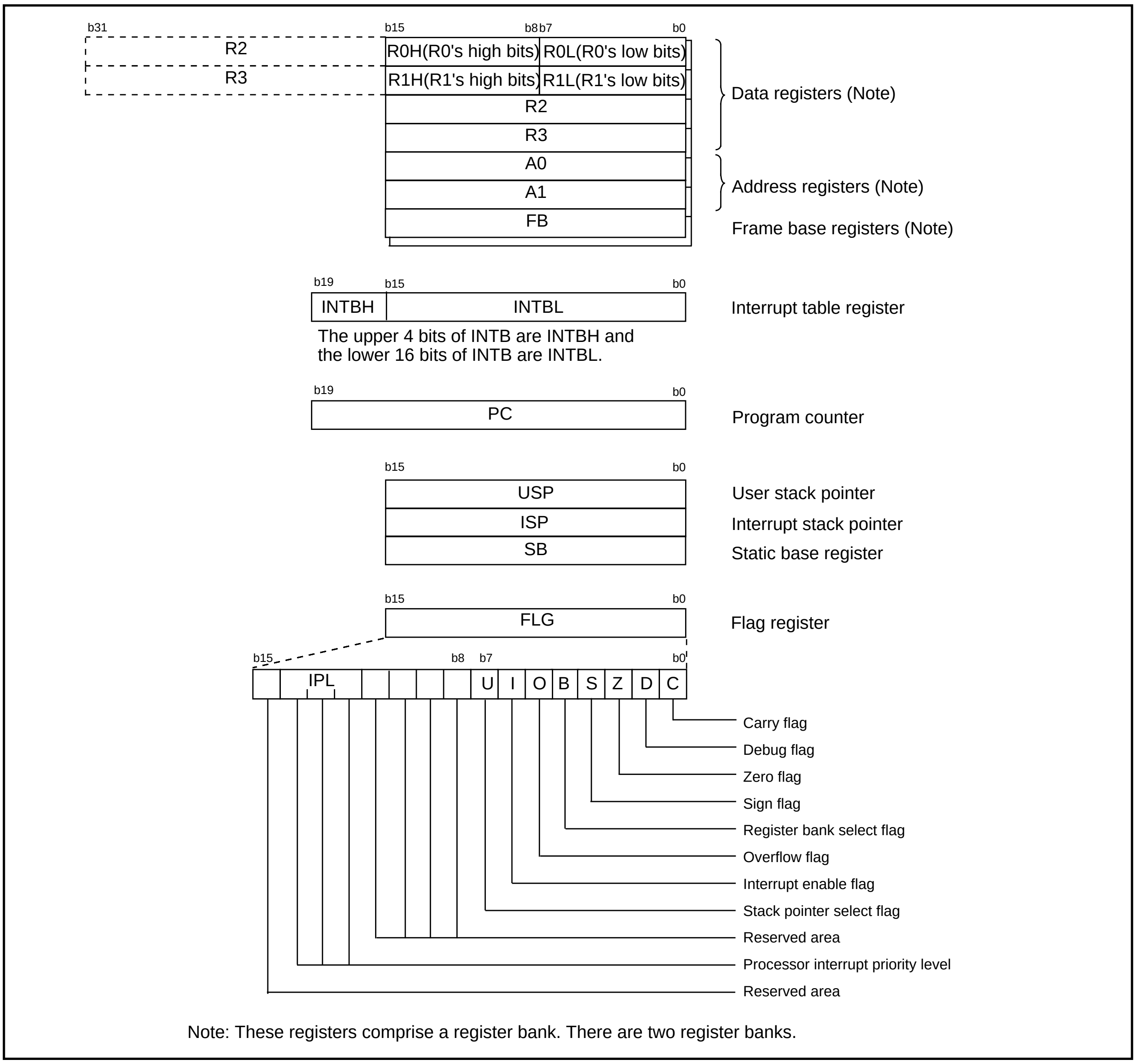


Figure 1.3.1. Central Processing Unit Register

(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_{16}			
0001_{16}			
0002_{16}			
0003_{16}			
0004_{16}	Processor mode register 0 (Note 2)	PM0	00000000_2(CNV$_{SS}$ pin is "L")
0005_{16}	Processor mode register 1	PM1	00001000_2
0006_{16}	System clock control register 0	CM0	01001000_2
0007_{16}	System clock control register 1	CM1	00100000_2
0008_{16}			
0009_{16}	Address match interrupt enable register	AIER	XXXXXX00_2
$000A_{16}$	Protect register	PRCR	XX000000_2
$000B_{16}$			
$000C_{16}$	Oscillation stop detection register (Note 3)	CM2	0000X000_2
$000D_{16}$			
$000E_{16}$	Watchdog timer start register	WDTS	$??_{16}$
$000F_{16}$	Watchdog timer control register	WDC	$00??????_2$
0010_{16}	Address match interrupt register 0	RMAD0	00_{16}
0011_{16}			00_{16}
0012_{16}			$X0_{16}$
0013_{16}			
0014_{16}	Address match interrupt register 1	RMAD1	00_{16}
0015_{16}			00_{16}
0016_{16}			$X0_{16}$
0017_{16}			
0018_{16}			
0019_{16}			
$001A_{16}$			
$001B_{16}$			
$001C_{16}$			
$001D_{16}$			
$001E_{16}$	Processor mode register 2	PM2	XXX00000_2
$001F_{16}$			
0020_{16}	DMA0 source pointer	SAR0	$??_{16}$
0021_{16}			$??_{16}$
0022_{16}			$X?_{16}$
0023_{16}			
0024_{16}	DMA0 destination pointer	DAR0	$??_{16}$
0025_{16}			$??_{16}$
0026_{16}			$X?_{16}$
0027_{16}			
0028_{16}	DMA0 transfer counter	TCR0	$??_{16}$
0029_{16}			$??_{16}$
$002A_{16}$			
$002B_{16}$			
$002C_{16}$	DMA0 control register	DM0CON	$00000?00_2$
$002D_{16}$			
$002E_{16}$			
$002F_{16}$			
0030_{16}	DMA1 source pointer	SAR1	$??_{16}$
0031_{16}			$??_{16}$
0032_{16}			$X?_{16}$
0033_{16}			
0034_{16}	DMA1 destination pointer	DAR1	$??_{16}$
0035_{16}			$??_{16}$
0036_{16}			$X?_{16}$
0037_{16}			
0038_{16}	DMA1 transfer counter	TCR1	$??_{16}$
0039_{16}			$??_{16}$
$003A_{16}$			
$003B_{16}$			
$003C_{16}$	DMA1 control register	DM1CON	$00000?00_2$
$003D_{16}$			
$003E_{16}$			
$003F_{16}$			

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_{16}			
0041_{16}			
0042_{16}			
0043_{16}			
0044_{16}	INT3 interrupt control register	INT3IC	XX00?000$_2$
0045_{16}			
0046_{16}	UART1 BUS collision detection interrupt control register	U1BCNIC	XXXX?000$_2$
0047_{16}	UART0 BUS collision detection interrupt control register	U0BCNIC	XXXX?000$_2$
0048_{16}	SI/O4 interrupt control register (S4IC)	S4IC	XX00?000$_2$
0049_{16}	SI/O3 interrupt control register	S3IC	XX00?000$_2$
$004A_{16}$	UART2 Bus collision detection interrupt control register	BCNIC	XXXX?000$_2$
$004B_{16}$	DMA0 interrupt control register	DM0IC	XXXX?000$_2$
$004C_{16}$	DMA1 interrupt control register	DM1IC	XXXX?000$_2$
$004D_{16}$			
$004E_{16}$			
$004F_{16}$	UART2 transmit interrupt control register	S2TIC	XXXX?000$_2$
0050_{16}	UART2 receive interrupt control register	S2RIC	XXXX?000$_2$
0051_{16}	UART0 transmit interrupt control register	S0TIC	XXXX?000$_2$
0052_{16}	UART0 receive interrupt control register	S0RIC	XXXX?000$_2$
0053_{16}	UART1 transmit interrupt control register	S1TIC	XXXX?000$_2$
0054_{16}	UART1 receive interrupt control register	S1RIC	XXXX?000$_2$
0055_{16}	Timer A0 interrupt control register	TA0IC	XXXX?000$_2$
0056_{16}	Timer A1 interrupt control register	TA1IC	XXXX?000$_2$
0057_{16}	Timer A2 interrupt control register	TA2IC	XXXX?000$_2$
0058_{16}	Timer A3 interrupt control register	TA3IC	XXXX?000$_2$
0059_{16}	Timer A4 interrupt control register	TA4IC	XXXX?000$_2$
$005A_{16}$			
$005B_{16}$			
$005C_{16}$			
$005D_{16}$	INT0 interrupt control register	INT0IC	XX00?000$_2$
$005E_{16}$	INT1 interrupt control register	INT1IC	XX00?000$_2$
$005F_{16}$	INT2 interrupt control register	INT2IC	XX00?000$_2$
0060_{16}			
0061_{16}			
0062_{16}			
0063_{16}			
0064_{16}			
0065_{16}			
0066_{16}			
0067_{16}			
0068_{16}			
0069_{16}			
$006A_{16}$			
$006B_{16}$			
$006C_{16}$			
$006D_{16}$			
$006E_{16}$			
$006F_{16}$			
0070_{16}			
0071_{16}			
0072_{16}			
0073_{16}			
0074_{16}			
0075_{16}			
0076_{16}			
0077_{16}			
0078_{16}			
0079_{16}			
$007A_{16}$			
$007B_{16}$			
$007C_{16}$			
$007D_{16}$			
$007E_{16}$			
$007F_{16}$			

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_{16}			
0081_{16}			
0082_{16}			
0083_{16}			
0084_{16}			
0085_{16}			
0086_{16}			
≈			≈
$01B0_{16}$			
$01B1_{16}$			
$01B2_{16}$			
$01B3_{16}$			
$01B4_{16}$			
$01B5_{16}$	Flash memory control register 1 (Note 2)	FMR1	0?00??0?$_2$
$01B6_{16}$			
$01B7_{16}$	Flash memory control register 0 (Note 2)	FMR0	??000001$_2$
$01B8_{16}$	Address match interrupt register 2	RMAD2	00_{16}
$01B9_{16}$			00_{16}
$01BA_{16}$			X0_{16}
$01BB_{16}$	Address match interrupt enable register 2	AIER2	XXXXXX00$_2$
$01BC_{16}$	Address match interrupt register 3	RMAD3	00_{16}
$01BD_{16}$			00_{16}
$01BE_{16}$			X0_{16}
$01BF_{16}$			
≈			≈
0250_{16}			
0251_{16}			
0252_{16}			
0253_{16}			
0254_{16}			
0255_{16}			
0256_{16}			
0257_{16}			
0258_{16}			
0259_{16}			
$025A_{16}$			
$025B_{16}$			
$025C_{16}$			
$025D_{16}$			
$025E_{16}$	Peripheral clock select register	PCLKR	00000011$_2$
$025F_{16}$			
≈			≈
0330_{16}			
0331_{16}			
0332_{16}			
0333_{16}			
0334_{16}			
0335_{16}			
0336_{16}			
0337_{16}			
0338_{16}			
0339_{16}			
$033A_{16}$			
$033B_{16}$			
$033C_{16}$			
$033D_{16}$			
$033E_{16}$			
$033F_{16}$			

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$_{16}$			
0341$_{16}$			
0342$_{16}$			
0343$_{16}$			
0344$_{16}$			
0345$_{16}$			
0346$_{16}$			
0347$_{16}$			
0348$_{16}$			
0349$_{16}$			
034A$_{16}$			
034B$_{16}$			
034C$_{16}$			
034D$_{16}$			
034E$_{16}$			
034F$_{16}$			
0350$_{16}$			
0351$_{16}$			
0352$_{16}$			
0353$_{16}$			
0354$_{16}$			
0355$_{16}$			
0356$_{16}$			
0357$_{16}$			
0358$_{16}$			
0359$_{16}$			
035A$_{16}$			
035B$_{16}$			
035C$_{16}$			
035D$_{16}$			
035E$_{16}$	Interrupt cause select register 2	IFSR2A	00XXXXXX$_2$
035F$_{16}$	Interrupt cause select register	IFSR	00$_{16}$
0360$_{16}$	SI/O3 transmit/receive register	S3TRR	??$_{16}$
0361$_{16}$			
0362$_{16}$	SI/O3 control register	S3C	01000000$_2$
0363$_{16}$	SI/O3 bit rate generator	S3BRG	??$_{16}$
0364$_{16}$	SI/O4 transmit/receive register	S4TRR	??$_{16}$
0365$_{16}$			
0366$_{16}$	SI/O4 control register	S4C	01000000$_2$
0367$_{16}$	SI/O4 bit rate generator	S4BRG	??$_{16}$
0368$_{16}$			
0369$_{16}$			
036A$_{16}$			
036B$_{16}$			
036C$_{16}$	UART0 special mode register 4	U0SMR4	00$_{16}$
036D$_{16}$	UART0 special mode register 3	U0SMR3	000X0X0X$_2$
036E$_{16}$	UART0 special mode register 2	U0SMR2	X0000000$_2$
036F$_{16}$	UART0 special mode register	U0SMR	X0000000$_2$
0370$_{16}$	UART1 special mode register 4	U1SMR4	00$_{16}$
0371$_{16}$	UART1 special mode register 3	U1SMR3	000X0X0X$_2$
0372$_{16}$	UART1 special mode register 2	U1SMR2	X0000000$_2$
0373$_{16}$	UART1 special mode register	U1SMR	X0000000$_2$
0374$_{16}$	UART2 special mode register 4	U2SMR4	00$_{16}$
0375$_{16}$	UART2 special mode register 3	U2SMR3	000X0X0X$_2$
0376$_{16}$	UART2 special mode register 2	U2SMR2	X0000000$_2$
0377$_{16}$	UART2 special mode register	U2SMR	X0000000$_2$
0378$_{16}$	UART2 transmit/receive mode register	U2MR	00$_{16}$
0379$_{16}$	UART2 bit rate generator	U2BRG	??$_{16}$
037A$_{16}$ 037B$_{16}$	UART2 transmit buffer register	U2TB	????????$_2$ XXXXXXX?$_2$
037C$_{16}$	UART2 transmit/receive control register 0	U2C0	00001000$_2$
037D$_{16}$	UART2 transmit/receive control register 1	U2C1	00000010$_2$
037E$_{16}$ 037F$_{16}$	UART2 receive buffer register	U2RB	????????$_2$?????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$_{16}$	Count start flag	TABSR	00$_{16}$
0381$_{16}$			
0382$_{16}$	One-shot start flag	ONSF	00$_{16}$
0383$_{16}$	Trigger select register	TRGSR	00$_{16}$
0384$_{16}$	Up-down flag	UDF	00$_{16}$
0385$_{16}$			
0386$_{16}$ 0387$_{16}$	Timer A0 register	TA0	??$_{16}$??$_{16}$
0388$_{16}$ 0389$_{16}$	Timer A1 register	TA1	??$_{16}$??$_{16}$
038A$_{16}$ 038B$_{16}$	Timer A2 register	TA2	??$_{16}$??$_{16}$
038C$_{16}$ 038D$_{16}$	Timer A3 register	TA3	??$_{16}$??$_{16}$
038E$_{16}$ 038F$_{16}$	Timer A4 register	TA4	??$_{16}$??$_{16}$
0390$_{16}$			
0391$_{16}$			
0392$_{16}$			
0393$_{16}$			
0394$_{16}$			
0395$_{16}$			
0396$_{16}$	Timer A0 mode register	TA0MR	00$_{16}$
0397$_{16}$	Timer A1 mode register	TA1MR	00$_{16}$
0398$_{16}$	Timer A2 mode register	TA2MR	00$_{16}$
0399$_{16}$	Timer A3 mode register	TA3MR	00$_{16}$
039A$_{16}$	Timer A4 mode register	TA4MR	00$_{16}$
039B$_{16}$			
039C$_{16}$			
039D$_{16}$			
039E$_{16}$			
039F$_{16}$			
03A0$_{16}$	UART0 transmit/receive mode register	U0MR	00$_{16}$
03A1$_{16}$	UART0 bit rate generator	U0BRG	??$_{16}$
03A2$_{16}$ 03A3$_{16}$	UART0 transmit buffer register	U0TB	????????$_{2}$ XXXXXXX?$_{2}$
03A4$_{16}$	UART0 transmit/receive control register 0	U0C0	00001000$_{2}$
03A5$_{16}$	UART0 transmit/receive control register 1	U0C1	00000010$_{2}$
03A6$_{16}$ 03A7$_{16}$	UART0 receive buffer register	U0RB	????????$_{2}$?????XX?$_{2}$
03A8$_{16}$	UART1 transmit/receive mode register	U1MR	00$_{16}$
03A9$_{16}$	UART1 bit rate generator	U1BRG	??$_{16}$
03AA$_{16}$ 03AB$_{16}$	UART1 transmit buffer register	U1TB	????????$_{2}$ XXXXXXX?$_{2}$
03AC$_{16}$	UART1 transmit/receive control register 0	U1C0	00001000$_{2}$
03AD$_{16}$	UART1 transmit/receive control register 1	U1C1	00000010$_{2}$
03AE$_{16}$ 03AF$_{16}$	UART1 receive buffer register	U1RB	????????$_{2}$?????XX?$_{2}$
03B0$_{16}$	UART transmit/receive control register 2	UCON	X0000000$_{2}$
03B1$_{16}$			
03B2$_{16}$			
03B3$_{16}$			
03B4$_{16}$			
03B5$_{16}$			
03B6$_{16}$			
03B7$_{16}$			
03B8$_{16}$	DMA0 request cause select register	DM0SL	00$_{16}$
03B9$_{16}$			
03BA$_{16}$	DMA1 request cause select register	DM1SL	00$_{16}$
03BB$_{16}$			
03BC$_{16}$			
03BD$_{16}$			
03BE$_{16}$			
03BF$_{16}$			

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
$03C0_{16}$			
$03C1_{16}$			
$03C2_{16}$			
$03C3_{16}$			
$03C4_{16}$			
$03C5_{16}$			
$03C6_{16}$			
$03C7_{16}$			
$03C8_{16}$			
$03C9_{16}$			
$03CA_{16}$			
$03CB_{16}$			
$03CC_{16}$			
$03CD_{16}$			
$03CE_{16}$			
$03CF_{16}$			
$03D0_{16}$			
$03D1_{16}$			
$03D2_{16}$			
$03D3_{16}$			
$03D4_{16}$			
$03D5_{16}$			
$03D6_{16}$			
$03D7_{16}$			
$03D8_{16}$			
$03D9_{16}$			
$03DA_{16}$			
$03DB_{16}$			
$03DC_{16}$			
$03DD_{16}$			
$03DE_{16}$			
$03DF_{16}$			
$03E0_{16}$			
$03E1_{16}$	Port P1 register□	P1□	$??_{16}$
$03E2_{16}$			
$03E3_{16}$	Port P1 direction register□	PD1□	00_{16}
$03E4_{16}$			
$03E5_{16}$			
$03E6_{16}$			
$03E7_{16}$			
$03E8_{16}$	Port P4 register□	P4□	$??_{16}$
$03E9_{16}$	Port P5 register□	P5□	$??_{16}$
$03EA_{16}$	Port P4 direction register□	PD4□	00_{16}
$03EB_{16}$	Port P5 direction register□	PD5□	00_{16}
$03EC_{16}$	Port P6 register□	P6□	$??_{16}$
$03ED_{16}$	Port P7 register□	P7□	$??_{16}$
$03EE_{16}$	Port P6 direction register□	PD6□	00_{16}
$03EF_{16}$	Port P7 direction register□	PD7□	00_{16}
$03F0_{16}$	Port P8 register□	P8□	$??_{16}$
$03F1_{16}$	Port P9 register□	P9□	$??_{16}$
$03F2_{16}$	Port P8 direction register□	PD8□	$00X00000_2$
$03F3_{16}$	Port P9 direction register□	PD9□	00_{16}
$03F4_{16}$			
$03F5_{16}$			
$03F6_{16}$			
$03F7_{16}$			
$03F8_{16}$			
$03F9_{16}$			
$03FA_{16}$			
$03FB_{16}$			
$03FC_{16}$	Pull-up control register 0 □	PUR0□	00_{16}
$03FD_{16}$	Pull-up control register 1□	□ PUR1□	00000000_2
$03FE_{16}$	Pull-up control register 2 □	PUR2□	00_{16}
$03FF_{16}$	Port control register □	PCR□	00_{16}

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 X_{IN} 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 td(P-R) or more until the internal power supply stabilizes.
(4) Supply a clock for 20 cycles or more to the X_{IN} 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.

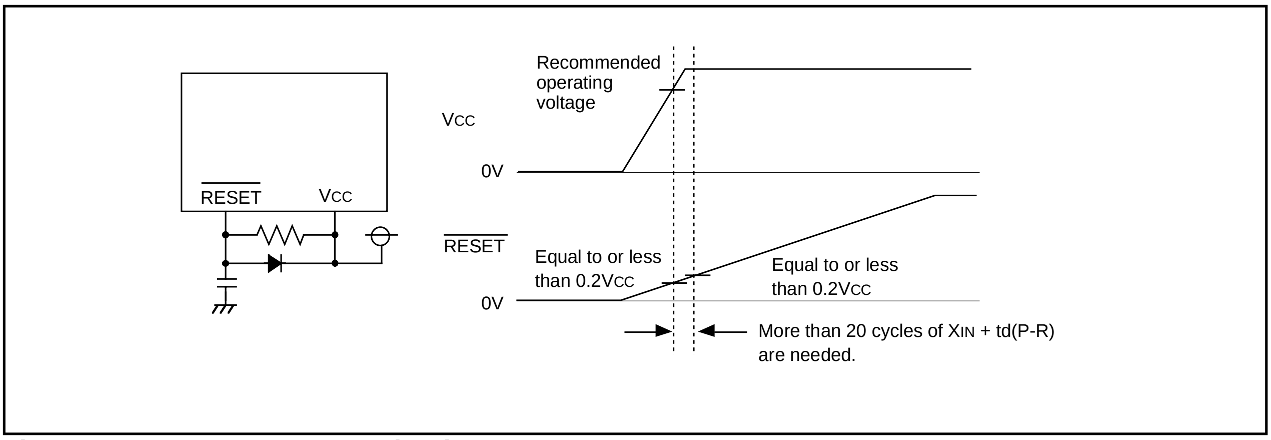

RESET
VCC
Recommended operating voltage
VCC
0V
RESET
Equal to or less than 0.2VCC
Equal to or less than 0.2VCC
0V
More than 20 cycles of XIN + td(P-R) are needed.

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.

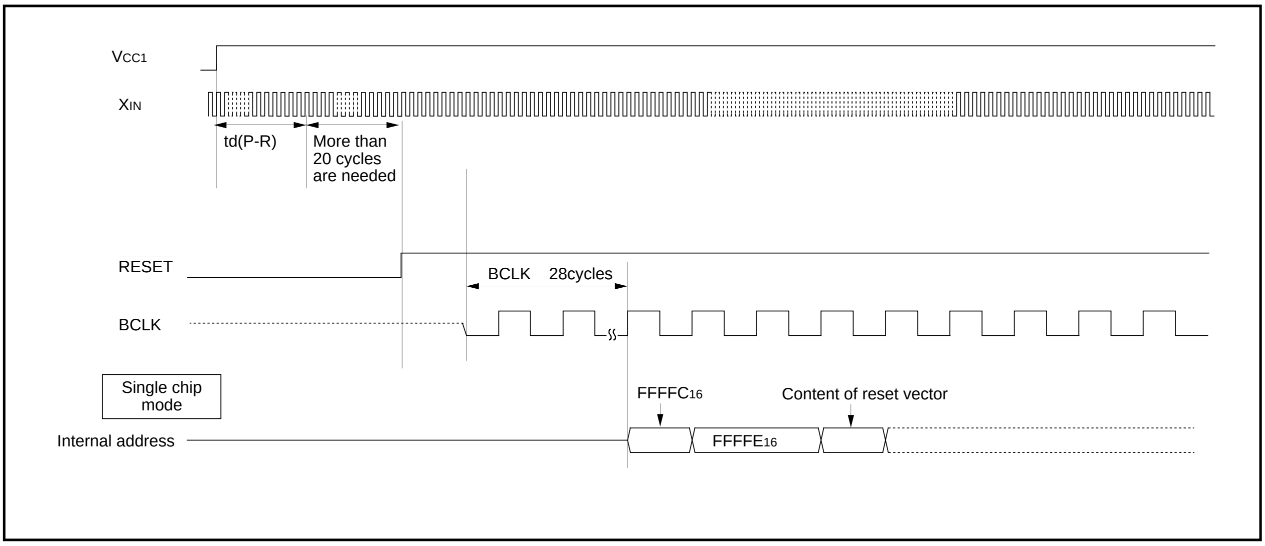


Figure 1.5.2. Reset Sequence

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

Pin name	Status
	CNVss = Vss
P15,	

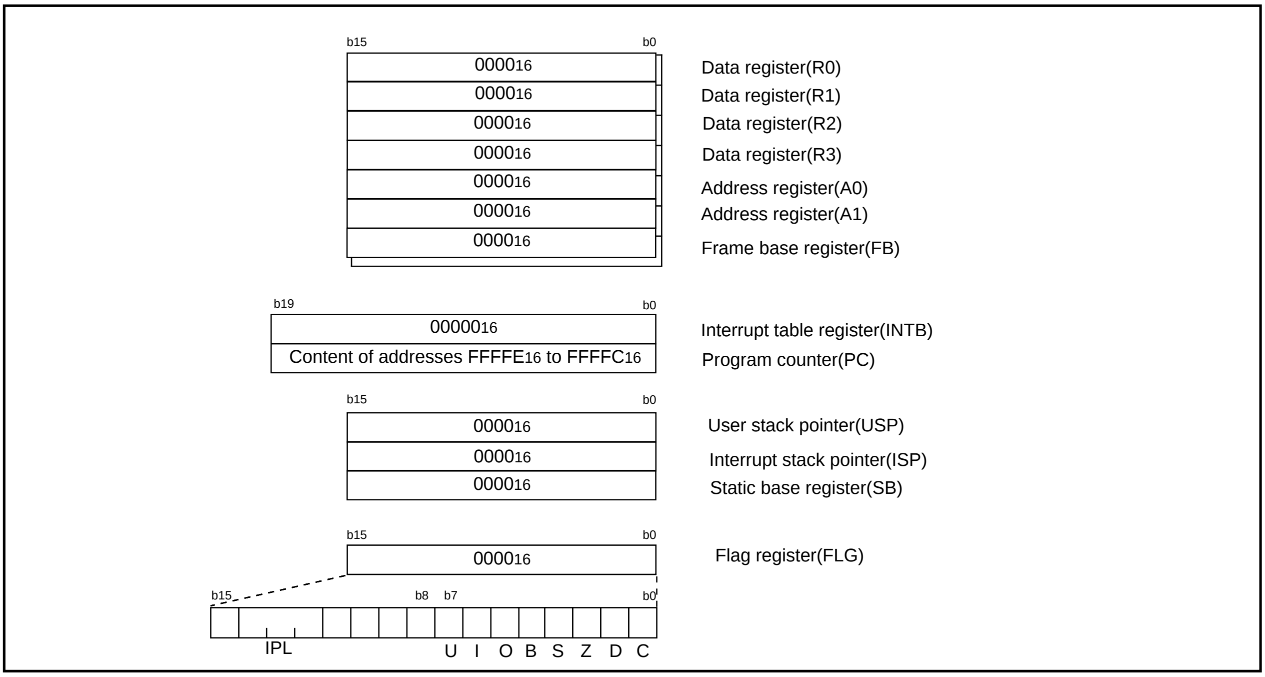


Figure 1.5.3. CPU Register Status After Rreset

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 1.6.2 shows the processor mode after hardware reset. Table 1.6.3 shows the PM01 to PM00 bit 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
00_2	Single-chip mode
01_2	Must not be set
10_2	
11_2	

Figure 1.6.4 show the memory map in single chip mode.

(3) Setting PLC Mode

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

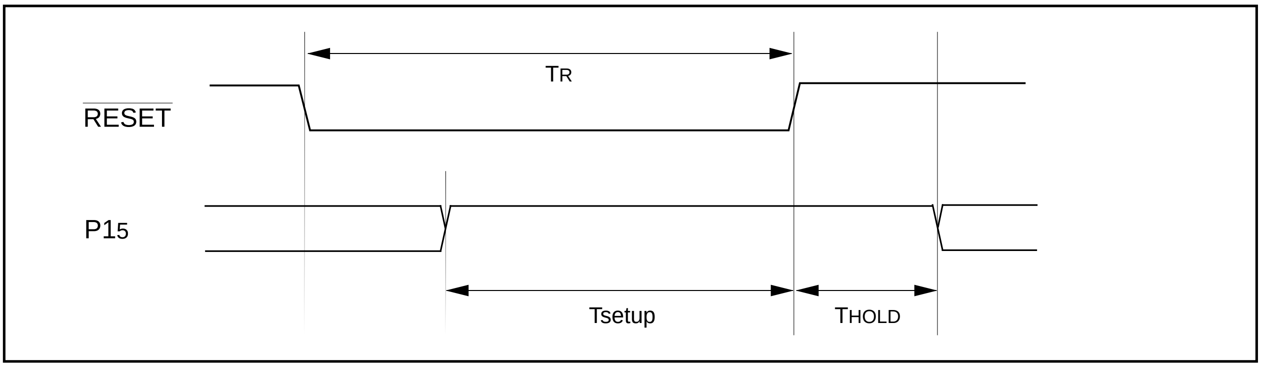
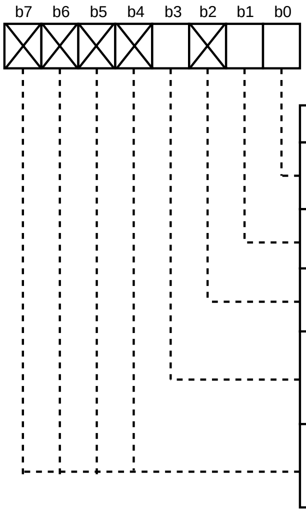


Figure 1.6.1. PLC mode by P1₅ simply setting

Table 1.6.3.

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

Processor mode register 0 (Note 1)



Symbol	Address	After reset (Note 2)
PM0	0004$_{16}$	00000000$_2$ (CNV$_{SS}$ pin = “L”) 00000011$_2$ (CNV$_{SS}$ pin = “H”)

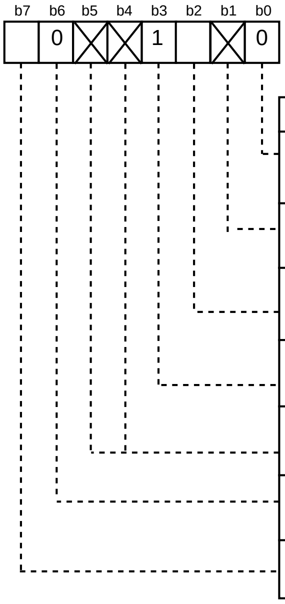
Bit symbol	Bit name	Function	RW
PM00	Processor mode bit (Note 2)	b1 b0 0 0: Single-chip mode 0 1: Must not be set 1 0: Must not be set 1 1: Must not be set	RW
PM01			RW
(b2)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
PM03	Software reset bit	Setting this bit to “1” resets the microcomputer. When read, its content is “0”.	RW
(b7-b4)	Nothing is assigned. When write, set to “0”. When read, their contents are indeterminate.		—

Note 1: Write to this register after setting the PRC1 bit in the PRCR register to "1" (write enable).

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

Figure 1.6.2. PM0 Register

Processor mode register 1 (Note 1)

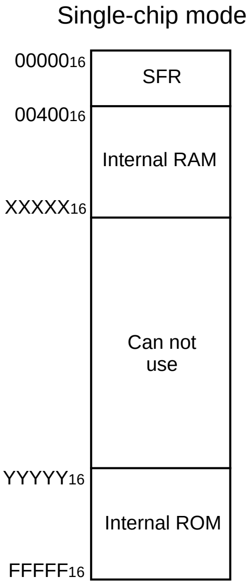


Symbol	Address	After reset
PM1	0005$_{16}$	0X001000$_2$

Bit symbol	Bit name	Function	RW
(b0)	Reserved bit	Should be set to “0”.	RW
(b2)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
PM12	Watchdog timer function select bit	0 : Watchdog timer interrupt 1 : Watchdog timer reset (Note 2)	RW
PM13	Internal reserved area expansion bit	Should be set to “1”.	RW
(b5-b4)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
(b6)	Reserved bit	Should be set to “0”.	RW
PM17	Wait bit (Note 3)	0 : No wait state 1 : With wait state (1 wait)	RW

Note 1: Write to this register after setting the PRC1 bit in the PRCR register to “1” (write enable).
Note 2: PM12 bit is set to “1” by writing a “1” in a program. (Writing a “0” has no effect.)
Note 3: When PM17 bit is set to “1” (with wait state), one wait state is inserted when accessing the internal RAM, internal ROM, or an external area.

Figure 1.6.3. PM1 Register



PM13=1 (Note1)

Internal RAM		Internal ROM	
Capacity	Address XXXXX₁₆	Capacity	Address YYYYY₁₆
24K bytes	063FF₁₆	96K bytes	E8000₁₆

Note 1: Since internal RAM which can be used becomes 15 K bytes when PM13 is “0”, please be sure to set PM13 to “1. ”

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

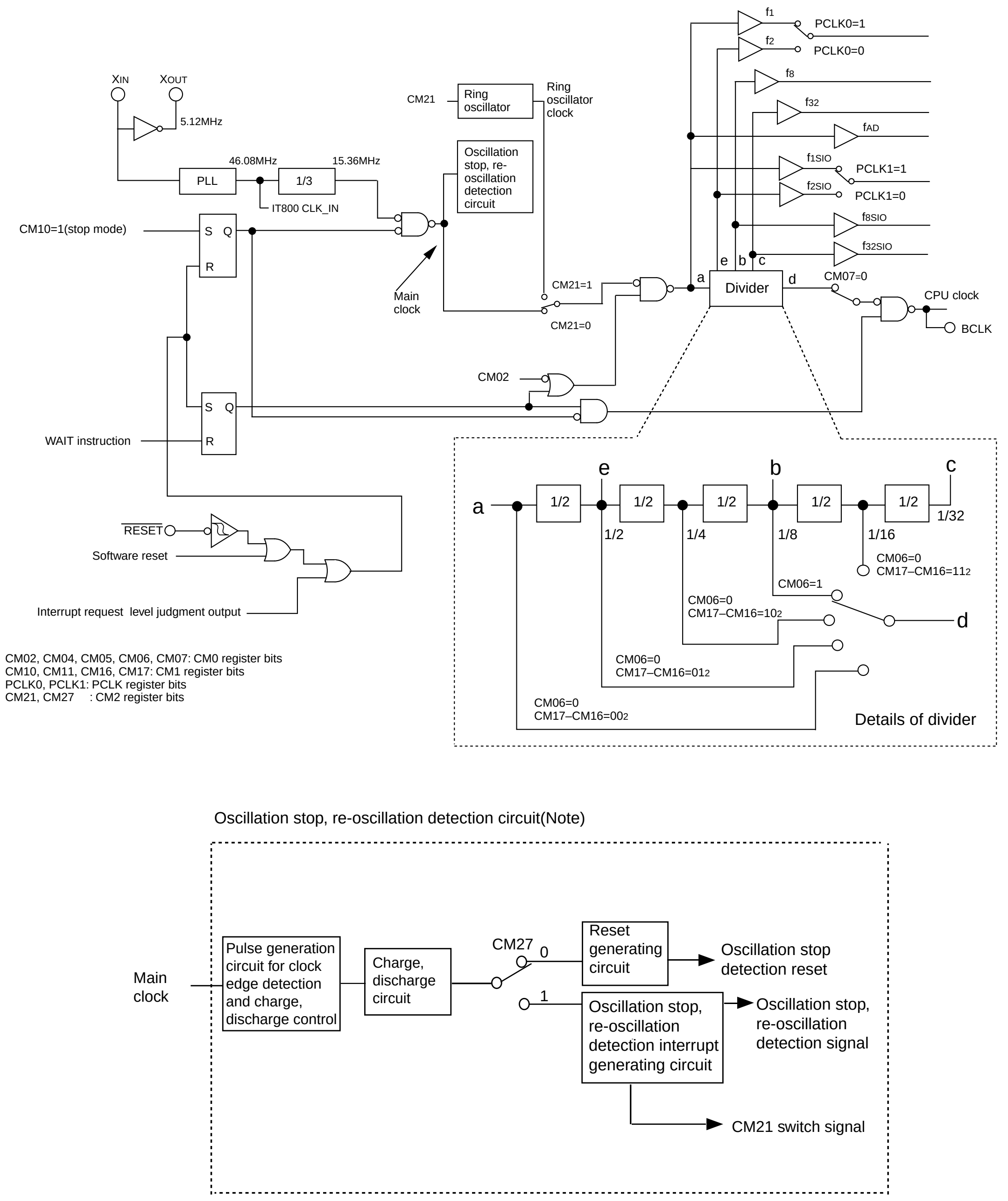
(2) Ring oscillator (oscillation stop detect function)

Table 1.7.1 lists the clock generation circuit specifications. Figure 1.7.1 shows the clock generation circuit. Figures 1.7.2 to 1.7.6 show the clock-related registers.

Table 1.7.1. Clock Generation Circuit Specifications

Item	Main clock oscillation circuit	Ring oscillator
Use of clock	• CPU clock source • Peripheral function clock source	• CPU clock source • Peripheral function clock source
Clock frequency	5.12 MHz	About 1 MHz
Usable oscillator	• Crystal oscillator (Note 1)	—
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	—

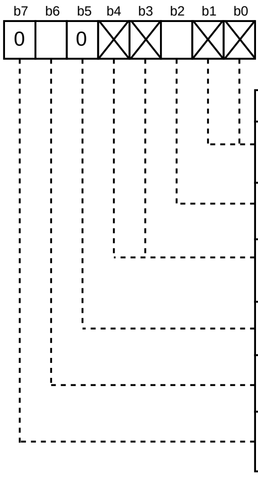
Note. Operating frequency must be 5.12 MHz, overall accuracy must be less than 150 ppm.


XIN
XOUT
5.12MHz
PLL
46.08MHz
1/3
15.36MHz
IT800 CLK_IN
CM21
Ring oscillator
Ring oscillator clock
Oscillation stop, re-oscillation detection circuit
CM10=1(stop mode)
S
Q
R
Main clock
CM21=1
CM21=0
CM02
WAIT instruction
RESET
Software reset
Interrupt request level judgment output
f1
PCLK0=1
f2
PCLK0=0
f8
f32
fAD
f1SIO
PCLK1=1
f2SIO
PCLK1=0
f8SIO
f32SIO
a
e
b
c
d
Divider
CM07=0
CPU clock
BCLK
CM02, CM04, CM05, CM06, CM07: CM0 register bits
CM10, CM11, CM16, CM17: CM1 register bits
PCLK0, PCLK1: PCLK register bits
CM21, CM27 : CM2 register bits
1/2
1/4
1/8
1/16
1/32
CM06=0
CM17–CM16=11₂
CM06=1
CM17–CM16=10₂
CM17–CM16=01₂
CM17–CM16=00₂
Details of divider
Oscillation stop, re-oscillation detection circuit(Note)
Main clock
Pulse generation circuit for clock edge detection and charge, discharge control
Charge, discharge circuit
CM27
0
1
Reset generating circuit
Oscillation stop detection reset
Oscillation stop, re-oscillation detection interrupt generating circuit
Oscillation stop, re-oscillation detection signal
CM21 switch signal

Note. Even if Xin input stops, PLL does not stop. Oscillation stop, re-oscillation detect circuit does not function.

Figure 1.7.1. Clock Generation Circuit

System clock control register 0 (Notes 1 and 4)



Symbol	Address	After reset
CM0	0006_{16}	01001000_2

Bit symbol	Bit name	Function	RW
— (b1-b0)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
CM02	WAIT peripheral function clock stop bit (Note 3)	0 : Do not stop peripheral function clock in wait mode 1 : Stop peripheral function clock in wait mode	RW
— (b4-b3)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
— (b5)	Reserved bit	Should be set to “0”.	RW
CM06	Main clock division select bit 0 (Notes 5, 13)	0 : CM16 and CM17 valid 1 : Division by 8 mode	RW
— (b7)	Reserved bit	Should be set to “0”.	RW

Note 1: Write to this register after setting the PRC0 bit of PRCR register to “1” (write enable).

Note 2: When entering stop mode from high or middle speed mode, ring oscillator mode or ring oscillator low power mode, the CM06 bit is set to “1” (divide-by-8 mode).

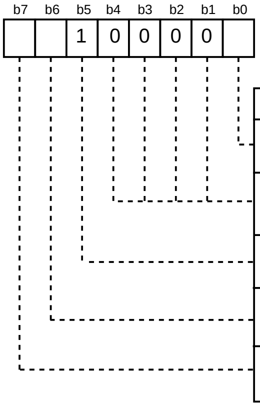
Note 3: When the PM21 bit of PM2 register is set to “1” (clock modification disable), writing to the CM02 bits has no effect.

Note 4: To use the main clock as the clock source for the CPU clock, follow the procedure below.
(1) Wait until td(M-L) elapses or the main clock oscillation stabilizes, whichever is longer.
(2) Set the CM21 to “0”.

Note 5: During ring oscillator low power dissipation mode, the divide-by-n value can be selected using the CM06 and CM17 to CM16 bits. To return to high or middle speed mode, however, set the CM06 bit to “1”, before selecting the desired mode.

Figure 1.7.2. CM0 Register

System clock control register 1 (Note 1)



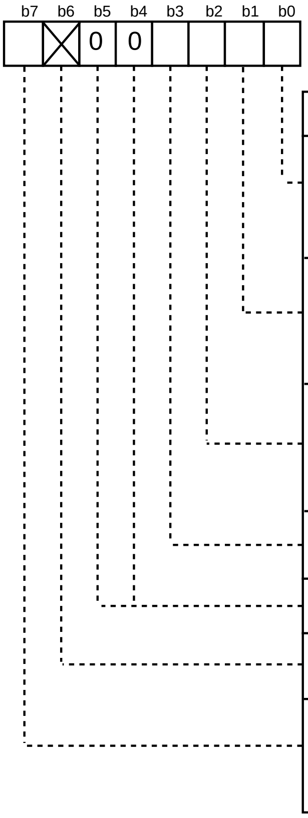
Symbol	Address	After reset
CM1	0007_{16}	00100000_2

Bit symbol	Bit	Function	RW
CM10	All clock stop control bit (Notes 3, 4)	0 : Clock on 1 : All clocks off (stop mode)	RW
—— (b4-b1)	Reserved bit	Must set to “0”	RW
—— (b5)	Reserved bit	Must set to “1”	RW
CM16	Main clock division select bit 1 (Note 2)	b7 b6 0 0 : No division mode 0 1 : Division by 2 mode 1 0 : Division by 4 mode 1 1 : Division by 16 mode	RW
CM17			RW

Note 1: Write to this register after setting the PRC0 bit of PRCR register to “1” (write enable).
Note 2: Effective when the CM06 bit is “0” (CM16 and CM17 bits enable).
Note 3: When the CM20 bit of CM2 register is set to “1” (oscillation stop, re-oscillation detection function enabled), do not set the CM10 bit to “1”.
Note 4: When the PM21 bit of PM2 register is set to “1” (clock modification disable), writing to the CM10 bits has no effect.
When the PM22 bit of PM2 register is set to “1” (watchdog timer count source is ring oscillator clock), writing to the CM10 bit has no effect.

Figure 1.7.3. CM1 Register

Oscillation stop detection register (Note 1)



Symbol	Address	After reset
CM2	000C$_{16}$	0X000000$_2$(Note 10)

Bit symbol	Bit name	Function	RW
CM20	Oscillation stop, re-oscillation detection bit (Notes 7, 8, 9, 10)	0: Oscillation stop, re-oscillation detection function disabled 1: Oscillation stop, re-oscillation detection function enabled	RW
CM21	System clock select bit 2 (Notes 2, 3, 6, 10)	0: Main clock (Ring oscillator turned off) 1: Ring oscillator clock (Ring oscillator oscillating)	RW
CM22	Oscillation stop, re-oscillation detection flag (Note 4)	0: Main clock stop, re-oscillation not detected 1: Main clock stop, re-oscillation detected	RW
CM23	X$_{IN}$ monitor flag (Note 5)	0: Main clock oscillating 1: Main clock turned off	RO
— (b5-b4)	Reserved bit	Must set to “0”	RW
— (b6)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
CM27	Operation select bit (when an oscillation stop, re-oscillation is detected) (Note 10)	0: Oscillation stop detection reset 1: Oscillation stop, re-oscillation detection interrupt	RW

Note 1: Write to this register after setting the PRC0 bit of PRCR register to “1” (write enable).

Note 2: When the CM20 bit is “1” (oscillation stop, re-oscillation detection function enabled), the CM27 bit is “1” (oscillation stop, re-oscillation detection interrupt), and the CPU clock source is the main clock, the CM21 bit is set to “1” (ring oscillator clock) if the main clock stop is detected.

Note 3: If the CM20 bit is “1” and the CM23 bit is “1” (main clock turned off), do not set the CM21 bit to “0”.

Note 4: This bit becomes “1” at main clock stop detection and main clock re-oscillation detection. When this bit changes from “0” to “1”, there arise oscillation stop, re-oscillation detection interrupt. Use this register to discriminate the causes for oscillation stop, re-oscillation detection interrupt and watchdog timer interrupt in the interrupt processing program. By writing “0” in the program, this bit becomes “0”. (Even when “1” is written in the program, no change is identified for the bit. Also, this bit is not set to “0” where there occur oscillation stop, re-oscillation detection interrupt.) When the CM22 bit is “1”, no oscillation stop, re-oscillation detection interrupt occur even if oscillation stop or re-oscillation is detected.

Note 5: Read the CM23 bit in an oscillation stop, re-oscillation detection interrupt handling routine to determine the main clock status.

Note 6: Effective when the CM07 bit of CM0 register is “0”.

Note 7: When the PM21 bit of PM2 register is “1” (clock modification disabled), writing to the CM20 bit has no effect.

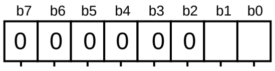
Note 8: Set the CM20 bit to “0” (disable) before entering stop mode. After exiting stop mode, set the CM20 bit back to “1” (enable).

Note 9: Set the CM20 bit to “0” (disable) before setting the CM05 bit of CM0 register.

Note 10: The CM20, CM21 and CM27 bits do not change at oscillation stop detection reset.

Figure 1.7.4. CM2 Register

Peripheral clock select register (Note)

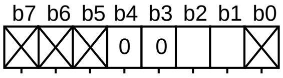


Symbol	Address	When reset
PCLKR	025E$_{16}$	00000011$_2$

Bit symbol	Bit name	Function	RW
PCLK0	Timers A, B clock select bit (Clock source for the timers A, B, and the dead time timer)	0 : f_2 1 : f_1	RW
PCLK1	SI/O clock select bit (Clock source for UART0 to UART2, SI/O3, SI/O4)	0 : f_{2SIO} 1 : f_{1SIO}	RW
—— (b7-b2)	Reserved bit	Must set to “0”	RW

Note: Write to this register after setting the PRC0 bit of PRCR register to “1” (write enable).

Processor mode register 2 (Note 1)



Symbol	Address	After reset
PM2	001E$_{16}$	XXX00000$_2$

Bit symbol	Bit name	Function	RW
— (b0)	Nothing is assigned. When write, set to “0”. When read, its content is interdeterminate.		—
PM21	System clock protective bit (Note 2, Note 3)	0 : Clock is protected by PRCR register 1 : Clock modification disabled	RW
PM22	WDT count source protective bit (Note 2, Note 4)	0 : CPU clock is used for the watchdog timer count source 1 : Ring oscillator clock is used for the watchdog timer count source	RW
— (b4-b3)	Reserved bit	Must set to “0”	RW
— (b7-b5)	Nothing is assigned. When write, set to “0”. When read, its content is interdeterminate.		—

Note 1: Write to this register after setting the PRC1 bit of PRCR register to “1” (write enable).
Note 2: Once this bit is set to “1”, it cannot be cleared to “0” in a program.
Note 3: Setting the PM21 bit to “1” results in the following conditions:

- The BCLK is not halted by executing the WAIT instruction.
- Writing to the following bits has no effect.
  CM02 bit of CM0 register
  CM05 bit of CM0 register (main clock is not halted)
  CM07 bit of CM0 register
  CM10 bit of CM1 register (stop mode is not entered)
  CM11 bit of CM1 register
  CM20 bit of CM2 register (oscillation stop, re-oscillation detection function settings do not change)

Note 4: Setting the PM22 bit to “1” results in the following conditions:

- The ring oscillator starts oscillating, and the ring oscillator clock becomes the watchdog timer count source.
- 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 or hold state.

Figure 1.7.5. PCLKR Register and PM2 Register

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

(1) Main Clock

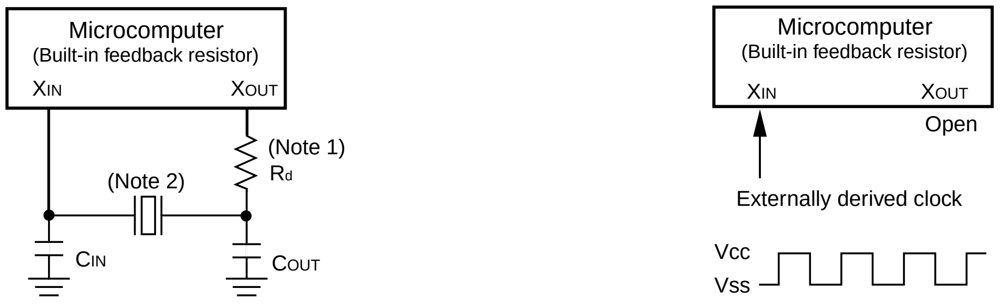
Main clock is supplied by IT800 with a tripled clock of XIN (main clock oscillator).

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 ocsillator does not stop.

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



Note 1: Insert a damping resistor if required. The resistance will vary depending on the oscillator and the oscillation drive capacity setting. Use the value recommended by the maker of the oscillator.
Also, if the oscillator manufacturer's data sheet specifies that a feedback resistor be added external to the chip, insert a feedback resistor between XIN and XOUT following the instruction.

Note 2: Operating frequency must be 5.12 MHz, overall accuracy must be less than 150 ppm.

Figure 1.7.6. Examples of Main Clock Connection Circuit

(3) Ring Oscillator Clock

This clock, approximately 1 MHz, is supplied by a ring 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” (ring oscillator clock for the watchdog timer count source), this clock is used as the count source for the watchdog timer. After reset, the ring oscillator clock is turned off. It is turned on by setting the CM21 bit of CM2 register to “1” (ring 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 ring oscillator clock.

If the main clock or ring 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, ring oscillator mode or low power ring oscillator mode, the CM06 bit of CM0 register is set to “1” (divide-by-8 mode).

(2) Peripheral Function Clock(f_1, f_2, f_8, f_{32}, f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO})

These are operating clocks for the peripheral functions.

Of these, fi (i = 1, 2, 8, 32) and f_{iSIO} are derived from the main clock, or ring oscillator clock by dividing them by i. The clock fi 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 fi, 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 ring 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 ring 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.
- **Ring Oscillator Mode**

  The ring oscillator clock divided by 1 (undivided), 2, 4, 8 or 16 provides the CPU clock. The ring 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	00_2	0
Midium-speed mode	divided by 2	0	01_2	0
	divided by 4	0	10_2	0
	divided by 8	0	—	1
	divided by 16	0	11_2	0
Ring oscillator mode	divided by 1	1	00_2	0
	divided by 2	1	01_2	0
	divided by 4	1	10_2	0
	divided by 8	1	—	1
	divided by 16	1	11_2	0

Note: The divide-by-n value can be selected the same way as in ring 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” (ring oscillator clock for the watchdog timer count source), the watchdog timer remains active. Because the main clock, and ring 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 f_1, f_2, f_8, f_{32}, f_{1SIO}, f_{8SIO}, and f_{32SIO} 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 “000_2” (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 periph eral 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 “000_2” (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 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. The least amount of power is consumed in this mode. If the voltage applied to Vcc pins is V_{RAM} or more, the internal RAM is retained.

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 seleted)

• 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 1.9.6 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 “000_2” (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 “000_2”.
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:

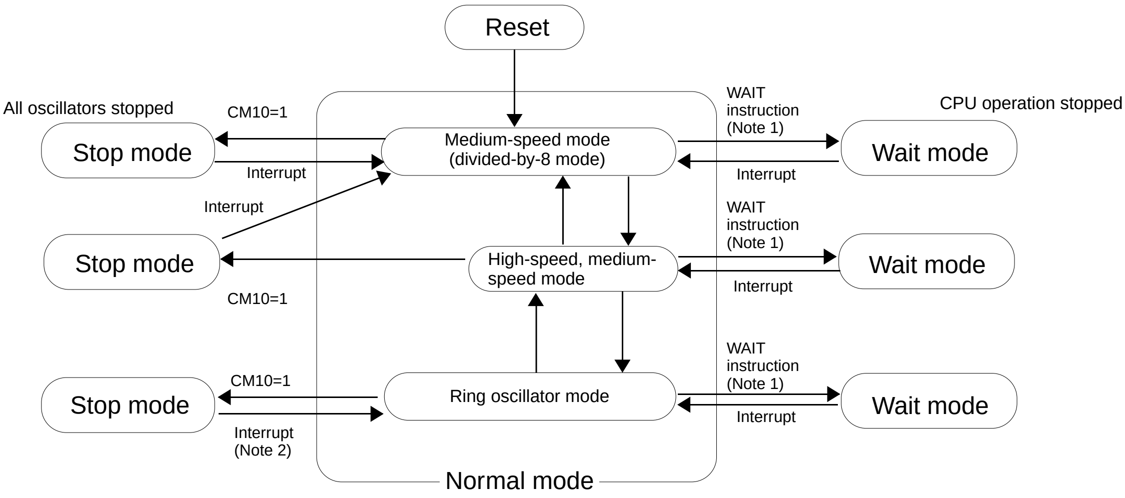
If the CPU clock before entering stop mode was derived from the main clock: main clock divide-by-8

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

Table 1.7.5. Pin Status in Stop Mode

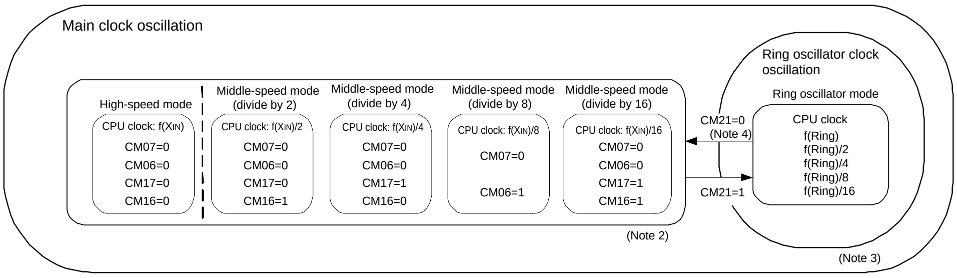
Pin	Status
I/O ports	Retains status before stop mode

Figure 1.7.7 shows the state transition from normal operation mode to stop mode and wait mode. Figure 1.7.8 shows the state transition in normal operation mode.
Table 1.7.6 shows a state transition matrix describing allowed transition and setting. The vertical line shows current state and horizontal line shows state after transition.



Note 1: When the PM21 bit = 0 (system clock protective function unused).
Note 2: The ring oscillator clock divided by 8 provides the CPU clock.
Note 3: Write to the CM0 register and CM1 register simultaneously by accessing in word units while CM21=0 (ring oscillator turned off).

Figure 1.7.7. State Transition to Stop Mode and Wait Mode



Notes:
1: Avoid making a transition when the CM20 bit is set to “1” (oscillation stop, re-oscillation detection function enabled).
Set the CM20 bit to “0” (oscillation stop, re-oscillation detection function disabled) before transiting.
2: Set the CM06 bit to “1” (division by 8 mode) before changing back the operation mode from ring oscillator mode to high- or middle-speed mode.
3: Please change according to the direction of an arrow.
4: When you return to high-speed and middle-speed mode from ring oscillator mode, please make CM06 into “1” (divede by 8) 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	Ring oscillator mode	Stop mode	Wait mode
Current state	High-speed mode, middle-speed mode	See Table A	(7)	(8)[1]	(9)
	Ring oscillator mode	(6)[2]	See Table A	(8)[1]	(9)
	Stop mode	(10)[3]	(10)[3]		--
	Wait mode	(10)	(10)	--	

--: Cannot transit

Table 1. State Transition with Main Clock Division Ration in High- or Middle-speed Mode, Ring Oscillator Mode, and Ring Oscillator Low Power Dissipation 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:
1. 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.
2. Set the CM06 bit to “1” (division by 8 mode) before transiting from ring oscillator mode to high- or middle-speed mode.
3. 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	Ring 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.

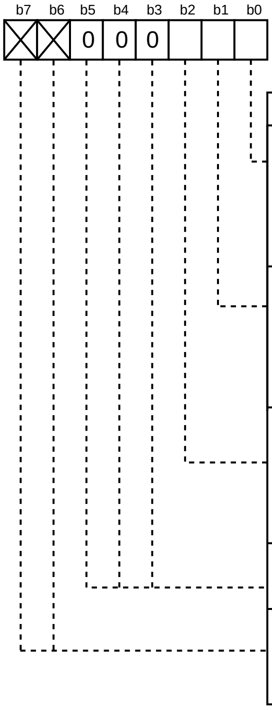
Protection

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.

Protect register



Symbol	Address	After reset
PRCR	000A$_{16}$	XX000000$_2$

Bit symbol	Bit name	Function	RW
PRC0	Protect bit 0	Enable write to CM0, CM1, CM2, PLC0 and PCLKR registers 0 : Write protected 1 : Write enabled	RW
PRC1	Protect bit 1	Enable write to PM0, PM1, PM2, TB2SC, INVC0 and INVC1 registers 0 : Write protected 1 : Write enabled	RW
PRC2	Protect bit 2	Enable write to PD9, S3C and S4C registers 0 : Write protected 1 : Write enabled	RW
(b5-b3)	Reserved bit	Must set to “0”	RW
(b7-b6)	Nothing is assigned. When write, set to “0”. When read, its content is interdeterminate.		—

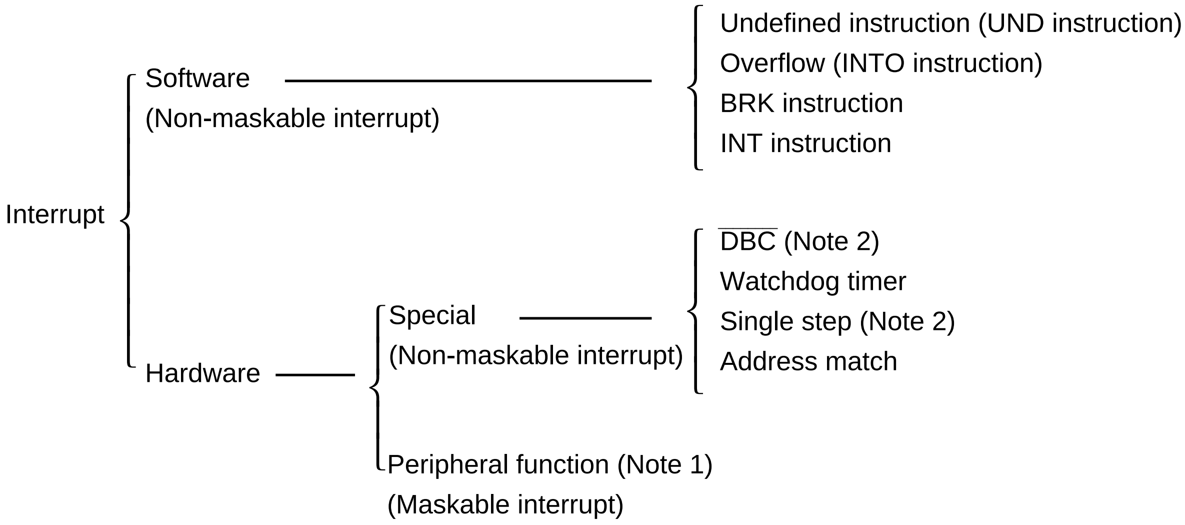
Note: The PRC2 bit is set to “0” by writing to any address after setting it to “1”. Other bits are not set to “0” by writing to any address, and must therefore be set in a program.

Figure 1.8.1. PRCR Register

Interrupts

Type of Interrupts

Figure 1.9.1 shows types of interrupts.



Note 1: Peripheral function interrupts are generated by the microcomputer's internal functions.

Note 2: Do not normally use this interrupt because it is provided exclusively for use by development support tools.

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 I0nterrupt: 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.

- **$\overline{\text{DBC}}$ 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 1.9.2. 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.

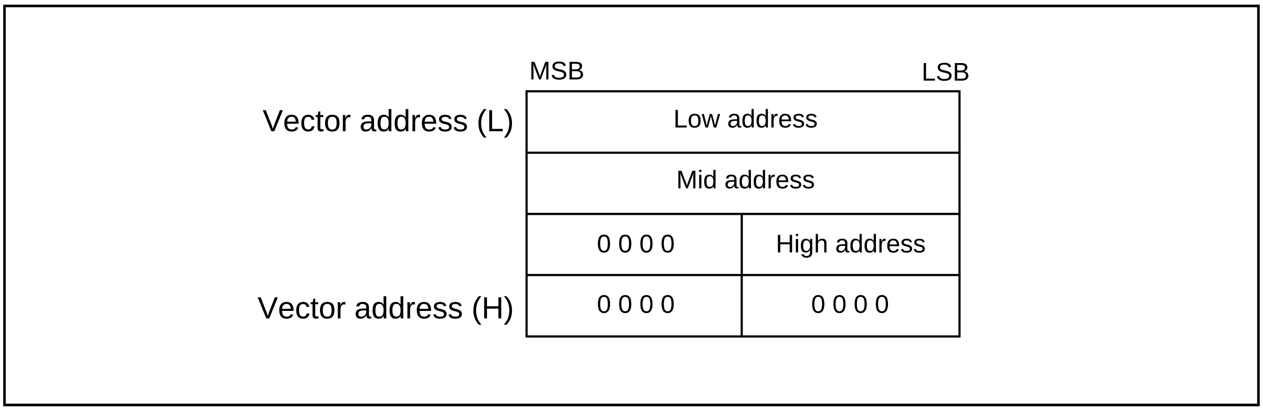


Figure 1.9.2. Interrupt Vector

- **Fixed Vector Tables**

The fixed vector tables are allocated to the addresses from FFFDC$_{16}$ to FFFFF$_{16}$. 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	FFFDC$_{16}$ to FFFDF$_{16}$	Interrupt on UND instruction	M16C/60, M16C/20 serise software maual
Overflow	FFFE0$_{16}$ to FFFE3$_{16}$	Interrupt on INTO instruction	
BRK instruction	FFFE4$_{16}$ to FFFE7$_{16}$	If the contents of address FFFE7$_{16}$ is FF$_{16}$, program execution starts from the address shown by the vector in the relocatable vector table.	
Address match	FFFE8$_{16}$ to FFFEB$_{16}$		Address match interrupt
Single step (Note)	FFFEC$_{16}$ to FFFEF$_{16}$		
Watchdog timer Oscillation stop and re-oscillation detection	FFFF0$_{16}$ to FFFF3$_{16}$		Watchdog timer Clock generating circuit
$\overline{\text{DBC}}$ (Note)	FFFF4$_{16}$ to FFFF7$_{16}$		
—— (Reserved)	FFFF8$_{16}$ to FFFFB$_{16}$		
Reset	FFFFC$_{16}$ to FFFFF$_{16}$		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 reloacatable 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$_{16}$ to 0003$_{16}$)	0	M16C/60, M16C/20 series software manual
——— (Reserved)		1 to 3	
$\overline{\text{INT3}}$	+16 to +19 (0010$_{16}$ to 0013$_{16}$)	4	$\overline{\text{INT}}$ interrupt
——— (Reserved)	+20 to +23 (0014$_{16}$ to 0017$_{16}$)	5	
Timer B4, UART1 bus collision detect (Note 4)	+24 to +27 (0018$_{16}$ to 001B$_{16}$)	6	Timer Serial I/O
Timer B3, UART0 bus collision detect (Note 4)	+28 to +31 (001C$_{16}$ to 001F$_{16}$)	7	
SI/O4, $\overline{\text{INT5}}$ (Note 2)	+32 to +35 (0020$_{16}$ to 0023$_{16}$)	8	$\overline{\text{INT}}$ interrupt Serial I/O
SI/O3, $\overline{\text{INT4}}$ (Note 2)	+36 to +39 (0024$_{16}$ to 0027$_{16}$)	9	
UART 2 bus collision detection	+40 to +43 (0028$_{16}$ to 002B$_{16}$)	10	Serial I/O
DMA0	+44 to +47 (002C$_{16}$ to 002F$_{16}$)	11	DMAC
DMA1	+48 to +51 (0030$_{16}$ to 0033$_{16}$)	12	
——— (Reserved)	+52 to +55 (0034$_{16}$ to 0037$_{16}$)	13	
——— (Reserved)	+56 to +59 (0038$_{16}$ to 003B$_{16}$)	14	
UART2 transmit, NACK2 (Note 3)	+60 to +63 (003C$_{16}$ to 003F$_{16}$)	15	Serial I/O
UART2 receive, ACK2 (Note 3)	+64 to +67 (0040$_{16}$ to 0043$_{16}$)	16	
UART0 transmit, NACK0 (Note 3)	+68 to +71 (0044$_{16}$ to 0047$_{16}$)	17	
UART0 receive, ACK0 (Note 3)	+72 to +75 (0048$_{16}$ to 004B$_{16}$)	18	
UART1 transmit, NACK1(Note 3)	+76 to +79 (004C$_{16}$ to 004F$_{16}$)	19	
UART1 receive, ACK1 (Note 3)	+80 to +83 (0050$_{16}$ to 0053$_{16}$)	20	
Timer A0	+84 to +87 (0054$_{16}$ to 0057$_{16}$)	21	Timer
Timer A1	+88 to +91 (0058$_{16}$ to 005B$_{16}$)	22	
Timer A2	+92 to +95 (005C$_{16}$ to 005F$_{16}$)	23	
Timer A3	+96 to +99 (0060$_{16}$ to 0063$_{16}$)	24	
Timer A4	+100 to +103 (0064$_{16}$ to 0067$_{16}$)	25	
——— (Reserved)	+104 to +107 (0068$_{16}$ to 006B$_{16}$)	26	
——— (Reserved)	+108 to +111 (006C$_{16}$ to 006F$_{16}$)	27	
——— (Reserved)	+112 to +115 (0070$_{16}$ to 0073$_{16}$)	28	
$\overline{\text{INT0}}$	+116 to +119 (0074$_{16}$ to 0077$_{16}$)	29	$\overline{\text{INT}}$ interrupt
$\overline{\text{INT1}}$	+120 to +123 (0078$_{16}$ to 007B$_{16}$)	30	
$\overline{\text{INT2}}$	+124 to +127 (007C$_{16}$ to 007F$_{16}$)	31	
Software interrupt (Note 5)	+128 to +131 (0080$_{16}$ to 0083$_{16}$) to +252 to +255 (00FC$_{16}$ to 00FF$_{16}$)	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^{2}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

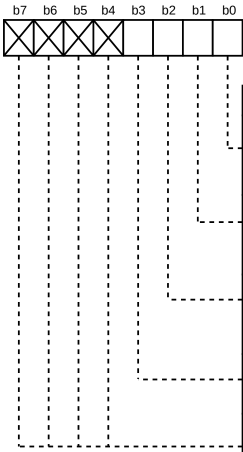
The following describes how to enable/disable the maskable interrupts, and how to set the priority in which order they are accepted. What is explained here does not apply to nonmaskable interrupts.

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.

Interrupt control register (Note 2)

Symbol	Address	After reset
U1BCNIC (Note 3)	0046_{16}	XXXXX000$_2$
U0BCNIC (Note 3)	0047_{16}	XXXXX000$_2$
BCNIC	$004A_{16}$	XXXXX000$_2$
DM0IC, DM1IC	$004B_{16}$, $004C_{16}$	XXXXX000$_2$
S0TIC to S2TIC	0051_{16}, 0053_{16}, $004F_{16}$	XXXXX000$_2$
S0RIC to S2RIC	0052_{16}, 0054_{16}, 0050_{16}	XXXXX000$_2$
TA0IC to TA4IC	0055_{16} to 0059_{16}	XXXXX000$_2$



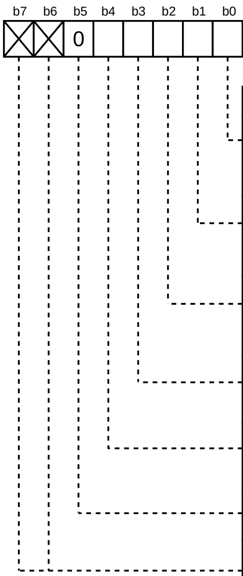
Bit symbol	Bit name	Function	RW
ILVL0	Interrupt priority level select bit	b2 b1 b0 0 0 0 : Level 0 (interrupt disabled) 0 0 1 : Level 1 0 1 0 : Level 2 0 1 1 : Level 3 1 0 0 : Level 4 1 0 1 : Level 5 1 1 0 : Level 6 1 1 1 : Level 7	RW
ILVL1			RW
ILVL2			RW
IR	Interrupt request bit	0 : Interrupt not requested 1 : Interrupt requested	RW (Note 1)
— (b7-b4)	No functions are assigned. When writing to these bits, write “0”. The values in these bits when read are indeterminate.		—

Note 1: This bit can only be reset by writing "0" (Do not write "1").

Note 2: To rewrite the interrupt control registers, do so at a point that does not generate the interrupt request for that register. For details, see the precautions for interrupts.

Note 3: Use the IFSR2A register to select.

Symbol	Address	After reset
INT3IC	0044_{16}	XX00X000$_2$
S4IC	0048_{16}	XX00X000$_2$
S3IC	0049_{16}	XX00X000$_2$
INT0IC to INT2IC	$005D_{16}$ to $005F_{16}$	XX00X000$_2$



Bit symbol	Bit name	Function	RW
ILVL0	Interrupt priority level select bit	b2 b1 b0 0 0 0 : Level 0 (interrupt disabled) 0 0 1 : Level 1 0 1 0 : Level 2 0 1 1 : Level 3 1 0 0 : Level 4 1 0 1 : Level 5 1 1 0 : Level 6 1 1 1 : Level 7	RW
ILVL1			RW
ILVL2			RW
IR	Interrupt request bit	0: Interrupt not requested 1: Interrupt requested	RW (Note 1)
POL	Polarity select bit	0 : Selects falling edge (Notes 3, 4) 1 : Selects rising edge	RW
—	Reserved bit	Must always be set to “0”	RW
— (b7-b6)	No functions are assigned. When writing to these bits, write “0”. The values in these bits when read are indeterminate.		RW

Note 1: This bit can only be reset by writing "0" (Do not write "1").

Note 2: To rewrite the interrupt control register, do so at a point that does not generate the interrupt request for that register. For details, see the precautions for interrupts.

Note 3: If the IFSR register’s IFSRi bit (i = 0 to 5) is "1" (both edges), set the INTiIC register’s POL bit to "0 "(falling edge).

Note 4: Set the S3IC or S4IC register’s POL bit to "0" (falling edge) when the IFSR register’s IFSR6 bit = 0 (SI/O3 selected) or IFSR7 bit = 0 (SI/O4 selected), respectively.

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
000_2	Level 0 (interrupt disabled)	—
001_2	Level 1	Low
010_2	Level 2	
011_2	Level 3	
100_2	Level 4	
101_2	Level 5	
110_2	Level 6	
111_2	Level 7	High

Table 1.9.4. Interrupt Priority Levels Enabled by IPL

IPL	Enabled interrupt priority levels
000_2	Interrupt levels 1 and above are enabled
001_2	Interrupt levels 2 and above are enabled
010_2	Interrupt levels 3 and above are enabled
011_2	Interrupt levels 4 and above are enabled
100_2	Interrupt levels 5 and above are enabled
101_2	Interrupt levels 6 and above are enabled
110_2	Interrupt levels 7 and above are enabled
111_2	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 00000_{16}. 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$^{(Note\ 1)}$.

(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 $^{(Note\ 1)}$ is saved to the stack.

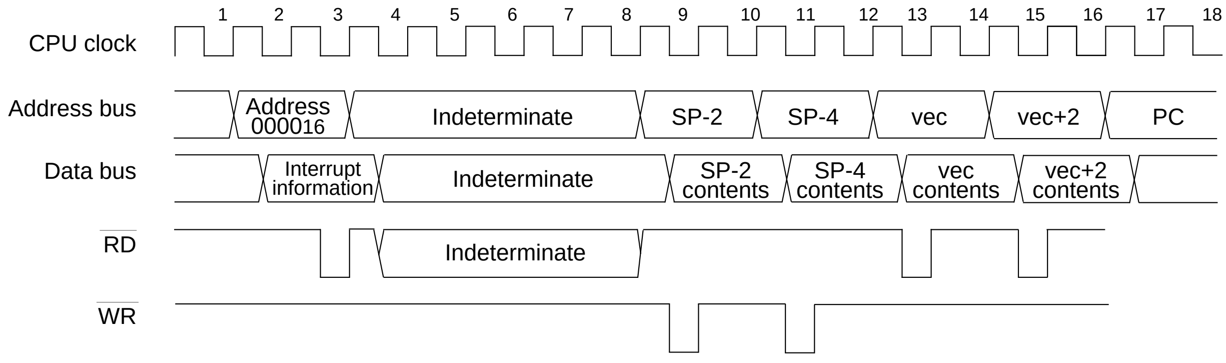
(5) The PC is saved to the stack.

(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.

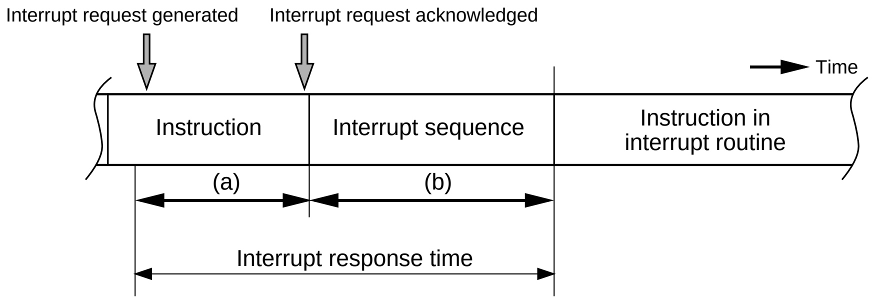


The indeterminate state depends on the instruction queue buffer. A read cycle occurs when the instruction queue buffer is ready to accept instructions.

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).


Interrupt request generated
Interrupt request acknowledged
Time
Instruction
Interrupt sequence
Instruction in interrupt routine
(a)
(b)
Interrupt response time

(a) A time from when an interrupt request is generated till when the instruction then executing is completed. The length of this time varies with the instruction being executed. The DIVX instruction requires the longest time, which is equal to 30 cycles (without wait state, the divisor being a register).

(b) A time during which the interrupt sequence is executed. For details, see the table below. Note, however, that the values in this table must be increased 2 cycles for the $\overline{\text{DBC}}$ interrupt and 1 cycle for the address match and single-step interrupts.

Interrupt vector address	SP value	16-Bit bus, without wait	8-Bit bus, without wait
Even	Even	18 cycles	20 cycles
Even	Odd	19 cycles	20 cycles
Odd	Even	19 cycles	20 cycles
Odd	Odd	20 cycles	20 cycles

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, $\overline{\text{DBC}}$, 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.

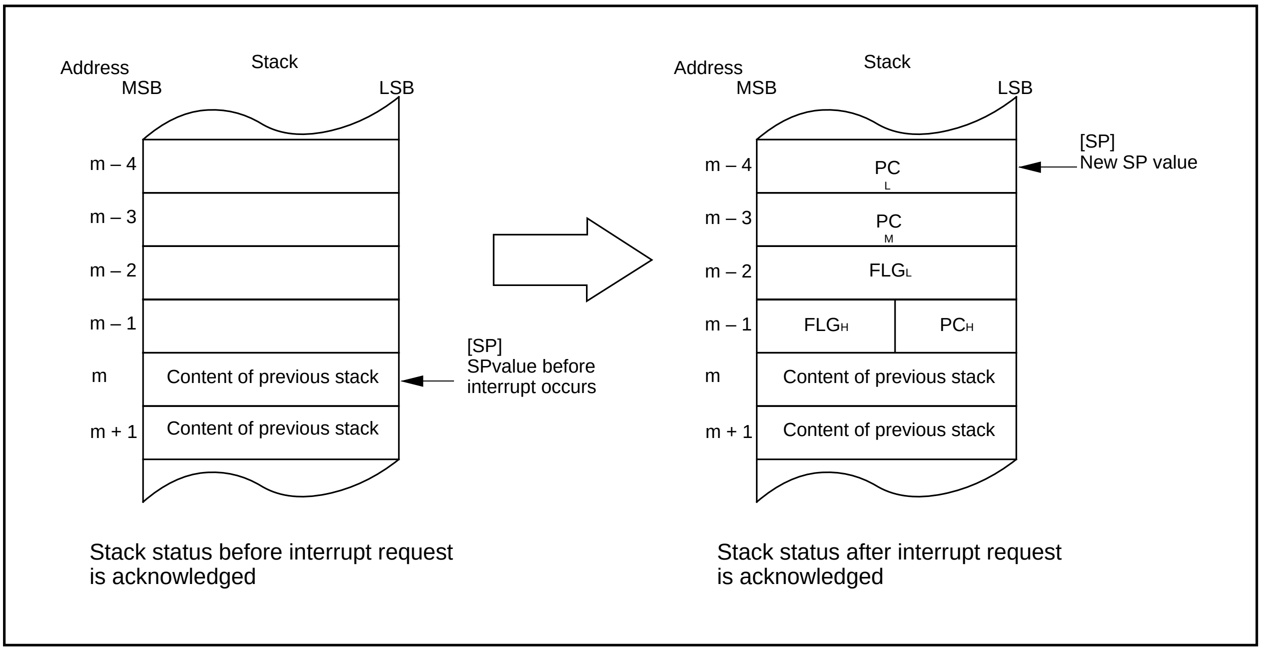
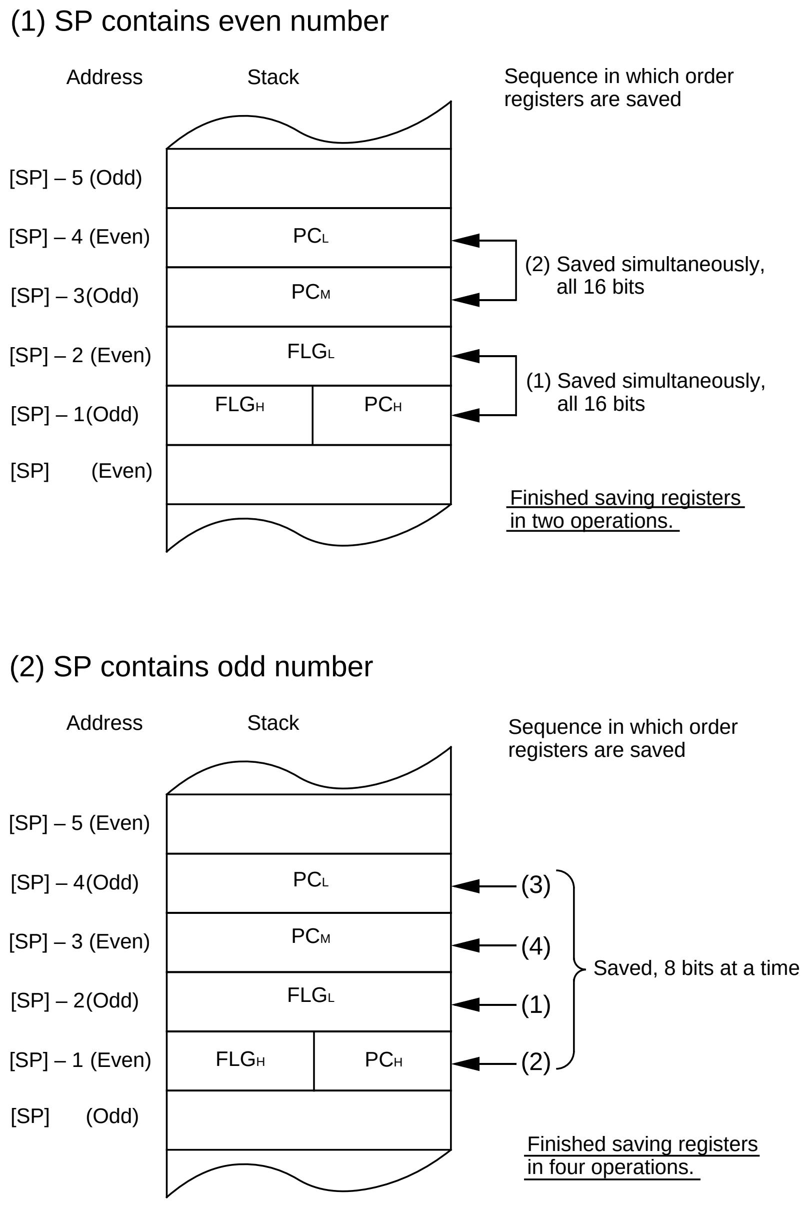

Address
Stack
MSB
LSB
m – 4
m – 3
m – 2
m – 1
m
m + 1
Content of previous stack
Content of previous stack
[SP]
SPvalue before
interrupt occurs
Stack status before interrupt request
is acknowledged
Address
Stack
MSB
LSB
[SP]
New SP value
PC L
PC M
FLGL
FLGH
PCH
Content of previous stack
Content of previous stack
Stack status after interrupt request
is acknowledged

Figure 1.9.6. Stack StatusBefore and After Acceptance of Interrupt Request

The operation of saving registers carried out in the interrupt sequence is dependent on whether the SP(Note), at the time of acceptance of an interrupt request, is even or odd. If the stack pointer (Note) 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.



Note: [SP] denotes the initial value of the SP when interrupt request is acknowledged.
After registers are saved, the SP content is [SP] minus 4.

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.

Reset > $\overline{\text{DBC}}$ > WDT > Peripheral function > Single step > Address match

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.

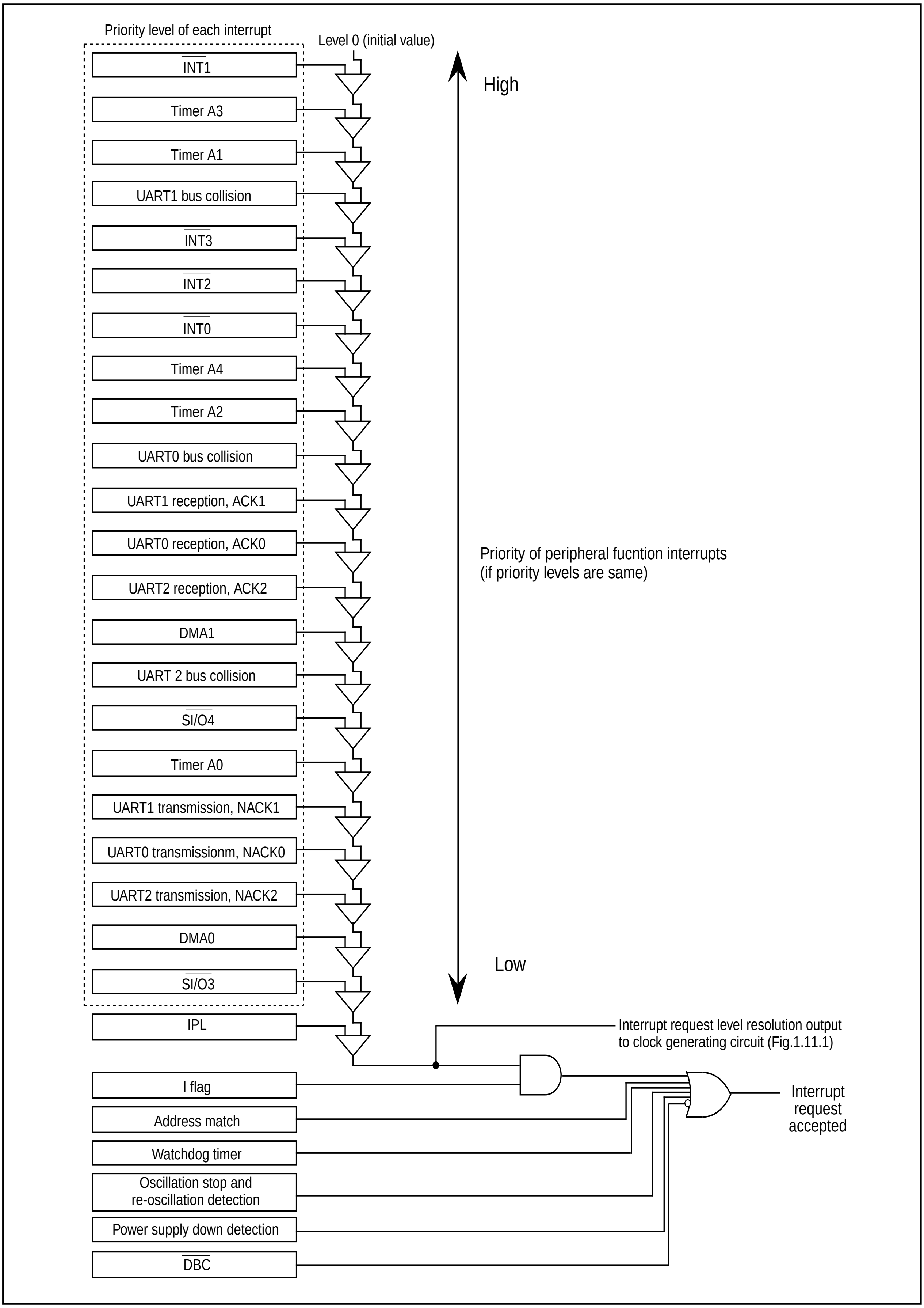
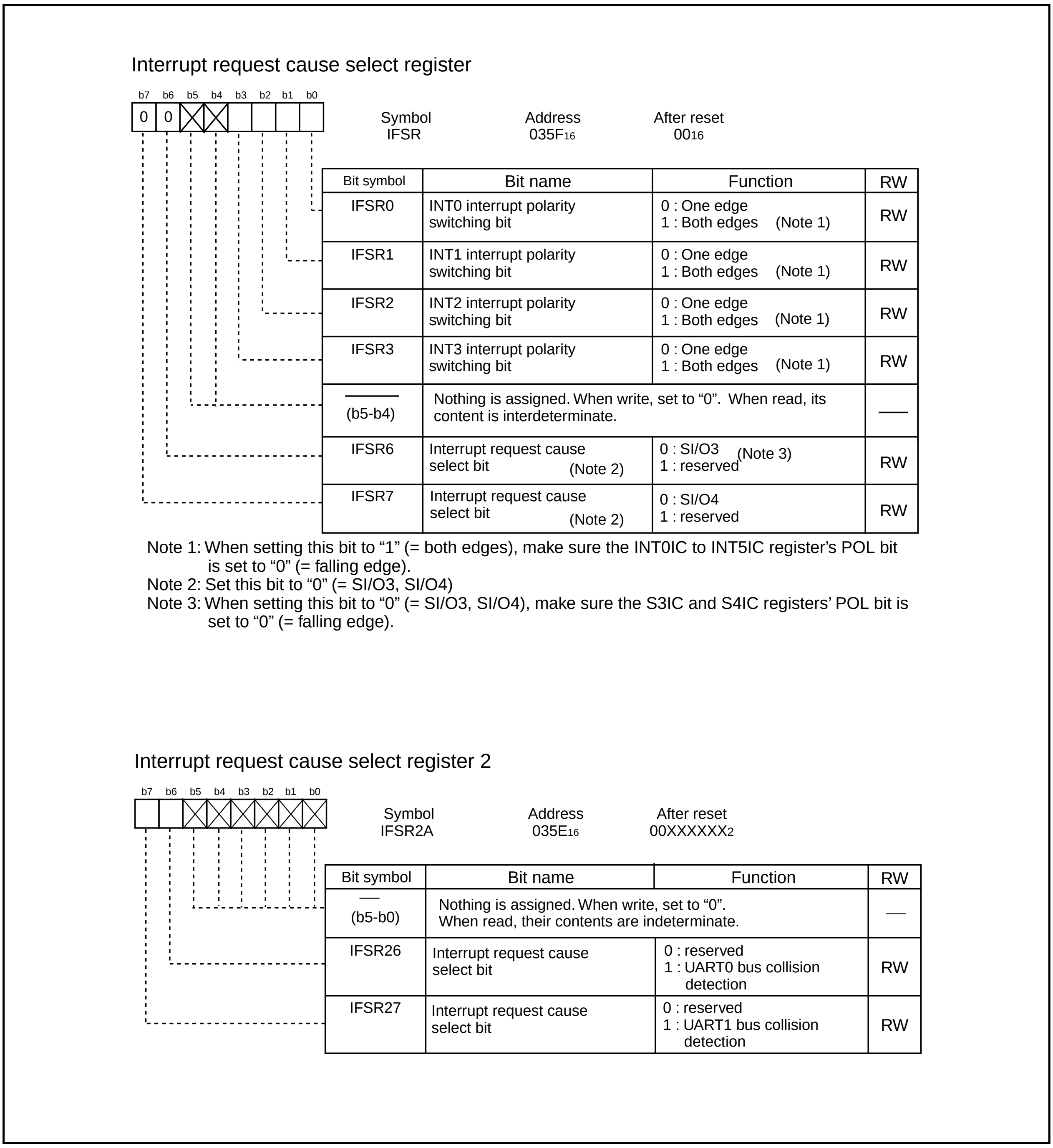

Priority level of each interrupt
Level 0 (initial value)
INT1
Timer A3
Timer A1
UART1 bus collision
INT3
INT2
INT0
Timer A4
Timer A2
UART0 bus collision
UART1 reception, ACK1
UART0 reception, ACK0
UART2 reception, ACK2
DMA1
UART 2 bus collision
SI/O4
Timer A0
UART1 transmission, NACK1
UART0 transmissionm, NACK0
UART2 transmission, NACK2
DMA0
SI/O3
IPL
I flag
Address match
Watchdog timer
Oscillation stop and re-oscillation detection
Power supply down detection
DBC
High
Low
Priority of peripheral fucntion interrupts (if priority levels are same)
Interrupt request level resolution output to clock generating circuit (Fig.1.11.1)
Interrupt request accepted

Figure 1.9.9. Interrupts Priority Select Circuit

$\overline{\text{INT}}$ Interrupt

$\overline{\text{INTi}}$ interrupt (i=0 to 3) is triggered by the edges of external inputs. The edge polarity is selected using the IFSR register's IFSRi bit.

Figure 1.9.10 shows the IFSR and IFSR2A registers.



Interrupt request cause select register

Symbol	Address	After reset
IFSR	035F$_{16}$	00$_{16}$

Bit symbol	Bit name	Function	RW
IFSR0	INT0 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR1	INT1 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR2	INT2 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR3	INT3 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
—— (b5-b4)	Nothing is assigned. When write, set to “0”. When read, its content is interdeterminate.		——
IFSR6	Interrupt request cause select bit (Note 2)	0 : SI/O3 (Note 3) 1 : reserved	RW
IFSR7	Interrupt request cause select bit (Note 2)	0 : SI/O4 1 : reserved	RW

Note 1: When setting this bit to “1” (= both edges), make sure the INT0IC to INT5IC register’s POL bit is set to “0” (= falling edge).
Note 2: Set this bit to “0” (= SI/O3, SI/O4)
Note 3: When setting this bit to “0” (= SI/O3, SI/O4), make sure the S3IC and S4IC registers’ POL bit is set to “0” (= falling edge).

Interrupt request cause select register 2

Symbol	Address	After reset
IFSR2A	035E$_{16}$	00XXXXXX$_2$

Bit symbol	Bit name	Function	RW
—— (b5-b0)	Nothing is assigned. When write, set to “0”. When read, their contents are indeterminate.		——
IFSR26	Interrupt request cause select bit	0 : reserved 1 : UART0 bus collision detection	RW
IFSR27	Interrupt request cause select bit	0 : reserved 1 : UART1 bus collision detection	RW

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. Figure 1.9.11 shows the instruction just before execution and address stored in the stack when there occurs interruption.

Note that when using the external data bus in width of 8 bits, the address match interrupt cannot be used for external area.

Figure 1.9.12 shows the AIER, AIER2, and RMAD0 to RMAD3 registers.

(1) Instructions in which the "return destination + 2" address is stored in the stack when address match interrupt occurs

- 16-bit operation code
- Instruction shown below among 8-bit operation code instructions

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)				

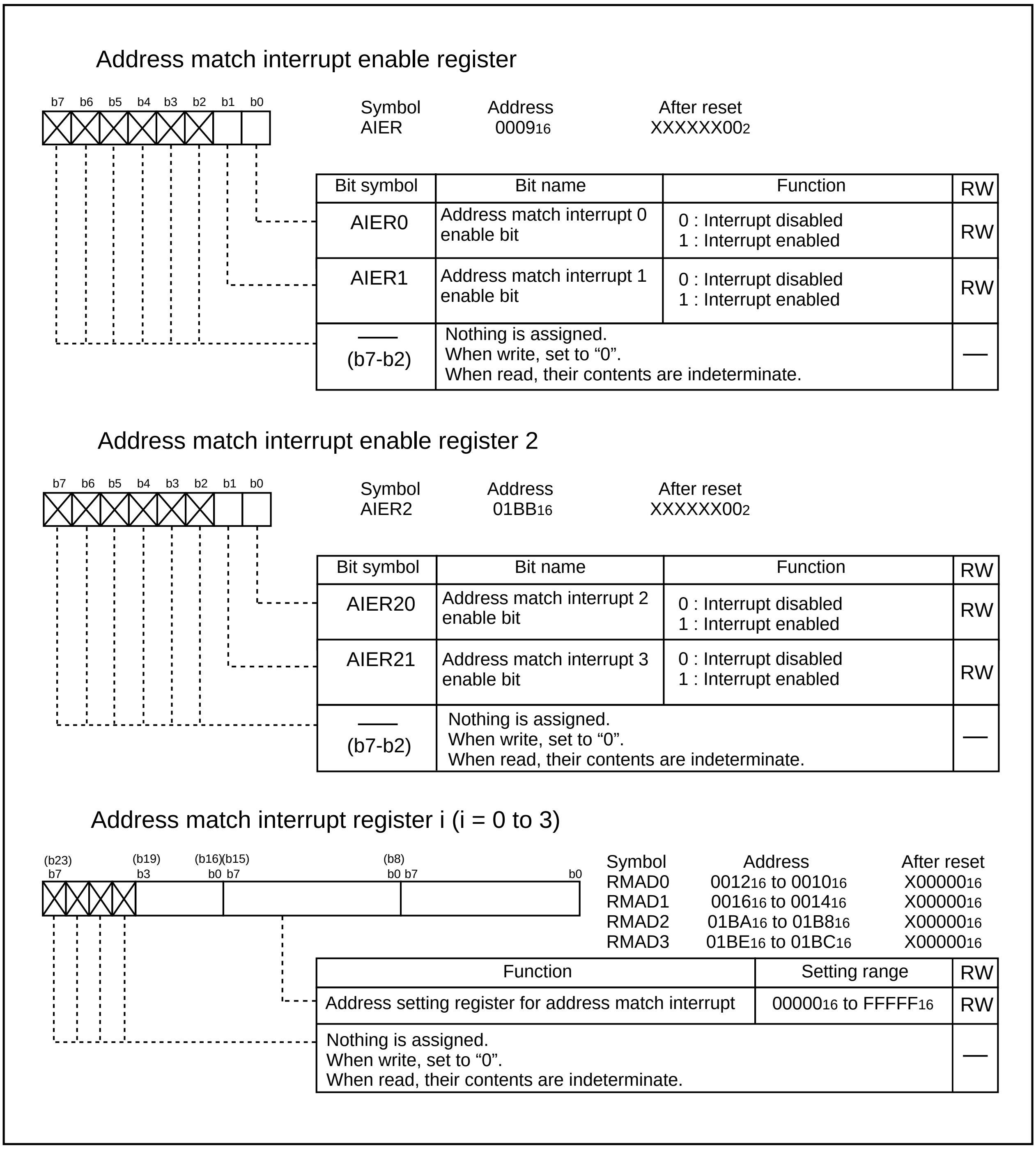
(2) Instructions in which the "return destination + 1" address is stored in the stack when address match interrupt occurs

- Instructions other than the above

Figure 1.9.11. Instruction Just Before Execution and Address Stored in Stack When There Occurs Interrupts

Table 1.9.6. 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



Address match interrupt enable register

Symbol	Address	After reset
AIER	0009_{16}	$XXXXXX00_2$

Bit symbol	Bit name	Function	RW
AIER0	Address match interrupt 0 enable bit	0 : Interrupt disabled 1 : Interrupt enabled	RW
AIER1	Address match interrupt 1 enable bit	0 : Interrupt disabled 1 : Interrupt enabled	RW
—— (b7-b2)	Nothing is assigned. When write, set to “0”. When read, their contents are indeterminate.		—

Address match interrupt enable register 2

Symbol	Address	After reset
AIER2	$01BB_{16}$	$XXXXXX00_2$

Bit symbol	Bit name	Function	RW
AIER20	Address match interrupt 2 enable bit	0 : Interrupt disabled 1 : Interrupt enabled	RW
AIER21	Address match interrupt 3 enable bit	0 : Interrupt disabled 1 : Interrupt enabled	RW
—— (b7-b2)	Nothing is assigned. When write, set to “0”. When read, their contents are indeterminate.		—

Address match interrupt register i (i = 0 to 3)

Symbol	Address	After reset
RMAD0	0012_{16} to 0010_{16}	$X00000_{16}$
RMAD1	0016_{16} to 0014_{16}	$X00000_{16}$
RMAD2	$01BA_{16}$ to $01B8_{16}$	$X00000_{16}$
RMAD3	$01BE_{16}$ to $01BC_{16}$	$X00000_{16}$

Function	Setting range	RW
Address setting register for address match interrupt	00000_{16} to $FFFFF_{16}$	RW
Nothing is assigned. When write, set to “0”. When read, their contents are indeterminate.		—

Figure 1.9.12. AIER Register, AIER2 Register and RMAD0 to RMAD3 Registers

Precautions for Interrupts

(1) Reading Address 00000$_{16}$

- Do not read the address 00000$_{16}$ 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$_{16}$ during the interrupt sequence. At this time, the IR bit for the accepted interrupt is cleared to “0”. If the address 00000$_{16}$ 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

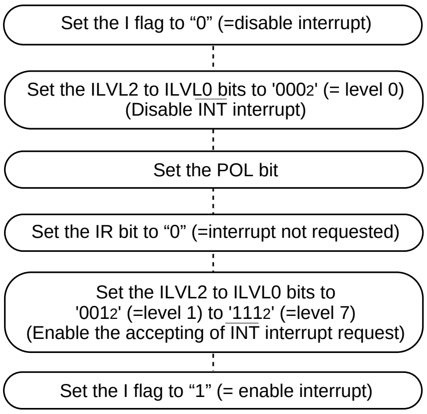
- Set any value in the SP before accepting an interrupt. The SP is cleared to ‘0000$_{16}$’ after reset. Therefore, if an interrupt is accepted before setting any value in the SP, the program may go out of control.

(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.


Set the I flag to “0” (=disable interrupt)
Set the ILVL2 to ILVL0 bits to '000$_2$' (= level 0)
(Disable $\overline{\text{INT}}$ interrupt)
Set the POL bit
Set the IR bit to “0” (=interrupt not requested)
Set the ILVL2 to ILVL0 bits to
'001$_2$' (=level 1) to '111$_2$' (=level 7)
(Enable the accepting of $\overline{\text{INT}}$ interrupt request)
Set the I flag to “1” (= enable interrupt)

Note: Execute the setting above individually. Do not execute two or more settings at once (by one instruction).

Figure 1.9.13. Switching Procedure for $\overline{\text{INT}}$ Interrupt Request

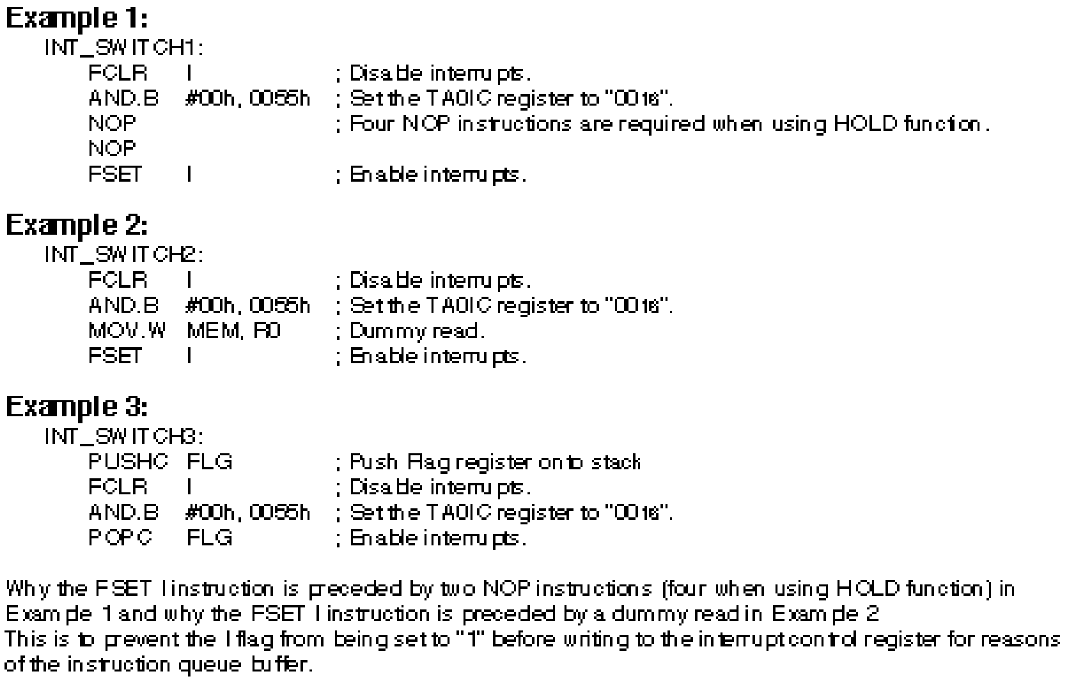
(5) Modifying Interrupt Control Register

- 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 onto 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. If a sub-clock is selected for CPU clock, the divide-by-N value for the prescaler is always 2 no matter how the WDC7 bit is set. 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

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

With sub-clock chosen for CPU clock

$$\text{Watchdog timer period} = \frac{\text{Prescaler dividing (2) X Watchdog timer count (32768)}}{\text{CPU clock}}$$

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, wait mode and hold state, 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 ring 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” (ring 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 ring oscillator starts oscillating, and the ring oscillator clock becomes the watchdog timer count source.
- 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.

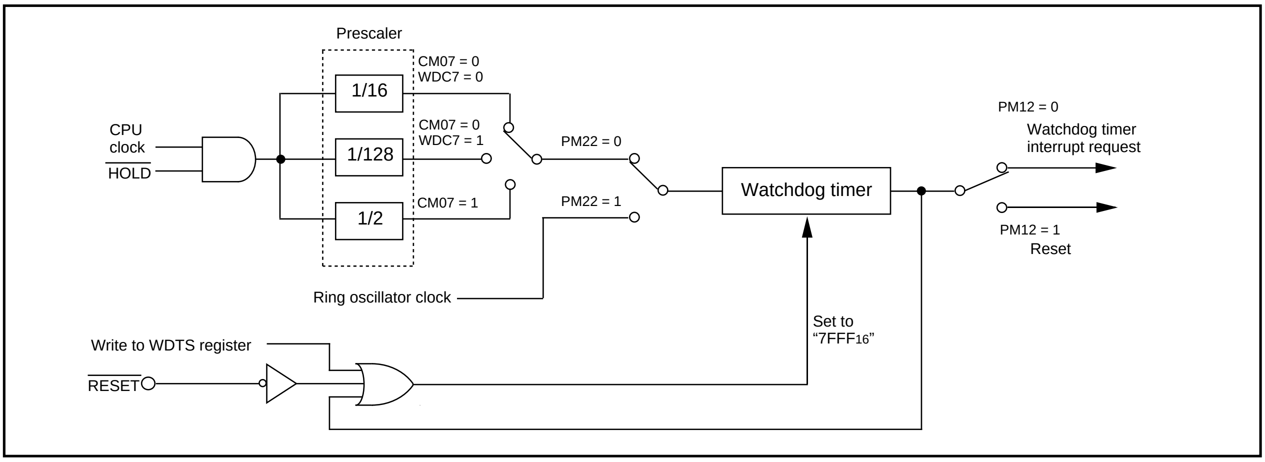


Figure 1.10.1. Watchdog Timer Block Diagram

Watchdog timer control register

b7	b6	b5	b4	b3	b2	b1	b0
	0	0					

Symbol□ WDC□ Address□ 000F$_{16}$□ After reset□ 00XXXXXX$_2$ □

Bit symbol	Bit name	Function	RW
(b4-b0)	High-order bit of watchdog timer		RO
(b6-b5)	Reserved bit	Must set to “0”	RW
WDC7	Prescaler select bit	0 : Divided by 16 1 : Divided by 128	RW

Watchdog timer start register (Note)

b7 … b0

Symbol WDTS Address 000E$_{16}$ After reset Indeterminate

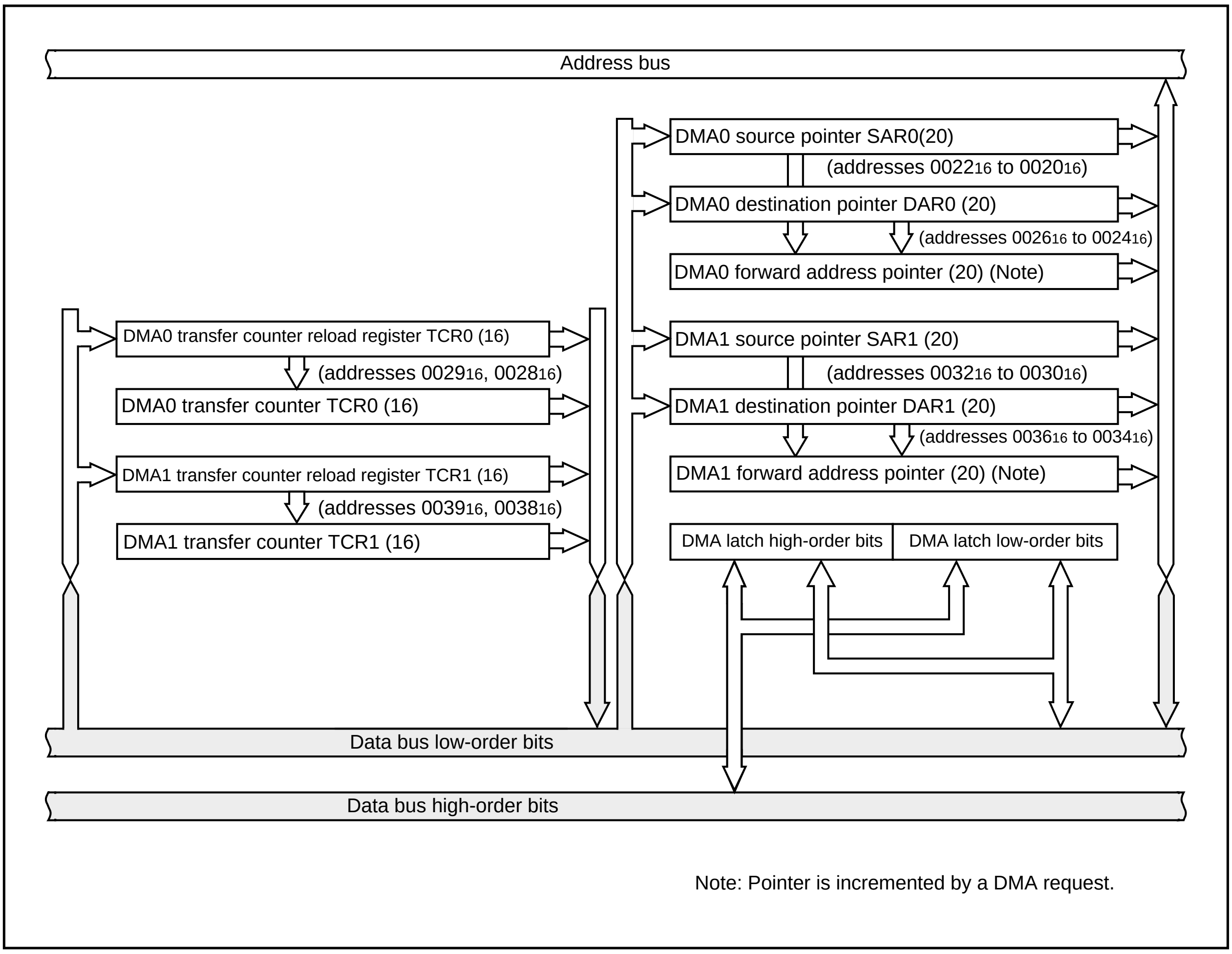
Function	RW
The watchdog timer is initialized and starts counting after a write instruction to this register. The watchdog timer value is always initialized to “7FFF$_{16}$” regardless of whatever value is written.	WO

Note : Write to the WDTS register after the watchdog timer interrupt occurs.

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.


Address bus
DMA0 source pointer SAR0(20)
(addresses 0022_{16} to 0020_{16})
DMA0 destination pointer DAR0 (20)
(addresses 0026_{16} to 0024_{16})
DMA0 forward address pointer (20) (Note)
DMA1 source pointer SAR1 (20)
(addresses 0032_{16} to 0030_{16})
DMA1 destination pointer DAR1 (20)
(addresses 0036_{16} to 0034_{16})
DMA1 forward address pointer (20) (Note)
DMA latch high-order bits
DMA latch low-order bits
DMA0 transfer counter reload register TCR0 (16)
(addresses 0029_{16}, 0028_{16})
DMA0 transfer counter TCR0 (16)
DMA1 transfer counter reload register TCR1 (16)
(addresses 0039_{16}, 0038_{16})
DMA1 transfer counter TCR1 (16)
Data bus low-order bits
Data bus high-order bits

Note: Pointer is incremented by a DMA request.

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”.

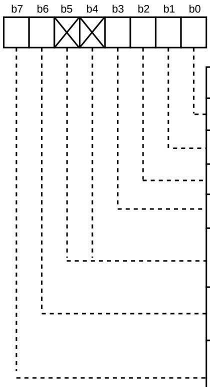
Table 1.11.1. DMAC Specifications

Item		Specification
No. of channels		2 (cycle steal method)
Transfer memory space		• From any address in the 1M bytes space to a fixed address • From a fixed address to any address in the 1M bytes space • From a fixed address to a fixed address
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 $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ Both edge of $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ 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 interrpt 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 DMAi transfer counter (i = 0–1) underflows after reaching the terminal count.
	•Repeat transfer	When the DMAi transfer counter underflows, it is reloaded with the value of the DMAi transfer counter reload register and a DMA transfer is continued with it.
DMA interrupt request generation timing		When the DMAi transfer counter underflowed
DMA startup		Data transfer is initiated each time a DMA request is generated when the DMAiCON register's DMAE bit = "1" (enabled).
DMA shutdown	•Single transfer	• When the DMAE bit is set to "0" (disabled) • After the DMAi transfer counter underflows
	•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 SARi or the DARi pointer whichever is specified to be in the forward direction and the DMAi transfer counter is reloaded with the value of the DMAi 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_{16}–$003F_{16}$) are accessed by the DMAC.

DMA0 request cause select register



Symbol	Address	After reset
DM0SL	$03B8_{16}$	00_{16}

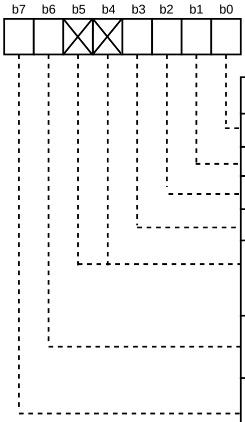
Bit symbol	Bit name	Function	RW
DSEL0	DMA request cause select bit	Refer to note	RW
DSEL1			RW
DSEL2			RW
DSEL3			RW
— (b5-b4)	Nothing is assigned. When write, set to “0”. When read, its content is “0”.		—
DMS	DMA request cause expansion select bit	0: Basic cause of request 1: Extended cause of request	RW
DSR	Software DMA request bit	A DMA request is generated by setting this bit to “1” when the DMS bit is “0” (basic cause) and the DSEL3 to DSEL0 bits are “0001_2” (software trigger). The value of this bit when read is “0” .	RW

Note: The causes of DMA0 requests can be selected by a combination of DMS bit and DSEL3 to DSEL0 bits in the manner described below.

DSEL3 to DSEL0	DMS=0(basic cause of request)	DMS=1(extended cause of request)
$0\,0\,0\,0_2$	Falling edge of $\overline{\text{INT0}}$ pin	–
$0\,0\,0\,1_2$	Software trigger	–
$0\,0\,1\,0_2$	Timer A0	–
$0\,0\,1\,1_2$	Timer A1	–
$0\,1\,0\,0_2$	Timer A2	–
$0\,1\,0\,1_2$	Timer A3	–
$0\,1\,1\,0_2$	Timer A4	Two edges of $\overline{\text{INT0}}$ pin
$0\,1\,1\,1_2$	–	–
$1\,0\,0\,0_2$	–	–
$1\,0\,0\,1_2$	–	–
$1\,0\,1\,0_2$	UART0 transmit	–
$1\,0\,1\,1_2$	UART0 receive	–
$1\,1\,0\,0_2$	UART2 transmit	–
$1\,1\,0\,1_2$	UART2 receive	–
$1\,1\,1\,0_2$	–	–
$1\,1\,1\,1_2$	UART1 transmit	–

Figure 1.11.2. DM0SL Register

DMA1 request cause select register



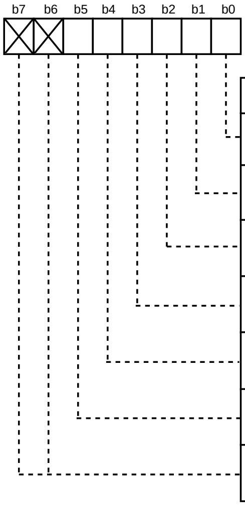
Symbol	Address	After reset
DM1SL	03BA$_{16}$	00$_{16}$

Bit symbol	Bit name	Function	RW
DSEL0	DMA request cause select bit	Refer to note	RW
DSEL1			RW
DSEL2			RW
DSEL3			RW
—— (b5-b4)	Nothing is assigned. When write, set to “0”. When read, its content is “0”.		——
DMS	DMA request cause expansion select bit	0: Basic cause of request 1: Extended cause of request	RW
DSR	Software DMA request bit	A DMA request is generated by setting this bit to “1” when the DMS bit is “0” (basic cause) and the DSEL3 to DSEL0 bits are “0001$_2$” (software trigger). The value of this bit when read is “0” .	RW

Note: The causes of DMA1 requests can be selected by a combination of DMS bit and DSEL3 to DSEL0 bits in the manner described below.

DSEL3 to DSEL0	DMS=0(basic cause of request)	DMS=1(extended cause of request)
0 0 0 0$_2$	Falling edge of $\overline{\text{INT1}}$ pin	–
0 0 0 1$_2$	Software trigger	–
0 0 1 0$_2$	Timer A0	–
0 0 1 1$_2$	Timer A1	–
0 1 0 0$_2$	Timer A2	–
0 1 0 1$_2$	Timer A3	SI/O3
0 1 1 0$_2$	Timer A4	SI/O4
0 1 1 1$_2$	–	Two edges of $\overline{\text{INT1}}$
1 0 0 0$_2$	–	–
1 0 0 1$_2$	–	–
1 0 1 0$_2$	UART0 transmit	–
1 0 1 1$_2$	UART0 receive/ACK0	–
1 1 0 0$_2$	UART2 transmit	–
1 1 0 1$_2$	UART2 receive/ACK2	–
1 1 1 0$_2$	–	–
1 1 1 1$_2$	UART1 receive/ACK1	–

DMAi control register(i=0,1)



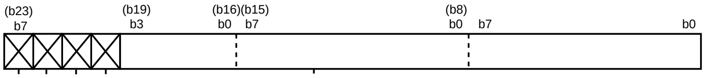
Symbol	Address	After reset
DM0CON	002C$_{16}$	00000X00$_2$
DM1CON	003C$_{16}$	00000X00$_2$

Bit symbol	Bit name	Function	RW
DMBIT	Transfer unit bit select bit	0 : 16 bits 1 : 8 bits	RW
DMASL	Repeat transfer mode select bit	0 : Single transfer 1 : Repeat transfer	RW
DMAS	DMA request bit	0 : DMA not requested 1 : DMA requested	RW (Note 1)
DMAE	DMA enable bit	0 : Disabled 1 : Enabled	RW
DSD	Source address direction select bit (Note 2)	0 : Fixed 1 : Forward	RW
DAD	Destination address direction select bit (Note 2)	0 : Fixed 1 : Forward	RW
—— (b7-b6)	Nothing is assigned. When write, set to “0”. When read, its content is “0”.		——

Note 1: The DMAS bit can be set to “0” by writing “0” in a program (This bit remains unchanged even if “1” is written).
Note 2: At least one of the DAD and DSD bits must be “0” (address direction fixed).

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

DMAi source pointer (i = 0, 1) (Note)

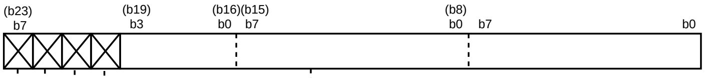


Symbol	Address	After reset
SAR0	0022_{16} to 0020_{16}	Indeterminate
SAR1	0032_{16} to 0030_{16}	Indeterminate

Function	Setting range	RW
Set the source address of transfer	00000_{16} to $FFFFF_{16}$	RW
Nothing is assigned. When write, set "0". When read, these contents are "0".		—

Note: If the DSD bit of DMiCON register is "0" (fixed), this register can only be written to when the DMAE bit of DMiCON register is "0" (DMA disabled).
If the DSD bit is "1" (forward direction), this register can be written to at any time.
If the DSD bit is "1" and the DMAE bit is "1" (DMA enabled), the DMAi forward address pointer can be read from this register. Otherwise, the value written to it can be read.

DMAi destination pointer (i = 0, 1)(Note)

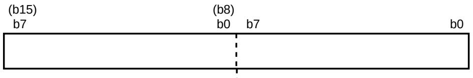


Symbol	Address	After reset
DAR0	0026_{16} to 0024_{16}	Indeterminate
DAR1	0036_{16} to 0034_{16}	Indeterminate

Function	Setting range	RW
Set the destination address of transfer	00000_{16} to $FFFFF_{16}$	RW
Nothing is assigned. When write, set "0". When read, these contents are "0".		—

Note: If the DAD bit of DMiCON register is "0" (fixed), this register can only be written to when the DMAE bit of DMiCON register is "0"(DMA disabled).
If the DAD bit is "1" (forward direction), this register can be written to at any time.
If the DAD bit is "1" and the DMAE bit is "1" (DMA enabled), the DMAi forward address pointer can be read from this register. Otherwise, the value written to it can be read.

DMAi transfer counter (i = 0, 1)



Symbol	Address	After reset
TCR0	0029_{16}, 0028_{16}	Indeterminate
TCR1	0039_{16}, 0038_{16}	Indeterminate

Function	Setting range	RW
Set the transfer count minus 1. The written value is stored in the DMAi transfer counter reload register, and when the DMAE bit of DMiCON register is set to "1" (DMA enabled) or the DMAi transfer counter underflows when the DMASL bit of DMiCON register is "1" (repeat transfer), the value of the DMAi transfer counter reload register is transferred to the DMAi transfer counter. When read, the DMAi transfer counter is read.	0000_{16} to $FFFF_{16}$	RW

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 $\overline{\text{RDY}}$ signal.

(a) Effect of Source and Destination Addresses

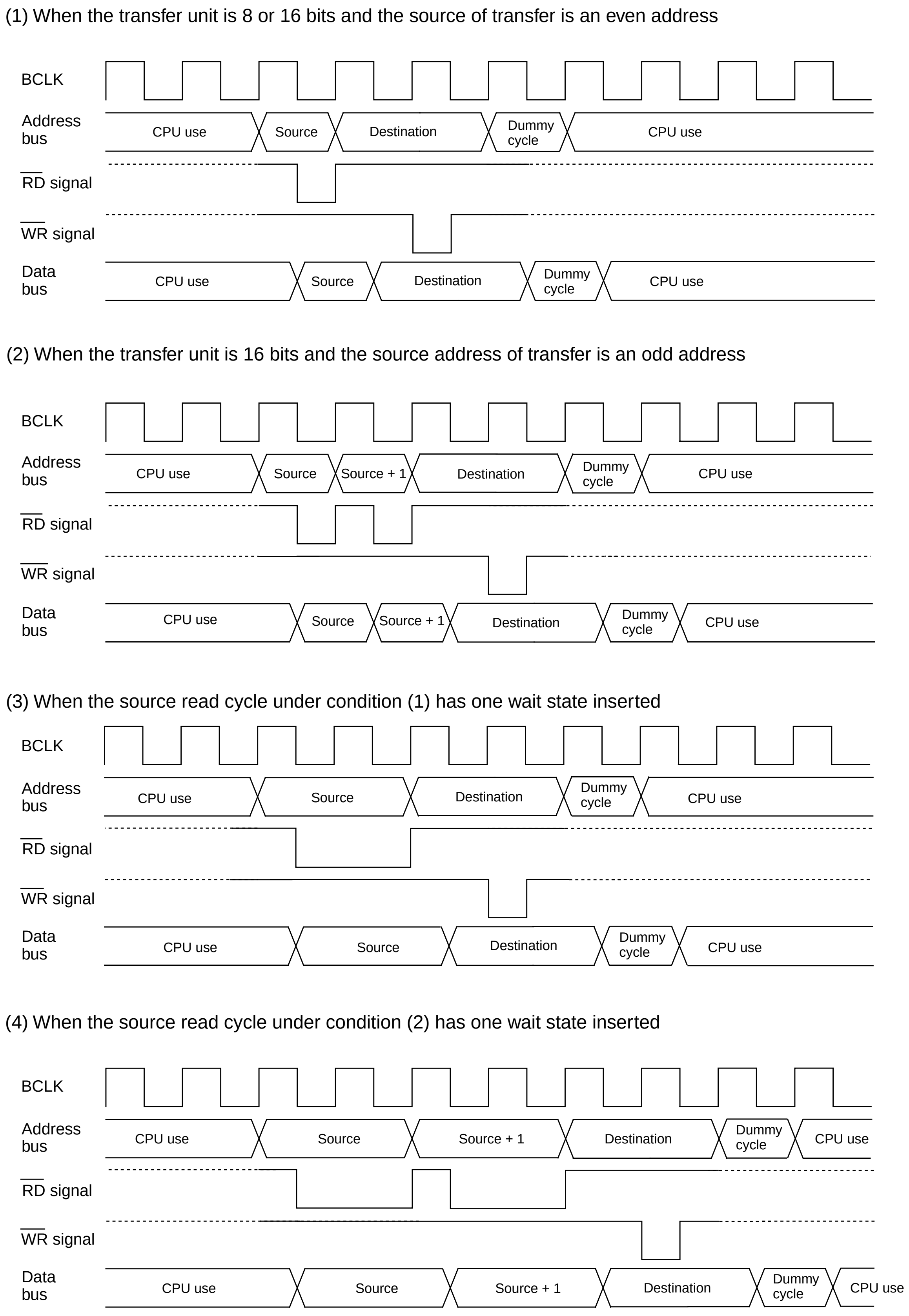
If the transfer unit and data bus both are 16 bits and the source address of transfer begins with an odd address, the source read cycle consists of one more bus cycle than when the source address of transfer begins with an even address.

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.



(1) When the transfer unit is 8 or 16 bits and the source of transfer is an even address

(2) When the transfer unit is 16 bits and the source address of transfer is an odd address

(3) When the source read cycle under condition (1) has one wait state inserted

(4) When the source read cycle under condition (2) has one wait state inserted

Note: The same timing changes occur with the respective conditions at the destination as at the source.

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:

No. of transfer cycles per transfer unit = No. of read cycles x j + No. of write cycles x 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[2]	2-wait[2]
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 1.13.4 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”	• Immediately before a data transfer starts • When set by writing “0” in a program
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.

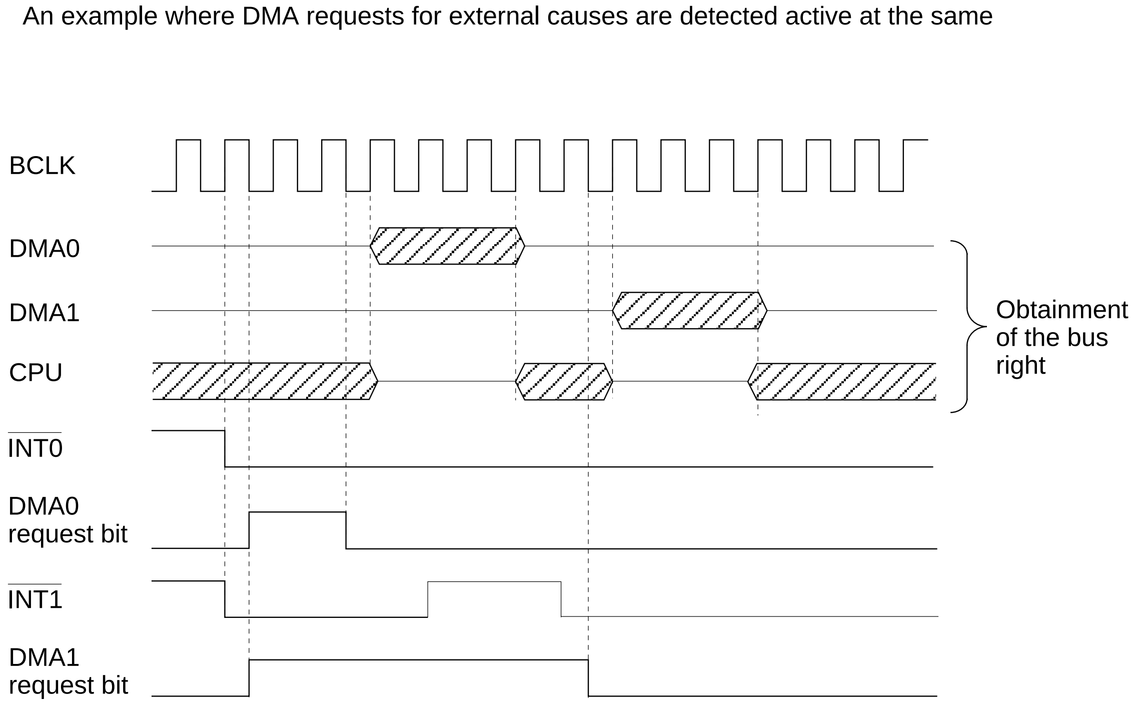
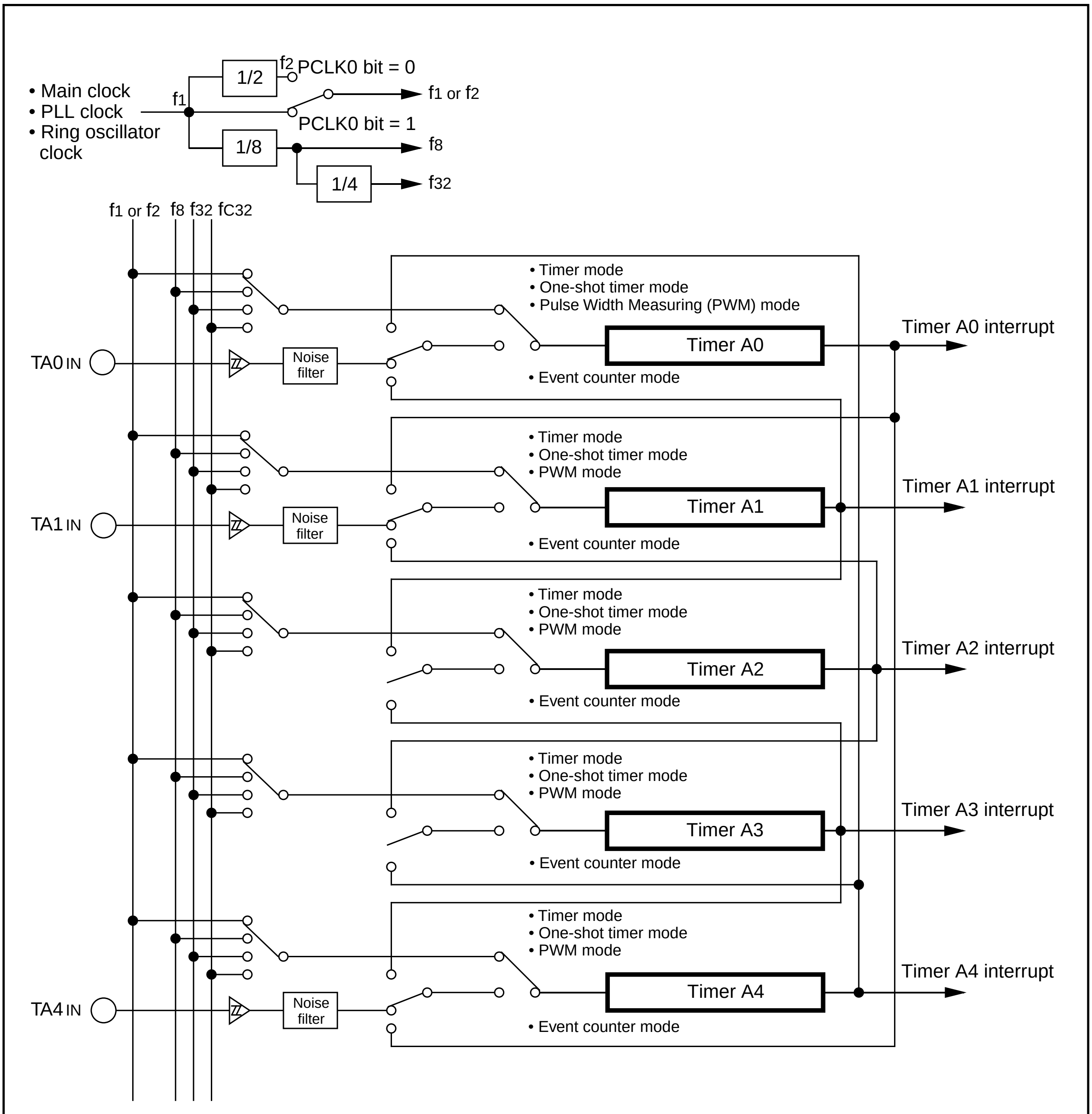

An example where DMA requests for external causes are detected active at the same
BCLK
DMA0
DMA1
CPU
Obtainment of the bus right
$\overline{\text{INT0}}$
DMA0 request bit
$\overline{\text{INT1}}$
DMA1 request bit

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.



Note: Be aware that TA0 IN shares the pin with RxD2.

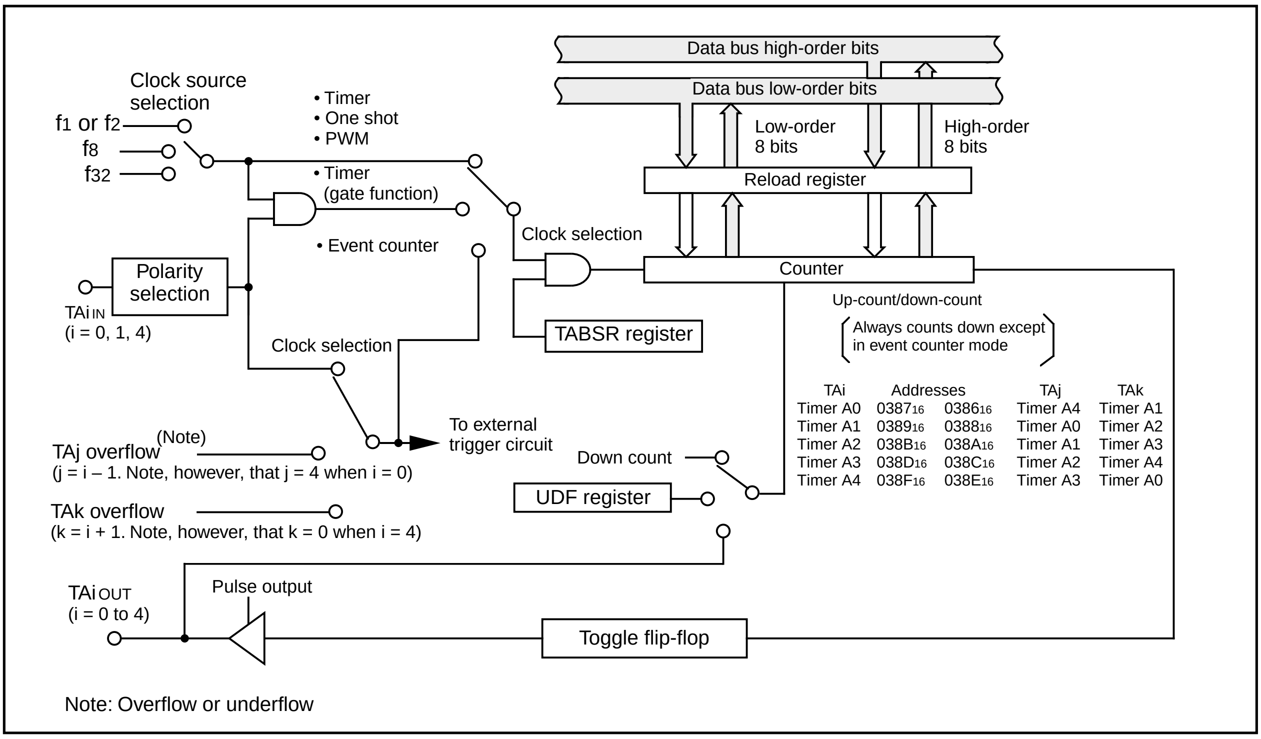
Figure 1.12.1. Timer A Configuration

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.

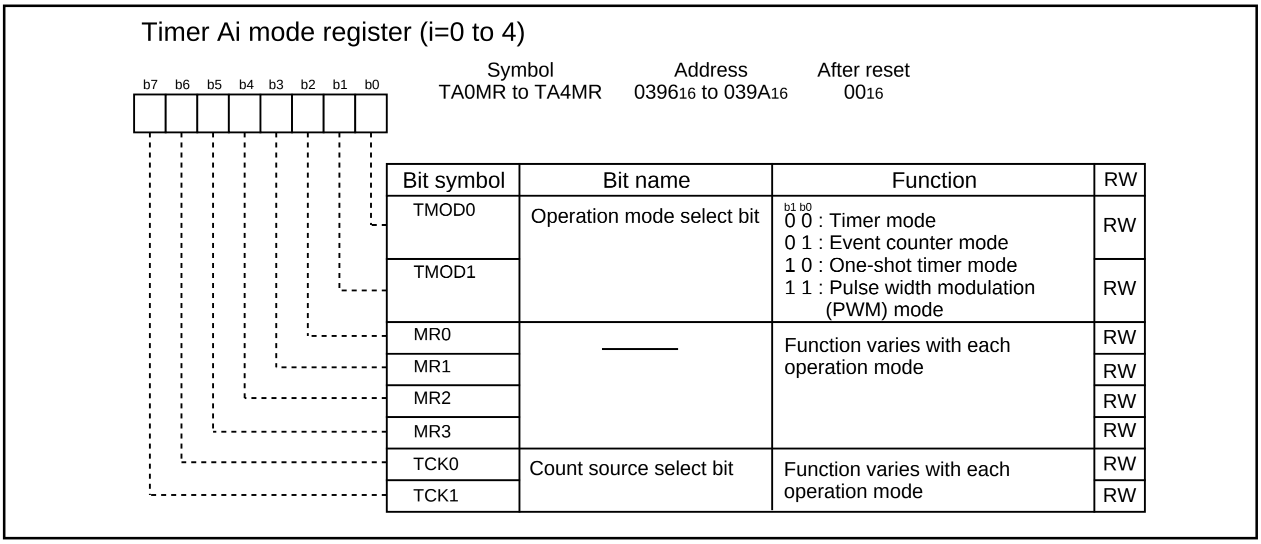
The timer A supports the following four modes. Except in event counter mode, timers A0 to A4 all have the same function. Use the TMOD1 to TMOD0 bits of TAiMR register (i = 0 to 4) to select the desired mode.

- 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$_{16}$."
- Pulse width modulation (PWM) mode: The timer outputs pulses in a given width successively.



TAi	Addresses		TAj	TAk
Timer A0	0387$_{16}$	0386$_{16}$	Timer A4	Timer A1
Timer A1	0389$_{16}$	0388$_{16}$	Timer A0	Timer A2
Timer A2	038B$_{16}$	038A$_{16}$	Timer A1	Timer A3
Timer A3	038D$_{16}$	038C$_{16}$	Timer A2	Timer A4
Timer A4	038F$_{16}$	038E$_{16}$	Timer A3	Timer A0

Figure 1.12.2. Timer A Block Diagram


b7 b6 b5 b4 b3 b2 b1 b0

Timer Ai mode register (i=0 to 4)

Symbol	Address	After reset
TA0MR to TA4MR	0396$_{16}$ to 039A$_{16}$	00$_{16}$

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 0 : Timer mode 0 1 : Event counter mode 1 0 : One-shot timer mode 1 1 : Pulse width modulation (PWM) mode	RW
TMOD1			RW
MR0	———	Function varies with each operation mode	RW
MR1			RW
MR2			RW
MR3			RW
TCK0	Count source select bit	Function varies with each operation mode	RW
TCK1			RW

Figure 1.12.3. TA0MR to TA4MR Registers

Timer Ai register (i= 0 to 4) (Note 1)

(b15) b7 (b8) b0 b7 b0

Symbol	Address	After reset
TA0	0387_{16}, 0386_{16}	Indeterminate
TA1	0389_{16}, 0388_{16}	Indeterminate
TA2	$038B_{16}$, $038A_{16}$	Indeterminate
TA3	$038D_{16}$, $038C_{16}$	Indeterminate
TA4	$038F_{16}$, $038E_{16}$	Indeterminate

Mode	Function	Setting range	RW
Timer mode	Divide the count source by n + 1 where n = set value	0000_{16} to $FFFF_{16}$	RW
Event counter mode	Divide the count source by $FFFF_{16}$ – n + 1 where n = set value when counting up or by n + 1 when counting down (Note 5)	0000_{16} to $FFFF_{16}$	RW
One-shot timer mode	Divide the count source by n where n = set value and cause the timer to stop	0000_{16} to $FFFF_{16}$ (Notes 2, 4)	WO
Pulse width modulation mode (16-bit PWM)	Modify the pulse width as follows: PWM period: (2^{16} – 1) / fj High level PWM pulse width: n / fj where n = set value, fj = count source frequency	0000_{16} to $FFFE_{16}$ (Note 3, 4)	WO
Pulse width modulation mode (8-bit PWM)	Modify the pulse width as follows: PWM period: (2^8 – 1) x (m + 1)/ fj High level PWM pulse width: (m + 1)n / fj where n = high-order address set value, m = low-order address set value, fj = count source frequency	00_{16} to FE_{16} (High-order address) 00_{16} to FF_{16} (Low-order address) (Note 3, 4)	WO

Note 1: The register must be accessed in 16 bit units.
Note 2: If the TAi register is set to '0000 16,' the counter does not work and timer Ai interrupt requests are not generated either. Furthermore, if "pulse output" is selected, no pulses are output from the TAiOUT pin.
Note 3: If the TAi register is set to '0000 16,' the pulse width modulator does not work, the output level on the TAiOUT pin remains low, and timer Ai interrupt requests are not generated either. The same applies when the 8 high-order bits of the timer TAi register are set to '001 6' while operating as an 8-bit pulse width modulator.
Note 4: Use the MOV instruction to write to the TAi register.
Note 5: The timer counts pulses from an external device or overflows or underflows in other timers.

Count start flag

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
TABSR	0380_{16}	00_{16}

Bit symbol	Bit name	Function	RW
TA0S	Timer A0 count start flag	0 : Stops counting 1 : Starts counting	RW
TA1S	Timer A1 count start flag		RW
TA2S	Timer A2 count start flag		RW
TA3S	Timer A3 count start flag		RW
TA4S	Timer A4 count start flag		RW
—— (b7-b5)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—

Up/down flag (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
UDF	0384_{16}	00_{16}

Bit symbol	Bit name	Function	RW
TA0UD	Timer A0 up/down flag	0 : Down count 1 : Up count Enabled by setting the TAiMR register's MR2 bit to "0" (= switching source in UDF register) during event counter mode.	RW
TA1UD	Timer A1 up/down flag		RW
TA2UD	Timer A2 up/down flag		RW
TA3UD	Timer A3 up/down flag		RW
TA4UD	Timer A4 up/down flag		RW
TA2P	Timer A2 two-phase pulse signal processing select bit	0 : two-phase pulse signal processing disabled 1 : two-phase pulse signal processing enabled (Notes 2, 3)	WO
TA3P	Timer A3 two-phase pulse signal processing select bit		WO
TA4P	Timer A4 two-phase pulse signal processing select bit		WO

Note 1: Use MOV instruction to write to this register.
Note 2: Make sure the port direction bits for the TA2 IN to TA4I N and TA2 OUT to TA4 OUT pins are set to "0" (input mode).
Note 3: When not using the two-phase pulse signal processing function, set the corresponding bit to "0" (TA2P and TA3P must be set "0").

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

One-shot start flag

b7	b6	b5	b4	b3	b2	b1	b0
		0					

Symbol ONSF Address 0382_{16} After reset 00_{16}

Bit symbol	Bit name	Function	RW
TA0OS	Timer A0 one-shot start flag	The timer starts counting by setting this bit to “1” while the TMOD1 to TMOD0 bits of TAiMR register (i = 0 to 4) = ‘10_2’ (= one-shot timer mode) and the MR2 bit of TAiMR register = “0” (=TAiOS bit enabled). When read, its content is “0”.	RW
TA1OS	Timer A1 one-shot start flag		RW
TA2OS	Timer A2 one-shot start flag		RW
TA3OS	Timer A3 one-shot start flag		RW
TA4OS	Timer A4 one-shot start flag		RW
— (b6)	Reserved bit	Should be set to “0”.	RW
TA0TGL	Timer A0 event/trigger select bit	b7 b6 0 0 : Input on $TA0_{IN}$ is selected (Note 1) 0 1 : TB2 overflow is selected (Note 2) 1 0 : TA4 overflow is selected (Note 2) 1 1 : TA1 overflow is selected (Note 2)	RW
TA0TGH			RW

Note 1: Make sure the PD7_1 bit of PD7 register is set to “0” (= input mode).
Note 2: Overflow or underflow

Trigger select register

b7	b6	b5	b4	b3	b2	b1	b0

Symbol TRGSR Address 0383_{16} After reset 00_{16}

Bit symbol	Bit name	Function	RW
TA1TGL	Timer A1 event/trigger select bit	b1 b0 0 0 : Input on $TA1_{IN}$ is selected (Note 1) 1 0 : TA0 overflow is selected (Note 2) 1 1 : TA2 overflow is selected (Note 2) (Note 3)	RW
TA1TGH			RW
TA2TGL	Timer A2 event/trigger select bit	b3 b2 1 0 : TA1 overflow is selected (Note 2) 1 1 : TA3 overflow is selected (Note 2) (Note 3)	RW
TA2TGH			RW
TA3TGL	Timer A3 event/trigger select bit	b5 b4 1 0 : TA2 overflow is selected (Note 2) 1 1 : TA4 overflow is selected (Note 2) (Note 3)	RW
TA3TGH			RW
TA4TGL	Timer A4 event/trigger select bit	b7 b6 0 0 : Input on $TA4_{IN}$ is selected (Note 1) 1 0 : TA3 overflow is selected (Note 2) 1 1 : TA0 overflow is selected (Note 2) (Note 3)	RW
TA4TGH			RW

Note 1: Make sure the port direction bits for the $TA1_{IN}$ to $TA4_{IN}$ pins are set to “0” (= input mode).
Note 2: Overflow or underflow
Note 3: Do not set cases which are not discribed.

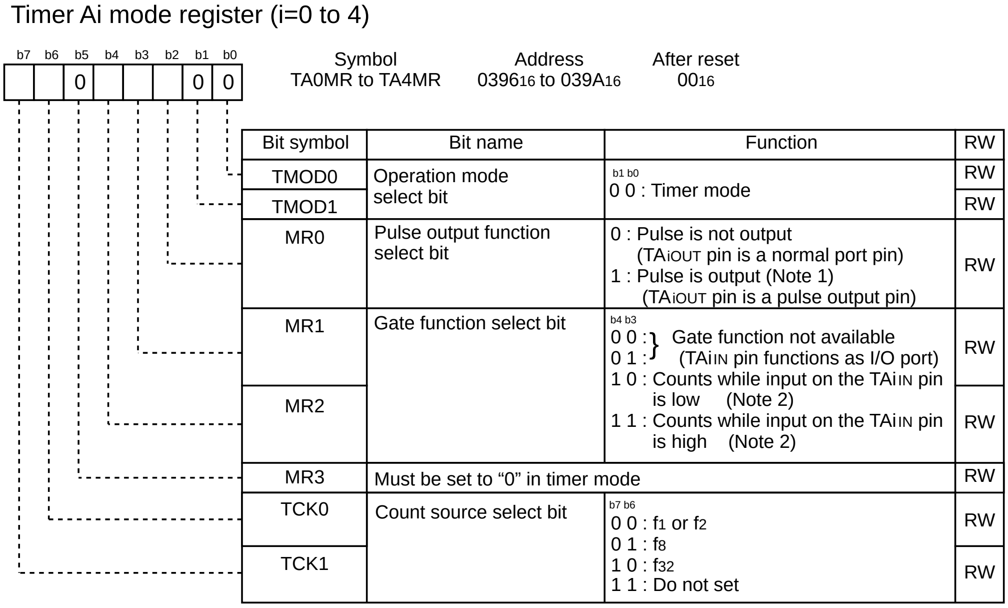
Figure 1.12.5. ONSF Register, TRGSR Register, and CPSRF Register

1. Timer Mode

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

Table 1.12.1. Specifications in Timer Mode

Item	Specification
Count source	f_1, f_2, f_8, f_{32}
Count operation	• Down-count • When the timer underflows, it reloads the reload register contents and continues counting
Divide ratio	1/(n+1) n: set value of TAiMR register (i= 0 to 4) 0000_{16} to $FFFF_{16}$
Count start condition	Set TAiS bit of TABSR register to “1” (= start counting)
Count stop condition	Set TAiS bit to “0” (= stop counting)
Interrupt request generation timing	Timer underflow
TAi$_{IN}$ pin function	I/O port or gate input i≠2, 3
TAi$_{OUT}$ pin function	I/O port or pulse output
Read from timer	Count value can be read by reading TAi 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 only reload register (Transferred to counter when reloaded next)
Select function	• Gate function Counting can be started and stopped by an input signal to TAi$_{IN}$ pin • Pulse output function Whenever the timer underflows, the output polarity of TAi$_{OUT}$ pin is inverted. When not counting, the pin outputs a low.



Symbol	Address	After reset
TA0MR to TA4MR	0396_{16} to $039A_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 0 : Timer mode	RW
TMOD1			RW
MR0	Pulse output function select bit	0 : Pulse is not output (TAi$_{OUT}$ pin is a normal port pin) 1 : Pulse is output (Note 1) (TAi$_{OUT}$ pin is a pulse output pin)	RW
MR1	Gate function select bit	b4 b3 0 0 : } Gate function not available 0 1 : } (TAi$_{IN}$ pin functions as I/O port) 1 0 : Counts while input on the TAi$_{IN}$ pin is low (Note 2) 1 1 : Counts while input on the TAi$_{IN}$ pin is high (Note 2)	RW
MR2			RW
MR3	Must be set to “0” in timer mode		RW
TCK0	Count source select bit	b7 b6 0 0 : f_1 or f_2 0 1 : f_8 1 0 : f_{32} 1 1 : Do not set	RW
TCK1			RW

Note 1: TA0$_{OUT}$ pin is N-channel open drain output.
Note 2: The port direction bit for the TAi$_{IN}$ pin must be set to “0” (= input mode).
There are not TA2$_{IN}$ and TA3$_{IN}$.

Figure 1.12.6. Timer Ai Mode Register in Timer Mode

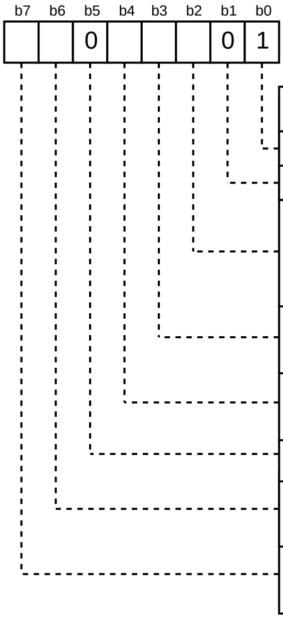
2. Event Counter Mode

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 TAiMR register in event counter mode (when not processing two-phase pulse signal). Figure 1.12.8 shows TA4MR 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	• External signals input to TAi_{IN} pin (i=0 to 4) (effective edge can be selected in program) timer Aj (j=i-1, except j=4 if i=0) overflows or underflows, timer Ak (k=i+1, except k=0 if i=4) overflows or underflows
Count operation	• Up-count or down-count can be selected by external signal or program • 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.
Divided ratio	1/ ($FFFF_{16}$ - n + 1) for up-count 1/ (n + 1) for down-count n : set value of TAi register 0000_{16} to $FFFF_{16}$
Count start condition	Set TAiS bit of TABSR register to “1” (= start counting)
Count stop condition	Set TAiS bit to “0” (= stop counting)
Interrupt request generation timing	Timer overflow or underflow
TAi_{IN} pin function	I/O port or count source input i≠2, 3
TAi_{OUT} 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	• 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 only reload register (Transferred to counter when reloaded next)
Select function	• Free-run count function Even when the timer overflows or underflows, the reload register content is not reloaded to it • Pulse output function Whenever the timer underflows or underflows, the output polarity of TAi_{OUT} pin is inverted . When not counting, the pin outputs a low.

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



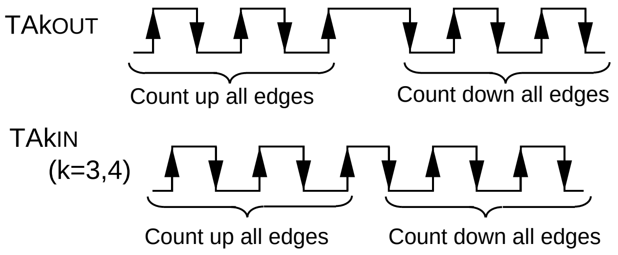
Symbol	Address	After reset
TA0MR to TA4MR	0396_{16} to $039A_{16}$	00_{16}

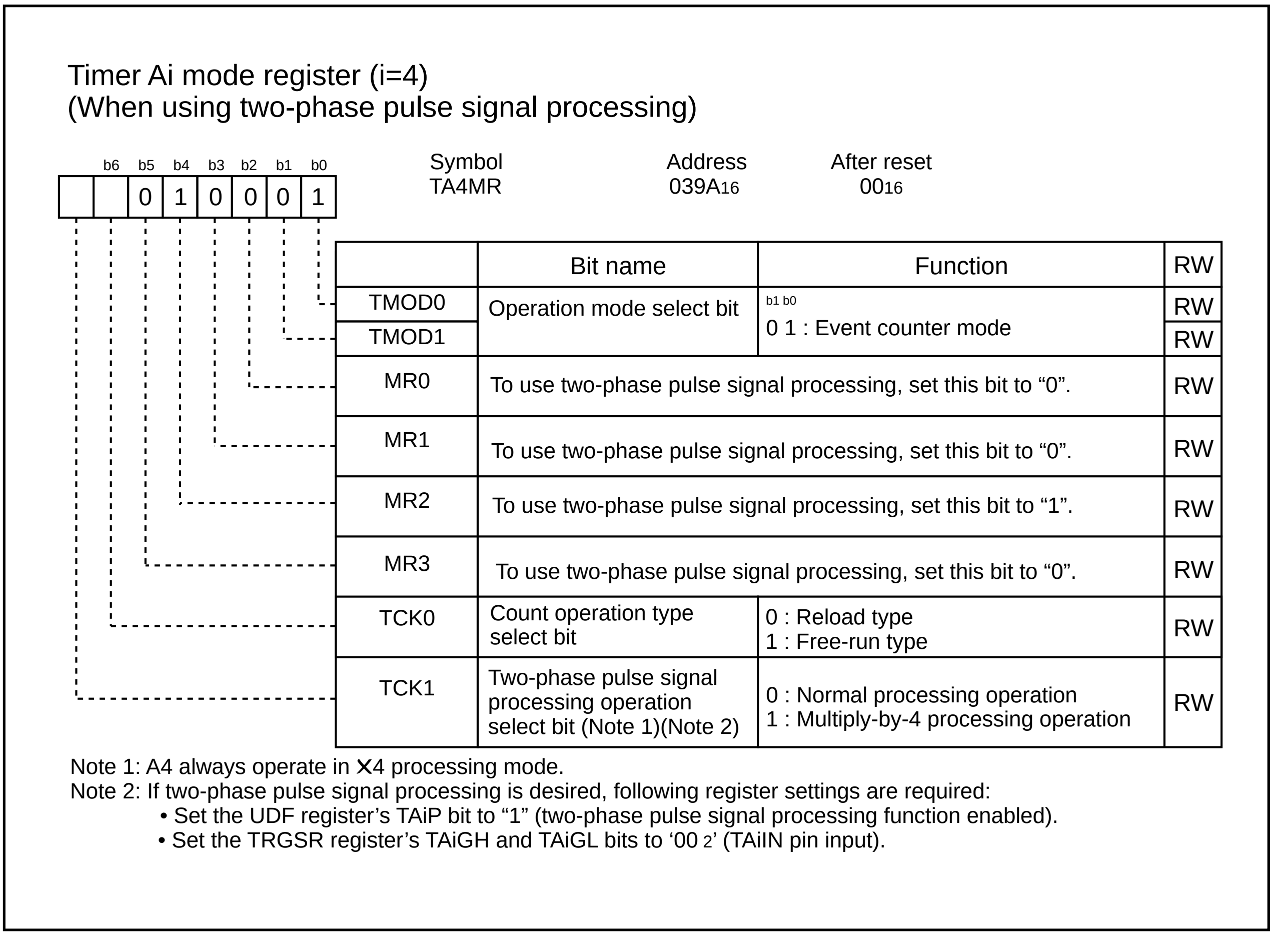
Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 1 : Event counter mode (Note 1)	RW
TMOD1			RW
MR0	Pulse output function select bit	0 : Pulse is not output (TAiOUT pin functions as I/O port) 1 : Pulse is output (Note 2) (TAiOUT pin functions as pulse output pin)	RW
MR1	Count polarity 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 ‘00_2’ (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/ ($FFFF_{16}$ - n + 1) for up-count 1/ (n + 1) for down-count n : set value of TAi register 0000_{16} to $FFFF_{16}$
Count start condition	Set TAiS bit of TABSR register to “1” (= start counting)
Count stop condition	Set TAiS 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	• Multiply-by-4 processing operation (timer A4) 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.  TAkOUT Count up all edges Count down all edges TAkIN (k=3,4) Count up all edges Count down all edges



Timer Ai mode register (i=4)
(When using two-phase pulse signal processing)

Symbol	Address	After reset
TA4MR	039A$_{16}$	00$_{16}$

	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 1 : Event counter mode	RW
TMOD1			RW
MR0	To use two-phase pulse signal processing, set this bit to “0”.		RW
MR1	To use two-phase pulse signal processing, set this bit to “0”.		RW
MR2	To use two-phase pulse signal processing, set this bit to “1”.		RW
MR3	To use two-phase pulse signal processing, set this bit to “0”.		RW
TCK0	Count operation type select bit	0 : Reload type 1 : Free-run type	RW
TCK1	Two-phase pulse signal processing operation select bit (Note 1)(Note 2)	0 : Normal processing operation 1 : Multiply-by-4 processing operation	RW

Note 1: A4 always operate in ×4 processing mode.
Note 2: If two-phase pulse signal processing is desired, following register settings are required:
- Set the UDF register’s TAiP bit to “1” (two-phase pulse signal processing function enabled).
- Set the TRGSR register’s TAiGH and TAiGL bits to ‘00$_2$’ (TAiIN pin input).

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

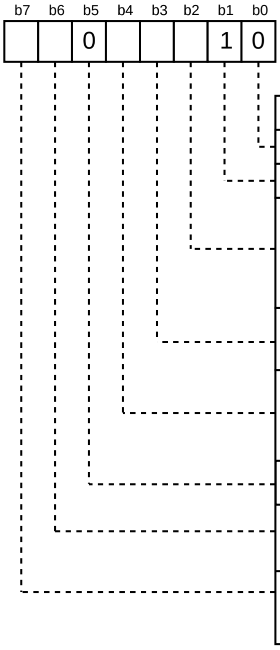
3. One-shot Timer Mode

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 TAiMR register in one-shot timer mode.

Table 1.12.4. Specifications in One-shot Timer Mode

Item	Specification
Count source	f_1, f_2, f_8, f_{32}
Count operation	• Down-count • When the counter reaches 0000_{16}, it stops counting after reloading a new value • If a trigger occurs when counting, the timer reloads a new count and restarts counting
Divide ratio	1/n n : set value of TAi register 0000_{16} to $FFFF_{16}$ However, the counter does not work if the divide-by-n value is set to 0000_{16}.
Count start condition	TAiS bit of TABSR register = “1” (start counting) and one of the following triggers occurs. • External trigger input from the TAi_{IN} pin • timer Aj (j=i-1, except j=4 if i=0) overflow or underflow, timer Ak (k=i+1, except k=0 if i=4) overflow or underflow • The TAiOS bit of ONSF register is set to “1” (= timer starts)
Count stop condition	• When the counter is reloaded after reaching “0000_{16}” • TAiS bit is set to “0” (= stop counting)
Interrupt request generation timing	When the counter reaches “0000_{16}”
TAi_{IN} pin function	I/O port or trigger input
TAi_{OUT} pin function	I/O port or pulse output
Read from timer	An indeterminate value is read by reading TAi 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 only reload register (Transferred to counter when reloaded next)
Select function	• Pulse output function The timer outputs a low when not counting and a high when counting.

Timer Ai mode register (i=0 to 4)



Symbol	Address	After reset
TA0MR to TA4MR	396_{16} to $039A_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 1 0 : One-shot timer mode	RW
TMOD1			RW
MR0	Pulse output function select bit	0 : Pulse is not output (TAiOUT pin functions as I/O port) 1 : Pulse is output (Note 1) (TAiOUT pin functions as a pulse output pin)	RW
MR1	External trigger select bit (Note 2)	0 : Falling edge of input signal to TAiIN pin (Note 3) 1 : Rising edge of input signal to TAiIN pin (Note 3)	RW
MR2	Trigger select bit	0 : TAiOS bit is enabled 1 : Selected by TAiTGH to TAiTGL bits	RW
MR3	Must be set to “0” in one-shot timer mode		RW
TCK0	Count source select bit	b7 b6 0 0 : f_1 or f_2 0 1 : f_8 1 0 : f_{32} 1 1 : Do not set	RW
TCK1			RW

Note 1: TA0OUT pin is N-channel open drain output.
Note 2: Effective when the TAiGH and TAiGL bits of ONSF or TRGSR register are ‘00_2’ (TAiIN pin input).
Note 3: The port direction bit for the TAiIN pin must be set to “0” (= input mode).
There are not TA2IN and TA3IN.

Figure 1.12.9. TAiMR Register in One-shot Timer Mode

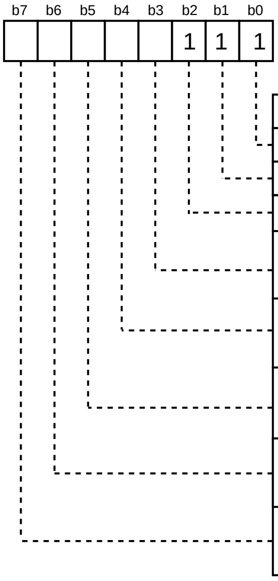
4. Pulse Width Modulation (PWM) Mode

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 TAiMR 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	f1, f2, f8, f32
Count operation	• Down-count (operating as an 8-bit or a 16-bit pulse width modulator) • The timer reloads a new value at a rising edge of PWM pulse and continues counting • The timer is not affected by a trigger that occurs during counting
16-bit PWM	• High level width n / fj n : set value of TAi register (i=o to 4) • Cycle time $(2^{16}-1)$ / fj fixed fj: count source frequency (f1, f2, f8, f32)
8-bit PWM	• High level width n x (m+1) / fj n : set value of TAiMR register high-order address • Cycle time $(2^{8}-1)$ x (m+1) / fj m : set value of TAiMR register low-order address
Count start condition	• External trigger input from the TAiIN pin • timer Aj (j=i-1, except j=4 if i=0) overflow or underflow, timer Ak (k=i+1, except k=0 if i=4) overflow or underflow • TAiS bit of TABSR register is set to “1” (= start counting)
Count stop condition	TAiS bit is set to “0” (= stop counting)
Interrupt request generation timing	PWM pulse goes “L”
TAiIN pin function	I/O port or trigger input
TAiOUT pin function	Pulse output
Read from timer	An indeterminate value is read by reading TAi 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 only reload register (Transferred to counter when reloaded next)

Timer Ai mode register (i= 0 to 4)

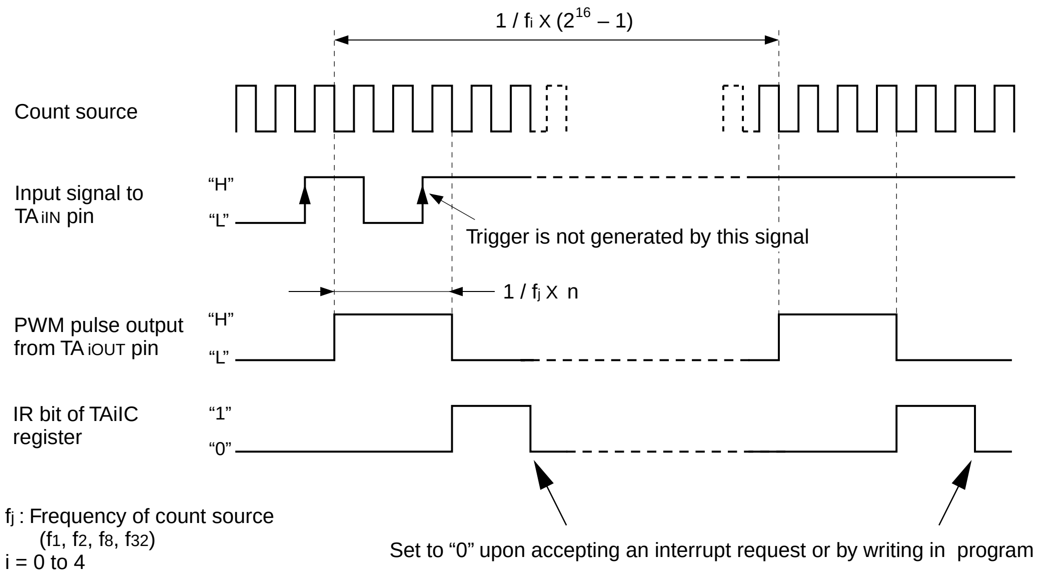


Symbol	Address	After reset
TA0MR to TA4MR	0396_{16} to $039A_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 1 1 : PWM mode (Note 1)	RW
TMOD1			RW
MR0	Must be set to “1” in PWM mode		RW
MR1	External trigger select bit (Note 2)	0: Falling edge of input signal to TAi_{IN} pin(Note 3) 1: Rising edge of input signal to TAi_{IN} pin(Note 3)	RW
MR2	Trigger select bit	0 : TAiOS bit is enabled 1 : Selected by TAiTGH to TAiTGL bits	RW
MR3	16/8-bit PWM mode select bit	0: Functions as a 16-bit pulse width modulator 1: Functions as an 8-bit pulse width modulator	RW
TCK0	Count source select bit	b7 b6 0 0 : f_1 or f_2 0 1 : f_8 1 0 : f_{32} 1 1 : Do not set	RW
TCK1			RW

Note 1: $TA0_{OUT}$ pin is N-channel open drain output.
Note 2: Effective when the TAiGH and TAiGL bits of ONSF or TRGSR register are ‘00_2’ (TAi_{IN} pin input).
Note 3: The port direction bit for the TAi_{IN} pin must be set to “0” (= input mode).
There are not $TA2_{IN}$ and $TA3_{IN}$.

Figure 1.12.10. TAiMR Register in PWM Mode

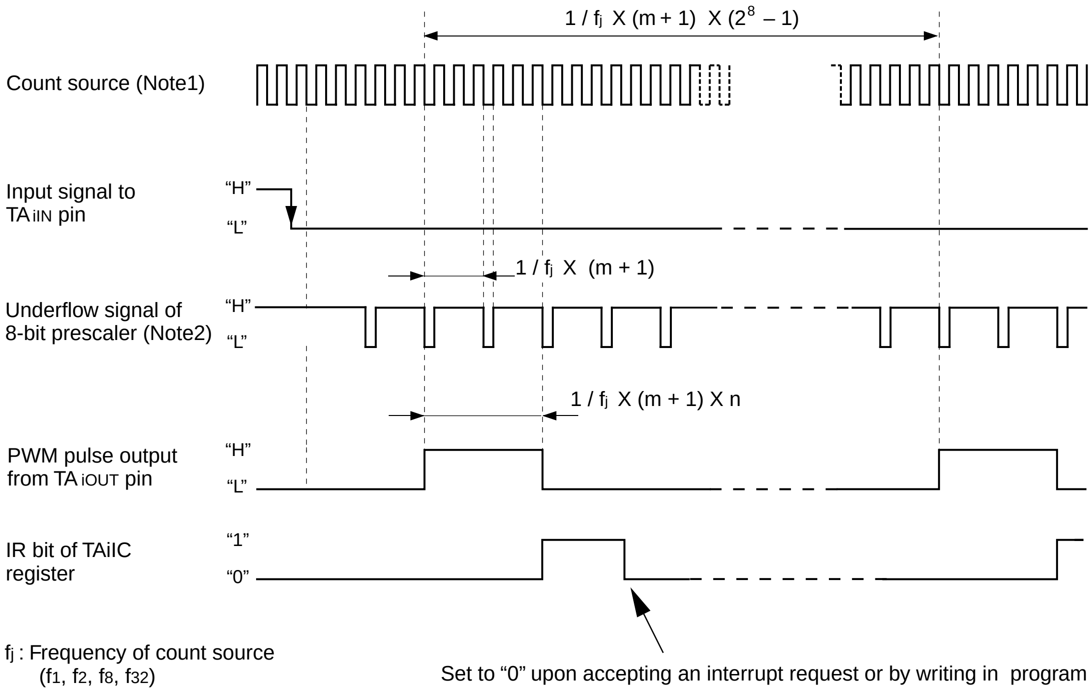


Set to "0" upon accepting an interrupt request or by writing in program

fj : Frequency of count source
(f1, f2, f8, f32)
i = 0 to 4

Note 1: n = 0000_{16} to $FFFE_{16}$.
Note 2: This timing diagram is for the case where the TAi register is '0003_{16},' the TAiGH and TAiGL bits of ONSF or TRGSR register = '00_2' (TAiIN pin input), the MR1 bit of TAiMR register = 1 (rising edge), and the MR2 bit of TAiMR register = 1 (trigger selected by TAiTGH and TAiTGL bits).

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



Set to "0" upon accepting an interrupt request or by writing in program

fj : Frequency of count source
(f1, f2, f8, f32)
i = 0 to 4

Note 1: The 8-bit prescaler counts the count source.
Note 2: The 8-bit pulse width modulator counts the 8-bit prescaler's underflow signal.
Note 3: m = 00_{16} to FF_{16}; n = 00_{16} to FE_{16}.
Note 4: This timing diagram is for the case where the TAi register is '0202_{16},' the TAiGH and TAiGL bits of ONSF or TRGSR register = '00_2' (TAiIN pin input), the MR1 bit of TAiMR register = 0 (falling edge), and the MR2 bit of TAiMR register = 1 (trigger selected by TAiTGH and TAiTGL bits).

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.

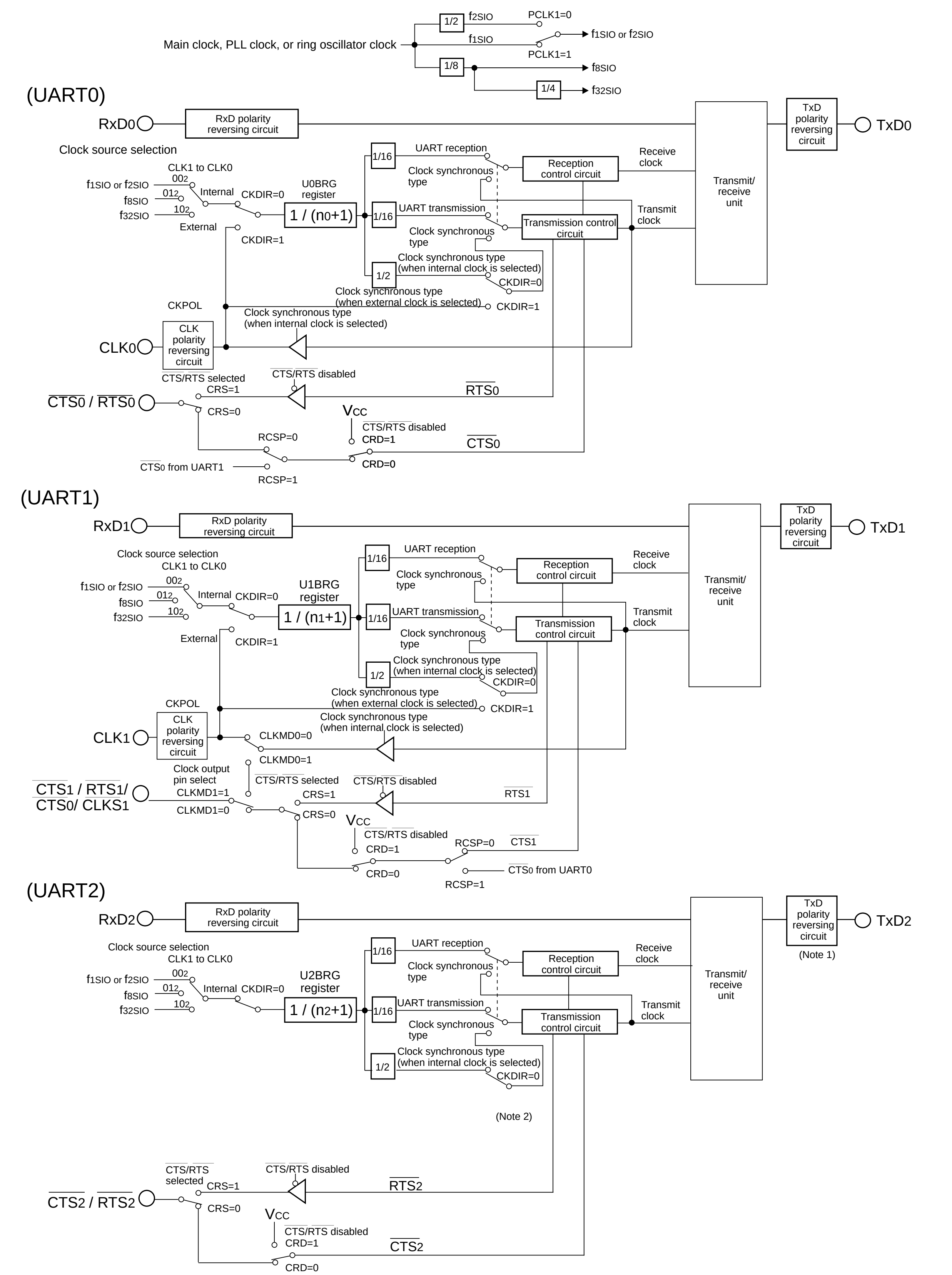
Figure 1.13.1 shows the block diagram of UARTi. Figures 1.13.2 shows the block diagram of the UARTi transmit/receive.

UARTi has the following modes:

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

Figures 1.13.3 to 1.13.8 show the UARTi-related registers.

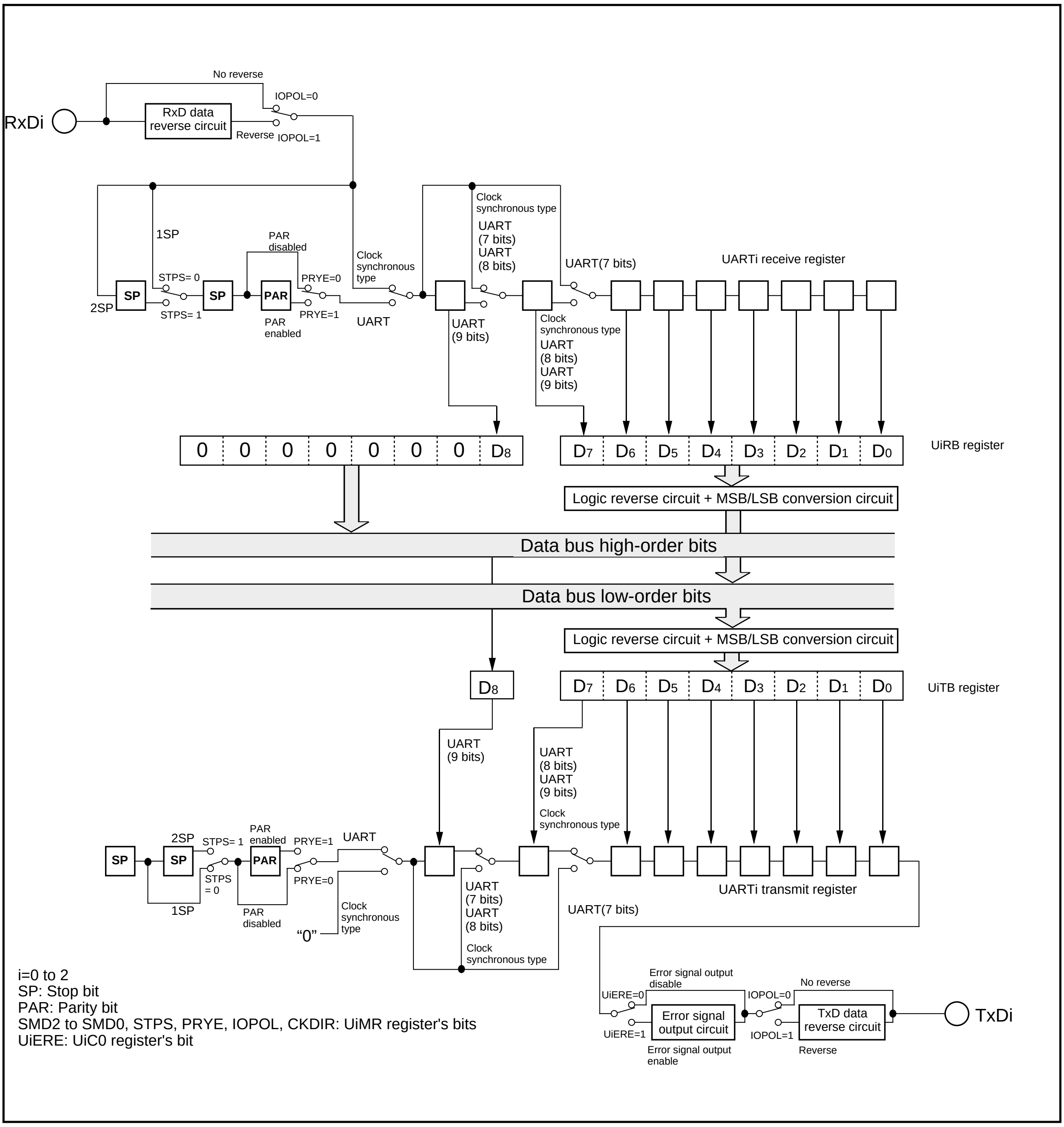
Refer to tables listing each mode for register setting.



i = 0 to 2
ni: Values set to the UiBRG register
SMD2 to SMD0, CKDIR: UiMR register's bits
CLK1 to CLK0, CKPOL, CRD, CRS: UiC0 register's bits
CLKMD0, CLKMD1, RCSP: UCON register's bits

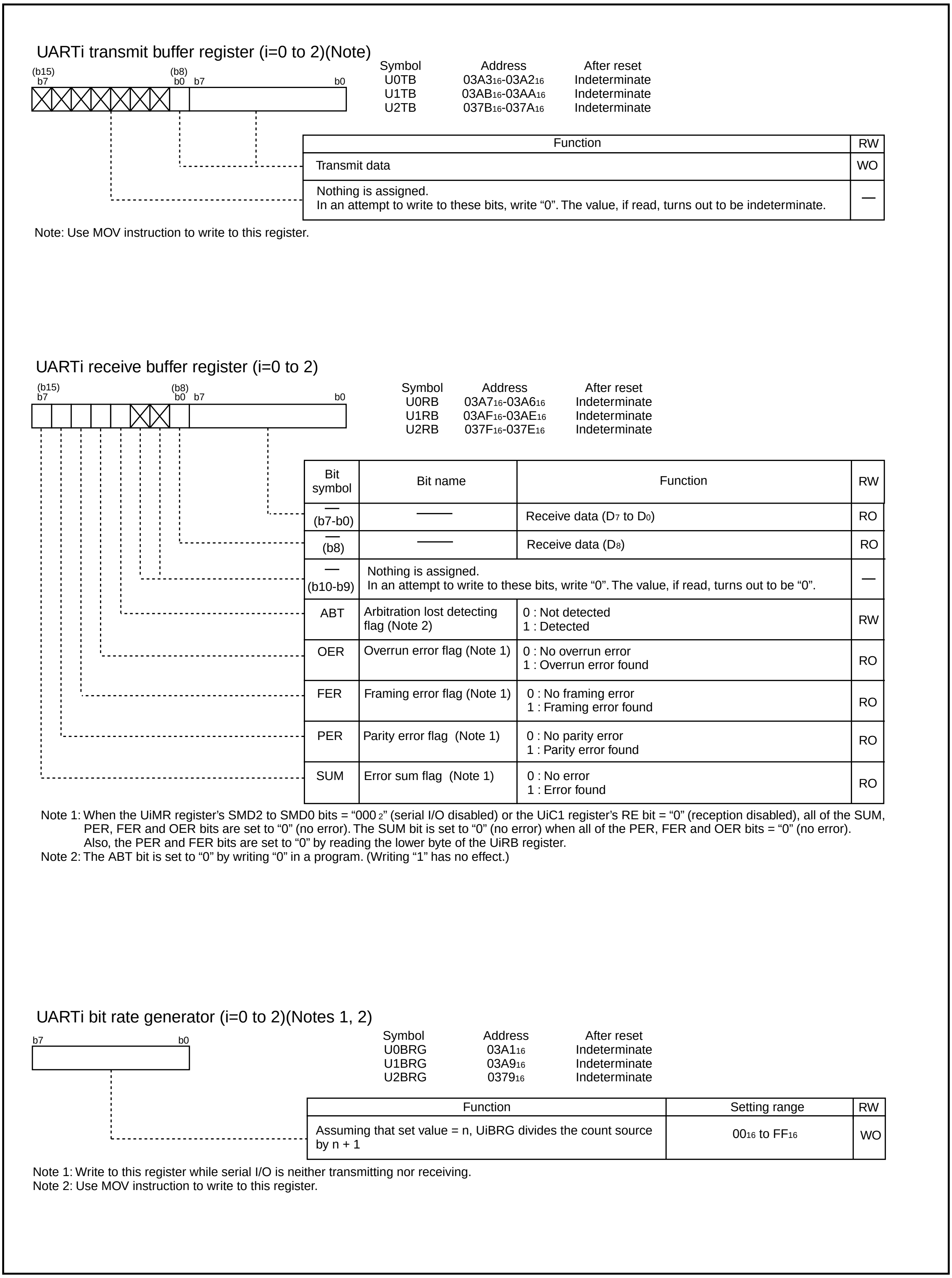
Note 1: TxD2 is the N-channel open-drain output. Cannot be set to the CMOS output.
Note 2: UART2 does not have CLK2 port So CKDIR must not be set “1”.

Figure 1.13.1. UARTi Block Diagram



i=0 to 2
SP: Stop bit
PAR: Parity bit
SMD2 to SMD0, STPS, PRYE, IOPOL, CKDIR: UiMR register's bits
UiERE: UiC0 register's bit

Figure 1.13.2. UARTi Transmit/Receive Unit



UARTi transmit buffer register (i=0 to 2)(Note)

Symbol	Address	After reset
U0TB	$03A3_{16}$-$03A2_{16}$	Indeterminate
U1TB	$03AB_{16}$-$03AA_{16}$	Indeterminate
U2TB	$037B_{16}$-$037A_{16}$	Indeterminate

Function	RW
Transmit data	WO
Nothing is assigned. In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate.	—

Note: Use MOV instruction to write to this register.

UARTi receive buffer register (i=0 to 2)

Symbol	Address	After reset
U0RB	$03A7_{16}$-$03A6_{16}$	Indeterminate
U1RB	$03AF_{16}$-$03AE_{16}$	Indeterminate
U2RB	$037F_{16}$-$037E_{16}$	Indeterminate

Bit symbol	Bit name	Function	RW
— (b7-b0)	——	Receive data (D_7 to D_0)	RO
— (b8)	——	Receive data (D_8)	RO
— (b10-b9)	Nothing is assigned. In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”.		—
ABT	Arbitration lost detecting flag (Note 2)	0 : Not detected 1 : Detected	RW
OER	Overrun error flag (Note 1)	0 : No overrun error 1 : Overrun error found	RO
FER	Framing error flag (Note 1)	0 : No framing error 1 : Framing error found	RO
PER	Parity error flag (Note 1)	0 : No parity error 1 : Parity error found	RO
SUM	Error sum flag (Note 1)	0 : No error 1 : Error found	RO

Note 1: When the UiMR register’s SMD2 to SMD0 bits = “000_2” (serial I/O disabled) or the UiC1 register’s RE bit = “0” (reception disabled), all of the SUM, PER, FER and OER bits are set to “0” (no error). The SUM bit is set to “0” (no error) when all of the PER, FER and OER bits = “0” (no error). Also, the PER and FER bits are set to “0” by reading the lower byte of the UiRB register.

Note 2: The ABT bit is set to “0” by writing “0” in a program. (Writing “1” has no effect.)

UARTi bit rate generator (i=0 to 2)(Notes 1, 2)

Symbol	Address	After reset
U0BRG	$03A1_{16}$	Indeterminate
U1BRG	$03A9_{16}$	Indeterminate
U2BRG	0379_{16}	Indeterminate

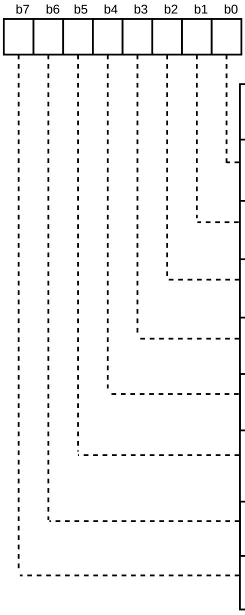
Function	Setting range	RW
Assuming that set value = n, UiBRG divides the count source by n + 1	00_{16} to FF_{16}	WO

Note 1: Write to this register while serial I/O is neither transmitting nor receiving.

Note 2: Use MOV instruction to write to this register.

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

UARTi transmit/receive mode register (i=0 to 2)

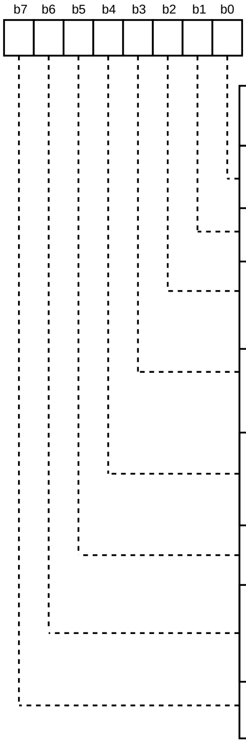


Symbol	Address	After reset
U0MR to U2MR	$03A0_{16}$, $03A8_{16}$, 0378_{16}	00_{16}

Bit symbol	Bit name	Function	RW
SMD0	Serial I/O mode select bit (Note 2)	b2 b1 b0 0 0 0 : Serial I/O disabled 0 0 1 : Clock synchronous serial I/O mode 0 1 0 : I^2C mode (Note 3) 1 0 0 : UART mode transfer data 7 bits long 1 0 1 : UART mode transfer data 8 bits long 1 1 0 : UART mode transfer data 9 bits long Must not be set except above	RW
SMD1			RW
SMD2			RW
CKDIR	Internal/external clock select bit	0 : Internal clock 1 : External clock (Note 1) (Note 4)	RW
STPS	Stop bit length select bit	0 : One stop bit 1 : Two stop bits	RW
PRY	Odd/even parity select bit	Effective when PRYE = 1 0 : Odd parity 1 : Even parity	RW
PRYE	Parity enable bit	0 : Parity disabled 1 : Parity enabled	RW
IOPOL	TxD, RxD I/O polarity reverse bit	0 : No reverse 1 : Reverse	RW

Note 1: Set the corresponding port direction bit for each CLKi pin to “0” (input mode).
Note 2: To receive data, set the corresponding port direction bit for each RxDi pin to “0” (input mode).
Note 3: Set the corresponding port direction bit for SCL and SDA pins to “0” (input mode).
Note 4: Set “0” to select internal clock of UART2.

UARTi transmit/receive control register 0 (i=0 to 2)



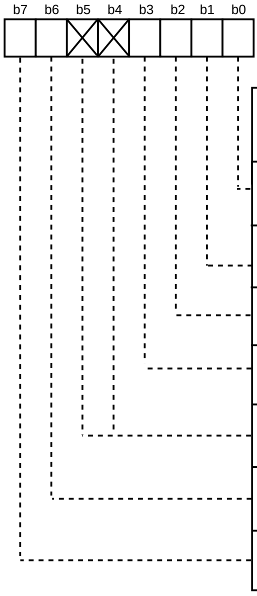
Symbol	Address	After reset
U0C0 to U2C0	$03A4_{16}$, $03AC_{16}$, $037C_{16}$	00001000_2

Bit symbol	Bit name	Function	RW
CLK0	BRG count source select bit	b1 b0 0 0 : f_{1SIO} or f_{2SIO} is selected 0 1 : f_{8SIO} is selected 1 0 : f_{32SIO} is selected 1 1 : Must not be set	RW
CLK1			RW
CRS	$\overline{CTS}$/$\overline{RTS}$ function select bit (Note 4)	Effective when CRD = 0 0 : $\overline{CTS}$ function is selected (Note 1) 1 : RTS function is selected	RW
TXEPT	Transmit register empty flag	0 : Data present in transmit register (during transmission) 1 : No data present in transmit register (transmission completed)	RO
CRD	$\overline{CTS}$/$\overline{RTS}$ disable bit	0 : $\overline{CTS}$/$\overline{RTS}$ function enabled 1 : CTS/RTS function disabled (P6$_0$, P6$_4$ and P7$_3$ can be used as I/O ports)	RW
NCH	Data output select bit (Note 2)	0 : TxDi/SDAi and SCLi pins are CMOS output 1 : TxDi/SDAi and SCLi pins are N-channel open-drain output	RW
CKPOL	CLK polarity select bit	0 : Transmit data is output at falling edge of transfer clock and receive data is input at rising edge 1 : Transmit data is output at rising edge of transfer clock and receive data is input at falling edge	RW
UFORM	Transfer format select bit (Note 3)	0 : LSB first 1 : MSB first	RW

Note 1: Set the corresponding port direction bit for each $\overline{CTSi}$ pin to “0” (input mode).
Note 2: TxD_2/SDA_2 are N-channel open-drain output. Cannot be set to the CMOS output. NCH bit of U2C0 register is effective in an output set up of SCL_2 pin.
Note 3: Effective for clock synchronous serial I/O mode and UART mode transfer data 8 bits long.
Note 4: CTS_1/RTS_1 can be used when the UCON register’s CLKMD1 bit = “0” (only CLK_1 output) and the UCON register’s RCSP bit = “0” (CTS_0/RTS_0 not separated).

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

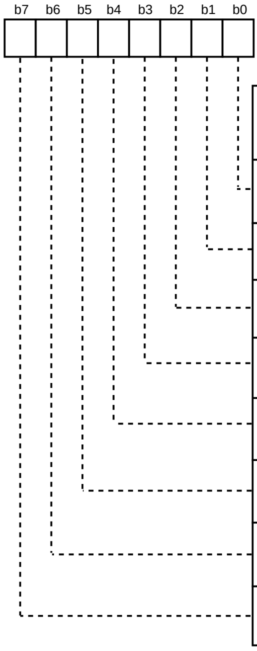
UARTi transmit/receive control register 1 (i=0, 1)



Symbol	Address	After reset
U0C1, U1C1	03A5$_{16}$,03AD$_{16}$	00000010$_2$

Bit symbol	Bit name	Function	RW
TE	Transmit enable bit	0 : Transmission disabled 1 : Transmission enabled	RW
TI	Transmit buffer empty flag	0 : Data present in UiTB register 1 : No data present in UiTB register	RO
RE	Receive enable bit	0 : Reception disabled 1 : Reception enabled	RW
RI	Receive complete flag	0 : No data present in UiRB register 1 : Data present in UiRB register	RO
— (b5-b4)	Nothing is assigned. When write, set “0”. When read, these contents are “0”.		—
UiLCH	Data logic select bit	0 : No reverse 1 : Reverse	RW
UiERE	Error signal output enable bit	0 : Output disabled 1 : Output enabled	RW

UART2 transmit/receive control register 1

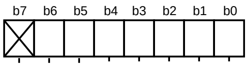


Symbol	Address	After reset
U2C1	037D$_{16}$	00000010$_2$

Bit symbol	Bit name	Function	RW
TE	Transmit enable bit	0 : Transmission disabled 1 : Transmission enabled	RW
TI	Transmit buffer empty flag	0 : Data present in U2TB register 1 : No data present in U2TB register	RO
RE	Receive enable bit	0 : Reception disabled 1 : Reception enabled	RW
RI	Receive complete flag	0 : No data present in U2RB register 1 : Data present in U2RB register	RO
U2IRS	UART2 transmit interrupt cause select bit	0 : Transmit buffer empty (TI = 1) 1 : Transmit is completed (TXEPT = 1)	RW
U2RRM	UART2 continuous receive mode enable bit	0 : Continuous receive mode disabled 1 : Continuous receive mode enabled	RW
U2LCH	Data logic select bit	0 : No reverse 1 : Reverse	RW
U2ERE	Error signal output enable bit	0 : Output disabled 1 : Output enabled	RW

Figure 1.13.5. U0C1 to U2C1 Registers

UART transmit/receive control register 2

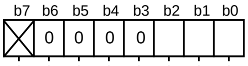


Symbol	Address	After reset
UCON	$03B0_{16}$	$X0000000_2$

Bit symbol	Bit name	Function	RW
U0IRS	UART0 transmit interrupt cause select bit	0 : Transmit buffer empty (TI = 1) 1 : Transmission completed (TXEPT = 1)	RW
U1IRS	UART1 transmit interrupt cause select bit	0 : Transmit buffer empty (TI = 1) 1 : Transmission completed (TXEPT = 1)	RW
U0RRM	UART0 continuous receive mode enable bit	0 : Continuous receive mode disabled 1 : Continuous receive mode enable	RW
U1RRM	UART1 continuous receive mode enable bit	0 : Continuous receive mode disabled 1 : Continuous receive mode enabled	RW
CLKMD0	UART1 CLK/CLKS select bit 0	Effective when CLKMD1 = “1” 0 : Clock output from CLK1 1 : Clock output from CLKS1	RW
CLKMD1	UART1 CLK/CLKS select bit 1 (Note)	0 : CLK output is only CLK1 1 : Transfer clock output from multiple pins function selected	RW
RCSP	Separate UART0 CTS/RTS bit	0 : $\overline{CTS}$/$\overline{RTS}$ shared pin 1 : $\overline{CTS}$/$\overline{RTS}$ separated (CTS_0 supplied from the $P6_4$ pin)	RW
—— (b7)	Nothing is assigned. When write, set “0”. When read, its content is indeterminate.		——

Note: When using multiple transfer clock output pins, make sure the following conditions are met:
U1MR register’s CKDIR bit = “0” (internal clock)

UART2 special mode register (i=0 to 2)



Symbol	Address	After reset
U0SMR to U2SMR	$036F_{16}$, 0373_{16}, 0377_{16}	$X0000000_2$

Bit symbol	Bit name	Function	RW
IICM	I^2C mode select bit	0 : Other than I^2C mode 1 : I^2C mode	RW
ABC	Arbitration lost detecting flag control bit	0 : Update per bit 1 : Update per byte	RW
BBS	Bus busy flag	0 : STOP condition detected 1 : START condition detected (busy)	RW (Note1)
—— (b3-b6)	Reserved bit	Set to “0”	RW
—— (b7)	Nothing is assigned. When write, set “0”. When read, its content is indeterminate.		——

Note 1: The BBS bit is set to “0” by writing “0” in a program. (Writing “1” has no effect.).

Figure 1.13.6. UCON Register and U0SMR to U2SMR Registers

UARTi special mode register 2 (i=0 to 2)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
U0SMR2 to U2SMR2	036E$_{16}$, 0372$_{16}$, 0376$_{16}$	X0000000$_2$

Bit symbol	Bit name	Function	RW
IICM2	I^{2}C mode select bit 2	Refer to Table 1.16.4	RW
CSC	Clock-synchronous bit	0 : Disabled 1 : Enabled	RW
SWC	SCL wait output bit	0 : Disabled 1 : Enabled	RW
ALS	SDA output stop bit	0 : Disabled 1 : Enabled	RW
STAC	UARTi initialization bit	0 : Disabled 1 : Enabled	RW
SWC2	SCL wait output bit 2	0: Transfer clock 1: 0 output	RW
SDHI	SDA output disable bit	0: Enabled 1: Disabled (high impedance)	RW
— (b7)	Nothing is assigned. When write, set “0”. When read, its content is indeterminate.		—

UARTi special mode register 3 (i=0 to 2)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
U0SMR3 to U2SMR3	036D$_{16}$, 0371$_{16}$, 0375$_{16}$	000X0X0X$_2$

Bit symbol	Bit name	Function	RW
— (b0)	Nothing is assigned. When write, set “0”. When read, its content is indeterminate.		—
CKPH	Clock phase set bit	0 : Without clock delay 1 : With clock delay	RW
— (b2)	Nothing is assigned. When write, set “0”. When read, its content is indeterminate.		—
NODC	Clock output select bit	0 : CLKi is CMOS output 1 : CLKi is N-channel open drain output	RW
— (b4)	Nothing is assigned. When write, set “0”. When read, its content is indeterminate.		—
DL0	SDAi digital delay setup bit (Note 1, Note 2)	b7 b6 b5 0 0 0 : Without delay 0 0 1 : 1 to 2 cycle(s) of UiBRG count source 0 1 0 : 2 to 3 cycles of UiBRG count source 0 1 1 : 3 to 4 cycles of UiBRG count source 1 0 0 : 4 to 5 cycles of UiBRG count source 1 0 1 : 5 to 6 cycles of UiBRG count source 1 1 0 : 6 to 7 cycles of UiBRG count source 1 1 1 : 7 to 8 cycles of UiBRG count source	RW
DL1			RW
DL2			RW

Note 1 : The DL2 to DL0 bits are used to generate a delay in SDAi output by digital means during I^{2}C mode. In other than I^{2}C mode, set these bits to “000$_2$” (no delay).

Note 2 : The amount of delay varies with the load on SCLi and SDAi pins. Also, when using an external clock, the amount of delay increases by about 100 ns.

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

UARTi special mode register 4 (i=0 to 2)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
U0SMR4 to U2SMR4	$036C_{16}$, 0370_{16}, 0374_{16}	00_{16}

Bit symbol	Bit name	Function	RW
STAREQ	Start condition generate bit (Note)	0 : Clear 1 : Start	RW
RSTAREQ	Restart condition generate bit (Note)	0 : Clear 1 : Start	RW
STPREQ	Stop condition generate bit (Note)	0 : Clear 1 : Start	RW
STSPSEL	SCL,SDA output select bit	0 : Start and stop conditions not output 1 : Start and stop conditions output	RW
ACKD	ACK data bit	0 : ACK 1 : NACK	RW
ACKC	ACK data output enable bit	0 : Serial I/O data output 1 : ACK data output	RW
SCLHI	SCL output stop enable bit	0 : Disabled 1 : Enabled	RW
SWC9	SCL wait bit 3	0 : SCL “L” hold disabled 1 : SCL “L” hold enabled	RW

Note: Set to “0” when each condition is generated.

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) : fj/ 2(n+1) fj = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}. n: Setting value of UiBRG register 00_{16} to FF_{16} • CKDIR bit = "1" (external clock) : Input from CLKi pin
Transmission, reception control	• Selectable from $\overline{\text{CTS}}$ function, $\overline{\text{RTS}}$ function or $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ 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 $\overline{\text{CTS}}$ function is selected, input on the $\overline{\text{CTS}}$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 $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ pins (UART0) $\overline{\text{CTS}_0}$ and $\overline{\text{RTS}_0}$ 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 “001_2”
	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 $\overline{CTS}$ or $\overline{RTS}$ to use
	TXEPT	Transmit register empty flag
	CRD	Enable or disable the $\overline{CTS}$ or $\overline{RTS}$ 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 $\overline{CTS_0}$ signal from the P6$_4$ 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
$\overline{\text{CTS}}$i/$\overline{\text{RTS}}$i (P60, P64)	$\overline{\text{CTS}}$ input	UiC0 register’s CRD bit=0 UiC0 register’s CRS bit=0 PD6 register’s PD6_0 bit=0, PD6_4 bit=0
	$\overline{\text{RTS}}$ output	UiC0 register’s CRD bit=0 UiC0 register’s CRS bit=1
	I/O port	UiC0 register’s CRD bit=1

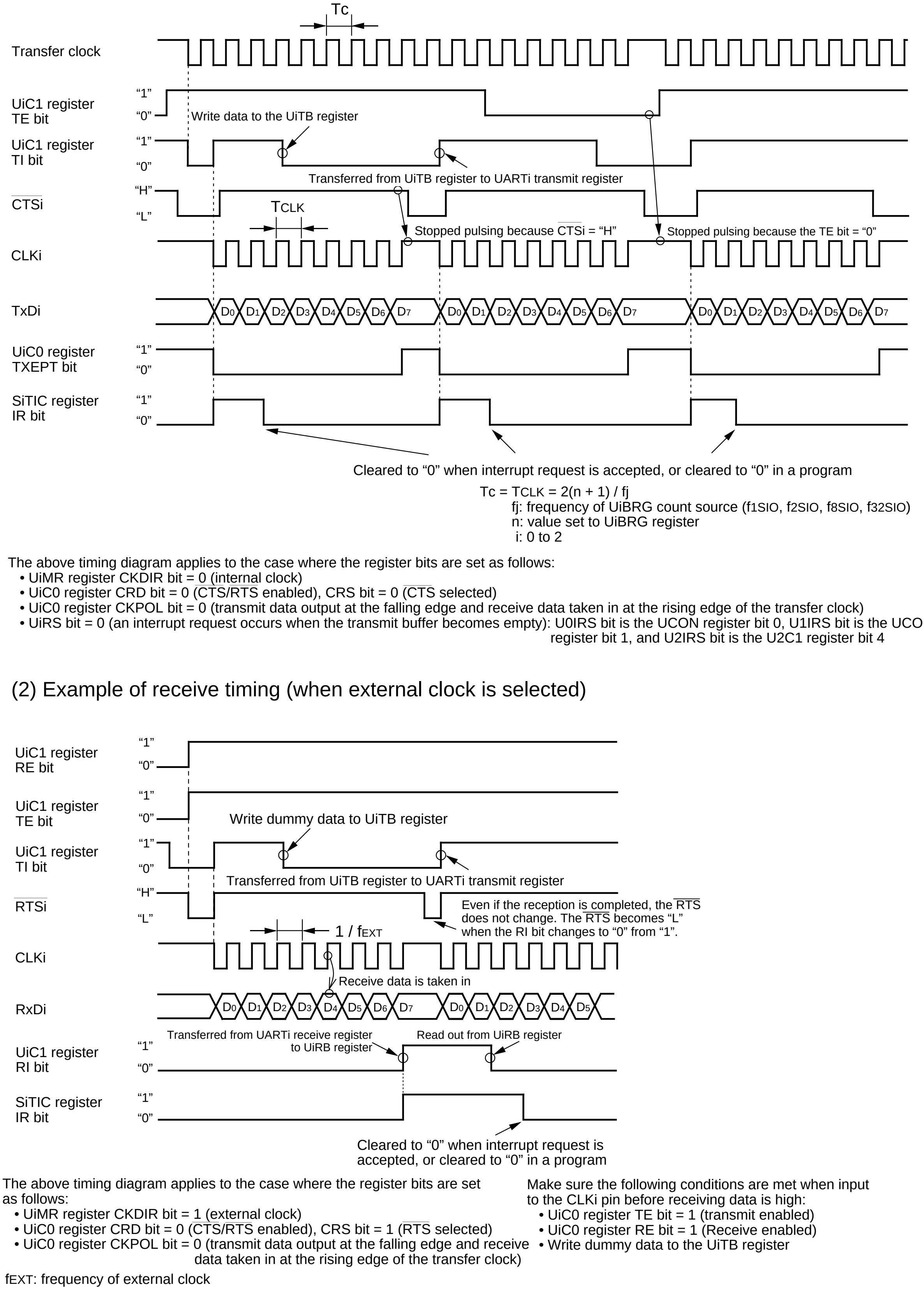
Table 1.14.4. P64 Pin Functions

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
$\overline{\text{CTS}}$1	0	0	0	0	—	0
$\overline{\text{RTS}}$1	0	1	0	0	—	—
$\overline{\text{CTS}}$0(Note1)	0	0	1	0	—	0
CLKS1	—	—	—	1(Note 2)	1	—

Note 1: In addition to this, set the U0C0 register’s CRD bit to “0” ($\overline{\text{CTS}}$0/$\overline{\text{RTS}}$0 enabled) and the U0 C0 register’s CRS bit to “1” ($\overline{\text{RTS}}$0 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)


Tc
Transfer clock
UiC1 register TE bit
"1"
"0"
Write data to the UiTB register
UiC1 register TI bit
Transferred from UiTB register to UARTi transmit register
CTSi
"H"
"L"
TCLK
Stopped pulsing because CTSi = "H"
Stopped pulsing because the TE bit = "0"
CLKi
TxDi
D0 D1 D2 D3 D4 D5 D6 D7
UiC0 register TXEPT bit
SiTIC register IR bit
Cleared to "0" when interrupt request is accepted, or cleared to "0" in a program

$T_c = T_{CLK} = 2(n + 1) / fj$
fj: frequency of UiBRG count source (f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO})
n: value set to UiBRG register
i: 0 to 2

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 ($\overline{CTS}/\overline{RTS}$ enabled), CRS bit = 0 ($\overline{CTS}$ 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)

UiC1 register RE bit
"1"
"0"
UiC1 register TE bit
Write dummy data to UiTB register
UiC1 register TI bit
Transferred from UiTB register to UARTi transmit register
RTSi
"H"
"L"
Even if the reception is completed, the RTS does not change. The RTS becomes "L" when the RI bit changes to "0" from "1".
1 / fEXT
CLKi
Receive data is taken in
RxDi
D0 D1 D2 D3 D4 D5 D6 D7 D0 D1 D2 D3 D4 D5
Transferred from UARTi receive register to UiRB register
Read out from UiRB register
UiC1 register RI bit
SiTIC register IR bit
Cleared to "0" when interrupt request is accepted, or cleared to "0" in a program

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 ($\overline{CTS}/\overline{RTS}$ enabled), CRS bit = 1 ($\overline{RTS}$ 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)

f_{EXT}: frequency of external clock

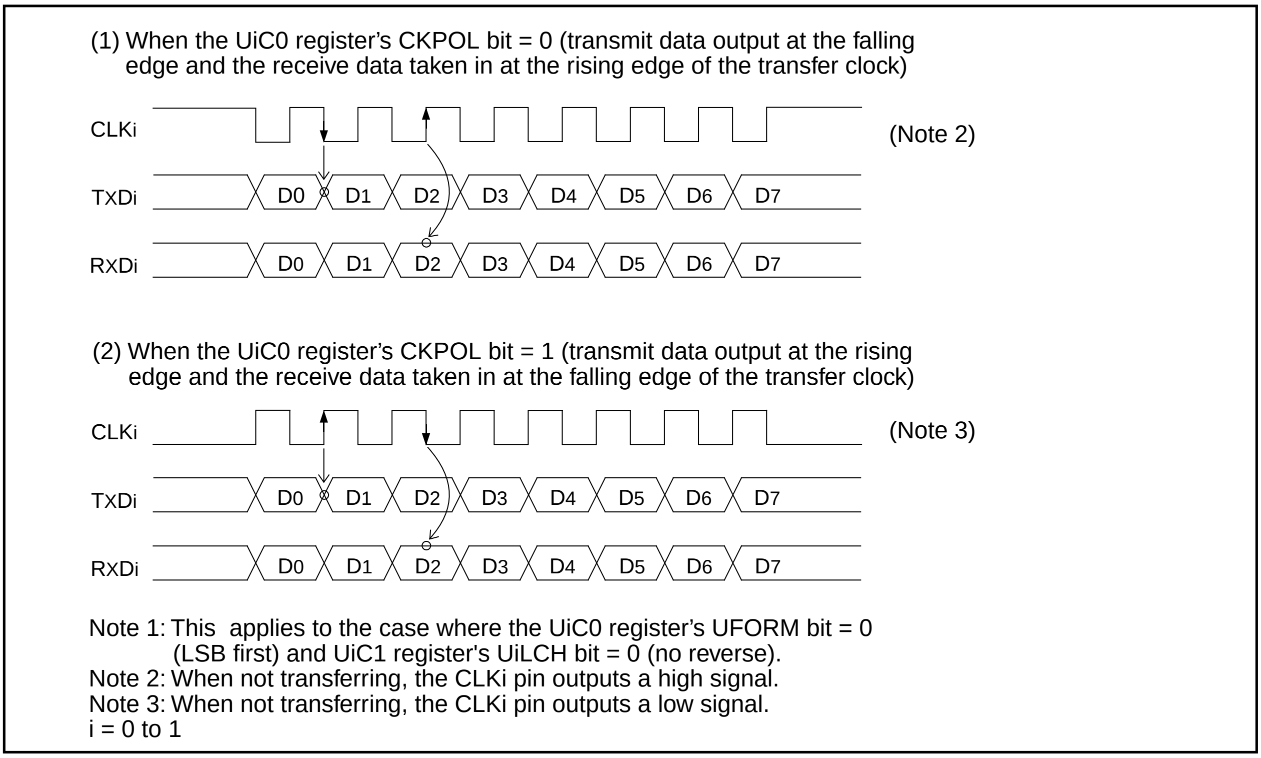
Make sure the following conditions are met when input to the CLKi pin before receiving data is high:

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

Figure 1.14.1. Transmit and Receive Operation

(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.



(1) When 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)

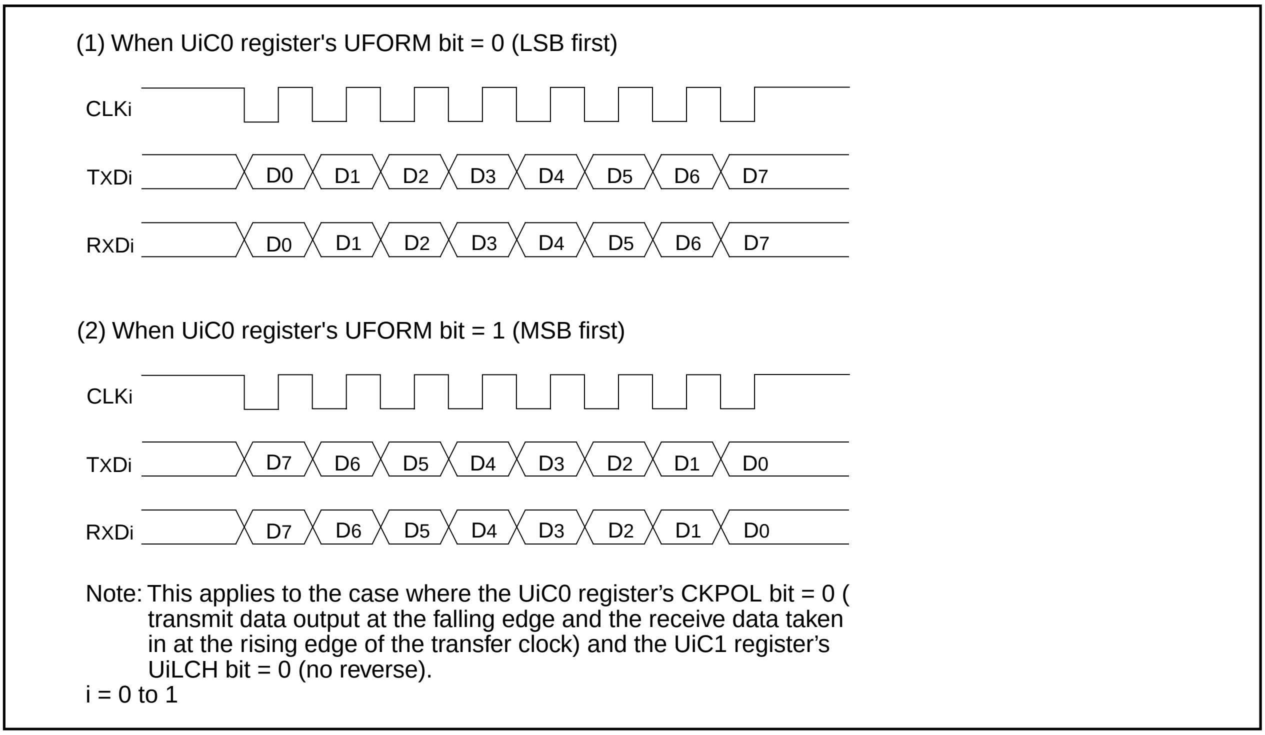
(2) When the UiC0 register's CKPOL bit = 1 (transmit data output at the rising edge and the receive data taken in at the falling edge of the transfer clock)

Note 1: This applies to the case where the UiC0 register's UFORM bit = 0 (LSB first) and UiC1 register's UiLCH bit = 0 (no reverse).
Note 2: When not transferring, the CLKi pin outputs a high signal.
Note 3: When not transferring, the CLKi pin outputs a low signal.
i = 0 to 1

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.



(1) When UiC0 register's UFORM bit = 0 (LSB first)

(2) When UiC0 register's UFORM bit = 1 (MSB first)

Note: This applies to the case where 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) and the UiC1 register's UiLCH bit = 0 (no reverse).
i = 0 to 1

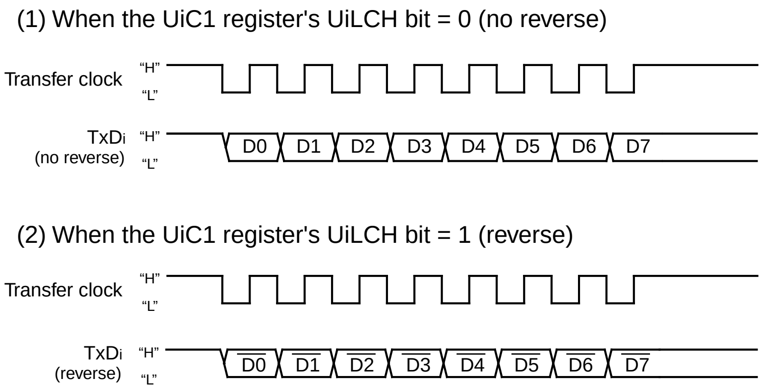
Figure 1.14.3. Transfer Format

(c) Continuous Receive 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.

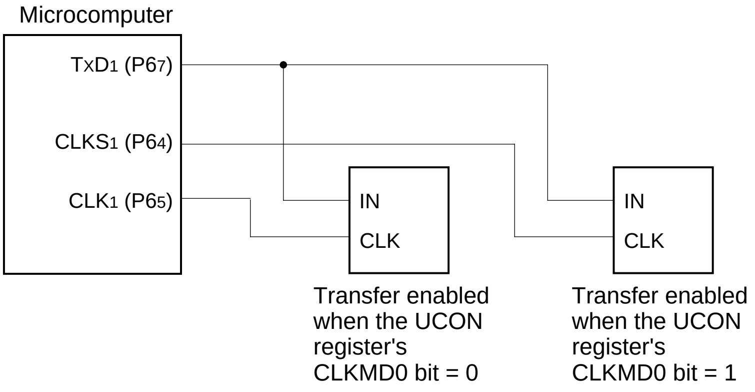


Note: This applies to the case where the UiC0 register's CKPOL bit = 0 (transmit data output at the falling edge and the receive data taken in at the rising edge of the transfer clock) and the UFORM bit = 0 (LSB first).
i = 0 to 1

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.


Microcomputer
TxD1 (P67)
CLKS1 (P64)
CLK1 (P65)
IN
CLK
IN
CLK
Transfer enabled when the UCON register's CLKMD0 bit = 0
Transfer enabled when the UCON register's CLKMD0 bit = 1

Note: This applies to the case where the U1MRregister's CKDIR bit = 0 (internal clock) and the UCON register's CLKMD1 bit = 1 (transfer clock output from multiple pins).

Figure 1.14.5. Transfer Clock Output From Multiple Pins

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

This function separates $\overline{\mathrm{CTS}}_0/\overline{\mathrm{RTS}}_0$, outputs $\overline{\mathrm{RTS}}_0$ from the $P6_0$ pin, and accepts as input the $\overline{\mathrm{CTS}}_0$ from the $P6_4$ pin. To use this function, set the register bits as shown below.

- U0C0 register's CRD bit = 0 (enables UART0 $\overline{\mathrm{CTS}}/\overline{\mathrm{RTS}}$)
- U0C0 register's CRS bit = 1 (outputs UART0 $\overline{\mathrm{RTS}}$)
- U1C0 register's CRD bit = 0 (enables UART1 $\overline{\mathrm{CTS}}/\overline{\mathrm{RTS}}$)
- U1C0 register's CRS bit = 0 (inputs UART1 $\overline{\mathrm{CTS}}$)
- UCON register's RCSP bit = 1 (inputs $\overline{\mathrm{CTS}}_0$ from the $P6_4$ pin)
- UCON register's CLKMD1 bit = 0 ($CLKS_1$ not used)

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

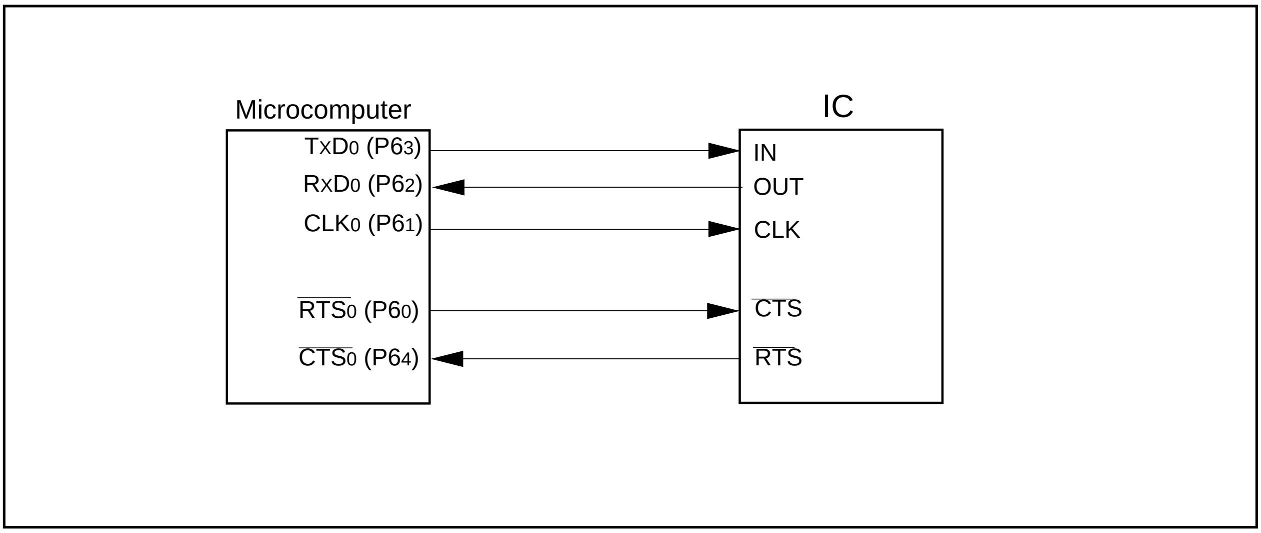

Microcomputer
IC
TxD_0 ($P6_3$)
RxD_0 ($P6_2$)
CLK_0 ($P6_1$)
$\overline{\mathrm{RTS}}_0$ ($P6_0$)
$\overline{\mathrm{CTS}}_0$ ($P6_4$)
IN
OUT
CLK
$\overline{\mathrm{CTS}}$
$\overline{\mathrm{RTS}}$

Figure 1.14.6. $\overline{\mathrm{CTS}}/\overline{\mathrm{RTS}}$ Separat 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. Tables 1.15.1 lists the specifications of the UART mode.

Table 1.15.1. UART Mode Specifications

Item	Specification
Transfer data format	• Character bit (transfer data): Selectable from 7, 8 or 9 bits • Start bit: 1 bit • Parity bit: Selectable from odd, even, or none • Stop bit: Selectable from 1 or 2 bits
Transfer clock	• UiMR(i=0 to 2) register's CKDIR bit = 0 (internal clock) : fj/ 16(n+1) fj = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}. n: Setting value of UiBRG register 00_{16} to FF_{16} • CKDIR bit = "1" (external clock) : f_{EXT}/16(n+1) (Note 3) f_{EXT}: Input from CLKi pin. n :Setting value of UiBRG register 00_{16} to FF_{16}
Transmission, reception control	• Selectable from $\overline{\text{CTS}}$ function, $\overline{\text{RTS}}$ function or $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ function disable
Transmission start condition	• Before transmission can start, the following requirements must be met – The TE bit of UiC1 register= 1 (transmission enabled) – The TI bit of UiC1 register = 0 (data present in UiTB register) – If $\overline{\text{CTS}}$ function is selected, input on the $\overline{\text{CTS}}$i pin = "L"
Reception start condition	• Before reception can start, the following requirements must be met – The RE bit of UiC1 register= 1 (reception enabled) – Start bit detection
Interrupt request generation timing	• For transmission, one of the following conditions can be selected – 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) – 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 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 • Framing error This error occurs when the number of stop bits set is not detected • 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 • Error sum flag This flag is set (= 1) when any of the overrun, framing, and parity errors is encountered
Select function	• LSB first, MSB first selection Whether to start sending/receiving data beginning with bit 0 or beginning with bit 7 can be selected • Serial data logic switch This function reverses the logic of the transmit/receive data. The start and stop bits are not reversed. • TxD, RxD I/O polarity switch This function reverses the polarities of hte TxD pin output and RxD pin input. The logic levels of all I/O data is reversed. • Separate $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ pins (UART0) $\overline{\text{CTS}_0}$ and $\overline{\text{RTS}_0}$ are input/output from separate pins

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	---	Set a transfer rate
UiMR	SMD2 to SMD0	Set these bits to ‘100$_2$’ when transfer data is 7 bits long Set these bits to ‘101$_2$’ when transfer data is 8 bits long Set these bits to ‘110$_2$’ 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 $\overline{\text{CTS}}$ or $\overline{\text{RTS}}$ to use
	TXEPT	Transmit register empty flag
	CRD	Enable or disable the $\overline{\text{CTS}}$ or $\overline{\text{RTS}}$ 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 $\overline{\text{CTS}_0}$ signal from the P6$_4$ 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 P6$_4$ 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

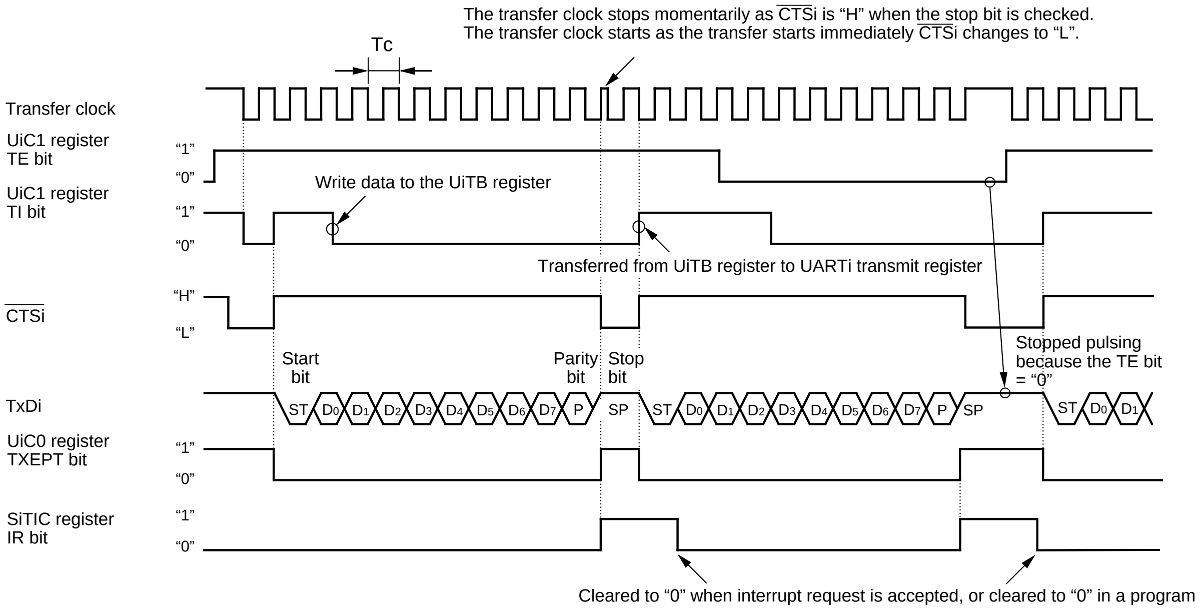
Pin name	Function	Method of selection
TxDi (i = 0 to 2) (P6$_3$, P6$_7$, P7$_0$)	Serial data output	(Outputs dummy data when performing reception only)
RxDi (P6$_2$, P6$_6$, P7$_1$)	Serial data input	PD6 register’s PD6_2 bit=0, PD6_6 bit=0, PD7 register’s PD7_1 bit=0 (Can be used as an input port when performing transmission only)
CLKi (P6$_1$, P6$_5$)	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
$\overline{\text{CTSi}}$/$\overline{\text{RTSi}}$ (P6$_0$, P6$_4$, P7$_3$)	$\overline{\text{CTS}}$ 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
	$\overline{\text{RTS}}$ 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. P6$_4$ Pin Functions

Pin function	Bit set value				
	U1C0 register		UCON register		PD6 register
	CRD	CRS	RCSP	CLKMD1	PD6_4
P6$_4$	1	—	0	0	Input: 0, Output: 1
$\overline{\text{CTS}_1}$	0	0	0	0	0
$\overline{\text{RTS}_1}$	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” ($\overline{\text{CTS}_0}$/$\overline{\text{RTS}_0}$ enabled) and the U0C0 register’s CRS bit to “1” ($\overline{\text{RTS}_0}$ selected).

(1) Example of transmit timing when transfer data is 8 bits long (parity enabled, one stop bit)

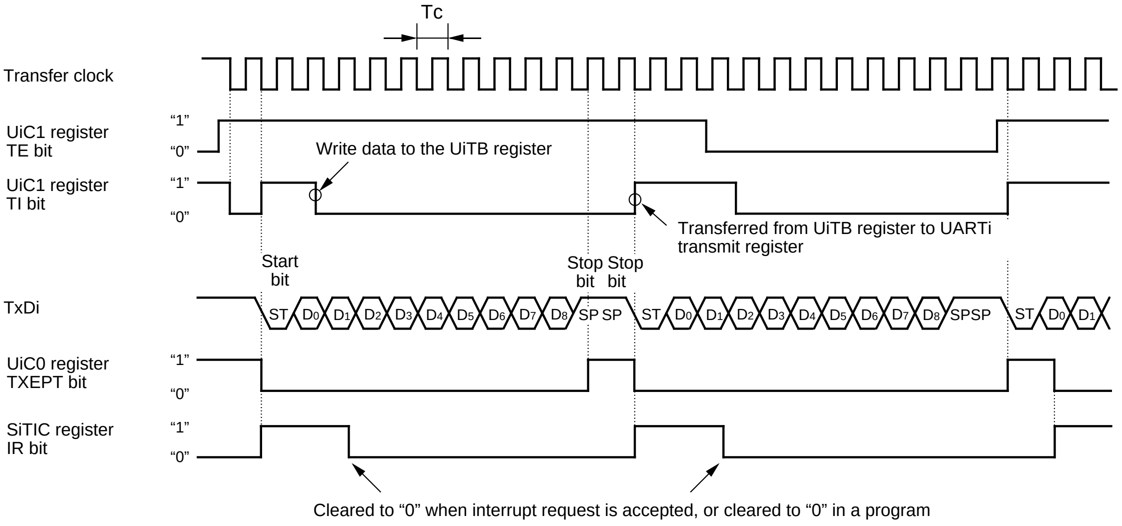

The transfer clock stops momentarily as CTSi is "H" when the stop bit is checked.
The transfer clock starts as the transfer starts immediately CTSi changes to "L".
Tc
Transfer clock
UiC1 register
TE bit
"1"
"0"
Write data to the UiTB register
UiC1 register
TI bit
"1"
"0"
Transferred from UiTB register to UARTi transmit register
CTSi
"H"
"L"
Start
bit
Parity
bit
Stop
bit
Stopped pulsing
because the TE bit
= "0"
TxDi
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP ST D0 D1 D2 D3 D4 D5 D6 D7 P SP ST D0 D1
UiC0 register
TXEPT bit
"1"
"0"
SiTIC register
IR bit
"1"
"0"
Cleared to "0" when interrupt request is accepted, or cleared to "0" in a program

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

- UiMR register PRYE bit = 1 (parity enabled)
- UiMR register STPS bit = 0 (1 stop bit)
- UiC0 register CRD bit = 0 (CTS/RTS enabled), CRS bit = 0 (CTS selected)
- UiRS bit = 1 (an interrupt request occurs when transmit completed):
  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

Tc = 16 (n + 1) / fj or 16 (n + 1) / f_{EXT}
fj : frequency of UiBRG count source (f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO})
f_{EXT} : frequency of UiBRG count source (external clock)
n : value set to UiBRG
i: 0 to 2

(2) Example of transmit timing when transfer data is 9 bits long (parity disabled, two stop bits)


Tc
Transfer clock
UiC1 register
TE bit
"1"
"0"
Write data to the UiTB register
UiC1 register
TI bit
"1"
"0"
Transferred from UiTB register to UARTi
transmit register
Start
bit
Stop
bit
Stop
bit
TxDi
ST D0 D1 D2 D3 D4 D5 D6 D7 D8 SP SP ST D0 D1 D2 D3 D4 D5 D6 D7 D8 SP SP ST D0 D1
UiC0 register
TXEPT bit
"1"
"0"
SiTIC register
IR bit
"1"
"0"
Cleared to "0" when interrupt request is accepted, or cleared to "0" in a program

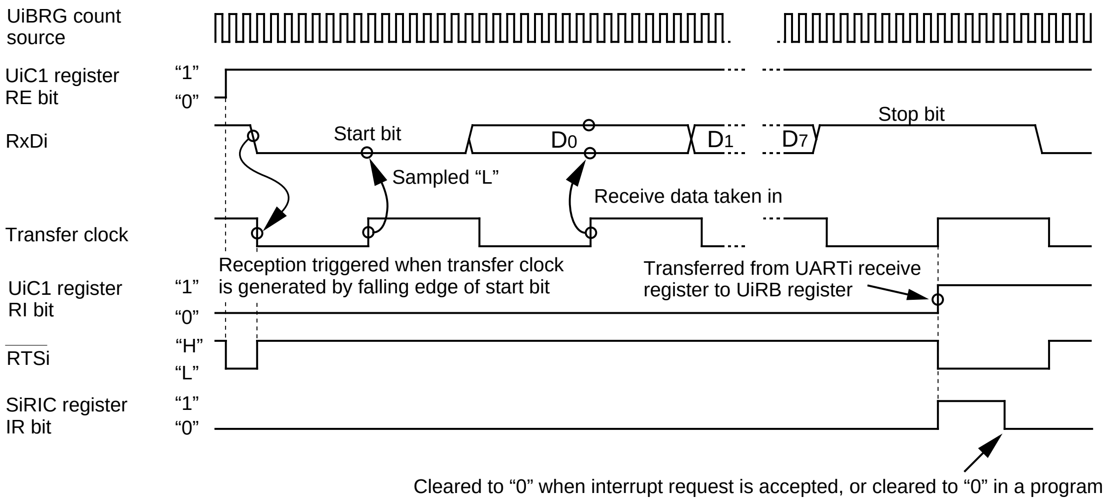
The above timing diagram applies to the case where the register bits are set as follows:

- UiMR register PRYE bit = 0 (parity disabled)
- UiMR register STPS bit = 1 (2 stop bits)
- UiC0 register CRD bit = 1 (CTS/RTS disabled)
- UiRS bit = 0 (an interrupt request occurs when 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

Tc = 16 (n + 1) / fj or 16 (n + 1) / f_{EXT}
fj : frequency of UiBRG count source (f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO})
f_{EXT} : frequency of UiBRG count source (external clock)
n : value set to UiBRG
i: 0 to 2

Figure 1.15.1. Transmit Operation

• Example of receive timing when transfer data is 8 bits long (parity disabled, one stop bit)


UiBRG count source
UiC1 register RE bit
"1"
"0"
RxDi
Start bit
D0
D1
D7
Stop bit
Sampled "L"
Receive data taken in
Transfer clock
Reception triggered when transfer clock is generated by falling edge of start bit
Transferred from UARTi receive register to UiRB register
UiC1 register RI bit
"1"
"0"
RTSi
"H"
"L"
SiRIC register IR bit
"1"
"0"
Cleared to "0" when interrupt request is accepted, or cleared to "0" in a program

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

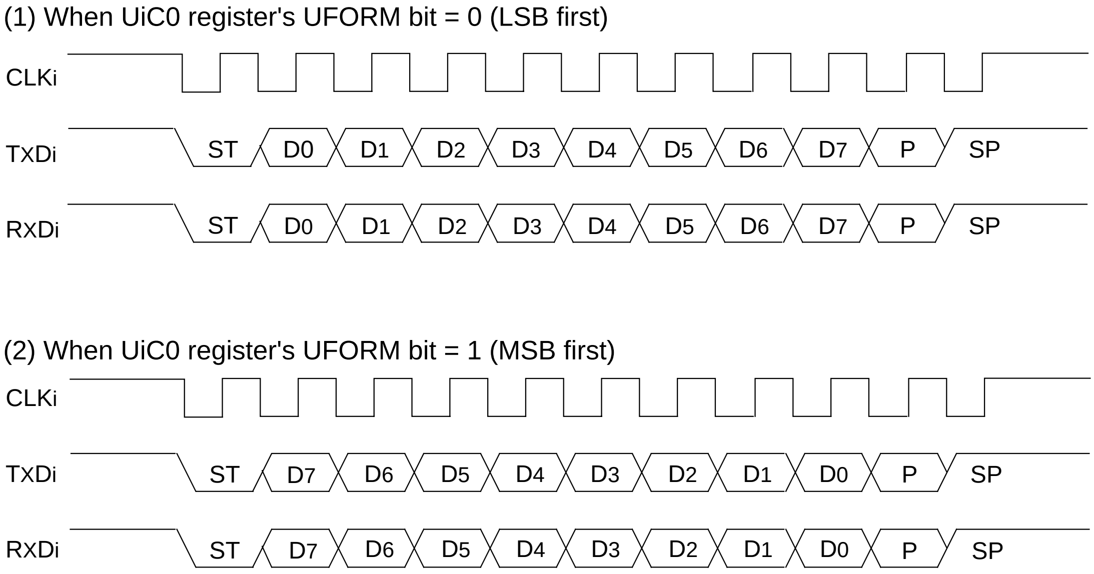
- UiMR register PRYE bit = 0 (parity disabled)
- UiMR register STPS bit = 0 (1 stop bit)
- UiC0 register CRD bit = 0 (CTSi/RTSi enabled), CRS bit = 1 (RTSi selected)

i = 0 to 2

Figure 1.15.2. Receive Operation

(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.


(1) When UiC0 register's UFORM bit = 0 (LSB first)

(2) When UiC0 register's UFORM bit = 1 (MSB first)

CLKi
TxDi
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP
RxDi
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP
CLKi
TxDi
ST D7 D6 D5 D4 D3 D2 D1 D0 P SP
RxDi
ST D7 D6 D5 D4 D3 D2 D1 D0 P SP

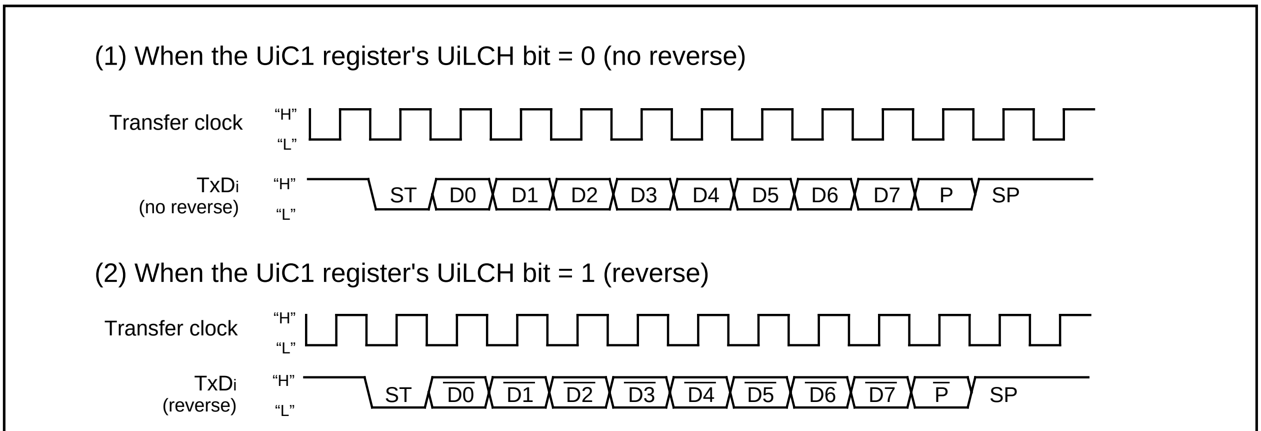
Note: This applies to the case where the UiC0 register's CKPOL bit = 0 (transmit data output at the falling edge and the receive data taken in at the rising edge of the transfer clock), the UiC1 register's UiLCH bit = 0 (no reverse), UiMR register's STPS bit = 0 (1 stop bit) and UiMR register's PRYE bit = 1 (parity enabled).

ST : Start bit
P : Parity bit
SP : Stop bit
i = 0 to 2

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.


(1) When the UiC1 register's UiLCH bit = 0 (no reverse)
Transfer clock
"H"
"L"
TxDi
(no reverse)
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP
(2) When the UiC1 register's UiLCH bit = 1 (reverse)
Transfer clock
TxDi
(reverse)
ST $\overline{D0}$ $\overline{D1}$ $\overline{D2}$ $\overline{D3}$ $\overline{D4}$ $\overline{D5}$ $\overline{D6}$ $\overline{D7}$ $\overline{P}$ SP

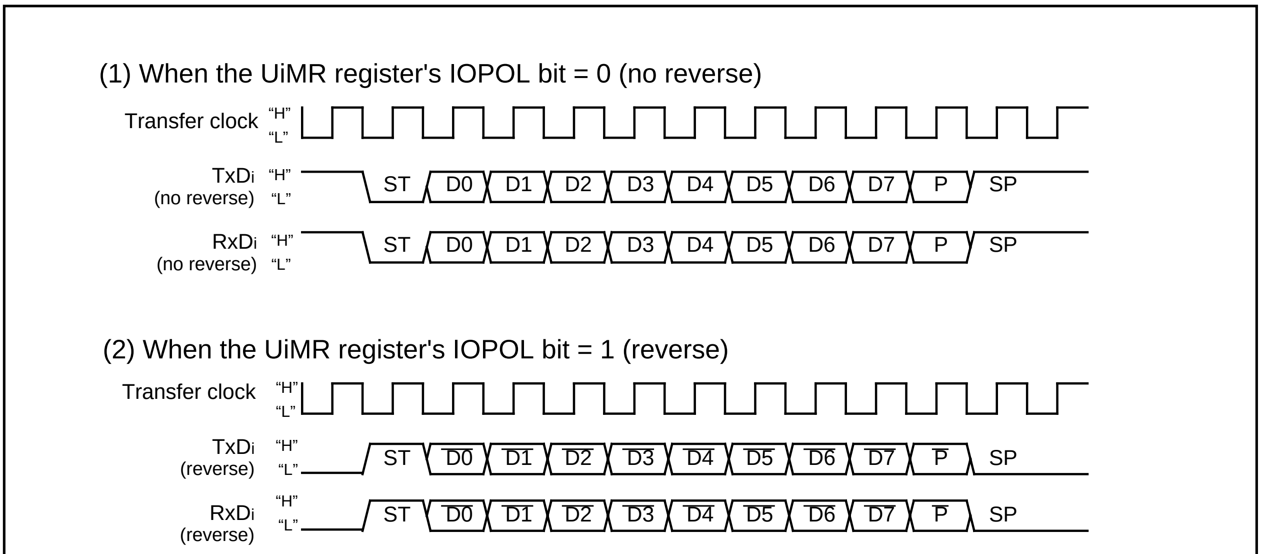
Note: This applies to the case where the UiC0 register's CKPOL bit = 0 (transmit data output at the falling edge of the transfer clock), the UiC0 register's UFORM bit = 0 (LSB first), the UiMR register's STPS bit = 0 (1 stop bit) and UiMR register's PRYE bit = 1 (parity enabled).

ST : Start bit
P : Parity bit
SP : Stop bit
i = 0 to 2

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 inversed. Figure 1.15.5 shows the TxD pin output and RxD pin input polarity inverse.


(1) When the UiMR register's IOPOL bit = 0 (no reverse)
Transfer clock
"H"
"L"
TxDi
(no reverse)
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP
RxDi
(no reverse)
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP
(2) When the UiMR register's IOPOL bit = 1 (reverse)
Transfer clock
TxDi
(reverse)
ST $\overline{D0}$ $\overline{D1}$ $\overline{D2}$ $\overline{D3}$ $\overline{D4}$ $\overline{D5}$ $\overline{D6}$ $\overline{D7}$ $\overline{P}$ SP
RxDi
(reverse)
ST $\overline{D0}$ $\overline{D1}$ $\overline{D2}$ $\overline{D3}$ $\overline{D4}$ $\overline{D5}$ $\overline{D6}$ $\overline{D7}$ $\overline{P}$ SP

Note: This applies to the case where the UiC0 register's UFORM bit = 0 (LSB first), the UiMR register's STPS bit = 0 (1 stop bit) and the UiMR register's PRYE bit = 1 (parity enabled).

ST : Start bit
P : Parity bit
SP : Stop bit
i = 0 to 2

Figure 1.15.5. TxD and RxD I/O Polarity Inverse

(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 P6_0 pin, and accepts as input the $\overline{\text{CTS}_0}$ from the P6_4 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 P6_4 pin)
- UCON register's CLKMD1 bit = 0 (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.

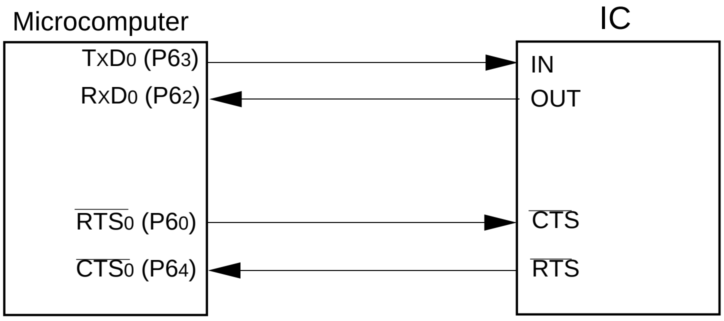

Microcomputer
IC
TxD_0 (P6_3)
RxD_0 (P6_2)
$\overline{\text{RTS}_0}$ (P6_0)
$\overline{\text{CTS}_0}$ (P6_4)
IN
OUT
$\overline{\text{CTS}}$
$\overline{\text{RTS}}$

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

Special Mode 1 (I^{2}C mode)

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

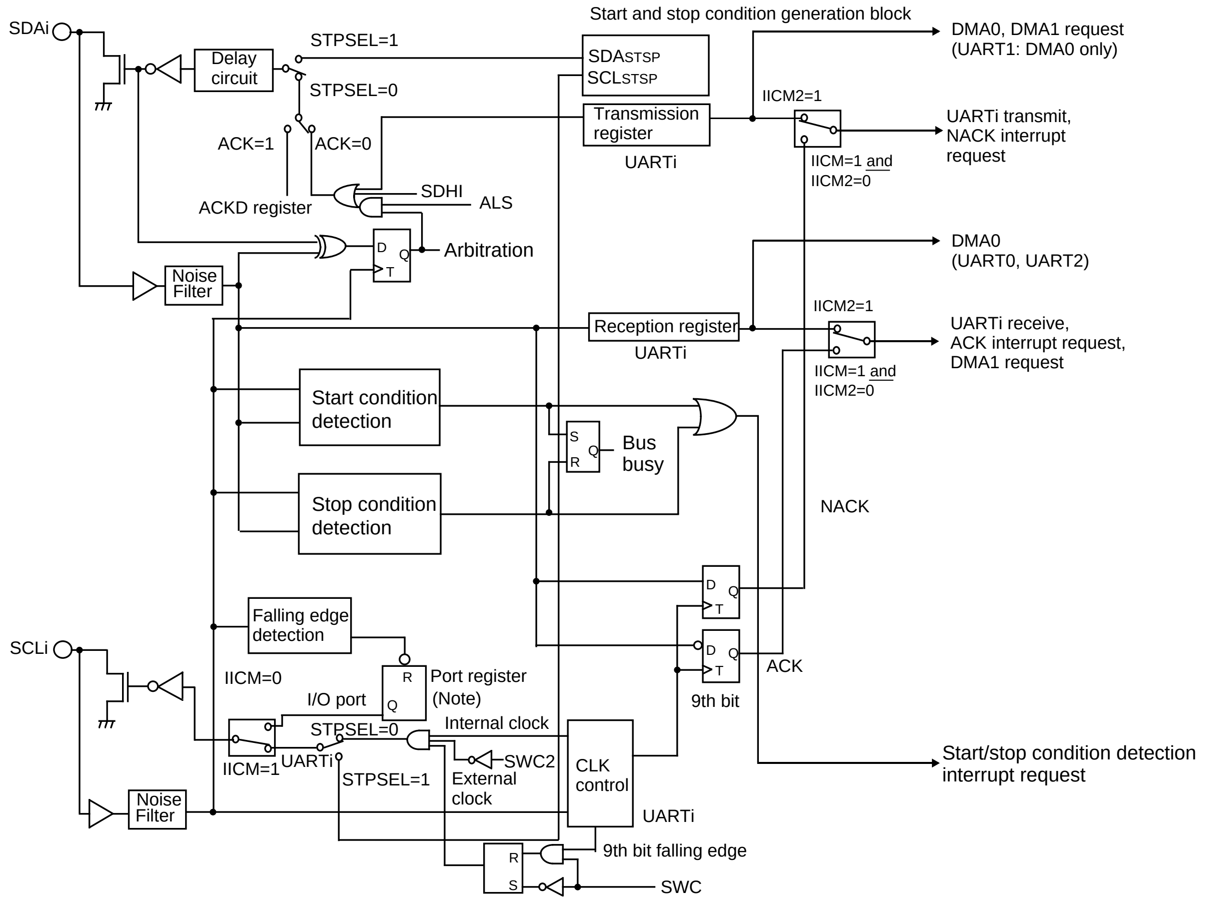
As shown in Table 1.16.3, the microcomputer is placed in I^{2}C mode by setting the SMD2 to SMD0 bits to ‘010_2’ and the IICM bit to “1”. Because SDAi transmit output has a delay circuit attached, SDAi output does not change state until SCLi goes low and remains stably low.

Table 1.16.1. I^{2}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) : fj/ 2(n+1) fj = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}. n: Setting value of UiBRG register 00_{16} to FF_{16} • 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 next 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.



This diagram applies to the case where the UiMR register's SMD2 to SMD0 bits = 010_2 and the UiSMR register's IICM bit = 1.

IICM: UiSMR register bit
IICM2: UiSMR2 register bit

i=0 to 2

Note: When the IICM bit =1, the pins can be read even if the direction bit = 1 (output).

Figure 1.16.1. I^{2}C Mode Block Diagram

Table 1. 16. 2. Registers to Be Used and Settings in I²C Mode (1) (Continued)

Register	Bit	Function	
		Master	Slave
UiTB[3]	0 to 7	Set transmission data	Set transmission data
UiRB[3]	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	---	Set a transfer rate	Invalid
UiMR[3]	SMD2 to SMD0	Set to ‘010$_2$’	Set to ‘010$_2$’
	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”[2]	Set to “1”[2]
	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[1]	Invalid	Invalid
	U2RRM[1], 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
	7	Set to “0”	Set to “0”
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 I²C mode.

Table 1. 16. 3. Registers to Be Used and Settings in I²C 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^{2}C Mode Functions

Function	Clock synchronous serial I/O mode (SMD2 to SMD0 = 001_2, IICM = 0)	I^{2}C mode (SMD2 to SMD0 = 010_2, 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 (Note 1)(Refer to Fig 1.16.2)	—	Start condition detection or stop condition detection (Refer to Fig 1.20.4)			
Factor of interrupt number 15, 17 and 19 (Note 1)(Refer to Fig 1.16.2)	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 (Note 1)(Refer to Fig 1.16.2)	UARTi reception When 8th bit received CKPOL = 0 (rising edge) CKPOL = 1 (falling edge)	Acknowledgment detection (ACK) Rising edge of SCLi 9th bit		UARTi transmission 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	RxDi input	SCLi input/output			
Functions of $P6_1$, $P6_5$ and $P7_2$ pins	CLKi input or output selected	— (Cannot be used in I^{2}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^{2}C mode (Note 2)			
Initial and end values of SCLi	—	H	L	H	L
DMA1 factor (Refer to Fig 1.16.2)	UARTi reception	Acknowledgment detection (ACK)		UARTi reception Falling edge of SCLi 9th bit	
Store received data	1st to 8th bits are stored in UiRB register bit 0 to bit 7	1st to 8th bits are stored in UiRB register bit 7 to bit 0		1st to 7th bits are stored in UiRB register bit 6 to bit 0, with 8th bit stored in UiRB register bit 8	1st to 7th bits are stored in UiRB register bit 6 to bit 0, with 8th bit stored in UiRB register bit 8 1st to 8th bits are stored in UiRB register bit 7 to bit 0 (Note 3)
Read received data	UiRB register status is read directly as is				Read UiRB register Bit 6 to bit 0 as bit 7 to bit 1, and bit 8 as bit 0 (Note 4)

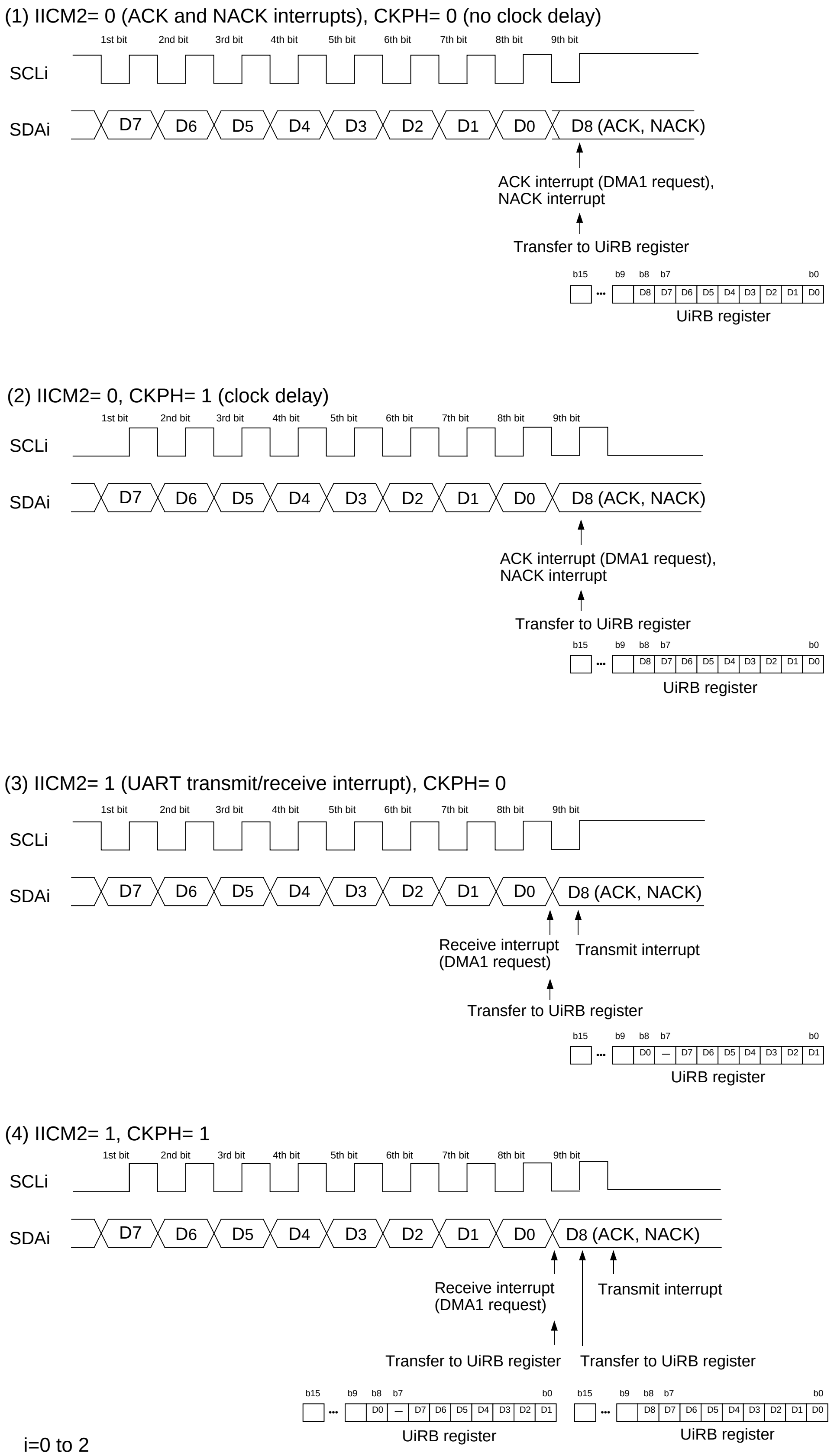
i = 0 to 2

Note 1: To change the interrupt sources from one to another, follow the procedure described below.
1. Disable the interrupt of the corresponding interrupt number to be changed.
2. Change interrupt sources from one to another.
3. Set the IR bit for the corresponding interrupt number to 0 (no interrupt request).
4. Set the IPL2 to IPL0 bits for the corresponding interrupt number.

Note 2: Set the initial value of SDAi output while the UiMR register's SMD2 to SMD0 bits = '000 $_2$' (serial I/O disabled).

Note 3: Second data transfer to UiRB register (Rising edge of SCLi 9th bit)

Note 4. First data transfer to UiRB register (Falling edge of SCLi 9th bit)


(1) IICM2= 0 (ACK and NACK interrupts), CKPH= 0 (no clock delay)
1st bit
2nd bit
3rd bit
4th bit
5th bit
6th bit
7th bit
8th bit
9th bit
SCLi
SDAi
D7
D6
D5
D4
D3
D2
D1
D0
D8 (ACK, NACK)
ACK interrupt (DMA1 request),
NACK interrupt
Transfer to UiRB register
b15
b9
b8
b7
b0
D8 D7 D6 D5 D4 D3 D2 D1 D0
UiRB register
(2) IICM2= 0, CKPH= 1 (clock delay)
SCLi
SDAi
D8 (ACK, NACK)
ACK interrupt (DMA1 request),
NACK interrupt
Transfer to UiRB register
D8 D7 D6 D5 D4 D3 D2 D1 D0
UiRB register
(3) IICM2= 1 (UART transmit/receive interrupt), CKPH= 0
SCLi
SDAi
D8 (ACK, NACK)
Receive interrupt
(DMA1 request)
Transmit interrupt
Transfer to UiRB register
D0 — D7 D6 D5 D4 D3 D2 D1
UiRB register
(4) IICM2= 1, CKPH= 1
SCLi
SDAi
D8 (ACK, NACK)
Receive interrupt
(DMA1 request)
Transmit interrupt
Transfer to UiRB register
Transfer to UiRB register
D0 — D7 D6 D5 D4 D3 D2 D1
UiRB register
D8 D7 D6 D5 D4 D3 D2 D1 D0
UiRB register

i=0 to 2

This diagram applies to the case where the following condition is met.
- UiMR register CKDIR bit = 0 (Slave selected)

Figure 1.16.2. Transfer to UiRB Register and Interrupt Timing

- **Detection of Start and Stop Condtion**

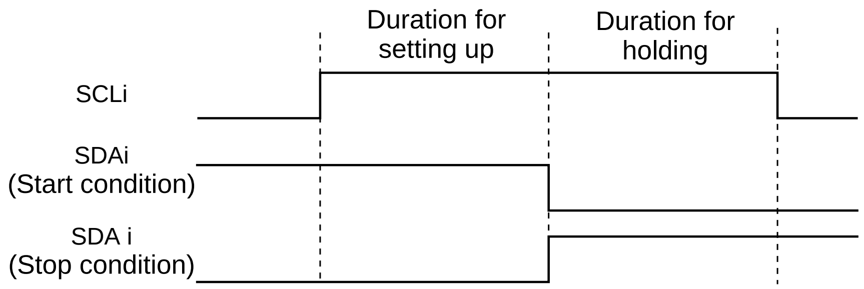
  Whether a start or a stop condition has been detected is determined.

  A start condition-detected interrupt request is generated when the SDAi pin changes state from high to low while the SCLi pin is in the high state. A stop condition-detected interrupt request is generated when the SDAi pin changes state from low to high while the SCLi 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.

3 to 6 cycles < duration for setting-up (Note)

3 to 6 cycles < duration for holding (Note)



i = 0 to 2

Note: When the PCLKR register's PCLK1 bit = 1, this is the cycle number of f_{1SIO}, and the PCLK1 bit = 0, this is the cycle number of f_{2SIO}.

Figure 1.16.3. Detection of Start and Stop Condition

- **Output of Start and Stop Condition**

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

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

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

  A start condition is output by setting the STAREQ bit to "1" and then the UiSMR4 register's STSPSEL bit to "1" (start). Similarly, a restart condition is output by setting the RSTAREQ bit to "1" and then the STSPSEL bit to "1", and a stop condition is output by setting the STPREQ bit to "1" and then the STSPSEL bit to "1".

  Table 1.16.5 and Figure 1.16.4 show the functions of the STSPSEL.

  If start, stop and restart conditions are to be output, make sure no interrupts will occur between the instruction that sets the STAREQ, STPREQ or RSTAREQ bit to "1" and the instruction that sets the STSPSEL bit to "1".

  Also, if a start condition is to be output, make sure the STAREQ bit is set to "1" before setting the STSPSEL bit to "1".

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

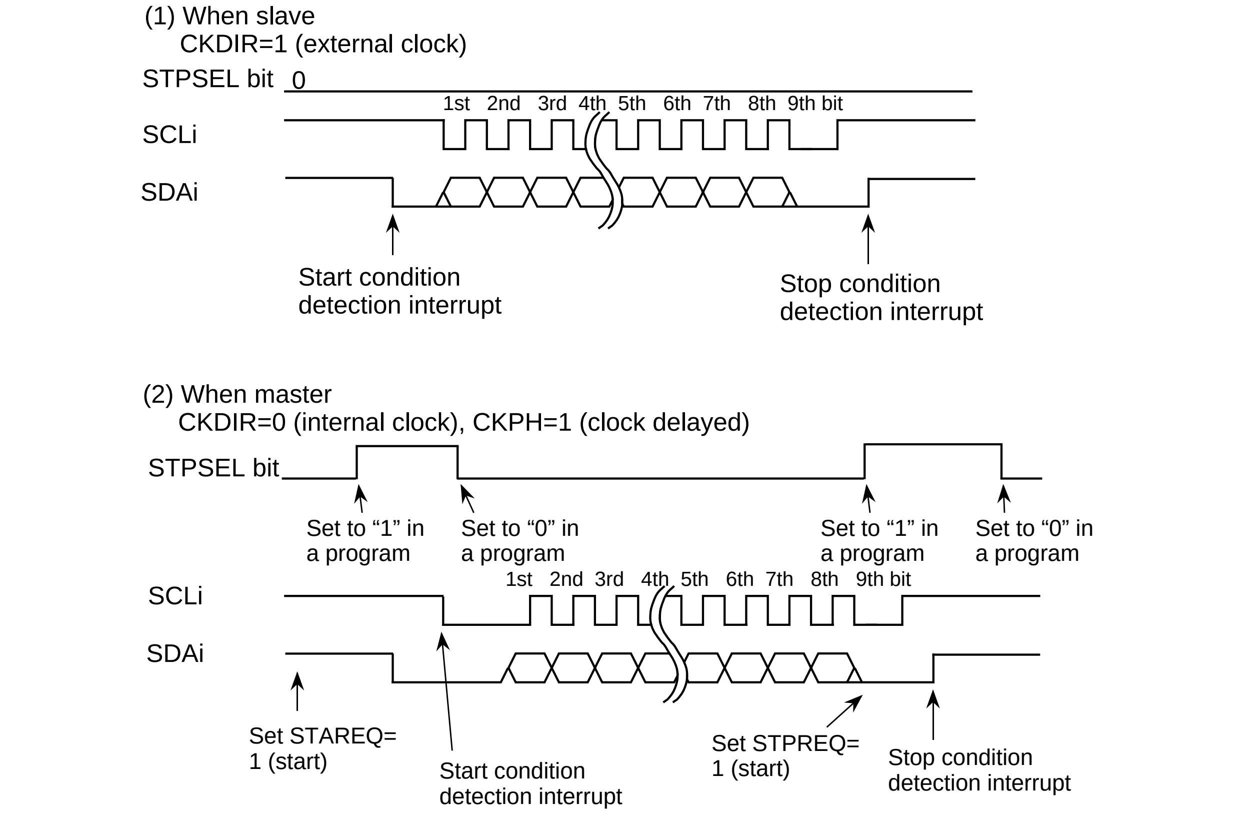

(1) When slave
CKDIR=1 (external clock)
STPSEL bit 0
1st 2nd 3rd 4th 5th 6th 7th 8th 9th bit
SCLi
SDAi
Start condition detection interrupt
Stop condition detection interrupt
(2) When master
CKDIR=0 (internal clock), CKPH=1 (clock delayed)
STPSEL bit
Set to "1" in a program
Set to "0" in a program
Set to "1" in a program
Set to "0" in a program
1st 2nd 3rd 4th 5th 6th 7th 8th 9th bit
SCLi
SDAi
Set STAREQ= 1 (start)
Start condition detection interrupt
Set STPREQ= 1 (start)
Stop condition detection interrupt

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 bytewise, 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 (D_7 to D_0) is sequentially output beginning with D_7. The ninth bit (D_8) is ACK or NACK.

The initial value of SDAi transmit output can only be set when IICM = 1 (I^2C mode) and the UiMR register's SMD2 to SMD0 bits = '000_2' (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 (D_7 to D_0) of received data are stored in the UiRB register bit 7 to bit 0. The 9th bit (D_8) is ACK or NACK.

When the IICM2 bit = 1, the 1st to 7th bits (D_7 to D_1) of received data are stored in the UiRB register bit 6 to bit 0 and the 8th bit (D_0) 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 se 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) : fj/ 2(n+1) fj = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}. n: Setting value of UiBRG register 00_{16} to FF_{16} • 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.

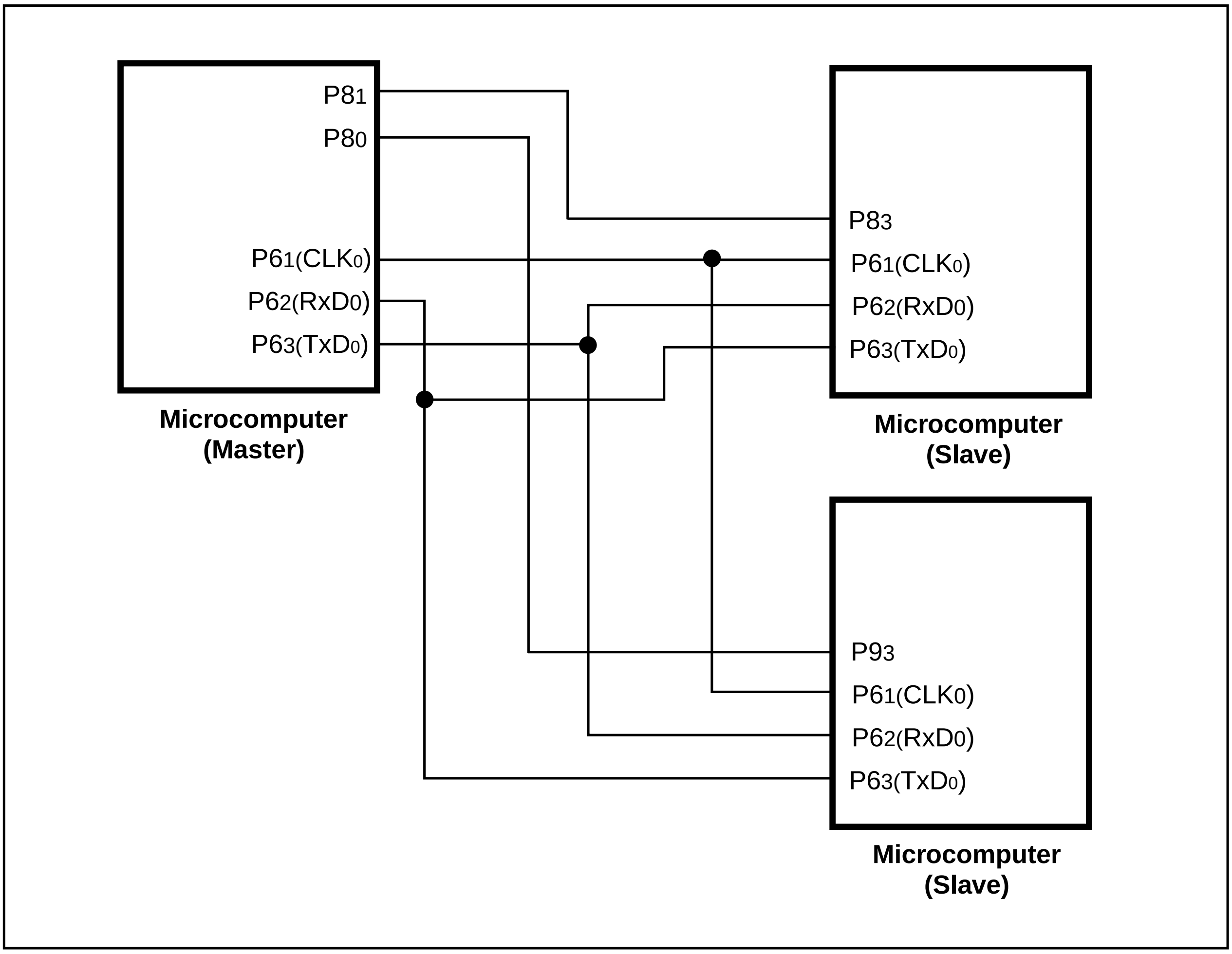


Figure 1.16.5. Serial Bus Communication Control Example (UART0)

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	Overrun error flag
UiBRG	0 to 7	Set a transfer rate
UiMR(Note2)	SMD2 to SMD0	Set to ‘001$_2$’
	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 salves to be communicated.

(a) Master (Internal Clock)

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

(b) Slave (External 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).

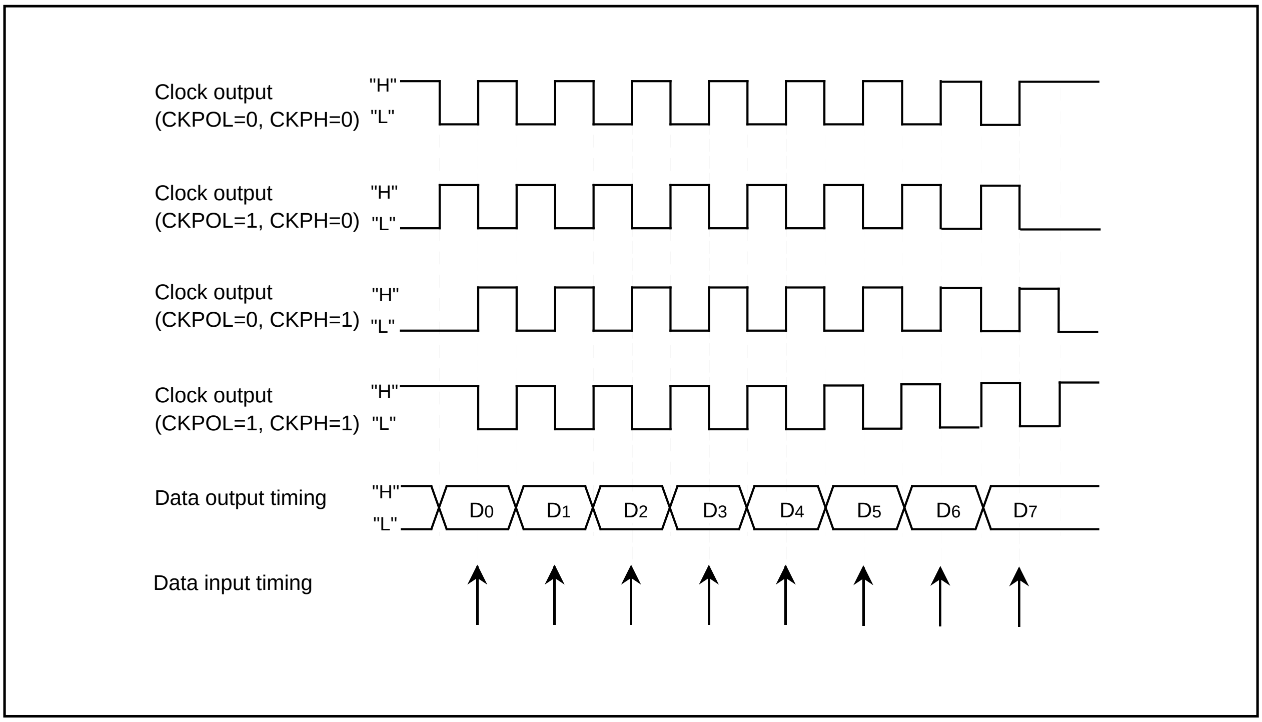


Figure 1.16.6. Transmission and Reception Timing in Master Mode (Internal Clock)

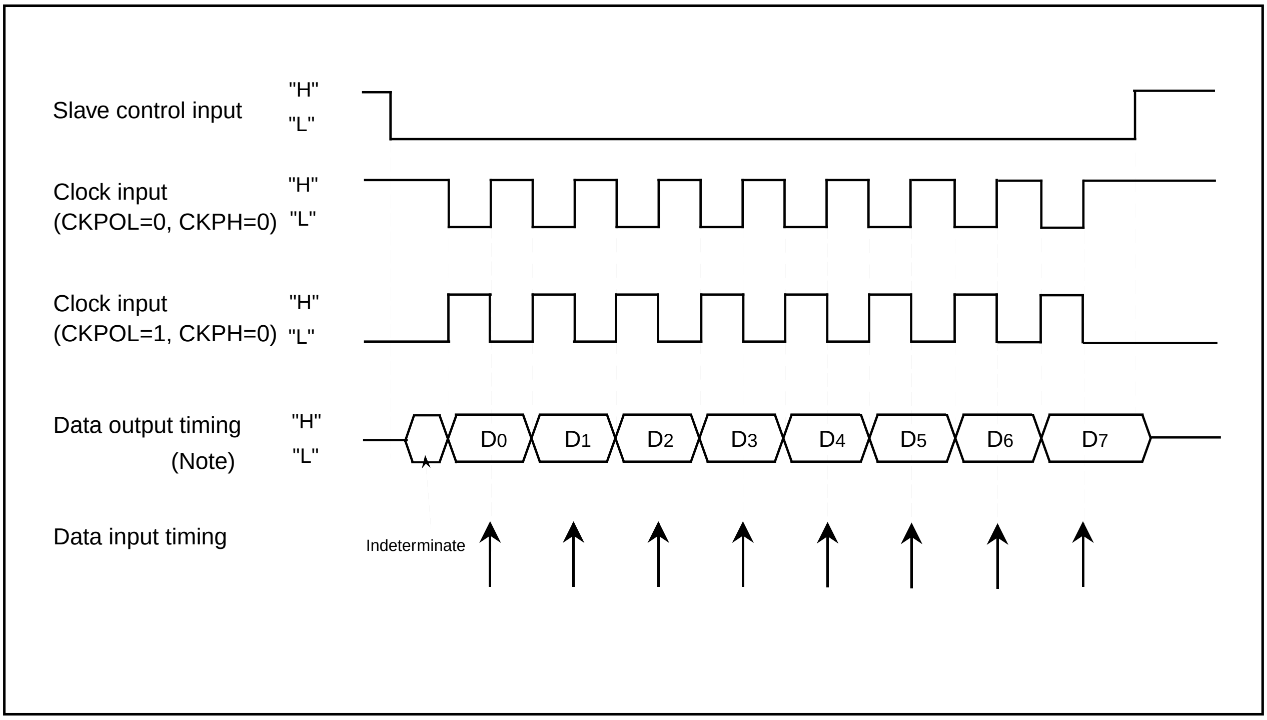


Figure 1.16.7. Transmission and Reception Timing (CKPH=0) in Slave Mode (External Clock)

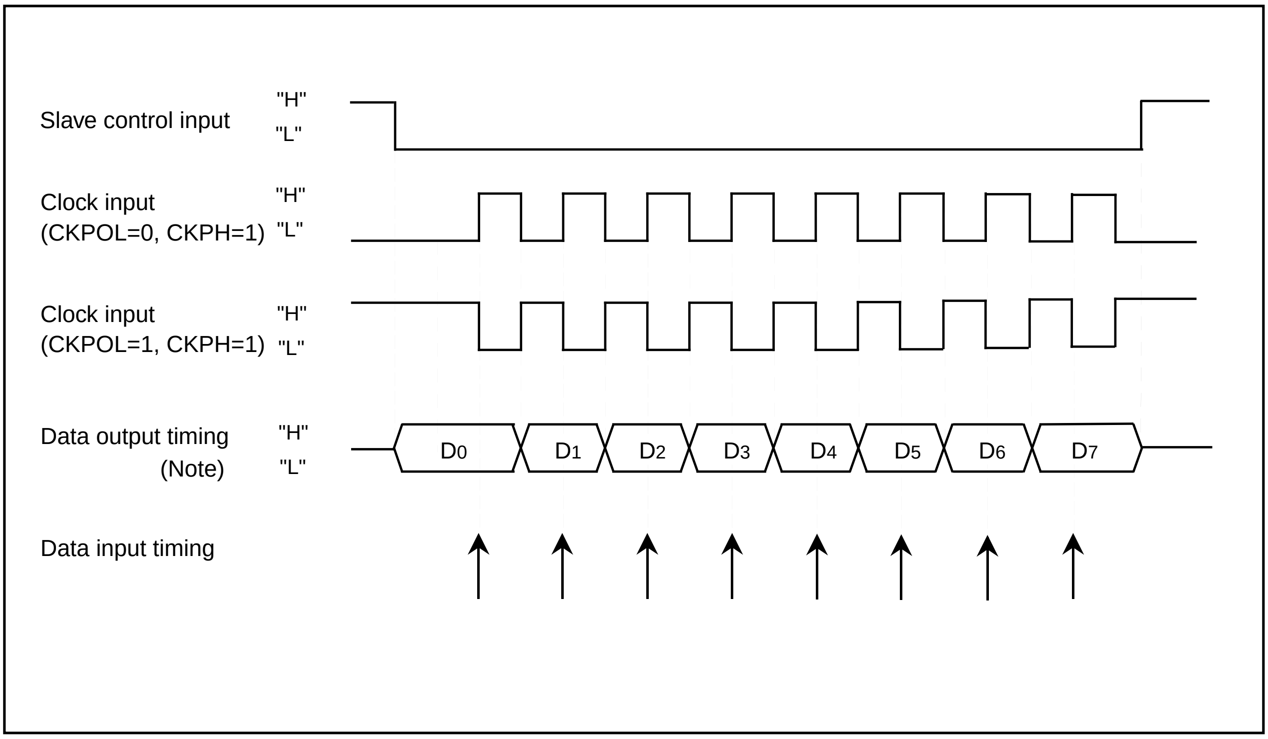


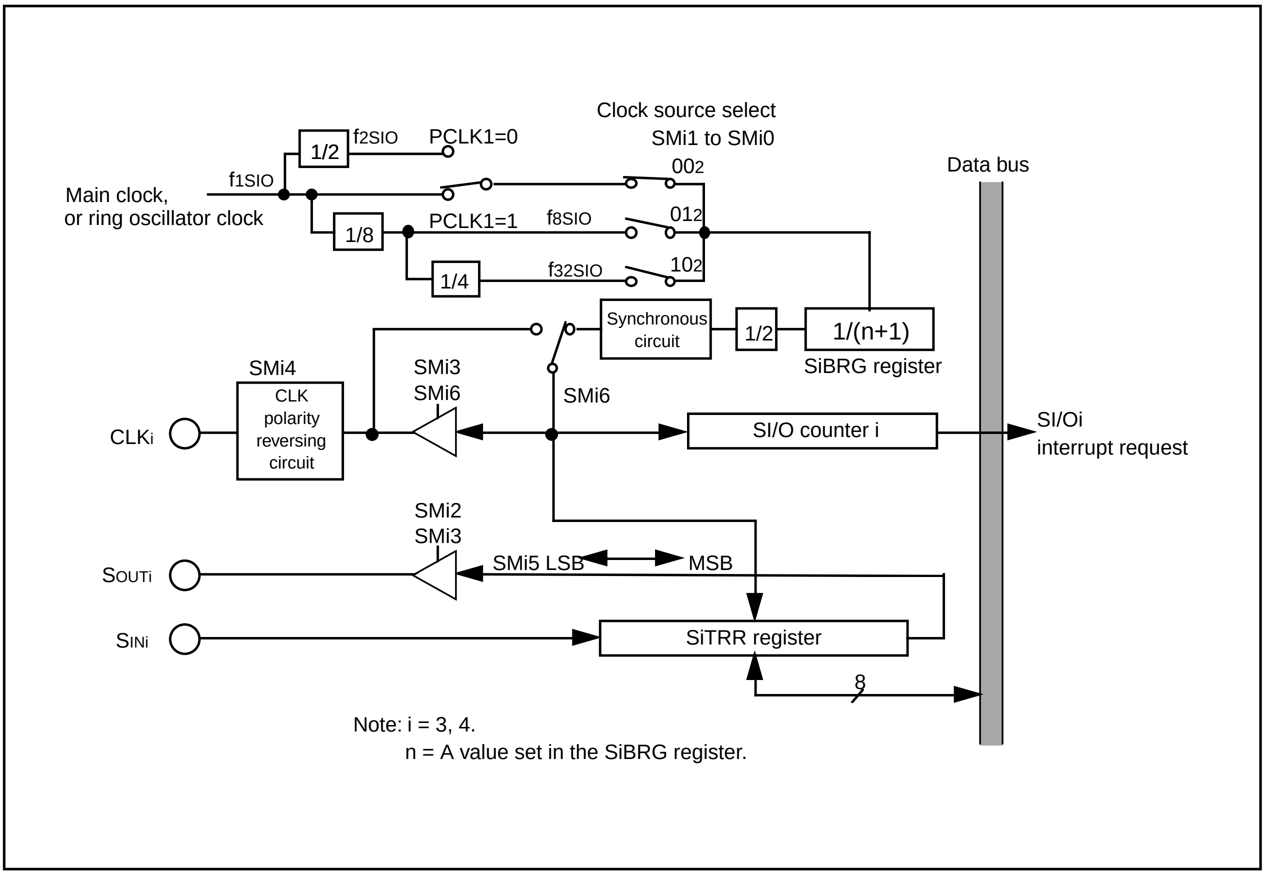
Figure 1.16.8. Transmission and Reception Timing (CKPH=1) in Slave Mode (External Clock)

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 derectly connected to IT800 internally.

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



Note: i = 3, 4.
n = A value set in the SiBRG register.

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

S I/Oi control register (i = 3, 4) (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
S3C	0362_{16}	01000000_{16}
S4C	0366_{16}	01000000_{16}

Bit symbol	Bit name	Description	RW
SMi0	Internal synchronous clock select bit	b1 b0 0 0 : Selecting f_{1SIO} or f_{2SIO} 0 1 : Selecting f_{8SIO} 1 0 : Selecting f_{32SIO} 1 1 : Must not be set.	RW
SMi1			RW
SMi2	SOUTi output disable bit (Note 4)	0 : SOUTi output 1 : SOUTi output disable(high impedance)	RW
SMi3	S I/Oi port select bit	0 : Input/output port 1 : SOUTi output, CLKi function	RW
SMi4	CLK polarity select bit	0 : Transmit data is output at falling edge of transfer clock and receive data is input at rising edge 1 : Transmit data is output at rising edge of transfer clock and receive data is input at falling edge	RW
SMi5	Transfer direction select bit	0 : LSB first 1 : MSB first	RW
SMi6	Synchronous clock select bit	0 : External clock (Note 2) 1 : Internal clock (Note 3)	RW
SMi7	SOUTi initial value set bit	Effective when SMi3 = 0 0 : “L” output 1 : “H” output	RW

Note 1: Make sure this register is written to by the next instruction after setting the PRCR register's PRC2 bit to “1” (write enable).
Note 2: Set the SMi3 bit to “1” (SOUTi output, CLKi function).
Note 3: Set the SMi3 bit to “1” and the corresponding port direction bit to “0” (input mode).
Note 4: Effective when SMi3 bit = 1.

SI/Oi bit rate generator (i = 3, 4) (Notes 1, 2)

b7 b0

Symbol	Address	After reset
S3BRG	0363_{16}	Indeterminate
S4BRG	0367_{16}	Indeterminate

Description	Setting range	RW
Assuming that set value = n, BRGi divides the count source by n + 1	00_{16} to FF_{16}	WO

Note 1: Write to this register while serial I/O is neither transmitting nor receiving.
Note 2: Use MOV instruction to write to this register.

SI/Oi transmit/receive register (i = 3, 4) (Note 1, 2)

b7 b0

Symbol	Address	After reset
S3TRR	0360_{16}	Indeterminate
S4TRR	0364_{16}	Indeterminate

Description	RW
Transmission/reception starts by writing transmit data to this register. After transmission/reception finishes, reception data can be read by reading this register.	RW

Note 1: Write to this register while serial I/O is neither transmitting nor receiving.
Note 2: To receive data, set the corresponding port direction bit for SINi to “0” (input mode).

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) : fj/ 2(n+1) fj = f_{1SIO}, f_{8SIO}, f_{32SIO}. n=Setting value of SiBRG register 00_{16} to FF_{16}. • 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 fucntion	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 tranmitting 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.

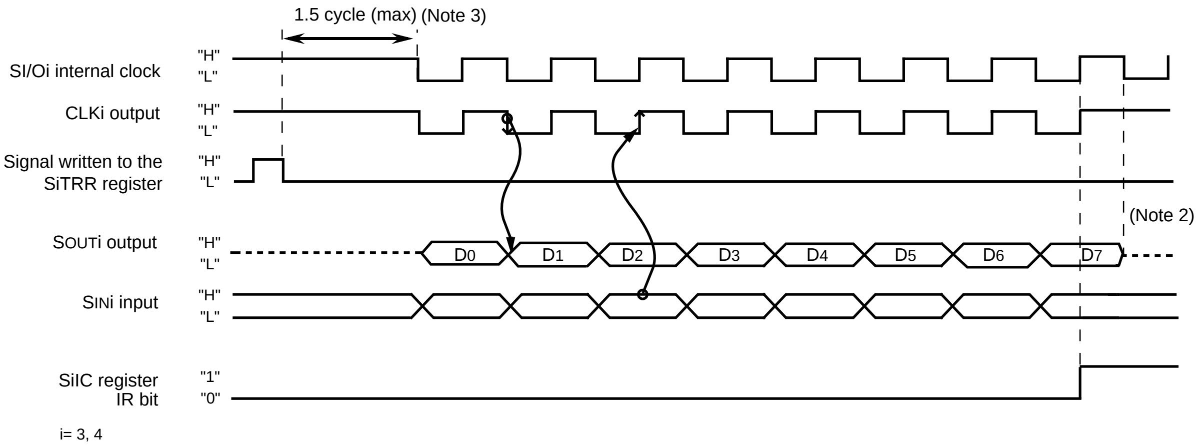
Note 2: Unlike UART0 to UART2, SI/Oi (i = 3 to 4) is not separated between the transfer register and buffer. Therefore, do not write the next transmit data to the SiTRR register during transmission.

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



Note 1: This diagram applies to the case where the SiC register bits are set as follows:
SMi2=0 (SOUTi output), SMi3=1 (SOUTi output, CLKi function), SMi4=0 (transmit data output at the falling edge and receive data input at the rising edge of the transfer clock), SMi5=0 (LSB first) and SMi6=1 (internal clock)

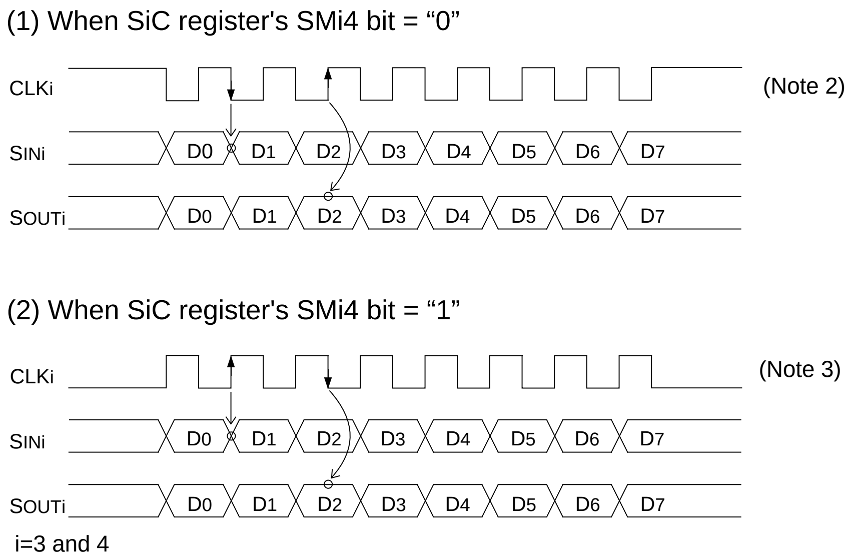
Note 2: When the SMi6 bit = 1 (internal clock), the SOUTi pin is placed in the high-impedance state after the transfer finishes.

Note 3: If the SMi6 bit=0 (internal clock), the serial I/O starts sending or receiving data a maximum of 1.5 transfer clock cycles after writing to the SiTRR register.

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.



Note 1: This diagram applies to the case where the SiC register bits are set as follows:
SMi5=0 (LSB first) and SMi6=1 (internal clock)

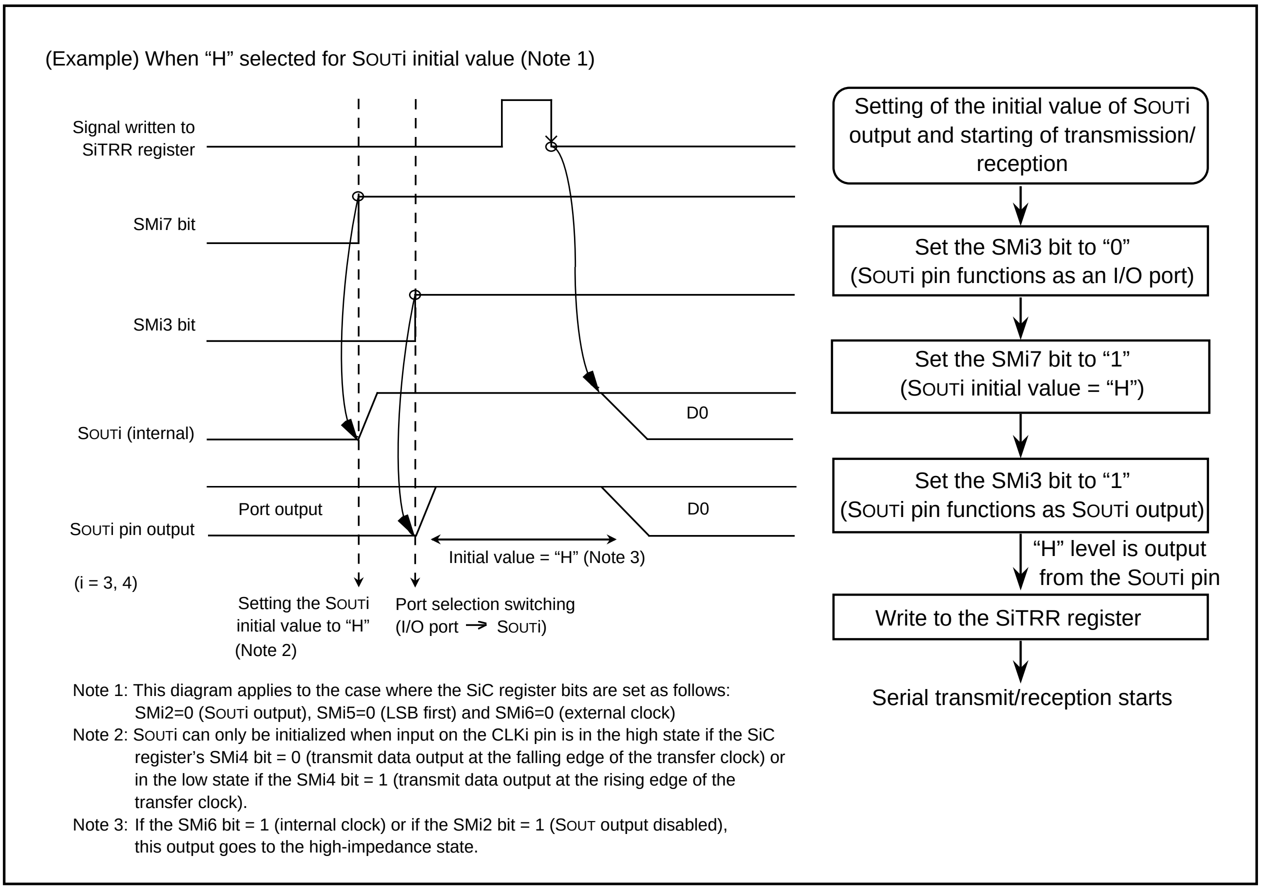
Note 2: When the SMi6 bit=1 (internal clock), a high level is output from the CLKi pin if not transferring data.

Note 3: When the SMi6 bit=1 (internal clock), a low level is output from the CLKi pin if not transferring data.

Figure 1.17.4. Polarity of Transfer Clock

(c) Functions for Setting an SOUTi Initial Value

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


(Example) When "H" selected for SOUTi initial value (Note 1)
Signal written to SiTRR register
SMi7 bit
SMi3 bit
SOUTi (internal)
D0
Port output
D0
SOUTi pin output
Initial value = "H" (Note 3)
(i = 3, 4)
Setting the SOUTi initial value to "H" (Note 2)
Port selection switching (I/O port → SOUTi)
Setting of the initial value of SOUTi output and starting of transmission/ reception
Set the SMi3 bit to "0" (SOUTi pin functions as an I/O port)
Set the SMi7 bit to "1" (SOUTi initial value = "H")
Set the SMi3 bit to "1" (SOUTi pin functions as SOUTi output)
"H" level is output from the SOUTi pin
Write to the SiTRR register
Serial transmit/reception starts

Note 1: This diagram applies to the case where the SiC register bits are set as follows: SMi2=0 (SOUTi output), SMi5=0 (LSB first) and SMi6=0 (external clock)

Note 2: SOUTi can only be initialized when input on the CLKi pin is in the high state if the SiC register's SMi4 bit = 0 (transmit data output at the falling edge of the transfer clock) or in the low state if the SMi4 bit = 1 (transmit data output at the rising edge of the transfer clock).

Note 3: If the SMi6 bit = 1 (internal clock) or if the SMi2 bit = 1 (SOUT output disabled), this output goes to the high-impedance state.

Figure 1.17.5. SOUTi's Initial Value Setting

Programmable I/O Ports

The programmable input/output ports (hereafter referred to simply as “I/O ports”) consist of 26 lines P1, P4 to P9 (except $P8_5$). 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. $P8_5$ is an input-only port and does not have a pull-up resistor. Port $P8_5$ shares the pin with $\overline{NMI}$, so that the $\overline{NMI}$ input level can be read from the P8 register P8_5 bit.

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 = 0 to 13)

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 $P8_5$ 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)

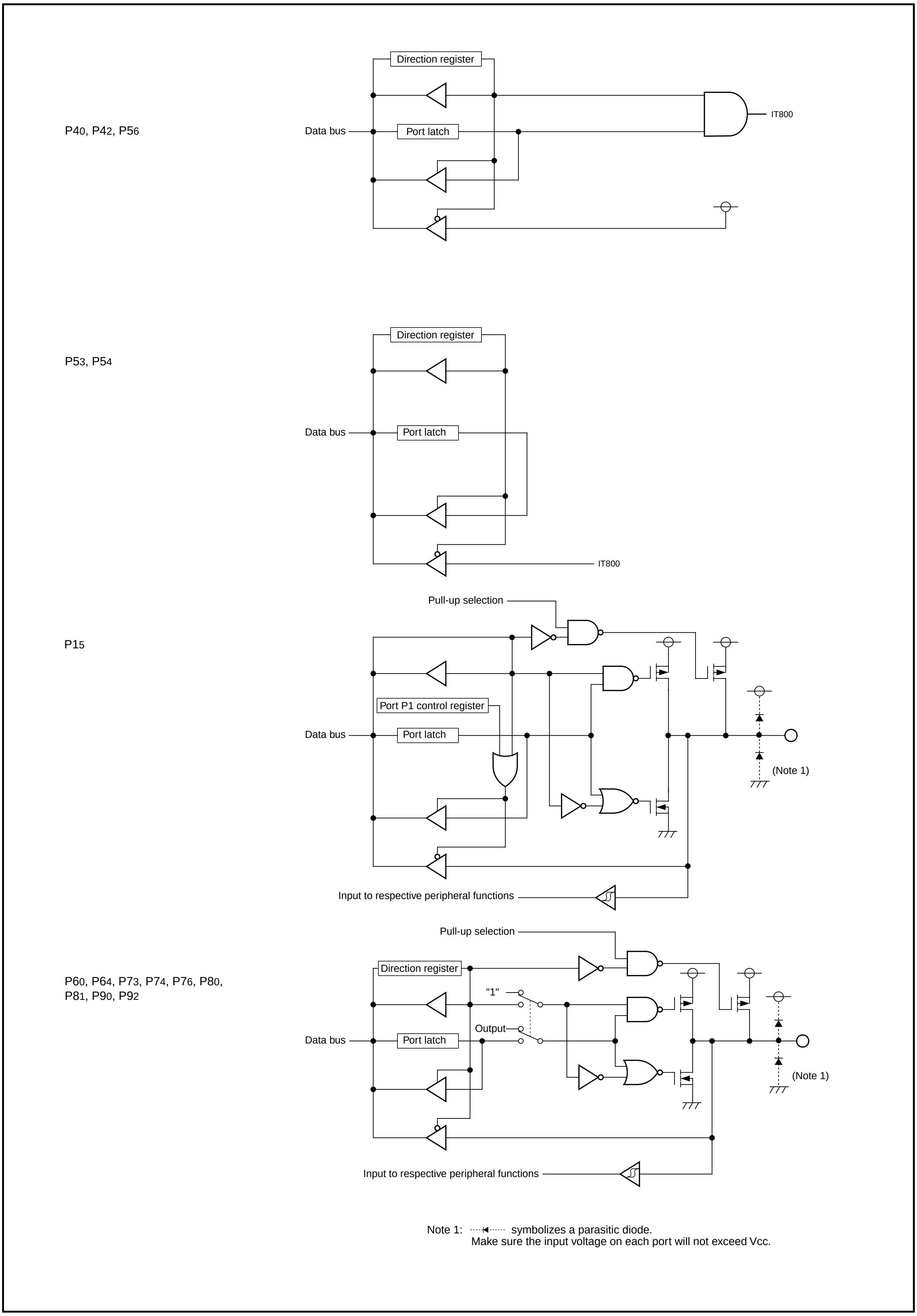
Figure 1.18.8 shows the PUR0 to PUR2 registers.

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

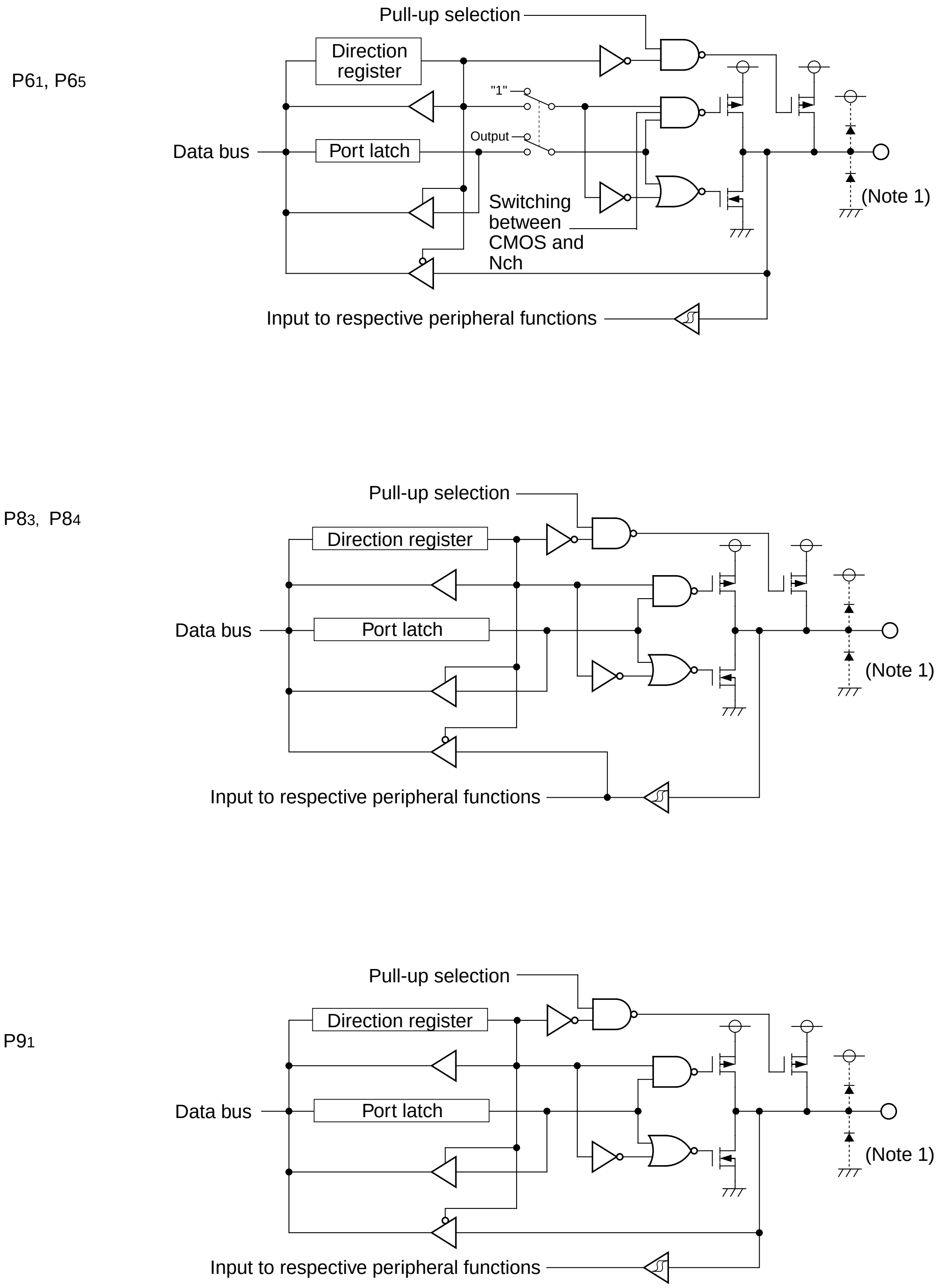
Figure 1.18.9 shows the port control register.

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.



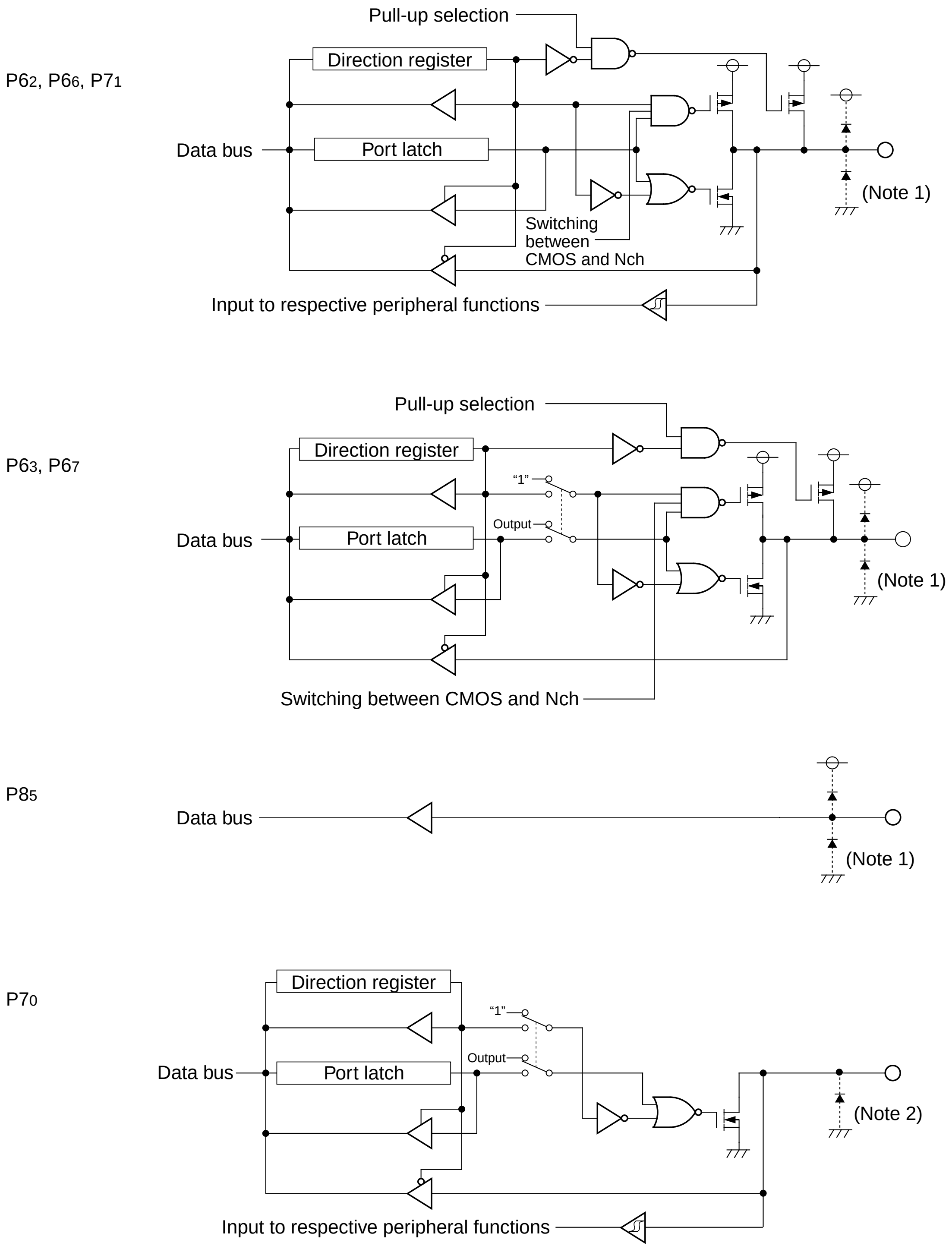
Note 1: ·····◄····· symbolizes a parasitic diode.
Make sure the input voltage on each port will not exceed Vcc.

Figure 1.18.1. I/O Ports (1)



Note 1: ····|◀···· symbolizes a parasitic diode.
Make sure the input voltage on each port will not exceed Vcc.

Figure 1.18.2. I/O Ports (2)



Note 1: symbolizes a parasitic diode.
Make sure the input voltage on each port will not exceed Vcc.
Note 2: symbolizes a parasitic diode.

Figure 1.18.3. I/O Ports (3)

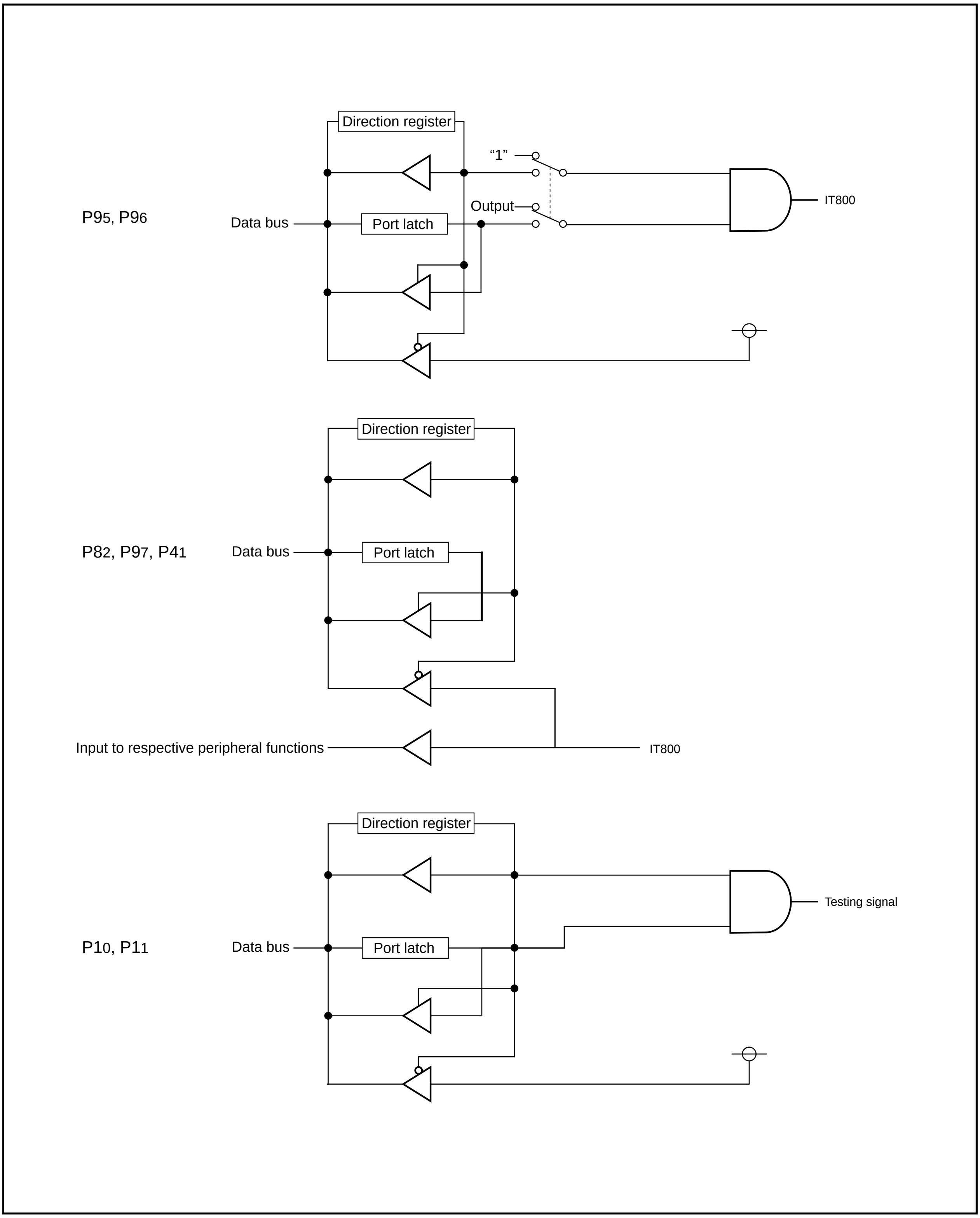
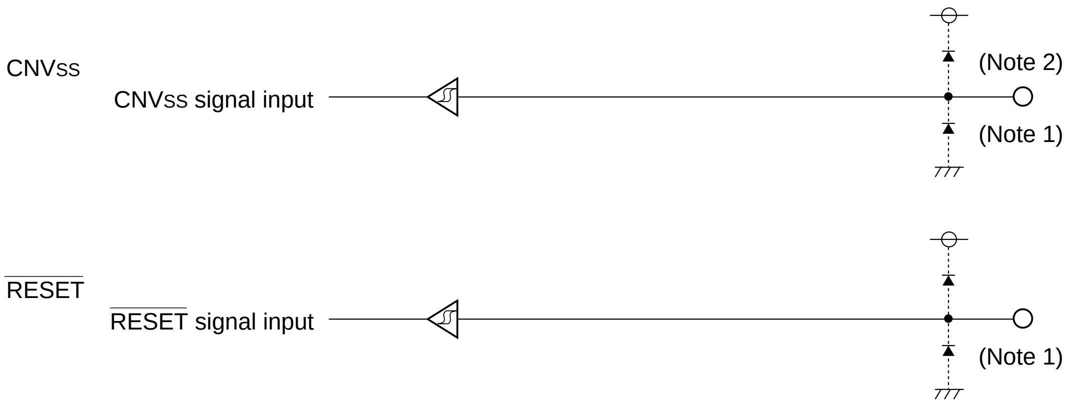


Figure 1.18.4. I/O Ports (4)

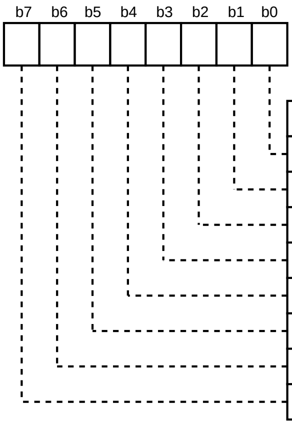


Note 1: ·····▸|······ symbolizes a parasitic diode.
Make sure the input voltage on each port will not exceed Vcc.
Note 2: A parasitic diode on the VCC side is added to the mask ROM version.
Make sure the input voltage on each port will not exceed Vcc.

Figure 1.18.5. I/O Pins

Port Pi direction register (i=1, 4 to 7 and 9) (Note 1)

Symbol	Address		After reset
PD1	03E3$_{16}$	(Note 2)	00$_{16}$
PD4 to PD7	03EA$_{16}$, 03EB$_{16}$, 03EE$_{16}$, 03EF$_{16}$		00$_{16}$
PD9	03F3$_{16}$		00$_{16}$



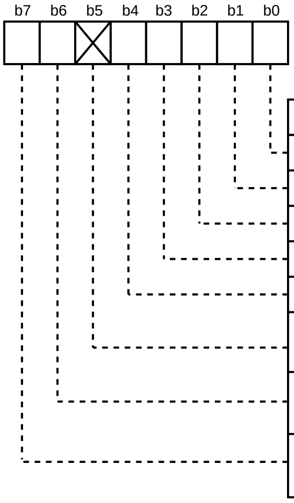
Bit symbol	Bit name	Function	RW
PDi_0	Port Pi_0 direction bit	0 : Input mode (Functions as an input port) 1 : Output mode (Functions as an output port) (i = 0 to 7 and 9 to 13)	RW
PDi_1	Port Pi_1 direction bit		RW
PDi_2	Port Pi_2 direction bit		RW
PDi_3	Port Pi_3 direction bit		RW
PDi_4	Port Pi_4 direction bit		RW
PDi_5	Port Pi_5 direction bit		RW
PDi_6	Port Pi_6 direction bit		RW
PDi_7	Port Pi_7 direction bit		RW

Note 1: Make sure the PD9 register is written to by the next instruction after setting the PRCR register's PRC2 bit to “1” (write enabled).

Note 2: Set PD1_0 and PD1_1 bits to “0.”

Port P8 direction register

Symbol	Address	After reset
PD8	03F2$_{16}$	00X00000$_2$

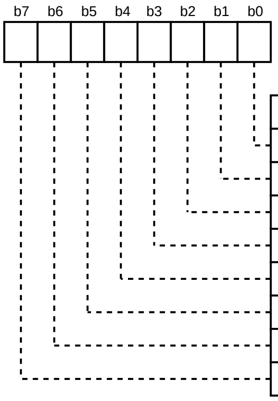


Bit symbol	Bit name	Function	RW
PD8_0	Port $P8_0$ direction bit	0 : Input mode (Functions as an input port) 1 : Output mode (Functions as an output port)	RW
PD8_1	Port $P8_1$ direction bit		RW
PD8_2	Port $P8_2$ direction bit		RW
PD8_3	Port $P8_3$ direction bit		RW
PD8_4	Port $P8_4$ direction bit		RW
—— (b5)	Nothing is assigned. In an attempt to write to this bit, write “0”. The value, if read, turns out to be indeterminate.		—
PD8_6	Port $P8_6$ direction bit	0 : Input mode (Functions as an input port) 1 : Output mode (Functions as an output port)	RW
PD8_7	Port $P8_7$ direction bit		RW

Figure 1.18.6. PD1, PD4 to PD9 Registers

Port Pi register (i=1, 4 to 7 and 9)

Symbol	Address		After reset
P1	03E1$_{16}$	(Note 2)	Indeterminate
P4 to P7	03E8$_{16}$, 03E9$_{16}$, 03EC$_{16}$, 03ED$_{16}$		Indeterminate
P9	03F1$_{16}$		Indeterminate

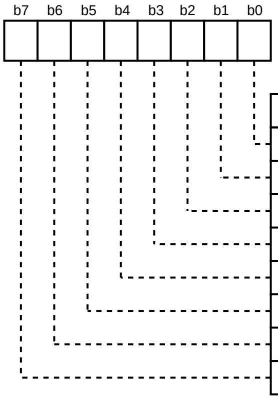


Bit symbol	Bit name	Function	RW
Pi_0	Port Pi$_0$ bit	The pin level on any I/O port which is set for input mode can be read by reading the corresponding bit in this register. The pin level on any I/O port which is set for output mode can be controlled by writing to the corresponding bit in this register 0 : “L” level 1 : “H” level (Note 1) (i = 0 to 7 and 9 to 13)	RW
Pi_1	Port Pi$_1$ bit		RW
Pi_2	Port Pi$_2$ bit		RW
Pi_3	Port Pi$_3$ bit		RW
Pi_4	Port Pi$_4$ bit		RW
Pi_5	Port Pi$_5$ bit		RW
Pi_6	Port Pi$_6$ bit		RW
Pi_7	Port Pi$_7$ bit		RW

Note 1: Since P7$_0$ is N-channel open drain ports, the data is high-impedance.
Note 2: Set P1_0 and P1_1 bits to “0.”

Port P8 register

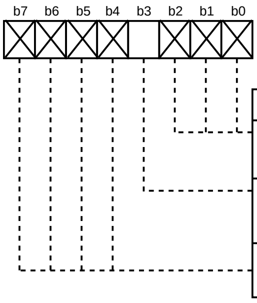
Symbol	Address	After reset
P8	03F0$_{16}$	Indeterminate



Bit symbol	Bit name	Function	RW
P8_0	Port P8$_0$ bit	The pin level on any I/O port which is set for input mode can be read by reading the corresponding bit in this register. The pin level on any I/O port which is set for output mode can be controlled by writing to the corresponding bit in this register (except for P8$_5$) 0 : “L” level 1 : “H” level	RW
P8_1	Port P8$_1$ bit		RW
P8_2	Port P8$_2$ bit		RW
P8_3	Port P8$_3$ bit		RW
P8_4	Port P8$_4$ bit		RW
P8_5	Port P8$_5$ bit		RO
P8_6	Port P8$_6$ bit		RW
P8_7	Port P8$_7$ bit		RW

Figure 1.18.7. P1, P4 to P9

Pull-up control register 0 (Note 1)

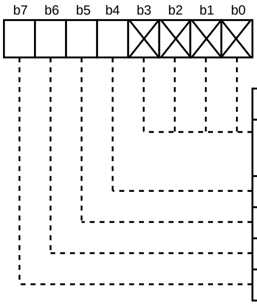


Symbol	Address	After reset
PUR0	$03FC_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
(b2-b0)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
PU03	$P1_4$ to $P1_7$ pull-up	0 : Not pulled high 1 : Pulled high (Note 2)	RW
(b7-b4)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—

Note 1: During memory extension and microprocessor modes, the pins are not pulled high although their corresponding register contents can be modified.
Note 2: The pin for which this bit is “1” (pulled high) and the direction bit is “0” (input mode) is pulled high.

Pull-up control register 1



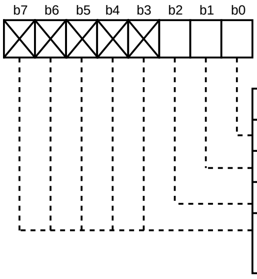
Symbol	Address	After reset(Note 5)
PUR1	$03FD_{16}$	00000000_2 00000010_2

Bit symbol	Bit name	Function	RW
(b3-b0)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		—
PU14	$P6_0$ to $P6_3$ pull-up	0 : Not pulled high 1 : Pulled high (Note 3)	RW
PU15	$P6_4$ to $P6_7$ pull-up		RW
PU16	$P7_2$ to $P7_3$ pull-up (Note 1)		RW
PU17	$P7_4$ to $P7_7$ pull-up		RW

Note 1: The $P7_0$ and $P7_1$ pins do not have pull-ups.
Note 2: During memory extension and microprocessor modes, the pins are not pulled high although the contents of these bits can be modified.
Note 3: The pin for which this bit is “1” (pulled high) and the direction bit is “0” (input mode) is pulled high.
Note 4: If the PM01 to PM00 bits are set to “01_2” (memory expansion mode) or “11_2” (microprocessor mode) in a program during single-chip mode, the PU11 bit becomes “1”.
Note 5: The values after hardware reset 1 and 2 are as follows:
- 00000000_2 when input on CNVss pin is “L“
- 00000010_2 when input on CNVss pin is “H“

The values after software reset, watchdog timer reset and oscillation stop detection reset are as follows:
- 00000000_2 when PM 01 to PM00 bits of PM0 register are “00_2“ (single-chip mode)
- 00000010_2 when PM 01 to PM00 bits of PM0 register are “01_2“ (memory expansion mode) or “11_2“ (microprocessor mode)

Pull-up control register 2



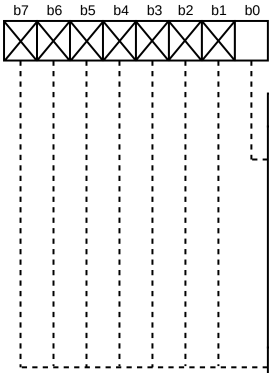
Symbol	Address	After reset
PUR2	$03FE_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
PU20	$P8_0$ to $P8_3$ pull-up	0 : Not pulled high 1 : Pulled high (Note 1)	RW
PU21	$P8_4$ to $P8_7$ pull-up (Note 2)		RW
PU22	$P9_0$ to $P9_3$ pull-up		RW
(b7-b3)	Nothing is assigned. In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”.		—

Note 1: The pin for which this bit is “1” (pulled high) and the direction bit is “0” (input mode) is pulled high.
Note 2: The $P8_5$ pin does not have pull-up.

Figure 1.18.8. PUR0 to PUR2 Registers

Port control register



Symbpl	Address	After reset
PCR	03FF$_{16}$	00$_{16}$

Bit symbol	Bit name	Function	RW
PCR0	Port P1 control bit	Operation performed when the P1 register is read 0: When the port is set for input, the input levels of P10 to P17 pins are read. When set for output, the port latch is read. 1: The port latch is read regardless of whether the port is set for input or output.	RW
—— (b7-b1)	Nothing is assigned. In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”.		—

Figure 1.18.9. PCR Register

Table 1.18.1. Unassigned Pin Handling in Single-chip Mode

Pin name	Connection
Ports P1, P6 to P9 (excluding $P8_5$)	After setting for input mode, connect every pin to V_{SS} via a resistor(pull-down); or after setting for output mode, leave these pins open.
X_{OUT} (Note 1)	Open
$P8_5$	Connect via resistor to V_{CC} (pull-up)
V_{CC}A	Connect to V_{CC}
V_{SS}A	Connect to V_{SS}

Note 1: With external clock input to X_{IN} pin.

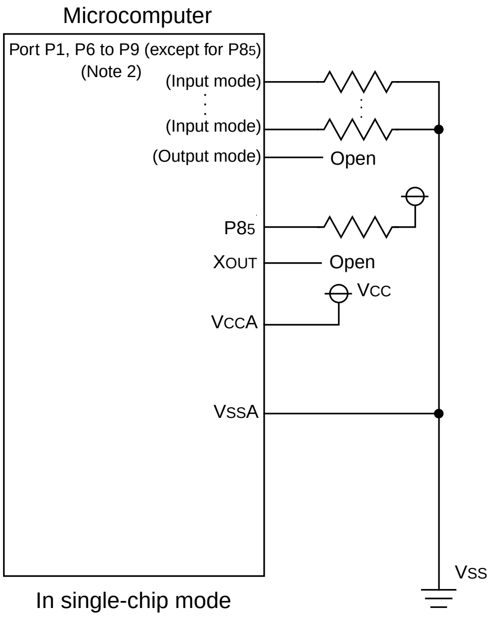


Figure 1.18.10. Unassigned Pins Handling

Electrical Characteristics

Table 1.19.1. Absolute Maximum Ratings

Symbol	Parameter		Condition	Rated value	Unit
V_{CC}	Supply voltage		V_{CC}=V_{CC}A	-0.3 to 4.2	V
V_{CC}A	Analog supply voltage		V_{CC}=V_{CC}A	-0.3 to 4.2	V
V_I	Input voltage	$\overline{RESET}$, CNV_{SS} $P6_0$ to $P6_7$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_5$, $P9_0$ to $P9_2$, X_{IN}		-0.3 to V_{CC}+0.3	V
		$P7_0$		-0.3 to 6.5	V
V_O	Output voltage	$P6_0$ to $P6_7$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_4$, $P9_0$ to $P9_2$, X_{OUT}, TS		-0.3 to V_{CC1}+0.3	V
		$P7_0$		-0.3 to 6.5	V
P_d	Power dissipation			500	mW
T_{opr}	Operating ambient temperature			-20 to 85	°C
T_{stg}	Storage temperature			-65 to 150	°C

Table 1.19.2. Recommended Operating Conditions (Note 1)

Symbol	Parameter			Standard			Unit
				Min.	Typ.	Max.	
V_{CC}	Supply voltage			3.0	3.3	3.6	V
$V_{CC}A$	Analog supply voltage				V_{CC}		V
V_{SS}	Supply voltage				0		V
$V_{SS}A$	Analog supply voltage				0		V
V_{IH}	HIGH input voltage	$P6_0$ to $P6_7$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_4$, $P9_0$ to $P9_2$ X_{IN}, $\overline{\text{RESET}}$, CNV_{SS}		0.8V_{CC1}		V_{CC1}	V
		$P7_0$		0.8V_{CC1}		6.5	V
V_{IL}	LOW input voltage	$P6_0$ to $P6_7$, $P7_0$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_4$, $P9_0$ to $P9_2$ X_{IN}, $\overline{\text{RESET}}$, CNV_{SS}		0		0.2V_{CC1}	V
$I_{OH\ (peak)}$	HIGH peak output current	$P6_0$ to $P6_7$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_4$, $P9_0$ to $P9_2$, TS				-10.0	mA
$I_{OH\ (avg)}$	HIGH average output current	$P6_0$ to $P6_7$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_4$, $P9_0$ to $P9_2$, TS				-5.0	mA
$I_{OL\ (peak)}$	LOW peak output current	$P6_0$ to $P6_7$, $P7_0$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_4$, $P9_0$ to $P9_2$, TS				10.0	mA
$I_{OL\ (avg)}$	LOW average output current	$P6_0$ to $P6_7$, $P7_0$, $P7_1$, $P7_3$, $P7_4$, $P7_6$, $P8_0$ to $P8_4$, $P9_0$ to $P9_2$, TS				5.0	mA
f (X_{IN})	Main clock input oscillation frequency (Note 4)		V_{CC}=3.0 to 3.6V		5.12		MHz
f (Ring)	Ring oscillation frequency				1		MHz
f (BCLK)	CPU operation clock				15.36		MHz
T_{SU}(PLL)	PLL frequency synthesizer stabilization wait time		V_{CC}=3.0V			50	ms

Note 1: Referenced to V_{CC} = 3.0 to 3.6V at Topr = -20 to 85 °C unless otherwise specified.
Note 2: The mean output current is the mean value within 100ms.
Note 3: The total I_{OL} (peak) for all ports must be 80mA max, the total I_{OH} (peak) for all ports must be -40mA max.

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

Parameter	Standard			Unit
	Min.	Typ.	Max	
Word program time		30	200	µs
Block erase time		1	4	s
Erase all unlocked blocks time		1 X n	4 X n	s
Lock bit program time		30	200	µs

Note 1: Referenced to VCC=3.0 to 3.6V at Topr = 0 to 60 °C unless otherwise specified.
Note 2: n denotes the number of block erases.

Table 1.19.4. Flash Memory Version Program/Erase Voltage and Read Operation Voltage Characteristics (at Topr = 0 to 60°C)

Flash program, erase voltage	Flash read operation voltage
VCC = 3.3 V ± 0.3 V	VCC = 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	VCC =3.0 to 3.6V			2	ms
td(R-S)	STOP release time				150	µs
td(M-L)	Time for internal power supply stabilization when main clock oscillation starts				50	µs

Note : When Vcc1 = 5V

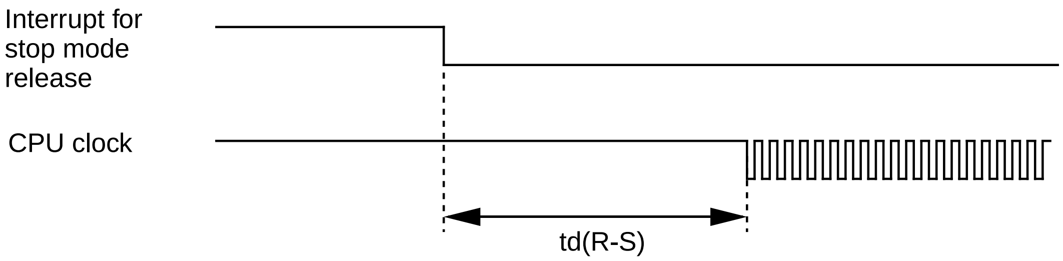


Table 1.19.6. Electrical Characteristics (Note)

Symbol	Parameter		Measuring condition	Standard			Unit
				Min.	Typ.	Max.	
V_{OH}	HIGH output voltage	P6$_0$ to P6$_7$, P7$_1$, P7$_3$, P7$_4$, P7$_6$, P8$_0$ to P8$_4$, P9$_0$ to P9$_2$, TS	I_{OH}= -1mA	$V_{CC-0.5}$		V_{CC}	V
V_{OH}	HIGH output voltage	X$_{OUT}$	I_{OH}= -0.1mA	$V_{CC-0.5}$		V_{CC}	V
V_{OL}	LOW output voltage	P6$_0$ to P6$_7$, P7$_1$, P7$_3$, P7$_4$, P7$_6$, P8$_0$ to P8$_4$, P9$_0$ to P9$_2$, TS	I_{OL}=1mA			0.5	V
V_{OL}	LOW output voltage	X$_{OUT}$	I_{OL}=0.1mA			0.5	V
V_{T+}-V_{T-}	Hysteresis	$\overline{\mathrm{TA0_{IN}}}$, $\overline{\mathrm{TA1_{IN}}}$, TA4$_{IN}$, INT$_1$ to INT$_3$, $\overline{\mathrm{CTS_0}}$ to $\overline{\mathrm{CTS_2}}$, $\overline{\mathrm{SCL}}$, SDA, CLK$_0$ to CLK$_4$, TA4$_{OUT}$, RxD$_0$ to RxD$_2$, S$_{IN3}$		0.2		0.8	V
V_{T+}-V_{T-}	Hysteresis	$\overline{\mathrm{RESET}}$		0.2	(0.7)	1.8	V
I_{IH}	HIGH input current	P6$_0$ to P6$_7$, P7$_0$, P7$_1$, P7$_3$, P7$_4$, P7$_6$, P8$_0$ to P8$_4$, P9$_0$ to P9$_2$ X$_{IN}$, $\overline{\mathrm{RESET}}$, CNVss	V_I=3V			4.0	μA
I_{IL}	LOW input current	P6$_0$ to P6$_7$, P7$_0$, P7$_1$, P7$_3$, P7$_4$, P7$_6$, P8$_0$ to P8$_4$, P9$_0$ to P9$_2$ X$_{IN}$, $\overline{\mathrm{RESET}}$, CNVss	V_I=0V			-4.0	μA
R_{PULLUP}	Pull-up resistance	P6$_0$ to P6$_7$, P7$_1$, P7$_3$, P7$_4$, P7$_6$, P8$_0$ to P8$_4$, P9$_0$ to P9$_2$	V_I=0V	66	160	500	kΩ
R_{fXIN}	Feedback resistance	X$_{IN}$			3.0		MΩ

Note : Referenced to V_{CC}=3.0 to 3.6V, V_{SS}=0V at Topr = -20 to 85 °C, f(BCLK)=15.36 MHz unless otherwise specified.

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

Symbol	Parameter		Me suring condition		Standard			Unit
					Min.	Typ.	Max.	
I_{CC}	Power supply current (V_{CC}=2.7 to 3.6V)	In single-chip mode, the output pins are open and other pins are V_{SS}	Flash memory	f(BCLK)=15.36 MHz, No division		70		mA
			Flash memory Program	f(BCLK)=10MHz, V_{CC1}=3.0V		TBD		mA
			Flash memory Erase	f(BCLK)=10MHz, V_{CC1}=3.0V		TBD		mA

Note 1: Referenced to V_{CC}= 3.0 to 3.6V, V_{SS}=0V at Topr = -20 to 85 °C, f(BCLK)=15.36 MHz unless otherwise specified.

Timing Requirements
(VCC = 3V, VSS = 0V, at Topr = – 20 to 85^{O}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
(VCC = 3V, VSS = 0V, at Topr = – 20 to 85°C unless otherwise specified)

Table 1.19.9. Timer A Input (Counter Input in Event Counter Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAiIN input cycle time	150		ns
$t_{w(TAH)}$	TAiIN input HIGH pulse width	60		ns
$t_{w(TAL)}$	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.	
$t_{c(TA)}$	TAiIN input cycle time	600		ns
$t_{w(TAH)}$	TAiIN input HIGH pulse width	300		ns
$t_{w(TAL)}$	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.	
$t_{c(TA)}$	TAiIN input cycle time	300		ns
$t_{w(TAH)}$	TAiIN input HIGH pulse width	150		ns
$t_{w(TAL)}$	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.	
$t_{w(TAH)}$	TAiIN input HIGH pulse width	150		ns
$t_{w(TAL)}$	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.	
$t_{c(UP)}$	TAiOUT input cycle time	3000		ns
$t_{w(UPH)}$	TAiOUT input HIGH pulse width	1500		ns
$t_{w(UPL)}$	TAiOUT input LOW pulse width	1500		ns
$t_{su(UP-TIN)}$	TAiOUT input setup time	600		ns
$t_{h(TIN-UP)}$	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.	
$t_{c(TA)}$	TAiIN input cycle time	2		µs
$t_{su(TAIN-TAOUT)}$	TAiOUT input setup time	500		ns
$t_{su(TAOUT-TAIN)}$	TAiIN input setup time	500		ns

Timing Requirements

(V_{CC} = 3V, V_{SS} = 0V, at Topr = – 20 to 85^{o}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 $\overline{\text{INTi}}$ Input

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

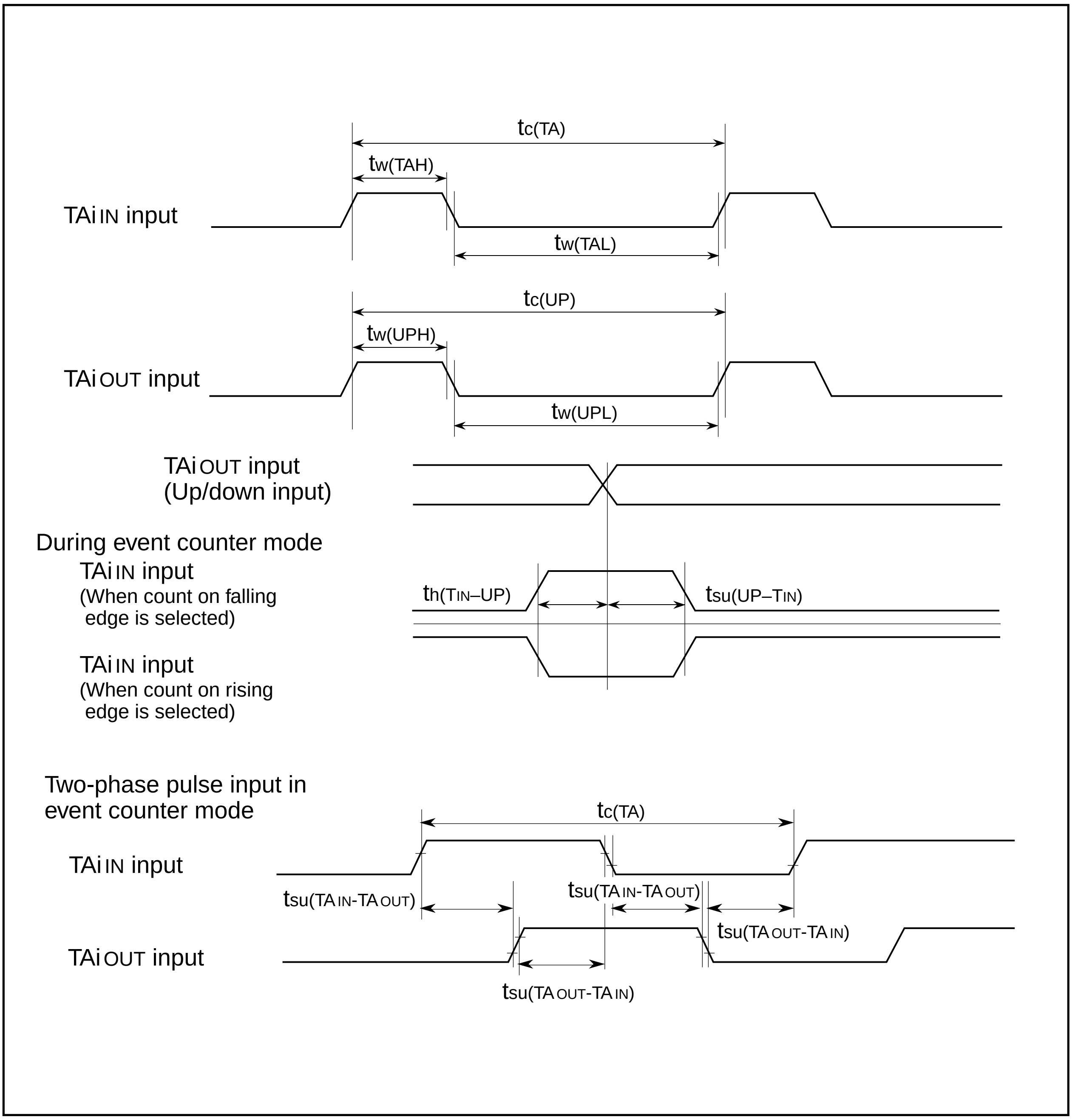


Figure 1.19.1. Timing Diagram (1)

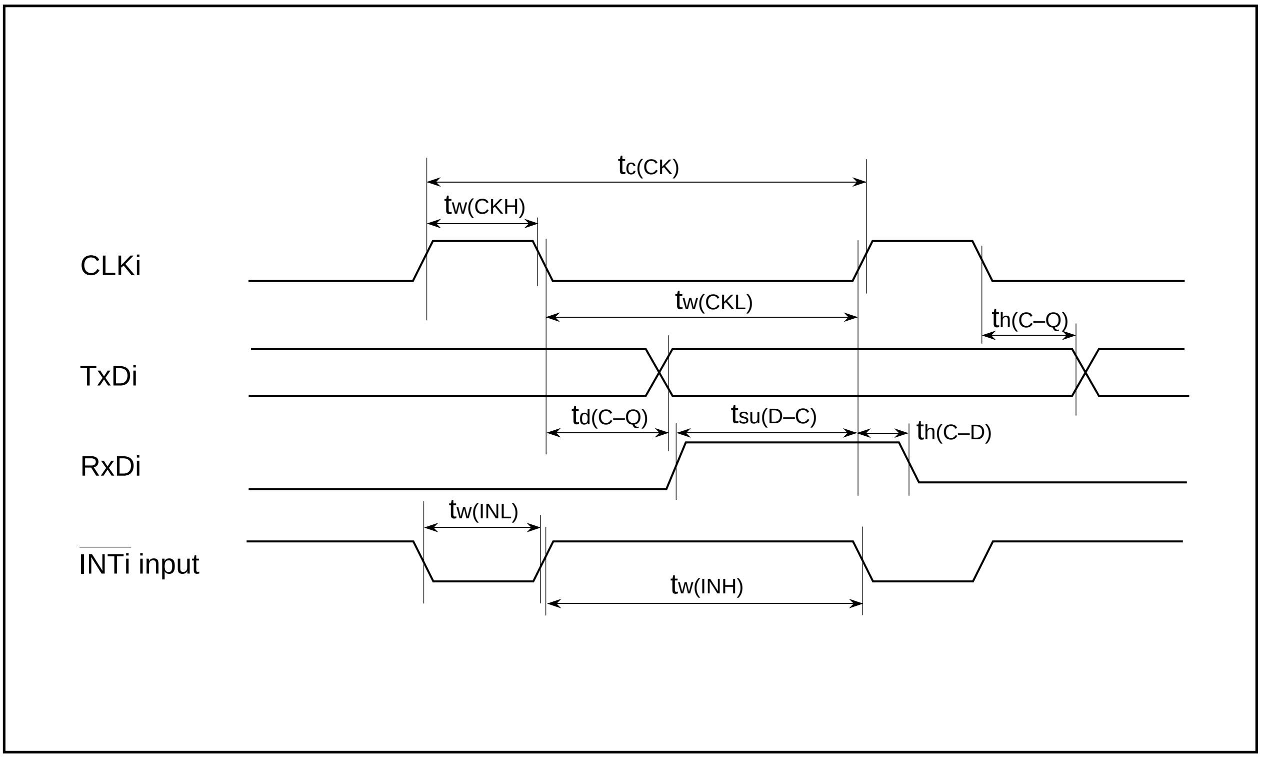


Figure 1.19.2. Timing Diagram (2)

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 writter 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		3 modes (CPU rewrite, standard serial I/O, parallel I/O)
Erase block	User ROM area	See Figure 1.20.1
	Boot ROM area	1 block (4 Kbytes) (Note 1)
Method for program		In units of word
Method for erasure		Block erase
Program, erase control method		Program and erase controlled by software command
Protect method		Two 8 Kbyte user blocks by register lock bit (FMR02)
Number of commands		5 commands
Number of program and erasure		100 times
Data Retention		10 years
ROM code protection		Parallel I/O and standard serial I/O modes are supported.

Note 1: The boot ROM area contains a standard serial I/O mode rewrite control program which is stored in it when shipped from the factory. This area can only be rewritten in parallel input/output mode.

Table 1.20.2. Flash Memory Rewrite Modes Overview

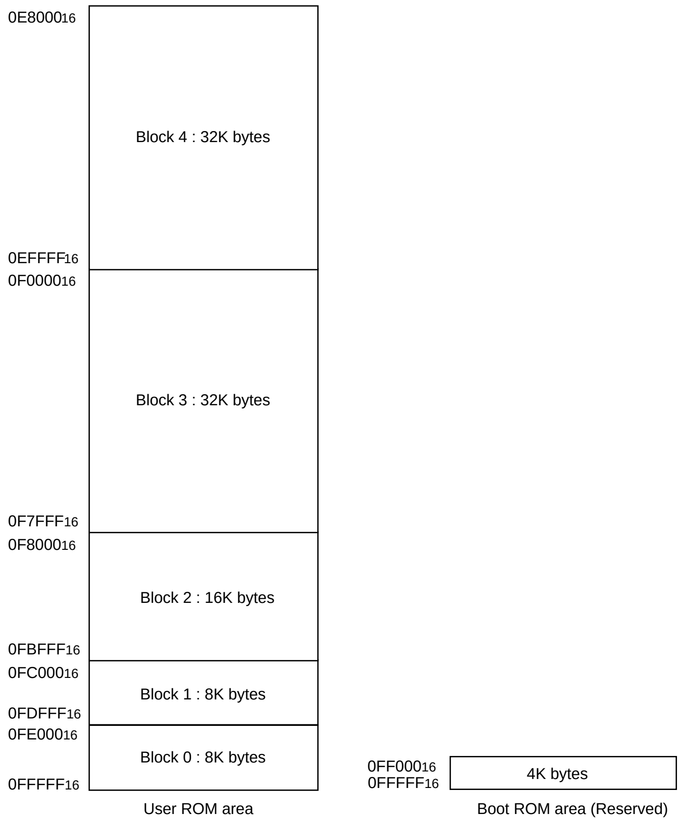
Flash memory rewrite mode	CPU rewrite mode	Standard serial I/O mode
Function	The user ROM area is rewritten by executing software commands from the CPU. EW0 mode: Can be rewritten in any area other than the flash memory EW1 mode: Can be rewritten in the flash memory	The user ROM area is rewritten by 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. Figure 1.20.1 shows the block diagram of flash momoery.

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 located at addresses that overlap the user ROM area, and can only be rewritten in parallel input/output mode.



Note 1: To specify a block, use an even address in that block.
Note 2: Blocks 0 and 1 can be rewritten if FMR02 of FMR0 register is set to "1" (only in case of CPU rewriting mode.)

Figure 1.20.1. Flash Memory Block Diagram

Boot Mode

After a hardware reset which is performed by applying a high-level signal to the CNVSS and P1$_5$ 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

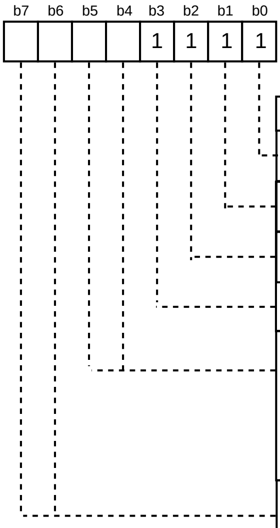
The ROM code protect function inhibits the flash memory from being read or rewritten during parallel input/output mode. Figure 1.20.2 shows the ROMCP register.

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 ‘00$_2$,’ with the flash memory thereby protected against reading or rewriting. Conversely, when the ROMCR bits are ‘00$_2$’ (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$_{16}$, 0FFFE3$_{16}$, 0FFFEB$_{16}$, 0FFFEF$_{16}$, 0FFFF3$_{16}$, 0FFFF7$_{16}$, and 0FFFFB$_{16}$. Prepare a program in which the ID codes are preset at these addresses and write it in the flash memory.

ROM code protect control address



Symbol ROMCP Address 0FFFFF$_{16}$ Value when shipped FF$_{16}$ (Note 4)

Bit symbol	Bit name	Function	RW
———	Reserved bit	Set this bit to “1”	RW
———	Reserved bit	Set this bit to “1”	RW
———	Reserved bit	Set this bit to “1”	RW
———	Reserved bit	Set this bit to “1”	RW
ROMCR	ROM code protect reset bit (Note 2, Note 4)	b5 b4 00: Removes protect 01: 10: } Enables ROOMCP1 bit 11:	RW
			RW
ROMCP1	ROM code protect level 1 set bit (Note 1, Note 3, Note 4)	b7 b6 00: 01: } Protect enabled 10: 11: Protect disabled	RW
			RW

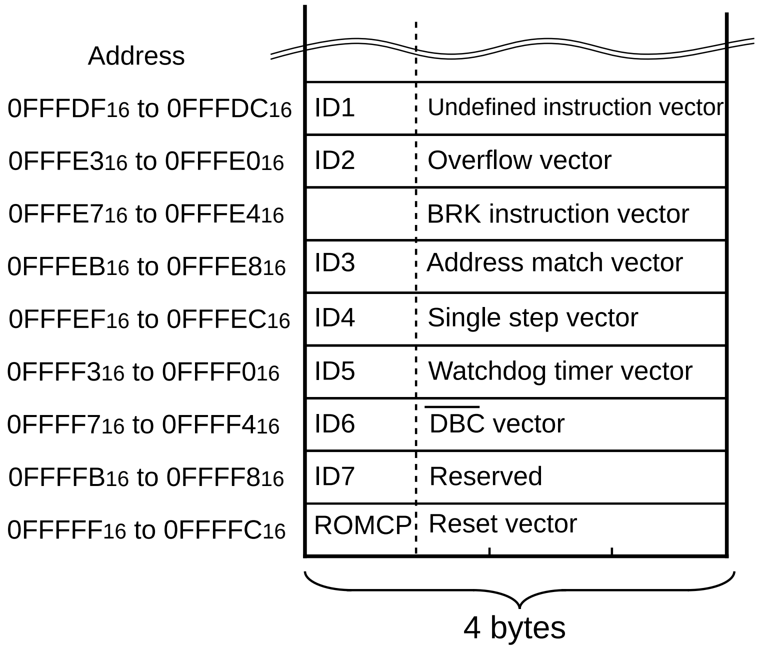
Note 1: If the ROMCR bits are set to other than ‘00$_2$’ and the ROMCP1 bits are set to other than ‘11$_2$’ (ROM code protect enabled), the flash memory is disabled against reading and rewriting in parallel input/output mode.

Note 2: If the ROMCR bits are set to ‘00$_2$’ when the ROMCR bits are other than ‘00$_2$’ and the ROMCP1 bits are other than ‘11$_2$,’ ROM code protect level 1 is removed. However, because the ROMCR bits cannot be modified during parallel input/output mode, they need to be modified in standard serial input/output or other modes.

Note 3: The ROMCP1 bits are effective when the ROMCR bits are ‘01$_2$,’ ‘10$_2$,’ or ‘11$_2$.’

Note 4: Once any of these bits is cleared to “0”, it cannot be set back to “1”. If a memory block that contains the ROMCP register is erased, the ROMCP register is set to ‘FF$_{16}$.’

Figure 1.20.2. ROMCP Register



Address		
0FFFDF$_{16}$ to 0FFFDC$_{16}$	ID1	Undefined instruction vector
0FFFE3$_{16}$ to 0FFFE0$_{16}$	ID2	Overflow vector
0FFFE7$_{16}$ to 0FFFE4$_{16}$		BRK instruction vector
0FFFEB$_{16}$ to 0FFFE8$_{16}$	ID3	Address match vector
0FFFEF$_{16}$ to 0FFFEC$_{16}$	ID4	Single step vector
0FFFF3$_{16}$ to 0FFFF0$_{16}$	ID5	Watchdog timer vector
0FFFF7$_{16}$ to 0FFFF4$_{16}$	ID6	$\overline{\text{DBC}}$ vector
0FFFFB$_{16}$ to 0FFFF8$_{16}$	ID7	Reserved
0FFFFF$_{16}$ to 0FFFFC$_{16}$	ROMCP	Reset vector

Figure 1.20.3. 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.

In CPU rewrite mode, only the user ROM area shown in Figure 1.20.1 can be rewritten and the boot ROM area cannot be rewritten. 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)(Note)
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 $\overline{\text{NMI}}$ 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.

Figure 1.21.2 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))
- When entering low power mode or ring low power mode

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 1.21.3 and 1.21.4 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.

Flash memory control register 0

b7	b6	b5	b4	b3	b2	b1	b0
		0	0				

Symbol	Address	After reset
FMR0	$01B7_{16}$	$XX000001_2$

Bit symbol	Bit name	Function	RW
FMR00	RY/$\overline{\text{BY}}$ status flag	0: Busy (being written or erased) 1: Ready	RO
FMR01	CPU rewrite mode select bit (Note 1)	0: Disables CPU rewrite mode 1: Inables CPU rewrite mode	RW
FMR02	Lock bit disable select bit (Note 2)	0: Inables lock bit 1: Disables lock bit	RW
FMSTP	Flash memory stop bit (Note 3, Note 5))	0: Enables flash memory operation 1: Stops flash memory operation (placed in low power mode, flash memory initialized)	RW
(b5-b4)	Reserved bit	Must always be set to “0”	RW
FMR06	Program status flag (Note 4)	0: Terminated normally 1: Terminated in error	RO
FMR07	Erase status flag (Note 4)	0: Terminated normally 1: Terminated in error	RO

Note 1: To set this bit to “1”, write “0” and then “1” in succession. Make sure no interrupts or DMA transfers will occur before writing “1” after writing “0”. Also, while in EW0 mode, write to this bit from a program in other than the flash memory.

Note 2: To set this bit to “1”, write “0” and then “1” in succession when the FMR01 bit = 1. Make sure no interrupts or no DMA transfers will occur before writing “1” after writing “0”.

Note 3: Write to this bit from a program in other than the flash memory.

Note 4: This flag is cleared to “0” by executing the Clear Status command.

Note 5: Effective when the FMR01 bit = 1 (CPU rewrite mode). If the FMR01 bit = 0, although the FMR03 bit can be set to “1” by writing “1” in a program, the flash memory is neither placed in low power mode nor initialized.

Flash memory control register 1

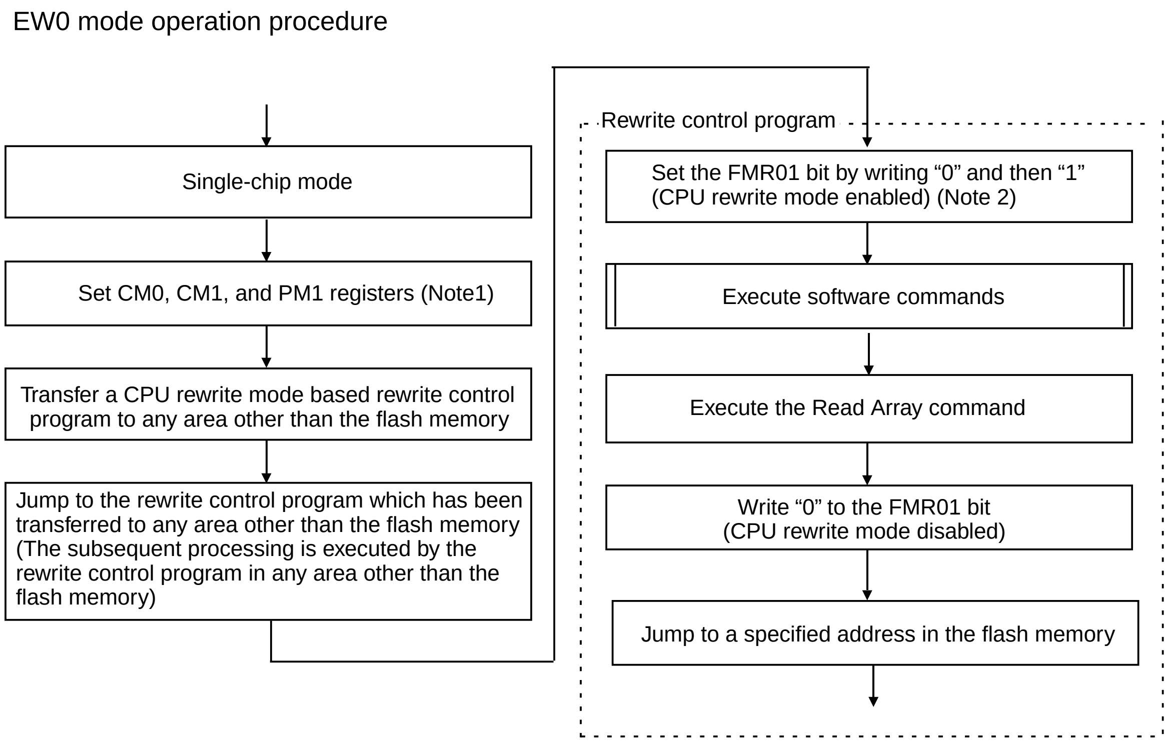
b7	b6	b5	b4	b3	b2	b1	b0
0	☒	0	0				0

Symbol	Address	After reset
FMR1	$01B5_{16}$	$0X00XX0X_2$

Bit symbol	Bit name	Function	RW
(b0)	Reserved bit	The value in this bit when read is indeterminate.	RO
FMR11	EW1 mode select bit (Note)	0: EW0 mode 1: EW1 mode	RW
(b3-b2)	Reserved bit	The value in this bit when read is indeterminate.	RO
(b5-b4)	Reserved bit	Must always be set to “0”	RW
(b6)	Nothing is assigned. When write, set to “0”. When read, their contents are indeterminate.		——
(b7)	Reserved bit	Must always be set to “0”	RW

Note : To set this bit to “1”, write “0” and then “1” in succession when the FMR01 bit = 1. Make sure no interrupts or no DMA transfers will occur before writing “1” after writing “0”.
The FMR01 and FMR11 bits both are cleared to “0” by setting the FMR01 bit to “0”.

Figure 1.21.2. FIDR Register and FMR0 and FMR1 Registers


EW0 mode operation procedure
Single-chip mode
Set CM0, CM1, and PM1 registers (Note1)
Transfer a CPU rewrite mode based rewrite control program to any area other than the flash memory
Jump to the rewrite control program which has been transferred to any area other than the flash memory (The subsequent processing is executed by the rewrite control program in any area other than the flash memory)
Rewrite control program
Set the FMR01 bit by writing “0” and then “1” (CPU rewrite mode enabled) (Note 2)
Execute software commands
Execute the Read Array command
Write “0” to the FMR01 bit (CPU rewrite mode disabled)
Jump to a specified address in the flash memory

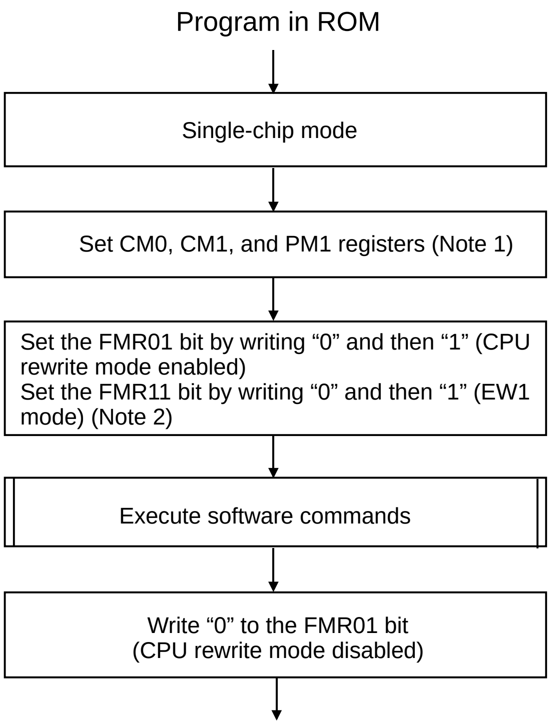
Note 1: Select 10 MHz or less for CPU clock using the CM0 register’s CM06 bit and CM1 register’s CM17 to 6 bits. Also, set the PM1 register’s PM17 bit to “1” (with wait state).

Note 2: To set the FMR01 bit to “1”, write “0” and then “1” in succession. Make sure no interrupts or no DMA transfers will occur before writing “1” after writing “0”.
Write to the FMR01 bit from a program in other than the flash memory.

Note 3: Disables the CPU rewrite mode after executing the Read Array command.

Figure 1.21.3. Setting and Resetting of EW0 Mode

EW1 mode operation procedure


Program in ROM
Single-chip mode
Set CM0, CM1, and PM1 registers (Note 1)
Set the FMR01 bit by writing “0” and then “1” (CPU rewrite mode enabled)
Set the FMR11 bit by writing “0” and then “1” (EW1 mode) (Note 2)
Execute software commands
Write “0” to the FMR01 bit
(CPU rewrite mode disabled)

Note 1: Select 10 MHz or less for CPU clock using the CM0 register’s CM06 bit and CM1 register’s CM17 to 6 bits. Also, set the PM1 register’s PM17 bit to “1” (with wait state).

Note 2: To set the FMR01 bit to “1”, write “0” and then “1” in succession. Make sure no interrupts or no DMA transfers will occur before writing “1” after writing “0”.
Write to the FMR01 bit from a program in other than the flash memory. Also write only when the $\overline{\text{NMI}}$ pin is “H” level.

Figure 1.21.4. 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$_{1t}$–D$_8$) are ignored.

Table 1.21.2. Software Commands

Command	First bus cycle			Second bus cycle		
	Mode	Address	Data (D$_0$ to D$_7$)	Mode	Address	Data (D$_0$ to D$_7$)
Read array	Write	X	xxFF$_{16}$			
Read status register	Write	X	xx70$_{16}$	Read	X	SRD
Clear status register	Write	X	xx50$_{16}$			
Program	Write	WA	xx40$_{16}$	Write	WA	WD
Block erase	Write	X	xx20$_{16}$	Write	BA	xxD0$_{16}$

SRD: Status register data (D$_7$ to D$_0$)
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$_{16}$)

This command reads the flash memory.

Writing ‘xxFF$_{16}$’ 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$_{16}$, FF$_{16}$, FF$_{16}$, and FF$_{16}$ 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$_{16}$ are in agreement.
(4) The head address +2 of (1) is specified.
(5) (4) is repeated until the read value and FFFF$_{16}$ are in agreement.
(6) Arbitrary addresses are specified.

Read Status Register Command (70$_{16}$)

This command reads the status register.

Write ‘xx70$_{16}$’ 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_{16})

This command clears the status register to “0”.

Write ‘xx50_{16}’ 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_{16})

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

Write ‘xx40_{16}’ 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.

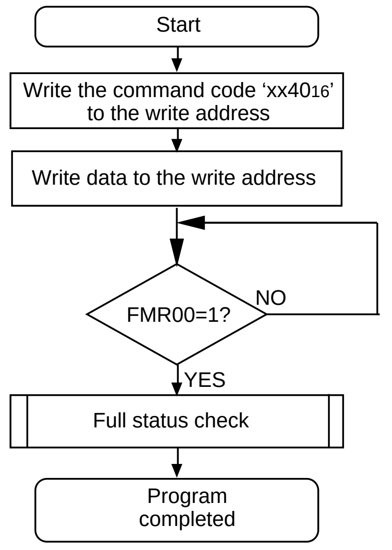
Check the FMR06 bit in the FMR0 register after auto programming has finished, and the result of auto programming can be known. (Refer to “Full Status Check.”)

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.


Start
Write the command code ‘xx4016’ to the write address
Write data to the write address
FMR00=1?
NO
YES
Full status check
Program completed

Note: Write the command code and data at even number.

Figure 1.21.5. Program Command

Block Erase

Write ‘xx20_{16}’ in the first bus cycle and write ‘xx$D0_{16}$’ 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 erasiing 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.

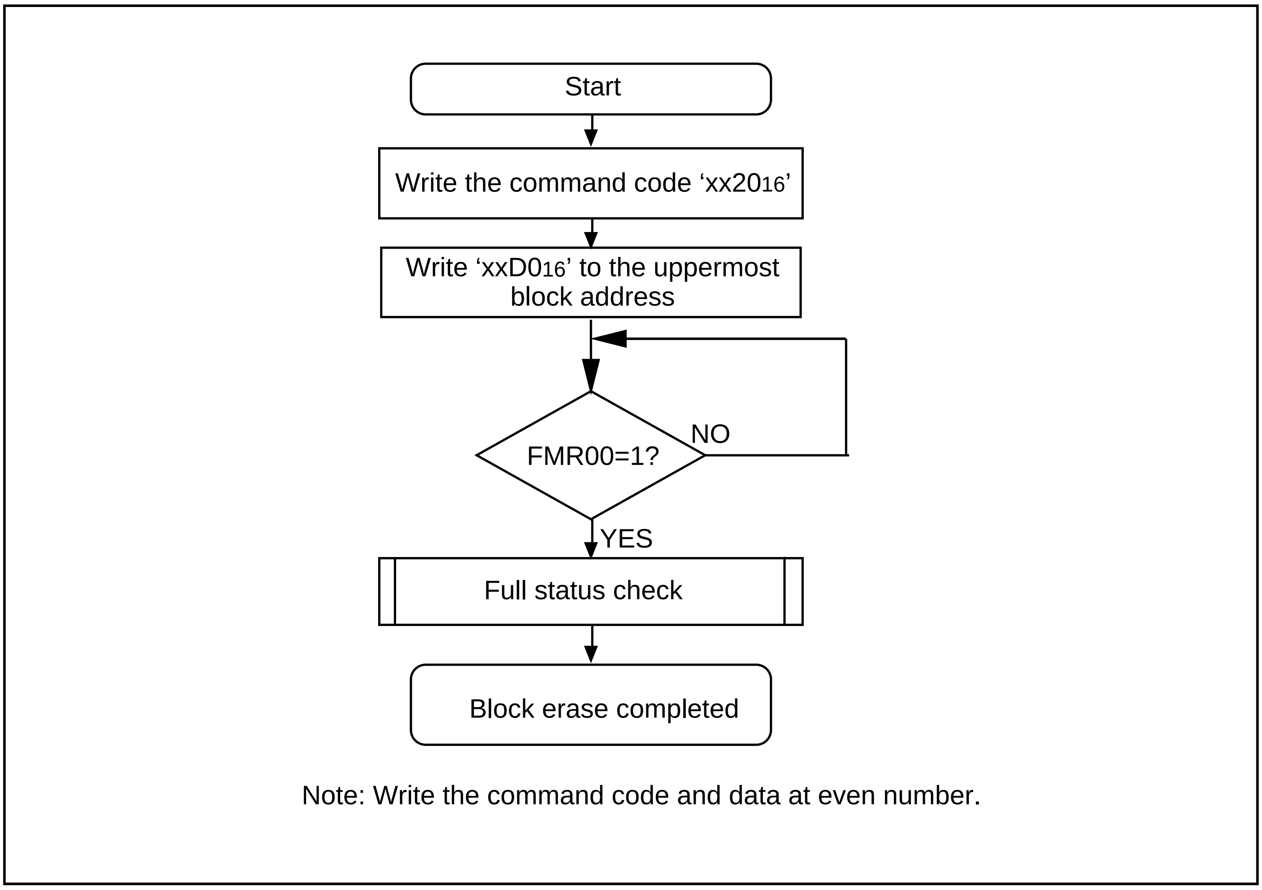
Figure 1.21.6 shows an example of a block erase flowchart.

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.


Start
Write the command code ‘xx2016’
Write ‘xxD016’ to the uppermost block address
FMR00=1?
NO
YES
Full status check
Block erase completed

Note: Write the command code and data at even number.

Figure 1.21.6. 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

- D_0 to D_7: 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 (D_7)	FMR00	Sequencer status	Busy	Ready	1
SR6 (D_6)	——	Reserved	-	-	——
SR5 (D_5)	FMR07	Erase status	Terminated normally	Terminated in error	0
SR4 (D_4)	FMR06	Program status	Terminated normally	Terminated in error	0
SR3 (D_3)	——	Reserved	-	-	——
SR2 (D_2)	——	Reserved	-	-	——
SR1 (D_1)	——	Reserved	-	-	——
SR0 (D_0)	——	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.

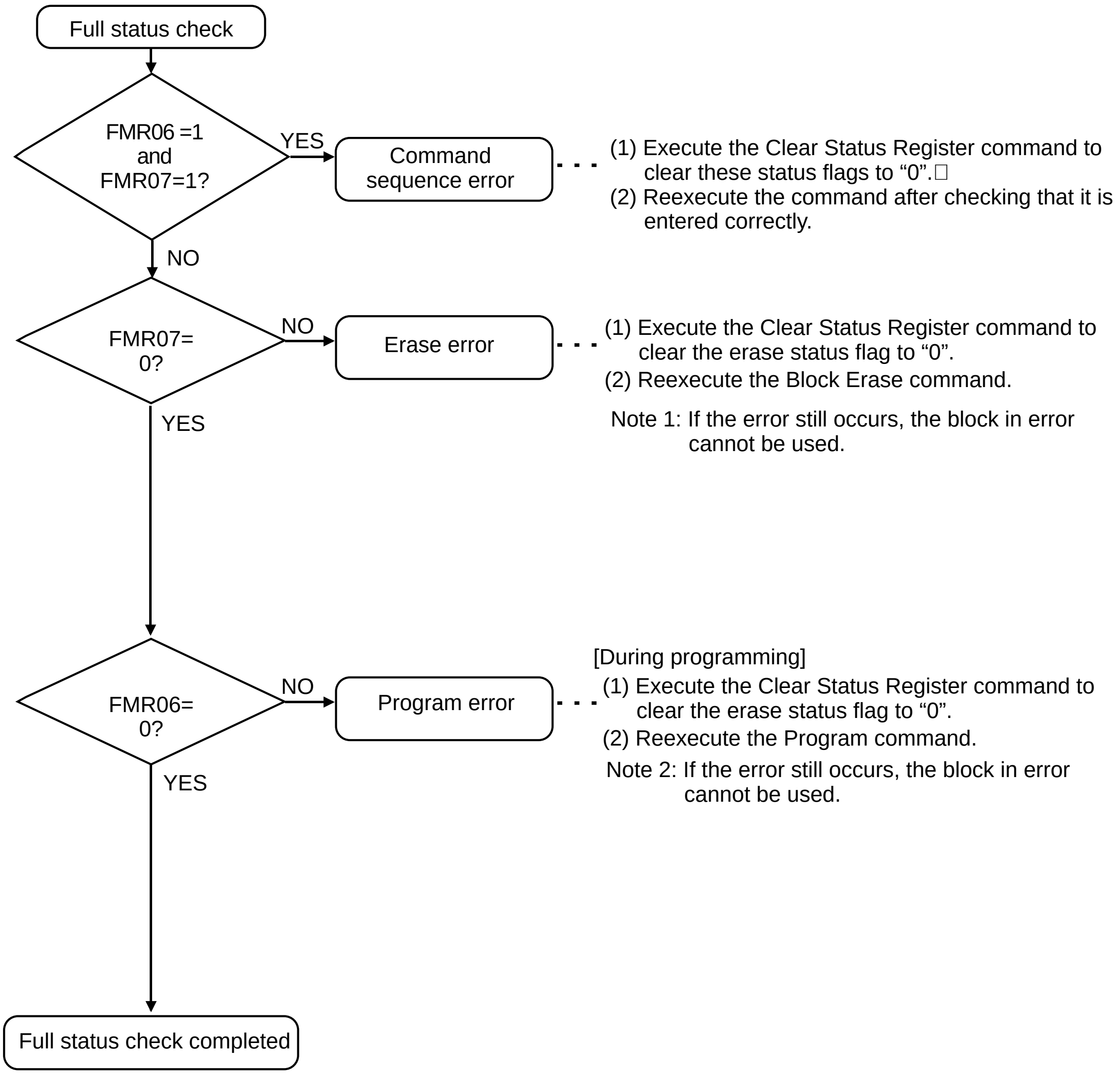
Full Status Check

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 1.21.7 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

FRM00 register (status register) status		Error	Error occurance condition
FMR07 (SR5)	FMR06 (SR4)		
1	1	Command sequence error	• When any command is not written correctly • When invalid data was written other than those that can be written in the second bus cycle of the Block Erase command (i.e., other than 'xxD0$_{16}$' or 'xxFF$_{16}$') (Note 1)
1	0	Erase error	• When the Block Erase was executed but the blocks were not automatically erased correctly
0	1	Program error	• When the Program command was executed but the blocks were not automatically programmed correctly.

Note 1: Writing "xxFF16" in the second bus cycle of these commands places the microcomputer in read array mode, and the command code written in the first bus cycle is nullified.



(1) Execute the Clear Status Register command to clear these status flags to “0”.□
(2) Reexecute the command after checking that it is entered correctly.

(1) Execute the Clear Status Register command to clear the erase status flag to “0”.
(2) Reexecute the Block Erase command.

Note 1: If the error still occurs, the block in error cannot be used.

[During programming]
(1) Execute the Clear Status Register command to clear the erase status flag to “0”.
(2) Reexecute the Program command.

Note 2: If the error still occurs, the block in error cannot be used.

Note 3: If FMR06 or FMR07 = 1, neither the Program nor Block Erase command is accepted. Execute the Clear Status Register command before executing those commands.

Figure 1.21.7. 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 P1$_5$ (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.)

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 asynchronized. Standard serial I/O swiches between mode 1 (clock synchronous) and mode 2 (clock asynchronous) depending on the level of CLK$_1$ pin when the reset is released.

To use standard serial I/O mode 1 (clock synchronous), set the CLK$_1$ pin to “H” level and release the reset. The operation uses the four UART1 pins CLK$_1$, RxD$_1$, TxD$_1$ and RTS$_1$ (BUSY). The CLK$_1$ pin is the transfer clock input pin through which an external transfer clock is input. The TxD$_1$ pin is for CMOS output. The RTS$_1$ (BUSY) pin outputs an “L” level when ready for reception and an “H” level when reception starts. In mode 1, be sure the TxD$_1$ (P6$_7$) pin is at high before reset being deasserted.

To use standard serial I/O mode 2 (clock asynchronous), set the CLK$_1$ pin to “L” level and release the reset. The operation uses the two UART1 pins RxD$_1$ and TxD$_1$.

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 desctiption of the functions to inhibit rewriting flash memory version.)

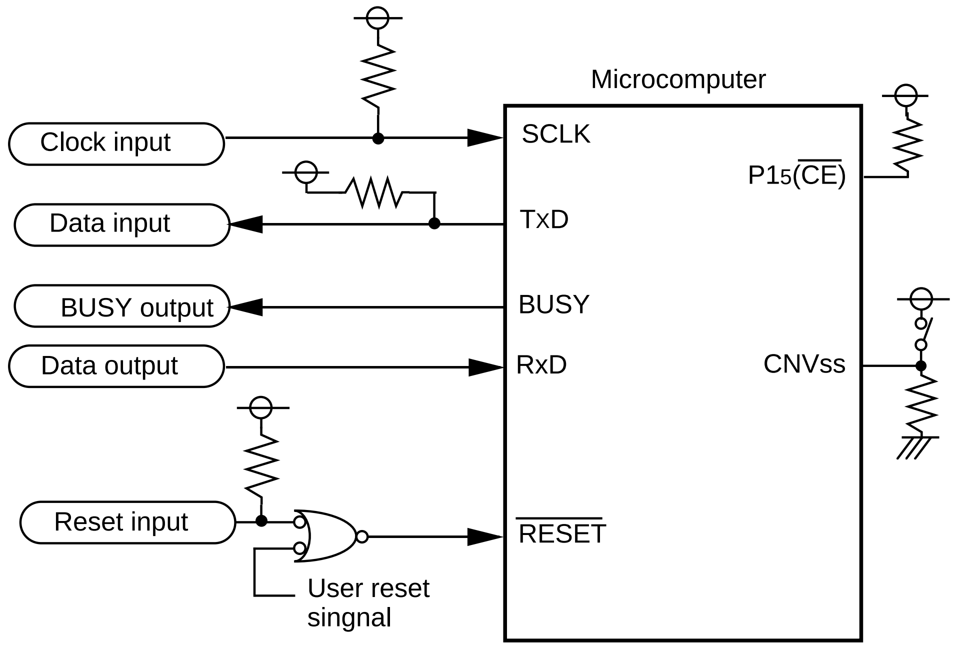
Table 1.22.1. Functional explanation of pin (flash memory standard serial I/O mode)

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.
$\overline{\text{RESET}}$	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.

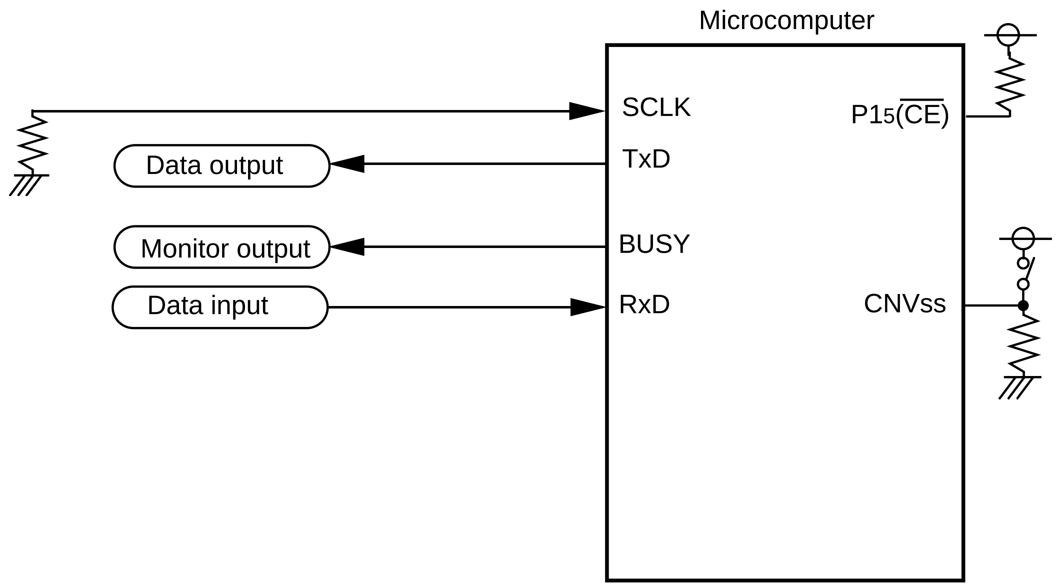
Example of Circuit Application in the Standard Serial I/O Mode

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 writer to handle pins controlled by a serial writer.



(1) Control pins and external circuitry will vary according to programmer.
For more information, see the programmer manual.
(2) In this example, modes are switched between single-chip mode and standard serial input/output mode by controlling the CNVss input with a switch.
(3) If in standard serial input/output mode 1 there is a possibility that the user reset signal will go low during serial input/output mode, break the connection between the user reset signal and RESET pin by using, for example, a jumper switch.

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



(1) In this example, modes are switched between single-chip mode and standard serial input/output mode by controlling the CNVss input with a switch.

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 is 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 is consists of Line coupling circuit, Differential preamplifier, Input filter group and ADC. Line coupling circuit is common to both transmission and reception.

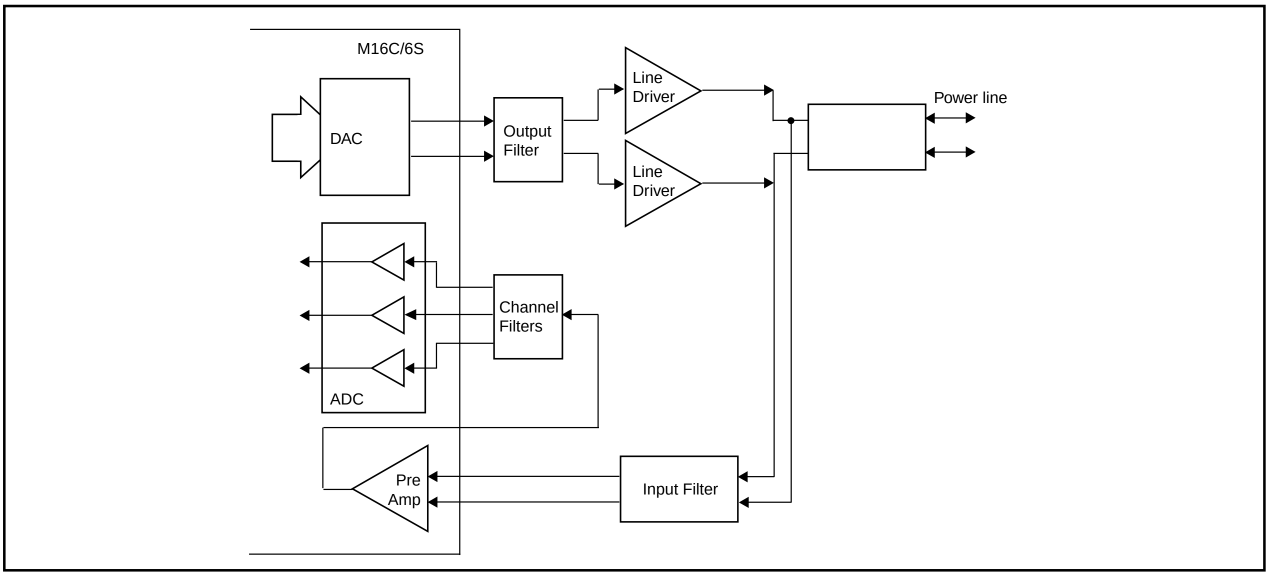


Fig.1.23.1 Analog Front End circuit block diagram

(1)Transmitting circuit

The characteristic required for transmitting course is as follows.

(a)A suitable output signal level is obtained to the load of rating

(b)It has the frequency characteristic to maximum zone width of 400kHz

(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:100k 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

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 Fig. 1.23.2. The typical characteristic is shown in Table 1.23.1.

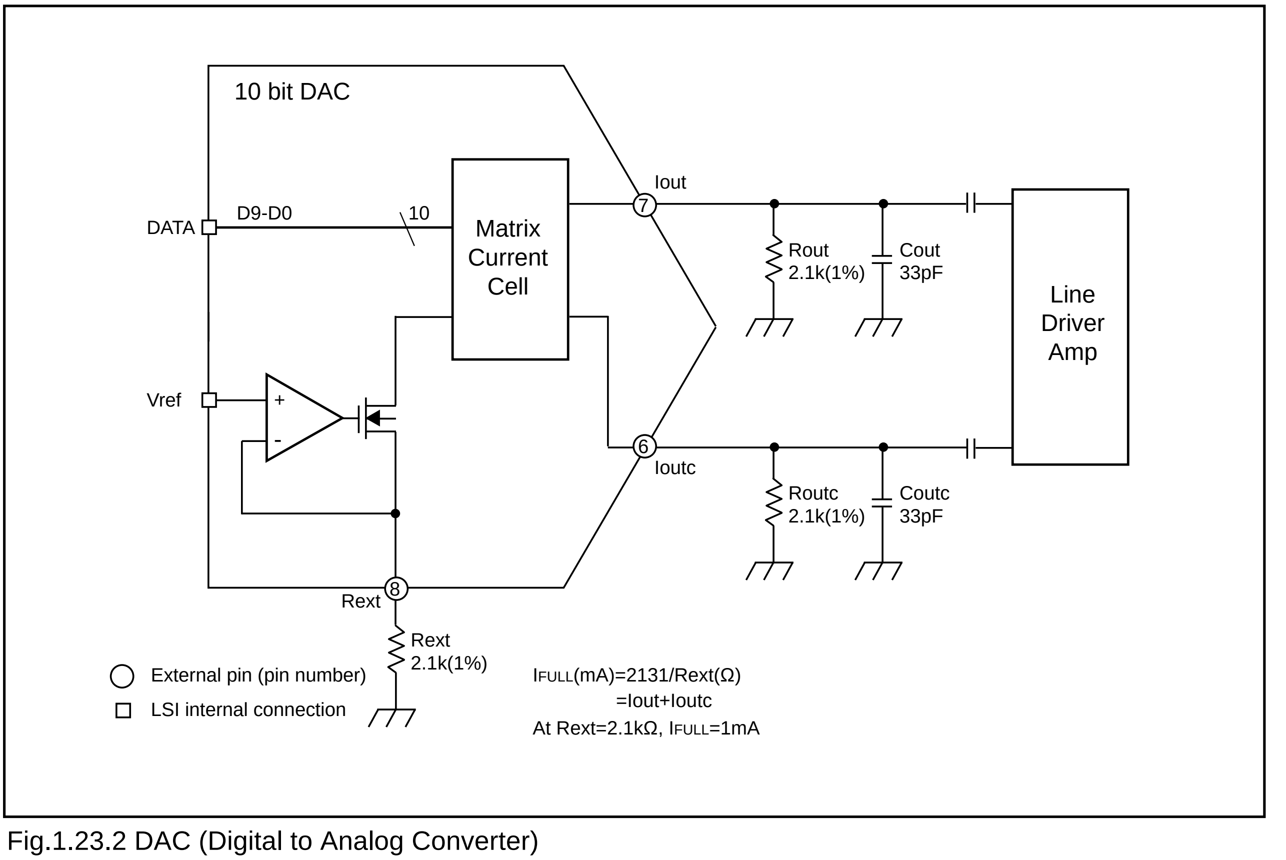


Fig.1.23.2 DAC (Digital to Analog Converter)

Table.1.23.1 DAC Electrical Characteristics (Vdd=3.3±0.3V, Ta=-20°C to 85°C unless otherwise specified)

Parameter	Symbol	Condition	Electrical Characteristics			Unit
			Min	Typ	Max	
Standard voltage	Vref	Rext=2.1kΩ		0.3		V
Standard current	Iref	Rext=2.1kΩ		0.14		mA
External standard register	Rext			2.1		kΩ
DAC output register	Rout/Routc	Rext=2.1kΩ			2.0	kΩ
Full-scale current	IFULL	Rext=2.1kΩ	0.9	1	1.1	mA
Maximum output voltage	Voutmax	Rout/Routc=2.1kΩ			Vdd-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 (Iout/Ioutc) 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/Rext(mA)

The unit of Rext is Ω

Since about 1mA of an IFULL value is the optimal value, Rext is 2.1kΩ.

DAC is total and Iout/Ioutc changes between 0 to IFULL at 1024 steps by the full range for a change width of 10 bits.

The value of Iout/Ioutc 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)

Iout=IFULL/1023*d

Ioutc=IFULL/1023*(1023-d)

(b)About DAC load resistance

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

Vout=Rout X Iout

Voutc=Routc X Ioutc

Since the recommendation value of Vout/Voutc at the time of a non-signal is about 1V, in Rext=2.1kΩ, Rout/Routc serves as 2.1kΩ.

The capacitor (Cout/Coutc) 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 Fig. 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.

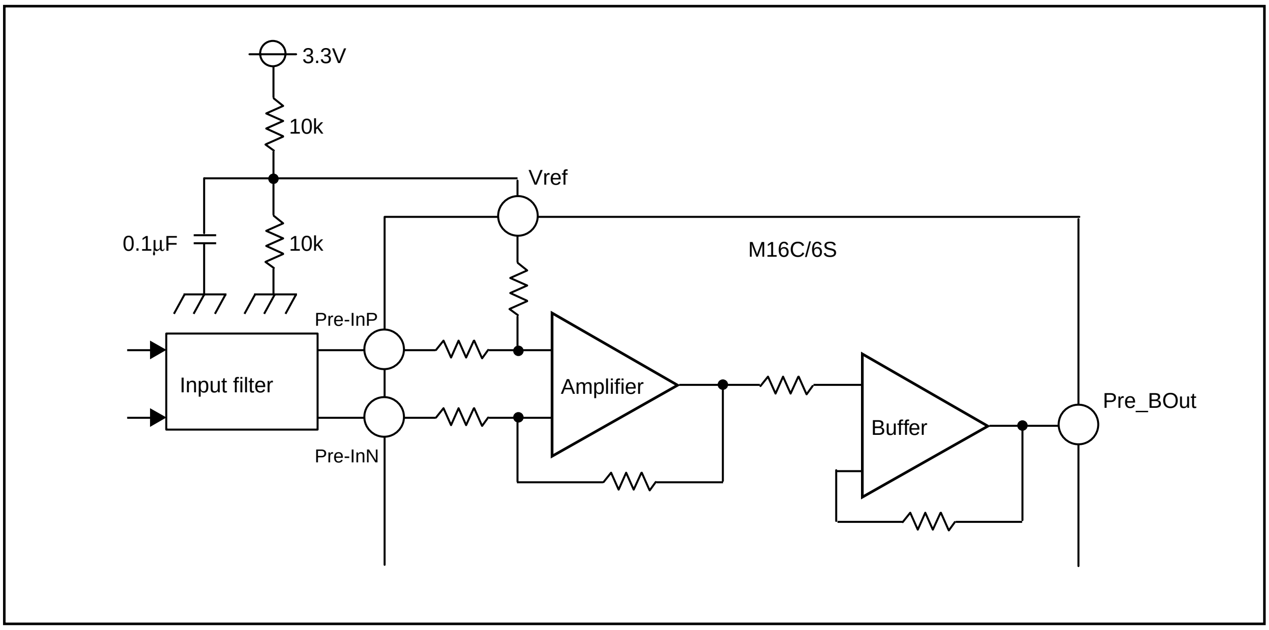


Fig. 1.23.3 Consists of Preamp circuit

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.

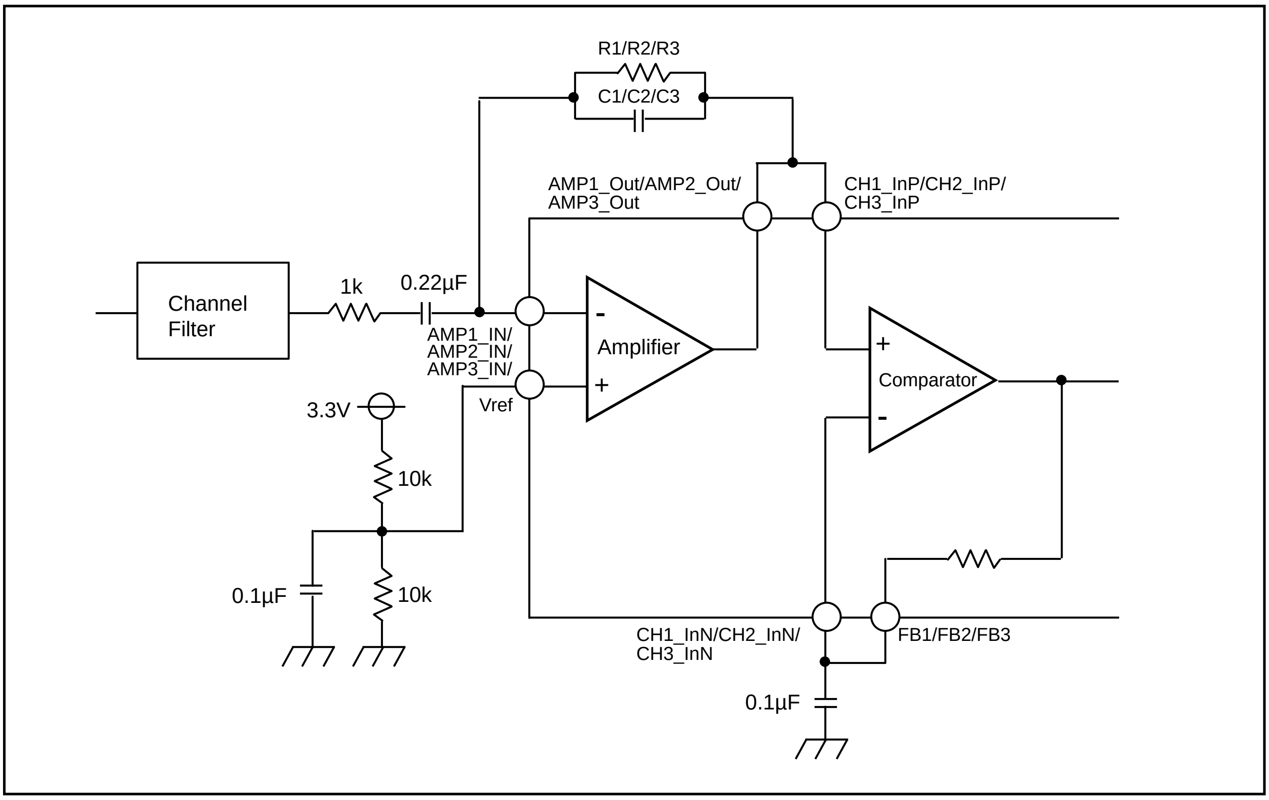


Fig.1.23.4 ADC block diagram

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

REVISION HISTORY	M16C/6S Group

Rev. No.	Revision Description	Rev. date
1.00	First Edition of PDF File	Jun 25, 2003
2.00	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.	Aug 18, 2003

Rev. No.	Revision Description	Rev. date
2.00	P121 T1.16.4 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.	Aug 18, 2003

Rev. No.	Revision Description	Rev. date
2.01	P3 L1 to L2 are changed. F1.1.1 is changed. P7 T1.1.3 is changed. P22 L3 to L4 are changed. P23 (3) Setting PLC Mode is changed. T1.6.4 and F1.6.1 and F1.6.2 are changed. P141 F1.18.4 is changed. P143 F1.18.6 is changed. P144 F1.18.7 is changed. P178 Standard Serial I/O Mode is changed.	Oct 27, 2003
3.00	P1 Table of Contents is changed. P2 T1.1.1 is changed. P4 T1.1.2 is changed. P5 F1.1.2 is changed. P6 F1.1.3 is changed. P7 T1.1.3 is changed. P9 F1.2.1 is changed. P22 Text is changed. T1.6.1 is delated. P23 Text is changed. T1.6.4 and F1.6.2 are delated. Table name of T1.6.3 is added. P26 T1.7.1 is changed. P27 F1.7.1 is changed. P32 Text is changed. F1.7.6 is changed. P33 Text is changed. P36 Text is changed. P37 T1.7.4 is changed. P38 Text is changed. P44 T1.7.7, F1.7.9 and text are delated. P45 Text is changed. P46 F1.9.1 is changed. P48 Text is changed (NMI Interrupt is delated.) P49 T1.9.1 is changed. P55 T1.9.5 is changed. P58 F.1.9.8 is changed. P59 F.1.9.9 is changed. BetweenP60 to P61 Text is delated (NMI Interrupt is delated.) P63 Text is delated (NMI Interrupt is delated.) P79 F.1.12.4 is changed. P137 Text is changed. P140 F.1.18.3 is changed. P147 T.1.18.1 is changed. P148 F.1.18.10 is changed. P149 T.1.19.1 is changed. P150 T.1.19.2 is changed. Deletion of a voltage desplay of a header. P151 Deletion of a voltage desplay of a header. P152 Deletion of a voltage desplay of a header.	Jul 22, 2004

Rev. No.	Revision Description	Rev. date
3.00	P153 T.1.19.6 is changed. Deletion of a voltage desplay of a header. P154 T.1.19.7 is changed. Deletion of a voltage desplay of a header. P155 Deletion of a voltage desplay of a header. P156 Deletion of a voltage desplay 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.	Jul 22, 2004

RenesasTechnology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.

RENESAS SALES OFFICES

http://www.renesas.com

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, United Kingdom
Tel: <44> (1628) 585 100, Fax: <44> (1628) 585 900

Renesas Technology Europe GmbH
Dornacher Str. 3, D-85622 Feldkirchen, Germany
Tel: <49> (89) 380 70 0, Fax: <49> (89) 929 30 11

Renesas Technology Hong Kong Ltd.
7/F., North Tower, World Finance Centre, Harbour City, Canton Road, Hong Kong
Tel: <852> 2265-6688, Fax: <852> 2375-6836

Renesas Technology Taiwan Co., Ltd.
FL 10, #99, Fu-Hsing N. Rd., Taipei, Taiwan
Tel: <886> (2) 2715-2888, Fax: <886> (2) 2713-2999

Renesas Technology (Shanghai) Co., Ltd.
26/F., Ruijin Building, No.205 Maoming Road (S), Shanghai 200020, China
Tel: <86> (21) 6472-1001, Fax: <86> (21) 6415-2952

Renesas Technology Singapore Pte. Ltd.
1, Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: <65> 6213-0200, Fax: <65> 6278-8001

Colophon .1.0